В.І.Шекета, В.М. Юрчишин , Л.М.Гобир

ОСНОВИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

НАВЧАЛЬНИЙ ПОСІБНИК

Івано-Франківськ
2020

Шекета В.І., Юрчишин В.М., Гобир Л.М. Основи програмної інженерії: навчальний посібник - Івано-Франківськ: Факел, 2020.- 161 с.

В навчальному посібнику відображено основні методи і засоби програмної інженерії (software engineering), що дозволяють створювати ефективне програмне забезпечення. Цей курс містить поглиблені відомості з основних розділів програмної інженерії та соціально-професійних питань програмування згідно освітньопрофесійної програми..

Так, в навяальному посібнику розглядаються увесь спектр процесів, що ведуть до створення програмного забезпечення: від початкової розробки системних вимог і далі через проектування, безпосереднє програмування і атестацію, до модернізації програмних систем.

Крім того, навчальний посібник знайомить студентів з соціальними та професіональними питаннями програмування, зокрема з професійною та етичною відповідальністю фахівця з інженерії прорамного абезпечення.

При побудові курсу враховані рекомендації Асоціації з обчислювальної техніки і Інституту інженерів з електротехніки і електроніки (ACM та IEEE), висловлені в документі Computing Curricula 2001 (навчальний план для комп'ютернх дисциплін) щодо змісту базового набору дисциплін, які утворюють основу комп'ютерних наук, зокрема до дисципліни основи програмної інженерії.

В кінці кожного розділу є контрольні питання самоконтролю для закріплення вивченого матеріалу.

Адресовано студентам які навчаються за напрямом підготовки 121, „ Інженерія програмного забезпечення ".

УДК 681.3201 Петрів Л.О.
МВ 02070855 – 1794 - ІФНТУНГ,2020

УДК 681.3
МВ 02070855 – 1794 -201

Рецензент:

Горбійчук М.І. локтор технічних наук, професор кафедри комп'ютерних систем та мереж ІФНТУНГ

Рекомендовано методичною радою університету

(протокол № від р.

Шекета В.І., Юрчишин В.М. , Гобир Л.М.

Основи програмної Інженерії: навчальний посібник.- Івано-Франківськ: Факел, 2020.- 161.с.

В навчальному посібнику відображено основні методи і засоби програмної інженерії (software engineering), що дозволяють створювати ефективне програмне забезпечення. Цей курс містить поглиблені відомості з основних розділів програмної інженерії та соціально-професійних питань програмування згідно освітрьо професійної програми .

***Мета*:** познайомити бакалаврів випуску та збереженням проекту у VBA, виділити основні етапи розробки проект, проектування форми, виховувати стійкий інтерес до розвитку інформаційних технологій.

При побудові курсу враховані рекомендації Асоціації з обчислювальної техніки і Інституту інженерів з електротехніки і електроніки (ACM та IEEE), висловлені в документі Computing Curricula 2001 (навчальний план для комп'ютерних дисциплін) щодо змісту базового набору дисциплін, які утворюють основу комп'ютерних наук, зокрема до дисципліни програмна інженерія.

УДК 681.3201 Петрів Л.О.
МВ 02070855 – 1794 - ІФНТУНГ,2012

Відповідальний за випуск,
Завідувач кафедри інженерії програмного забезпечення Шекета В.І.
Член експертно-рецензійної комісії університету М.І. Гордійчук
Нормо контролер О.О.Петрів
Інженер 1 категорії НТБ Л.Я.Тимчишин

ЗМІСТ

.

Розділ 1. Місце програмної інженерії в галузі інформаційних технологій.

Сьогодні ми є свідками і учасниками еволюції постіндустріального суспільства до суспільства, яке називають "інформаційним". Поняття "інформаційні технології" (IT) з'явилося не так давно, їхніми прабатьками були електронні обчислювальні машини, створені в середині XX століття. Вони, у свою чергу, породили "Computer Science" - науку про комп'ютери та "Informatigue" -інформатику. Перше поняття виникло в США, друге - в Європі. По суті вони означали одне й те ж - обширну галузь теоретичних і прикладних знань, пов'язаних з отриманням, збереженням, обробкою, передаванням і використанням інформації. Саме вони і стали базою для розвитку та становлення IT. Перші кроки на шляху до сучасних інформаційних технологій в Україні були зроблені в першій половині XX століття. На жаль, вони до останнього часу залишалися "білими плямами" в історії розвитку цього важливого напрямку науки і техніки. "Білі плями" - це події, винаходи, відкриття, які протягом довгого часу в силу певних причин залишаються невідомими або забутими, їх відкриття іноді значно змінює усталені уявлення про історію тієї чи іншої науки або напрямку в техніці та роль учених в їх розвитку. Світова історія IT не є виключенням, у тому числі й та її частина, що стосується України. Завдяки розвитку інформаційно-комунікаційних технологій технологічна складова суспільного розвитку сьогодні настільки вагома, що на очах одного покоління відбуваються фундаментальні зміни в соціальній сфері, економіці, розвитку інститутів демократії. В кінці 1990-х років стало ясно, що область знань, пов'язана з інформаційними технологіями, дуже сильно розрослася і її важко, якщо взагалі можливо, повністю освітити в рамках одного університетського курсу. У зв'язку з цим було ухвалено рішення про її розділення на чотири основні дисципліни — інформатика (informatigue), коипютерні науки (computer science), програмна інженерія (software engineering), проектування апаратних платформ (hardware engineering) і інформаційні системи (information systems) В 2004 році було виданню і розіслано по університетах країн СНД російської версії документа Computing Curricula 2001: Computer Science. Оскільки російський переклад російською мовою документа «Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software

Engineering», в якому зібраний усесвітній досвід викладання програмної інженерії в університетах і коледжах. Історія проекту Computing Curricula, в рамках якого був випущений даний документ, веде свій відлік з 1968 року, коли була опублікована перша версія рекомендацій по викладанню інформатики в університетах. З тих пір ці рекомендації оновлювалися приблизно раз в десять років сумісним комітетом за освітою під егідою професійних асоціацій ACM (Association for Computing Machinery) і IEEE (IEEE Computer Society Educational Activities Board) Computer Society. Матеріал, що викладається, по програмній інженерії (**Software Engineering Education Knowledge,** також використовуватиметься абревіатура **SEEK**) повинен знати кожен випускник вузу, що спеціалізується інженерї програмного забезпечення. Як це не дивно, однією з основних труднощів проекту став переклад слова **«computing»,** що позначає узагальнену область знань, в яку входять інформатика, програмна інженерія, проектування апаратних платформ і інші дисципліни, так або інакше пов'язані з інформаційними технологіями. Це слово можна приблизно перевести як **«обчислювальні науки»** або **«обчислювальна техніка»,** але, на жаль, обидва цих терміни дуже вузькі, оскільки **computing** включає і науку, і техніку, і інженерні дисципліни. Від варіанту **«інформаційні технології»** теж довелося відмовитися, оскільки інформаційна технологія є лише однією з самостійних дисциплін в рамках ширшій області знань під назвою **computing.** У зв'язку з цим ми ухвалили непросте рішення про використання при перекладі англійського слова, тобто **«компьютинг».** Цей переклад не ідеальний, але дозволяє уникнути подвійного тлумачення термінів і тому поступово входить у вживання серед спеціалісті інформаційних технологій. Програмне забезпечення відіграє важливу роль практично у всіх аспектах повсякденного життя: державному управлінні, банківській справі і фінансах, освіті, транспорті, індустрії розваг, медицині, сільському господарстві, юриспруденції, нафтогазовій галузі тощо. Кількість, розміри і області застосування комп'ютерних програм різко збільшилися. В результаті сотні мільярдів доларів витрачаються на розробку програмного забезпечення, і від ефективності цих програм залежать заробітки і навіть життя більшості людей. Програмні продукти допомагають працювати ефективніше і продуктивно. Вони допомагають у вирішенні завдань і надають середовище для роботи і розваг. Проте, не дивлячись на всі ці успіхи, досягнення адекватної вартості, термінів розробки і якості програмних продуктів, існує безліч причин виникнення проблем, включаючи наступні:

1.Програмні продукти відносяться до найскладніших систем, які створюються людиною, і програмне забезпечення в основі своїй поряд з позитивними властивостями мають і негативні, які утрудняють роботу (наприклад, складність, ненаочність і змінність);

2.Методи і процеси програмування, які ефективно працюють для однієї людини або для невеликої команди при розробці програм відносно невеликих розмірів, погано масштабуються для розробки крупних і складних систем (тобто систем, що складаються з мільйонів рядків кодів і вимагають декількох років роботи для значної кількості людей при розробці програмного забезпечення);

3. Швидкість зміни інформаційних технологій створює потребу в нових і еволюційних програмних продуктах. Очікування користувача і конкурентна боротьба, що виникають в таких умовах, істотно утрудняють можливість випускати якісне програмне забезпечення в прийнятні терміни.

Термін ***«програмна інженерія»***, сьогодні широко використовується в індустрії, державних і навчальних установах — сотні тисяч фахівців іменують себе «програмними інженерами» (software engineers), термін «програмна інженерія» фігурує в назвах безлічі публікацій, груп, організацій і професійних конференцій, існує безліч навсальних курсів і програм навчання програмної інженерії. Проте як і раніше існують розбіжності і різні думки за значенням даного терміну. Приведені нижче визначення дають декілька різних уявлень про значення і природу програмної інженерії. Проте їм всім властива одна загальна межа: ***всі вони сходяться в тому, що програмна інженерія — щось більше, ніж просто написання програмного коду (coding) і включає аспекти якості, управління і економіки, а також знання і застосування на практиці цих принципів і дисциплін.***

Протягом багатьох років давалися різні визначення дисципліни програмної інженерії. Відзначимо наступні визначення:

1.«Встановлення і використання правильних інженерних принципів (методів) для економічного отримання надійного і працюючого на реальних машинах програмного забезпечення» ;

2. «Програмна інженерія є такою формою інженерії, як застосовує принципи інформатики **(computer science)** і математики для отримання рентабельних рішень в області програмного забезпечення» ;

3. «Застосування систематичного, дисциплінованого, такого, що піддається кількісному визначенню підходу до розробки, експлуатації і супроводу програмного забезпечення» .

Кожне з цих визначень містить окремі аспекти, що вплинули на загальне розуміння програмної інженерії, Одне з найбільш важливих спостережень полягає в тому, що програмна інженерія грунтується на інформатиці і математиці. Проте, враховуючи інженерні традиції можна стверджувати , що програмна інженерія виходить за рамки цього технічного базису і використовує результати ширшого діапазону дисциплін. Дані визначення явно формулюють, що програмна інженерія присвячена систематичним, керованим і ефективним методам створення високоякісного програмного забезпечення. Тому особлива увага приділяється аналізу і оцінці, специфікації, проектуванню і еволюції програмного забезпечення. Крім того, в рамки даної дисципліни потрапляють питання, пов'язані з управлінням і якістю, новизною і творчістю, стандартами, індивідуальними навиками і командною роботою, а також професійною діяльністю, які відіграють життєво важливу роль в програмній інженерії.

Помилкою, що часто зустрічається, щодо програмної інженерії є уявлення про те, що вона переважно пов'язана з діяльністю, орієнтованою на процеси (наприклад, управління вимогами, проектування, забезпечення якості, вдосконалення процесів і управління проектом). З цієї точки зору для досягнення компетентності в області програмної інженерії досить мати інженерну підготовку, представляти у загальних рисах процес розробки програмного забезпечення і володіти мінімальними знаннями про компьютер, включаючи навики використання одного або декількох мов програмування. Проте на практиці таких знань абсолютно недостатньо, а помилка, що приводить до подібної точки зору, заснована на неповному уявленні про природу і проблеми програмної інженерії. Професійний програміст не может бути обмежений одією алгоритмічною мово. Він повинен мати можливість вибирати мову програмування найбіш "зручною" і максимально задовольняти умовам поставленої задачі.

В ході розвитку обчислювальної техніки історично склалося, що фахівці з комп'ютерних наук писали програмне забезпечення **(Soft),** а спеціалісти в обдасті електроніки **(Hard**) проводили апаратне забезпечення, на якому працювали програми. У міру збільшення розмірів, складності і критичності програмного забезпечення зростала і потреба у відповідності програмного забезпечення початковим вимогам. На початок 1970-х років стало очевидне, що для грамотної розробки програмного забезпечення недостатньо простого застосування основних принципів інформатики. Потрібні аналітичні і описові засоби, розроблені фахівцями

з інформатики, і ретельність, з якою інженерні дисципліни добиваються надійності і достовірності програмних продуктів, що створюються .

Таким чином, програмна інженерія якісно відрізняється від інших інженерних дисциплін нематеріальністю програмного забезпечення і дискретною природою його функціонування. Програмна інженерія прагне інтегрувати принципи математики і інформатики з інженерними підходами, розробленими для виробництва відчутних матеріальних програмних продуктів. Грунтуючись на математиці і комп'ютинзі, програмна інженерія займається розробкою систематичних моделей і надійних методів виробництва високоякісного програмного забезпечення, і даний підхід розповсюджується на всі рівні - від теорії і принципів до реальної практики створення програмного забезпечення, яка краще всього помітна стороннім спостерігачам. Хоча і не передбачається, що кожен фахівець з розробки та тестування програмного забезпечення володіє глибокими знаннями у всіх аспектах комп'ютинга, але загальне розуміння їх застосування і кваліфікації в певних предметних областях є абсолютно необхідними. На дослідження і практику в області програмної інженерії впливають як інші дисципліни, так і її встановлення як самостійної інженерної дисципліна. Отже, дисципліна програмної інженерії може розглядатися як інженерна область, що має тісніші зв'язки з іншими базовими дисциплінами.

Питання для самоконтролю.

1. Сформулюйте визначення терміну **..інформаційні технології"**
2. Що означає абревіатура SEEK
3. Перелічите проблеми, які виникають при розробці програмних продуктів,.
4. Що таке програмна інженерія ?
5. Що означає термін ..комп'ютінг" ?

Розділ 2. Ознайомлення з освітньо-професійною програмою та освітньо-кваліфікаційною характеристикою з програмної інженерії

Освітньо-професійна програма **(ОПП)** є галузевим нормативним документом, у якому визначається нормативний термін та зміст навчання, нормативні форми державної атестації, встановлюються вимоги до змісту, обсягу й рівня освіти та професійної підготовки фахівця з напряму підготовки 201 «Іженерія програмного забезпечення».

Цей стандарт установлює:

1) нормативну частину змісту навчання у навчальних об’єктах, засвоєння яких забезпечує формування системи умінь відповідно до вимог освітньо-кваліфікаційної характеристики;

2(рекомендований перелік навчальних дисциплін і практик;

3) нормативний термін навчання за денною формою навчання;

4) нормативні форми державної атестації.

Стандарт є обов’язковим для вищих навчальних закладів, що готують фахівців даного профілю. Підприємства, установи, організації повинні забезпечити необхідні умови для використання фахівців відповідно до здобутих ними у вищому навчальному закладі кваліфікації та спеціальності згідно з чинним законодавством.

Стандарт придатний для цілей ліцензування та акредитації вищих навчальних закладів, атестації осіб, які закінчили навчання у вищих навчальних закладах, та сертифікації фахівців.

У цьому стандарті використано такі терміни та відповідні визначення:

Архітектура програмного забезпечення (*Software Architecture*) – опис підсистем і компонентів програмної системи, а також зв’язків між ними. Архітектура визначає внутрішню структуру програмної системи, що проектується, задаючи спосіб організації або конструювання системи.

Проектування програмного забезпечення *(Software Design)* – процес визначення архітектури, компонентів, інтерфейсів та інших характеристик програмної системи чи її складових. Проектування програмного забезпечення можна розглядати як діяльність, результатом якої може бути:

- архітектурний або високорівневий дизайн (*Architectural Design, Top-Level Design*) – опис високорівневої структури та організації компонентів системи;
- деталізована архітектура (*Software Detailed Design)* – опис кожного компоненту в тому обсязі, що є необхідним для конструювання.

Конструювання програмного забезпечення (*Software Construction*) – процес створення працюючої функціональної програмної системи за допомогою кодування, верифікації, модульного тестування, інтеграційного тестування та налагодження .

Мови конструювання (*Construction Languages*) включають всі форми комунікацій, за допомогою яких людина може задати рішення проблеми, де виконало на комп'ютері.

Типи мов конструювання:

1)конфігураційна мова (*Configuration Language*), що дозволяє задавати параметри виконання програмної системи;

2)інструментальна мова (*Toolkit Language*) – мова конструювання з елементів, що використовуються повторно; звичайно будується як сценарна мова (*script*), що виконана у відповідному середовищі.

3)мова програмування (*Programming Language*) – найгнучкіший тип мов конструювання .

Освітньо-професійна програма передбачає такі цикли підготовки:

1) цикл гуманітарної та соціально-економічної підготовки і цикл природничо-наукової підготовки, що забезпечують певний освітній рівень;
2) цикл професійної (професійно-орієнтованої) та практичної підготовки, що разом із попередніми циклами забезпечує певний освітньо-кваліфікаційний рівень.

Розподіл змісту освітньо-професійної програми підготовки фахівця та навчальний час за нормативною та варіативною частинами програми підготовки, навчальний час за циклами підготовки, кількість навчальних годин/кредитів вивчення кожної з рекомендованих навчальних дисциплін і практик нормативної частини програми підготовки . У викладанні навчальних дисциплін нормативної частини змісту навчання приймають участь доктори наук, професори, кандидати наук, доценти, які мають певний стаж практичної, наукової та педагогічної роботи. Доцільно, щоб викладачі, які забезпечують дисципліни циклу професійної та практичної підготовки, в переважній більшості мали наукові ступені в галузі технічних або фізико-математичних наук.

Професорсько-викладацький склад, який здійснює навчальний процес, повинен періодично та своєчасно проходити стажування. Доцільно, щоб викладачі, які забезпечують викладання дисциплін циклу професійної та практичної підготовки, проходили стажування в провідних українських ІТ-компаніях.

Кафедри, які беруть участь у реалізації освітньо-професійної програми підготовки бакалаврів з напряму підготовки «Інженерія програмного зпюезпечення», складають та видають навчальні посібники, конспекти лекцій та методичні розробки щодо вивчення навчальних дисциплін.

Тематика наукових досліджень, які проводять кафедри, за напрямом і змістом відповідають дисциплінам, що викладаються; результати впроваджуються у навчальний процес.

Освітньо-кваліфікаційна характеристика(ОКХ) випускників вищого навчального закладу є галузевим нормативним документом, в

якому узагальнюється зміст вищої освіти, тобто відображаються цілі вищої освіти та професійної підготовки, визначається місце фахівця в структурі галузей економіки держави і вимоги до його компетентності, інших соціально важливих властивостей та якостей.

Цей стандарт є складовою галузевих стандартів вищої освіти, в якій узагальнюються вимоги з боку держави, світового співтовариства та споживачів випускників до змісту вищої освіти. ОКХ відображає соціальне замовлення на підготовку фахівця з урахуванням аналізу професійної діяльності та вимог до змісту вищої освіти з боку держави та окремих замовників фахівців.

ОКХ установлює галузеві кваліфікаційні вимоги до соціально-виробничої діяльності випускників вищого навчального закладу з напряму підготовки 121 «Інженерія програмного забезпечення» і державні вимоги до властивостей та якостей особи, яка здобула певний освітній рівень відповідного фахового спрямування.

У цьому стандарті використано такі терміни та відповідні визначення:

Аналіз вимог (*Requirements Analysis*) – трансформація інформації, отриманої від користувачів (та інших зацікавлених осіб) в чітко та однозначно визначені програмні вимоги, що передаються інженерам для реалізації в програмному коді. Аналіз вимог включає:

1)виявлення і розв’язання конфліктів між вимогами;

2)визначення меж задачі, що вирішується створюваним програмним забезпеченням; в загальному випадку – визначення меж (*Scope*) і змісту програмного проекту;

3)деталізацію системних вимог для встановлення програмних вимог .

Верифікація (*Verification*) та **атестація** *(Validation)* – упорядкований підхід щодо оцінювання програмних продуктів, який застосовується протягом усього життєвого циклу. Зусилля, прикладені в рамках робіт з верифікації та атестації, спрямовані на забезпечення якості як невід’ємної характеристики програмного забезпечення та задоволення потреб користувачів. Верифікація – спроба забезпечити *правильну розробку продукту*, в тому значенні, що одержуваний в рамках відповідної діяльності продукт відповідає специфікаціям, заданим в процесі попередньої діяльності. Атестація – спроба забезпечити *створення правильного продукту* з точки зору досягнення поставленої мети

Вимога *(Requirement)* – умова або можливість, що визначена користувачем для вирішення проблеми або досягнення мети, та, якій повинна відповідати або якою повинна володіти система чи її компонент, щоб задовольняти умови контракту, стандарту, специфікації або іншого

формально репрезентованого документа .

Вимоги користувача *(User Requirements)* описують цілі/задачі користувачів системи, які повинні досягатися/виконуватися користувачами за допомогою створюваної програмної системи .

Потреби *(needs)* – відображають проблеми бізнесу, персоналій або процесу, що повинні співвідноситися з використовуванням або придбанням системи .

Програмна інженерія *(Software Engineering)* – дисципліна, спрямована на розробку й супроводження програмного забезпечення систем, що функціонують надійно та ефективно, можуть вдосконалюватися й еволюціонувати, та відповідають вимогам, визначеним замовником.

Програмне забезпечення (*Software*) – комп'ютерні програми, процедури, а також документація й дані, що з ними асоційовані, які стосуються функціонування комп'ютерної системи.

Програмні вимоги (*Software Requirements*) – див. специфікація програмних вимог.

Програмний продукт (*Software Product)* – множина комп'ютерних програм, процедур, а також асоційованих з ними даних та документації (термін об'єднує проміжні продукти - робочі продукти, та продукти, які призначені для розробників і фахівців, які виконують розроблення і супровід).

Системні вимоги *(System Requirements)* – див. специфікації системних вимог.

Специфікація *(Specification*) – документ, що в закінченій, точній і перевіреній формі описує вимоги, проект, поведінку або інші характеристики компоненту або системи, а також процедури, спрямовані на визначення того, чи задовольняються описані характеристики . Для опису комплексних проектів (в частині вимог) використовують три основні специфікації:

- визначення системи (*System Definition*), або специфікація вимог користувачів (*User Requirements Specification*)
- системних вимог (*System Requirements*);
- програмних вимог (*Software Requirements*).

Специфікація вимог користувачів (*User Requirements Specification*) або **концепція** (*concept <of operation>*) визначає високорівневі вимоги, часто – стратегічні цілі, для досягнення яких створюється програмна

система. Принциповим моментом є те, що такий документ описує вимоги до системи з позицій прикладної галузі – домену .

Специфікація системних вимог (*System Requirements)* – описує програмну систему в контексті системної інженерії. Зокрема, високорівневі вимоги до програмного забезпечення, що містить кілька або багато взаємозв'язаних підсистем і застосувань. При цьому, система може бути як цілком програмною, так і містити програмні та апаратні компоненти. В загальному випадку, до складу системи може належати персонал, який виконує певні функції системи, наприклад, авторизацію виконання певних операцій з використанням програмно-апаратних підсистем.

Специфікація програмних вимог (*Software Requirements Specification – SRS*) встановлює основні угоди між користувачами (замовниками) і розробниками (виконавцями) відносно того, що робитиме система і чого від неї не варто чекати. Цей документ може включати процедури перевірки створеного програмного забезпечення на відповідність вимогам, що пред'являються (аж до планів тестування), описи характеристик стосовно якості та методів його оцінювання, питань безпеки тощо. Часто програмні вимоги описуються звичайною мовою. В той же час, існують напівформальні та формальні методи та підходи, що використовуються для специфікації програмних вимог. У будь-якому випадку, завдання полягає в тому, щоб програмні вимоги були ясні, зв'язки між ними прозорі, а зміст специфікації не допускав різночитань і інтерпретацій, через які програмний продукт не буде відповідати потребам зацікавлених осіб .

Тестування (*Software Testing*) – діяльність, що виконується для оцінювання та поліпшення якості програмного забезпечення. Ця діяльність , у загальному випадку, базується на виявленні дефектів і проблем програмного забезпечення .

Управління вимогами *(Requirements Management)*– діяльність, виконання якої забезпечує опис вимог, відстежування їх змін, перевірки на несперечливість і на порушення наперед визначених правил.

Управління програмною інженерією (*Software Engineering Management*) – це застосування питань управління (*Management Activities*) – планування, координації, кількісної оцінки, моніторингу, контролю та звітності – до інженерної діяльності для систематичного, впорядкованого та кількісно вимірюваного забезпечення розробки та супроводу програмних систем .

Управлінська діяльність у програмній інженерії відбувається на трьох рівнях:

1)організаційне управління та управління інфраструктурою;

2)управління проектами;

3)планування та контроль програм кількісного оцінювання .

Функціональна вимога *(Functional Requirements)* – вимога, конкретизує функцію, яку система або її компонент повинен виконувати.

Загальні вимоги до властивостей і якостей випускників вищого навчального закладу як соціальних особистостей подаються у вигляді переліків компетенцій щодо вирішення певних проблем і задач соціальної діяльності, інструментальних, загально-наукових і професійних компетенцій та системи умінь, що забезпечують наявність цих компетенцій.

Таким чином ОПП і ОКХ , що розглянуті вище, є галузевими стандартами вищої освіти України для напряму підготовки **121 ..Інженерія програмного забезпеченн ”** з присвоєнням кваліфікації **..Фахівець з розробки та тестування програмного забезпечення”** на основі захисту дипломної роботи.

Питання для самоконтролю.

1.Що уявляє собою освітньо-професійна програма бакалавра ?

2.Які обов’язкові розділи включає в себе стандарт ОПП ?

3.Які типи мов конструювання програмних продуктів ?

4.Що таке освітньо-кваліфікаційна характеристика ?

5. Які вимоги до реалізації в програмному коді згідно ОКХ ?

6.Що таке верифікація та атестація програмних продуктів ?

7.Яку кваліфікацію присвоюють при захисті диплому бакалавра з напряму підготовки ..Інженерія програмного забезпечення ” ?

Розділ 3. Короткий історичний нарис розвитку обчислювальної техніки та програмної інженерії

У квітні 1914 року за чотири місяці до початку Першої світової війни професор Харківського технологічного інституту Олександр Миколайович Щукарьов на прохання Московського Політехнічного музею приїхав до Москви й прочитав лекцію "Пізнання і мислення". Лекція супроводжувалася демонстрацією створеної професором "машини логічного мислення", здатної механічно здійснювати прості логічні висновки на основі початкових смислових посилань. Лекція мала великий

резонанс. Присутній на ній професор А.Н.Соков відгукнувся статтею з провідчою назвою "Мисляча машина" (журнал "Навколо світу", №18, 1914 р.), у якій написав: "Якщо ми маємо арифмометри, що додають, віднімають, множать мільйонні числа поворотом важеля, то, очевидно, час вимагає мати логічну машину, здатну робити логічні висновки й умовиводи одним натискуванням відповідних клавіш. Це збереже масу часу, залишивши людині область творчості, гіпотез, фантазії, натхнення - душу життя". Нагадаємо, що в 1914 р., коли була опублікована стаття, геніальному англійському математикові Алану Тьюрінгу, який опублікував у 1947 р. статтю "Думаюча машина. Еретична теорія", а в 1950 р. другу "Чи може машина мислити?", йшов другий рік.

"Машина логічного мислення" О.М. Щукарьова являла собою ящик висотою 40 см, довжиною 25 см і шириною 20 см. У машині знаходились 16 штанг, що приводилися в рух натисненням кнопок, розміщених на панелі введення початкових даних (смислових посилань). Кнопки діяли на штанги, ті на світлове табло, де висвічувався (словами) кінцевий результат (логічні висновки із заданих смислових посилань).

О.М. Щукарьов вів велику просвітницьку роботу, виступав з лекціями на тему пізнання і мислення в багатьох містах України, Москві та Петербурзі.

Головне, що зробив Щукарьов, полягало в тому, що він, на відміну від інших дослідників, бачив у машині не тільки навчальний посібник, а представляв її своїм слухачам як технічний засіб механізації формалізованих сторін мислення.

Оскільки в Радянському Союзі кібернетику вважали спочатку псевдонаукою, то в 20-і роки минулого століття погляди О.М. Щукарьова, окрім доброзичливого відношення, оцінювалися деякими вченими різко негативно..

В історії розвитку інформаційних технологій (ІТ) в Україні ім'я професора Щукарьова пов'язане з важливим кроком у галузі засобів обробки інформації - розумінням та активною пропагандою важливості й можливості механізаці (у подальшому автоматизації) формальних сторін логічного мислення.

У середині XX ст. з переходом від десяткової до двійкової системи числення замість зубчастих коліс почали використовувати електромагнітні реле, електронні лампи та ферритові сердечники. Скоро на зміну їм прийшли транзистори, які вдосконалюючись, перетворились у інтегральні схеми, що містили спочатку тисячі, а згодом мільйони компонентів.

За п'ятдесят років використання транзисторів у них не з'явилося

серйозних конкурентів. Природно задати запитання, хто був першовідкривачем фізичних ефектів, покладених в основу транзистора. Щоб відповісти, розкриємо ще одну "білу пляму" в історії розвитку ІТ в Україні. Вона пов'язана з іменем та діяльністю видатного українського фізика Вадима Євгеновича Лашкарьова (1903-1974). Він по праву повинен був би увійти до групи американських учених (Джон Бардін, Вільям Шоклі, Уотер Браттейн), які в 1956 р. були удостоєні Нобелівської премії з фізики за відкриття транзисторного ефекту.

Ще в 1941 р. В.Є. Лашкарьов опублікував статтю "Дослідження запірних шарів методом термозонду" (Известия АН СССР, сер. физ., т. 5, 1941 г.) і у співавторстві з К.М. Косоноговою статтю "Вплив домішок на вентильний фотоефект у закислі міді" (там же). Він установив, що обидві сторони "запірного шару", розміщеного паралельно границі розділу мідь - закисел міді, мають протилежні знаки носіїв струму. Згодом це явище отримало назву *p-n* переходу (*p* – від pozitive, *n* — від negative). Ним же "був розкритий механізм інжекції—дуже важливого явища, на основі якого діють напівпровідникові діоди та транзистори" (Н.Н. Боголюбов й др. Памяти Вадима Евгеньевича Лашкарева, Успехи физических наук, т. 117, вып. 2, с. 377-378, 1975г.).

Перше повідомлення в американській пресі про появу напівпровідникового підсилювача-транзистора з'явилося в липні 1948 г., через сім років після статті Лашкарьова. Його винахідники американські вчені Бардін та Браттейн пішли по шляху створення так званого точкого транзистора на базі кристалу германію *n* типу. Перший обнадійливий результат був одержаний ними в кінці 1947 р. Проте прилад вів себе нестало, його характеристики вирізнялись непередбачуваністю, і тому практичного використання точковий транзистор не отримав. У 1951 р. у США з'явився більш надійний площинний транзистор *n-p-n* типу. Його створив Шоклі. Транзистор складався із трьох шарів германію *n, p* і *n* типу, загальною товщиною 1 см і був зовсім не схожий на пізніші мініатюрні, а з часом і невидимі оку компоненти інтегральних схем.

Уже через кілька років значимість винаходу американських учених стала очевидною, і вони були відзначені Нобелівською премією. Можливо, що розпочата "холодна війна" або існуюча тоді "залізна завіса" завадили добавити ще одного лауреата - В.Є. Лашкарьова. Його інтерес до напівпровідників не був випадковим. Починаючи з 1939 р. і до кінця життя, він послідовно й результативно займався дослідженням їх фізичних властивостей. Після двох перших робіт у 1950 р. він і В.І.Ляшенко опублікували статтю "Електронні стани на поверхні напівпровідників"

(Ювілейний збірник до 70-річчя А.Ф. Іоффе, 1950 р.), у якій описали результати досліджень поверхневих явищ у напівпровідниках, покладені в подальшому в основу роботи інтегральних схем на базі польових транзисторів.

Під його керівництвом на початку 50-х років в Інституті фізики НАН України було організовано виробництво точкових транзисторів. Сформована В.Є. Лашкарьовим наукова школа в галузі фізики напівпровідників стала однією з провідних у Радянському Союзі. Визнанням значимості її наукових результатів стало створення в 1960 р. Інституту напівпровідників НАН України, директором якого був призначений В.Є. Лашкарьов.

Учений народився і одержав вищу освіту в Києві, потім працював у Ленінграді. На жаль, перші роки його творчої діяльності співпали з роками репресій, що розпочалися з убивства Кірова в 1934 р. Він був заарештований і засланий до Архангельська, де завідував кафедрою фізики в мединституті до 1939 р. Наступні самі плодотворні 35 років свого життя він провів у Києві, залишивши після себе цілу плеяду учнів, які стали великими вченими і успішно продовжували розпочаті Лашкарьовим дослідження.

Підводячи підсумок, можна сказати, що В.Є. Лашкарьов є піонером у розвитку обчислювальної техніки у бувшому Радянському Союзі в галузі технології компонентів (транзисторної елементної бази) засобів обчислювальної техніки. Цілком справедливо вважати його й одним з перших у світі основоположників транзисторної мікроелектроніки.

Важливою складовою ІТ, у якій сконцентрований їх інтелект, є програмне забезпечення (ПЗ) комп'ютерів. Про його важливість для економіки свідчить те, що розробка ПЗ в країнах - технологічних лідерах стала самодостатньою індустрією. Сьогодні ІТ-індустрія безпосередньо забезпечує роботою 9 млн. високооплачуваних кваліфікованих співробітників у більш ніж 4 тис. компаній в усьому світі. Крім того, цей сектор економіки створює зайнятість ще для 21 млн. ІТ-спеціалістів у найрізноманітнітніших сферах діяльності. Число робочих місць в ІТ-індустрії в цілому за період з 1996-2002 рік виросло на 40%, а в галузі програмного забезпечення - на 76%. ІТ-індустрія приносить до бюджетів своїх країн понад 700 млрд. дол. податкових надходжень на рік. Внесок ІТ-індустрії у світову економіку складає майже 1 трлн. дол. на рік, у тому числі 330 млрд. дол. надходять від галузі виробництва комп'ютерного обладнання, 180 млрд. дол. - від галузі "тиражного" програмного забезпечення і ще 420 млрд.дол.-відІТ-послуг.

Завдяки своїм особливостям індустрія програмування як ніяка інша опинилась у центрі процесів глобалізації світової економіки. Незважаючи на те, що в Україні фактично не проводилась державна політика у цьому напрямку, де-факто вона приймає участь у цих процесах. На жаль, не як суб'єкт, а як об'єкт політики інших держав. Створення індустрії ПЗ в Україні - проблема державної ваги.

Значний внесок у розвиток обчислювальної техніки вніс Сергій Олексійович Лебедєв. Усе його творче життя ділиться на два періоди, перший пов'язаний з діяльністю в галузі енергетики, а другий - повністю відданий справі комп'ютеробудування: створення ЕОМ та налагодження їх серійного виробництва. Між цими двома періодами стоять п'ять років (1946-1951 рр.), проведених Лебедєвим у Києві, куди його запросили на посаду директора Інституту електротехніки НАН України, і де згодом була створена перша ЕОМ.

У грудні 1976 р. відбулося засідання Вченої ради Інституту кібернетики НАН України, присвячене 25-річчю введення в регулярну експлуатацію першої на континентальній частині Європи Малої електронної лічильної машини МЕСМ, створеної в Інституті електротехніки НАН України під керівництвом Сергія Олексійовича Лебедєва (1902-1974).

Виступаючи на засіданні, директор інституту академік В.М. Глушков так оцінив творчий вклад творця МЕСМ: "Незалежно від зарубіжних учених С.О. Лебедєв розробив принципи будови ЕОМ з програмою, що зберігається в пам'яті. Під його керівництвом була створена перша на континентальній частині Європи ЕОМ, за короткі строки були розв'язані важливі науково-технічні задачі, чим було покладено початок радянській школі програмування. Опис МЕСМ став першим підручником у країні з обчислювальної техніки. МЕСМ стала прототипом Великої електронної лічильної машини БЕСМ. Лабораторія С.О.Лебедєва стала організаційним зародком Обчислювального центру - згодом Інституту кібернетики НАН України".

Твердження В.М. Глушкова про те, що С.О. Лебедєв незалежно від учених Заходу розробив принципи будови ЕОМ з програмою, що зберігається в пам'яті, являється принципово важливим. Саме збереження програми в оперативній пам'яті стало завершальним кроком у розвитку перших ЕОМ. На Заході цей крок пов'язується з Джоном фон Нейма-ном. Оскільки висловлення В.М. Глушкова підтверджується низкою архівних документів і спогадами людей, близьких до С.О. Лебедєва, можна стверджувати, що цей крок слід пов'язувати не лише з ім'ям Джона фон

Неймана, але й С.О. Лебедєва.

Згідно протоколу №1 засідання закритої Вченої ради Інституту електротехніки та Інституту теплоенергетики АН УРСР від 8 січня 1951 р. С.О. Лебедєв, відповідаючи на задані йому запитання після доповіді про МЕСМ, сказав: "Я маю дані на 18 машин, розроблених американцями, ці дані носять характер реклами, без будь-яких відомостей, як машини побудовані... Використати закордонний досвід важко, оскільки опубліковані відомості досить скупі".

У короткій записці, поданій до Ради з координації АН СРСР на початку 1957 р., С.О. Лебедєв констатує: "У 1948-1949 рр. мною були розроблені основні принципи будови подібних машин. Ураховуючи їх виключне значення для народного господарства, а також відсутність у Союзі якого-небудь досвіду їх побудови та експлуатації, я прийняв рішення якомога швидше створити малу електронну лічильну машину, на якій можна було б досліджувати принципи будови, перевірити методику рішення окремих задач і накопичити експлуатаційний досвід".

Не випадково абревіатура МЕСМ спочатку розшифровувалася як Модель електронної лічильної машини, і лише пізніше слово *„Модель”* було замінене словом *„Мала”*. Розробка основних елементів була проведена у 1948 р., на кінець 1949 р. було розроблене загальне компонування машини та принципові схеми її блоків. У першій половині 1950 р. виготовлені окремі блоки і розпочали їх налагодження у взаємозв'язку. Уже в кінці 1950 р. запрацював макет МЕСМ. У грудні 1951 р. МЕСМ була прийнята державною комісією до регулярної експлуатації. На ній, єдиній на той час, весь 1952 р. розв'язувалися найважливіші для всієї країни задачі того часу: фрагменти розрахунків з термоядерних процесів, космічних польотів та ракетної техніки, віддалених ліній електропередач та ін.

Досвід створення та експлуатації МЕСМ, як і передбачав Лебедєв С.О., дозволив йому за короткі строки (за наступні два роки уже в Інституті точної механіки та обчислювальної техніки АН СРСР) створити Велику електронну лічильну машину БЕСМ.

У статті "У колиски першої ЕОМ" С.О. Лебедєв назвав МЕСМ "первістком радянської обчислювальної техніки". Про БЕСМ Сергій Олексійович написав: "Коли машина була готова, вона нічим не поступалася американським зразкам і являла собою справжнє торжество ідей її творців".

Основні принципи будови МЕСМ випливають з опису машини, наявного в книжці (раніше секретної) "Мала електронна лічильна

машина" (автори С.О. Лебедєв, Л.Н. Дашевський, К.О.Шкабара, 1952р.):

- у машині використовується двійкова система числення;
- до складу машини входять п'ять пристроїв: арифметичний, пам'яті, управління, введення та виведення;
- обчислення здійснюються автоматично за рахунок програми, що зберігається в пам'яті;
- до числа операцій, крім арифметичних, входять логічні операції: порівняння, умовного й безумовного переходів;
- пам'ять будується за ієрархічним принципом;
- для обчислення використовуються числові методи розв'язування задач.

У 1956 р. на конференції у Дармштадті доповідь С.О. Лебедєва про БЕСМ справила сенсацію: мало відома за межами СРСР машина була визнана самою швидкодіючою в Європі.

Слід відзначити вклад у розвиток обчислювальної технік зарубіжних вчених фон Неймана, який запропонував принципи і структура ЕОМ, що стали називатися нейманівськими, а також Мочлі й Еккерта, які створили першу ЕОМ в 1945 році ENIAC. До речі ENIAC, створена під керівництвом фон Неймана, була закінчена на рік пізніше МЕСМ. У той час МЕСМ була засекречена, і творчий вклад С.О. Лебедєва не був відомий зарубіжним вченим.

На фоні чудових досягнень зарубіжних вчених результати діяльності С.О. Лебедєва в галузі комп'ютеризації за наступні двадцять років (після створення МЕСМ і БЕСМ) вражають своєю масштабністю. Під його керівництвом і безпосередньою участю були створені ще 18 ЕОМ, причому 15 з них випускалися великими серіями. І це за існуючої технологічної відсталості (тоді ще невеликої). Не випадково учень С.О. Лебедєва академік В.І. Мельников відзначав: "Геніальність С.О. Лебедєва полягала в тому, що він ставив мету з врахуванням перспективи розвитку структури майбутньої машини, умів правильно вибрати засоби для її реалізації стосовно до можливостей вітчизняної промисловості" (УСиМ, 1976, №6). Серед розроблених під його керівництвом були супер-ЕОМ для обчислювальних центрів, ЕОМ для протиракетних систем та протилітакової ракетної зброї.

Його діяльність розпочалася з лампових ЕОМ, що виконують десятки тисяч операцій за секунду. На той час це були супер-ЕОМ. Створені в 1958 і 1959 рр. ЕОМ М40 та М50 виявилися самими швидкодіючими у світі. З появою напівпровідникових та магнітних елементів він перейшов до створення супер-ЕОМ другого покоління.

Створена в 1967 р. БЕСМ-6 з продуктивністю 1 млн. операцій за секунду випускалася 17 років. Нею були оснащені кращі обчислювальні центри СРСР. Про те, що БЕСМ-6 зайняла гідне місце у світовому комп'ютерному будівництві говорить той факт, що Лондонський музей науки в 1972 р. придбав машину, щоб зберегти її для історії. Завершенням діяльності вченого стало створення супер-ЕОМ на інтегральних схемах, продуктивністю в мільйони операцій за секунду. Кожна ЕОМ була новим словом в історії обчислювальної техніки - більш продуктивна, більш надійна та зручна в експлуатації.

Генеральним принципом побудови всіх машин було розпаралелювання обчислювального процесу. В МЕСМ і БЕСМ з цією метою були використані арифметичні пристрої паралельної дії. У М20, М40, М50 додалася можливість роботи зовнішніх пристроїв паралельно з процесором. У БЕСМ-6 з'явився конвейєрний (або "водопровідний", як називав його сам Лебедєв) спосіб виконання обчислень. У наступних ЕОМ використовувалася багато процесорність та ін. Кожна нова ЕОМ була результатом радикальної переробки та критичного осмислення всього нового, що з'явилося в країні і за кордоном.

Активна творча діяльність С.О.Лебедєва призвела до створення великої наукової школи, а керований ним Інститут точної механіки та обчислювальної техніки АН СРСР став провідним у країні й не поступався своїми досягненнями в 50—70-і роки відомій американській фірмі IBM. Найбільш повну та всебічну характеристику діяльності С.О. Лебедєва дав президент НАН України Б.Є. Патон. "Ми завжди будемо гордитися тим, що саме в Академії наук України, у нашому рідному Києві, розцвів талант С.О. Лебедєва як видатного вченого в галузі обчислювальної техніки і математики, а також величезних автоматизованих систем”. Він поклав початок створенню в Києві чудової школи в галузі інформатики. Цю естафету підхопив В.М.Глушков, ім.’я якого інститут кібернетики при НАНУкраїни.

Питання для самоконтролю.

1. Який вклад Щукарьова О.М. у розвитк інформаційних технологій на Україні ?
2. Шо запропоновано В.Є. Лашкарьовим у розвиток обчислювальної техніки ?
3. Чому електронні обчислювальні машини працюють у двійковій системі числення ?
4. Під керівництвом яких вчених була створена перша ЕОМ , де і коли ?

5. Під керівництвом якого вченого була створена перша вітчизняна ЕОМ , де і коли ?

Розділ 4. Структура персонального комп'ютера , основні характеристики та складові апаратного забезпечення

Комп'ютер - це універсальна технічна система, спроможна чітко виконувати визначену послідовність операцій певної програми. Персональним комп'ютером (ПК) може користуватись одна людина без допомоги обслуговуючого персоналу. Взаємодія з користувачем відбувається через багато середовищ, від алфавітно-цифрового або графічного діалогу за допомогою дисплея, клавіатури та мишки до пристроїв віртуальної реальності.

Конфігурацію ПК можна змінювати в міру необхідності. Але, існує поняття базової конфігурації, яку можна вважати типовою:

- системний блок;
- монітор;
- клавіатура;
- мишка.

Комп'ютери випускаються і у портативному варіанті (laptop або notebook виконання). В цьому випадку, системний блок, монітор та клавіатура містяться в одному корпусі: системний блок прихований під клавіатурою, а монітор вбудований у кришку.

Системний блок - основна складова, в середині якої містяться найважливіші компоненти. Пристрої, що знаходяться в середині системного блока називають внутрішніми, а пристрої, що під'єднуються ззовні називають зовнішніми. Зовнішні додаткові пристрої, що призначені для вводу та виводу інформації називаються також периферійними.

Основними вузлами системного блоку є:

- електронні плати, що керують роботою комп'ютера (мікропроцесор, оперативна пам'ять, контролери пристроїв тощо);
- накопичувач на жорсткому диску (вінчестер), призначений для читання або запису інформації;
- накопичувачі (дисководи) для гнучких магнітних дисків (дискет), флешок тощо.

Основною платою ПК є материнська плата (MotherBoard). На ній розташовані:

- **процесор** - основна мікросхема, що виконує математичні та логічні операції;

- **чіпсет** (мікропроцесорний комплект) - набір мікросхем, що керують роботою внутрішніх пристроїв ПК і визначають основні функціональні можливості материнської плати;
- **шини** - набір провідників, по яких відбувається обмін сигналами між внутрішніми пристроями комп'ютера;
- **оперативний запам'ятовуючий пристрій** (ОЗП) - набір мікросхем, що призначені для тимчасового зберігання даних, поки включений комп'ютер;
- **постійний запам'ятовуючий пристрій** (ПЗП) - мікросхема, призначена для довготривалого зберігання даних, навіть при вимкненому комп'ютері;
- **роз'єми** для під'єднання додаткових пристроїв (слоти).

Процесор - головна мікросхема комп'ютера, його "мозок". Він дозволяє виконувати програмний код, що знаходиться у пам'яті і керує роботою всіх пристроїв комп'ютера. Швидкість його роботи визначає швидкодію комп'ютера. Конструктивно, процесор - це кристал кремнію дуже маленьких розмірів. Процесор має спеціальні комірки, які називаються регістрами. Саме в цих регістрах містяться команди, які виконуються процесором, а також дані, якими оперують ці команди. Робота процесора полягає у вибиранні з пам'яті у певній послідовності команд та даних і виконанні їх. На цьому і базується виконання програм.

Основними параметрами процесорів є:

- тактова частота,
- розрядність,
- робоча напруга,
- коефіцієнт внутрішнього домноження тактової частоти,
- розмір кеш пам'яті.

Тактова частота визначає кількість елементарних операцій (тактів), що виконуються процесором за одиницю часу. Тактова частота сучасних процесорів вимірюється у МГц (1 Гц відповідає виконанню однієї операції за одну секунду). Чим більша тактова частота, тим більше команд може виконати процесор, і тим більша його продуктивність.

Розрядність процесора показує, скільки біт даних він може прийняти і обробити в свої регістрах за один такт. Розрядність процесора визначається розрядністю командної шини, тобто кількістю провідників у шині, по якій передаються команди. Сучасні процесори сімейства Intel є 32-розрядними.

Робоча напруга процесора забезпечується материнською платою, тому різним маркам процесорів відповідають різні материнські плати.

Зараз робоча напруга процесорів не перевищує 3В. Пониження робочої напруги дозволяє зменшити розміри процесорів, а також зменшити тепловиділення в процесорі, що дозволяє збільшити його продуктивність без загрози перегріву.

Коефіцієнт внутрішнього домноження тактової частоти - це коефіцієнт, на який слід помножити тактову частоту материнської плати, для досягнення частоти процесора. Тактові сигнали процесор отримує з материнської плати, яка з чисто фізичних причин не може працювати на таких високих частотах, як процесор. На сьогодні тактова частота материнських плат складає 100-133 МГц. Для отримання більш високих частот у процесорі відбувається внутрішнє домноження на коефіцієнт 4, 4.5, 5 і більше.

Кеш-пам'ять. Обмін даними всередині процесора відбувається набагато швидше ніж обмін даними між процесором і оперативною пам'яттю. Тому, для того щоб зменшити кількість звертань до оперативної пам'яті, всередині процесора створюють так звану надоперативну або кеш-пам'ять. Коли процесору потрібні дані, він спочатку звертається до кеш-пам'яті, і тільки якщо там потрібні дані відсутні, відбувається звертання до оперативної пам'яті. Чим більший розмір кеш-пам'яті, тим більша ймовірність, що необхідні дані знаходяться там. Тому високопродуктивні процесори оснащуються підвищеними обсягами кеш-пам'яті. Розрізняють кеш-пам'ять першого рівня (виконується на одному кристалі з процесором і має об'єм порядку декілька десятків Кбайт), другого рівня (виконується на окремому кристалі, але в межах процесора, з об'ємом в сто і більше Кбайт) та третього рівня (виконується на окремих швидкодійних мікросхемах із розташуванням на материнській платі і має обсяг один і більше Мбайт).

У процесі роботи процесор обробляє дані, що знаходяться в його регістрах, оперативній пам'яті та зовнішніх портах процесора. Частина даних інтерпретується як власне дані, частина даних - як адресні дані, а частина - як команди. Сукупність різноманітних команд, які може виконати процесор над даними, утворює так звану систему команд процесора. Чим більший набір команд процесора, тим складніша його архітектура, тим довший запис команд у байтах і тим довша середня тривалість виконання команд.

Шини. З іншими пристроями, і в першу чергу з оперативною пам'яттю, процесор зв'язаний групами провідників, які називаються шинами. Основних шин три:

- адресна шина,

- шина даних,
- командна шина.

Адресна шина. Дані, що передаються по цій шині трактуються як адреси комірок оперативної пам'яті. Саме з цієї шини процесор зчитує адреси команд, які необхідно виконати, а також дані, із якими оперують команди. У сучасних процесорах адресна шина 32-розрядна, тобто вона складається з 32 паралельних провідників.

Шина даних. По цій шині відбувається копіювання даних з оперативної пам'яті в регістри процесора і навпаки. У ПК на базі процесорів Intel Pentium шина даних 64-розрядна. Це означає, що за один такт на обробку поступає відразу 8 байт даних.

Командна шина. По цій шині з оперативної пам'яті поступають команди, які виконуються процесором. Команди представлені у вигляді байтів. Прості команди вкладаються в один байт, але є й такі команди, для яких потрібно два, три і більше байтів. Більшість сучасних процесорів мають 32-розрядну командну шину, хоча існують 64-розрядні процесори з командною шиною.

Шини на материнській платі використовуються не тільки для зв'язку з процесором. Усі інші внутрішні пристрої материнської плати, а також пристрої, що підключаються до неї, взаємодіють між собою за допомогою шин. Від архітектури цих елементів багато в чому залежить продуктивність ПК у цілому.

Внутрішня пам'ять Під внутрішньою пам'яттю розуміють всі види запам'ятовуючих пристроїв, що розташовані на материнській платі. До них відносяться:

- оперативна пам'ять,
- постійна пам'ять,
- енергонезалежна пам'ять.

Оперативна пам'ять RAM (Random Access Memory). Пам'ять RAM - це масив кристалічних комірок, що здатні зберігати дані. Вона використовується для оперативного обміну інформацією (командами та даними) між процесором, зовнішньою пам'яттю та периферійними системами. З неї процесор бере програми та дані для обробки, до неї записуються отримані результати. Назва "оперативна" походить від того, що вона працює дуже швидко і процесору не потрібно чекати при зчитуванні даних з пам'яті або запису. Однак, дані зберігаються лише тимчасово при включеному комп'ютері, інакше вони зникають.

Оперативна пам'ять у комп'ютері розміщена на стандартних панельках, що звуться модулями. Модулі оперативної пам'яті вставляють у

відповідні роз'єми на материнській платі.

Постійна пам'ять ROM (Read Only Memory) В момент включення комп'ютера в його оперативній пам'яті відсутні будь-які дані, оскільки оперативна пам'ять не може зберігати дані при вимкненому комп'ютері. Але процесору необхідні команди, в тому числі і відразу після включення. Тому процесор звертається за спеціальною стартовою адресою, яка йому завжди відома, за своєю першою командою. Ця адреса вказує на пам'ять, яку прийнято називати постійною пам'яттю ROM або постійним запам'ятовуючим пристроєм (ПЗП). Мікросхема ПЗП здатна тривалий час зберігати інформацію, навіть при вимкненому комп'ютері. Кажуть, що програми, які знаходяться в ПЗП, "зашиті" у ній - вони записуються туди на етапі виготовлення мікросхеми. Комплект програм, що знаходиться в ПЗП утворює базову систему введення/виведення BIOS (Basic Input Output System). Основне призначення цих програм полягає в тому, щоб перевірити склад та працездатність системи та забезпечити взаємодію з клавіатурою, монітором, жорсткими та гнучкими дисками.

Енергонезалежна пам'ять. На материнській платі є мікросхема енергонезалежної пам'яті, яка по технології виготовлення називається CMOS. Від оперативної пам'яті вона відрізняється тим, що її вміст не зникає при вимкненні комп'ютера, а від постійної пам'яті вона відрізняється тим, що дані можна заносити туди і змінювати самостійно, у відповідності з тим, яке обладнання входить до складу системи. Мікросхема пам'яті CMOS постійно живиться від невеликої батарейки, що розташована на материнській платі. У цій пам'яті зберігаються дані про гнучкі та жорсткі диски, процесори тощо. Той факт, що комп'ютер чітко відслідковує дату і час, також пов'язаний з тим, що ця інформація постійно зберігається (і обновлюється) у пам'яті CMOS. Таким чином, програми BIOS зчитують дані про склад комп'ютерної системи з мікросхеми CMOS, після чого вони можуть здійснювати звертання до жорсткого диска та інших пристроїв.

Всі роботи з ПК діляться на три категорії:

1.Епізодичне прочитування і введення інформації не більше 2 години за 8-годинну робочу зміну.

2.Прочитування інформації або творча робота не більше 4 години за 8-годинну зміну.

3.Прочитування інформації або творча робота більше 4 годин за 8-годинну зміну.

При покупці монітора, завжди виникають питання про безпеку для здоров'я. Монітор є джерелом резидентної (постійної) радіації, може

провокувати загострення захворювань очей. Неправильна організація робочого місця, зокрема, неправильне розташування монітора на столі може послужити причиною комп'ютерного зорового синдрому, а також постійних болів у шийному відділі хребта

Більшість користувачів при тривалій роботі з монітором відчувають біль в очних яблуках, чи сльозотечу, навпаки, сухість, почервоніння очей. При цьому часто турбують головні болі, з'являється швидка стомлюваність. Американські вчені знайшли, що все це може бути наслідком тривалої роботи з монітором. Особливостями дисплейного зображення є його висока частота регенерації (частота кадрів), відносно низька контрастність, а також той факт, що монітор є джерелом світла. Центральна нервова система людини сприймає всю інформацію, що надходить через очі, однак далеко не усе доходить до свідомості. Маса непотрібної інформації, наприклад, мелькання за межами монітора, може викликати через визначений час стомлення. Ця реакція спрямована на те, щоб відвернути людину від якоїсь роботи, змусити її зробити перерву, а потім з новими силами відновити роботу. Ті ж, хто цього не розуміє, ризикують постійно випробувати симптоми комп'ютерного зорового синдрому. Відомо, що рано чи пізно такий комплекс виникає у всіх користувачів. Для профілактики комп'ютерного зорового синдрому необхідно проводити комплекс вправ для очей. Програму з роз'ясненнями можна знайти на сайті: http://www.gi.ru/eyeskeeper.

Принцип зберігання інформації в пам’яті комп’ютера. Основними принципами зберігання інформації в пам’яті комп’ютера є: використання двійкової системи числення для кодування інформації; програмне керування роботою комп’ютера; збереження програм у пам’яті комп’ютера; адресація пам’яті.

Інформація (команди і дані) у комп'ютері кодуються у двійковій системі числення. Зручність використання двійкової системи в обчислювальній техніці обумовлена тим, що електронні перемикачі можуть перебувати тільки в одному із двох станів: увімкненому чи вимкненому. Ці стани можна кодувати двома цифрами: одиницею або нулем. Так само в двох станах у кожен окремий момент часу може перебувати канал передачі даних: «рівень напруги високий» або «рівень напруги низький». Крім того, логіка висловлень є двійковою логікою: будь-яке висловлення в кожен момент є істинним або хибним. Якщо висловлення є істинним, йому відповідає значення 1, якщо хибне —значення 0. Одна двійкова цифра називається бітом (від англ. binary digit — двійкова цифра). За допомогою одного біта можна закодувати два

інформаційних повідомлення, що умовно позначаються символами «0» та «1». Із двох бітів можна утворити коди чотирьох інформаційних повідомлень: «00», «01», «10» та «11»; із трьох бітів — восьми. У загальному випадку за допомогою n бітів можна закодувати 2n інформаційних повідомлень. Послідовність, що складається з восьми бітів, у сучасній обчислювальній техніці прийнято називати байтом. За допомогою одного байта можна закодувати 2^8 = 256 різних повідомлень і зобразити, наприклад, значення цілих чисел від 0 до 255. Ці самі повідомлення можна розглядати і як числа від -128 до 127 (їх кількість теж дорівнює 256), символи, логічні значення тощо. Байт є одиницею обсягу пам'яті. Для позначення тисяч та мільйонів байтів використовуються такі одиниці, як кілобайт (1 Кбайт = 2^10 = 1 024 байт), мегабайт (1 Мбайт = 2^20 = 1 048 576 байт) і гігабайт (1 Гбайт = 2^30 = 1 073 741 824 байт). Послідовностями двійкових цифр можна кодувати не лише числа, а й довільні інформаційні повідомлення. Зокрема, загальноприйнята система кодування ASCII ставить коди у відповідність 256 символам. Наприклад, латинській літері «a» відповідає код 97_(10) = 1100001_(2) , а символу «@» - код 64_(10) = 1000000_(2). Складніші системи кодування дають можливість закодувати зображення, звук тощо.

Питання для самоконтролю.

1.Перелічите основні складові базової конфігурації персонального комп’ютера ?,

2.Які основні вузли системного блоку ?

3.Що розташовано на материнській платі ?

4.Перелічите основні параметрами процесора і дайте їм коротку характеристику.

5.Які існують типи памяті в ПК?

6.Які типи шинні Ви знаєте ?

7.На які категорії трудової діяльності поділяють ПК в залежності від режимів праці і відпочинку при роботі з ним ?

8. Які основними принципами зберігання інформації ?

9. В чому різниця між позиційною і непозиційною система ислення ?

Розділ 5. Основи операційних систем

5.1. Основні поняття, функції і складові частини операційної системи

Напевно, жодне з існуючих на даний момент визначень поняття ***«операційна система»*** не може претендувати на універсальність і повноту. При умові даного застереження зупинимося на наступному формулюванні:

Операційна система (ОС) — це комплекс спеціальних програмних засобів, призначених для керування завантаженням, запуском і виконанням інших програм, а також для планування і керування обчислювальними ресурсами ПК. Важливо підкреслити те, що операційна система — це саме комплекс програм, неоднорідний по характеру і багатоплановий за рівнем. Цей комплекс програм є динамічним за своїм складом: з нього можна вилучати й додавати певні частини. Та частина програм, що взаємодіє з апаратними засобами безпосередньо і тому повинна постійно зберігається в комп'ютері, являє ядро операційної системи. Зокрема, програмне забезпечення, що входить до складу ядра, відповідає за перевірку працездатності комп'ютера і виконання елементарних (базових) операцій, зв'язаних з роботою дисплея, клавіатури, магнітних накопичувачів тошо.

Операційна система утворює автономне середовище, що не зв'язане з жодною з мов програмування. Будь-яка прикладна програма пов'язана з операційною системою і може експлуатуватися тільки на тих комп'ютерах, де існує аналогічне системне середовище (або повинна бути забезпечена можливість конвертації — перетворення програм). Очевидно, що операційна система повинна зберігатися на зовнішньому запам'ятовуючому пристрої, до якого може бути забезпечений відносно швидкий доступ.

Операційна система MS DOS, що була до останнього часу найбільше популярним продуктом даного класу для персональних комп'ютерів, була створена фірмою Microsoft у 1961 р. У даний час існують її версії 6.22 і 7,0 (у складі Windows 95), а також, умовно говорячи, системи-дублери інших фірм-розроблювачів (DR DOS, PC DOS). Починаючи з 1996 р. MS DOS розповсюджується у складі Windows 95 — 32-розрядної багатозадачної системи з графічним інтерфейсом і розширеними мережними можливостями.

Програма, що ховає від програміста всі реалії апаратури і надає можливість простого, зручного перегляду зазначених файлів, чи читання запису, — це, звичайно, операційна система. Точно так само, як ОС огороджує програмістів від апаратури дискового накопичувача і надає

йому простий файловий інтерфейс, операційна система бере на себе всі справи, зв'язані з обробкою переривань, керуванням таймерами й оперативною пам'яттю, а також інші низькорівневі проблеми.

Операційна система як система керування включає рішення двох загальних задач, що не залежать від типу ресурсу:

• планування ресурсу — тобто визначення, кому, коли, а для ділених ресурсів — і в якій кількості необхідно виділити даний ресурс;

• відстеження стану ресурсу — тобто підтримка оперативної інформації про те, зайнятий чи не зайнятий ресурс, а для ділених ресурсів — яка кількість ресурсу вже розподілена, а яка вільна.

Від ефективності алгоритмів керування локальними ресурсами комп'ютера багато в чому залежить ефективність усієї мережевої ОС у цілому. Тому, характеризуючи мережеву ОС, часто приводять найважливіші особливості реалізації функцій ОС по керуванню процесорами, пам'яттю, зовнішніми пристроями автономного комп'ютера. Так, наприклад, у залежності від особливостей використаного алгоритма керування процесором операційні системи поділяють на багатозадачні й однозадачні, системи, що підтримують багатовимірну обробку і не підтримуючі її, на багатопроцесорні й однопроцесорні системи.

1. Підтримка багатозадачності. По кількості задач, що одночасно виконуються, операційні системи можуть бути розділені на два класи:

• однозадачні (наприклад, MS DOS, MSX);

• багатозадачні (ОС ЕС, ОС/2, UNIX, Windows 95/98, Windows 200, Windows XP, Linux тощо).

Однозадачні ОС в основному виконують функцію надання користувачу віртуальної машини, роблячи більш простим і зручним процес взаємодії користувача з комп'ютером. Однозадачні ОС включають засоби керування періферійними пристроями, засоби керування файлами, засоби спілкування з користувачем.

Багатозадачні ОС, крім виконання вище наведених функцій, керують також розділенням ресурсів, що сумісно використовуються, таких як процесор, оперативна пам”ять, файли та зовнішні пристрої.

2. Підтримка багатокористувальницького режиму. По кількості користувачів, що одночасно працюють, ОС поділяються:

• на однокористувальницькі (MS DOS, Windows 3.1);

• багатокористувальницькі (UNIX, Windows NT).

3. Виштовхуюча та невиштовхуюча багатозадачність. Важливішим ресурсом є прцесорний час. Спосіб розподілу процесорного часу між декількома одночасно існуючими в системі процесами багато в чому

визначає специфіку ОС. Серед багатьох існуючих варіантів реалізації багатозадачності можна виділити дві групи алгоритмів:

• невиштовхуюча багатозадачність (NetWare, Windows 3.x);

• виштовхуюча багатозадачність (Windows NT, OS/2, UNIX).

4. Багатопроцесорна обробка. Важливою властивістю ОС є відсутність або наявність в ній засобів підтримки багатопроцесорної обробки – мультипроцесіювання. Мультипроцесіювання призводить до ускладнення усіх алгоритмів керування ресурсами.

5.2. Основні функції файлової системи

Основною задачею, яку вирішує операційна система, є забезпечення взаємодії програм і фізичних пристроїв вводу/виводу, таких як накопичувачі на зовнішніх носіях.

Основні функції, що виконуються файловою системою, можна умовно поділити на дві групи: функції для роботи з файлами, тобто їх створення, вилучення, зміна атрибутів, визначення структури файлів; функції для роботи з даними, що зберігаються в файлах, тобто запис, пошук тощо.

Таким чином, в логічному плані файлову систему можна поділити на такі складові частини: файли, що зберігаються на пристрої вводу/виводу; структура файлів; функції роботи з файлами та їх структурою.

В різних джерелах з інформатики та обчислювальної техніки визначення терміна “файл” так само, як і терміна “операційна система” можуть відрізнятись. Найбільш поширеними є: “файл – це найменший іменований масив інформації” або “файл – основна одиниця організації на носії”. В різних файлових системах файли можуть описуватись різними, взагалі кажучи, наборами параметрів і характеристик.

Основним атрибутом файлу є його ім”я. Ім”я файлу – це символьний рядок, довжина якого залежить від файлової системи. Так в файловій системі FAT (File Allocation Table), що використовується в MS DOS, довжина імені не повинна перевищувати 11 символів, 3 з яких – це розширення. В OS UNIX System 5 під ім’я відводиться 14 символів, а в файловій системі NTFS для Windows NT – 255 символів. Від файлової системи також залежить, які саме символи можуть бути використані. Оскільки існують різні файлові системи, перед їх розробниками постає проблема забезпечення сумісності між ними. Наприклад, система,що дозволяє надавати файлам довгі імена, повинна вміти коректно перетворювати їх в короткі. Прикро, що ще не вдалось вирішити проблему використання єдиного імені файла в різних файлових системах.

В деяких системах (UNIX) одному файлу може бути надано декілька імен. Це зручно для систем з багатьма користувачами. В такому випадку

необхідно адресувати файл ідентифікатором, який буде пов’язувати файл і його імена.У системі UNIX таким ідентифікатором служить номер індексного дескриптора.

Інші атрибути файлу, що можуть використовуватися файловою системою, перераховані нижче: поточний розмір файлу; максимальний розмір файлу; довжина запису; часи створення, останнього доступу й останньої зміни; власник файлу; творець файлу; інформація про доступ до файлу; пароль для доступу до файлу; ознака «тільки для читання»; ознака «схований файл»; ознака «системний файл»; ознака «архівний файл»; ознака «двійковий/символьний»; ознака «тимчасовий», ознака блокування.

Для логічної організації файлів використовуються каталоги. Каталог містить файли, об'єднані по якій-небудь ознаці — їхній творець, їхній тип, тема і т.д. Каталог — це файл, що містить інформацію про файли, які він містить. Каталогів на носії може бути багато, і вони можуть мати різні ступені вкладеності. Усі каталоги, що знаходяться на носії, утворять ієрархічну структуру. Структура каталогів у залежності від файлової системи може бути деревоподібною, коли один файл може входити тільки в один каталог, і мережною, коли файл може входити в різні каталоги. Приклад системи з деревоподібною структурою каталогів — система FAT. Мережна структура, більш придатна для систем з багатьма користувачами, реалізована в UNIX.

У так званих DOS-сумісних (утім, і в деяких інших) системах до файлу звертаються за допомогою повного імені, що складається:

• з шляху (англ. Path) — послідовності імен каталогів, у яких міститься файл, розділених символом \ (у UNIX для цього використовується символ /); • власне імені файлу (англ. Name);

• розширення (англ. Extension), що є необов'язковим елементом і, як правило, інформує про тип даних, збережених у файлі, Розширення відділяється від імені крапкою (.)

У різних файлових системах існують обмеження на те, які символи можуть бути присутніми в іменах файлів.

Фізична організація даних описує правила розташування файлів на носії. Розташування файлу описується розташуванням приналежних йому блоків. Блоком називається найменша одиниця даних, якою пристрій введення/виводу може обмінюватися з пам'яттю.

Найпростіший спосіб розташування файлу — неперервна послідовність блоків. Такий спосіб поряд зі своїм основним перевагами —

простотою, котра дозволяє адресувати файл усього лише адресою його першого блоку, має ряд істотних недоліків:

• по-перше, під час створення файлу системі може бути не відомий його розмір, тобто система не знає, скільки місця на носії треба зарезервувати;• по-друге, неминуча сильна фрагментація носія.

Для усунення цих недоліків можуть використовуватися зв'язані блоки. У такому випадку блок крім даних містить посилання на наступний блок тощо. едоліком такого способу є те, що програма не може звернутися до довільної ділянки файлу, і щоб прочитати, , останній блок, необхідно послідовно звернутися до всіх блоків файлу. Недоліком такого способу організації даних є також те, що інформація, яка зберігається в блоці даних, втрачає однорідність, тому що містить нетільки дані файлу, але і службову інформацію.

Вирішити ці проблеми може використання зв'язаного списку індексів, що робиться, наприклад, у MS DOS. При такій організації даних немає необхідності переглядати всі дані для читання останнього блоку файлу, досить переглянути таблицю індексів, При цьому зберігається однорідність даних, що зберігаються в блоці, тому що службова інформація зберігається в окремій області, що розташовується, у випадку жорсткого диску, на внутрішніх доріжках, що забезпечує швидкий доступ до неї.

В системах з багатьма користувачами доступ до файлу різних користувачів повинен бути розмежований, тобто операції з визначеними файлами чи каталогами повинні бути дозволені для одних користувачів і заборонені для інших. Це стосується операцій як з даними, так і з їхньою структурою.

В загальному випадку система розмежування доступу використовує матрицю доступу, стовбці в якій відповідають файлам системи, а рядки — користувачам. На перетинанні описуються операції, які даний користувач може робити з даним файлом чи каталогом. полегшення адміністрування деякі системи дозволяють описувати правила доступу для груп користувачів, тоді правила, встановлені для , діють для усіх користувачів, що входять до неї. Системою може бути надана можливість включення одного користувача в різні групи, а також опис правил як для груп, так і для окремих користувачів, що дозволяє гнучко розмежовувати доступ до файлів.

5.3.Ознайомлення з основними складовими частинами MS DOS

Перелічимо основні складові частини DOS:

1. Модуль розширення системи введення/виводу (IO.SYS).

2. Базовий модуль операційної системи (MSDOS.SYS).
3. Командний процесор (чи командна оболонка) (COMMAND.COM).
4. Зовнішні команди і драйвери, утиліти — файли з розширенням *.COM, *.EXE, *.SYS.
5. В окрему складову частину також можуть бути виділені інструментальні засоби DOS (система програмування, текстовий редактор, налагоджувальна програма DEBUG, що реалізує найпростіші функції тестування і налагодження програм).

В основу поділу DOS на перераховані блоки був покладений принцип модульності: розроблювачами в ідеологію системи була початково закладена можливість відносно безболісної заміни одного блоку при збереженні незмінними інших.

Необхідно також сказати кілька слів про програмне забезпечення, що знаходиться «ближче» до апаратних ресурсів комп'ютера, ніж DOS. Це завантажник і базова система введення/виводу. Вони записані в так звану постійну пам’ять комп’ютера і є його незмінними атрибутами (по крайній мірі, у звичайних умовах експлуатації). Завантажник, як неважко догадатися за назвою, є невеликою програмою, що виконує ініціалізуючі дії комп'ютера відразу після включення живлення.

Базова система введення/виведення BIOS — (Basic Input/Output System) називається так , що містить у собі великий набір модулів, завдяки яким операційна система й інші прикладні програми можуть взаємодіяти з різноманітними пристроями комп'ютера (дисплеєм, клавіатурою, дисководом, таймером тощо). Ще раз підкреслимо, що, з одного боку, BIOS розглядати як складову частину апаратних засобів, а з іншого – вона програмним забезпеченням.

Більшість сучасних відеоадаптерів, а також контролери накопичувачів мають власну систему BIOS, що звичайно доповнює системну. В багатьох випадках програми, що входять у конкретну BIOS, заміняють відповідні програмні модулі основної BIOS. Виклик програм BIOS, як правило, організований у формі так званої обробки апаратних і програмних переривань. Виділення BIOS в окремий модуль дозволяє забезпечити незалежність програмного забезпечення від специфіки конкретної моделі комп'ютерів.

Коротко перелічимо допоміжні функції BIOS:

• пошук на гнучкому і на жорсткому диску програми-завантажника операційної системи і завантаження з диска в оперативну пам'ять;•
тестування апаратної частини (у тому числі оперативної пам'яті);
• ініціалізація векторів переривання нижнього рівня.

BIOS містить спеціальні програми по керуванню роботою стандартними зовнішніми пристроями (драйвери — програми, що розширюють можливості операційної системи); тестові програми для контролю працездатності апаратури; програму початкового завантаження операційної системи.

Як основні функції драйвера можуть бути названі:• прийом і обробка запиту до периферійного пристрою;

• перетворення запиту в серію команд керування пристроєм;

• обробка сигналу переривання.

Драйвери можуть бути:

1. Стандартні (внутрішні) — програми, що знаходяться всередині BIOS чи його модуля розширення EM BIOS. Вони підключаються до системи автоматично після переходу комп'ютера в робочий стан.

2. Завантажувальні (зовнішні) — програми, що знаходяться на диску і призначені для керування зовнішніми пристроями, що відрізняються від стандартних своїми технічними параметрами; підключаються до системи тільки тоді, коли вони зазначені у файлі конфігурації CONFIG.SYS.

MS DOS розподіляються в оперативній пам’яті, доступні в будь-який момент часу. Як приклад вбудованих команд можуть бути названі:

• DIR — виводить інформацію про зміст каталогів логічних пристроїв;

• COPY — виконує операції копіювання файлів і каталогів;

• DEL – виконує операції вилучення файлів і каталогів;

• REN — виконує операції по перейменуванню файлів і каталогів;

• CLS — очищає вміст екрана;

• DATE — виведення/коректування поточної дати;

• TIME — виведення/коректування поточного часу;

• PATH — виведення/задання списку каталогів, у якому (послідовно) буде шукатись файл із програмою, що запускається, якщо він не знайдений у поточному каталозі і явно не задане його повне ім'я.Транзитні команди (утиліти) реалізуються в вигляді файлів з розширеннями *.COM або *.EXE. Ідеологія реалізації основної частини функцій операційної системи через зовнішні утиліти, закладена в MS DOS, дозволила на початковому етапі розвитку забезпечити виняткову компактність її ядра, що було дуже важливо для перших моделей персональних комп'ютерів, обмежених в апаратних можливостях. Надалі це забезпечило широкі можливості по нарощуванню функціональності системи винятково за рахунок додавання все нових і нових утиліт. Одночасно така відкритість в організації MS DOS дуже швидко сформувала розвинутий ринок

додаткового сервісного програмного забезпечення, пропонованого незалежними стосовно Microsoft розроблювачами.

Питання для самоконтролю

1. Що таке операційна система?
2. Які принципи класифікації операційних систем ви можете назвати?
3. Що представляє собою операційна система як система керування ресурсами?
4. Які задачі вирішує файлова система?
5. Сформулюйте основні принципи організації файлової системи FAT.
6. Що таке каталог, підкаталог?
7. Що явдяє собою кластер?
8. Дайте визначення терміну “файл”.

Розділ 6. Ознайомлення з версіями операційних систем Windows

6.1.Зальна характеристика і історія розвитку.

Подальшим кроком у розвитку оболонок операційних систем стала поява у 1986 р. графічної багатовіконної операційної оболонки Windows фірми Microsoft. Після свого виникнення вона пережила ряд модифікацій, не всі з них були вдалими. Однак до завершення 1991 р. вийшла версія Windows 3.1 а трохи згодом— мережний варіант Windows 3.11 (Windows 3,11 For WorkGroups), що завоювало широке визнання мільйонів користувачів. Подальший гомологічний ряд Windows - продуктів продовжили високопродуктивні багатозадачні напівфункціональні операційні системи з графічним інтерфейсом Windows 95, потім Windows 98 і Windows 2000.

Коротко перелічимо основні характеристичні риси програмних продуктів серії Windows:

• ключовою ідеєю Windows є забезпечення повної незалежності програм від апаратної частини комп'ютера — програмна сумісність; • єдиний графічний користувальницький інтерфейс;

• багатозадачність;

• можливість роботи в мережному середовищі;

• наявність універсальної системи засобів обміну даними між додатками;

Графіка у Windows також є універсальною. У такий спосіб знімається проблема забезпечення сумісності з конкретним типом дисплея чи принтера.

Windows 3.1 була популярна графічна операційна оболонка, що запускається на виконання, як звичайна програма MS DOS. Windows 3.1. працювала на базі MS DOS , використовуючи на нижньому рівні внутрішні функції і процедури даної операційної системи.

Принциповою умовою для програмних додатків, призначених для роботи в середовищі Windows, є те, що вони повинні працювати з зовнішніми пристроями (монітором, принтером, плотером і т.д.) не прямо, а через універсальну систему команд. Керуюча система транслює виклики (звертання до того чи іншого фізичного пристрою) і передає їх відповідномуWindows-драйверу даного пристрою, що безпосередньо відповідає за роботу з ним з урахуванням конкретних особливостей його роботи. Майже всі пристрої Windows 3.1 фактично виконують функції BIOS. Користувальницький інтерфейс Windows 3.1 являється графічним (використовує графічний режим роботи відеомонітора). Його основу складає ієрархічно організована система вікон і інших графічних об'єктів, Вікно— структурний і керуючий елемент користувальницького інтерфейсу, що являє собою обрамлену частину екрана, у якому може відображатися додаток, документ чи повідомлення. Цей інтерфейс Windows (на відміну від інтерфейсів командного рядка в DOS і оболонки Norton Commander) реалізує оперативне керування на основі вибору того чи іншого графічно візуалізованного елемента (кнопки, піктограми, списки тощо.) з допомогою манипулятора миші (команди клавіатури, як правило, мають допоміжне чи резервне значення). Набір елементів інтерфейсу стандартний, що дозволяє легко освоювати інтерфейс прикладних програм, що працюють під Windows.

Windows 3.1 об’єктно-орієнтована система. Загальну суть об’єктно-орієнтованого програмування можна сформулювати в такий спосіб : не програми керують даними, а дані керують програмами. Документ, його розглядати з позицій об’єктно-орієнтованого програмування, навряд чи можна вважати об'єктом, однак у Windows за допомогою самостійного механізму зв'язку можна визначити, яким додатком документ буде оброблятися. Реалізація даного підходу в Windows відповідає теорії об'єктно-орієнтованого управління і звільняє користувача від необхідності пам'ятати, яким саме додатком повинний оброблятися документ; досить двічі виконати фіксацію курсором мишкипо значку документа чи файлу, і відповідний додаток запускається, завантажуючи цей документ у своє робоче вікно.

Другою важливою характеристикою Windows як багатозадачного середовища з'явилася реалізація в ній технологічного обміну даними між

різними додатками (причому відразу на декількох рівнях). До них відносяться: передача даних через *буфер обміну* (Clipboard); DDE (Dynamic Data Exchange); OLE (Object Linking and Embedding).

Буфер обміну — це спеціальним образом організоване динамічний простір оперативної пам'яті для тимчасового розміщення даних, прицьому в запам'ятовуються як самі дані, так і те, до якого програмного додатка вони відносяться. Кожне наступне занесення в буфер інформації знищує попереднє. Обмін даними як усередині програм Windows, так і між побудований на базі універсальних системних процедур:

• для копіювання/перенесення даних у буфер обміну використовуються команди "Копіювати/Вирізувати" (вони доступні з будь-яких додатків по натисканні комбінацій клавіш Ctrt+Ins і Shift+Del);

• для вставки (копіювання в додаток-приймач) даних з буфера обміну використовується команда "Вставити" (викликається комбінацією клавіш Shift+Ins). Для того щоб безпосередньо працювати з даними, що зберігаються в буфері обміну, потрібно використовувати вікно „Головна Папка обміну.” Цей додаток дозволяє вміст буфера обміну, зберегти у його файлі і, нарешті, очистити буфер. Черговий буфер одержує назву сторінки в "Папці обміну". Користувач може самостійно привласнити йому ім'я і зберегти у файлі з розширенням *.CLP

Технологія DDE являє собою набір системних процедур ,що дозволяють звертатися з одного додатка (DDE-клієнта) у його виконання до іншого, активному на той момент програмному (DDE-серверу). По запиту клієнта сервер обробляє так названий DDE-запит і повертає результати в тій чи іншій формі.

Технологія OLE ще один метод організації взаємодії програм у середовищі Windows і дає можливість впровадження даних одного типу (оброблюваних однією програмою) у дані, що відносяться до іншої програми. При цьому при звертанні до упровадженим даним (допустимо, по клацанню миші) автоматичний виклик того додатка, до якого вони відносяться. Класичним прикладом застосування технології OLE є впровадження малюнка чи електронної таблиці в текстовий документ.

Windows 95 — високопродуктивна, багатозадачна і багатопотоковая 32-розрядна операційна система з графічним інтерфейсом і розширеними можливостями, що працює в захищеному режимі, що підтримує 16-розряднідодатки без усякої їхньої модифікації. Це інтегроване середовище , яке забеспечує ефективний обмін текстових, графічних, звукових і між окремими програмами.

Windows 95. Функціональні можливості Windows 95 на якісному рівні перевершують усе те, що було закладено в MS DOS і Windows 3.*. Windows 95 — повномасштабна операційна система (сімейства Windows), не потребуюча MS DOS. Вона цілком сумісна програмними й апаратними засобами що використовуються в даний час. Windows 95 — перший представник нового покоління 32-бітових багатопотокових операційних систем. Перелічимо основні переваги Windows 95:

• інтегрована операційна система (операційна система, ядро якої завантажується в момент включення комп'ютера, активізує графічний інтерфейс користувача і забезпечує повну сумісність з операційною системою MS DOS):

• витісняє багатозадачність (властивість операційної системи самостійно в залежності від внутрішньої ситуації передавати чи забирати керування у того чи іншого додатка, що не дозволяє одному додатку зайняти всі апаратні ресурси);

• багатопоточність (властивість операційної системи виконувати операції одночасно над потоками декількох 32-бітових додатків. Windows 95 використовує технологію Plug and Play, спрощуючи роботу з комп’ютером за рахунок наступних сервісних функцій :

• допомога при розпізнаванні пристроїв для їхньої установки і настроювання;

• динамічної зміни стану системи й автоматичного повідомлення про це програмних додатків;

• інтеграції драйверів пристроїв, системних компонентів і користувацького інтерфейсу.

Windows 95 забезпечує динамічні зміни конфігурації системи, побудованій на базі технології Plug and Play. Технологія Plug and Play, закладена в Windows 95, дозволяє працювати з пристроями , що не підкоряються специфікації Plug and Play, спрощуючи їхнє настроювання і керування устаткуванням. Для коректного користування системними ресурсами комп'ютера Windows 95 відслідковує всі пристрої і виділені їм ресурси. Диспетчер пристроїв дозволяє одержати інформацію про всі знайдені системою пристрої і змінити при необхідності їхню конфігурацію. Windows 95 — високоефективна платформа для мультимедіа, містить у собі лазерний програвач чи CD-плейер; забезпечує підтримку відеодисків івідеомагнітофонів тощо.. Крім цього Windows 95 має особливі можливості,призначені для користувачів з обмеженими можливостями (недоліки слуху, зору тощо):

• можливості маштабування елементів інтерфейсу;

• «залипаючі» клавіші;
• режим MouseKeys (усі дії з мишею виконуються через клавіатуру);
• візуальне дублювання звуків системи.

Нарешті, у Windows 95 замість розрізнених INI-файлів для збереження конфігурації як самої операційної системи, так і інших , які використовуються в її рамках, програма стала використовуватися як інформаційна база, що одержала назву*системного реєстру* (Registry). Дане рішення дозволило централізувати і жорстко упорядкувати процес керування з боку операційної системинастроюваннями встановленого програмного забезпечення.

Windows 98. У порівнянні з Windows 95, Windows 98 включає засоби, що дозволяють комп'ютеру працювати швидше без додавання нового обладнання. До складу Windows 98 входять ряд програм, спільне застосування яких підвищує виробничість комп'ютера. У Windows 98 надійність комп'ютера підвищується за рахунок застосування нових майстрів (майстер обслуговування дозволяє швидше виконувати програми, перевіряти твердий диск на наявність помилок і звільняти місце на диску), службових програм і ресурсів, що забезпечують безперебійну роботу системи:

1.Перевірка системних файлів дозволяє відслідковувати стан найбільш важливих файлів, що забезпечують роботу комп'ютера. Якщо ці файли пошкоджені чи переміщені, програма перевірки їх відновлює.

2.Перевірка диска запускається автоматично після невірного вимикання операційної системи. Програма перевірки диска виявляє найбільш ймовірні ушкодження файлів і папок і виконує виправлення помилок. Крім того, користувач має можливість виконати перевірку диска в будь-який час.

3.Новий Web-вузол ресурсів Microsoft Windows Update автоматизує процес драйверів і системних файлів і забезпечує новітні можливості технічної підтримки. На вузлі Windows Update можна виконати пошук наявних на комп'ютері драйверів пристроїв і системних програм, зрівняти його результати з основною базою даних у Web і після цього одержати рекомендації і установити відновлення, що підходять саме для даного комп'ютера. Засіб видалення установки дозволяє повернутися до попередніх драйверів чи системних програм.

4.Перевірка реєстру є системною програмою, що дозволяє виявляти й усувати помилки в реєстрі. При кожнім запуску комп'ютера програма перевірки реєстру автоматично перевіряє реєстр на наявність непогодженості структури даних. Крім того, програма перевірки реєстру щодня виконує резервування реєстру. Якщо виявляються серйозні помилки в реєстрі, реєстр

можна відновити по резервній копії. Програма реєстру підтримує до п'яти стиснутих архівних копій реєстру, при яких комп'ютер успішно запускався. Якщо архів не вдається знайти, програма перевірки реєстру виправляє помилки реєстру.

5.Програма установки автоматично запускає перевірку реєстру при кожнім відновленні операційної системи комп'ютера. При установці Windows 98 програмаперевірки реєстру виправляє більшість помилок у реєстрі, навіть ті, про які було невідомо користувачу.

6.Програма архівації надає розширені можливості архівації і відновлення даних, у тому числі підтримку великого числа нагромаджувачів на магнітній стрічці і найсучаснішому устаткуванні. Користувачу все простіше зберігати важливі дані. Файли з твердого диска можна резервувати на гнучких дисках, на магнітній стрічці чи на іншому комп'ютері в мережі. Якщо вихідні файли ушкоджені чи загублені, їх можна відновити з архіву.

У Windows 98 є спеціальні можливості (такі як залипання клавіш, субтитри і керування покажчиком із клавіатури), що допомагають цілком використовувати комп'ютер користувачам з фізичними недоліками. Деякі з цих засобів, такі як керування покажчиком із клавіатури, можуть зацікавити і більш широке коло користувачів.

У Windows 98 за допомогою додатку Internet Explorer можна виконувати ряд функцій доступними з робочого столу Windows: можливість пошуку в Web із будь-якого місця на комп'ютері, канали Web-вузлів на робочому столі, підписка на обрані вузли, настроювана панель посилань, панелі оглядача, а також контролер вмісту і зони безпечності при пошуку Web.

Наприклад :• автоматичне доповнення раніше викликаних адрес Web у міру їхнього введення;• поліпшені списки Web-вузлів, що часто відвідуються;

• поліпшений журнал і можливості відстеження Web-вузлів, що часто відвідуються;

• підтримка всіх основних стандартів Інтернету, у тому числі ActiveX, Java тощо;

• висока продуктивність динамічної мови HTML, що дозволяє зробити Web-сторінки більш багатими і цікавими;

• новий майстер підключення до Інтернету допомагає зареєструватися для доступа до Інтернету й автоматично виконує кроки по настроюванню програмного забезпечення, необхідні для доступу до Інтернету.

Windows 2000. Перелічимо головні з нових технологічних рішень, реалізованих у Windows 2000:

1. Захист даних засобами Kerberos. Kerberos — це розроблений у Масачусецьким технологічним інстетутом (MIT) протокол аутентифікації, що забезпечує передачу даних по незахищених мережах. Саме він зі службовихкаталогів Active Directory (AP) складає найважливішу особливість Windows 2000, що відрізняє цю систему від Windows NT 4.0. Kerberos має переваг у порівнянні з NT LAN Manager (NTLM), протоколом аутентифікаціїWindows NT 4.0. Одне з них полягає в тому, що Kerberos забеспечує створення транзитивних довірчих відносин, тоді як NT 4.0 дозволяє формувати лише нетранзитивні.Інфраструктура відкритих ключів. Операційна система Windows 2000 оснащена засобами захисту інформації на базі інфраструктури відкритих ключів (public key infrastructure, PKI). Ця інфраструктура являє систему цифрових сертифікатів і організацій, що засвідчують сертифікати (Certificate Authorities, CAs). Так само, як і протокол Kerberos, вона дає можливість обом учасникам транзакції перевіряти, чи дійсно партнер є тим, за кого себе видає, а також шифрувати транзакції. Системазахисту даних на базі PKI найкраще підходить для використання в середовищі Інтернет і тому являє собою корисний засіб для організації електронної комерції між підприємствами. Поштове відомство (U.S. Postal Service) використовує PKI для надання клієнтам цілодобового доступу по каналах Інтернету до інформації про поштові витрати. Інфраструктура відкритих ключів ще не вступила в пору зрілості, але включаючи її в систему Windows 2000 дає підстави думати, що рівні безпеки і шифрування даних стануть невід'ємною частиною повсякденної практики, а це, у свою чергу, відкриє перед електронною торгівлею нові обрії. Windows 2000 оснащується рядом засобів, що призначені для користувачів малих офісів. До складу системи входить цілком якісний маршрутизатор, що володіє інтерфейсом підключення по запиту (demand-dial interface), значно полегшує процес установки з'єднання з Інтернет-провайдером. За допомогою таких засобів, як реалізований у Windows 2000 Server і у Windows 2000 Professional модуль Internet Connection Sharing, доступ до наданого єдиному системою виходу в Інтернет одержують відразу декілька користувачів мережі. У даному модулі реалізована технологія трансляції мережних адрес Network Address Translation (NAT), що дає можливість комп’ютерам, об'єднаним у локальну Windows-мережу, виходити через маршрутизатор в Інтернет, використовуючи єдину IP-адресу, наданий.У підсумку безліч користувачів локальної мережі є, з погляду провайдера, одним (але дуже активним) клієнтом.

4. Windows 2000 оснащена удосконаленими засобами симетричної

багатопроцесорної обробки.

5Убудовані засоби вилученого доступу.

6. Нові функції керування збереженням даних (керування квотами дискового простору, ієрархічна система збереження даних, динамічне управління томами, а також шляхом внесення змін у файлову систему NTFS).

Додаткові можливості для мобільних користувачів. Реалізована в Windows 2000 функція роботи з файлами в автономному режимі (offline files feature) дозволяє відбирати мережні файли і папки для подальшої роботи з ними без підключення до мережі. Потім ОС створює локальний кэш і зберігає в ньому ці файли і папки, причому вони синхронізуються з оригіналами у фоновому режимі . Крім того, у Windows 2000 передбачена функція переходу в «сплячий» режим ,перед відключенням живлення записуюча в спеціальний файл всі дані фізичної пам'яті, що забезпечить тим самим збереження усіх даних про стан системи. В режимі чекання система практично не

енергії; перехід у цей стан займає порядку 15с., а повернення до робочого режиму - 45 с.(для системи з процесором Pentium I I 366 МГц).

8. Розподілене адміністративне керування операційним середовищем. Windows 2000 оснащена цілим рядом засобів керування клієнтами і серверами, що дозволяють знизити загальну вартість експлуатації операційної системи.

Windows XP (кодова назва при розробці — *Whistler*; внутрішня версія — *Windows NT 5.1*) — операційна система сімейства Windows NT від компанії Microsoft. Вона ула випущена 25 жовтня 2001 року і є розвитком Windows 2000 Professional. Назва **XP** походить від англ. *experience* (досвід, враження, від прикметника професійний). Назва увійшла до практики використання, як професійна версія. На відміну від попередньої системи Windows 2000, яка поставлялася як в серверному, так і в клієнтському варіантах, Windows XP є виключно клієнтською системою. Її серверним варіантом є випущена пізніше система Windows Server 2003. Windows XP і Windows Server 2003 побудовані на основі одного і того ж ядра операційної системи, в результаті їхній розвиток і оновлення йде більш-менш паралельно.

Windows XP випускається в багатьох варіантах:

1.Windows XP Professional Edition була розроблена для підприємств і підприємців і містить такі функції, як віддалений доступ до робочого столу комп'ютера, шифрування файлів (при допомозі Encrypting File System),

центральне управління правами доступу і підтримка багатопроцесорних систем.

2.Windows XP Home Edition — система для домашнього застосування. Випускається як недорога «урізана» версія Professional Editon, але базується на тому ж ядрі і за допомогою деяких прийомів дозволяє провести оновлення до майже повноцінної версії Professional Edition.

3.Windows XP Tablet PC Edition базується на Professional Edition і містить спеціальні застосування, оптимізовані для введення даних стілусом на планшетних персональних комп'ютерах. Найважливішою властивістю є чудове розуміння текстів, написаних від руки і адаптація графічного інтерфейсу до поворотів дисплея. Ця версія продається тільки разом з відповідним комп'ютером.

4.Windows XP Media Center Edition базується на Professional Edition і містить спеціальні мультимедійні застосунки. Комп'ютер, як правило, оснащений ТБ-картою і пультом дистанційного керування (ПДК). Найважливішою властивістю є можливість підключення до телевізора і управління комп'ютером через ПДК завдяки спрощеній системі управління Windows. Ця система містить також функції для прийому УКВ-радіо.

5.Windows XP Embedded базується на Professional Edition і призначена для управління вбудованою системою різних пристроїв: банкоматів, медичних приладів, касових терміналів, ігрових автоматів, VOIP- компонентів тощо

6.Windows XP Professional x64 Edition — спеціальна 64-розрядна версія, розроблена для процесорів з технологією AMD64 Opteron і Athlon 64 від фірми AMD і процесорів з технологією EM64T від фірми Intel. Ця система не підтримує процесори інших виробників, а також не працює з процесором Intel Itanium. Хоча перші 64-розрядні процесори з'явилися в 2003 році, Windows XP Professional x64 Edition вийшла в світ тільки в квітні 2005 року. Основною гідністю системи є швидка робота з великими числами (Long Integer і Double Float). Таким чином, ця система дуже ефективна, наприклад, при виконанні обчислень, що використовують числа з плаваючою комою, необхідних в таких областях, як створення спецефектів для кінофільмів і тривимірною анімації, а також розробка технічних і наукових застосувань. Дана система підтримує *змішаний режим*, тобто одночасну роботу 32- і 64-розрядних застосувань, проте для цього всі драйвери повинні бути в 64-розрядному виконанні. Це означає, що більшість 32-розрядних застосувань можуть працювати і в цій системі. Виняток становлять лише ті застосування, які сильно залежать від апаратного забезпечення комп'ютера, наприклад, антивіруси і дефрагментатори.

7.Windows XP 64-bit Edition — це видання розроблялося спеціально для робочих станцій з архітектурою IA-64 і мікропроцесорами Itanium. Це видання Windows XP не розвивається з 2005 року, після того, як HP припинив розробку робочих станцій з мікропроцесорами Itanium. Підтримка цієї архітектури залишилася в серверних версіях операційної системи Windows.

8.Windows XP Edition N — система без Windows Media Player і інших мультимедіа-застосунків. Ці версії створені під тиском Європейської Антимонопольної Комісії. За бажанням користувач може безкоштовно завантажити всі бракуючі застосування веб-сайту Microsoft. Існує як в Home, так і в Professional варіантах.

9.Windows XP Starter Edition — сильно функціонально обмежена версія для країн, що розвиваються і фінансово слабких регіонів. У цій версії можлива одночасна робота тільки 3 застосувань, і кожне застосування може створити не більше 3 вікон. У системі повністю відсутні мережеві функції, не підтримується висока роздільна здатність, а також не допускається використання більше 256 мегабайт оперативній пам'яті або жорсткого диска об'ємом більше 80 гігабайт. Система може працювати на процесорах рівня Intel Celeron або AMD Duron.

10.Windows Fundamentals for Legacy PCs — Урізана Версія *Microsoft Windows XP Embedded Service Pack 2* призначена для застарілих комп'ютерів.

Windows XP аналізує продуктивність системи з певними візуальними ефектами і залежно від цього активує їх чи ні, враховуючи можливе падіння або зростання продуктивності. Користувачі також можуть змінювати дані параметри, використовуючи діалогові вікна настройки, при цьому можна або гнучко вибрати активність тих або інших візуальних ефектів, або віддати це на управління системі або ж вибрати максимальну продуктивність або кращий вид графічного інтерфейсу. У Windows XP з'явилася можливість використовувати «Visual Styles», що дозволяє змінити графічний інтерфейс користувача. Luna — новий стиль графічного інтерфейсу, що входить в постачання XP і є інтерфейсом за умовчанням для ПК, що мають більше 64 мегабайт RAM. Можливо використовувати і інші «Visual Styles», але вони повинні бути підписані цифровим підписом Microsoft (оскільки мають важливе значення у функціонуванні системи). Для Windows XP було створено більше 100 «ікон» компанією The Iconfactory.

Windows XP також має інтерфейс командного рядка (CLI, «консоль»), cmd.exe, для управління системою командами з консолі або запуску сценаріїв, званих «командними файлами» (з розширеннями cmd),

заснованими на «пакетних» (batch) файлах MS-DOS. Синтаксис Windows XP CLI не дуже добре задокументований у вбудованій системі допомоги. Докладнішу загальну інформацію можна отримати, набравши в командному рядку «help» для отримання загальних відомостей про доступні команди і «ім'я команди /?». Інтерфейс командного рядка доступний як у вигляді вікна, так і в повноекранному вигляді (перемикання між ними здійснюється натисненням Alt+Enter), вигляд, що віддається перевага, можна вказати у відповідному діалозі настройки, разом з такими параметрами, як розмір і тип шрифтів і т. д. При роботі в даному режимі користувач може викликати попередні команди (так, клавіша «вгору» повертає попередню команду), використовувати автодоповнення імен файлів і каталогів, а також команд. Багато дій з управління операційною системою можна виконати, використовуючи інтерфейс CLI.

Основними відмінностями Windows XP в порівнянні з Windows 2000 є:

1..Нове оформления графичнооого інтерфейса, включаючи більш округлі форми , а також додаткові функциональні можливості (такі як представлення папки у вигляді слайд-шоу в проводнику Windows).

2. Підтримка методу згладжування тексту ClearType, що покращує відображення тексиу рідинно-кристалевих дисплеях (по замовчуванню відключено).

3. Можливість швидкого переключення користувачів, що дозволяє тимчасово перервати роботу одного користувача і виконати вхід в систему під ім’ям іншого користувача, залишаючи при цьому включеними додатки, які завантажені першим користувачем.

4. Функція *«відлаленого помічника»*, яка дозволяє досвідченим користувачам підключатися до комп’ютера з системою Windows XP по мережі. При цьму користувач, шо допомагає може бачити вміст екрану монітора, вести діалог і (з дозволу відаленого користувача) керувати роботою комп’ютера.

5. Програма *відновлення системи*, що призначена для повернення системи у визначений попередній стан, а також покращеня інших способів відновлення системи. Наприклад при завантаженні останньої вдалої конфігурації завантажується і попередній набір драйверів, що дозволяє в багатьх випадках легко відновити систему при проблемах, що виникають в результаті встановлення драйверів .

6. Покращена сумісність зі старими програмами і комп’ютерними іграми . Специальний *майстер сумісності дозволяє* відновити для окремої

програми поведінку однієї з попередніх версій ОС (починаючи з Windows 95).

7. Можливість віддаленого доступу до рабочої станції завдяки включеної в систему мініатюрного сервера терминалів.

8. Більш розвинені функції управління системою з командного.

Підтримка проводником Windows цифрових фотоформатів і аудіофайлів

9. Windows XP включає технології, що розраболені фірмою Roxio, які дозволяють виконувати прямий запис CD з провідника, не встановлюючи додаткового ПЗ , а робота з компакт-дисками стає подібною до работи з дискетам, флешами чи жорсткими дисками.

Windows Vista. Для позначення «Windows Vista» іноді використовують абревіатуру **«WINVI»,** яка об'єднує назву «Vista» і номер версії, записаний римськими цифрами. Windows Vista, як і Windows XP, виключно клієнтська система. 30 листопада 2006 року Microsoft офіційно випустила Windows Vista і Office 2007 для корпоративних клієнтів. 30 січня 2007 року почалися продажі системи для звичайних користувачів. Назва **«Vista»** була оголошена 22 липня 2005 року. У перекладі українською **«vista»** означає «перспектива» (огляду) або «краєвид» , і є запозиченням з іспанської, де «vista» — зір, бачення. Первинна назва була визначена і схвалена співробітниками *Microsoft*, а потім протестована в декількох регіонах світу методом фокус-груп. За заявами компанії в Windows Vista була оновлена підсистема управління пам'яттю і введенням-виведенням. Новою функціональністю також є «Гібридний сплячий режим», при використанні якого вміст оперативної пам'яті додатково записується на HDD, але і з пам'яті також не витирається. В результаті якщо подача енергії не припинялася, то комп'ютер відновлює свою роботу користуючись інформацією з RAM.

З 28 липня 2005 року, розробникам і IT-професіоналами була розіслана перша бета-версія. У неї були включені всі розроблені на той день технічні можливості і наочно представлені основи нової архітектури системи. Перша бета-версія була випущена для того, щоб у IT-аудиторії склалося перше враження про нову операційну систему і щоб виявити помилки в новій системі ще до її офіційного випуску. Згідно з результатами першого етапу бета-тестування були доопрацьовані призначені для користувача функції системи, які потім були представлені в другій бета-версії. Остаточна версія Windows Vista представлена у варіантах для 64- і для 32-розрядних процесорів. Windows Vista має також новий логотип. На думку дизайнерів компанії, цей логотип ілюструє зміни

в призначеному для користувача інтерфейсі нової операційної системи (який через зовнішний вигляд називають «скляним» — «Aero Glass»).

Windows 7 (кодова назва під час розробки — *Vienna*) — назва найновішої версії операційної системи Windows, яка вийшла 22 жовтня 2009 року, менше, ніж через три роки після випуску попередньої операційної системи комерційно невдалої Windows Vista. Розробка Windows 7 під різними кодовими назвами велась з 2000 року. Над нею працювали 2500 осіб. Windows 7, кодове найменування нової операційної системи Microsoft, стане її остаточною назвою. У Microsoft пояснюють вибір назви його простотою, відзначаючи, що раніше Microsoft вже перепробувала декілька способів найменування своєї лінійки операційних систем. Windows 7 буде п'ятою схемою найменування. Крім того, це сьома версія Windows, якщо вважати за внутрішніми версіями.

Усього Microsoft пропонує 6 різних варіантів постачання, проте через роздрібну мережу поширюються переважно версії *Home Premium* і *Professional*. Інші варіанти поширюються обмежено, наприклад, тільки для корпоративних клієнтів або тільки в країнах, що розвиваються. Всі варіанти Windows 7, крім найбільш спрощених, працюють як на 32-бітній, так і на 64-бітній платформах. Для приватних користувачів передбачається спеціальна сімейна версія: умови ліцензії дозволяють встановити її не на одному, а на двох або трьох комп'ютерах.

Початкова редакція (Windows 7 Starter) поширюється виключно з новими комп'ютерами, вона не включає функціональної частини для програвання H.264, AAC, Mpeg-2. Домашня базова — призначена виключно для випуску в країнах, що розвиваються, в ній немає інтерфейсу Windows Aero з функціями Peek і Shake і передперегляду в панелі завдань, загального доступу до підключення в інтернет і деяких інших функцій. Всі 32-бітові версії підтримують до 4 Гб ОЗУ (фізичний максимум — у 32-біти можна вмістити лише 4 Гб даних). 64-бітові редакції підтримує від 8 Гб (Домашня базова) до 192 Гб пам'яті у всіх останніх редакціях, за винятком Домашньої розширеної, в ній обмеження — 16 Гб.

Найвідчутнішим нововведенням розробник називає широке застосування інтерфейсу вводу, що управляється дотиком (touchscreen), в варіанті реагування на кілька дотиків водночас (multi-touch). Істотно перероблений інтерфейс системи, панель завдань і меню «Пуск». Тепер назви програм і заголовки відкритих документів в панелі завдань не відображаються — тут немає ніякого тексту, а тільки іконки (icon). Дістати доступ до основних функцій спрощено — досить натиснути на відповідну іконку в панелі завдань правою кнопкою миші. Наприклад, у випадку з

Windows Media Player користувач може відкрити список відтворення, а у випадку з Internet Explorer — проглянути зменшені зображення відкритих веб-сторінок і перемкнутися між ними. При натисненні мишкою на вільне місце на робочому столі, всі відкриті вікна стають прозорими — це допомагає дістатися до папок (віджетів), які в Windows 7 можуть розташовуватися довільно на робочому столі, а не в його певній частині, як в Vista. При перетягуванні вікна воно збільшується — коли користувач утримує його мишкою, і зменшується, коли відпускає в потрібному місці — як в Mac OS. Якщо відкрите вікно підвести до краю робочого столу, воно зменшиться до 50% від номінального розміру — це зручно для організації вікон. Покращено виведення зображення на мультиекранні системи.

Windows 7 потрібно менше часу на перехід в "сплячий режим" і вихід з нього, а також на відновлення зв'язку з бездротовими мережами Wi-Fi. Установка і підготовка до роботи запам'ятоувальних та інших пристроїв, що підключаються через роз'єм USB «займає лічені секунди», пошук інформації на комп'ютері і сортування його результатів відбуваються також значно швидше.

Для перегляду веб-сторінок в Windows 7 передбачена восьма версія браузера Internet Explorer. До нього, крім прискорених алгоритмів роботи, увійшов новий сервіс, що сповіщає про зміни на зазначеному раніше веб-сайті.

Так само, як в Windows Vista, на робочому столі Windows 7 можна розмістити *віджети* — невеликі зображення, що показують час, картинки, які транслюють текстові новини, що програють музичні або відеофайли і так далі.

Для любителів комп'ютерних ігор у новинці передбачена нова версія Games Explorer, яка здатна завантажувати новини та оновлення до вже встановленим іграм, а функція Play To допоможе відтворити обрану музику на сумісних комп'ютерах. Також в систему додано розпізнавання почерку і покращена взаємодія з багатоядерними процесорами. Контроль облікових записів менш нав'язлива, а *Контрольну панель* можна буде викликати з області повідомлень панелі завдань. Перероблений калькулятор, а в *WordPad* і *Microsoft Paint* може з'явитися інтерфейс, звичний для користувачів Microsoft Office 2007. Крім того, в Windows 7 включена нова версія Windows Media Player, а також впроваджено нову вдосконалену технологію Natural ClearType, яка має зробити шрифти ще гладкішими. До ОС також вбудовано близько 120 фонових малюнків, унікальних для кожної країни і мовної версії. Так, російська версія

включає тему «Росія» з шістьма унікальними шпалерами високої роздільної здатності. Всі версії включають 50 нових шрифтів. У існуючих шрифтах пророблена робота над коректним відображенням всіх символів. Windows 7 — перша версія Windows, яка включає більше шрифтів для відображення нелатинських символів, ніж для відображення латинських. Для зручності управління панель управління шрифтами також піддалася поліпшенню. За умовчанням, в ній відображуватимуться лише ті шрифти, розкладка для яких встановлена в системі. Реалізована підтримка Unicode 5.1. Панель пошуку Instant Search тепер розпізнає більше мов. Наприклад, розпізнаються російські відмінки, відміни, рід, однина і множина.

Windows 7 підтримує три варіанти оформлення, призначеного для інтерфейсу користування:

1. Windows Aero — це оригінальний стиль оформлення з прозорими багатоколірними рамками вікон, вживаний поза умовчуванням для комп'ютерів з більш ніж 1 Гб ОЗУ. Доступний на Windows 7 Home Premium, Windows 7 Professional і старших редакціях.

***2.Windows 7* — спрощений стиль** — Windows Aero з деякими відключеними можливостями (наприклад, прозорість вікон і Windows Flip 3d). Вимоги до системи — такі ж, як і в Windows Aero. Доступний на всіх редакціях Windows 7, і є основним в редакції Windows 7 Starter. Цей стиль так само застосовується при запуску в режимі сумісності.

3. Класичний — мінімальні вимоги до системи, оформлення вікон в стилі «класичної» теми Windows XP. Доступні різні колірні схеми, зокрема, подібні до схем Windows 98; користувач може створювати свої колірні схеми.

Windows 7 була випущена 22 жовтня 2009. До кінця січня 2010 р. було продано понад 60 мільйонів копій операційної системи. Наприкінці квітня 2010 р. число проданих ліцензій перевищила 100 мільйонів. У червні 2010 р. Microsoft продала 150 мільйонів ліцензій Windows 7. У середньому щосекунди продається 7 копій системи, що є найбільшим показником за всю історію корпорації.

За перший рік з дня початку офіційних продаж було продано 240 мільйонів ліцензій на Windows 7. Станом на кінець січня 2011 було продано 300 мільйонів ліцензійних копій.

Слід зауважити, що підчас тестування і після офіційного випуску системи Windows 7 надійшли численні скарги користувачів ноутбуків, що заряд справних акумуляторів з нормальним часом автономної роботи більше двох годин «виснажується» за півгодини і навіть скоріше. Також є скарги на появу повідомлень про помилку з рекомендацією замінити акумулятор.

Критика ґрунтується на таких аргументах:

1. Windows 7 має такі ж високі системні вимоги, як і Vista. Так, рекомендовані Microsoft апаратні вимоги для нормальної роботи Windows 7 і Vista складають 1 ГБ ОЗУ і 16 ГБ вільного дискового простору, тоді як для Windows XP вони складають 128 МБ ОЗУ і 1,5 ГБ на диску.
2. Результати незалежних тестів показують, що програмні продукти на Windows 7 працюють ненабагато швидше, ніж на Vista, і повільніше, ніж на старій Windows XP..
3. Низька продуктивність системи, показана у ряді незалежних тестів, пов'язана з ускладненням ОС і, отже, інтенсивнішим використанням обчислювального ресурсу, що, у свою чергу, може виражатися у збільшенні енергоспоживання і зниженні часу роботи батарей нетбуків до 30% в порівнянні з Windows XP.
4. В Windows 7 використовуються технології, котрі дозволяють Microsoft вторгатися в особисте життя користувача.

Windows 8 — була випущена фірмою Microsoft літом 2012 року таими системними вимогами:

1) один гігагерц (ГГц) або більш потужний 32-розрядний (x86) або 64-розрядний (x64) процесор
2) один гігабайт (ГБ) оперативної пам'яті (32-розрядна) або 2 ГБ оперативної пам'яті (64-розрядна версія);
3) 16 ГБ вільного місця на жорсткому диску (32-біт) або 20 ГБ (64-розрядна версія);
4) DirectX 9 графічного пристрою з WDDM 1.0 або більш новіші версії;
5) Для того, щоб скористатися перевагами сенсорного введення в новій операційній системі, потрібно екран, що підтримує мультитач.
6) Для того щоб запустити Microsoft Windows 8, нові програми стилі Metro, розширення екрану повинно бути більше ніж 1024 x 768.

Перші відомості про Windows 8, почали з'являтися у квітні 2009 року, коли Microsoft розмістила у відділі вакансій пропозицію для розробників і тестерів брати участь у розробці Windows 8. «Також ми скоро почнемо розробляти істотні вдосконалення для Windows 8, в які будуть включені інноваційні особливості, які зроблять революцію в доступі до файлів на віддалених машинах», — було написано в рекламі на посаду провідного інженера з розробки й тестування ПЗ.

У зв'язку з даними першими непрямими відомостями почали активно з'являтися різні чутки і домисли щодо Windows 8. Наприклад, висувалися припущення про те, що Windows 8 буде поставлятися тільки в 64-бітній редакції, буде мати повністю інший інтерфейс без меню «Пуск», що вихід

в Інтернет буде можливий прямо з робочого столу і буде присутня недеревоподібна файлова система. Також виникали припущення про повне злиття Windows 8 з концепцією, закладеною у MinWin. Ще однією групою чуток було присвоєння Windows 8 сторонніх кодових імен. Найбільш часто Windows 8 називали кодовим ім'ям «Midori», тоді як «Midori» є повністю окремою науково-дослідною ОС. Крім «Midori», Windows 8 привласнювали імена «Mojave», «Orient» і деякі інші. Крім чуток, в Інтернеті час від часу з'являлися сфальсифіковані скріншоти Windows 8, на яких були зображені вікна з системною інформацією або інші вікна або навіть робочі столи. Наприклад, в кінці травня 2009 року в Інтернет потрапив черговий підроблений скріншот «Windows 8 Alpha Build 7504», в якому були знайдені сліди обробки програмою «Photoshop».

Windows 8 Відповідно до офіційно непідтверджених відомостей, головною особливістю Windows 8 буде підтримка 128-бітної (128-розрядної) архітектури. У жовтні 2009 року з'явилася неофіційна інформація про те, що розробники Windows 8 в цей момент реалізують впровадження в IA-128 повної бінарної сумісності з існуючими 64-бітними інструкціями. Встановлюються угоди з провідними виробниками процесорів та інших апаратних засобів, включаючи компанії Intel, AMD, Hewlett-Packard і IBM. Виходячи з непідтвердженої інформації, в Windows 8 буде приділено увагу системі розпізнавання голосу і голосовому управлінню. Серед інших можливих нововведень передбачається присутність поліпшеної роботи віртуалізації, що дозволяє відобразити всю операційну систему; тісної інтеграції з мобільною версією операційної системи, підтримка розпізнавання жестів. Передбачається, що Windows 8 матиме вдосконалену систему ,,глибокого сну і відновлення", а також нові функції забезпечення безпеки, включаючи модифікацію PatchGuard.

У Windows 8 планується істотно скоротити час завантаження і вимикання в порівнянні з попередніми ОС сімейства Microsoft Windows. Пильна увага буде приділена енергоспоживанню та енергоефективності. Заявлена підтримка інтерфейсу USB 3.0, з'єднань Bluetooth 3.0, а також бездротових і стереоскопічних дисплеїв. Одним з нововведень буде система розпізнавання особи користувачів за допомогою веб-камери. Ця система зможе самостійно переводити ПК в різні режими енергоспоживання залежно від того, чи знаходиться користувач перед веб-камерою чи ні. У певному вигляді подібна концепція реалізована в ігровому контролері Kinect для ігрової консолі Xbox 360. У презентації

багато уваги приділено конкурентам — компанії Apple та її продуктам і сервісам, Google Chrome OS, Android, WebOS.

Так, Microsoft збирається запустити службу Windows Store, що повторює багато функцій Apple App Store для iPhone. Ця служба буде інтегрована у Windows 8. Необхідність у збільшенні швидкості завантаження і вимикання продиктована Google Chrome OS. У Windows 8 будуть присутні нові інструменти для забезпечення безпеки, які спростять діагностику та усунення несправностей. Згадується функція «скидання системи», яка дозволить перевстановити Windows 8, зберігаючи при цьому всі призначені для користувача файли. Також Windows 8 буде підтримувати акселерометри, шифрування жорсткого диска, підтримку GPS в нетбуках і ноутбуках, використання WWAN-модулів

Microsoft вирішила розділити розробку Windows 8 на 3 основних фази:

1) Планування (від формування до бачення): охоплення ситуації в цілому, основні думки, потім сценарії і визначення списку функцій;

2) Розробка (від бачення до Beta): Проектування і розробка функцій, перевизначення переліку редакцій та цінової політики, початок випуску збірок;

3) Готовність (від Beta до GA +90): Завершення розробки функціональності і виправлення помилок, вимірювання і відстеження готовності та орієнтація на створення комфортної роботи в зв'язці Dell + Windows. Апаратні вимоги для Windows 8 ще не визначені, але вони безумовно мають зрости. Згідно з ним, будуть потрібні потужності в 2 рази вище, ніж для Windows 7.

Зміни, що торкнуться насамперед таких аспектів:

1) Більш широкий доступ до даних, їх конфігурації, для обміну та використання;
2) Можливість для інших компаній, налаштувати ОС під свої вимоги;
3) Природна підтримка USB 3.0 і Bluetooth 3.0;
4) Використання більш складних моделей, з можливістю використання системи для аутентифікації користувача, який працює з ПК;
5) Оптимізація для швидкого запуску, в Microsoft, очевидно, бажають майже миттєвого запуску. Windows 8 буде оптимізована для кожного режиму і в ній розробники зосередили увагу на таких речах, як продуктивність POST або відновлення після входу в режим S3. У цьому випадку метою є досягнення активації BIOS і драйверів менш ніж за секунду;

6) Використання детекторів. Таким чином, фіксуючи навколишнє освітлення, можна буде регулювати такі параметри, як яскравість, і відповідним чином адаптуватися до умов автоматично;

7) Підтримка нового мультимедійного обладнання, такого як 3D монітори і 3D телевізори, бездротове шифрування даних по всьому жорсткому диску.

8) Новий інтерфейс Windows Metro чи Immersive.

Отже, компанія Microsoft вперше показала свою нову операційну систему Windows 8 в дії, хоча якихось деталей про продукт вона тоді не розкривала.

Компанія Microsoft в рамках конференції для розробників Build провела анонс своєї нової операційної системи, Windows 8, а також показала пристрої під її управлінням.

Ідеологія спеціально адаптованих під Windows 8 додатків теж дещо змінилася, причому мова йде не тільки про інтеграцію в інтерфейс за допомогою живих панелей. Наприклад, з'явився повно-екранний режим, коли немає ніяких вікон та інших відволікаючих елементів. Чимось нагадує аналогічну функцію в OS X Lion. Програми здатні взаємодіяти один з одним і навіть з мережевими сервісами. Так, при створенні електронного листа, поштовий клієнт може взяти зображення не тільки з HDD, але з Facebook або модифікована файлова система, яка заборонить запуск файлів, що викликають Flickr.

Функції Windows 8 :

1) У Windows 8 буде використовуватися модифікована файлова система, яка заборонить запуск файлів, що викликають підозру;

2) Вбудований переглядач PDF файлів Modern Reader.

3) Інтеграція з SkyDrive та Live Mesh.

4) У новій версії інтерфейсу Aero, колір графічної оболонки буде змінюватися в залежності від поточного фонового зображення. Схожі можливості використовуються частково і в Windows 7. Наприклад, піктограми додатків на панелі завдань підсвічуються специфічним кольором, коли програмі потрібно привернути увагу користувача. Спостерігати цей ефект можна і просто навівши курсор мишки на піктограми запущеного додатка.

5) Windows 8 отримала інтерфейс користувача Metro, знайомий по Windows Phone 7. Він адаптований не тільки для сенсорного введення, а й під традиційну клавіатуру з мишею. Крім того, в будь-який момент можна переключитися на стандартний інтерфейс Windows, хоча в попередній версії Windows 8 це реалізовано у вигляді окремого додатка.

Розробку додатків з інтерфейсом Metro можна буде робити за допомогою технології WinRT 8.

6) Замість звичного жорсткого диска користувачі отримають можливість зберігати якомога більше інформації в Інтернеті, в так званій „хмарі”:

Питання для самоконтролю

1.У чому особливості операційної оболонки Windows 3.1?
2.У чому складаються основні переваги Windows 95?
3.Перелічите так називані «особливі можливості» Windows 95.
4.Порівняєте можливості Windows 95 і Windows 98,
5.Які особливості операційної системи Windows 2000 ?
6.Розшифруйте визначення операційної системи Windows XP ?
7.Дайте короткий аналіз варіантів використання Windows XP ?
8. Які основні відмінності Windows XP в порівнянні з Windows 2000 ?
9. Що означає назва операційної системи «Windows Vista» ?
10 . Які системні вимоги до операційної системи Windows Vista ?
11 Коли вперше була запропонована ОС Windows 7 ?
12.Які варіанти оформлення для інтерфейсу користувача використовують у Windows 7 ?
13. Які недоліки відмічають критики операційної системи Windows 7.
14. Які системні вимоги до Windows 8 ?
15. Які основні характеристики і особливості Windows 8?
16. Сформулюйте основні апаратнф вимоги для Windows 8 ?
17. Прелічіть функції Windows 8 ?

Розділ 7. Засоби та технологія створення алгоритмів та програм.

7.1.Поняття аглоритму, основні його властивості та способи зображення

Алгоритм – скінчений набір правил, що визначають зміст та послідовність операцій, які перетворюють вихідні дані в необхідний результат.

Для алгоритму характерні такі властивості: детермінованість (визначеність); масовість; результативність (спрямованість); дискретність.

Детермінованість, або визначеність, полягає у чіткості вказівок, що утворюють алгоритм, в їх повній зрозумілості і однозначності, тобто опис кожної дії в алгоритмі повинен усіма трактуватись однаково.

Масовість – властивість алгоритму, що зумовлює розв'язання не однієї, а множини однотипових задач.

Результативність, або **спрямованість**, полягає у тому що вказаний у алгоритмі обчислювальний процес приводить від вихідних даних до кінцевого результату. Із властивості результативності безпосередньо випливає поняття області застосування алгоритму, що визначається найбільшою областю вихідних даних у якій алгоритм є результативним.

Дискретність характеризує можливість поділу алгоритму на окремі ділянки, а етапів на окремі кроки, виконання яких зводиться до обчислення елементарних операцій.

У практиці розробки інформаційних систем управління існують різноманітні засоби представлення алгоритмів, починаючи від опису на звичайною мовою і закінчуючи формальним представленням алгоритмів у вигляді таблиць, формул, набору певних операторів.

Словесний спосіб характеризує якісний зміст етапів обчислень, що описується звичайною мовою. Відсутність неточності технологічного процесу опрацювання інформації є основним недоліком словесного способу. Разом з тим він дає можливість описувати алгоритми з довільним степенем деталізації. Словесний опис алгоритму дозволяє призначити невелику кількість умовних позначень і використовується в основному в евристичних алгоритмах, що моделюють розумну поведінку при виробленні керуючого рішення і не описується точним алгоритмом.

Формульний спосіб зображення алгоритмів полягає в тому, що інструкція про виконання обчислювального процесу в певній послідовності задається з використанням математичних символів, виразів та необхідних пояснень. Цей спосіб компактніший та наочніший за словесний, разом з тим він є суворо формалізований.

Графічний спосіб (у вигляді блок-схем) використовується при зображенні логічної структури алгоритмів за допомогою відповідних фігур, що виконують ту чи іншу обчислювальну операцію.

Існує державний стандарт умовних графічних позначень для того щоб описати схеми алгоритмів опрацювання даних.

Операторний спосіб являє собою послідовність операторів, кожний з яких відбирає певну групу елементарних операцій. Такий запис досить наочно відображує логічну структуру алгоритму та дозволяє використовувати формальні методи для його аналізу та перетворень.

При побудові операторних схем усі оператори ділять на дві групи: арифметичні та логічні.

Арифметичні оператори, або оператори обчислень, починають дії, пов’язані з обчисленням за заданими формулами. Істотною властивістю арифметичних операторів є те, що після виконання цього оператора незалежно від результату обчислень здійснюється перехід до виконання наступного за ним (правого) в операторній схемі оператора.

7.2.Організація програм з використанням підпрограм.

Якщо деяку сукупність операторів треба використати в різних місцях програми, то щоб її не повторювати кілька разів, її записують у програмі один раз і звертаються до неї, коли в цьому є потреба. Таку групу операторів називають підпрограмою.

Підпрограми в будь-якій алгоритмічній мові істотно підвищують ефективність праці програмістів. Полегшується налагодження програми, бо робота кожної підпрограми може бути перевірена окремо. Використання підпрограм зменшує загальну кількість операторів у програмі і тим самим вимагає для розміщення меншу кількість пам’яті.
Підпрограми в алгоритмічних мовах можуть бути оформлені як оператори функцій, процедури-функції та процедури підпрограм.

Принципова відмінність перелічених типів підпрограм полягає ось в чому. В операторі функції за допомогою ідентифікатора та списку формальних параметрів можна визначити значення функції за певною формулою при різних фактичних даних, тобто при виконанні одного оператора одержуємо один вихідний параметр. У процедурі-функції при виконанні сукупності операторі одержують один вихідний параметр. В процедурі-підпрограмі при виконанні сукупності операторів можна одержати кільки вихідних параметрів.

7.3.Засоби створення програм

До засобів створення програм відносяться мови і системи програмування. Основна функція всіх мов програмування, крім машинної, полягає у тому, щоб надати програмісту засоби абстрагування характеристик та особливостей апаратного забезпечення, на якому виконуватимуться програми. Системи програмування містять автоматизовані засоби розробки програм.

Спосіб написання програми, яка містить числові коди операцій і числові значення адрес комірок пам’яті називається *програмуванням у числових кодах,* а мова, якою записуються такі програми, - машинною. *Машинна мова* – «природна мова» певного комп’ютера, яка визначається під час проектування його апаратних засобів. Машинні мови важкі для людського сприйняття. Тому природним виявилось прагнення автоматизувати процес написання програм, для того щоб полегшити

працю програміста, частково поклавши його роботу на саму машину. Програмісти почали використовувати більш звичну для людини символьну форму обчислень, а перетворення програм у машині коди здійснювали за допомогою програм трансляції (від англ. translation - переклад), які називаються *асемблерами* (від англ. to assemble - збирати).

Мови програмування, у яких числове кодування команд було замінено їх символьним зображенням, називалися мовами *символічного кодування*. Нині такі мови програмування перетворилися в досить потужні засоби програмування, названі асемблерами.

Під час написання програм мовами асемблерного типу в ролі засобів програмування використовуються такі абстракції, як зміна та самовільне зображення операцій, що дає змогу програмісту позбутися проблем, пов’язаних з формою зображення чисел, кодуванням операцій і розподілом пам’яті. Тепер перераховані вище проблеми має вирішувати програма-транслятор на основі інформації, яку їй передає розроблювач програм.

Слід зазначити, що навіть проста програма, написана мовою асемблера, складається з довгої послідовності команд, за структурою близьких до машинних. Написати таку програму нелегко, до того ж потрібно знати дуже багато подробиць щодо архітектури комп'ютера (наприклад, для чого призначено ті чи інші регістри, які адреси пам'яті можна використовувати, а які — ні). Тому програмування мовою асемблера вимагає додаткових знань програмістів.

Для прискорення процесу програмування були розроблені мови програмування високого рівня, які дозволяли писати програми, за формою близькі до людської мови, та використовували загальноприйняту математичну нотацію. Перша з них з'явилася наприкінці 1950-х років і називалася ***Fortran***. Назва була скороченням від слів *FORmula TRANslation*, що у перекладі означає трансляція формул.

Одночасно з мовами високого рівня розроблялися *транслятори (компілятори)* - програмні засоби, призначені для перекладу високорівневих програм у машині. Досвід створення мов високого рівня та їх трансляторів з роками накопичувався. Зокрема, було розроблено математичні основи та технологію реалізації цих засобів. На сьогоднішній день кількість мов програмування й трансляторів вимірюється уже тисячами і продовжує зростати.

7.4. Технологія створення програми

Розглянемо процес створення програми у найбільш загальних рисах, які будуть розглянуті детальніше в інших дисциплінах напряму підготовки ”Інженерія програмного забезпечення ”.

Розробка програми починається з постановки задачі, яку пропонує замовник. Іноді аналіз і уточнення задачі дають можливість формалізувати її постановку, в результаті з'являється математично точний і однозначний опис задачі. Після уточнення постановки задачі починається *проектування програми.* Як правило, в задачі можна виділити декілька *підзадач* і описати процес їх розв'язування окремо. Відповідно й алгоритм складається зі зв'язаних та узгоджених між собою частин, які описують процес розв'язання підзадач. У початковому алгоритмі дії подано в абстрактному вигляді, далекому від того, що може виконувати комп'ютер. Алгоритм уточнюють декілька разів і надають йому вигляд, за яким легко написати програму або її частину.

Написання програми або окремих її частин прийнято називати *кодуванням,* або *розробкою.* Найчастіше програму записують однією з мов високого рівня, але іноді деякі її частини записують різними мовами. Далі програма перекладається (транслюється) на машинну мову (зазвичай частинами). Під час кодування програмісти можуть припускатися помилок. Процес виявлення й виправлення помилок називається *налагодженням* програми. Він дозволяє виявити помилки перелічених нижче типів.

1. Помилки, пов'язані з порушенням правил граматики в тексті програми, написаної мовою високого рівня. їх можна виявити у процесі трансляції, тому вони називаються *помилками часу трансляції* (compiler error).
2. Помилки, що виявляються під час виконання робочої програми. Вони можуть виникати, наприклад, в результаті переповнення розрядної сітки чи при спробі видобути квадратний корінь із від'ємного числа. Такі помилки називаються *помилками часу виконання* (run time error).
3. Помилки, що не виявляються ні під час трансляції, ні під час виконання програми. Це змістові помилки, пов'язані з некоректністю логічних умов, неправильним використанням розрахункових формул і т. ін. Їх називають *семантичними.*
4. Помилки у вихідних даних.

Налагодження - це процес багаторазового виконання програми з різними варіантами даних, які вона має обробляти. Дані спеціально добираються таким чином, щоб можна було виявити якнайбільше помилок, якщо такі існують. Ця цілеспрямована перевірка працездатності

програми називається *тестуванням.* Тестування не гарантує відсутності помилок у програмі, а лише дозволяє виявити деякі з них. Чим ретельніше проводиться тестування, тим більше помилок виявляється та виправляється.

Після налагодження програма проходить *дослідно-виробничу експлуатацію,* для якої необхідно розробити супровідні документи під назвами «Керівництво розробника програми» і «Керівництво користувача», які описують устрій та використання програми. Перший документ дає можливість виправляти помилки під час експлуатації програми та розвивати її надалі, а у другому пояснюється, як використовувати програму.

Проте може з'ясуватися, що під час проектування програми було обрано не найкращий алгоритм, через що програма виконується надто повільно або витрачає забагато ресурсів пам'яті, тобто є неефективною. До програми нерідко вносяться зміни, які вимагають повторного кодування і налагодження. Якщо у замовника з'являються нові ідеї, необхідно заново ставити задачу та проектувати програму.

З метою розробки ефективних з точки зору використаних ресурсів програм і прискорення процесу їх проектування були створені технології, тобто системи методів, які дозволяють «не робити зайвих кроків» на кожному з етапів, від аналізу задачі до налагодження програми. У 70-х роках минулого століття домінувала *технологія структурного програмування* — система правил створення програм, що характеризуються ясністю, простотою тестування та налагодження, легкістю модифікації. Починаючи з 90-х років ключовою технологією є *технологія об'єктно-орієнтованого програмування* — програмування на основі абстрактних типів даних. Застосування різних технологій потребує постійного удосконалення інструментів, які дозволяють створювати програми швидко, якісно й економічно. Такі інструменти називаються *системами програмування* і реалізують дуже важливий *принцип повторного використання коду* завдяки створенню модулів і компонентів. У процедурному програмуванні принцип повторного використання коду реалізується за допомогою процедур і функцій,.

7.5. Перетворення програми і система програмування

Розглянемо, як із програми, написаної мовою високого рівня, утворюється інша - машинна. Програму (вихідний текст) за допомогою спеціальної програми (вона називається *текстовим редактором)* найчастіше записують на диск у вигляді вихідного файлу (рис. 7.1).

Програма може складатися з кількох вихідних файлів - у великих програмах їх може нараховуватися десятки.

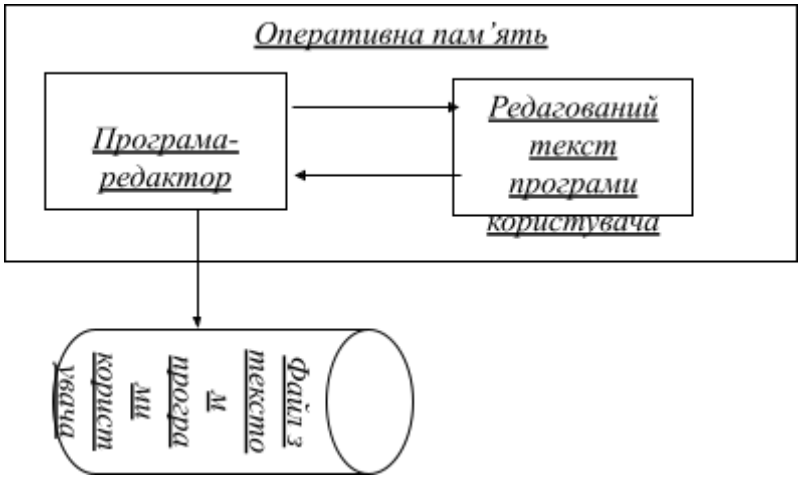


Рис. 7.1. Схема створення тексту програми

Під час роботи транслятора прочитується вихідний файл і створюється його машинний еквівалент - *об'єктний код.* Процес виконання програми-транслятора називається *трансляцією,* або *компіляцією* вихідного тексту. Як правило, об'єктний код програми містять далеко не всі необхідні команди — програма може складатися з частин або включати підпрограми з бібліотек. Об'єктний код обробляється ще однією програмою — *редактором зв'язків,* або *компонувальником,* яка «збирає» (компонує) повний код програми і записує (завантажує) його або в оперативну пам'ять, або на диск у вигляді готового до виконання файлу (рис. 7.2), який можна завантажити пізніше.

Ще один спосіб обробки вихідної програми поєднує в собі трансляцію й інтепритацію. Програма перекладається (транслюється) не в машинні команди, а в деяке проміжне зображення, що потім інтерпретується. Такий підхід реалізовано, зокрема, в мові Java, яка швидко знайшла собі багатьох прихильників серед програмістів.

Інтерпретатор на відміну від транслятора не створює машинну програму. Вхідні дані для інтерпретатора — це високорівнева програма й дані, що мають зчитуватися під час її виконання (рис.7.3). *Інтерпретація* програми полягає в тому, що дії, задані програмою, відразу виконуються.

Файл *.obj з об'єктним кодом

Файл *.exe, що виконується

Рис. 7.2. Схема створення виконуваної машинної програми

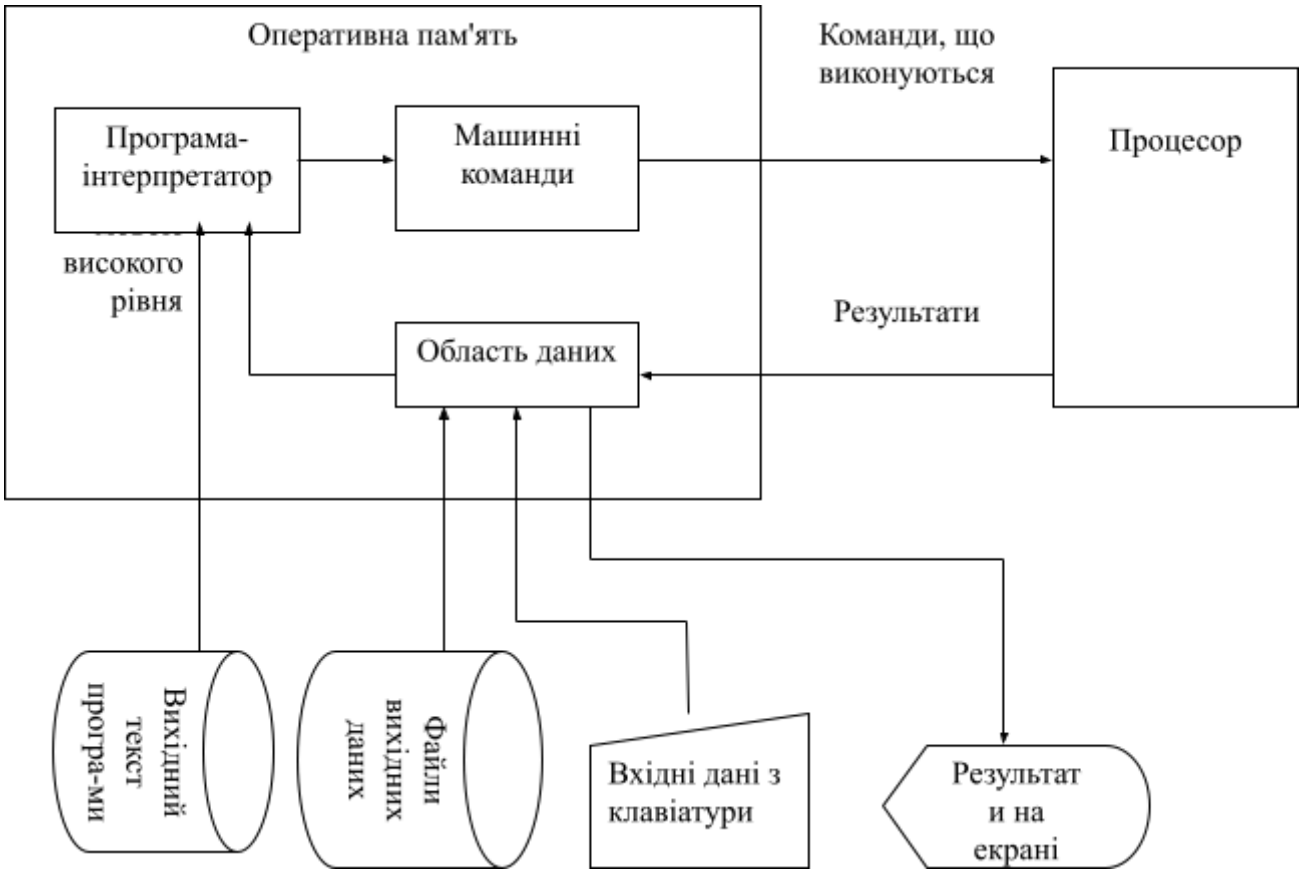


Рис 7.3. Інтерпретація високорівневої програми

Зазвичай інтерпретація вихідної програми відбувається повільніше, ніж виконання відповідної машинної програми.

Ще один спосіб обробки вихідної програми поєднує в собі трансляцію й інтепритацію. Програма перекладається (транслюється) не в машинні команди, а в деяке проміжне зображення, що потім інтерпретується. Такий підхід реалізовано, зокрема, в мові Java, яка швидко знайшла собі багатьох прихильників серед програмістів.

Інтерпретація програми здійснюється за допомогою такого інструменту, як *налагоджувач*. Він забезпечує інтерпретацію вихідної програми невеликими порціями (кроками) і дає можливість побачити результати виконання кожного кроку. Це полегшує пошук помилки (її локалізацію) у вихідній програмі.

Описані засоби (текстовий редактор, транслятор та (або) інтерпретатор, компонувальник, завантажувач і налагоджувач) разом утворюють *систему програмування,* або *інтегроване середовище розробки* (Integrated Development Environment, IDE) . Крім них до складу IDE входить бібліотека стандартних підпрограм, які можна використовувати під час створення програми.

7.6. Методологія структурного програмування.

Структурне програмування - це сукупність методів проектування та написання програм за жорсткими правилами, дотримання яких підвищує продуктивність праці програмістів, поліпшує читабельність і полегшує процес тестування програм. Витоки цієї методології сягають 60-х років XX століття, коли швидкими темпами почала зростати складність програм, а отже, постала потреба у методах подолання такої складності. Методологія структурного програмування ґрунтується на трьох методах: *низхідного проектування прграм, модульного програмування* і *методи структурування програм.*

1. Низхідне проектування програм. В основу методу *низхідного проектування* покладено *алгоритмічну декомпозицію*,згідно з якою велика задача поділяється на дрібніші підзадачі, що їх можна розв'язувати окремо. Алгоритми розв'язання підзадач розглядаються як цілісні алгоритмічні блоки, або підпрограми, іменами яких можна оперувати при розв'язанні загальної задачі. Отже, програма, що розв'язує загальну задачу, спочатку розглядається як незалежний модуль. Згодом вона поділяється на підпрограми, які декомпонуються на підмодулі наступного рівня. Процес декомпозиції триває доти, доки не будуть отримані блоки, що є достатньо малими для їх безпосереднього кодування. При цьому керуючу програму

проектують раніше, ніж реалізують її складові частини. Таким чином, програма ієрархічно структурується і розробляється шляхом послідовного уточнення на кожному рівні ієрархії. В основу цього процесу, крім принципу ієрархічності, покладено принципи абстрагування, специфікації інтерфейсів і модульності.

Абстрагування — це спрощений опис системи, в якому зосереджують увагу на певних властивостях і деталях, а на інші не зважають. Вдалою є та абстракція, що підкреслює суттєві деталі і відкидає несуттєві. Під час низхідного проектування програми на верхніх рівнях абстракції деталі реалізації приховуються, а нанижніх рівнях вони описуються конкретною мовою програмування.

Специфікація інтерфейсів — це формалізований опис входів, виходів і функцій, що мають бути реалізовані програмним модулем. Коли інтерфейс модуля специфіковано, можна спробувати в явному вигляді записати його код. Якщо це зробити неможливо, модуль поділяють на певну кількість невеликих підмодулів. Коли модуль не можна ані закодувати, ані декомпонувати, його вважають *модулем-заглушкою*, інтерфейс якого відомий, а реалізація — ні. Одним із прийомів формалізованого підходу до низхідного проектування є *метод ієрархічних діаграм*, що позначається абревіатурою HIPO (від англ. Hierar-chical Input Processing Output — діаграма входу, обробки, виходу). Згідно з цимм етодом структура всієї програми подається у вигляді дерева, в якому підпрограми зображуються вузлами, а їхні виклики — ребрами.

2. Модульне програмування. Модульне програмування — це організація програми у вигляді сукупності незалежних блоків, структура і функції яких підпорядковуються певним вимогам. Такі блоки називаються модулями. Модульне програмування дає можливість:

- простити процес тестування і налагодження програми;
- розробити програму зусиллями більше ніж одного програміста;
- локалізувати дію змін і доповнень до програми в межах одного модуля;
- скоротити термін і вартість розробки програми.

Програму можна вважати модульною, якщо її логічна і фізична структура відповідає таким умовам:

- модулі є незалежними програмними блоками, код яких логічно і фізично відокремлений від коду інших модулів;
- модулі є неподільними блоками програми, які можна використовувати по-вторно;

- розмір модуля обмежується розміром сегмента коду, який виділяється під час компіляції модуля.

Незалежність модулів є визначальним принципом модульного програмування. Це означає, що програма має бути поділена на модулі таким чином, щоб зв'язки між модулями були слабкими, а зв'язки всередині модулів — сильними. Точніше, модулі можуть вважатися незалежними, якщо вони задовольняють таким вимогам:

- кожен модуль можна замінити іншим функціонально еквівалентним модулем, що має той самий інтерфейс;
- взаємозв'язки між модулями встановлені за ієрархічним принципом;
- усі структури даних інкапсульовані в модулі, тобто доступ до даних може здійснюватися лише через процедури та функції модуля;
- модуль має одну точку входу та одну точку виходу;
- модуль повертає керування тому програмному блоку, що його викликав.

Під час поділу програми на модулі потрібно враховувати динаміку викликів процедур і функцій, а також їх розташування в оперативній пам'яті. Якщо поділ програми на модулі здійснено не раціонально, зростає кількість взаємних викликів підпрограм, розташованих у різних сегментах пам'яті, і це призводить до зниження швидкодії програми. У випадку колективної розробки модульної програми робота розподіляється так: більш досвідчені програмісти відповідають за інтерфейс модулів, а менш досвідчені — за їх реалізацію.

3.Методи структурування програм. Методи низхідного проектування та модульного програмування регламентують процес побудови архітектури програм на макрорівні. Методи структурування програм, навпаки, працюють на мікрорівні, регламентуючи процес створення програмного коду. В основу методів структурування програм покладено *теорему про структуру'вання*, що її у 1965 році довели Бом і Джакопіні. Згідно з цією теоремою будь-яку програму можна побудувати з використанням лише трьох керуючих структур: *послідовності, вибору і повторення*. Метою структурування є перетворення неструктурованої програми на еквівалентну їй структуровану, тобто таку, що складається з обмеженого набору керуючих алгоритмічних структур. Методи структурування ґрунтуються на поняттях функціонального вузла, а також на поняттях простої, елементарної і складеної програми .

Функціональним вузлом називається частина алгоритму, що має один вхід та один вихід. Прикладом функціонального вузла є окремий оператор - присвоєння, виклик процедури тощо.

Програма називається *простою*, якщо вона має один вхід, один вихід і через кожен її функціональний вузол проходить деякий шлях від входу до виходу. Проста програма може містити прості підпрограми. Вся проста програма може розглядатись як один функціональний вузол.

Елементарна програма — це проста програма, що не містить підпрограм, які складаються більше ніж з одного вузла. Зазначимо, що існує лише обмежена кількість елементарних програм, з яких основними є чотири програми, які зображені на рис. 7.4.

Якщо функціональний вузол елементарної програми замінити елементарною програмою, утвориться *складена програма*. Якщо вузол складеної програми замінити елементарною програмою, знов утвориться складена програма. В такий спосіб із будь-якої множини елементарних програм утворюється певний клас складених програм. Так, множина {послідовність, *if...then...else*} формує клас програм без циклів, а множина {*if...then...else, while...do*} приводить до утворення класу програм, які містять цикли. Отже, будь-яка підмножина множини елементарних програм є базисною множиною для певного класу складених програм. Складена програма, побудована з певної множини елементарних програм, називається *структурованою*.

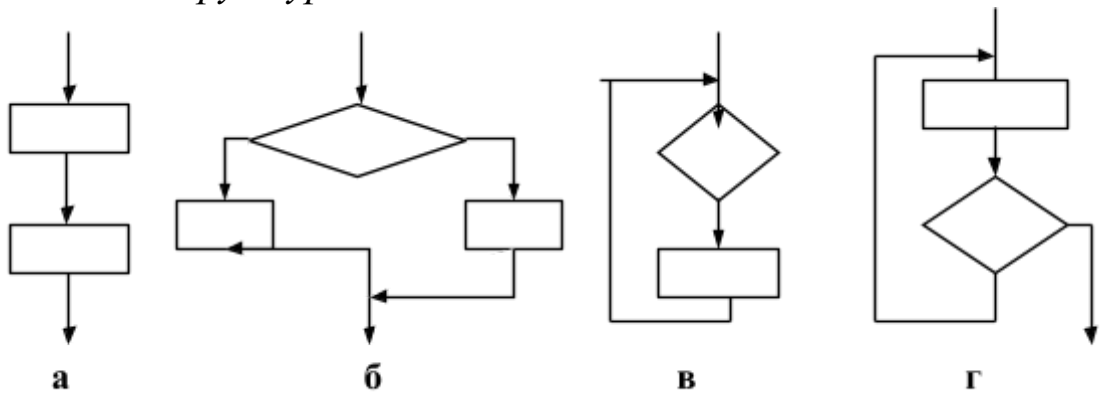


Рис. 7.4. Основні елементарні програми:лінійна (а), розгалужена (б), циклічна із передумовою (в), циклічна із післяумовою (г)

Теорема про структурування стверджує, що будь-яку просту програму шляхом покрокового перетворення можна замінити функціонально еквівалентною структурованою програмою. Згадане покрокове перетворення здійснюється за переліченими нижче правилами.

1. Правило *простоти*: створення програми слід починати із простої програми;

2. Правило *пакетування*: кожну дію можна замінити двома послідовними діями;

3. Правило *вкладання* — кожну дію можна замінити будь-якою структурою керування;

4. Правила 2 і 3 можна застосовувати у будь-якій послідовності необмежену кількість разів.

Принцип застосування правил пакетування та вкладання до простої програми продемонстровано на рис. 7.5. Для перетворення неструктурованих програм у структуровані використовуються такі методи:дублювання кодів, введення змінної стану і метод булевих ознак.

Метод дублювання кодів застосовується для перетворення неструктурованих програм, що не містять циклів. Для отримання структурованої програми дублюють ті модулі, у які можна увійти з декількох точок.

Приклад 1 Розглянемо неструктуровану програму, блок-схема якої зображена на рис. 7.3. Блоки позначені ідентифікаторами модулів

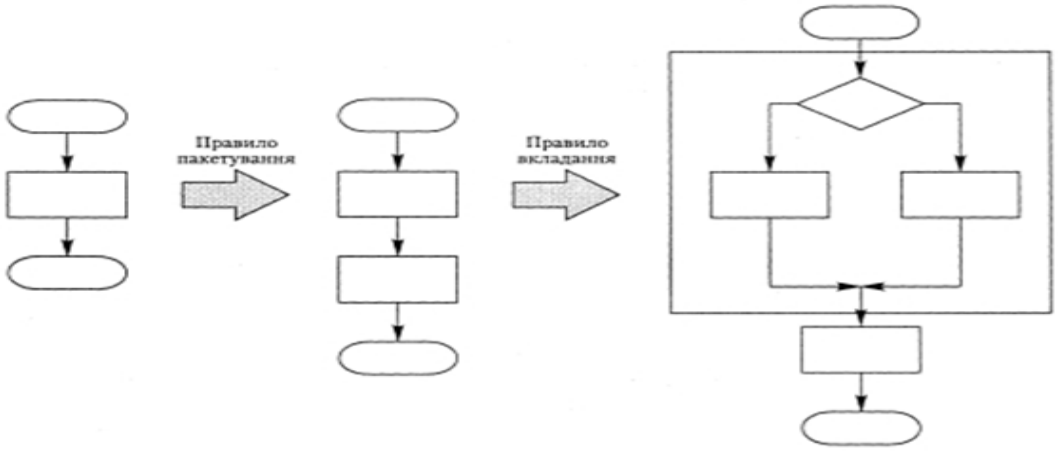


Рис. 7.5. Застосування правил пакетування та вкладання до простої програми

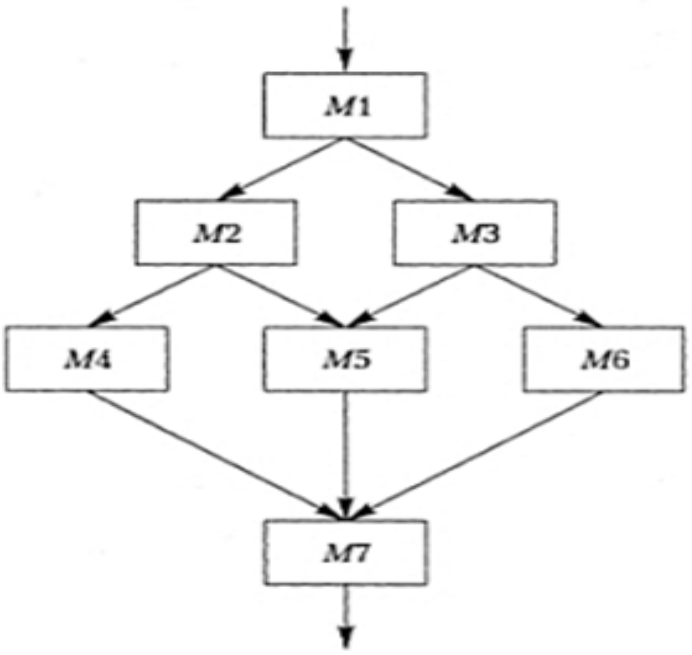


Рис. 7.6. Блок-схема неструктурованої програми

Із блок-схеми видно, що модуль М5 не може розпочати своєї роботи доти, доки не стануть відомими результати роботи модулів М2 і М3. Як правило, в модулі М5 міститься фрагмент коду, що активізується під час виклику цього модуля з модуля М2, а також інший фрагмент коду, що активізується під час виклику модуля М5 з модуля М3. Перетворимо зображену на рис. 7.6 неструктуровану програму, застосовуючи правила пакетування та вкладання. Зобразимо початкову програму як просту, що складається з елементарних програм типу вибору **(`if...then...else`).** Користуючись правилом вкладання, розширимо блоки MX і MY, замінивши їх елементарними програмами типу **`if...then...else`.** У результаті заміни модуль М5 продублюється (рис. 7.7).

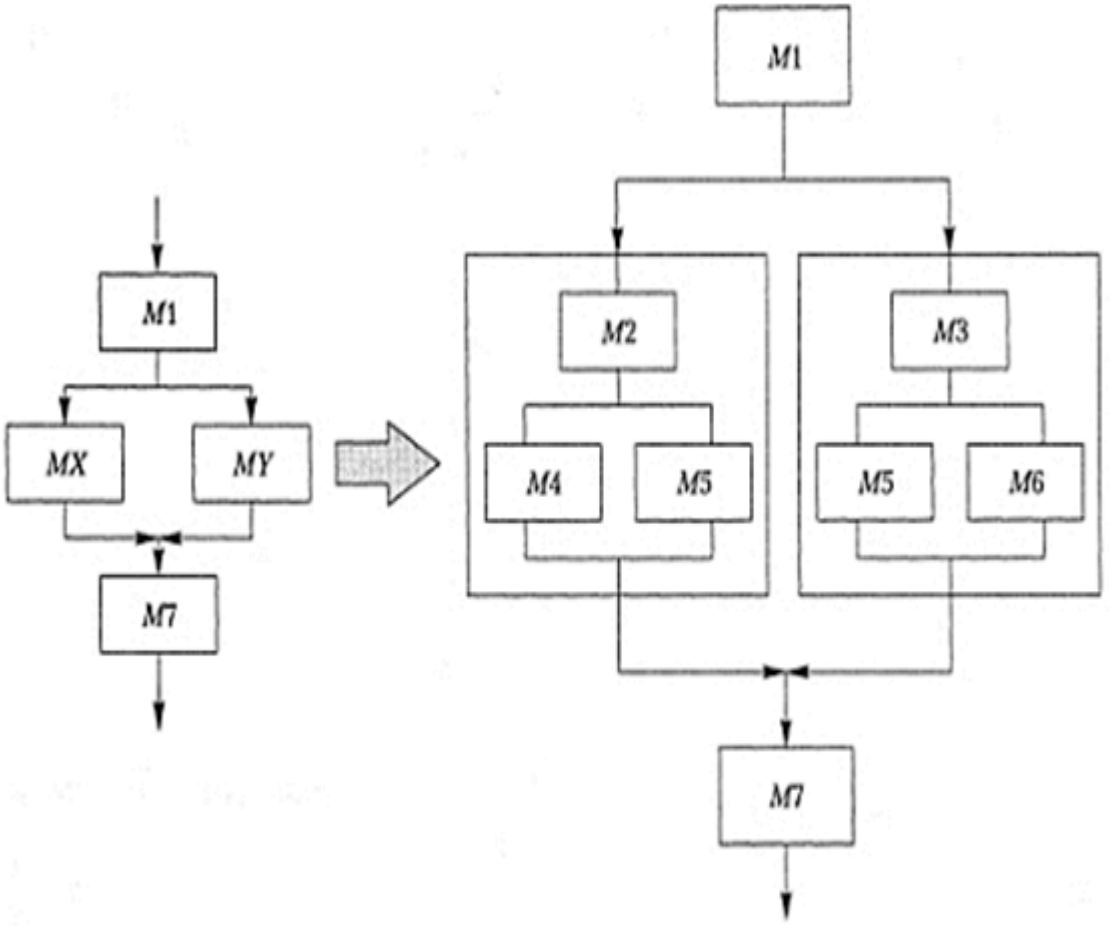


Рис. 7.7. Процес перетворення неструктурованої програми у структуровану

Приклад 2 Розглянемо блок-схему неструктурованої програми, що містить цикли (рис. 7.8). Кожному блоку приписано довільний номер. У програму вводиться новазмінна, Цій змінній присвоюється номер блока, на який потрібно передати керування. Зокрема, якщо істинна умова A, то цій змінній присвоюється номер блока B, тобто 2, якщо істинна умова C, змінній і присвоюється 1 — номер блока A. Логічні блоки неструктурованої блок-схеми замінюються елементарними програмами типу **`if...then...else`**, що послідовно перевіряють значення змінної стану. Гілки «**Так**» програм типу **`if...then...else`** мають містити оператори присвоєння нових значень змінній стану і оператори перевірки певних умов початкової програми.

Блок-схему неструктурованої програми зображено на рис. 7.8 а блок-схему відповідної структурованої програми — на рис. .

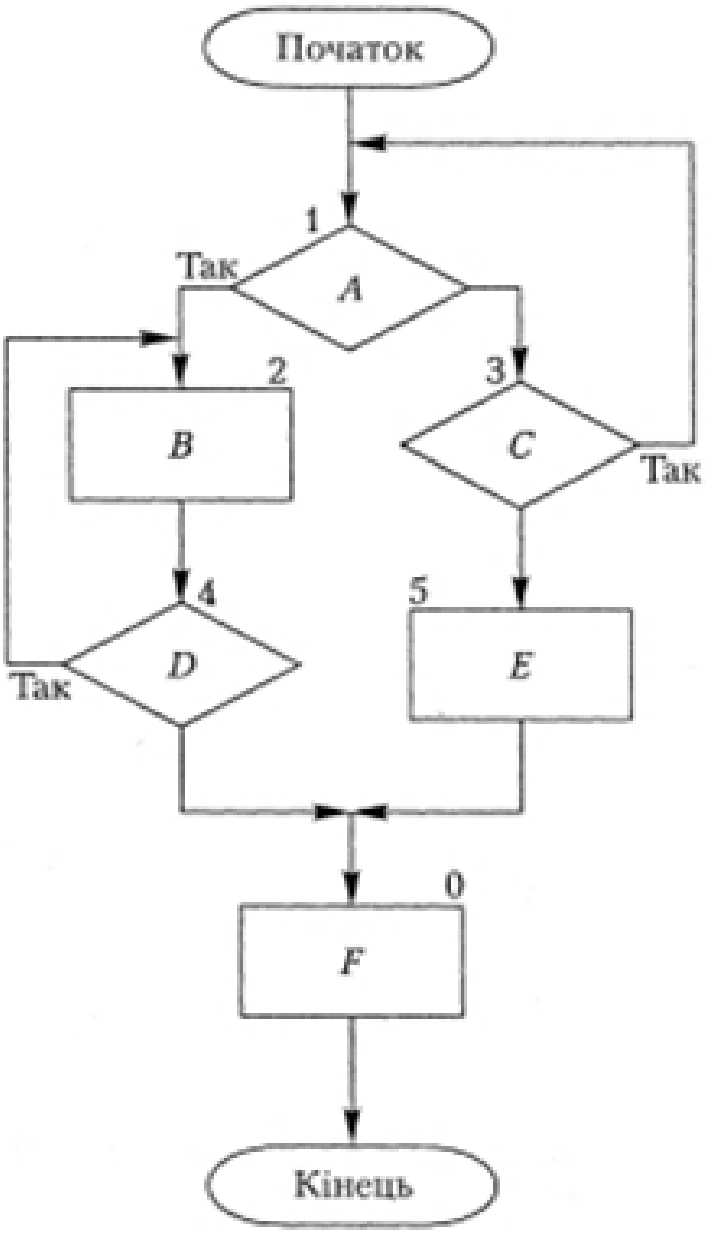


Рис. 7.8. Блок-схема неструктурованої програми, що містить цикли

Метод булевої ознаки використовується ддя перетворення неструктурованих програм, що містять цикли. В програму вводиться додаткова змінна, яка набуває значення **`true`** чи **`false`.** Доки змінна ознаки зберігає це значення, виконання циклу триває. Значення змінної ознаки всередині циклу модифікується лише за певних умов.

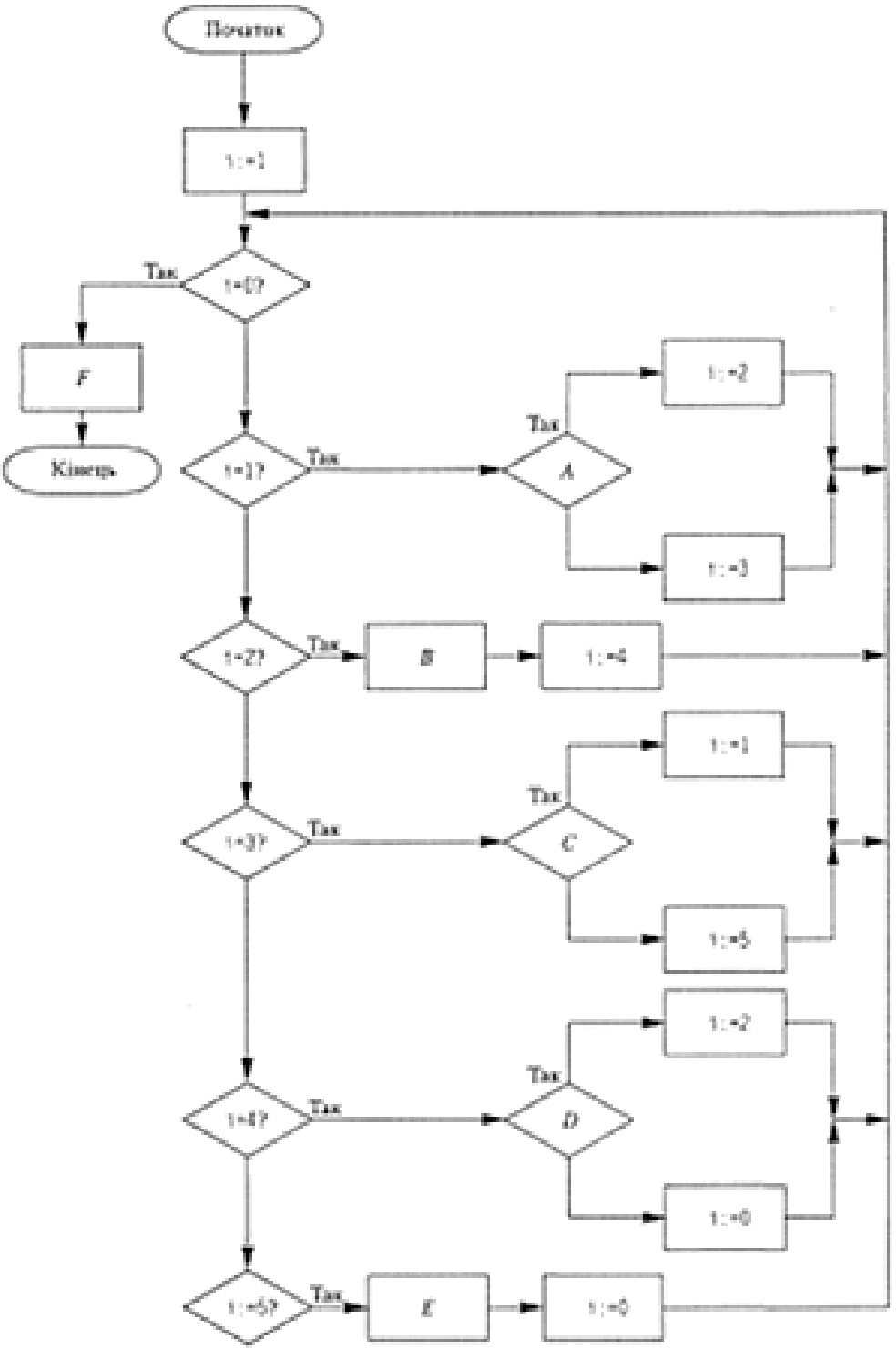


Рис. 7.9. Блок-схема структурованої циклічної програми

Приклад 3 Розглянемо блок-схему неструктурованого циклу, що має одну точку входу та дві точки виходу (рис. 7.10). У програму вводиться змінна `flag`, що ініціалізується нулем **(`false`).** Якщо істинною є умова P або Q, змінна `flag` набуває значення 1 **(`true`),** що приводить до виходу з циклу. В такий спосіб неструктурований цикл із двома виходами перетворюється на структурований цикл з одним виходом (рис. 7.11.).

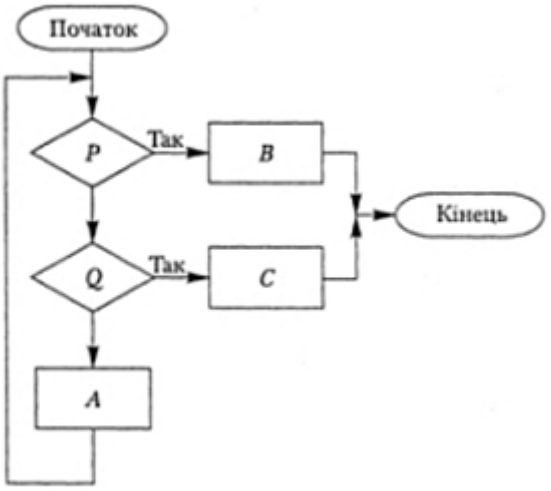


Рис. 7.10. Блок-схема неструктурованого циклу

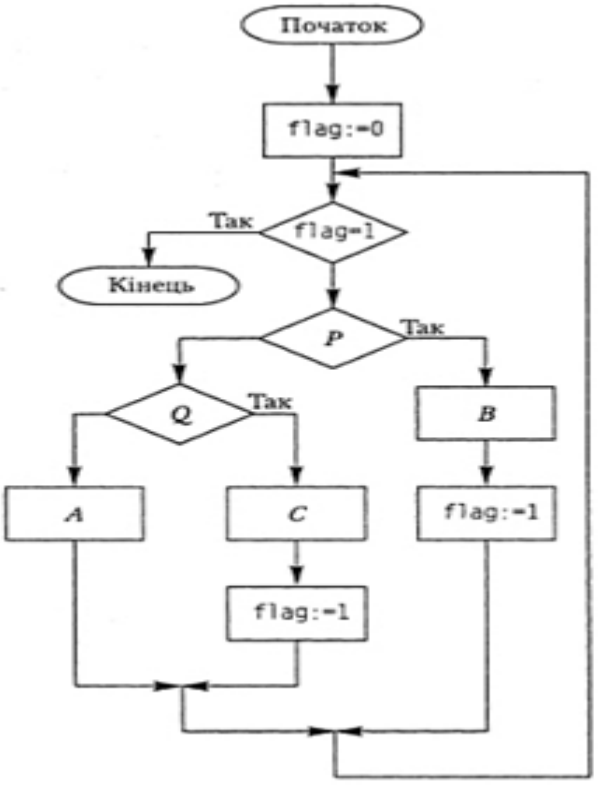


Рис.7.11. Блок-схема структурованого циклу.

7.7. Поняття про об’єкто- орієнтоване програмування.

Методологія об'єктно-орієнтованого програмування виникла як результат природної еволюції мов структурного програмування. З погляду цієї методології програма є сукупністю об'єктів, кожен об'єкт є екземпляром певного класу, а класи утворюють ієрархію успадкування.

Об'єкт — це реальна або абстрактна сутність, яка моделює частину навколишньої дійсності. Отже, кожний реальний предмет — це об'єкт.

Наприклад, нафтогазовим об’єктом може бути резервуар, пастка , поклад, частина покладу, пласт, сукупність пластів, колектор, свердловина тощо. При описі параметрів такого об’єкта необхідно вказувати відомості про степінь освоєння такого об’єкта, фазовий стан флюїду, літологію колектора, просторове положення (нафтогазоносна провінція, область, район і.т.і.), тектонічна належність (ера, система, відділ, ярус, світа, горизонт), стратиграфічна належність (блок, склепіння тошо).

Прикладами реальних об'єктів є також родовище, автомобіль, літак, завод, людина, матриця, вектор тощо. Ті самі слова: «родовище», «автомобіль», «літак», «людина» тощо позначають не об'єкти, а *класи*, коли йдеться не про конкретне родовище, автомобіль, літак або людину, а, наприклад, про нафтове, газове, вугільне чи сланцеве родовище як різновид родовища, про автомобіль або літак як різновид транспортного засобу і про людину як біологічний вид. Отже, клас є абстракцією множини об'єктів, що мають спільні властивості і поведінку. Тип родовища, транспортний засіб і біологічний вид — це теж класи.

Відношення між газовим родовище і типом родовища, між літаком і транспортним засобом або між видом «homo sapiens» і біологічним видом взагалі є відношенням *успадкування*, або відношенням «є»: газовим родовище є типом родовища, літак є різновидом транспортного засобу, вид «homo sapiens» є різновидом біологічного виду. Коли клас B є різновидом класу A, то A називається класом-предком, B — класом-нащадком.

З погляду програмування об'єкт складається з атрибутів і методів.

Атрибути описують властивості об'єкта у певний момент часу, а *методи* — описує властиву для об'єкта поведінку. Всі об'єкти, що є екземплярами одного класу, мають однаковий набір атрибутів і методів. Значення атрибутів зберігаються в змінних, а дії методів описуються в процедурах або функціях.Тому клас може бути визначений як набір оголошень змінних-атрибутів і підпрограм-методів. Визначені всередині класу елементи даних називаються *змінними-членами* класу, процедури і функції —функціями-членами, або методами класу. Оголошення класу називається його *інтерфейсом*, а опис його методів — *реалізацією*. Як правило, інтерфейс класу відокремлюється від його реалізації.

Об'єктно-орієнтоване програмування ґрунтується на трьох концепціях: інкапсуляції, успадкування і поліморфізму.

Інкапсуляція — це механізм, який дозволяє захистити атрибути й методи об'єкта від некоректного використання. Згідно з принципами інкапсуляції атрибути класу не можуть бути доступними для екземплярів інших класів безпосередньо. Доступ до атрибутів має здійснюватися лише

через методи класу. Наприклад, доступ до двигуна автомобіля можна здійснити лише за допомогою методів «завести», «вимкнути», «перемкнути швидкість» тощо.

Саме завдяки інкапсуляції можна отримати зиск у разі відокремлення інтерфейсу класу від його реалізації. Адже інкапсуляція дає можливість зробити програми, що використовують об'єкти певного класу, незалежними від способу реалізації цього класу.

Успадкування дозволяє систематизувати подібні класи на підставі їх спільних властивостей. Клас, що містить атрибути і методи, спільні для групи подібних один до одного класів, називається *базовим класом*, або класом-предком. Класи, що успадковують властивості і функціональні особливості базового класу, називають *похідними класами*, або класами-нащадками. Як уже зазначалося, успадкування застосовують для класів, пов'язаних семантичним відношенням «є». Іншими словами, вважається, що екземпляр похідного класу водночас є екземпляром базового класу. Це дає можливість об'єктам-нащадкам використовувати атрибути і методи об'єктів-предків як свої власні (хоча слід зазначити, що деякі мови програмування, наприклад C++, дозволяють встановлювати обмеження на доступ до атрибутів і методів предків з боку методів нащадків). Найважливішою властивістю успадкування є те, що воно дає можливість уникнути повторень коду, адже спільний для множини подібних класів код може бути винесений в методи їх спільного предка. За допомогою механізму успадкування можна побудувати ієрархію класів. В об'єктно-орієнтованому програмуванні взаємодія між об'єктами здійснюється шляхом передачі повідомлень. Об'єкт, що отримав повідомлення, реагує на нього викликом відповідного методу. Оскільки кожен об'єкт-нащадок є водночас будь-яким своїм предком, то надіслане такому об'єкту повідомлення надсила ється насправді декільком об'єктам різних класів. Рішення про те, якому саме об'єкту слід опрацювати повідомлення, залежить від обставин, що з'ясовуються під час виконання програми. Здатність об'єктів похідних класів порізному реагувати на ті самі повідомлення, називаєть *поліморфізмом*. Поліморфізм дає мож ливість визначати метод, що його буде викликано, під час виконання програми, а не під час її компіляції, а також можливість використовувати однойменні мето ди з різними заголовками.

Питання для самоконтролю.

1.Дайте визначення алгоритму ?

2. Перелічите властивості алгоритму та дайте їх коротку характеристику.

3.Чому в алгоритмічних мовах виникає потреба у використанні підпрограм ?
4.Які типи підпрограм використовують в алгоритмічних мовах ?
5.В чому суть оператора функції, процедури функції та процедури підпрограми ?
6.В чому полягає основна функція мов програмування ?
7.Яка роль систем програмування при створенні програмних подуктів ?
8.Що таке символьне кодування ?
1. Перелічите основні етапи технології створення програми.
2. Які типи помилок можна виявити при налагодженні програм ?
3. В чому різниця між технологією структурного програмування і технологією об'єктно-орієнтованого програмування ?
4. Що таке об'єктний код ?
13.В чому різниця між інтерпретацією і трансляцією програми
14.У чому полягає метод низхідного проектування?
15.Що таке алгоритмічна декомпозиція?
16.Що таке програмний модуль?
17.Чим відрізняється проста програма від елементарної?
18.Сформулюйте теорему про структурування.
19.Викладіть сутність методів структурування циклічних програм.
20.Які оператори неприпустимо використовувати у структурованій програмі ?
21.Дайте визначення методології об'єктно-орієнтованого програмування.
22.Чим відрізняється клас від об'єкта?
23.Які концепції покладено в основу об'єктно-орієнтованого програмування?
24.Які вигоди можна отримати від використання інкапсуляції?
25.Які вигоди можна отримати від використання інкапсуляції?

Розділ 8. Аналіз сучасних інформаційні технології..

8.1.Поняття про інформаційні технології.

Інформаційна технологія (ІТ) , відповідно до визначення, прийнятого ЮНЕСКО, - це комплекс взаємозалежних, наукових, технологічних, інженерних дисциплін, що вивчають методи ефективної

організації праці людей, зайнятих опрацюванням і збереженням інформації; обчислювальну техніку і методи організації і взаємодії з людьми і виробничим устаткуванням, практичні додатки, а також пов'язані з усім цим соціальні, економічні і культурні проблеми. Самі інформаційні технології вимагають складної підготовки, великих початкових витрат і наукомісткої техніки. Їхнє введення повинно починатися зі створення математичного забезпечення, формування інформаційних потоків у системах підготовки спеціалістів.

Інформація - явище незрівнянно більш древнє, ніж сама людина. Уже природа у процесі своєї еволюції передавала закодовану інформацію в рослинах і живих організмах. З перших своїх кроків люди шукають і знаходять нові засоби передачі, збереження та обробки інформації. Однак ніколи раніше людство не накопичувало інформацію й знання настільки стрімкими темпами. Тому закономірним є те, що жодна галузь людської діяльності не зазнала такого розвитку як інформаційні технології. Саме вони були покликані збільшити ефективність та зручність використання різноманітних видів інформації. За останні десятиріччя інформаційні технології зазнали такого глобального поширення, що зараз уже важко уявити життя сучасної людини без них. На сучасному етапі можна без особливих труднощів навести приклади використання інформаційних технологій у всі галузях: від освіти і до менеджменту. Сьогодні успіх буде мати та фірма, той заклад, який володіє найсучаснішими комп’ютерними технологіями. Значного прогресу можна досягти і в галузі освіти з впровадженням відповідних інформаційних компю’ютерних технологій, які зможуть зробити процес здобуття освіти більш гнучким, індивідуалізованим і одночасно нададуть змогу студентам використовувати глобальні ресурси для навчання, спілкуватись та обмінюватись досвідом із студентами інших міст, країн тощо.

Технологія - це комплекс наукових та інженерних знань, реалізованих у прийомах праці, наборах матеріальних, технічних, енергетичних, трудових факторів виробництва, засобах їх об'єднання для створення продукту або послуги, що відповідають певним вимогам. Тому технологія нерозривно пов'язана з машинізацією виробничого або невиробничого, насамперед управлінського процесу. Управлінські технології грунтуються на застосуванні комп'ютерів і телекомунікаційної техніки.

8.2. Етапи розвитку інформаційних технологій

Існує декілька точок зору щодо розвитку інформаційних технологій із використанням комп'ютерів, що визначаються різноманітними ознаками

поділу. Загальним для усіх викладених нижче підходів є те, що з появою персонального комп'ютера почався новий етап розвитку інформаційної технології. Основною ціллю стає задоволення персональних інформаційних потреб людини як для фахової сфери, так і для побутової.

Ознака поділу - вид задач і процесів опрацювання інформації:

1-й етап (60 - 70-і рр.) - опрацювання даних в обчислювальних центрах у режимі колективного користування. Основним напрямком розвитку інформаційної технології була автоматизація операційних рутинних дій людини.

2-й етап (з початку 80-х рр.) - створення інформаційних технологій, спрямованих на розв'язання стратегічних задач.

Ознака поділу - проблеми, які стоять на шляху інформатизації:

1- й етап (до кінця 60-х рр.) характеризується проблемою опрацювання великих обсягів даних в умовах обмежених можливостей апаратних засобів.

2-й етап (до кінця 70-х рр.) пов’язаний з поширенням ЕОМ серії IBM/360. Проблема цього етапу - відставання програмного забезпечення від рівня розвитку апаратних засобів.

3-й етап (з початку 80-х рр.) - комп'ютер стає інструментом непрофесійного користувача, а інформаційні системи - засобом підтримки прийняття його рішень. Проблемами цього етапу є максимальне задоволення потреб користувача і створення відповідного інтерфейсу для роботи в комп'ютерному середовищі.

4-й етап (з початку 90-х рр.) - створення сучасної технології міжустановних зв'язків і інформаційних систем. Проблеми цього етапу дуже багаточислені. Найбільше суттєвими з них є:

• укладання угод і встановлення стандартів, протоколів для комп'ютерного зв'язку;

• організація доступу до стратегічної інформації;

• організація захисту і безпеки інформації.

Ознака поділу - перевага, яку надає комп'ютерна технологія:

1-й етап (з початку 60-х рр.) характеризується досить ефективним опрацюванням інформації при виконанні рутинних операцій з орієнтацією на централізоване коллективне використання ресурсів обчислювальних центрів. Основним критерієм оцінки ефективності інформаційних систем, які створювались, була різниця між витраченими на розробку і зекономленими в результаті впровадження коштами. Основною проблемою на цьому етапі була психологічна - погана взаємодія користувачів, для яких створювалися інформаційні системи, і

розроблювачів через розходження їхніх поглядів і розуміння розв'язуваних проблем. Як наслідок цієї проблеми, створювалися системи, які користувачі погано сприймали і, незважаючи на їх достатньо великі можливості, не використовували повною мірою їх потенціал.

2-й етап (з середини 70-х рр.) пов'язаний з появою персональних комп'ютерів. Змінився підхід до створення інформаційних систем -орієнтація зміщається убік індивідуального користувача для підтримки прийнятих ним рішень. Користувач зацікавлений у проведеній розробці, налагоджується контакт із розроблювачем, виникає порозуміння між обома групами спеціалістів. На цьому етапі використовується як централізоване опрацювання даних, характерне для першого етапу, так і децентралізоване, що базується на розв'язанні локальних задач і роботі з локальними базами даних на робочому місці користувача.

3-й етап (з початку 90-х рр.) пов'язаний з поняттям аналізу стратегічних переваг у бізнесі і заснований на досягненнях телекомунікаційної технології розподіленого опрацювання інформації. Інформаційні системи мають своєю метою не просто збільшення ефективності опрацювання даних і допомога керівнику. Відповідні інформаційні технології повинні допомогти організації вистояти в конкурентній боротьбі й одержати перевагу.

Ознака поділу - види інструментарію інформаційної технології:

1-й етап (до другої половини XIX ст.) - "ручна" інформаційна технологія інструментарій якої складали: ручка, чорнильниця, книга. Комунікації здійснювалися ручним способом шляхом переправки через пошту листів, пакетів, депеш. Основною метою інформаційної технології цього періоду було представлення інформації в потрібній формі.

2-й етап (з кінця XIX ст.) - "механічна" технологія, інструментарій якої складали: друкарська машинка, телефон, диктофон, оснащена більш досконалими засобами доставки пошта. Основна мета технології - представлення інформації в потрібній формі більш зручними засобами.

3-й етап (40-60-і рр. XX ст.) - "електрична" технологія, інструментарій якої складали: великі ЕОМ і відповідне програмне забезпечення, електричні друкарські машинки, ксерокси, портативні диктофони. На цьому етапі відбувається зміна мети технології. Акцент в інформаційній технології починає зміщуватись з форми представлення інформації на формування її змісту.

4-й етап (з початку 70-х рр.) - "електронна" технологія, основним інструментарієм якої стають великі ЕОМ і створені на їхній базі автоматизовані системи керування (АСК) і інформаційно-пошукові

системи (ІПС), оснащені широким спектром базових і спеціалізованих програмних комплексів. Центр ваги технології ще більш зміщується на формування змістовної сторони інформації для управлінського середовища різноманітних сфер громадського життя, особливо на організацію аналітичної роботи. Безліч об'єктивних і суб'єктивних факторів не дозволили вирішити поставлені перед новою концепцією інформаційної технології задачі. Проте був здобутий досвід формування змістовної сторони управлінської інформації і підготовлена фахова, психологічна і соціальна база для переходу на новий етап розвитку технології.

5-й етап (з середини 80-х рр.) - "комп'ютерна" ("нова") технологія, основним інструментарієм якої є персональний комп'ютер із широким спектром стандартних програмних продуктів різного призначення. На цьому етапі відбувається процес персоналізації АСК, що проявляється у створенні систем підтримки прийняття рішень певними спеціалістами. Подібні системи мають умонтовані елементи аналізу та інтелекту для різних рівнів керування, вони реалізуються на персональному комп'ютері і використовують телекомунікації. У зв'язку з переходом на мікропроцессорну базу суттєвим змінам піддаються і технічні засоби побутового, культурного та інших призначень. Починають широко використовуватися в різноманітних галузях глобальні і локальні комп'ютерні мережі.

8.3. Складові частини інформаційної технології

Такі технологічні поняття, що використовуються у виробничій сфері, як норма, норматив, технологічний процес, технологічна операція і т.п., можуть застосовуватися і в інформаційній технології. Перед тим, як розробляти ці поняття в будь-якій технології, у тому числі й в інформаційній, завжди варто починати з визначення мети. Потім варто спробувати провести структурування всіх дій, що призводять до наміченої мети, і вибрати необхідний програмний інструментарій.

Необхідно розуміти, що освоєння інформаційної технології і подальше її використання повинні бути зведені до того, щоб було потрібно спочатку добре оволодіти набором елементарних операцій, кількість яких обмежена. З цієї обмеженої кількості елементарних операцій у різних комбінаціях складається дія, а з дій, також у різних комбінаціях, складаються операції, що визначають той або інший технологічний етап. Сукупність технологічних етапів утворить технологічний процес (технологію). Він може починатися з будь-якого рівня і не включати, наприклад, етапи або операції, а складатися тільки з дій. Для реалізації

етапів технологічного процесу можуть використовуватися різні програмні середовища.

Інформаційна технологія, як і будь-яка інша, повинна відповідати таким вимогам:

1) забезпечувати високий рівень розчленування всього процесу опрацювання інформації на етапи (фази), операції, дії;

2) включати весь набір елементів, необхідних для досягнення поставленої мети;

3) мати регулярний характер. Етапи, дії, операції технологічного процесу можуть бути стандартизовані й уніфіковані, що дозволить більш ефективно здійснювати цілеспрямоване керування інформаційними процесами.

Реалізація технологічного процесу матеріального виробництва здійснюється за допомогою різноманітних технічних засобів, до яких відносяться: устаткування, верстати, інструменти, конвеєрні лінії і т.п.

За аналогією і для інформаційної технології повинно бути щось подібне. Такими технічними засобами виробництва інформації будуть апаратне, програмне і математичне забезпечення цього процесу. З їхньою допомогою відбувається переробка первинної інформації в інформацію нової якості. Виділимо окремо з цих засобів програмні продукти і назвемо їх інструментарієм, а для більшої чіткості можна його конкретизувати, назвавши програмним інструментарієм інформаційної технології.

Інструментарій інформаційної технології - один або декілька взаємозалежних програмних продуктів для певного типу комп'ютера, технологія роботи в яких дозволяє досягти поставленої користувачем мети. Як інструментарію можна використовувати такі поширені види програмних продуктів для персонального комп'ютера як текстовий процесор (редактор), настільні видавничі системи, електронні таблиці, системи керування базами даних, електронні записні книжки, електронні календарі, інформаційні системи функціонального призначення (фінансові, бухгалтерські, для маркетингу та ін.), експертні системи і т.д.

8.4. Види сучасних інформаційних технологій

ІТ відрізняються за типом інформації, що обробляється, але можуть бути об’єднані в інтегровані технології (рис.8.1).

Ця класифікація умовна тому, що більшість цих ІТ дозволяє підтримувати і інші види інформації. Модифікація елементів, що складають поняття ІТ, дає можливість утворення значної їх кількості в різних комп’ютерних середовищах. І тому можна говорити про забезпечувальні (ЗІТ) і функціональні (ФІТ) ІТ.

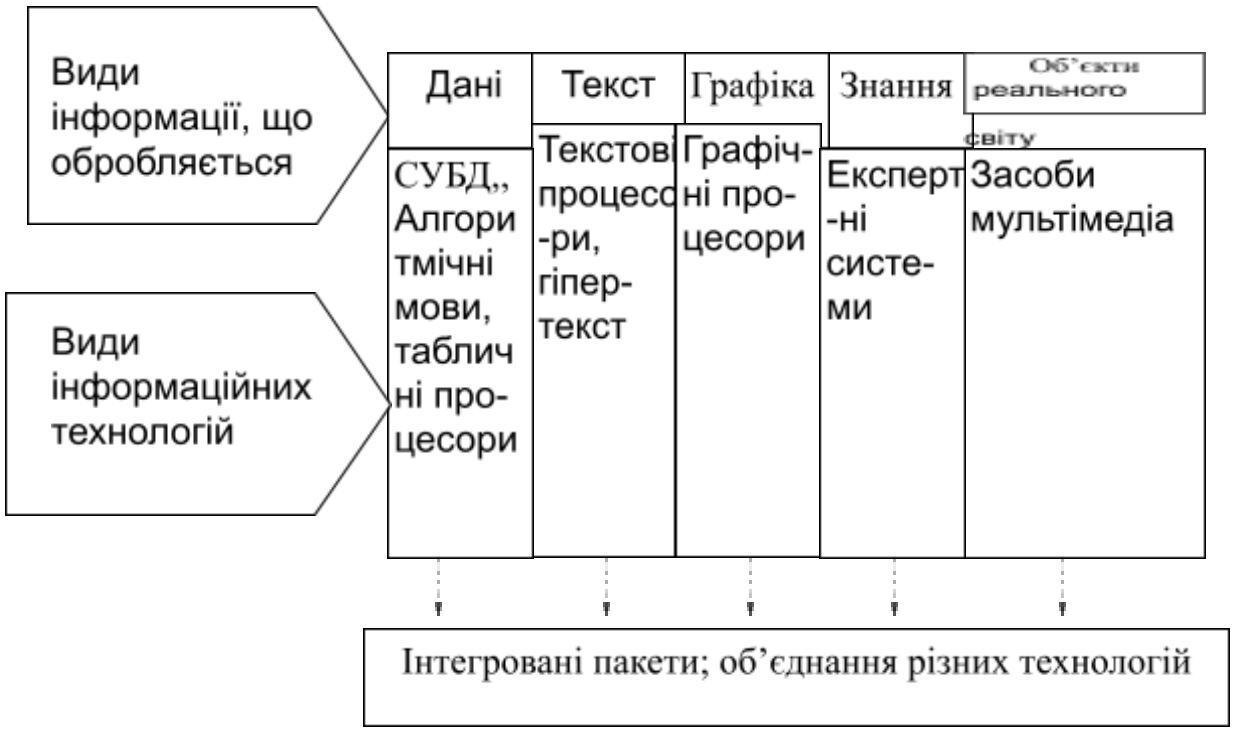


Рис.8.1. Класифікація ІТ в залежності від типу інформації, що обробляється.

ЗІТ – технологія обробки інформації, які можуть виконуватися як інструментарій в різних предметних областях для рішення різних задач. ІТ забезпечувального типу можуть бути класифіковані відносно класів задач, на які вони орієнтовані. Забезпечувальні технології базуються на цілком різних платформах, що обумовлено відмінністю видів комп'ютерів і програмних середовищ, тому при їх об'єднанні на основі предметної технології (яка уявляє собою послідовність технологічних етапів по модифікації первинної інформації в результативну) виникає проблема системної інтеграції. Вона заключається в необхідності проведення різних ІТ до єдиного стандартного інтерфейсу

ФІТ –- уявляє собою таку модифікацію забезпечувальних ІТ, при якій реалізується яка-небудь з предметних технологій. Наприклад, робота співробітника кредитного відділу банку з використанням ПК обов'язково перебачає застосування сукупності банківських технологій оцінки кредитоздатності кредитозаємника, формування кредитного договору і термінових зобов'язань розрахунку графіка платежів і інших технологій, що реалізовані в якій-небуть ІТ і СУБД, текстовому процесорі і т.д. Трансформація ЗІТ в ФІТ (модифікація деякого загально вживаного інструментарія і спеціальний) може бути зроблена як спеціалістом-проектувальником, так і самим користувачем. Це залежить від того, наскільки складна така трансформація, тобто від того, наскільки

вона доступна самому користувачеві-програмісту. Ці можливості рік за роком стають все більш дружелюбними, все більш розширяються, тобто, в арсеналі співробітника кредитного відділу можуть знаходитися як ЗІТ, з якими він постійно працює: текстові і табличні процесори, так і спеціальні функціональні технології: табличні процесори, СУБД, ЕС, що реалізують предметні технології.

Питання для самоконтролю

1.Що таке інформаційна технологія відповідно до визначення, прийнятого ЮНЕСКО ?
2.Назвіть етапи розвитку інформаційної технології згідно ознак поділу по виду задач і процесів опрацювання інформації.
3.Назвіть етапи розвитку інформаційної технології згідно ознак поділу з проблеми, які стоять на шляху інформатизації.
4.Назвіть етапи розвитку інформаційної технології згідно ознак поділу про перевагу, яку надає комп'ютерна технологія:
5.Назвіть етапи розвитку інформаційної технології згідно ознак поділу за видами інструментарію інформаційної технології:
6.Перелічите вимоги яким повинні відповідати інформаційні технології.
7.В чому суть інструментарія інформаційної технології ?
8.Перелічіть види інформації, що обробляються.
9. Перелічіть види інформаційних технологій, які використовуються для обробки інформації.
10.В чому різниця між забезпечувальними і і функціональними інформаційними технологіями ?

Розділ 9. Уніфікована мова моделювання програмних продуктів UML

9.1. Передумови й історія виникнення UML

При створенні складних інженерних систем прийнято використовувати прийоми моделювання. Складність більшості програмних систем, що створюються не поступається складності багатьом інженерним спорудженням, тому моделювання програмних систем є досить актуальним завданням. Більш того, у таких концепціях, як MDA (Model Driven Architecture - архітектура на основі моделей) і MDD (Model Driven Development - розробка на базі моделей), моделям приділяється центральна роль у процесі створення програмного продукту. Основною ідеєю цих концепцій є представлення процесу створення

програмного продукту у вигляді ланцюжка трансформацій його вихідної моделі в готову програмну систему.

Майже у всіх інструментальних засобах, що втілює ідеї MDD, як мова моделювання використовується мова UML (Unified Modeling Language - уніфікована мова моделювання), цілком або які-небудь його частини.

9.2. Методики об’єктно-орієнтованого моделювання

Розвиток об’єктно-орієнтованих мов моделювання в 1980-х 1990-х роках призвів до появи великого числа об’єктно-орієнтованих підходів до моделювання. Зокрема, у період з1989 по 1994 роки загальна кількість відомих мов моделювання зросло з 10 до більш ніж 50. Кожна з мов мала свої переваги й недоліки також як і свою нотацію. Один і тойже символ у різних нотаціях найчастіше міг мати абсолютно різне значення.

Це ситуація суворої конкуренції між методиками моделювання дістала назву «війни методів».

«Війна методів» обумовила необхідність створення єдиної мови, що поєднувала би сильні сторін відомих методів. І в жовтні 1994 року почалося створення мови UML, основу якої склали кілька об'єктно-орієнтованих методів, що зарекомендували себе щонайкраще на практиці, але кожний з яких був націлений на рішення окремого завдання аналізу й проектування:

- метод Граді Буча, умовна назва BOOCH (Booch'91, BooCH Lite, Booch'93) вважався найбільш ефективним на етапах проектування й розробки програмних систем;
- метод Джеймса Рамбо, Object Modeling Technique (OMT, OMT-2) – оптимально підходив для аналізу процесів обробки даних в інформаційних системах ;
- метод Айвара Джекобсона (Ivar Jacobson), Object-Oriented Software Engineering (OOSE) – містив засоби представлення варіантів використання, що мають істотне значення на етапі аналізу вимог у процесі проектування бізнес-додатків .

Спочатку Г. Буч і Д. Рамбо з компанії Rational Software Corporation почали роботу з уніфікації своїх методів. Незважаючи на те, що самі по собі ці методи були досить популярні, спільна робота їх була спрямована на вивчення всіх відомих об’єктно-орієнтованих методів з метою виявлення їхніх переваг. Восени того ж року до них приєднався Айвар Якобсон, головний технолог компанії Objectory AB,, до того, щоб інтегрувати свій метод OOSE із двома вищезгаданими. Протягом усього року творці займалися збором відкликань від основних компаній, що

працюють в області створення ПЗ. За цей час стало ясно, що більшість таких компаній, що працюють в області створення , порахувала UML мовою, що має стратегічне значення для їхнього бізнесу.

У результаті був створений консорціум UML, у який увійшли організації, що виявили бажання надати ресурси для роботи, спрямованої на створення повної специфікації UML. Версія 1.0 мови з'явилася в результаті спільних зусиль компаній Digital Equipment Corporation, Hewlett Packard, I-Logix, Intellicprp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments, Unisys. UML 1.0 виявився добре визначеною, виразною, потужною мовою, яка може бути застосованою для рішення великої кількості різноманітних завдань.

На протязі декількох років спеціальна робоча група OMG (OMG Revision Task Force) підтримувала просування проекту UML. Були створені версії 1.3, 1.4, і 1.5. За 2000-2003 була розроблена версія UML 2.0. У листопаді 2007 випущена поточна версія UML 2.1.2.

9.3. Короткий опис та завдання UML.

Мова UML призначена для рішення наступних завдань:

1. надати в розпорядження користувачів готову до використання виразну потужну мову візуального моделювання, що дозволяє розробляти осмислені моделі й обмінюватися ними;

2. передбачити внутрішні механізми розширюваності й спеціалізації базових концепцій мови;

3. забезпечити максимальну незалежність проекту створення програмного забезпечення від конкретних мов програмування й процесів розробки;

4. забезпечити формальну основу для однозначної інтерпретації мови;

5.стимулювати розширення ринку об’єктно-орієнтованих інструментальних засобів створення програмного забезпечення;

6. інтегрувати кращий практичний досвід використання мови й реалізації програмних засобів його підтримки.

У значній мірі мова UML не залежить від процесу розробки програмного забезпечення(ПЗ). Уніфікований процес розробки ПЗ (Rational Unified Process, RUP) - це один з підходів до організації життєвого циклу ПЗ, який особливо добре сполучається з UML. Цей комерційний продукт задає строгий регламент розподілу завдань і відповідальності між виконавцями в процесі розробки ПЗ.

UML — мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого

профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаною UML моделлю. UML був створений для визначення, візуалізації, проектування й документування здебільшого програмних систем.

Використання UML не обмежується моделюванням програмного забезпечення. Їх також використовують для моделювання бізнес-процесів, системного проектування й відображення організаційних структур.

UML дозволяє розроблювачам ПЗ досягти угоди в графічних позначеннях для представлення загальних понять (таких як клас, компонент, узагальнення (generalization), об'єднання (aggregation) та поведінка) і більше сконцентруватися на проектуванні й архітектурі.

9.4. Загальна структура мови та концептуальна модель UML

Семантика мови UML визначається для двох видів об'єктних моделей: структурних і поводження. Структурні (статичні) моделі описують структуру сутностей або компонентів системи, включаючи їхні класи, інтерфейси, атрибути й зв'язки. Моделі поводження (динамічні) описують поведінку або функціонування об'єктів системи, включаючи їхні методи, взаємодію (співробітництво) між ними, а також процес зміни станів окремих компонентів і системи в цілому.

Формальний опис мови UML ґрунтується на наступній загальній ієрархічній структурі модельних подань, що складається із чотирьох рівнів абстракції:

- позначка-метамодель,
- метамодель,
- модель,
- об'єкти користувача .

Рівень метаметамоделі утворить базову основу для всіх метамодельних представлень і визначає мову для специфікації метамоделі. Позначка-Метамодель визначає модель мови UML на найвищому рівні абстракції (відповідно на найнижчому рівні конкретизації) і є найбільш компактним його описом. Метамодель - екземпляр або конкретизація позначка-метамоделі - визначає мову для специфікації моделей. Всі основні поняття мови UML - це поняття рівня метамоделі. Модель у контексті мови UML є екземпляром (конкретизацією) метамоделі в тому розумінні, що кожна (конкретна) модель системи повинна використовувати тільки поняття метамоделі, конкретизувавши їх стосовно відповідної ситуації. Змістовно говорячи, рівень моделі призначений для опису конкретної предметної області. Конкретизація понять моделі відбувається на рівні об'єктів, які є екземплярами моделі й містять

конкретну інформацію про предметну область відповідно до понятть моделі.

Основою представлення UML на метамодельному рівні є опис трьох його логічних блоків (пакетів):

- основні елементи,
- елементи поводження
- загальні механізми .

Концептуальна модель мови UML включає основні будівельні блоки, правила їхні сполучення й загальні механізми.

Словник мови UML містить сутності (абстракції, що є основними елементами моделі) і відносини (основні сполучні будівельні блоки), Сутності й відносини за певними правилами з'єднуються в конструкції - діаграми.

В UML визначено чотири типи сутностей :

− **структурні сутності**, що поділяються на основні
клас (Class), інтерфейс (Interface)
кооперація (Collaboration),
прецедент (Use case),
активний клас (Active class),
компонент (Component),
вузол (Node)
різновиди основних актор (Actor),
сигнал (Signal),
утиліта (Utility, вид класів),
процес (Process),
нитка (Thread, вид активних класів))
інші
додаток (Application),
документ (Document),
файл (File), бібліотека (Library),
сторінка (Page),
таблиця (Table, вид компонентів));

− **сутності поводження** (Behavioral things)
взаємодія (Interaction)
автомат (State machine);

− **сутності, що групують**, - пакет (Packages);

− **анотаційні сутності** – примітка (Note).

Основними типами відносин в UML є відносини:
залежності (Dependency),

асоціації(Association) (різновидом асоціації є відношення агрегації (Aggregation)),

узагальнення(Generalization)

реалізації (Realization).

Існують також їхні варіації, наприклад, уточнення, трасування, включення й розширення (для відносин залежності).

Для побудови коректно оформленої моделі в UML визначені правила, що дозволяють коректно й однозначно визначати:

(1) імена сутностей, відносин і діаграм,

(2) область дії імен (контекст, у якому ім'я має деяке значення),

(3) видимість імен (для використання іншими елементами),

(4) цілісність (правильність і погодженість співвідношення елементів),

(5) виконання моделі.

Ефективність і спрощення застосування мови забезпечується використанням певних угод, так званих, загальних механізмів:

- специфікацій (Specifications),
- доповнень (Adornments),
- прийнятих розподілів (Common divisions
- механізмів розширення (Extensibility mechanisms) [7, 17, 18].

Кожний елемент нотації UML має унікальне графічне позначення й специфікацію - текстове подання синтаксису й змістовної семантики відповідного будівельного блоку.

Практично всі будівельні блоки характеризуються дихотомією “клас/ об'єкт” і “інтерфейс / реалізація”. Це основні підходи розподілу реальності при об’єктно-орієнтованом моделюванні систем.

UML допускає контрольовані розширення для адаптації мови до конкретних потреб. Наявність внутрішніх механізмів розширення принципово відрізняє UML від таких засобів моделювання як IDEF0, IDEF1X, IDEF3, DFD і ERM , що є замкнутими й не допускають разширення засобами самої мови.

До механізмів розширення UML відносяться:

- **стереотип** (Stereotype), що розширює словник мови (дозволяє створювати з існуючих блоків нові, специфічні для конкретного розв'язуваного завдання);

- **тег-значення** (Tagged value), що розширює властивості будівельного блоку (дає можливість включати нову інформацію в специфікацію елемента);

- обмеження (Constraint), що розширює семантику будівельного блоку (дозволяє додавати нові або модифікувати існуючі правила за допомогою семантичних обмежень, заданих природною мовою або формальною мовою OCL). Деякі розширення придбали таку популярність, що були внесені в стандарт поточної версії.

9.5. Діаграми в UML.

У нотації UML всі представлення про моделі складної системи фіксуються у вигляді спеціальних графічних конструкцій, що одержали назву діаграм . Діаграма в UML - це графічне подання набору елементів, що зображується, як правило, у вигляді зв'язного графа з вершинами (сутностями) і ребрами (відносинами). Теоретично діаграми можуть містити будь-які комбінації сутностей і відносин. Однак на практиці застосовується невелика кількість типових комбінацій.

В UML використовуються наступні види діаграм :

Діаграма класів (Class diagram) — статична структурна діаграма, що описує структуру системи, вона демонструє класи системи, їхні атрибути, методи й залежності між класами.

Діаграма компонентів (Component diagram) — статична структурна діаграма, показує розбивку програмної системи на структурні компоненти й зв'язки (залежності) між компонентами. Як фізичні компоненти можуть виступати файли, бібліотеки, модулі, що виконуються файли, пакети й також

Діаграма композитної/складеної структури (Composite structure diagram) — статична структурна діаграма, демонструє внутрішню структуру класів і, по можливості, взаємодію елементів (частин) внутрішньої структури класу.

Підвидом діаграм композитної структури є **діаграми кооперації** (Collaboration diagram, уведені в UML 2.0), які показують ролі й взаємодія класів у рамках кооперації. Кооперації зручні при моделюванні шаблонов проектування.

Діаграми композитної структури можуть використовуватися разом з діаграмами класів. Діаграма розгортання (Deployment diagram) — служить для моделювання працюючих вузлів (апаратних засобів) і артефактів, розгорнутих на них. В UML 2.0 на вузлах розвертаються артефакти (англ. artifact), у той час як в UML 1.0 на вузлах розверталися компоненти. Між артефактом і логічним елементом (компонентом), що він реалізує, установлюється залежність маніфестації. Діаграма об'єктів (Object diagram) — демонструє повний або частковий знімок що моделюється системи в заданий момент часу. На діаграмі об'єктів відображаються

екземпляри класів (об'єкти) системи із вказівкою поточних значень їхніх атрибутів і зв'язків між об'єктами.

Діаграма пакетів (Package diagram) — структурна діаграма, основним змістом якої є пакети й відносини між ними. Твердого поділу між різними структурними діаграмами не проводиться, тому дана назва пропонується винятково для зручності й не має семантичного значення (пакети й діаграми пакетів можуть бути присутнім на інших структурних діаграмах). Діаграми пакетів служать, у першу чергу, для організації елементів у групи по якій-небудь ознаці з метою спрощення структури й організації роботи з моделлю системи.

Діаграма діяльності (Activity diagram) — діаграма, на якій показане розкладання деякої діяльності на її складові частини. Під діяльністю (activity) розуміється специфікація поводження, що виконується, у вигляді координованого послідовного й паралельного виконання підлеглих елементів - вкладених видів діяльності й окремих дій (action), з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого.

Діаграми діяльності використовуються при моделюванні бізнес-процесів, технологічних процесів, послідовних і паралельних обчислень.

Аналогом діаграм діяльності є схеми алгоритмів.

Діаграма автомата (State Machine diagram) (діаграма кінцевого автомата, діаграма станів) — діаграма, на якій представлений кінцевий автомат із простими станами, переходами й композитними станами.

Кінцевий автомат (State machine) — специфікація послідовності станів, через які проходить об'єкт або взаємодія у відповідь на події свого життя, а також відповідні дії об'єкта на ці події. Кінцевий автомат прикріплений до вихідного елемента (класу, кооперації або методу) і служить для визначення поведінки його екземплярів.

Діаграма прецедентів (Use case diagram) (діаграма варіантів використання) — діаграма, на якій відбиті відносини, що існують між акторами і прецедентами.

Основне завдання - являти собою єдиний засіб, що дає можливість замовникові, кінцевому користувачеві й розроблювачеві спільно обговорювати функціональність і поводження системи.

Діаграми комунікації й послідовності транзитивни, виражають взаємодію, але показують його різними способами й з достатнім ступенем точності можуть бути перетворені одна в іншу.

Діаграма комунікації (Communication diagram) (в UML 1.x — діаграма кооперації, collaboration diagram) — діаграма, на якій

зображуються взаємодії між частинами композитної структури або ролями кооперації. На відміну від діаграми послідовності, на діаграмі комунікації явно вказуються відносини між елементами (об'єктами), а час як окремий вимір не використовується (застосовуються порядкові номери викликів).

Діаграма послідовності (Sequence diagram) — діаграма, на якій зображене впорядковане в часі взаємодія об'єктів. Зокрема, на ній зображуються об'єкти, що беруть участь у взаємодії, і послідовність повідомлень, якими вони обмінюються.

Діаграма огляду взаємодії (Interaction overview diagram) — різновид діаграми діяльності, що включає фрагменти діаграми послідовності й конструкції потоку керування.

Діаграма синхронізації (Timing diagram) — альтернативне подання діаграми послідовності, що явно показує зміни стану на лінії життя із заданою шкалою часу. Може бути корисна в додатках реального часу. Інженерія програмного забезпечення з комп'ютерною підтримкою

Останні п'ятдесят років дослідники й розроблювачі програмного забезпечення створюють абстракції, що допомагають їм програмувати в термінах цілей свого проекту, а не використовувати комп'ютерне середовища, і не захищати їх цілі від складностей цього середовища. Із самого початку ці абстракції включали технології мов програмування й операційних систем. Наприклад, ранні мови програмування (мови асемблера й Fortran) захищали розроблювачів від складностей програмування в машинних кодах.

Аналогічно, ранні операційні системи захищали їх від складностей програмування прямо на рівні апаратури. Хоча ці ранні мови й платформи підвищували рівень абстракції, вони явно були "орієнтованими на обчислення". Зокрема, вони забезпечували абстракції простору рішень (тобто області самих комп'ютерних технологій), а не абстракції, що дозволяють звести розробку в термінах прикладної області. Згодом вживали численні спроби подальшого підвищення рівня абстракції.

Одним з найбільш відомих напрямків, що утворилися в 1980-е рр., є інженерія програмного забезпечення з комп'ютерною підтримкою (Computer-Aided Software Engineering, CASE), що забезпечує методи й засоби розробки програмного забезпечення, які дозволяють розроблювачам виражати свої конструкції з використання графічних програмних засобів загального призначення, різного виду діаграм. Однієї із цілей CASE-засобів було забезпечення більше ретельного аналізу графічних програм за рахунок їхньої меншої складності, ніж у програм, представлених на традиційних мовах програмування (наприклад, у

графічних програмах неможливі помилки, що приводять до ушкодження пам'яті).

Іншою проблемою підходу CASE була його нездатність масштабуватися до складних, виробничих систем у широкому діапазоні прикладних областей. Загалом кажучи, CASE-засоби не підтримували паралельну розробку; на їхній основі можна було розробляти програми поодинці або групою, члени якої по черзі зверталися до файлів, що використовували ці засоби.

У результаті в 1990-е рр. підхід CASE робив порівняно невеликий вплив на розробку комерційного програмного забезпечення, фокусуючись на декількох прикладних областях, наприклад, на обробці викликів у телекомунікаційних системах, де добре підходили подання у вигляді кінцевих автоматів. Навіть там, де CASE-засоби застосовувалися на практиці, звичайно використовувалася тільки їхня частина, що дозволяє розроблювачам малювати діаграми програмних архітектур і документувати свої рішення, на основі чого програмісти потім вручну робили й розвивали реалізації. Однак, оскільки був відсутній прямий зв'язок між діаграмами й реалізаціями, розроблювачі не прагнули до великої точності діаграм, оскільки вони рідко синхронізувалися із кодом на подальших стадіях проектів.

В останні двадцять років досягнення в області мов програмування й платформ привели до підвищення рівня абстракцій, доступних для програмістів, зм'якшивши один з недоліків підходу CASE. Наприклад, сьогодні програмісти звичайно використовують більше виразні об’єктно-орієнтовані мови (зокрема, C++, Java і C#), а не Fortran або C. Повторно що використані бібліотеки класів і платформи підтримки додатків мінімізують потребу у винаході загальних і прикладних сервісів - транзакцій, відмовостійкості, оповіщення про події, безпеку, розподіленого керування ресурсами й т.д. Все це дозволяє програмістам краще захиститися від складності, пов'язаної зі створенням додатків на основі традиційних технологій.

Незважаючи на ці досягнення, залишається кілька неприємних проблем. У центрі цих проблем лежить складність платформ, що росте швидше здатності мов загального призначення маскувати її. Наприклад, популярні платформи проміжного програмного забезпечення J2EE, .NET і CORBA містять тисячі класів і методів з багатьма складними залежностями й тонкими побічними ефектами, що вимагає значних зусиль при програмуванні й ретельному настроюванні.

Родинна проблема полягає в тому, що код більшості додатків і платформ як і раніше пишеться на мовах третього покоління, для чого потрібно багато часу й зусиль, особливо для виконання інтеграційних дій - розгортання, конфігурування й підтримка якості системи. Наприклад, на Java або C# важко написати код, коректно й оптимально розгортати великомасштабну розподіленусистему із сотнями тисяч взаємозалежних компонентів.

Ситуацію не рятує навіть використання описів розгортання мовою XML, що використовується у компонентних і сервіс-орієнтованих платформах проміжного програмного забезпечення, оскільки в цьому випадку є семантичний розрив між метою розробки й вираженням цієї мети в тисячах рядків вручну зробленого XML, синтаксис якого не має відносини ні до семантики прикладної області, ні до мети розробки. Через наявність цих проблем індустрія програмного забезпечення наближається до гранично припустимої складності.

У той же час платформні технології нового покоління, наприклад, Web-Сервіси й архітектури продуктових ліній (product-line architecture) стають настільки складними, що роками опановують платформними API і паттернами використання й при цьому часто виявляються знайомими тільки із частиною можливостю регулярно використовувати платформи. Більше того, при використанні мов третього покоління програмісти змушені приділяти настільки велику увагу тактичним деталям імперативного програмування, що вони часто не можуть концентруватися на стратегічних архітектурних проблемах, таких як коректність системи в цілому і її продуктивність. Такі фрагментовані представлення проекту в цілому затрудняють програмістам розуміння того, які частини їхніх додатків є чутливими до побічних ефектів, що виникають при зміні вимог замовників і або середовища розробки. Часто це змушує програмістів робити неоптимальні рішення, у яких дублюються частини коду, порушуються ключові архітектурні принципи, ускладнюється розвиток системи й забезпечення необхідної якості.

Багатообіцяючим підходом, спрямованим на рішення цих проблем, є розробка технологій інженерії, керованої моделями, (Model-Driven Engineering, MDE). При використанні MDE розробка ведеться на предметно-орієнтованих мовах моделювання (Domain-Specific Modeling Language, DSML), у системах типів яких формалізується структура, поводження й вимоги додатку в середині відповідної предметної області. DSML описуються з використанням метамоделей, у яких визначаються

зв'язки між поняттями предметної області й точно специфікується основна семантика й обмеження, асоційовані із цими поняттями.

Для того щоб добре орієнтуватися в UML, тобто успішно використовувати його на практиці, треба мати досить глибокі представлення про методи об’єктно-орієнтованого аналізу, проектування й програмування.

Початковою метою розвитку UML було забезпечення більш точного опису мови моделювання - підтримка формальної основи для розуміння мови моделювання. Однак дотепер формальна семантика не є частиною стандарту. Огляд декількох підходів, що стосується визначення такої семантики вказує на використання підходів, де розглянуто теоретико-множинний, трансляційний, метамодельний підходи й запропонована так звана “вільна семантика”.

Першу категорію становлять зміни, що перетинають ієрархію подання моделі. Прикладом є ефект зміни пропускної здатності на якість обслуговування компонентів авіаційного електронного встаткування, які повинні відображати в реальному часі відео-потік. Щоб оцінити наслідки такої зміни, програміст вручну обходить ієрархію моделі, рекурсивно виконуючи фіксацію (click) по кожній підмоделі. Цей процес стомлюючий і приводить до помилки, оскільки моделі систем часто містять ієрархії глибиною в кілька рівнів.

Друга категорія змін включає збільшення масштабу частин моделі, що доставляє особливі занепокоєння при розробці великомасштабних вбудованих систем реального часу, що містять тисячі крупно модульних компонентів. Для цього типу зміни потрібне створення декількох модельних елементів і з'єднань між ними. При роботі з інструментом моделювання для масштабування базової моделі з декількох елементів до моделі з тисячами елементів потрібна велика кількість дій з мишею й клавіатурою. При виконанні цього процесу легко робляться помилки, наприклад, можна забути встановити з'єднання між двома задубльованими елементами. Зрозуміло, що ручне масштабування впливає не тільки на ефективність моделювання, але й на коректність представлення.

Обидві ці категорії еволюції змін істотно виграли б від автоматизації. Із цією метою розроблено узагальнений процесор трансформацій для маніпулювання моделями, названий як C-Saw (Constraint-Specification Aspect Weaver).

Для роботи зі змінами, що перетинають ієрархію, в C-Saw використовується кілька принципів аспектно-орієнтованого підходу. Комбінація трансформації моделі з компонуванням аспектів забезпечує

потужну технологію для швидкого перетворення успадкованих систем на основі високорівневих властивостей, що описується моделлю. Далі, шляхом застосування аспектно-орієнтованих методів і перетворення програм невеликі зміни на модельному рівні можуть ініціювати дуже великі трансформації на рівні вихідного коду.

Досвід показує, що ефективні механізми керування складністю автоматизують звичайні завдання розробки й забезпечують надійну підтримку поділу відповідальностей. Наприклад, сучасні високо-рівневі мови програмування й інтегровані середовища розробки забезпечують абстракції, що захищають програмістів від складних низько-рівневих деталей і підтримують автоматичне перетворення абстрактних представлень вихідного коду в дійсні форми, які виконує машина.

Досягнення в областях розробки програмного забезпечення й технологій обробки інформації привели до спроб створення більш складних програмних систем. Ці спроби демонструють неадекватність абстракцій, що забезпечені сучасними мовами високого рівня. Виникає потреба в мовах, моделях і технологіях, що підвищують рівень абстракції, на якому замислюються, створюються й розвиваються.

OMG (Object Management Group) відповідає на цю вимогу підготовкою версії 2.01 мови UML і ініціативою MDA (Model Driven Architecture). Проблеми, на рішення яких на початку були націлені розроблювачі UML 2.0, включали очевидне розпухання ранніх версій UML і відсутність правильно певної семантики.

В процесі розробки стандарту в число вимог була додана підтримка модельно-керованої розробки (MDD), а точніше, підходу MDA до MDD. Широка поінформованість про проблеми ранніх версій UML разом зі зростаючим інтересом до MDD породили надії на те, що розроблювачам UML 2.0 удасться зробити версію мови з істотно скороченим набором модельних понять для точного й зручного описів найрізноманітніших додатків.

Очікувалося також, що в цій версії буде бути точна семантика, що могла б полегшити автоматизацію, необхідну для просування MDD за межі традиційних можливостей існуючих CASE-засобів. Деякі люди очікували, що розроблювачам UML 2.0 удасться зробити срібну кулю мов моделювання або, принаймні, тісно наблизитися до цього.

Ці реальні проблеми вимагають рішення, але не слід дивуватися тому, що ця мова моделювання першого покоління далека від досконалості.

Як відзначають деякі розроблювачі UML, розробка мов моделювання усе ще переживає період становлення. Для прискорення процесу виявлення знань, пов'язаних з MDD, необхідна конструктивна критика UML. У цьому змісті UML 2.0 може зіграти важливу роль як явна форма експерименту з мовою моделювання, що може вивчатися, аналізуватися й критикуватися зацікавленими сторонами.

Захисники UML 2.0 відзначають, що в цьому стандарті представили кращий виробничий досвід застосування моделювання. Вони говорять про наступні основні вдосконалення:

1. Поліпшено підтримку UML як сімейства мов за рахунок використання профілів і крапок семантичних варіацій (semantic variation point), які позначають частини UML, які навмисне залишені без семантики, щоб можна було навантажити їхньою семантикою, обумовленої користувачами.

2) Поліпшено виразність моделювання, включаючи моделювання бізнес-процесів, підтримку класифікаторів повторного використання моделювання й підтримку архітектур розподілених неоднорідних систем.

3) Зроблено інтеграцію семантики дій (Action Semantics), що може використовуватися програмістами для визначення семантики моделей під час виконання й забезпечує семантичну точність, необхідну для аналізу моделей і їхньої трансляції в реалізації.

Занадто висока оцінка UML і відповідних технологій MDD може бути настільки ж пагубною, як і несправедлива критика.

9. 6. Критика й переваги UML

Незважаючи на те, що UML досить широко розповсюджений стандарт, що часто використовується , його критикують через наступні недоліки :

1. Надмірність мови. UML часто критикується, як невиправдано велику і складну мову. Вона включає багато надлишкових діаграм і конструкцій, що рідко використовуються.

2. Недостатня семантика. Тому що, UML визначена комбінацією себе (абстрактний синтаксис), OCL (мовою опису обмежень — формальної перевірки правильності) і Англійської (докладна семантика), тобто вона позбавлена скутості властивим мовам, тобто формальному опису. У деяких випадках абстрактний синтаксис UM і OCL і Англійської суперечать один одному, в інших випадках вони неповні. Неточність опису самої UML однаково відбивається на користувачах і постачальниках програмних продуктів, приводячи до несумісності інструментів через унікальне трактування специфікацій.

3. Проблеми при вивченні й впровадженні. Вищеописані проблеми роблять проблематичним вивчення й впровадження UML, особливо коли керівництво насильно змушує використовувати UML інженерів при відсутності в них попередніх навичок.

4. Тільки код відображає код. Ще одна думка — що важливо робочі системи, а не гарні моделі. Як лаконічно виразився Джек Ривс, «The code is the design» (англ. «Код і є проект»). Відповідно до цієї думки, існує потреба в кращій способі написання ПЗ; UML цінується при підходах, які компілюють моделі для генерування вихідного або здійсненного коду. Однак цього все-таки може бути недостатньо, тому що UML не має властивостей повноти по Тьюрінгу і будь-який згенерований код буде обмежений тим, що може розглянути або припустити інтерпретуючий UML як інструмент.

5.Кумулятивне навантаження/Неузгодженість навантаження (Cumulative Impedance/Impedance mismatch). Неузгодженість навантаження — термін з теорії системного аналізу для позначення нездатності входу системи сприйняти вихід іншої. Як у будь-якій системі позначень UML може представити одні системи більш коротко й ефективно, чим інші. Таким чином, програміст приходить до рішень, які більш комфортно підходять для використання сильних сторін UML і мов програмування. Проблема стає більш очевидної, якщо мова розробки не дотримується принципів ортодоксальної об’єктно-орієнтованої.

6. Намагається бути всім для всіх. UML — це мова моделювання загального призначення, що намагається досягти сумісності з усіма можливими мовами розробки. У контексті конкретного проекту, для досягнення командою проектувальників певної мети, повинні бути обрані можливості UML. Крім того, шляхи обмеження області застосування UML у конкретній області проходять через формалізм, що не повністю сформульований, і який сам є об'єктом критики.

Переваги UML:

1. UML об’єктно-орієнтована, у результаті чого методи опису результатів аналізу й проектування семантично близькі до методів програмування на сучасних мовах програмування високого рівня;

2 UML дозволяє описати систему практично із всіх можливих точок зору й різні аспекти поведінки системи;

3. Діаграми UML порівняно прості для читання після досить швидкого ознайомлення з його синтаксисом;

4. UML розширює й дозволяє вводити власні текстові й графічні стереотипи, що сприяє його застосуванню не тільки в сфері програмної інженерії;
5. UML одержала широке поширення й динамічно розвивається.

Отже, UML є потужним, гнучким засобом моделювання, опис стандарту якого є відкритим для наступного вдосконалювання. Неоднозначність як деяких конструкцій самої мови, так і підходів до його формальної семантики, наявність у специфікації неформальних описів вимагає подальшого розвитку формальної основи для повної й несуперечливої інтерпретації мови.

Питання для самоконтролю.

1. Що означає абревіатура UML ?
2. Перелічите методи об’єктно-орієнтованих мов моделювання.
3 Для рішення яких завдань призначена мова UML ?
4. На яких чотирьох рівнів абстракції ґрунтується формальний опис мови UML ?
5. Які логічні блоки є основою представлення UML на модельному рівні?
6. Які типи сутностей визначено UML ?
7. Які внутрішні механізми розширення використовують в UML ?
8. Які типи діаграма використовують в UML ?
9. Що означає інженерія програмного забезпечення з комп'ютерною підтримкою CASE ?
10. В чому суть технологій програмної інженерії MDE ?
11. Які категорії еволюції змін привели до розроблення узагальненого процесора трансформацій для маніпулювання моделями C-Saw ?
12. Які можливі вдосконалення мови UML 2.0 ?
13. За які недоліки критикують мову UML ?:
14. Які переваги мови моделювання UML ?

Розділ 10. Ознайомлення з обєетно-орієнтованою мовою програмування Microsoft Visual Basic.

10.1.Історія розвитку Visual Basic

У далекому 1970 році на березі озера в місті Сієтлі навчалися два товариша Біл Гейтс і Пол Аллен . Вони дуже любили возитися з комп'ютерами, написали багато комп'ютерних ігор, і найвідоміші з них – це Тіс-Тас-Тое (хрестики – нолики) і moon landing (місячна поверхня). Ці ігри були написані на комп'ютерній мові, яка мала назву **BASIC.**

BASIC – абревіатура початкових англійських слів: **B**eginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode (***Початкова Універсальна Символьна Система Команд).***

Коли в середині 1970-х був розроблений перший персональний комп'ютер Біл Гейтс і Пол Аллен мали досить знань, щоб успішно застосовувати мову BASIC в роботі з новими комп'ютерами. Вони почали продавати їхню версію мови BASIC всім бажаючим (за ціною $350 за програму, записану на магнітофонній касеті). На виручені таким чином гроші, вони створили компанію **Microsoft.** Саме так, перші програмні продукти всесвітньо відомої компанії Microsoft були написані мовою BASIC. З того часу компанія Microsoft випустила інші версії мови BASIC, які називаються GW-BASIC, QuickBasic і QBasic, а останні новітні версії називаються **Visual Basic.** .Microsoft Visual Basic — засіб розроблення програмного забезпечення, створений і підтримуваний корпорацією Microsoft, який складається з мови програмування і середовища розроблення. Мова Visual Basic успадкувала дух, стиль і частково синтаксис свого предка - мови Бейсік, в якої є чимало діалектів. У той же час Visual Basic поєднує в собі процедури та елементи об'єктно-орієнтованих та компонентно-орієнтованих мов програмування. Середовище розробки VB містить інструменти для візуального конструювання користувальницького інтерфейсу.

Visual Basic одночасно улюблений, і зневажений багатьма програмістами. Visual Basic вважається ідеальним засобом швидкої розробки прототипів програми, розробки додатків баз даних і взагалі для компонентного способу створення програм, що працюють під управлінням операційних систем родини Microsoft Windows.

Перше визнання серйозними розробниками Visual Basic отримав після виходу версії 3 - VB3. Остаточне визнання як повноцінного засобу програмування для Windows - при виході версії 5 - VB5. Версія VB6, що входить до складу Microsoft Visual Studio 6.0, стала по-справжньому зрілим і функціонально багатим продуктом. Після цього розробники з Microsoft суттєво змінили напрямок розвитку даної технології.

Visual Basic.NET не дозволяє програмувати по-старому, бо по суті є зовсім іншою мовою, такою самою, як і будь-яка інша мова програмування для платформи .NET. Індивідуальність мови і її переваги (простота, скромність створення програм, легкість використання готових компонент) при використанні в середовищі .NET не мають такого значення, як раніше - усе зосереджено на можливостях самої системи .NET, на її бібліотеці класів. Тому в даний час треба говорити про

класичний Visual Basic, його діалекти Visual Basic for Applications (VBA) і Visual Basic Scripting Edition (VBScript) і про мову для платформи .NET - Visual Basic.NET.

Розглянемо основні етапи розвитку Visual Basic.

Травень 1991 - випущений Visual Basic 1.0 для Microsoft Windows. За основу мови був узятий синтаксис QBasic, а нововведенням, що принесло потім мові величезну популярність, стала основа зв'язку мови та графічного інтерфейсу. Цей принцип був розроблений Аланом Купером (Alan Cooper) і реалізований в прототипі Tripod (також відомому як Ruby). Перший Visual Basic був інтерпретатором.

Вересень 1992 - випущений Visual Basic 1.0 під DOS. Він не був повністю сумісний з Windows-версією VB, оскільки по суті був наступною версією QuickBASIC і працював у текстовому режимі екрана.

Листопад 1992 - випущений Visual Basic 2.0. Середовище розробки стало простішим у використанні і працювало швидше.

Влітку 1993 - вийшов у світ Visual Basic 3.0 у версіях Standard і Professional. До складу цієї версії увійшов додавток для роботи з базами даних Access.

Серпень 1995 - Visual Basic 4.0 - версія, яка могла створювати як 32-х так і 16-розрядні Windows-програми. Крім того, з'явилася можливість писати на VB класи, а також Visual Basic нарешті став повноцінним компілятором, що значно збільшило швидкість виконання програм.

Лютий 1997 - Visual Basic 5.0 - починаючи з цієї версії, стало можливо поряд зі звичайними застосунками (додатками) розробляти COM-компоненти. Скомпілювавши такий компонент у OCX-файл і поставляючи його, можна надати свій об'єкт управління не тільки кінцевому користувачеві, але й іншим розробникам, після чого можна інтегрувати цей об'єкт в свої додатки.

У середині 1998 - вийшла остання версія - Visual Basic 6.0. Після цього Microsoft різко змінила політику стосовно мов сімейства Basic. Замість розвитку Visual Basic було створено абсолютно новий мова Visual Basic.net, перша версія якого з'явилася в 2001. Це принципово нова мова, що має, крім синтаксису, дуже мало схожого з VB 6.0; та відрізняється від нього так сильно, як у свій час VB відрізнявся від QBASIC. VB.NET відразу ж зайняв місце VB в Visual Studio, і на цьому розвиток класичного Visual Basic зупинився.

10.2. Основні різновиди Visual Basic

Класичний Visual Basic (Visual Basic Classic версії 5-6). Ця мова дуже сильно прив'язана до свого середовища розроблення й до

операційної системи Windows, оскільки вона є виключно інструментом написання Windows-додатків. Прив'язаність до середовища полягає в тому, що існує велика кількість засобів, призначених для допомоги й зручності програмування: вбудований відладчик, перегляд змінних і структур даних, вікно налагодження, спливаюче підказування при наборі тексту програми (Intellisense). Всі ці переваги роблять марним і навіть неможливим використання Visual Basic поза середовищем для розроблення, наприклад, у звичайному текстовому редакторі.

Visual Basic for Applications (VBA). Це засіб програмування, практично нічим не відрізняється від класичного Visual Basic, що призначене для написання макросів та інших прикладних програм для конкретних програм. Найбільшу популярність здобув завдяки своєму використання в пакеті Microsoft Office. Широке розповсюдження Visual Basic for Applications в поєднанні з споконвічно недостатньою увагою до питань безпеки призвело до значного поширення макровірусов.

Visual Basic Scripting Edition (VBScript). Це скриптова мова, що є дещо урізаною версією звичайного Visual Basic. Використовується в основному для автоматизації адміністрування систем Windows і для створення сторінок ASP та сценаріїв для Internet Explorer.

Розглянемо переваги та недоліки Visual Basic.

Переваги. Висока швидкість створення програм із графічним інтерфейсом для MicroSoft (MS) Windows. Простий синтаксис, що дозволяє дуже швидко освоїти мову. Можливість як компіляції в машинний код, так і інтерпретації під час налагодження.

Недоліки. Підтримка операційних систем тільки сімейства Windows (Виняток — VB1 for DOS). Відсутність механізму успадкування об'єктів. Існуючі в мові спадкування, дозволяє успадковувати тільки інтерфейси об'єктів, а не їх самих. Таким чином, в успадкуванні класів повинні бути явно переписані всі функції базового класу.

*Критика.Ч*асто критикують такі аспекти Visual Basic, як можливість відключити засоби спостереження за оголошеним змінними, можливість неявного перетворення змінних, наявність типу даних "Variant". На думку критиків, це дає можливість писати вкрай поганий код. З іншого боку, це можна розглядати як плюс, тому що VB не нав'язує "хороший стиль", а дає більше свободи програмісту. Відсутні покажчики, низькорівневий доступ до пам'яті, ASM-включення. Попри те, що парадигма Visual Basic дозволяє середньому VB-програмісту обходитися без усього цього, перераховані речі також нерідко стають об'єктами критики. І хоча, використовуючи недокументовані можливості і певні хитрощі, все це

можна реалізувати і на VB; користуватися цими трюками набагато складніше, ніж, наприклад, на C++.

Одже, Visual Basic – це одна з найбільш простих для вивчення мов програмування. Але, не дивлячись на це, з її допомогою можна створювати досить складні комп'ютерні програми. Вивчившись програмувати на Visual Basic від Microsoft, можна створювати свої власні програми.

10.3. Знайомство з середовищем програмування Visual Basic

Для написання программ мовою програмування можна застосовувати текстовий редактор, наприклад, Блокнот . Але зручніше користуватись середовищем програмування.

Середовище програмування – комп'ютерна система, призначена для створення програм. До складу системи входять: текстовий редактор для введення та редагування тексту програм, компілятор, налагоджувач програм тощо.

Інтегроване середовище програмування Visual Basic включає набір меню, панелей і вікон, службові програми, довідкову систему, що у сукупності утворюють робоче місце програміста.

Розробка додатку на VB складається з наступних етапів:

1) продумати програму (продумати, що саме програма повинна робити, які саме задачі повинна вирішити, реалізувати їх, продумати структуру даних тощо).

2) проектування інтерфейсу, тобто розміщення на формі потрібних керуючих елементів, кнопок, списків тощо. Цей етап називають складанням кістяка програми.

3) написання програмного коду, який поєднує розміщені на формі керуючі елементи.

4) налагодження програми. Цей етап часто займає більше часу, ніж попередні.

5) остаточна компіляція і, якщо це необхідно, створення інсталяційного файлу setup.exe.

Середовище програмування Visual Basic IDE є інтегрованим, тому що в ньому можна виконувати різні дії при розробці програмного продукту, такі, як проектування і опис складовий частин програми, редагування програмного коду, компіляцію усіх елементів додатку у виконуваний файл, відлагодження додатку.

Для запуску Visual Basic:

1) нео бхідно виконати фіксацію по кнопці *Пуск* на панелі задач Windos..

2) в меню, що відкрилося, вкажіть на рядок *Програми* і, потім перейдіть на рядок Microsoft Visual Studio 6.0.

3) виконайте фіксацію по рядку Microsoft Visual Basic 6.0.

Це приведе до появи діалогове вікно *New Project (Новий проект*), де потрібно вказати тип проекту, що створюється (наприклад, Standart EXE Рис.10.1). Натиснувши кнопку *Open (Відкрити)* можна потрапити в середовище розробки Visual Basic IDE. Це приведе до появи головного вікно *(Main Window),* яке використовується для управління створенням проекту і запуску створених програм.

Головне вікно має *Рядок заголовка (Title Bar)* у верхній частині. У рядку заголовка вказується ім'я проекту і поточний режим роботи Visual Basic Нижче рядка заголовка знаходиться *Головне меню* (Main *Menu),* необхідне для доступу до різних функцій Visual Basic. Рядок меню дозволяє керувати всією роботою по створенню програмного продукту. У рядку меню знаходяться *«випадаючі»* меню, через які здійснюється управління середовищем розробки Visual Basic. Під головним меню знаходиться Панель інструментів *(Toolbar)*. На панелі інструментів знаходяться кнопки із зображеннями, що призначені для швидкого доступу до деяких команд меню. При цьому покажчик курсору миші на кнопку панелі інструментів вказує на спливаюче підказування (tooltip), що повідомляє про призначення даної кнопки. Майже всі додатки Windows мають панель інструментів з підказками для розв'язування різних задач.

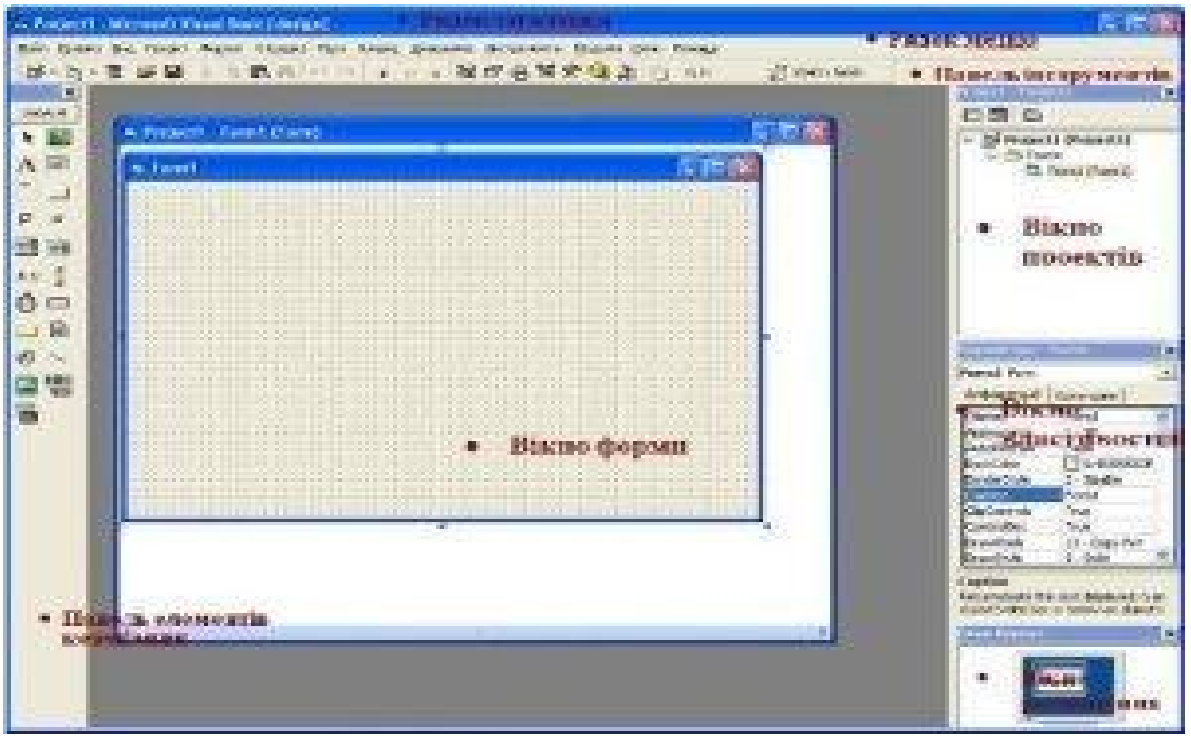


Рис.10.1. Діалогове вікно New Project (Новий проект),

Form Desiner*(Дизайнер Form)*. Ця форма є основним елементом управління при розробці проекту в Visual Basic, Форми основні будівельні блоки програм на Visual Basic - представлені у вікні дизайнера форм. Вікно *Form* є початком потрібно програми і призначене для редагування форм, тобто додавання та вилучення з них різних елементів управління. Форма є основною при створенні проекту в Visual Basic.

Елементи управління (Controls), що розміщені на формі, дозволяють взаємодіяти користувачу і комп'ютеру. Користувач робить події (events), керуючи роботою комп'ютера.. Кожна відкрита форма має свій дизайнер форм, який у середовищі розробки звичайно розташований у центрі. Його можна переміщати по екрану, збільшувати, зменшувати, змінювати його властивості у вікні властивостей, упорядковувати на ньому об'єкти й у вікні *Code* писати для нього процедури. Якщо вікна форми не видно на екрані, необхідно виконаьт фіксацію по *View* у головному меню, потім натиснути *Object Toolbox (Панель елементів управління).*

Панель *TooIBox* використовується для додавання у форми проекту різних елементів. Тут можна вибрати об'єкти, які потрібно помістити на форму. Наприклад, щоб намалювати лінію, необхідно виконати фіксацію на символ лінії. Потім виконати фіксацію на тому місці форми, де повинна починатися лінія, і перемістти курсор кнопку з натиснутою кнопкою туди, де ця лінія повинна кінчатися.

У вікні *Project Explorer* відображаються ті проекти, над якими необхідно працювати в даний момент, а також різні частини кожного з них. *Properties Window (Вікно властивостей)* призначене для перегляду й установки вихідних значень деяких параметрів (властивостей) елементів управління. Список, що розкривається, у верхній частині вікна містить усі елементи управління, що розташовані на формі, з якою в даний момент працюють.

Перелік вдастивостей обраного об'єкта пропонується в двох виглядах: *Alphabetic (Заалфавітом) і* Categorized *(По категоріях)*. Один з об'єктів завжди є «актуальним об'єктом», властивості якого будуть перелічені у вікні властивостей. Об'єкт стає актуальним, при фіксації по ньому.

Для налагодження властивостей об'єкта призначена спеціальна панель *Propertie*s*:*

1. Виберіть об'єкт, для якого потрібно змінити настройку.
2. Потім виберіть потрібну властивість у вікні *Properties*.

3. Задайте цій властивості потрібне значення.

Якщо ви не бачите на екрані вікно властивостей, то необхідно клацнути *View* у головному меню, потім *Properties Window*.

Form Layout *(Вікно* розміщення форм) призначене для точного позиціонування форми на екрані при виконанні додатку. Щоб задати для форми яку-небудь область, необхідно у віщі розміщення форм за допомогою миші перемістити її зображення в задане місце

У Visual Basic можна створити як об'єктні, так і керуючі програми, що підтримують функцію Automation (автоматизація). Програми для Windows, повністю підтримують функцію Automation, дозволяють здійснювати доступ до своїх функцій у вигляді набору об'єктів з відповідними їм властивостями і командами.

10.4. Створення нового проекту.

Для того, щоб створити новий проект, у меню *File* необхідно виконати команду *New Project (Новий проект)*. На екрані з'явиться діалогове вікно *New Project (Створити проект)*. Необхідно виділити значок *Standard Exe*, а потім виконати фіксацію по меню діалогово вікна кнопцки *«Відкрити»*. При створенні проекту на екрані з'являється порожня форма заготівки вікна майбутньої програми.

Для запуску проекту на виконання нобхідно вибрати :

1. спосіб. *Run - Start*

2. спосіб. *Кнопка "►"* панелі інструментів

3. спосіб. *Натискання клавіші F5.*

Зупинити виконання проекту можна двома способами:

1 спосіб. Натиснути на кнопку панелі інструментів. Виконання проекту буде зупинене, і буде повернення в режим розробки.

2 спосіб. Натиснути на кнопку *(Закрити)* на самій формі.

Для збереження проекту необхідно виконати такі три кроки:

1. Створення папки для проекту.
2. Збереження файлу форми всередині папки проекту.
3. Збереження файлу проекту в папці проекту.

Щоб зберегти проект, який створено, або обновити вже існуючий шляхом зроблених доповнень, у меню *File (Файл*) необхідно виконати команду *Save Projec (Сохранить проект)*. Якщо проект зберігається вперше, то після цього необхідно вказати імена для всіх компонентів (форм, модулів), що складають проект, а також дати ім'я створеному проекту.

Для створення *EXE-файл необхідно* в меню *File (Файл)* виконати команду *makeProjec.exe(Делать проект.exe*). Після цього потрібно дати ім'я створеному проекту та вибрати папку для збереження.

Для закриття Visual Basic і закінчення роботи над проектом необхідно , необхідно вийти з Visual Basic відкривши пункт меню *File (Файл)* у головному меню. Перейдіть на рядок *Exit (Вихід)* і закривати всі свої вікна.

Етапи розробки програми на Visual Basic:

1.Створення інтерфейсу майбутньої програми (з використанням вікна *Конструктор форм*). Це перший крок у створенні прикладної програми. Тут вирішується, який набір форм і керуючих об'єктів (вікна, кнопки, меню, лінійки прокрутк тощо) зробить зручним взаємодія з програмою.

2. Завдання значень властивостей об'єктів графічного інтерфейсу (вікно *Вл активості об'єкта*);

Кожен з керуючих об'єктів, що застосовуються у програмі, має набір властивостей, які визначають їх зовнішній вигляд і поведінку. На цьому кроці задаються значення властивостей кожного вибраного керуючого об'єкта.

3.Створення програмного коду (вікно *Програмний код* або *Редактор код*);

4. Збереження проекту: *File - Save Project As ...*

5. Компіляцію проекту до програми (додаток)

Компіляція - це переклад слів мови програмування в цифровий код, який розуміє процесор.

Етапи проектування форми — створення екранної форми. Цей етап називають також *розробкою інтерфейсу* (Interface — буквально: сполучення (англ.).

Розробка інтерфейсу складається з таких кроків:

1) створення ескізу екранної форми;

2) вхід у середовище проектування Visual Basic;

3)створення екранної форми й установка значень властивостей цієї форми;

4)створення на формі об'єктів керування й установка значень властивостей цих об'єктів.

Розглянемо приклад створення нового проекту. Для цього необхідно вибрати значок *StandardEXE* в закладці *New*, ознайомитися з розташуванням панелей і вікон середовища та оголосимо проект . назвою «Об’єм», вибравши пункт Project1 вікна Project (Ртс.10.2).

Рис. 10.2. Проект «Об'ємг»,

Створимо для цього проекту екранну форму за зразком, що містить 6 об'єктів класу *Метка,* 4 об'єкти класу *Текстове поле* і по 1 об'єкту класу *Зображення* і *Командна кнопка* (рис. 10.3.) та збережемо проект в окремій папці «Об'єм

Крім текстових полів, на екранній формі повинні бути всі необхідні для користувача *пояснення*. (Точніше, дуже бажано, щоб вони були.) У вигляді *заголовка*, *написів*, *розрахункової формули* і навіть невеликого *креслення*.

Слід не забути і про такий важливий елемент як *командна кнопка*. Її натискання повинне служити сигналом для розрахунку по формулі і видачі на екран результату

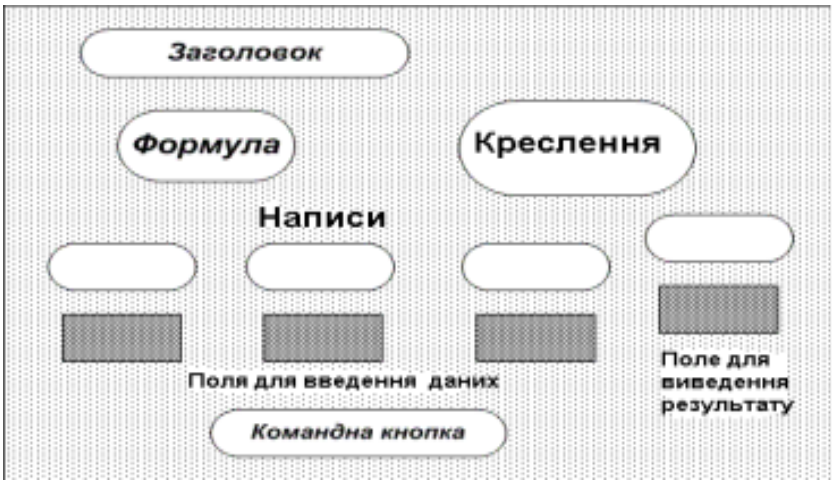


Рис 10.3. Ескіз екранної форми (вікна додатка)

Вхід у середовище проектування Visual Basic Тепер, коли ескіз вікна майбутнього додатку готовий, можна приступати до роботи на комп'ютері. При цьому необхідно переконатися в тому, що на комп'ютері

уже встановлена система проектування Visual Basic, і відомо, як ввійти в цю систему і почати в ній працювати.

Розглянемо створення екранної форми й установку значень властивостей цієї форми Якщо на *Головній панелі проекту* немає Вікна екранної форми, то необхідно відкрити його. Зробити це можна, вибравши команду 

меню ***View***.

Екранна форма є першим об'єктом, з яким починають працювати**.** Розглянемо створення на екранній формі об'єктів керування й установка значень властивостей цих об'єктів

 — піктограма цього інструмента.

Розглянемо технологію створення міток на прикладі розміщення на екранній формі однієї з них.

Необхідно виконати фіксацію курсором миші по піктограмі ***Label*** у вікні ***Toolbox***. Потім помістити покажчик миші в те місце екранної форми, де буде знаходитися ***лівий верхній кут*** майбутнього об'єкта. При натиснутій лівій клавіші миші необхідно перемістити покажчик у те місце, де буде знаходитися ***правий нижній кут*** майбутнього об'єкта. Цей процес буде супроводжуватися розтяганням пунктирної рамки — границі створюваного об'єкта.

Розміри і положення прямокутника, що з'явився, можна відредагувати за допомогою прямокутника, який оточений вісьмома маленькими квадратиками — ***маркерами***.

Треба створити об'єкти ***Мітка,*** пересіщаючи курсор миші, потім перейти до розміщення ***текстових полів***.

У ***Вікні інструментів*** потрібно знайти піктограму інструмента, за допомогою якого можна створити будь-яку кількість об'єктів керування класу ***Текстове поле***. Це інструмент ***TextBox***.

 — піктограма цього інструмента.

Не забувши залишити невеликий простір екранної форми для креслення і для командної кнопки, розмістіти 4 об'єкти ***Текстове поле*** для вхідних даних і для результату. Цей процес зовсім аналогічний створенню об'єктів ***Мітка***.

Точно таким же способом необхідно помістити на екранну форму ще два об'єкти: об'єкт ***Командна кнопка***та об'єкт ***Зображення***. (Рис.10.4). Використати для цього мож інструменти ***CommandButton*** та ***Image***.

 — піктограма інструмента ***CommandButton***.

 — піктограма інструмента ***Image***.

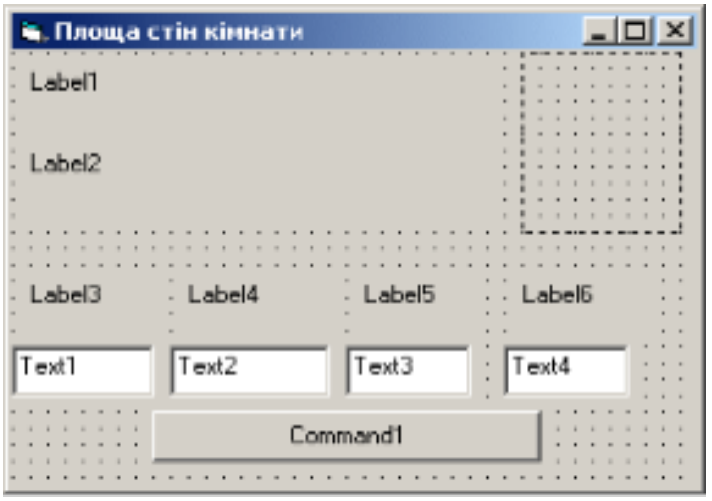


Рис.10.4.Вікно* Object *з екранною формою додатка

Останній з розміщених на формі об'єктів оточений вісьмома маркерами. Це об'єкт ***Зображення***. Тут буде знаходитися креслення — прямокутного паралелепіпеда.

А тепер необхідно подбайти про наповнення створених об'єктів конкретним змістом. Цей зміст, визначається значенням їхніх властивостей.

Слід зауважити, що в кожен момент проектування тільки один об'єкт є ***виділеним*** (***активним)***

У *Вікні властивостей* на *Головній панелі* можна побачите список властивостей активного об'єкта. Варто тільки виконати фіксацію курсором миші по іншому об'єкті екранної форми, як до цього об'єкта переходить вся активність і миттєво змінюється вміст ***Вікна властивостей***!

Можна наповнювати об'єкти змістом (тобто задавати значення їхніх властивостей) один за одним. Для цього можна по черзі їх активувати і працювати з ***Вікном властивостей*** кожного активного об'єкта.

Можна почати з ***міток***. (Усі написи, заголовок і формулу можна розмістіть в шести таких об'єктах.)

Спочатку, як і для екранної форми, можна визначити для них значення властивості (***Name***).

Іншою важливою властивістю об'єкта ***Мітка*** є ***Caption*** — напис, що буде розміщений в тому місці екранної форми, куди цю мітку поміщено..

В таблиці 10.1. вказано ім’я міток та значення їх властивостей.

Таблиці 10.1. Імена міток та значення їх властивостей.

Ім'я мітки по замовчуваню	**Значення властивості (*Name*)**	**Значення властивості *Caption***
Labell	Мітка1	Площа стін кімнати, обраховується за формулою
Label2	Мітка2	S = 2 * (А + У) * Н
Label3	Мітка3	Довжина (А):
Label4	Мітка4	Ширина (У):
Label5	Мітка5	Висота (Н):
Label6	Мітка6	Площа стін (S):

На закінчення необхідно встановити значення властивості об’єкта ***Зображення*** — властивості ***Picture*** На рис..10.5. відображено вигляд екранної форми після установки значень властивостей.

Висновок. Microsoft Visual Basic (VB) перетворився в самий популярний у світі інструмент розробки додатків. Більше того, знання його основ сьогодні фактично є обов'язковим для всіх програмістів, якими б засобами вони не користувалися. Це пояснюється двома взаємопов'язаними факторами. *По-перше*, VB дуже широко поширений і використовується не тільки в якості самостійного програмного середовища, але і у вигляді системи програмування, вбудованої в численні прикладні програми (зокрема MS Office). *По-друге*, перетворившись на серйозний професійний інструмент, VB залишився дуже зручним засобом для навчання програмування і рішення невеликих завдань.

Рис.10.5.Вигляд екранної форми після установки значень властивостей об'єктів

Питання для самоконтролю.

1. Що означає абревіатура BASIC ?
2. Перелічите основні етапи розвитку Visual Basic.
3. Які Ви знаєте основні різновиди Visual Basic.
4. З яких етапів складається розробка додатку на VB ?:
5. Опишіть діалогове вікно New Project (Новий проект),
6. Вкажіть послідовність виконання етапів для налагодження властивостей об'єкта за допомогою спеціальної панелі *Properties* у VB ?
7. Як створити новий проект ?
8. Опищість послідовність дій для запуску проекту на виконання ?
9. Які етапи розробки програми на Visual Basic ?
10. З кроків складається розробка інтерфейсу Visual Basic ?
11. Як вионати вхід у середовище проектування Visual Basic ?
12. Опишіть основні піктограми *у* вікні інструментів Visual Basic ?
13. Для чого використовується вікні властивостей ?
14. Чому знання основ VB є обов'язковим для програмістів ?

Розділ11. Застосування VBA в додатках MS Office.

11.1. Короткий опис VBA

Visual Basic for Applications (VBA) являє собою новаторський досягнення в мовах програмування, можливо найбільш значне з тих пір, як відбувся випуск першої інтегрованого середовища розробки. Це

означає, не те що VBA - кращий засіб розробки, а лише те, що VBA існує. Visual Basic, ймовірно, найпопулярніший засіб розробки, і тепер зовсім несподівано ця мова включена в не менш популярний комплект прикладних програм.

Вбудовування VBA в якийсь додаток означає для нього створення об'єктної моделі, в результаті VBA-програма може звертатися до об'єктів цього додатка і маніпулювати ними.

Запровадження досить потужногї мови програмування робить цю програмну платформу дуже привабливою для професіоналів, які займаються розробкою спеціалізованих прикладних систем. Розробка мови програмування VBA, що вбудовується в прикладні системи, є одним із стратегічних напрямів компанії Microsoft.

Щоб пояснити різницю між VBA і автономною версією Visual Basic , потрібно розділити дві складові:

1. Ядро мови Visual Basic.
2. Об’єкти, таких, як кнопки, вікна списків і керуючих елементів ActiveX, які доступні при програмуванні.

Ядро мови залишається таким самим як для автономною версією Visual Basic, так і для VBA. В той же час, мова – це набір об’єктів, склад яких змінюється. Кожний додаток, що має доступ до VBA, містить свій власний набір об’єктів, який можна використати в своєму коді. Наприклад, Word включає об’єкти Document (Документ) і Paragraph (Абзац). Як тіольки отримано навики роботи з цими об’єктами, можна робити з ними практично усе що завгодно в потрібному коді VBA.

Слід відзначити одну суттєву річ. Якщо в ситемі встановлено пакет Microsoft Office, то усі об’єкти цього пакету доступні до будя-якої версії Visual Basic.

VBA містить редактор форми і налагоджувальник, які в основному подібні на тіж інструментальні засоби автономномної версії Visual Basic, але мають декілька відмінностей за своїми властивостями. У VBA не можна використати компіляцію коду до автономного EXE-файлу. Крім того, VBA не має можливості компіляції до безпосередніх інструкцій процесора. Тут також немає елемента управління Data, але залишається можливість доступу до баз даних з коду VBA.

VBA має і унікальну можливість : є можливість почати програмування з запису макросів, а це означає, що Office самостійно пише потрібний код.

Встановлення VBA. Версія VBA встановлюється автоматично під час встановлення Office. Щоб встановити файли довідки для VBA, необхідно запустити програму Office і вибрати команду *Add/Remove Components*

(*Добавити/Вилучити компоненти)* . В розділах *Help filand end samples (Справка файлів і прикладів)* для Word і Excel необхідно виділити пунк *Help for Visual Basic* (*Справка по Visual Basic).* Для Access цей пункт називається *Programming Help (Справка по прграмуванню).* Кожний додаток встановлює правку для своїх об'єктів, причому довідникові дані будуть різними.

Якщо необхідно писати у VBA програми для баз даних, то потрібно встановити об'єкти доступу до даних *Data Access Objectsfor Visual Basic.* Цей пунк знаходиться в розділу *Data Access* (*Доступ до даних)* програми встановлення Office.

Слід зауважити, файли VBA мають значний об'єм, тому не потрібно встановлювати їх для додатків встановлювати, якшо немає потреби їх використовувати при написанні програми. Нприклад, встановити довідку для Word, але не встановлювати для інших додатків Office.

VBA має власне середовище програмування, яке зберігається для усіх додатків Office. Щоб увійти в VBA небхідно виконати таку послідовність команд *Tools>Macro> Visual Basic Editor (Сервіс>Макрос>Редактор Visual Basic).*

Інший варіат запуску VBA – вивести список макросів, вбравши таку послідовність команд: *Tools>Macro>Macros>(Сервіс>Макрос>Макрои>.* Вибрати потрібний макрос і виконати фіксацію по кеопці *Edit (Змінити)* для запуску VBA або виконати фіксацію по кнопці *Create (Створити),* щрб почати новий макрос.

11.2. Структура проекту у VBA,

Програми на VBA зберігаються в проектах, які містит модулі різних типів, а модулі включають різні процедури та макроси.

Поняття макроса. Програми пакету MicrosoftOffice мають широкий спектр функцій для створення різноманітних офісних завдань. Проте чсто для багатьх рутиних задач приходиться виконувати одні і ті ж послідовності коман. Наприклад, у текстовому редакторі Word небхідно оформлювати текст курсивом, разміром кегля 14, гарнітурою шрифта Arial.

В цьому випадку доцільно цій послідовності команд присвоїти відповідне ім'я, створивши таким чином макрокоманду, яку називають *макросом*. У VBA,передбачена можливість утвореня макросі за допомогою *макрорекордера*, засобу, який записує всі дії користувача, в тому числі помилки і неправильні запуски тих чи інших процедур. Коли програма відтворює макрос вона виконує кожну записану операцію в послідовності, виконаній під час запису.

Слід зауважити, що перші макрорекордери мали суттєві недоліки: якщо в процесі запису макроса було допущена помилку, то для її усунення необхідно було повторення запису усієї макрокоманди, оскільки небуло жодної можливості її виправити, відредагувати послідовність операцій. Розвиток мови Visual Basic та її версій, удосконалити процес запису макросів. Тепер макроси у VBA можна не лише редагувати, але й удосконалювати їх та поліпшувати.

Поняття про процедури. Макроси – це надзвичайно потужний інструмент для розробки додатку. Проте макрокоманди не дозволяють редагувати значення параметрів з однієї процедури в іншу. Крім того, макроси не дозволяють запрограмувати реакцію системи на можливу помилку. Для вирішення таких проблем необхідно скористатися можливостями мови VBA, а саме – створювати послідовності команд, не використовуючи їх автоматичний запис. Процедури, створені засобами VBA, на відміну записаних за допомогою макрорекордера, можуть реагувати на зіну умов. Мова програмування VBA, дозволяє створювати процедуру, яка перевірятиме різноманітні умови та перевірятиме ту послідовність команд, яка повинна виконуватися залежно від значення в залежності від тієї чи іншої умови. Крім того, макрорекордер не дозволяє передбачити можливість повторення записаної процедури декілька разів. Для цього користувачеві необхідно щоразу в ручну запускати відповідний макрос. Конструкції мови VBA, дозволяють сунути такі незручності. Крім цього, використання можливостей мови VBA, дозволяє об'єднувати декілька макросів в єдине ціле, створювати меню користувача, діалові вікна та панелі інструментів, які налаштовують та змінюють інтерфейс відомих програм пакету MicrosoftOffice залежно від потреб додатку для користувача.

Існує три типи процедур:

1. Процедура Sub - набір команд, з допомогою якої можна вирішити певну задачу. При її запуску виконуються команди процедури, а потім управління передається в додаток пакету MS Office або процедуру, яка викликала дану процедуру.

2. Процедура Function (функція) також уявляє собою ідно набір команд, яякий вирішує певну задачу. Відмінність полягає в тому, такі процедури обов'язково повертають значення, тип якого можна описати при створенні функції.

3. Процедура Property використовується для посилання на властивості об'єкта. Даний тип процедур застосовується для встановлення або отримання значення користувацьких властивостей форм і модулів.

Одже, інструментарії, що написані мовою VBA прийнято називати *процедурами,* а створені макроси можна легко включати в ці процедури VBA.

Поняття про модулі. Макроси і процедури зберігаються в спеціальній частині файла даних, яку нахивають *модулем* VBA. Модуль містить програмний код – текстове подання інструкцій. Кожен фай даних може не містити або містити один або декілька модулів. Модулі, що збережені в одному документі, називають *проектом.* Проект може містити декілька модулів. Існує три типи модулів:

1. *Стандартні модулі* - це модулі, в яких можна описати доступні у проекті процедури;
2. *Модулі класу* містять опис об'єкта, який є членом класу. Процедури, що написані в модулі класу використовують тільки в цьому модулі. Серед модулів класу виділяють модулі форм і звітів, які зв'язані з конкретною формою або звітом. Модулі форм і звітів часто містять процедури обробки подій, які у відповідь на подію в форме або звіті отчете. Процедури обробки подій використовуються для управління поведінкою форм і звітів та їх реакцію на дії користувача.
3. *Модулі містять описи і процедури* - набори описів і інструкцій, що згруповані для виконання.

У VBA проект - це сукупність усіх файлів, що складають програму, включаючи форми, модулі, графіку, елементи управління, список усіх використовуваних програмою файлів, назва проекту конфігурація IDE для роботи над даним проектом тощо. Проект має модульну структуру. Модулі бувають трьох типів: модуль форми, модуль класу, стандартний модуль.

Ми будемо мати справу з проектами, які складаються із однієї форми, будуть містити тільки модуль форми. Кожен файл форми містить оголошення змінних, констант, типів даних, процедур обробки подій. Опис тиу файлу представлено в таблиці 10.1.

У VBA підтримується наступна структура програми. На вищому рівні ієрархії стоїть додаток, далі йдуть проекти, пов'язані з фактичними документами цієї програми, на третьому рівні знаходяться модулі (модулі програми, модулі користувача, модулі класу, модулі форм і модулі посилань). А на останньому рівні знаходяться процедури і функції цих модулів. Дана структуризація програм повністю, задовольняє принципам структурного і модульного програмування.

В основі програмування в VBA є розуміння поняття об'єкту, особливо коли справа стосується утворення користувацьких діалогових вікон і використання можливостей VBA-додатків.

Табиця 10.1. Опис типу файлу

Тип файлу	Опис
.FRM	Форма
.BAS	Модуль
.FRX	Файл для графічних елементів
.OCX	Елемент управління ActiveX
.CLS	Модуль класу
.VBP	Проект Visual Basic

Об'єктом називають будб-яку іменовану сутність, що має:
1. *Властивості* - це характеристики об'єкта. Кожна властивість зберігає інформацію про деякий аспект зовнішньог виду, поведінки, цього об'єкта. Головною задачою властивостей є опису деякої характеристики об'єкта.
2. *Методи* – це іменовані дії, які об'єкт можуть виконати по команді. Із за того, що будь-який метод є невідємною частиною об'єкта, об'єкт сам знає, що йому робити, коли викликається метод. Таким чином, методи – не що інше, як процедури, прив'язані до конкретного об'єкту. Щоб викликати метод, необхідно надрукувати ім'я об'єкта печатать, крапку, а потім ім'я методу.
3. *Подія* уявляє собою дещо те, що сталося з об'єктом, і те, на що об'єкт може відповісти наперед передбачити відповідні дії. До події можна віднести наступне:
1) фізичні діє користувача програми, наприклад фіксацією клавіші мишки, переміщення курсора тощо;
2) ситуації, в які попадає об'єкт в ході виконання програми.

У VBA є об'єкт – спеціального призначення, який називають *колекцією*. Колекції призначені для спрощення роботи з набором об'єктів, коли цей набір об'єктів потрібно використати як одне ціле. Як правило, , всі об'єкти в колекції мають один і той же тип. Наприклад, колекція *Pages* складається з об'єктів *Page.*

Але в VBA існує родовий об'єкт *Collection*, призначений для збереження в ньому об'єктів будь-яких типів в будь-якій комбінації.

Формою називають будь-яке створене в VBA користуваць вікно. Офіційно форми в VBA описується в термінах об'єкта инах *UserForm.* Кожний об'єкт *UserForm* належить одночасно временно двом колекциям

об’єктів: VBA-проекту, в якому зберігається форма, і колекції *UserForms*, що містить всі форми, завантажені програмою.

Мова VBA є об’єктно- орієнтованою. Це означає, що багато її команд має особливий формат. Типова команда VBA має вигляд:

<Об’єкт>.<Об’єкти, що входят в перший об’єкт>.<...>.<Той об’єкт , з яким потрібно провести дії >.<собственно действие>

Ішими словами, кажна команда пишеться ніби з «кінця»: спочатку визначається те, над чим потрібно виконати дії, – об’єкт, а потім саму дію – метод. Разподілом компонентів команди є знак «крапка».

Наприклад:

Application.activDocument.PageSetup.Orientation=wdOrientLandscape

Ця команда встановлює альбомну орієнтацію листа в докумтті.

За допомогою VBA можна змінювати зовнішній вигляд або засобів додатк що є в наявності, , а також добавляти свої, цілком нові можливості. В даний час VBA рухається в напрамк до того, щоб стати стандартом в індустрії або створення програм. Після засвоєння VBA можна використовувати цю мову в будь-якому додатку, шо підтримує VBA. Microsoft створила VBA і забезпечила підтримку VBA в усіх головних додатках Microsoft Office: Word, Excel, Access і і інших програм для Windows. VBA дозволяє створювати програмні модулі, меню, діалогові вікна та інші ресурси в середовищі Windows.

Питання для самоконтролю.

1. Що означає абревіатура VBA ?
2. Як різниця між VBA і автономною версією Visual Basic ?
3. Як встановити VBA ?
4. Яка особливість середовище програмування VBA ?
5. Яка структура проекту у VBA ?
6. Що таке макрос у VBA ?
7. Чим відрізняється процедура від макроса ?
8. Які існують типи процедур ?
9. Чим відрізняється модуль від макроса і процедури ?
10. Які існують типи модулів ?
11. Які іменовані сутністі мє мти об’єкт у VBA ?
12. Для чого викоритовуються колекція у VBA ?

Розділ 12. Система оброблення тексту за допомогою VBA у текстовому редакторі MS Word.

12.1. Особливості текстового редактора Word.

Word являє собою текстовий процесор, за допомогою якого можна підготувати різноманітні документи будь-якої складності: від звичайних листів і довідок до виразно оформлених рекламних листків і каталогів, статей і книг. В документі Word можна розміщувати текст, малюнки, таблиці, діаграми і графіки.

Текстовий процесор Word призначений для роботи під керівництвом сучасних операційних систем, таких як Windows ‘95 або Windows ’98 тощо. Працюючи в Word, користувач може застосовувати всі можливості і переваги цих операційних систем: швидкісні характеристики роботи в 32-бітному середовищі; стандартизований інтерфейс; зручність обміну даними з іншими програмами; використання різноманітних шрифтів та ін.

Текстовий процесор Word отримав велику популярність у вітчизняних користувачів, майже витіснивши всі інші програмні продукти такого ж цільового призначення. І це не випадково. Порівняно з Лексиконом Word надає дуже багато додаткових можливостей і зручностей. Наприклад, він володіє суттєво більшою наочністю – документ на екрані виглядає так, як в надрукованому вигляді.

Word як складова частина Microsoft Office має можливість інтеграції з іншими компонентами цієї і більш ранніх версій пакету. В документах Word можна легко розміщувати дані(таблиці, графіки), створені в середовищі Excel, PowerPoint і Access. За форматом Word повністю сумісна з своїми попередніми версіями – при передачі файлів документів між ними не потрібно застосовувати конвертори.

Серед можливостей, які надаються в Word слід виділити наступні: досить зручна система довідникової допомоги, запровадження і пов’язування графічних об’єктів, редагування малюнків засобами самого Word, форматування абзаців і символів, автоматизоване форматування документів на основі стилів, зручність побудови і редагування таблиць, гнучкі засоби структурної перебудови документу, засоби контролю орфографії і граматичної правильності тексту, наявність конвертерів для зв’язку з іншими прикладними програмами.

Використовуючи засоби таких прикладних програм Windows, як Paint, Microsoft Equation, Microsoft WordArt, Microsoft Graph, в програмі Word можна створювати малюнки, формули, текстові ефекти, а також діаграми різноманітних типів на основі табличних даних.

Порівняно з попередниками Word має наступні особливості:

1) покращення діалогу пошуку, відкриття і управління файлами згідно з форматом інтерфейсу Windows;

2) подальше вдосконалення системи довідникової системи на основі технології IntelliSense;

3) можливість автоматичної перевірки орфографії з виділенням помилок;

4) можливість використання ресурсів роботи як в локальній мережі так і в мережі WWW;

5) підвищення зручності автоматичного форматування документу при введенні тексту: виділення заголовків, створення нумерованих і маркірованих списків, а також обмежувальних ліній.

Шаблони, які використовуються при створенні файлів, для зручності розділені на групи: Публікації, Листи і Факси, Записки, Звіти, Інші документи, Загальні, аналогічні групи шаблонів з англійськими назвами; Word отримав більш розширені можливості редактора електронної пошти.

Але слід відмітити, що в Word порівняно з більш старими версіями менш зручною реалізація функції зміни джерел зв'язку з графічними і іншими типами об'єктів. Також слід відмітити порівняно низьку швидкість при роботі з головними і вкладеними документами. Довідникова система Word досить зручна і наочна. В принципі вона дає можливість користувачу розібратись з вирішенням будь-якої прикладної задачі розробки документа. Але через багатогранність і високий розвиток засобів, які надаються в Word , самостійне освоєння техніки вирішення тієї чи іншої задачі може зайняти дуже багато часу.

Етапи роботи з документом. Підготовка документу об'єднує в собі наступні етапи: створення нового або відкриття вже створеного файла документу, запис файла та друк.

При створенні нового документу Word завжди спирається на шаблон. Якщо не вказати шаблон, то в якості основи використовується шаблон *"Обычный"*, який зберігається в файлі *NORMAL.DOT*. можна вибрати інший шаблон, з існуючого в складі Word , або створити свій.

Для створення документів стандартної структури в складі Word існують шаблони, які для зручності розділені на групи: *"Публикации", "Письма и Факсы", "Отчеты", "Другие документы", "Записки", "Общие"*. Обов'язковими елементами шаблонів є набір стилів, формат документу, а також стандартні для документів даного типу текст і графіка. Крім того, шаблони можуть містити макроси, елементи "авто тексту" і установки, зроблені при налагоджені команд, панелей інструментів і клавіатури. Будь-які елементи шаблону при необхідності можуть бути змінені.

При відкритті вже існуючого на даний момент файла документу його вміст завантажується в основну пам'ять і стає доступним для редагування.

Якщо файл документу створений або відредагований в іншому текстовому процесорі, то при його відкритті виконується конвертування в формат, прийнятний для редагування в Word.

Редагування – це зміна вмісту документу шляхом перестановки, заміни, вставки і знищення фрагментів документу. Елемент документу, до якого застосовуються ці дії, попередньо має бути виділеним. Операції вирізання, копіювання і вставки тексту можуть бути виконані за допомогою команд меню або за допомогою існуючих кнопок панелі інструментів *"Стандартная"(Standart).* В Word існує зручний спосіб переміщення фрагментів тексту за допомогою мишки – *"drag-and-drop"* (перемістити і покласти).

Форматування документу – це зміна його зовнішнього вигляду. Параметри форматування можна встановлювати *до* і *після* набору тексту. В Word розрізняють форматування символів і форматування абзаців. Поняття символ містить в собі не тільки окремий символ, але і слово, фразу, а також фрагмент тексту, який не є абзацом. При форматуванні символів, як правило, задаються параметри шрифту: гарнітура і розмір, написання і тип підкреслення, між літерна відстань, скритий текст та ін. При форматуванні абзаців крім параметрів шрифту задаються параметри розміщення абзацу: вирівнювання і відступи відносно полів сторінки, інтервали між абзацами і між рядками в середині абзацу, а також положення самого абзацу на сторінці.

Параметри форматування зберігаються в символі кінця абзацу. Якщо знищити символ кінця поточного абзацу, то абзац приймає параметри форматування абзацу, наступного за ним в документі. Новий абзац, який утворено натисканням клавіші <Enter>, набуває параметри форматування попереднього абзацу.

В Word реалізовані два принципіально відмінних способи форматування тексту – пряме (або безпосереднє) форматування і форматування з використанням стилів. При прямому форматуванні виділеному фрагменту по черзі назначаються необхідні параметри оформлення. Перевага стильового форматування полягає в тому, що вибраному об'єкту(частіше всього це абзац) призначається цілий *набір* параметрів форматування(стиль форматування), який формується завчасно і має унікальне ім'я.

Стиль форматування може містити параметри шрифту, абзацу, табуляції, заливки, кадру, нумерації і вказівка на мову, яка використовується для розстановки переносів і перевірки орфографії. Стилі форматування можуть бути призначені виділеним абзацам чи

абзацу, всередині якого знаходиться курсор. Стилі, з якими працює активний документ, об'єднані в список стилів. При створенні нового документу список стилів містить набір стилів того шаблону, на основі якого створений даний документ. Користувач може модифікувати які-небудь стилі з цього списку, а також додавати нові стилі шляхом копіювання з інших шаблонів і створення власних стилів.

Пряме форматування є найбільш зручним способом оформлення невеликих документів разового користування. При підготовці великих документів доцільно виконувати стильове форматування, що дозволяє автоматизувати процес оформлення і забезпечує ідентичність форматування однотипних елементів документу(єдиний стиль всього документу). Обидва способи не виключають, а доповнюють один одного.

Автоматичне форматування документів в Word базується на застосуванні набору стилів шаблона, приєднаного до документу. На відміну від стилів форматування абзаців документу, які визначають зовнішній вид окремих абзаців, шаблони визначають вид документа в цілому. Завдяки бібліотеці стилів можна скористуватися набором стилів будь-якого шаблону, який міститься в Word .

До поняття форматування можна віднести і форматування сторінок документу, яке полягає у встановленні формату, орієнтації і полів сторінки. Причому ці установки можуть бути застосовані до всього документу, до поточного розділу і до частини документа, починаючи з поточної сторінки і до кінця документу.

Після набору і редагування тексту документу виконується орфографічний, граматичний і синтаксичний контроль правильності тексту. Якщо файл документу планується використовувати в іншому текстовому процесорі, то при його збереженні на диску треба подумати про його конвертування в формат, який підходить для забезпечення можливості подальшої обробки даного документу. Перед друкуванням документу може виконуватись налагоджування пристроїв друку.

Інтеграція можливостей програм. Документ, створений в текстовому процесорі може включати в себе зовсім різні об'єкти, наприклад, таблиці, малюнки, діаграми. Створення і редагування таких об'єктів може здійснюватись без виходу з Word по місцю розташування об'єкта в документі. При цьому можуть використовуватись не тільки власні засоби Word , але і засоби інших прикладних програм Windows.

Спільне застосування різноманітних програм з можливістю доступу до функцій один одного без виходу зі своїх середовищ передбачено одним з найбільш розповсюджених стандартів інтеграції OLE(Object Linking and

Embedding) – зв’язування і вбудування об’єктів), покладеного в основу розробки сучасних програмних систем. Об’єктами можуть бути малюнки, діаграми, таблиці, а також інші елементи документа. Сучасний варіант технології – OLE 2.0 – дозволяє при створенні документів в середовищі текстового процесора(OLE-клієнта) використовувати дані і функціональні можливості інших програм(OLE-серверів). Зокрема, технологію OLE 2.0 підтримують програми WordPad, Word 6.0, Word 7.0 та ‘97(OLE-клієнти) і Excel, PowerPoint, Paint(OLE-сервери).

Об’єкт, який вміщено в документ, зв’язується з файлом документу за посиланням або вбудовується в нього.

При зв’язуванні об’єкт зберігається тільки в файлі OLE-сервера, а в документ, який ми редагуємо об’єкт викликається при необхідності його відображення на екрані або при друці. Такий варіант дозволяє зекономити зовнішню пам’ять, яка потрібна для розміщення файла документа. При переміщенні файла документа на інший комп’ютер необхідно перенести також і файл OLE-сервера з збереженим у ньому зв’язаного об’єкта.

При вбудовуванні об’єкт розміщується в самому файлі документу текстового процесора. Після вбудування об’єкта файл OLE-сервера можна зберегти або знищити. Якщо файл OLE-сервера зберігається і в ньому виконані всі дії над об’єктом, то для внесення цих змін в файл документа виконується актуалізація(поновлення) зв’язку.

В обох випадках(зв’язування або вбудовування об’єкта) при необхідності зміни об’єкта в ході редагування документа здійснюється автоматичний виклик OLE-сервера в вікно текстового процесора без переключання програм Windows.

Вбудовування і зв’язування об’єктів може здійснюватись або з використанням буферу обміну, або шляхом вказування специфікації OLE-сервера. Крім того, існує можливість переносу OLE-об’єкта з вікна однієї прикладної програми(OLE-сервера) у вікно іншої прикладної програми(OLE-клієнта) технікою *“drag-and-drop”*. Наприклад, можна взяти графік у вікні електронної таблиці і перетягнути у вікно текстового процесора.

12.2. Программирование на VBA в Word.

Із усього набору додатків Office Word пропонує найбільш багатий набір засобів програмування.

Основним додатком VBA в об’єктній моделі Word є об’єкт *Application*. Цей об’єкт містить всі інші об’єкти Word. Оскільки об’єкт *Application* центральне місце у программировании на VBA в Word, навідь не потрібно явно вказувати його ім’я при роботі з багатьма важливими

об’єктами. Але не потрібно забувати про роли, яку відіграє цей об’єкт , оскільки він потребує при работі з властивостями і методами самого додатку, а також при звертанні до деяких інших об’єктів. обращении к некоторым другим объектам.

Наприклад, наведедемо інструкція по використанню методу *List Commands* об’єкта *Application:*

Application.ListCommands (True)

Слід зауважити, що метод List Commands створює новий документ і розміщує в нього таблицю, що містить комбінацію клавіш і команди меню Word. Якщо передати методу List Commands значення *True,* то новий документ містить комбинації клавіш і команди меню Word. Якщо передати йому значення False, то будуть перераховані тільки команди.

При записі макросу в користувач може визначати лише документ чи шаблон, в якому зберігається створени макрос. По замоврвленю, макрос автоматично зберігається в модулі NewMacros (якщо такого модуля немає, то Word сам його створить). Макроси, що збережені в модулі NewMacros (якщо такого модуля немає, то його створить MS Word). Макроси, що збережені в шаблоні Normal.dot, доступні для всіх документів, створених на основі цих документів.

Якщо макроси повинні бути доступними лише для деяких текстових документі вів документів, то для них потрібн створити окремий шаблон і в ньому зберігати відповідні макроси.

Введення тексту із макроса Word.

1. Необхідно створити документ Word і помістити в текст закладку використовуючи пункт *Bookvark (Закадка)* в меню *Insert (Вставка)* та присвоїти закладці ім’я, наприклад, *Моя закладка.*
2. Вдкрити редактор VBA і перевірити, що вікно проекту відкрито (можна вибрати його у вікні *Вид*). Виконати фіксацію правою кнопки миші по утвореному документу і вибрати з меню, яке з’явилося, послідовність команд *Insert>Module (Вставте>Модуль)*. Виконайте фіксацію у вікні редактора коду і надрукуйте Sub Ввести Текст. VBA автоматично включить в код оператор завершення процедури процедури End Sub. Тепер надрукуйте наступний код:

```
With ActiveDocument
If ,Bookmarks.Exists(“Моя закладка”) Then
 .Bookmarks (“Моя закладка”) , Select.
End if
End With
Selection.InsertAfter (“Цей текст надрукований із макроса”)
```

Selection.Сщддфзіу (wdCollapseend)

3. Виконайте макрос і перевірте документ Word. Текст *“Цей текст надрукований із макроса”* буде виведено автоматично.

Створення діалогу в Word.

Visual Basic оправдує назву “Візуальний”, використовують його для створення додатку що включають форми і діалоги.

В формах у VBA, використовуючи елементи управлінн ActivX, можна ввести в документ діаграми, мультимедіа, анімації тощо.

Для створення і використання діалога необхідно виконати такі дії.

1. Створіть у Word новий документ, після чого редактор VBA. У вікні проекту необхідно виконати праву фіксацію проекта , що зв’язано з документом і у меню, що з’явиться виконати команди *Insert> UserForm (Вставити> UserForm)*. У відповідь відкриється чиста форма і зявиться панель інструментів. Якщо потрібно макрос використовувати регулярно, то закінчений макрос краще зберегти в шаблоні Normal.dot, а не в документі. Експерементальні версії макроса краще зберігати в документі.

2. Необхідно помістити у форму два перемикачі і дві кнопки з заголовками. Потім встановити для верхнього перемикача властивість *Value (Значення)* в стан *True* . Дальше втановити властивість *Default (По замовчуванню)* і натиснути на кнопку Ок., а потім у властивості *Cancel (Відміна)* кнопку *“ВІдміна”* встновити в стан *True.* Властивості *Default (По замовчуванню)* і *Cancel (Відміна)* визначають, як працює діалог, коли користувач натиска клавіщі *Evter* або *Escape.* . У лівому вікні, що з’явився в списку вибрати пункт *General (”Загальна область ”).* У правому випадаючому списку в пункті *Declarations (Опис)* необхідно виконати фіксацію в редакторі коду і надрукувати *Public DocOption As Integtr.* Ця зиінна буде використовуватися для зберігання результату вибору, що зроблений користувачем в діалозі.

4. Виконайте фіксацію кнопки *Ок,* щоб відкрити код події *Click (Фіксація)* і надрукуйте наступний код:

```
If OptionButton1.Value=True Then
User Form1..DocOption=1
Else If Form.DocOption2.Value= True Then
User Form1..DocOption=2
Else
User Form1..DocOption=0
Else If
Me. Hide
```

Потрібно виконати подвійну фіксацію кнопки *Відміна* і ввести єдиний

рядок:
Me. Hide.

5. Потім ввести в проект модуль або відкрити модуль, якщо він уже існує. Відкрити його в редакторі коду і надрукувати:

Sub DocStart()
UserForm1.DocOption=0
UserForm1.Show
Select Case UserForm1. DocOption
Case 1'Стандартний документ
Documents.Add 'Буде використано Normal.dot
Case 2 'Документ лист
Documents.Add Template: ="C:|Program Files
\Microsoft Offict\Шаблони|Листи і факси
\ Стандартний лист .do
End Select
End Sub

В даному випадку необхідно виправити шлях і ім'я файлу шаблону згідно з місцем встановлення Word в даній системі.

6. Потрібно закрити VBA і повернутися у Word. Потім виконати макрос DocStart. Він створить новий документ на основі вибраного шаблону.

Питання для самоконтролю

1. Перелічте відомі Вам текстові редактори алгоритмвчних мов, операційних систем та інших програмних продуктів.
2. В чому відмінність текстового редактора Word в порівняні з іншими текстовими редакторами ?
3. Перелічте команди основного меню текстового редактора Word та дайте їм коротку характеристику.
4. В чому суть панелі інструментів у Word ?
5. В чому суть панелі форматування у Word ?
6. Перелічте етапи роботи з документом у Word.
7. Які два принципово відмінних способи форматування тексту у Word ?
8. Опишіть характеристику об'єкту VBA *Application* у Word.
9. Які дії необхідно виконати для введення тексту із макроса Word ?
10. Які дії необхідно виконати для створення і використання діалога у Word на основі VBA ? .

Розділ 13. Система оброблення даних в електронної таблиці MS Excel за допомогою VBA.

13.1. Ознайомлення з основними командами меню електронної таблиці MS Excel

Як і будь-яка Windows – програма, Microsoft Excel містить ряд типових елементів. Рядок заголовка (верхній рядок вікна) містить назву програми (“Microsoft Excel”), а також назву книги, з якою працюють в даний момент. Крайня ліва кнопка є кнопкою виклику системного меню, за допомогою якого можна керувати розмірами і положенням програми, а також закривати її. Праворуч розміщені відповідно кнопка згортання, відновлення та закриття програми.

Рядок меню. Рядок меню звичайно розміщується під вікном заголовку і включає такі пункти:

1. *”Файл”* – робота з файлами книг (створення, збереження, відкриття файлів, друкування файлів книг);
2. *“Правка”* – вміщує команди роботи з книгою, пошуку та заміни;
3. *”Вид”* – дозволяє змінити спосіб відображення книги на екрані (звичайний/розмітка сторінки), а також задати панелі інструментів, що будуть відображені, та змінити масштаб;
4. *“Вставка”* – вставка в книги нових листів, малюнків, діаграм та інших типів даних;
5. *“Формат”* – дозволяє змінити формат комірки, рядка / колонки чи цілого листа, управління стилями;
6. *“Сервіс”* – сервісні функції (перевірка орфографії, настройка автозаміни, встановлення захисту на лист чи цілу книгу, робота з макросами та настройка параметрів Excel);
7. *“Дані”* – робота з базами даних (сортування, фільтр та ін.);
8. *“Вікно”* – робота з вікнами книг ;
9. *“Справка”* – виклик довідкової інформації.

13.2. Панель інструментів *«Стандартнаная»*

 - Створення нового файла на основі активного шаблона. (Файл⇒Создать… /Ctrl + N/)

 - Відкриття чи пошук файл (Файл⇒Открыть… /Ctrl + O/)

 - - Збереження поточного файла без зміни його назви, формату і місця розташування (Файл⇒Сохранить… /Ctrl + S/)

 Друк поточного файла чи виділеного елемента (Файл⇒Печать… /Ctrl+P/)

 Попередній перегляд документа в тому виді, в якому він буде надрукований (Файл⇒Предварительный просмотр)

 -Перевірка орфографії в поточному файлі, документі, книзі чи повідомленні (Сервис⇒Орфография… /F7/)

 Вилучення виділеного фрагмента з поточного документа і вставка його в буфер обміну (Правка⇒Вырезать /Ctrl + X/)

 Копіювання виділеного фрагмента в буфер обміну (Правка⇒Копировать /Ctrl + C/)

 Вставка фрагмента з буферу обміну в поточну позицію з заміною виділеного фрагмента текста. Команда доступна тільки в тому випадку, коли буфер обміну містить будь – які дані. (Правка⇒Вставить /Ctrl + V/)

 - Копіювання форматування виділеного об’єкту чи тексту. Це форматування буде потім присвоєно об’єкту чи тексту, який буде вибрано наступним. Якщо форматування необхідно присвоїти кільком об’єктам то спочатку виділяють об’єкт, формат якого будуть копіювати, після цього натискують цю клавішу двічі і послідовно виділяють ті об’єкти, яким необхідно присвоїти нове форматування. Після цього повторно натискають цю клавішу для виходу з режиму **Формат по образцу**

 - Відміна останньої виконаної команди або вилучення останнього введеного фрагменту. Для відміни декількох останніх команд треба натиснути стрілку і вибрати дії, які необхідно відмінити. Якщо відмінити останню дію неможливо, то назва цієї команди міняється на **Нельзя отменить**. (Правка⇒Отменить /Ctrl+Z/)

 - Відміна дії останньої дії клавіші **Отменить**. Як і в попередньому випадку є можливість вернутись на декілька кроків. (Правка⇒Вернуть /Ctrl+Y/)

 - Вставка чи редагування заданої гіперссилки (Вставка⇒Гиперссылка… /Ctrl+K/)

 - Відображає / ховає панель WEB

 - Автоматично добавляє функцію знаходження суми **СУММ** в Microsoft Excel. Діапазон комірок, що додаються буде обрано автоматично. Щоб вибрати діапазон самостійно необхідно провести через нього мишу з нажатою клавішею, а потім натиснути клавішу **ENTER**.

 - Вивід списку функцій і їх прототипів з можливістю задавати значення аргументів. (Вставка⇒Функция…)

 - Сортування вибраних елементів в порядку від початку алфавіту до його кінця, від менших чисел до більших і від більш ранніх дат до більш пізніх по колонці, в якій знаходиться курсор. Якщо до цього були встановлені інші параметри сортування, то вони зберігають свою дію.

- Сортування вибраних елементів в порядку від кінця алфавіту до його початку, від більших чисел до менших і від пізніх дат до більш ранніх по колонці, в якій знаходиться курсор. Якщо до цього були встановлені інші параметри сортування, то вони зберігають свою дію.

 - Запуск майстра діаграм, який дозволяє крок за кроком створити нову діаграму, чи замінити існуючу. (Вставка⇒Диаграмма…)

 - Створення карти на основі виділених даних. Дані повинні вміщувати посилання на географічні назви, наприклад назви країн чи регіонів. (Вставка⇒Карта…)

 - Виводить панель WordArt для створення текстового ефекту.

 - Ввід масштабу від 10 до 200% для збільшення чи зменшення зображення активного документа на екрані. (Вид⇒Масштаб…)

 - Клавіша виклику **Помощника**.

13.3. Панель інструментів *«Форматирование».*

Зміна шрифту виділеного тексту

 - Шрифт може бути змінений в полі **Шрифт**.

 - -Зміна розміру шрифту виділеного тексту. В полі **Размір шрифта** необхідно ввести нове значення розміру шрифту. Набір доступних розмірів залежить від вибраного шрифту і поточного принтера.

 - Оформлення виділеного тексту напівжирним шрифтом. Якщо виділений текст вже є напівжирний то ця клавіша зніме цей режим.

 - Оформлення виділеного тексту *курсивом*. Якщо виділений текст вже є *курсив* то ця клавіша зніме цей режим.

 - Оформлення виділеного тексту підкресленням. Якщо виділений текст вже є підкреслений то ця клавіша зніме цей режим.

 - Вирівнювання виділеного тексту, чисел і вкладених об’єктів по лівому краю з нерівним правим.

 - Вирівнювання виділеного тексту, чисел і вкладених об’єктів по центру.

 - Вирівнювання виділеного тексту, чисел і вкладених об’єктів по правому краю з нерівним лівим.

 - Об’єднання декількох виділених комірок в одну. Ця комірка буде містити дані тільки із лівої верхньої комірки, розміщені по центрі об’єднаної комірки. Посилатись на цю комірку потрібно по адресі лівої верхньої комірки групи.

 - Форматування виділених комірок в **Міжнародному грошовому форматі** у відповідності з настройками Windows в **Панелі управління.**

 - Установка процентного формату для виділених комірок

-Установка формату з розділювачем для виділених комірок

- Збільшення числа дробових знаків для виділених комірок

- Зменшення числа дробових знаків для виділених комірок

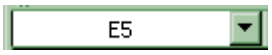
- Додає границі до виділеної комірки чи діапазону

-Зменшення відступу виділеної комірки приблизно на ширину символу стандартного шрифта

- Збільшення відступу виділеної комірки приблизно на ширину символа стандартного шрифта

- Форматування виділеного тексту заданим кольором

"Рядок формул"

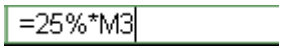


В даному полі можна задати назву для поточної комірки чи групи комірок і в подальшому звертатись до неї можна буде за цим іменем.

Наступні 3 клавіші використовуються для вводу формул:

- Вийти з режиму редагування формул без запам'ятовування змін

- Запам'ятати зміни в формулі і вийти з режиму редагування



- Використовується для входу в режим редагування формул

=

- Вміст рядка формул

13.4. Поняття про робочі книги і робочі листи у Excel

Робоча книга являє собою набір робочих листів, що зберігаються в одному файлі з розширенням .**xls** і містить згруповані робочі листи, що належать одному проекту. Робочий лист - це таблиця Excel що складається з рядків і стовпчиків, які містять дані і формули.

Нова утворена робоча книга містить три робочих листа з іменами "Лист 1", "Лист 2", "Лист 3", що міститься на панелі стану. Якщо виконати фіксацію по робочому листу то він буде відображений на передньому плані. Стовпчики мають заголовок A,B,C, рядки 1,2,3.Перетин будь-якого стовпчика і рядка називають чарунка з відповідним ім"ям (наприклад,A1). Лінії сітки дозволяють інформацію відділяти в різних чарунках. Для вилучення ліній на екрані, необхідно в пункті меню **"Сервіс"** вибрати **"Параметри"** , а потім у вкладці **"Вид"** після появи діалогового вікна **"Параметри"** скинути прапорець команди **"Сітка".**

Excel надає можливість працювати одночасно з декількома листами. Для того, щоб вставити новий лист в робочу книгу, необхідно виконати фіксацію правої кнопки миші на ярлику листа, за яким буде створено новий лист. В контекстному меню необхідно вибрати пункт **"Добавити".**

Після появи діалогового вікна **"Вставка"** необхідно виконати фіксацію по значку **"Лист",** а потім по кнопці **"ОК".**

Вставити новий лист в робочу книгу можна також, вибравши в пункті меню **"Вставка"** команду **"Лист".** Для вилучення зайвих листів з робочої книги необхідно виконати фіксацію правою кнопкою миші по ярлику листа, якій необхідно вилучити, а потім в контекстному меню вибрати пункт **"Вилучити".**

Переходити з однієї робочої книги до іншої можна з допомогою комбінації клавіш Ctrl+F6. Для того, щоб узнати які робочі книги відкриті в даний момент, необхідно відкрити меню "Окно" і в його нижній можна побачити перелік відкритих книг.

Введення даних в таблицю. Вести дані в Excel можна у відповідну чарунку переміщенням табличного курсора. Але різні дані можуть вести себе по різному. Excel може працювати з такими даними;

1) текстові значення, такі як імена чарунок і адрес;
2) числові значення: 35, 35.2 ,45.2 E-02;
3) формули - вирази, які дають можливість працювати з числовими даними, а деякі формули працюють і з текстовими значеннями;
4) дані спеціальних форматів: значення дати і часу.

Відносна і абсолютна адресація чарунок При копіюванні формули, що містить адреси чарунок Excel перетворює їх, якщо це адреси відносні, тобто відносна адреса посилається на чарунку відносно чарунки, що містить формулу. Наприклад, якщо в чарунку A1 ввести формулу =A2+A3, тобто ця формула містить дві адреси відносно адреси A2 і A3 в чарунку B5, то одержимо там формулу B6+B7.

Абсолютною адресою називається адреса, що не змінюється при копіюванні формули. Перед абсолютною адресою встановлюється знак $.

Наприклад, якщо потрібно просумувати значення двох чарунок, а потім помножити на деяке постійне число, що міститься в іншій чарунці, то можна записати формулу :=(A1+B1)*D1, тут A! і B1 відносні адреси, D1 абсолютна адреса. Якщо потрібно скопіювати цю формулу в рядок нижче, то вона прийме такий вид: =(A2+B2)*D1

Можливі часткові абсолютні адреси. Наприклад, $ B5 - це частково абсолютна адреса. Якщо скопіювати формулу з такою адресою, то позначення стовпця залишиться без зміни, але посилення на рядок (5) буде перетворена згідно положення чарунки, в яку скопійована формула. Якщо в чарунку A1 ввести формулу =2*$B5, а потім її скопіювати в чарунку F6, то в ній одержимо таку формулу =2*$B10.

В цьому прикладі скопійовано формулу в чарунку на 5 рядків нижче і 5 стовпчиків лівіше від початкового положення.Excel не перетворила позначення стовпчика В як абсолютного, але до позначення рядка додала 5 рядків, оскільки ця частина адреси відносна і відкрита для зміни при копіюванні формул.

Робота з базами даних в Excel. Робочу книгу Microsoft Excel можна розглядати як базу даних. Кожну колонку тоді розглядають як поле, а кожен рядок – як один запис в базі даних. Оскільки в базах даних кожне поле має свою назву, тип, ширину та інші параметри то аналогічно потрібно встановити ці параметри і для кожної колонки бази даних.

В Microsoft Excel є набір функцій, що полегшують обробку і аналіз баз даних. Наприклад для відображення рядків і колонок, використання форматованих заголовків колонок, відображення рядків і колонок. використання границі комірок, відсутність порожніх рядків і колонок. відсутність початкових і кінцевих пропусків.

Режим вводу даних в базу даних з допомогою форми включається командою *Данные-* в режимі вводу даних через "форму" вікно зліва містить назви полів і навпроти кожного з них є поле вводу, куди потрапляють записи з бази даних при переміщенні повзунка, чи при натисненні клавіш *Назад* і *Далее.* В правому верхньому куті розміщений вказівник відношення поточного запису до їх загальної кількості.

Клавіші *Добавить* і *Вилчити* служать для вставки/вилучення записів з бази даних. При вставці запису він з’явиться в кінці бази даних, а при вилученні – вилучається поточний запис. При вилученні буде видано вікно підтвердження дії з повідомленням про те, що відновити даний запис після вилучення буде неможливо.

Клавіша *Вернути* призначена для того випадку, коли зміни зроблені в поточному записі, що відображається на екрані не потрібно зберігати. Слід зазначити що ця команда не відновить запис, що був вилучений командою *Удалить*. Клавіші *Назад* і *Далі с*лужать для переміщення по записам бази даних назад\вперед.

Клавіша *Критерії* дозволяє знайти записи по заданих умовах. При натисненні клавіші *Критерії* номер запису в правому верхньому куті форми міняється на напис *Критеїїи*. В полях, розміщених в лівій частині форми, необхідно ввести одне чи кілька значень, які будуть використовуватись в якості умов, Потім натисніть кнопку *Далі* або *Назад*.

Клавіша *Закрыти* закриває форму. Для запису введених записів необхідно зберегти книгу.

Для сортування бази даних необхідно вибрати команду *Дані= Сортиртування.* Як і в попередньому випадку, табличний курсор необхідно встановити на будь-який запис в базі даних інакше буде видано повідомлення про те, що список не знайдено. Якщо перед вибором даної команди був виділений діапазон то сортуватись буде саме він. В противному випадку Excel сам виділить цілу базу даних. Не рекомендується сортувати частини полів (коли виділені не цілі записи а тільки окремі поля з них), тому що це призведе до того, що будуть просортовані тільки окремі поля бази даних без корекції решти полів.

В полі *Сортиртувати по…* необхідно вказати поле, по якому буде сортуватись база даних. Справа від даного поля розміщені перемикачі. Перемикач *по зрастанню* дозволяє здійснити сортування в порядку від найменшого числа до найбільшого, від першої літери алфавіту до останньої, від давньої дати до недавньої. Перемикач *по зменшенню* дозволяє здійснити сортування в порядку від найбільшого числа до найменшого, від останньої літери алфавіту до першої і від недавньої дати до давньої.

13.5. Використання функцій в Microsoft Excel.

Функції уявляють собою наперед визначені формули, які виконують обчислення за даними значеннями і у вказаному напрямку.

Імена функцій затверджуються круглими дужками в яких розміщені аргументи, що відділяються один від одного комами. Аргументів у функції може бути декілька, а може їх і не бути.

Функції полегшують роботу з клітинками електронної таблиці. Використання функцій можна здійснювати частково за допомогою панелі інструментів (наприклад SUM (<СУММ>), а також за допомогою команди Function Wizard (<мастер функцій>), виникнення якого здійснюється за допомогою клавіш Shift + F3 або за допомогою відповідної кнопки на панелі інструментів. Функції в Excel можна поділити на такі групи:

1. ***математичні функції***

- SUM(< діапазон комірок > або < числові значення >) [<=СУММ>] – для суми значень вказаних в діапазоні клітин або числових значень.

- PRODUCT(<діапазон клітин> або < числові значення>) [<=ПОИЗВЕД>] – для обчислення значень діапазону клітин або числових значень.

2. ***статичні функції***

- **ARERAGE** (<сукупність чисел) [<ФЗНАЧ>] – обчислює середнє арифметичне.

- **MAX** (<сукупність чисел) [<МАКС>] – обчислює найбільше значення із сукупності

- **MIN** (<сукупність чисел) [<МІН>] – обчислює найменше значення в даній області (чисел або адрес комірок)

- **RAN**K (<число або адрес комірки; <область дії функції>; <порядок>) [<РАНГ>] – виконує сортування значень в порядку спадання якщо <аргумент> не виконано, або значення якщо <аргумент> більше нуля.

- **TREND** (<перелік відомих значень; <нові значення>; <константи>) [<ТЕНДЕНЦІЯ>] – визначає прогноз на <константу> кроків певних значень (прибутків на n років).

- **COUNT** (<перелік значень >) [<СЧЕТ>] – підраховує кількість чисел в списку аргументів, використовують, наприклад, для обчислення кількості комірок, в які вписано числові значення.

- **VAR** (<сукупність чисел >) [<ДИСП>] – обчислює дисперсію випадково вибраних чисел.

3. ***фінансові функції***

- **PMT** (<ставака-X1>; <термін погашення кредиту-X2>; <сума кредиту-X3>; <залишок після сплачення останнього кредиту-X4>; <тип-X5) [<ППЛАТ>] – визначає тип звернення до послуг кредиторів на діяльність фірми, <тип> може приймати значення 0 або 1;

1 – якщо виплата проводиться на початку платіжного періоду (авансова);

0 – якщо виплата проводиться в кінці (додаткова);

- NPER (<X1>; < X2>; < X3>; < X4>; < X5>) [<КПЕР>] – визначає кількість платежів, яка необхідна для того, щоб знаючи суму періодичних (щомісячних, наприклад) виплат і процентну ставку,

X1- ставка; X2 - термін погашення кредиту; X3 - сума кредиту; X4 залишок після сплачення останнього кредиту; X5 - тип

- **SLN** (<вартість; <залишок; <період) [<АМП>] – обчислює суму періодичної амортизації

<вартість> початкова вартість майна;

<залишок> залишкова вартість майна в кінці періоду амортизації

- **SYD** (<вартість>; <залишкова вартість>; <час експлуатації>; <період>) [<АМГД>] – повертає річну амортизацію майна до вказаного періоду.

- FV (<X1>; < X2>; < X3>; < X4>; < X5>) [<БЗ>] – повертає майбутні значення вкладу на основі періодичних постійних платежів і постійної процентної ставки.

X1- процентна ставка за період;

X2 – загальне число періодів оплати річної ренти;

X3 – виплати, що здійснюються кожного періоду; X4 -залишок після сплачення останнього кредиту; X5 – тип; 0 в кінці періоду оплата; 1 на початку періоду оплата;

- **IRR** (<діапазон клітин>; <прогноз>) [<ВНДО>] – обчислює внутрішню швидкість обігу.

4. ***функції дати і часу***

- <TODAJ>[<СЕГОДНЯ>] - обчислює поточну дату в числовому форматі
- NOW [<СЕГОДНЯ>] – вводиться в таблицю поточна дата а також час доби

5. ***Логічні функції***

- IF (<логічний вираз>; <значення якщо істина>; <значення якщо хибність>) [<ЕСЛИ>] – визначає значення логічного виразу.

13.6. Утворення діаграм в Excel.

Для утворення діаграм в Excel використовують кнопку «Мастер диаграмм» на панелі форматування або відповідну команду в пункті меню.

Майстер діаграм дозволяє відобразити табличні дані у вигляді кольорової діаграми, необхідний тип діаграми можна вибрати у першому вікні майстера діаграм:

1. **«Гістограма»(Area) –** дозволяє відобразити зміну даних в часі; цей тип діаграм зручний також для наглялного порівняння різних величин;
2. **«Лінійчата»(Bar)** – дозволяє порівнювати окремі значення;
3. **«Графік»(Line)** – на графіку з рівними проміжками вказані значення, що прогнозуються;
4. **«Колова»(PiL)** – показує відношення розмірів елементів, що утворюють ряд даних, до суми елементів;
5. **«Точкова»(XY/Scatter)** – або показує відношення між числовими значеннями в декількох різних даних, або відображає дві групи чисел, як один ряд координат X і У;
6. **«З областями»(Column)** – підкреслює зміну в часі. Відображаючи суму значень рядів, така діаграма наглядно показує вигляд кожного ряду;
7. **«Кільцева»(Doughnut)** – як і колова діаграма, показує відношення частин до цілого, але цей тип діаграми можна включати декілька рядів даних; кожне кільце в діаграмі відповідає одному ряду даних;
8. **«Пелюсткова»(Radar) –** усі категорії мають власні осі координат, що розходяться променями з початку координат; лініями з’єднуються значення, що відносяться до одного ряду;

8.1.«**Поверхня»(Surface)** – дану діаграму доцільно використовувати для пошуку найкращого поєднання у двох наборах даних; як на топографічній карті, області, що відносяться до одного діапазону значень, виділяються однаковим кольором або орнаментом;

8.2. **«Бульбашкова»** – різновид точкової діаграми; розмір маркера даних показує значення третьої змінної;

8.3. **«Біржева»** – часто використовується для відображення зміни температури; для побудови біржових діаграм необхідно розмістити дані у вірному порядку;

8.4. .**«Циліндрична, конічна, пірамідальна»** - використання маркерів даних циліндричної, конусоподібної та пірамідальної форми може суттєво покращити зовнішній вигляд і наглядність об'єктноїПР діаграми

13.7. Взаємодія EXCELз VBA.

У табличному процесорі MS Excel створені мкроси можуть зберігатися в поточній книзі, в особистій книзі макросів. По замовчуванню макроси зберігаються в поточній книзі. Програма сама вибирає модуль, в якому буде записуватися макрос, і при потребі створює його.

Програма Excel сама вибирає модуль в якому буде записуватися макрос, і за необхідностю створює його При створення модуля йому присвоюється імя' *ModuleleN*, де N- кількість модулів для відповідної робочої книги. Якщо ж робоча кига містить модулі з іменами *Module1* і *Module2*, то програма збільшить число в назві модуля *(Module3)*, так щоб воно не збігаося з існуючим.

Починаючи, з версії 5.0 в програмі Excel включений, спеціальна мова програмування VBA. Завдяки цій мові з'являється можливість значно розширити набір функцій у Excel, а також створювати функції, значення яких залежать від деяких умов і подій. У принципі, можна повністю перепрограмувати всі функції програми Excel, якщо в цьому з'явилася необхідність.

Розглянемо програмування табличних функцій. Щоб створити окремий робочий лист для програмного модуля, необхідно виконати фіксацію по піктограмі **Insert Module** з піктографічного меню Visual Basic (1-а піктограма) або викликати директиву **Module** з меню **Insert Macro.** Після цього з'явиться новий робочий аркуш **"Modele1"**. У програмному модулі потрібно описати функцію на мові VBA. У вікні програмного модуля можна працювати, як у вікні невеликого текстового редактора.

Опис функції має починатися оператором *Function*, за яким через пробіл слідують назву функції і її аргументи, укладені в дужки і розділені

комами. Потім йде власне текст програмного коду функції, а закінчуватися опис має оператором *End Function*. Якщо в тексті програмного коду ім'я обумовленою функції буде знаходитися в лівій частині оператора присвоювання (позначається знаком рівності), то присвоєне значення і буде результатом обчислення функції при заданих аргументах. Як приклад можна розглянути функцію, яка обчислює податок на додану вартість.

```
Function NDS (Value)
NDS = Value * 0.15
End Function
```

Вбудовування функцій. Для вбудування функції небхідно виконати фіксацію пр **Object Browser** з піктографічного меню VBA або викличте однойменну директиву з меню **View.** Функції, визначені користувачем, розглядаються в програмі Excel як самостійні об'єкти. У полі списку **Methods / Properties:** буде знаходитися ім'я нової функції. Треба виконати фіксацію спочатку по імені, а потім по командній кнопці **Options,** тоді відкриється діалогове вікно **Macro Options.** У полі **Description:** необхідно ввести пояснювальний текст, який пізніше буде використаний *Конструктором функцій*. У списку **Function Category** необхідно вказати категорію, до якої потрібно записати свою функцію. Наприклад, функцію, яка обчислює податок на додану вартість, слід помістити в категорію **Financial.** Надалі *Конструктор функцій* помістить утворену функцію у вибрану категорію. Закрийти вікно **Macro Options,** виконавши фіксацію по командній кнопці **OK,** а у вікні **Object Browser** –по кнопці **Close.**

Застосування функцій. Для застосування функцій необхідно перейти на робочий лист, де буде розташована таблиця. Перемістіть покажчик комірок у клітинку, в якій буде перебувати формула, і введіть в неї знак рівності. Потім необхідно виконати фіксацію клацніть по піктограмі *Конструктора функцій* на основний піктографічної панелі.

З'явиться діалогове вікно *Конструктора функцій*. На першому кроці необхідно виберати категорію **Financial** і в правому полі знайти функцію NDS. Виконати фіксацію по назві вибраної функції, після чого перейти до наступного кроку, виконавши фіксацію по командній кнопці **Next.**

Відкриється друге діалогове вікно *Конструктора функцій*. Тут можна побачити коментар до функції, який був введений раніше в вікні макроопцій. Треба вказати єдиний аргумент для цієї функції *Value* і закрийте діалогове вікно *Конструктора* фіксацію по кнопці **Finish.**

У таблиці з'явиться значення, яке становить 15% величини аргументу. У таблиці з цією функцією можна працювати як зі звичайною функцією програми Excel.

Запис макроса в Excel.

1. Необхідно запустити Excel і перевірити, чи завантажена нова робоча книга. Потім виконати таку послідовність команд: *Tools>Macro>Record New Macro (Сервіс>Макрос> Почати Запис).*
2. В діалоговому вікні, що з'явилося необхідно надрукувати і'мя макроса, наприклад, *Макрос відкпити.* Потім перевірити, що у списку, що з'явився *Store Macro in (Зберегти в)* вибрати пункт *This Workbook (Ця книга)*. Потім виконати фіксацію по кнопці Ок. Макрос самостійно записує код Visual Basic
3. Необхідно звернути увагу ра появу невеликого вікна макрорекодера. Поки видно це вікно усі дії записуються. Тепер необхідно в меню Excel вибрати послідовність команд *File>Open (Файл>Відкрити)* і відкити будь-яку робочу книгу за влаим вибором. Коли робоча книга відкриється, необхідно виконати фіксацію по кнопці *Stop Recording (Зупинити Запис).*
4. Можна легко перевірити роботу макроса. Закрити відкриту робочу книгу і ввійти у меню *Tools>Macro>Macros(Сервіс>Макро >Макроси),* а потім виконати фіксацію по кнопці *Run (Виконати).* Робоча книга буде знову відкрита.

Процедура запису в Excel

```
 1 Sub Форматування ()
 2 ‘
 3 ‘ Форматування Мокрас
 4 ‘ Марос записаний 20.02.2020 (Lidia Gobyr)
 5 ‘
 6     Range(“C3:H9”) Select
 7     Selection.Font.Bolt=True
 8    With Selection
 9         ,HorizontalAllgnment=xlCenter
10        ,VerticalAllgnment=xlBottm
11        ,WrapText=False
12       ,Orlentation=0
13       ,Addlndept=False
14       ,ShrinkToFit=False
15       ,MergeCells=Falls
16     End With
17     Range(“C1”):Select
```

18 End Sub

Першим рядком є команда опису процедури, яка розпочинається службовим словоь Sub Sub, за яким йде назва процедури та пара дужок. Цей рядок називають *рядком опиву макросу* чи *процедури.*

У рядках 2-5 знаходяться комендарі *Коментар* – рядок процедури, яка не містить інструкції, але є складово програми. Коментарі містять допоміжну інформацію (призначення макросу, опис процедури тощо). Кожний рядок коментара починається з апострофа (‘) . Для VBA будь-який текст, який починається з апострофа , ідентифікується як коментар. У даному прикладі коментарі ідентифікують інформацію про назву макросу та дату його створення.

Наступною частиною програмного коду є *тіло процедури.* Кожний рядок тіла складається з одного або декілька операторів. У тілі процедури також можна також можна розташовувати коментарі, які містять вказівки та пояснення для зручності читання програми.

Після тіла процедури йде заключний рядок програми *End Sub.* Як видно з прикладу, багато рядків прогами мають вілступи , а увесь текс - зміщено вправо між ключовими словаит *Sub* і *End Sub.* При написанні текстів у редакторі VBA ці відступи можна зробити а допомогою клавіщі *Тав.*

Питання для самоконтролю

1. В чому відмінність рядка основного меню Excel від Word
2. В чому суть панелі інструментів у Excel ?
3. В чому суть панелі форматування у Excel ?
4. В чому різниця між робочою книгою і робочим листом у Excel ?
5. Скільки листів по замовчуванню виводиться у робочій книзі у Excel ?
6. Вкажіть можливі способи вставлення нового листа в робочу книгу у Excel.
7. Які типи даних можна вводити в Excel ?.
8. Вкажіть послідовність дій для вилучення фільтра зі списка в Excel?
9. Які можливості роботи з базами даних в Excel ?
10. Які функції полегшують обробку і аналіз баз даних ?
11. Опишіть послідовність дій для введення даних в базу даних за допомогою форми в Excel.
12. Як виконати сортування бази даних в Excel ?
13. В чому різниця між відносною і абсолютною адресацією чарунок.
14. Перелічите основні типи функцій, що використовуються у Excel.
15. Як можна утворити діаграму в Excel ?
16. Які діаграми в Excel є об’ємні ?

17. Що необхідно зробити щоб створити окремий робочий лист для програмного модуля в Excel з використанням VBA ?
18. Що необхідно зробити для вбудування функції VBA в Excel ?
19. Опишіть послідовність дій для запису макроса в Excel.

Розліл 14. . Система управління базами даних MS Access за допомогою VBA.

14.1.Моделі баз даних

Автоматизований банк даних (БнД) визначають як систему інформаційних, математичних , програмних, організаційних і технічних засобів, призначених для централізованого накопичення і колективного багатоаспектного використання даних з метою одержання необхідної інформації.

Основу БнД складають база даних (БД). Базу даних складають як сукупність взаємозв'язаних даних , що зберігаються разом при наявності такої організації і мінімальної надлишковості , яка допускає їх оптимальне використання одним або кількома користувачами. Дані в БД запам'ятовуються і використовуються так , щоб вони були незалежними від програм , які використовують ці дані.

Використання БД дозволяє багаторазово використовувати дані ; добавляти дані,не змінюючи існуючих прикладних програм ; виключати надлишковість даних ;забезпечити захист даних від випадкового використання їх ; скорочувати витрати на розміщення робіт по опрацюванню даних ; забезпечити незалежність даних від прикладних програм.

Кожна БД характеризується своїми логічною і фізичною структурами. Під логічною структурою БД вважають її логічний опис разом з методом доступу до даних ; фізична структура являє собою запис даних на запам'ятовувальному пристрої.

Оскільки БД обслуговує значну кількість користувачів , необхідно щоб вони чітко знали , що являють собою ті чи інші дані , а також орієнтувалися як вони розміщені. Роль такого показчика – каталога виконує словник даних.

Словник даних – централізоване сховище відомостей про дані , взаємозв'язок між ними , джерела одержання і формати представлення.

Для забезпечення централізованого управління даними , а також організації доступу до них при розв'язанні різноманітних задач використовують систему управління базових даних (СУБД).

СУБД – це програмний продукт , який є центральним елементом БД і по суті визначає ефективність його функціонування. СУБД є зв’язковою ланкою між вимогами користувача і способом , яким повинні бути організовані дані , що знаходяться на запам’ятовувальному пристрої. На рис 14.1. представлена логічна і фізична структура даних.

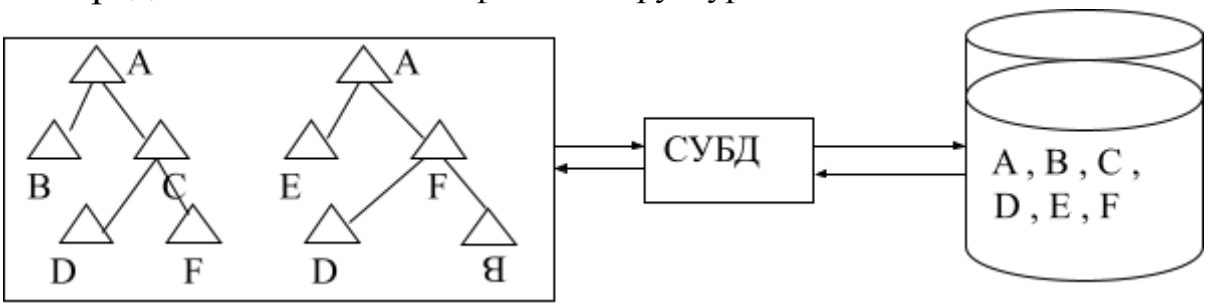


Рис. 14.1. Логічна і фізична структура даних.

Система баз повинна мати можливість представляти не тільки сукупність даних , але й зв’язки між ними. Для побудови структури зв’язків використовують три моделі представлення даних :

1) ієрархічна – це така організація даних , за якої дані підпорядковані один одному по старшинству , тобто зв’язки між ними можна зобразити у вигляді дерева за типом “ один до багатьох” (рис. 14.2.а)

2) сіткова – являє собою організацію даних у вигляді орієнтованого графу , де зв’язки між даними дозволяють утворювати більш складну модель співвідношення типу “ багато до багатьох “ , тобто в порівнянні з ієрархією , це більш загальна структура. (рис. 14.2.б)

3) реляційна – використовує двохвимірні таблиці для представлення більшості структур даних , створює простоту опрацювання БД для різних користувачів і дає можливість одержувати файли необхідної форми за рахунок простоти опису зв’язків між даними. (типу "один до одного" рис. 14.2.в)

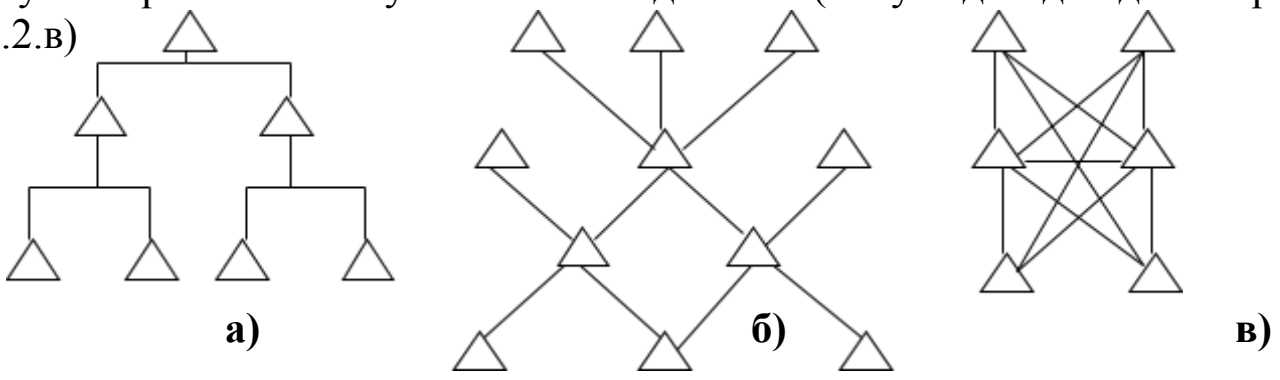


Рис. 14.2. Моделі баз даних

14.2. Загальна характеристика пакета Microsoft Acces

Система управління базами даних (СУБД) Microsoft Access разом з іншими подібними програмами, такими як Microsoft Word та Microsoft

Excel відноситься до сімейства Microsoft Office. Вона призначена для обробки баз даних (БД) реляційного типу. База даних цього типу складається з таблиць, запитів, форм, звітів, макросів і модулів. Таблиці є головним джерелом даних, вони можуть бути зв'язані між собою за допомогою спільного поля (стовпця). Це дозволяє ефективно використовувати оперативну та магнітну пам'ять ПК за рахунок уникнення повторення інформації, спростити процес обробки даних. Можливість зв'язування таблиць і дала назву базам даних, у перекладі з англійської relation означає зв'язок.

Одна з перших СУБД – dBASE була створена в 1970 році, вона поклала початок цілому сімейству систем: dBASE-Ш+, Карат, Пролог, Clipper, dBASE-IV, FoxPro. Від своїх попередників MS Access відрізняється функціонуванням під управлінням операційної системи типу Windows та використанням її інструментів.

Серед засобів Microsoft Access можна умовно виділити такі три типи:

1) основні, вони, власне, і призначені для виконання всіх робіт;

2) додаткові, які забезпечують покращання візуального сприйняття об'єктів системи (зміна типу та розміру шрифтів, кольору ліній та фону);

3) дублюючі засоби, вони часто дублюють основні засоби обробки даних, що, звичайно, затрудняє вивчення системи.

Характер обробки інформаціїї в Access відбувається подібно до роботи з іншими пакетами сімейства Microsoft Office: практично всі команди виконуються за допомогою натискання відповідних кнопок. Специфікою СУБД MS Access є наявність порівняно великої кількості спеціальних вікон. Кожне вікно має власну, відмінну від інших, назву, яка знаходиться в лівому верхньому куті вікна. Назва вікна відповідає темі роботи, яка виконується з його допомогою. Крім назви та пояснювальних написів, вікно може мати:

1) кнопки-команди, при їх натисканні відбуваються певні дії;

2)кнопки-перемикачі, кожна має два стани: увімкнений, що відображається наявністю точки чи іншого значка навпроти відповідного напису, і вимкнений;

3) поля для вводу, наприклад, для введення імені таблиці, імені поля таблиці і т.д.;

4) списки для вибору запропонованих об'єктів, наприклад, імен полів

Операції в Access. За допомогою Access можна виконувати такі операції:

1) проектування базових інформаційних систем – двох вимірних таблиць з різними типами даних:

1.1 текстові дані, значення яких уявляють сукупність довільних алфавітно-цифрових символів, довжина яких не перевищує 255 символів;
1.2 числові дані, що в основному використовується для представлення атрибутів, зі значеннями яких потрібно виконувати арифметичні операції;
1.3 дані типу дата і (або) часу задаються в заданому машині форматі (наприклад дд.мм.гг. або мм.дд.рр);
1.4 логічні дані, як можуть приймати тільки одно з двох значень: TRUE або FALSE. Роль таких даних в інформатиці має дуже важливе значення;
1.5 поле об'єкта OLE. Це цікавий тип даних, який може приймати значення у вигляді графіки, звуку, відео; (Object Linking and Embedding).
1.6 типи користувача, які дають можливість утворювати власні типи даних (наприклад, "день тижня", "Адреса").

2) встановлення звязків між таблицями, з підтримкою цілісності даних, каскадного оновлення полів і каскадного вилучення записів.

3) введення, зберігання, перегляд, сортування, модифікація і вибірка даних з таблиць з використанням різних засобів контролю інформації, індексування таблиць і апарату алгебри логіки для фільтрації даних.

4) Утворення, модифікація і використання похідних обєктів бази даних: форм, запитів і звітів.

Форма – не є обов'язковим елементом бази даних (без неї можна обійтися), але вона дозволяє спрощувати операції введення і перегляду даних.

Запити - збирають дані з інших таблиць і виконують над ними різні операції, зокрема, в запиті можуть зявитися обчислювальні поля, тобто поля, значення яких є функціями інших полів (можливо, з різних таблиць). Наприклад, вартість продукції дорівнює добутку кількості товарів з таблиці "Замовлення" на ціну з таблиці "Продукти". Крім того, запити дозволяють виконувати групові операції, тобто операції над групою записів, що об'єднані якою-небудь ознакою. (наприклад, можна просумувати кількість для записів з одним і тим же кодом результатів). На кінець, запити дозволяють складати вибірки з таблиць за певною умовою (наприклад вибрати з таблиці "Замовлення" записи з заданим кодом клієнта). В таких випадках застосовується алгебра логіки.

Звіт – це фактично той же запит, але оформлений так, щоб його можна було надрукувати на папері і подати керівництву (з красивими заголовками, проміжними підсумками і т. д.).

По суті справи, таблиці і форми застосовуються для обчислювання накопичення бази даних, а запити і звіти виконують основну функцію

інформаційних систем (ІС) – вилучення, перетворення і представлення інформації.

Крім того СУБД MS Access має потужні можливості, зручні і гнучкі засоби візуального проектування об'єктів за допомогою майстрів, а це дає можливість користувачеві при мінімумі попередньої підготовки досить швидко утворювати повноцінну ІС - на рівні таблиць, форм, запитів, виборок і звітів.

Доброзичливий інтерфейс дає можливість також спочатку навчитися складати макроси (тобто набори простих інструкцій по управлінню даними); а потім засвоїти підготовку власних програм-додатків мовою VBA.

14.3.Застосування VBA в Access,

У програмі MS Access технологія створення макросів відрізняється від робрти у Word та Excel. Тут і макроси і модулі – окремі об'єкти, які входять в структурі файла бази даних (разом із таблицями, із таблицями, запитами, звітами та формами).

Програма MSAcces містить сукупність готовність макрокоманд, які дозволяють виконати всі операції в процесі розв'язання завдань. Макроси, які є надбудовою на VBA забезпечують користувачеві можливість вирішенн завлань, які не вимагають знань в детальному програмуванні.

В будь-якій мові програмування, зокрема у VBA Access, є свій власний синтаксис. Під синтаксисом вважаються правила комбінування компонентів мови, які дозволяють враховувати відповідну інструкцію і дотримуватися її. У VBA Access для представлення значень можна використат константи і змінні. Константи і змінні необхі для збереження деякої інформации: константи зберігають незмінні значення, а змінні — значення, які можна змінювати, в будь-який час.

Оголошення змінних у VBA ACCES. Перед тим, як використати змінну, її необхідно оголосити за допомогою оператора **Dim**:

Dim *ім'я _змінної* As *тип_даних*

Імя змінної задає користувач , тип даних – це тип, який використовується у VBA Access. Якщо тип даних не вказується, то по замовчуванню використовується тип **Variant**. В одному рядку можна оголосити декілька змінних:

Dim *змінна1* As *тип_даних1*, *змінна2* As *тип_даних2.*

***Імена змінних* В VBA ACCESS.**

1) Іи'я змінної починається з букви;
2) В іиені неможна використовувати символи: **, @ ! # $. %**

3) Імана повинні бути унікальними , тобто не повинні бути двох зінних з одинаковими іменами;
4) Кількість символів в назві змінної повинно бути не більше 255 символів.

Оголошення констант у VBA ACCES. Так як мова йде про оголошення змінних і констант у VBA Access, про змінні розглянуто вище, перейдемо до констант. Для оголошення константи використовується оператор **Const**. Синтаксис наступний:
[Public | Private] Const *ім'я_констани* As *тип_даних = вираз.*

Оголошення константы може починатися ачинаться словами *Public*, *Private* або *Const*

Для автоматизації дій над об'єктами в Microsoft Access і в інших додатках Microsoft Office застосовуються макроси і модулі. *Макроси* - це невеликі програми мовою *макрокоманд* (мова сценариіїв). *Модулі*- це набір описів і процедур мово програмування VBA для додатків, тобто модулі – це об'єкти, що містить програми мовою Visual Basic.

Основне призначення макросів і модулів — це створення зручного інтерфейсу додатку, в якому форми і звіту відкрились би при натисненні кнопок в цих формах або на панелях інструментів. Модулі є більш потужним засобом створення програмних розширен в снредовиші Microsoft Office. Застосування модулів вимагає від користувачів і вимагає знань основних принципів об'єктно-орієнтованого програмування.

Програмування на VBA в СУБД Access використовується в основному розробниками (програмістами) в процесі створення додатків (різних баз даних), з якими працюють користувачі.

Для програмування в Access використовуються вбудована в Microsoft Office система программирования VBA.

В Access існує два типи модулів : *стандартні модулі* і *модулі класу*. Основновний зміст модулів — це *процедури* на мові VBA. *Процедура* - це сукупність описів і інструкцій в модулі, які виконуються як одна програмна одиниця. у VBA існують процедури-підпрограми *Sub* і процедури- функції *Function*.

Стандартні модулі містять загальні процедури, які не зв'язані з конкретним об'єктом (формою, звітом). Стандартний модуль – це модуль, в якому розміщують процедуру *Sub* і *Function,* які повинні бути доступні для усіх процедур в даном додатку. Стандартні модулі можуть використовуватися іншими додатками. Крім загальних процедур, в стандартних модулях можуть міститися глобальні змінні і функції, а також об'єкти, які доступні із інших об'єктів бази даних.

Модуль класу відрізняється від стандартного модуля тим, що крім процедур, він містить опис об'єкта і використовується для створення класів (об'єктів). Окремі модулі класу, розміщених на вкладці Модулі вікон баз даних, містять опис класу (об'єкта), сствореному користувачем. До модулів класу таож відносяться модулі об'єктів (форм, звітів), які зв'язані з конкретними формами або звітами і містить процедури обробки подій форм (звітів) і їх елементів управління.

Модуль об'єкта (форми, звіти) – це модуль класу, що містить програми усіх процедур обрлбки подій, що виконують в конкретному об'єкті (формі, звіті) або в його елементах управлення. Всі процедури подій для форми чи звіті зберігаються в модулі об'єкта (форми чи звіту). Знову створена форма (звіт) не містить модулів, але їх можна створювати декількома способами. Таким чином, форми або звіти, що зв'язані зі створеними модулями об'єктів (форм, звітів). Якщо процедура використовується тільки формою чи звітом, то вона зберігається в коді форми або звіті. Якщо процедура використовується в багатьох в формах і звітах, то вона зберігається в окремому модулі.

Перший спосіб створення пустого модуля: вибрати "*Так*" в полі наявності модуля на вкладці "*Усі*" у вікні діалогу *Форма* чи *Звіт*. Вікно діалогу викликається командою "*Властивості*" з контекстного меню, що знаходиться в конструкторі форм чи звітів.

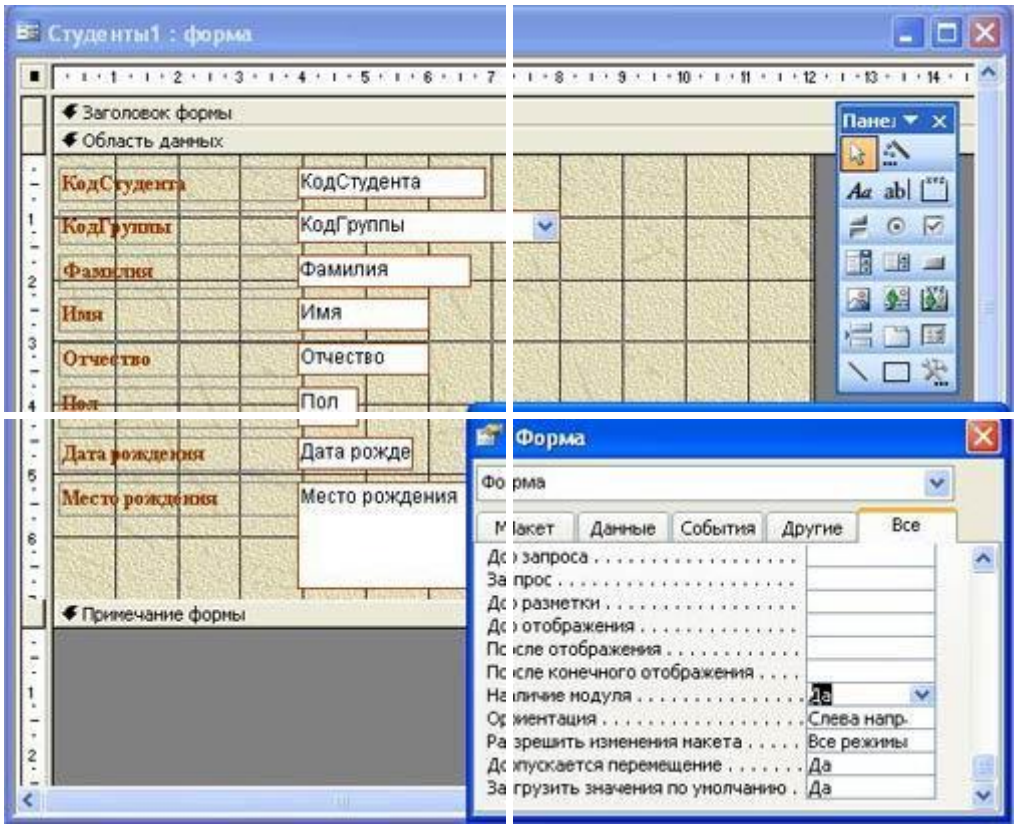


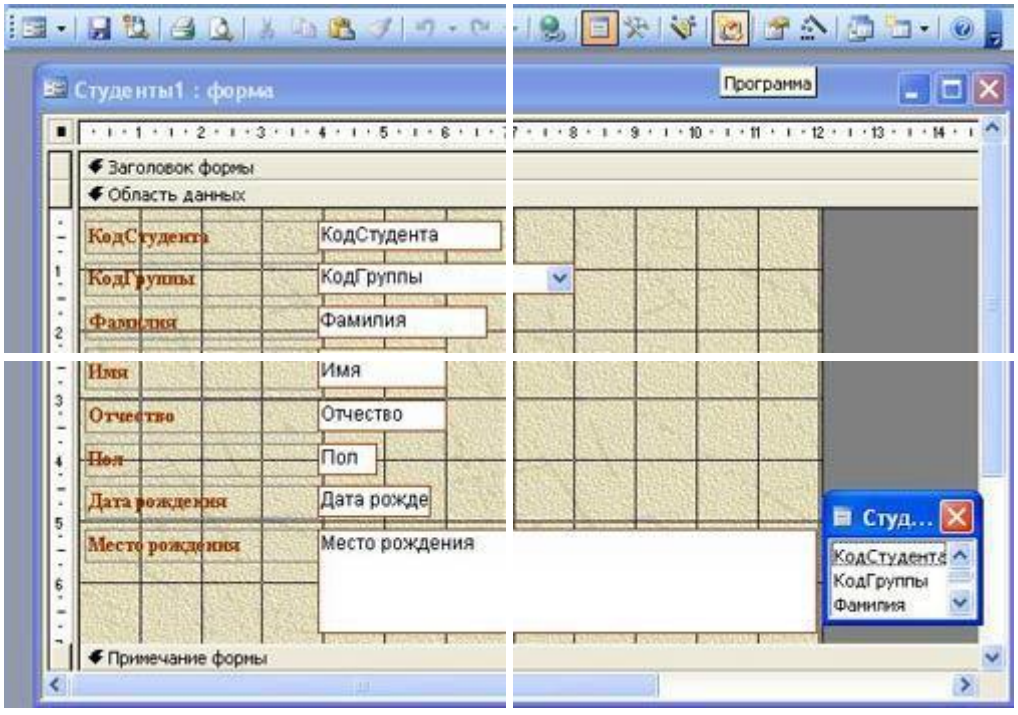
Рис.14.3. . Перший спосіб створення пустого модуля.

Другий спосіб створення модуля виконується кнопкой *"Програма"* на панелі інструментів в режимі конструктора форм чи звітів. Якщо виконати фіксація курсором мишкою по кнопці *"Програма"*, то запуститься редактор VBA, в якому можна вводити текст програми відповідного модуля (Рис.14.4.).

Третій спосіб здійснюється шляхом обробки деяких подій, що зв'язані з формою або яким-небудь елементом управлення форми, що знаходяться в режимі *Конструктора форм* чи *звітів* (Рис.14.5).

Для цього необхідно відкртит вікно редактора VBA, виконавши фіксацію по команді *"Программы"*..

Для перегляду модулів класу, які зв'язані з конкретними формами або звітами і міститься в модулях об'єктів, потрібно виділити форму або звіт у вікні бази даних на вкладці форми чи звіті і виконати фіксацію по піктограмі *Програма* на панелі інструментів в головному вікні Microsoft Access.



Ртс.14.4. Другий спосіб створення модуля

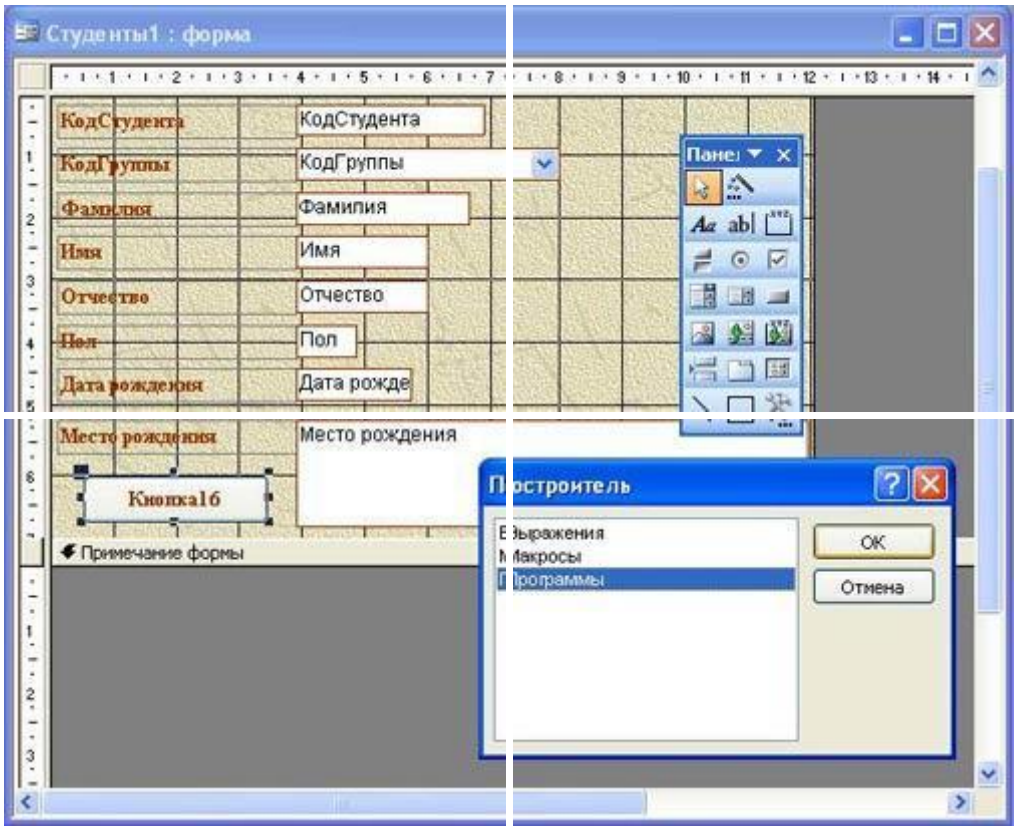


Рис.14.5. Третій спосіб створення модуля

Щоб створити окремий модуль класу або стандартний модуль, потрібно вибрати пункт *Модуль класу* або *Модуль* в меню *Вставка*. Стандартний модуль можна створити, наприклад, шляхом перетворення макросу. Модулі відображаються у вікні бази даних на вкладці *Модулі*. На рисунке 14.6. представлено вікно бази даних Access (на вкладці *Модулі*), в якому знаходяться модулі об’єктів, три стандартні модулі і один модуль класу.

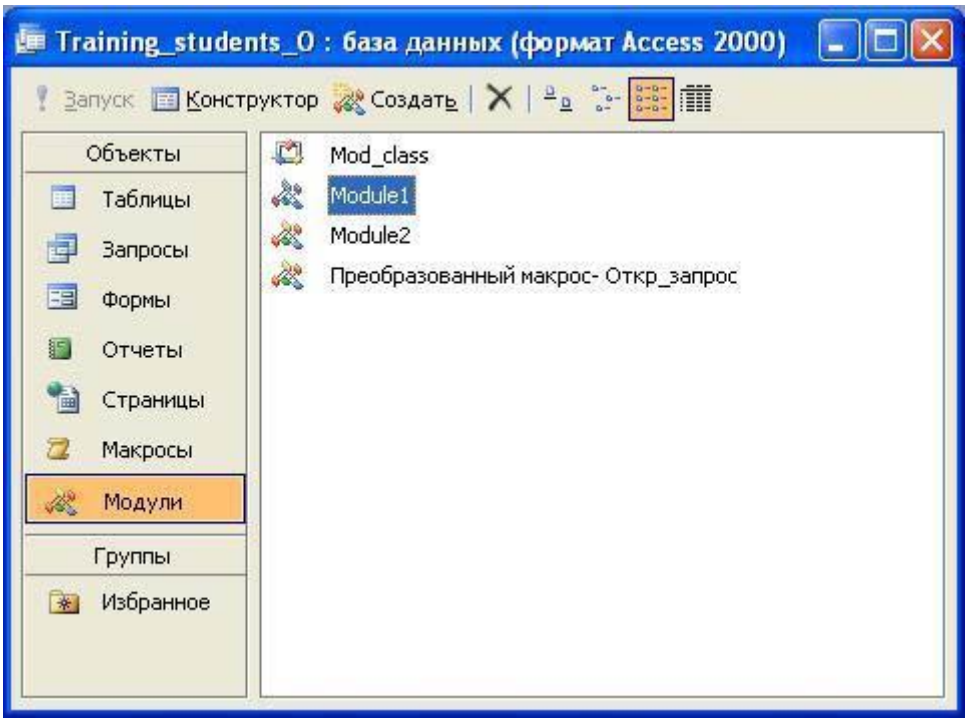


Рис. 14.6. Представлення вікна бази даних Access

Для перегляду процедури в стандартному модулі або в модулі класу потрібно виділити потрібний модуль у вікні бази даних на вкладці *Модулі* і виконати фіксацію по піктограмі *Програма* на панелі інструментів в главному вікні Microsoft Access або виконати фіксацію по кнопці *Конструктор* у вікні бази даних. Відкриється редактор VBA на панелі редактора коду, що відображає процедура.

В СУБД Access для кожного об'єкти визначені можливі *події*. Деякі *події* виникають від дії користувачів (фіксація курсора мишки, натиснення клавіші клавіатури тощо), а друга частина подій здійснюється в результате виконання інших подій, наприклад відкриття. Кажна подія проявляється в певних діях програми.

Існують дві групи *дій* на події. Дії *першої групи* визначені властивостями об'єкта і їх змінити в процесі програмування на VBA неможна. Ці властивості встановлюються ОС Windows і системою програмування VBA. Прикладом може бути згорнення вікна при фіксації по кнопці *Звернути*. *Друга група* дій (відклику) на події визначається програмістом. Для цієї групи дій програміст може створити процедури VBA для кожного можливої події, але на практиці програміст заповнює кодом процедури тільки для тих подій, які потрібні для даного додатку ного Access.

14.4. Стардатні мікрокоманди VBA Access

Модулі і макроси вбазі даних в Access – об’єкти бази даних, для яких відведені у вікні окремі вкладки.

Бази даних, яка призначена для автоматизації окремих операцій часто містять багато таблиць, форм, запитів та звітів. Макроси доцільно розробляти для автоматизації декількох процесів, зокрема для відкриття та закриття об’єктів бази даних (БД), виведення на екран чи друк декілька документів, встановлення та відміни відображення панеля інструменті тощо.

Макроси Access – об’єкти , які містять сукупність стандартних макросів із використанням користувачем властивостей, які відображені в табиці 14.1.

Таблиця 14.1 . Стандартні макрокоманди з переліком властивостей

Макрокоманда (властивість)	**Опис**
Open Form (Відкрити форму)	**Відрти форму**
Form Name (Ім’я форми)	Ім’я форми, яка з’являється у вікні бази даних
Viem (Режим)	Режим, в якому відображається форма
Filter Name (Ім’я фільтра)	Ім’я збереження фільтру чи запиту, яке використовується для фільтрування даних форми
Where Condition (Умова відбору)	Вираз WHERE SQL, який використовується для фільтрації даних форми
Data Mode (Режим даних)	Вибір параметрів введення даних
Window Mode (Режим вікна)	Режим відкриття вікна форми (нормальне, заховане, згорнуте тощо)
Open Report (Відкрити звіт)	**Відкриває звіт**
Report Name (І’мя звіту)	Ім’я звіту, яке з’являється у вікні бази даних
View (Режим)	Режим, якому відображається звіт
Filter Name (Ім’я фільтра)	Ім’я збереження фільтру чи запити, який використовується для фільтрування даних звіту
Where Condition (Умова відбору)	Вираз WHERE SQL, який використовується для фільтрації даних звіту

Open Qurery (Відкрити запит)	**Відкриває запит на вибірку чи перехресний запит або запускає запит макрокоманди**
Qurery Name (Ім'я запиту)	Ім'я запиту, який з'являється у вікні бази даних
View (Режим)	Режим відображання запиту
Date Mode (Режим даних)	Режим введення для запиту на вибірку, який відкритий у режимі *Таблиця*
Close (Закрити)	**Закрити вікно Access або активне вікно бази даних**
Object Type (Тип об'єкта)	Тип об'єкта бази даних, який необхідно закрити
Object Name (Ім'я об'єкта)	Ім'я об'єкту, який закривається
Save (Зберігання)	Визначає, чи зберігається об'єкт при його закритті
MsgBox (Повідомлення)	**Виводить вікно з повідомленням або з попередженням**
Message (Повідомлення)	Текс у вікні повідомлень (до 255 символів)
Beep (Сигнал)	Визначає, чи виведення вікна з повідомленням буде супроводжуватися звуковим сигналом
Type (Тип)	Тип вікна з повідомленням (критичне, повідомлення, попередження , інформаційне тощо).
Title (ЗаголGoовок)	Текст, який виводиться у заголовку вікна повідомлнь
SetValue (Задати Значення)	**Встановлює для форми чи звіту значення поля, елемент керування чи властивість**
Item (Елемент)	Ім'я поля, елемента керування чи властивість, для яких встановлюється значення
Expression (Вираз)	Вираз, за допомогою встановлюється значення
GoToRecord (ПерейтиДозапис)	**Здійснює перехід на конкретний запис у відкриій таблиці, формі чи запиті**
Object Type	Тип об'єкту бази даних, який містить запис

(Тип об'єкту)	
Object Name (Ім'я об'єкту)	Ім'я об'єкту бази даних, який містить запис
Record (Запис)	Запис, на який здійснюється перехід (може бути поточним)
Offset (Зміщення)	Номер запису, до якого здійснюється перехід (відповідає числу *Номер запису* в області *Запис вікна форми)*

14.5.Створення, редагування та запус макросів в Access

В прогамі Access макроси створюються в спеціальному вікні – конструктора макросів , яке відображається на екрані , після натиснення кнопки *Створити* на вкладці *Макроси* вікна програми.

По замовчуванню, у вікні конструктора відображаються створені *Макрокоманда* і *Зауваження.* Крім них, можна також відобразити стовбці *Ім'я макроса* та *Умови* за допомогою відповідних кнопок на панелі інструментів чи пуктів меню команди *Вид.*

Створення макроса в Access здійснюється в діалоговому режимі й полягає в записі у вікні макросу послідовності команд і параметрів для них. Кожному макросу присоюється ім'я. При виконанні макросу, команди які в ньому містяться, виконуються в тому порядку, в якрму вони записані.

Процес створення макросів в середовищі Access суттєво відрізняються від створення макросів в Word і Excel. Тут автоматичного запису макросу не відбувається. При створенні макросу команди вводяться в комірки стовпця *Макрокоманда* вікна конструктора. Команди вибираються зі списку, який розгортається після активізації комірки (можна також ввести команду з клавіатури, про те ций спосіб вимагає досконалого знання усіх команд). В нижній частинв конструктора відображається область з аргументів цієї команди. З начення аргументів або вибирають зі списків, які розгортається, або вводять із клавіатури. У комірці стовпця *Примітка* можна ввести коментарі для макрокоманд. Для для *збереження макроса* необхідно виконати таку послідовність команд : ***Файл >Зберегти*** або натиснути відповідну кнопку вікна програми.

Запустити макрои на виконання можна різними способами :

1) з будь-якого вікна програми виконати таку пслівність команд: ***Сервіс>Макрос>Виконати макрос*** і вибрати із списку діалогового вікна *Запуск макроса* потрібний об'єкт.

2) з вікна бази даних перейти на вкладку *Макроси,* позначити піктограму потрібного об'єкта та натиснути клавішу ***Enter*** або кнопку **Запуск** вікна бази даних.
3) з вікна конструктора макросу натиснути кнопку ***Запуск*** панелі інструментів вікна програми.

Для редагування збереженого макроса потрібно на вкладці макросі позначити піктограму відповідного макроса та відобразити вікно конструктора за допомогою:
1) кнопки ***Констуктор*** панелі задач на базі даних;
2) кнопки ***Констукто*** контекстного мегю виділеного об'єкту.

При цьому розгорнеться вікно конструктора макроса з переліком введених команд та їх властивостей.

Питання для самоконтролю

1. Яка різниця між банком даних і базою даних ?.
2. Що таке СУБД та її основне призначення ?
3. Яка різниця між логічною і фізичною структурою даних?
4. Які існують моделі баз даних і в чому між ними різниця ?
5. Чим відрізняється СУБД Access від попередніх версій ?
6. Які операції в Access можна виконувати ?
7. Які похідні об'єкти бази даних і яка між ними різниця ?
8. Чим відрізняється технологія створення макросів у програмі MS Access від роботи у Word та Excel ?
9. Як виконується оголошення змінних у VBA Access ?
10. Які вимоги пред'являють до імен змінних у VBA Access ?
11. Чим відрізняється модуль класу від стандартного модуля у VBA Access ?
12. Перелічите спосіб створення модуля у VBA Access .
13. Чим відрізняється діїї першої групи при визначені властивостей об'єкта від другої групи у VBA Access ?
14. В яких випадках варта використовувати макроси, а коли модулі у VBA Access ?
15. Чим відрізняються процес створення макросів в середовищі Access від створення макросів в Word і Excel ?
16. Якими способами можна запустити макрои у VBA Access ?

Розділ. 15.Письмова комунікація та супровід програмного забезпечення

Супровід - це звичайний процес зміни системи після її поставки замовникові. Ці зміни можуть бути як елементарно простими (виправлення помилок програмування), так і більше серйозними, пов'язаними з коректуванням окремих недоробок або приведенням у відповідність із новими вимогами. Слід зауважити що, супровід не зв'язаний зі значною зміною архітектури системи. При супроводі тактика проста: зміна існуючих компонентів системи або додавання нових.

Існує три види супроводу системи.

1. **Супровід з метою виправлення помилок.** Звичайно помилки в програмуванні досить легко виправити, однак помилки проектування коштують дорого й вимагають корегування або перепрограмування деяких компонентів. Найдорожчі виправлення пов'язані з помилками в системних вимогах, тому що тут може знадобитися перепроектування системи.

2. **Супровід з метою адаптації програмного забезпечення (ПЗ) до специфічних умов експлуатації.** Це може знадобитися при зміні певних складових системи, наприклад апаратних засобів, операційної системи або програмних засобів підтримки. Щоб адаптуватися до цих змін, система повинна бути піддана певним модифікаціям.

3. **Супров д з метою зміни функціональних можливостей системи.** У відповідь на організаційні або ділові зміни в організації можуть змінитися вимоги до програмних засобів. У таких випадках застосовується даний тип супроводу. Найбільш істотні зміни при цьому відбуваються саме з програмним забезпеченням.

На практиці однозначно чітке розмежування між різними видами супроводу провести досить складно. Помилки в системі можуть бути виявлені в тому випадку, якщо, наприклад, система використалася непередбаченим способом. Тому найкращий спосіб виправлення помилок — розширення функціональних можливостей програми для того, щоб зробити роботу з нею як можна простішою. При адаптації програмного забезпечення до нового робочого оточення розширення функціональних можливостей системи буде сприяти поліпшенню її роботи. Також додавання певних функцій у програму може виявитися корисним, якщо у випадку помилок був змінений шаблон використання системи й побічною дією при розширенні функціональних можливостей буде вилучення помилок.

Перераховані типи супроводу широко використаються, хоча їм часом, дають різні назви. Супровід з метою виправлення помилок звичайно називають коригувальним. Назва "адаптивний супровід" може

ставитися як адаптація до нового робочого оточення, так і до нових вимог. Удосконалення програмного забезпечення може означати поліпшення шляхом відповідності новим вимогам, а також удосконалення структури й продуктивності зі збереженням функціональних можливостей.

Знайти сучасні дані щодо того, як часто використається той або інший тип супроводу, буде нелегко. Відповідно до досліджень , які вже трохи застаріли, 65% супроводу пов'язане з виконанням нових вимог, 18% приділяється на зміни системи з метою адаптації до нового оточення й 17% пов'язане з виправленням помилок.

Із цього можна зробити висновок, що виправлення помилок не є найпоширенішим видом супроводу. Модернізація системи відповідно до нового робочого оточення або відповідно до нових вимог більше ефективна. Тому супровід сам по собі є природним процесом продовження розробки системи зі своїми процесами проектування, реалізації й тестування.

Значна частина бюджету більшості організацій іде на супровід ПЗ, а не на саме використання програмних систем. В 1980-х роках було виявлено, що в багатьох організаціях щонайменше 50% всіх засобів, витрачених на програмування, іде на розвиток уже існуючих систем. З аналізу літературних джерел встановлено схоже співвідношення витрат на різні види супроводу, при цьому від 65 до 75% засобів загального бюджету витрачається на супровід тому що, підприємства заміняють старі системи комерційними ПЗ, наприклад для програм планування ресурсів, ці цифри ніяк не будуть зменшуватися. Тому можна стверджувати, що зміна ПЗ усе ще залишається домінуючої в статті витрат організацій на програмне забезпечення.

Співвідношення між величинами засобів на супровід і на розробку може бути різним залежно від предметної області, де експлуатується система. Для прикладних систем, що працюють у діловій сфері, співвідношення витрат на супровід в основному відповідає засобам, витраченим на розробку. Для убудованих систем реального часу витрати на супровід можуть у чотири рази перевищувати вартість самої розробки. Високі вимоги відносно продуктивності й надійності таких систем припускають їхню тверду структуру, що сутужніше піддається модифікації.

Можна одержати значну загальну економію засобів, якщо заздалегідь витратити фінанси й зусилля на створення систем, що не вимагають дорогого супроводу. Досить важко проводити зміни в системі після її встановлення замовникові, оскільки для цього потрібно добре

знати систему й провести аналіз реалізації цих змін. Тому зусилля, витрачені під час розробки програми на зниження вартості такого аналізу, автоматично знизять і витрати на супровід. Такі технології розробки ПЗ, як формування чітких вимог, об'єктно-орієнтоване програмування й керування конфігурацією, сприяють зниженню вартості супроводу.

Із аналізу досвіду програмістів випливає, що збільшення засобів на розробку системи пропорційно знизить витрати на її супровід.

Причиною високих витрат на супровід є складність модернізації системи після її впровадження, оскільки розширити функціональні можливості набагато легше в процесі створення системи. Нижче наведені ключові фактори, які визначають вартість розробки й супроводу та можуть привести до подорожчання супроводу.

Стабільність команди програмістів. Цілком природно, що після впровадження системи команда програмістів розпадається, фахівці будуть працювати над іншими проектами. Новим членам команди або ж окремих фахівців, які візьмуть на себе подальший супровід системи, буде важко зрозуміти всі особливості. Тому на розуміння системи перед внесенням у неї змін іде багато часу й засобів.

Відповідальність відповідно до контракту. Контракт на супровід звичайно полягає окремо від договору на розробку програми. Більше того, часто контракт на супровід може одержати фірма, яка сама не займається розробкою. Разом з фактором нестабільності команди це може стати причиною відсутності в команді стимулу створити легко замінну, зручну в супроводі систему. Якщо членам команди вигідно піти найкоротшим шляхом з мінімальними витратами зусиль, то навряд чи вони відмовляться від цього навіть із ризиком підвищення наступних витрат на супровід.

Кваліфікація фахівців. Фахівці, що займаються супроводом, часто не знайомі із предметною областю, де експлуатується система. Супровід не користується популярністю серед програмістів. Це вважається менш кваліфікованою розробкою й часто доручається молодшому персоналу. Більше того, старі системи можуть бути написані на застарілих мовах програмування, не знайомих молодим фахівцям які потребують додаткового навчання.

Вік і структура програми. З віком структура програм порушується внаслідок частих змін, тому їх стає складніше розуміти й змінювати. Крім того, багато попередніх систем були створені без використання сучасних технологій. Вони ніколи не відрізнялися високою якісною структурою; зміни, зроблені в них, були спрямовані скоріше на підвищення

ефективності функціонування, ніж на підвищення зручності супроводу. Документація на старі системи часто буває неповною або взагалі відсутня.

Перші три ключові фактори об'єднані тим, що багато організацій усе ще роблять розходження між розробкою системи і її супроводом. Супровід вважається справою другорядною, тому немає ніякого бажання інвестувати засоби для зниження витрат на майбутній супровід. Керівництву організацій необхідно пам'ятати, що в систем рідко буває чітко певний строк функціонування, навпаки, вони можуть мати невизначений час кваліфікованої роботи.

Проблема полягає в наступному: створювати системи й підтримувати їх доти, поки це можливо, і потім заміняти їх новими, або розробляти еволюційні системи, які можуть змінюватися відповідно до нових вимог. Їх можна удосконалювати на основі попередніх систем, поліпшуючи структуру або зміну архітектури останніх.

Профілактичні міри при супроводі будуть корисні, якщо виникне необхідність удосконалити систему й зробити її більше зручної для змін.

Процес супроводу. Процеси супроводу можуть бути самими різними, що залежить від типу програмного забезпечення, технології його розробки, а також від фахівців, які безпосередньо займалися створенням системи. У багатьох організаціях супровід носить неформальний характер. У більшості випадків програмісти одержують інформацію про проблеми системи від самих користувачів в усній формі. Інші ж компанії мають формальний процес супроводу зі структурованою документацією на кожний його етан. На самому загальному рівні будь-які процеси супроводу мають такі етапи; аналіз змін, планування версій, реалізація нової версії системи й встановлення системи замовникові.

Процес супроводу починається при наявності достатньої кількості запитів па зміни від користувачів, менеджерів або покупців. Далі оцінюються можливі зміни для того, щоб визначити рівень модернізації системи, а також вартість впровадження цих змін. Якщо приймається рішення про модернізації системи, починається етап планування нової версії системи. Під час планування аналізується можливість реалізації всіх необхідних змін, чи це виправлення помилок, адаптація чи розширення функціональних можливостей системи. Тільки після цього приймається остаточне рішення про те, які саме зміни будуть внесені в систему. Коли зміни реалізовані, виходить чергова версія системи..

Цей процес модернізації повинен привести до зміни системної специфікації, архітектури й програмної реалізації. Нові вимоги повинні відображати зміни, внесені в систему. Введення в систему нових

компонентів вимагає її перепроектування, після чого необхідно повторне тестування системи. Для аналізу внесених змін при необхідності можна створити прототип системи. На цій стадії проводиться докладний аналіз змін, при якому можуть виявитися ті наслідки модернізації, які не були встановлені при початковому аналізі змін

Іноді в екстрених випадках потрібне швидке внесення змін, наприклад з наступних причин:.

1. Збій у системі, внаслідок чого виникла надзвичайна ситуація, що вимагає екстреного втручання для продовження нормальної роботи системи.

2. Зміна робочого оточення системи з непередбачуваним впливом на неї.

3. Несподівані зміни в діловій сфері організації (через дії конкурентів або через введення нового законодавства).

У таких випадках швидка реалізація змін має більшу важливість, ніж чітке проходження формальності процесу модернізації системи. Замість того щоб змінювати вимоги або структуру системи, краще швидко внести корегування в програмний код . Однак цей підхід небезпечний тим, що вимоги, системна архітектура і програмний код поступово губить цілісність. Цього важко уникнути, якщо взяти до уваги необхідність швидкого виконання завдання, коли ретельна доробка системи відкладається на потім. Якщо програміст, що змінив код, раптово йде з команди, то його колезі буде складно привести у відповідність зробленим змінам специфікацію й структуру системи.

Ще одна проблема термінових змін системи полягає в тому, що через дефіцит часу із двох можливих рішень буде прийнято не найкраще (в аспекті збереження структури системи), а те, яке можна швидше й ефективніше реалізувати систему. В ідеалі після термінової корекції коду системи запит на зміни повинен усе ще залишатися в силі. Тому після ретельного аналізу змін можна скасувати внесені зміни й прийняти більш оптимальне рішення для модернізації системи. Однак, на практиці така можливість використається вкрай рідко, найчастіше в силу суб'єктивних причин.

Прогнозування супроводу Менеджери терпіти нс можуть сюрпризів, особливо якщо вони виливаються в непередбачене високі витрати. Тому краще передбачити заздалегідь, які зміни можливі в системі, з якими компонентами системи буде найбільше проблем при супроводі, а також розрахувати загальні витрати на супровід протягом певного періоду часу.

Прогнозування кількості запитів на зміни системи залежить від розуміння взаємозв'язків між системою і її оточенням. Деякі системи перебувають у досить складній взаємозалежності із зовнішнім оточенням і зміна оточення обов’язково вплине на систему. Для того щоб правильно судити про ці взаємини, необхідно оцінити наступні показники.

Кількість і складність системних інтерфейсів. Чим більше системних інтерфейсів і чим більше складними вони є, тим вище ймовірність змін у майбутньому.

Кількість системних вимог, що змінюються. Як згадувалося в главі 6, вимоги, що відображають ділову сферу або стандарти організації, частіше змінюються, ніж вимоги, що описують предметну область.

Бізнеси-процеси, у яких використається дана система. Розвиток бізнес- *процеси* приводить до появи нових вимог до системи.

Щоб коректно спрогнозувати процес супроводу, потрібно знати кількість і тини взаємозв'язків між різними компонентами системи, а також ураховувати складність цих компонентів. Дослідження присвячені взаємозв'язкам між складністю систем і процесом супроводу показали, що, чим вища складність системи і її компонентів, тим більш дорогим виявиться супровід, чого й слід було очікувати.

Наприклад, при проведені досліджень комерційних програм, написаних мовою COBOL, з використанням різних методик виміру складності, включаючи розмір процедур, розмір модулів і кількість розгалужень, яка визначає складність системи, встановлено, що зниження складності програмування значно скорочує витрати на супровід системи.

Вимір рівня складності систем виявився досить корисним для встановлення тих компонентів систем, які будуть особливо складні для супроводу. Результати аналізу ряду системних компонентів показали, що супровід часто зосереджений на обслуговуванні невеликої кількості частин системи, які відрізняються особливою складністю. Тому, економічно вигідно замінити складні системні компоненти більше простими відповідними версіями.

Після введення системи в експлуатацію з'являються дані, що дозволяють прогнозувати подальший супровід системи. Перераховані нижче показники корисні для оцінювання зручності супроводу.

Кількість запитів на корегування системи. Зростання кількості звітів про збої в системі означає збільшення кількості помилок, що підлягають виправленню при супроводі. Це говорить про погіршення зручності супроводу.

Середній час, витрачений па аналіз причин системних збоїв і відмов. Цей показник пропорційний кількості системних компонентів, у які потрібно внести зміни. Якщо цей показник зростає, система вимагає численних змін.

Середній час, необхідне на реалізацію змін. Не слід плутати цей показник з попередньої, хоча вони тісно зв'язані. Тут ураховується не час аналізу системи по виявленню причин збоїв, а час реалізації змін й їхнього документування, що залежить від складності програмного коду. Збільшення цього показника означає складність супроводу.

Якість незавершених запитів не піддається покращенню. Зі зростанням кількості таких запитів утрудняється супровід системи.

Для визначення вартості супроводу використається попередня інформація про запити на зміни. прогнозування щодо зручності супроводу системи. У рішенні цього питання більшості менеджерів допоможе також інтуїція й досвід.

Питання для самоконтролю.

1.Що таке супровід програмного забезпечення ?
2.Які види супроводу програмного забезпечення використовують?
3.Що сприяє зниженню вартості супроводу програмного забезпечення ?
4.Які ключові фактори, що визначають вартість розробки й супроводу можуть привести до подорожчання програмного забезпечення ?
5.В яких випадках потрібне швидке внесення змін супроводі програмного забезпечення ?
6.Які показники необхідно оцінити, щоб прогнозувати кількість запитів на зміни в систем на основі взаємозалежності її із зовнішнім оточенням ?
7.Які показники корисні для оцінювання зручності супроводу програмних продуктів ?

Розділ 16. Основи процесу розробки програмного забезпечення згідно вимог Software Engineering 2004

В проекті **«Software Engineering 2004»,** зібрано всесвітній досвід викладання програмної інженерії в університетах і коледжах, де приводиться безліч шаблонів складання учбових програм для різних країн з урахуванням їх традицій викладання. На жаль, серед цієї множини немає шаблону, що враховує особливості освіти в країнах СНД, тому шо в цих кhаїнах відбулося становлення програмної інженерії як самостійної дисципліни.

Основною метою даного проекту є розробка рекомендацій для освітніх установ і агентств по акредитації змісту учбових планів для підготовки бакалаврів в області програмної інженерії. Ці рекомендації були розроблені групою добровольців, що складається з великого числа фахівців з різних країн. При складанні даного документа враховувалися результати, отримані в області навчання програмної інженерії за о станніх 25 років. Зараз рекомендації до складання учбових планів по програмній інженерії особливо актуальні, оскільки у ряді країн різко зросла кількість новостворюваних програм навчання програмної інженерії, і виникає потреба в процесі акредитації цих програм.

Розробка даного документа здійснювалася в три основні етапи із залученням великої кількості добровольців, а також всіх членів організаційного комітету. Перший етап включав розробку набору очікуваних результатів навчання і об'єму знань, необхідних кожному випускникові, що навчався за фахом «Програмна інженерія». В рамках другого етапу були визначені і класифіковані знання, які повинні бути включені в матеріал, що викладається в університетах, по програмній інженерії (SEEK). На третьому етапі був розроблений набір рекомендацій по створенню учбових планів, що описує, яким чином учбовий план по програмній інженерії, що включає SEEK, може бути структурований в різних контекстах.

На дослідження і практику в області програмної інженерії впливають як її коріння в інформатиці, так і її встановлення як самостійна інженерна дисципліна.

Слід відзначити, що одночасно із значною схожістю між програмною інженерією і традиційною, існують і деякі відмінності (не обов'язково в збиток програмної інженерії):

- Підставою програмної інженерії є інформатика, а не природні науки.
- Основою є дискретна, а не безперервна (класична) математика.
- Концентрація на абстрактні логічні об'єкти замість використання конкретних фізичних об’єктів.
- Відсутність «виробничої» фази в традиційному промисловому процесі.
- «Супровід» програмного забезпечення в основному пов'язаний з розробкою, що продовжується, або еволюцією, а не з традиційним фізичним зносом.

Існує набір характеристик, що є не тільки загальними для всіх інженерних дисциплін, але і настільки істотних і критичних, що вони можуть використовуватися для опису основ інженерії як такий. Саме такі характеристики повинні розглядатися як бажані для всіх програмних

інженерів (software engineers). Цей набір характеристик зробив істотний вплив як на розвиток програмної інженерії, так і на зміст даного документа (Software Engineering 2004, SE2004):

1. Інженери в своїй діяльності приймають ряд рішень, ретельно оцінюючи альтернативи і вибираючи в кожній точці ухвалення рішення підхід, оптимально відповідний вирішуваному завданню з урахуванням існуючого контексту. Вибір підходу здійснюється в процесі аналізу альтернатив, під час якого ретельно зіставляються можливі витрати і очікуваний прибуток.

2. Інженери, по можливості, працюють з використанням вимірних кількісних характеристик; вони удосконалюють і уточнюють існуючі методи вимірювань і при необхідності видають наближені рішення на основі досвіду і емпіричних даних.

3. Інженери надають особливе значення використанню дисциплінованого процесу при здійсненні проекту і розуміють важливість питань ефективної організації командної роботи.

4. Інженери можуть відповідати за виконання найширшого спектру завдань, починаючи з досліджень, розробки, проектування, виробництва, тестування, впровадження, експлуатації управління, та закінчуючи продажами, консультуванням і навчанням.

5. Інженери в процесі виконання своїх обов'язків широко використовують інструментальні засоби. Тому вибір і використання засобів, що є схожими, є украй важливим питанням.

6. Об'єднуючись в професійні співтовариства, інженери сприяють розвитку своєї галузі шляхом розробки і впровадження рекомендацій, атестаційних принципів, стандартів, розповсюдженню підходів, що добре зарекомендували себе (best practices).

7. Інженери повторно використовують (reuse) результати проектування і проектні об’єкти. .

Слід підкреслити, що проектування є важливою складовою будь-якої інженерної діяльності і відіграє важливу роль при розробці програмного забезпечення. Діяльність в рамках інженерного проектування пов'язана із створенням нових об’єктів шляхом знаходження технічних рішень для специфічних практичних завдань з урахуванням економічних, юридичних і соціальних міркувань. Інженерне проектування, по суті, надає необхідні умови для «фізичної» реалізації рішення, що досягається шляхом проходження систематичному процесу, який найкращим чином задовольняє набору поставлених вимог в рамках потенційних суперечливих обмежень.

Програмна інженерія відрізняється від традиційної інженерії особливою природою програмного забезпечення, у зв'язку з чим основна увага зосереджена на абстракції, моделюванні, організації і представленні інформації, а також на управлінні змінами. Програмна інженерія також включає діяльність по реалізації проекту і контролю якості, яка в традиційному інженерному циклі зазвичай відноситься до фаз проектування виробничого процесу і виробництва. Крім того, безперервна еволюція (тобто «супровід») також є критично важливою для програмного забезпечення.

Отже, центральним завданням програмної інженерії є інженерне проектування — різновид діяльності по ухваленню рішень. Важливим аспектом даного завдання є необхідність застосування відповідних процесів на численних рівнях абстракції. Зростаюча популярність підходів, заснованих на повторному використанні компонент, вселяє надію на появу нових, способів, що покращують роботу в даній області.

Для того, щоб оцінювати можливі рішення з урахуванням різних чинників, пов'язаних з функціонуванням, вартістю, продуктивністю і технологічністю, інженер повинен володіти досвідом і освітою у відповідній наочній області.

Інженерам необхідно ухвалювати рішення, які із стандартних елементів можуть бути використані в здійснюваному проекті, а які необхідно розробити з нуля. Для ухвалення такого роду рішень вони повинні володіти основними знаннями в конкретній наочній області і, можливо, пов'язаних з нею областях.

Ефективне використання специфічних для предметної області методів, засобів і компонент в більшості випадків забезпечує успішність розробок з використанням програмної інженерії. Прекрасних результатів досягають в добре вивчених предметних областях, де широко застосовуються різні стандартизовані підходи до реалізації. Бакалаври програмній інженерії повинні бути знайомі хоч би з однією з прикладних предметних областей, наприклад, нафтогазової . Тобто вони повинні розуміти коло завдань, які визначають предметну область, а також загальні підходи, включаючи стандартні компоненти (якщо такі є), що використовуються у виробництві програмного забезпечення для вирішення завдань даної предметної області.

Проблема критичної залежності суспільства від якості і вартості програмного забезпечення в умовах відносної незрілості програмної інженерії робить питання професіоналізму ще важливішим бакалаврів з програмної інженерії, чим для інших інженерних освітньо-професійних

напрямів.. Випускникам за фахом «програмна інженерія» необхідне прийти на робочі місця підготовленими як до вирішення реальних завдань, так і бути готовим до забезпечення скрізь, де це доречно і допустимо, уміти знаходити необхідну для ухвалення рішень кількісну інформацію, а також бути здатними ефективно функціонувати в умовах невизначеності і уникати невиправданих спрощень при моделюванні. Програмна інженерія як професія має певні зобов'язання перед суспільством. Продукти, створені програмістами, впливають на життя і діяльність клієнтів і користувачів. Очевидно, що розробники програмного забезпечення повинні діяти етично і професійно. Унаслідок специфіки своїх ролей в процесі створення програмних систем інженери по програмному забезпеченню мають необмежені можливості приносити користь або заподіювати шкоду як самостійно, так і сприяючи іншим або впливаючи на інших. Інженери повинні прийняти на себе зобов'язання зробити програмну інженерію корисною і поважаною професією, щоб бути упевненими в тому, що їх робота використовується на благо. Як наслідок даного зобов'язання інженери по програмному забезпеченню повинні строго дотримуватися етичних норм професіонала в області програмної інженерії.

Розвиток і широке поширення засобів обчислювальної техніки в останні десятиліття послужило поштовхом до розробки програмного забезпечення різного рівня складності та різного за призначенням.

У не професійних розробників програмних продуктів існує думка, що можна розробити якийсь один універсальний метод, який забезпечить розробку програмного забезпечення тривіально. Практики в області професійної розробки програмного забезпечення знають, що ніяких універсальних методів не існує.

З кожним роком програмне забезпечення стає все більш складним, об’ємним та вимагає більших капітальних затрат. Програмне забезпечення(ПЗ), як правило, створюється великими командами професіоналів, які представляють різні сфери інтересів часто далекі від комп’ютерних наук.

Під час розробки програмного забезпечення виникають наступні питання:

1. Складність програмного забезпеченняю
2. Як організувати командну роботу ?
3. Як оптимально організувати спілкування у групі професіоналів різних дисциплін ?

4. Які методи необхідно використовувати щоб виконати якісний і не дуже дорогий програмний продукт в необхідні терміни ?

Рішення всіх цих питань приводить до успішної розробки програмного забезпечення.

На сьогодні інформаційні системи, які мають практичний інтерес є складними, вимагають, в процесі розробки, багато часу, значних коштів і нажаль, у більшості випадків, вони є незадовільні. Аналіз ринку розробки програмного забезпечення показує, що зі всіх проектів, що розробляються, лише третина є успішною. Причинами складності програмного забезпечення є :

1. Велика кількість напрямків в інформаційних технологіях.
2. Складнощі спілкування членів однієї команди різних напрямків.
3. Динамічні зміни в технологіях та доступних технічних і програмних засобах
4. Зміна вимог коритсувача і непевність в процесі формулювання вимог

Розробка програмного забезпечення не у всіз випадках є успішною, тому в процесі розробки виникають наступні питання:

1) Що необхідно зробити, щоб підвищити шанси успішності розробки ПЗ?

2) Як бути впевненим в тому, що результати роботи адовольнять користувача?

3) Як перевірити безпомилковість ПЗ?

4) Як сформулювати вимоги до продукту, щоб ці вимоги були зрозумілі для замовника, який не завжди має великий досвід роботи з ПЗ і з другої строни, щоб ці вимоги були чіткими, зрозумілими та досяжними в процесі моделювання та програмування?

На всі ці питання намагаєтьяс знайти відповідь практична дисципліна «Програмотехніка», яка включає в себе такі етапи розробки програмного забезпечення :

1. Методи управління під час розробки ПЗ;
2. Оцінки ціни або вартості розробки:
3. Розклад проведення робіт:
4. Моніторинг процесу розробки ПЗ:
5. Аналіз різних методів проектування системи:
6. Технології підвищення надійності ПЗ:
7. Методи підготовки технічної і користувацької документації:
8. Аналіз процедур контролю якості:
9. Різні методи зменшення витрат на підтримку, усунення неполадок, модифікацію ПЗ:

1. Технології командної роботи та різні аспекти, які впливають на

командну роботу:

Останнім часом на розробку програмних продуктів спостерігають кризу пов’язану з розробкою ПЗ. До основних причин кризи відносять:

1.Постійно зростаючу складність ПЗ та складність процесу їх розробки

2. Невідповідність, а іноді суперечність між очікуваною ефективністю систем та її надійністю (причиною цього є складність розроблюваних систем та не ідеальність методів її створення):

3. Вартісна підтримка систем:

4. Рідкісне повторне використання вже існуючих проектів і компонентів ПЗ в нових розробках (це пов’язане із специфікою розробки систем):

5. Тривалий і дорогий цикл розробки ПЗ (тому існують великі шанси, що проект буде незавершений):

6. Практична необхідність робити зміни та оновлювати систему

7. Велика кількість та специфіка мов програмування, які використовуються в процесі розробки:

8. Велика залежність результатів роботи від зміни методів, пристроїв, та інших умов, які безпосередньо впливають на розробку:

9. Проблеми, які пов’язані з інтеграцією готових компонентів розроблених різними командами:

На практиці всі ці явища намагаються обмежити завдяки: різних методів та інструментів, що полегшують роботу зі складними системами; використання методів аналізу нових проблем, які можуть виникнути в процесі розробки процедури розробки ПЗ зробити систематизованими, спланованими та керованими; переконання як виробників так і покупців, що в розроблення сучасної, надійної, ефективної системи вимагає відповідних капіталовкладень.

Питання для самоконтролю.

1. Що уявляє собою проект «Software Engineering 2004» ?
2. Які три основні етапи були виконані при розробці документа «Software Engineering 2004» ?
3. Які відмінності між програмною інженерією і традиційною інженерією ?
4. Сформулюйте набір характеристик, які зробили істотний вплив на розвиток програмної інженерії,
5. Яка роль інженерне проектування в програмній інженерії ?
6. Чому бажано, щоб бакалаври програмної інженерії бути знайомі хоч би з однією з прикладних предметних областей, ?
7. Які питання виникають під час розробки програмного забезпечення ?

8. Які причини складності програмного забезпечення ?
9. Чрму розробка програмного забезпечення не у всіз випадках є успішною ?
10. Які етапи розробки програмного забезпечення включає в себе дисципліна «Програмотехніка» ?
11. Перелічите основні причини кризи при розробці програмного забезпечення ?

Перелік викоистаних джерел.

1. Бабенко Л.П., Лавріщева К.М. Основи програмної інженерії: Навч. Посіб.-К. : т-во ,,Знання", КОО, 2001.- 269 с.
2. Ковалюк Т.В. Основи програмування. – К.:Видавнича група BHV, 2005.–384 с.
3. Ходаков В.Є., Пилипинко Н.В., Соколова Н.А. Вступ до комп'ютерних наук: навчальний посібник/ За ред. Ходакова В.Є..-К: Центр навчальної літератури.- 2005.-496 с.
4. Иан Соммервил. Инженерия программного обеспечения. 6-е издание. – М. – Спб. – Киев, 2002. – 623 с.
5. Брукшир Дж. Г. Введение в компьютерные науки.–Шестое издание.–Изд. дом "Вильямс", М. "С.-Пб." Киев, 2001.–685с.
6. Технологии разработки программного обеспечения: Учебник/ С. Орлов. — СПб.: Питер, 2002. — 464 с.: ил.
7. Вигерс Карл. Разработка требований к программному обеспечению/Пер, с англ. — М.: Издательсш-торговый дом "Русская Редакция", 2004. —576с.
8. Основи современых компьютерных технологий.:Учебное пособие/Под ред.проф.Хомоненко А.Д..- СПб.:КОРОНА принт,1999.-448с.
9. Дибкова Л.М.Інформатика та комп'ютерна техніка; Посібник для студентів вищих навчальних закладів.-К.:Видавничий центр" Акеадемія",2002.-320с.
10. Інформатика: Комп'ютерна техніка.Комп'ютерні технології, Посібник/Під ред.О.І.Пушкаря К.:Видавничий центр"Акеадемія",2001.-696с.
11. Юрчишин В.М. Інформатика, програмування і ЕОМ. -К.: НМК ВО, 1992.-223с.
12. Юрчишина В.М., Клим Б.В., Кропивницька В.Б. Організація баз даних: Навч. посіб. ІВФНТУНГ.-224 с. (електронний варіант).
13 Лавріщева К.М. Програмна інженерія. –К.-2008.-319 с.

14. Основи алгоритмізації та прогамування: середовище VBA: Навчальний посібник / Заг.ред.к.т.н, доц..Р.Б.Чеповська – Чернівці: Книги – ХНІ, 2006.-430 с.”
15. Андерсен Т. Visual Basic : Пер. англ..- М. ЗАО. Издательство “БИНЩМ “ , 1998,- 224 с.
16. Кучеренко В. Хитрости, трюки и секрети програмирования на Visual Basic (5-6 версии). Ссерии книг “Кратко, доступно просто ” - М.: Познавательная книга плюс,2000.-160 с.
17. Литвиненко Т.В. Visual Basic 6.0 Учебное псобие для вузов.-М.-Горячая линия – Телеком, 2001.-140с.
18 Браун С. Visual Basic 6: ученый курс СПб: ЗАО Издательство “Питер” – 1990 -576 с.