Вкладені цикли

Мета.

Навчальна. повторити з учнями різновиди циклів, та вивчити вкладені цикли, формувати вміння та навички створювати і реалізовувати програми з вкладеними циклами мовою Паскаль.

Розвиваюча. Розвивати логічне мислення, самостійність, вміння застосовувати набуті знання до практичних завдань.

Виховна. Виховувати наполегливість, естетичність у оформленні, грамотно висловлювати свої думки, вміння раціонально використовувати час.

Тип уроку. Засвоєння нових знань, вмінь і навичок.

План

Перевірка домашнього завдання.
Актуалізація опорних знань.
Вкладені цикли.
Розв’язування задач.
Підсумки уроку.
Домашнє завдання.

Пам’ятка для учня!

- Пригадайте правила техніки безпеки при роботі з ПК.
- Через кожні 15 хв. виконуйте вправи для очей та для зняття м’язової втоми.

Хід уроку

1. Перевірка домашнього завдання.

1. Наявність.
2. Питання.

---

2. Актуалізація опорних знань.

У Паскалі передбачено **три різновиди операторів циклу:** цикл із передумовою, цикл з післяумовою, цикл із лічильником (із покроковою зміною аргументу). Також реалізована робота із вкладеними циклами.

I. Питання.

Поясніть синтаксис і правила виконання оператора циклу з передумовою.
Поясніть синтаксис і правила виконання оператора циклу з післяумовою.
Поясніть структуру і правила виконання оператора циклу з параметром.

II. Вказівка повторення з передумовою (Цикл— while)

Вказівка повторення з передумовою — while призначена для організації багатократного виконання групи вказівок (тіло циклу) до тих пір, поки залишається істинною умова виконання циклу.

Значення службового слова while — поки

Синтаксис оператора while:

while <умова> do <оператор>;

де <умова> - вираз логічного типу;

<оператор> - простий або складений оператор Паскаля.

Вказівка повторення з післяумовою (Цикл-repeat-until)

Вказівка повторення з післяумовою призначена для організації багатократного виконання групи вказівок (тіло циклу) до тих пір, поки умова виконання циклу не стане істинною.

Синтаксис оператора repeat:

repeat <оператори> until <умова>;

де <умова> - вираз логічного типу;

<оператор> - послідовність операторів Паскаля, яка складає тіло циклу.

Вказівка повторення з параметром виконується таким чином:

1. Вказівка For — to — do.

Наприклад.

```
for i:=K to N do
        begin
       <вказівка1>;
       <вказівка2>;
       …..............
        <вказівка N>;
end;
```

Параметру циклу **i** присвоюється початкове значення К. Він порівнюється з кінцевим значенням N. Якщо К<=N, то виконується тіло вказівки повторення. Значення К автоматично збільшується на 1 (тобто стає наступним елементом) і знову порівнюється зі значенням N. Якщо під час перевірки отримаємо, що К>N, то виконання вказівки повторення припиняється і виконується наступна після неї вказівка програми. Якщо під час першого порівняння К і N виявиться що К>N, то тіло вказівки не виконується жодного разу.

2. Вказівка For — downto — do.

Наприклад.

```
for i:=К downto to N do
        begin
```

```
        <вказівка1>;
        <вказівка2>;
        ……..
        <вказівка N>;
    end;
```

Параметру циклу і присвоюється початкове значення К. Він порівнюється з кінцевим значенням N. Якщо К>=N, то виконується тіло вказівки повторення. Значення К автоматично зменшується на 1 (тобто стає попереднім елементом) і знову порівнюється зі значенням N. Якщо під час перевірки отримаємо, що К<N, то виконання вказівки повторення припиняється і виконується наступна після неї вказівка програми. Якщо під час першого порівнення К і N виявиться, що К<N, то тіло вказівки не виконується жодного разу.

---

3. Вкладені цикли.

У деяких випадках важливо повторити підзадачу кілька разів усередині більш загальної задачі. Один зі способів написання такої програми - включити цикл у набір інструкцій, що повторюються всередині іншого циклу. Така структура, що складається з циклу в циклі, називається **вкладеними циклами**.

Вкладення циклів використовується зокрема при розв'язуванні таких задач:

- ***задачі на перебір варіантів;***
- ***табулювання функцій;***
- ***обробка двовимірних масивів.***

Якщо в програмі використовуються вкладені цикли, то для підвищення наочності програмного коду прийнято кожний наступний рівень вкладання зміщувати відносно попереднього.

Правило вкладення циклів: *внутрішній цикл цілком укладається в тіло зовнішнього циклу.*

Приклад № 1.

Обчислити значення змінної Y=2*K+N при всіх значеннях змінних N=1, 2, 3 і K=2, 4, 6, 8.

Алгоритм обчислення:

1. Для початку нам потрібно задати наші змінні. Як ми бачимо, змінні N і K у нас відомі N=1, 2, 3 і K=2, 4, 6, 8, а змінна Y залежить від них.

2. Якщо перебирати всі значення N і K, ми повинні отримати 12 значень змінної Y.

Скласти програму можна в такий спосіб: для кожного значення N перебрати всі значення K від 2 до 8, тобто N використати як параметр зовнішнього циклу, K - як параметр внутрішнього циклу.

3. Тоді записуємо внутрішній цикл з умовою, поки k<=8 (тобто поки не пройшлися по всіх числах) і обраховуємо значення змінної Y.

4. Виводимо все на екран.

5. Збільшуємо значення k на 2.

6. Завершуємо програму.

Блок-схема алгоритму побудуйте самостійно.

```
1   Program priklad_1;
.   var n, k, y: integer;
.   begin
.   for n:=1 to 3 do begin
5   k:=2;
.   while k<=8 do begin
.   y:=2*k+n;
.   writeln('n=',n,'  k=',k,'   y=',y);
.   k:=k+2;
10  end;
.   end;
12  readln;
.   end.
14
```

Параметр N змінюється з кроком 1, тому зовнішній цикл організований з використанням оператора **For**; параметр К змінюється з кроком 2, тому внутрішній цикл є циклом **While**.

В процесі виконання вкладених циклів змінні набувають таких значень:

N	1	1	1	1	2	2	2	2	3	3	3	3
K	2	4	6	8	2	4	6	8	2	4	6	8
Y	5	9	13	17	6	10	14	18	7	11	15	19

Приклад 2.

Програма друкує на екрані таблицю Піфагора:

*project1.lpr

```
1   Program priklad_1;
.     Var i, j : integer;
.   Begin
.     For I := 1 To 10 Do
5       Begin
.         For j := 1 To 10 Do
.           write (i*j:3);
.           WriteLn;
.       End;
10    readln;
.   End.
.
13
```

Приклад 3.

Програма для обчислення значень функції

$Y=1+x/1!+x^2/2!+x^3/3!+...$ для x = 1..5.

Y обчислюється з точністю ϵ=0.001, тобто обчислення членів послідовності A (A1=x/1!; A2= $x^2/2!$ і т.д.) продовжується доти, поки A>0.001.

*project1.lpr

```
Program priklad_1;
    var i, x : integer;
        A, Y : Real;
    Begin
     For X := 1 To 5 Do
      Begin
       Y := 1;
       A := 1;
       i := 0;
       While A > 0.001 Do
         Begin
          i := i + 1;
          A := A * X / i;
          Y := Y + A
         End;
       WriteLn ('X=', x,    ',        Y=', Y:6:3)
      End;
      readln;
    End.
```

---

4. Розв’язування задач.

Задача1.

Вивести на екран всі прості числа на інтервалі [A; B].

1. Простими є числа (крім 1), які діляться без остачі тільки на 1 і на самого себе.

Розглянемо алгоритм перевірки, чи є число C простим:

- Вводиться змінна Prap, яка відіграє роль прапорця (за її значенням можна визначити, чи відбулась деяка подія). Змінній Prap присвоюється початкове значення False.
- Перебираються всі числа від 2 до (C div 2). Якщо число i є дільником числа C (тобто C ділиться на i без остачі), то прапорцевій змінній Prap присвоюється значення True.
- Після завершення перебору можливих дільників перевіряється значення змінної Prap. Якщо значення Prap в процесі повторень циклу змінилось на True, то це означає, що C має хоча б один дільник і не є простим.

2. ***Запишемо програму для визначення, чи є число С простим.***

Зверніть увагу: задання значення змінної С навмисно пропущене.

```
  var A, B, C, I : Integer;   Prap : Boolean;
Begin
  { задания значення змінної С }
   Prap := False;
   For I := 2 To C div 2 Do
      if C Mod I = 0 Then
         Prap := True;
     If Not Prap   Then
         WriteLn (C, '- просте число')
     Else WriteLn (C,' - складене число');
End.
```

3. За умовою задачі потрібно перебрати всі числа в інтервалі [A; B], тобто С послідовно набуває значень з діапазону A..B. Для кожного С від А до В потрібно виконати алгоритм перевірки, чи є число С простим.

4. Внесемо зміни до програми. Значення А і В вводяться з клавіатури:

```
    Var A, B, C, I : integer;   Prap : Boolean;
Begin
    write ('A, B =>');   ReadLn (A, B);
```

5. Додаємо керуючий рядок циклу For, тілом циклу якого є програмний код перевірки, чи є число С простим: в

```
For C := A то B Do Begin
      Prap := False;
      For I := 2 To C div 2 Do
```

```
                If C Mod I = 0 Then
                  Prap := True;
              If Not Prap    Then
                 WriteLn (C, ' - просте число');
        End;
```

6. Збережіть файл. Виконайте програму для різних значень A і B.

Задачі з вкладеними циклами

Будь-який складний алгоритм завжди складається з декількох простих алгоритмів. Потрібно навчитись їх бачити та комбінувати.

Приклад 4

Знайти 10 перших простих чисел.

В цій задачі використовуються такі алгоритми:

- Пошук декількох чисел (10), що задовольняють деякій умові (в нашій задачі: чи є число простим?);
- Визначення, чи є число простим?

Зрозуміло, що другий алгоритм вкладений у перший.

Змінні:

Вхідних даних немає.

Вихідні:

- *n* – шукані числа (цілого типу)

Проміжні:

- *k* – лічильник шуканих чисел (цілого типу)
- *p* – ознака наявності дільників (логічного типу, *p=false* - немає дільників, *p=true* - є дільники)
- *i* – дільники та параметр циклу (цілого типу)

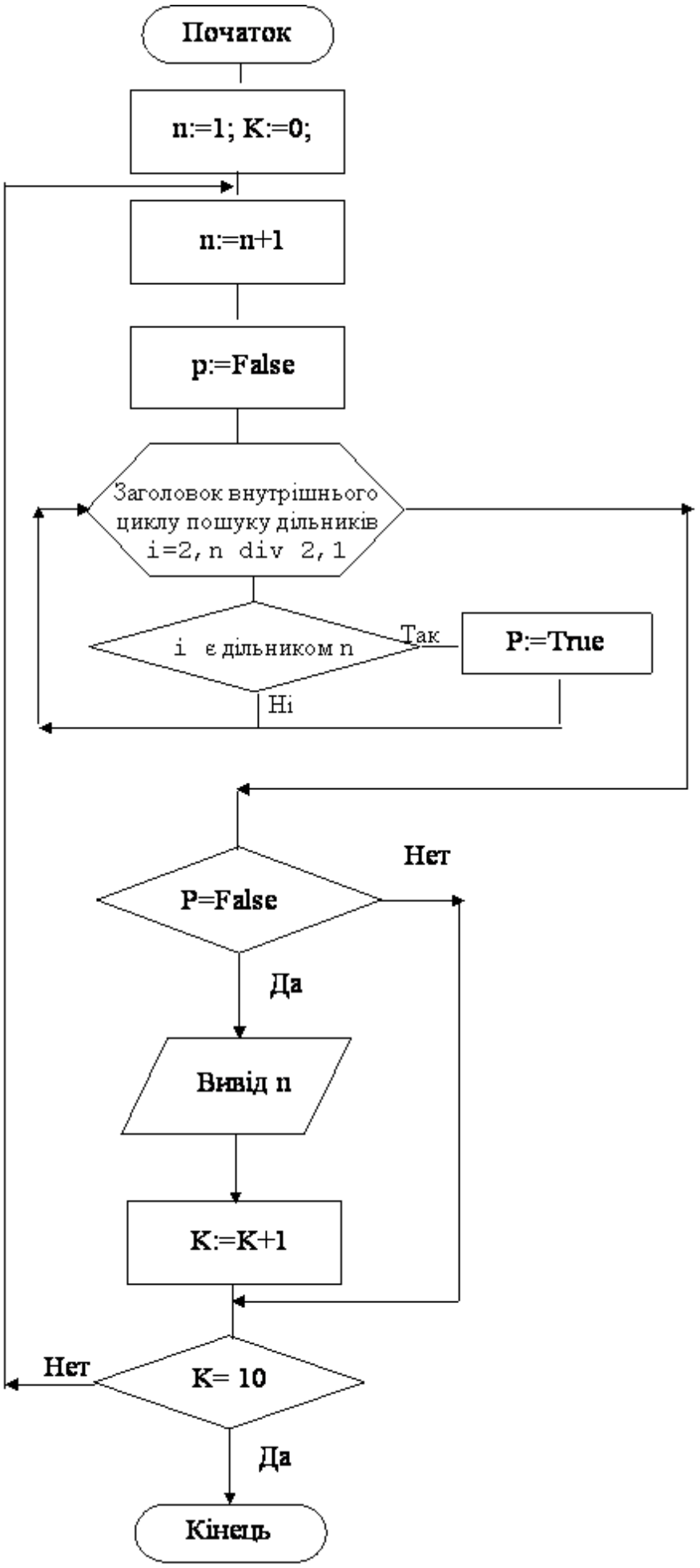
Алгоритм

1. Якщо згадати перший алгоритм, то у ньому:

- до циклу присвоюються початкові значення числу *n* та кількості чисел *k*;
- у тілі циклу *repeat*:
  - збільшується значення числа *n*;
  - число *n* перевіряється на якусь умову: якщо умова вірна, то число друкується та збільшується лічильник таких чисел *k*.
- перевіряється умова завершення циклу: чи надрукована потрібна кількість таких чисел?
  - Якщо так, то цикл завершується і кінець програми.
  - Якщо ні, то перехід на початок тіла циклу.

2. Зрозуміло, що цей алгоритм буде зовнішнім циклом у нашій програмі, і перебір та перевірку чисел потрібно починати з числа 1.
3. Умова, на яку у цьому циклі перевіряється число *n*, – чи є це число простим? Алгоритм, який це визначає, буде вкладений в перший.
4. Згадаємо, що число *n* називається простим, якщо в нього немає дільників в інтервалі [2, n div 2].
5. Тепер згадаємо другий алгоритм, який перевіряє, чи просте число *n*:
   - До циклу присвоюємо початкове значення ознаці *p:=false*. Тобто вважаємо, що дільників у числа *n* немає.
   - У циклі *for i:=2 to n div 2 do* будемо шукати дільники числа *n*. Для кожного будемо перевіряти умову *n mod i=0* і, якщо вона вірна (тобто *i* є дільником *n*), то встановимо значення ознаки *p:=true*.
   - Коли цикл закінчиться, то перевіримо значення ознаки:
     - якщо *p=false*, то дільників немає, число *n* - просте.
     - якщо *p=true*, то є дільники, число - не просте.
6. Другий алгоритм вкладений у перший перед перевіркою числа *n* (виділено жирним).

Блок–схема програми


Початок
n:=1; K:=0;
n:=n+1
p:=False
Заголовок внутрішнього циклу пошуку дільників
i=2,n div 2,1
i є дільником n
Так
P:=True
Ні
P=False
Нет
Да
Вивід n
K:=K+1
K= 10
Нет
Да
Кінець

Програма

```
 Var n,k,i:word; p:boolean;
Begin
 n:=1; k:=0;
 repeat
   n:=n+1;
   p:=false;
   for i:=2 to n div 2 do
    if n mod i=0 then p:=true;
   if not p then
    begin
    write(n,' '); k:=k+1;
    end;
 until k=10;
end.
```

Результат роботи програми

Відповідь
2 3 5 7 11 13 17 19 23 29

Приклад 5

Дано натуральне число *n*. Знайдіть його цифровий корінь (від слова цифра, не плутати з коренем квадратним!)

Визначення

Цифровий корінь числа *n* - це цифра (1, 2, 3,...9), яка обчислюється з цифр числа *n* за таким алгоритмом:

1. Знайдемо у числі *n* суму цифр *s*.
2. Перевіримо цю суму *s*:
    - Якщо вона є цифрою (1, 2, 3,...9), то ця сума є цифровий корінь числа *n*
    - Якщо вона не є цифрою (>9), то візьмемо замість числа *n* отриману суму цифр *s* та повторимо алгоритм з пункту 1.

Результат роботи програми

Ввід	Вивід
3456	9
555	6

Змінні:

Вхідні:

- *n* – натуральне число (цілого типу)

Вихідні:

- *cn* – цифровий корінь числа n (цілого типу)

Проміжні:

- *c* – остання цифра числа n (цілого типу)
- *s* – сума цифр числа n (цілого типу)

Алгоритм

1. В цій задачі використовуються такі алгоритми:
    - знаходження суми цифр числа;
    - заміна числа сумою його цифр, поки ця сума >9.
2. Зрозуміло, що перший алгоритм вкладений у другий.
3. Згадаємо алгоритм знаходження суми цифр числа *n*: обчислюємо останню цифру числа, додаємо її до суми і відкидаємо останню цифру від числа. Все це робимо, поки не переберемо всі цифри числа (у програмі виділено жирним).

4. У другому алгоритмі:
    - Після знаходження суми цифр числа, замінимо число сумою його цифр (*n:=s*).
    - Перевіримо цю суму *s*:
        - Якщо вона є цифрою (*s<=9*), то цикл завершується. Виконується перехід на перший оператор після циклу (пункт 5).
        - Якщо вона не є цифрою (*s>9*), повторимо алгоритм з пункту 3.
5. Значення отриманої суми присвоюється змінній *cn* та виводиться на екран.

Програма

```
 var n:longint;
c,s,cn:byte;
begin
 read(n);
 repeat
   s:=0;
   repeat
    c:= n mod 10;
    s:=s+c;
    n:=n div 10;
   until n=0;
   n:=s;
 until s<=9;
 cn:=s;
 writeln(cn);
end.
```

Приклад 6

Знайдіть цифрові корені всіх простих чисел з інтервалу [100, 200]. Надрукувати число та відповідний цифровий корінь.

Результат роботи програми

Відповідь
101 2 103 4 107 8 109 1 113 5 127 1 131 5 137 2 139 4 149 5 151 7 157 4 163 1 167 5 173 2 179 8 181 1 191 2 193 4 197 8 199 1

Змінні:

Вхідних даних немає.

Вихідні:

- *n* – просте число (цілого типу)
- *cn* – цифровий корінь числа n (цілого типу)

Проміжні:

- *i* – дільники числа n та параметр циклу (цілого типу)
- *p* – ознака простого числа (логічного типу, *p=false* - просте, *p=true* - ні)
- *m* – копія числа *n* (цілого типу)
- *c* – остання цифра числа *n* (цілого типу)
- *s* – сума цифр числа *n* (цілого типу)

Алгоритм

1. В цій задачі використовуються такі алгоритми:
    - перебір всіх чисел з інтервалу [100, 200];
    - для кожного з цих чисел визначення, чи є воно простим;
    - знаходження цифрового кореню кожного простого числа. Цей алгоритм в свою чергу складається з двох алгоритмів (дивись приклад 2).
2. Зрозуміло, що першій алгоритм - це зовнішній цикл for з параметром *n*. Він перебирає числа в інтервалі [100, 200].
3. Другий алгоритм - це пошук дільників *i* для кожного числа *n*. Це теж цикл for з параметром *i* (у програмі виділено жирним). Цей цикл вкладений у перший.
4. Якщо число просте, тобто дільників у числі немає (*p=false*), то виконується третій алгоритм, обчислення цифрового кореню числа *n* (у програмі виділено жирним). Цей алгоритм змінює число *n*, яке є

параметром циклу і не може змінюватись, тому робимо копію цього числа у змінній *m* та застосовуємо цей алгоритм для змінної *m*.

Програма

```
 var n,m,i:longint; c,s,cn:byte;
p:boolean;
begin
 for n:=100 to 200 do
 begin
   p:=false;
   for i:=2 to n div 2 do
    if n mod i=0 then p:=true;
   if p=false then
    begin
         m:=n;
         repeat
              s:=0;
              repeat
              c:= m mod 10;
              s:=s+c;
              m:=m div 10;
              until m=0;
              m:=s;
         until s<=9;
         cn:=s;
         write(n,' ',cn,' ');
    end;
 end;
end.
```

Приклад 7

Дано натуральне число *n*. З'ясуйте, чи є воно симетричним (3434, 345345).

Визначення

Натуральне число *n* називається *симетричним*, якщо існує таке натуральне число *m*, що виконується рівність n div 10^m=n mod 10^m. Число *m* дорівнює кількості цифр у числі *n*, поділеній на 2.

Результат роботи програми

Ввід	Вивід
3453 45	yes
345	no

Змінні:

Вхідні:

- *n* – натуральне число (цілого типу)

Вихідні:

- *m* – степінь числа 10, половина від *k* кількості цифр числа (цілого типу)

Проміжні:

- *i* – параметр циклу (цілого типу)
- *k* – кількість цифр числа n (цілого типу)
- *x* – копія числа n (цілого типу)
- *d* – 10^m (цілого типу).

Алгоритм

1. У цій задачі використовуються такі алгоритми:
    - знаходження кількості цифр числа;
    - піднесення числа до натурального степеня.
2. Зрозуміло, що алгоритми використовуються послідовно.
3. Алгоритм програми докладніше:
4. Вводимо число *n*.

5. Запам’ятовуємо це число у змінну *x*, бо алгоритм знаходження кількості цифр у числі змінює число *n*.
6. Встановимо початкове значення *k* лічильника цифр у числі.
7. Використаємо алгоритм знаходження кількості цифр числа *x*: відкидаємо останню цифру від числа, накопичуючи лічильник *k*. Все це робимо поки не переберемо всі цифри числа (у програмі виділено жирним).
8. Знайдемо *m:=k div 2*.
9. Знайдемо $d=10^m$ (у програмі виділено жирним). Для цього:
    - встановимо початкове значення добутку *d:=1*;
    - у циклі for *m* раз виконаємо оператор *d:=d*10*.
10. Порівняємо частку та остачу від ділення *n* на *d*:
    - Якщо співпадають, то виводимо „число симетричне”;
    - Якщо не співпадають, то виводимо „число не симетричне”.

Програма

```
 var n,x,d,i,k,m:longint;
begin
 read(n); x:=n;
 k:=0;
 repeat
    k:=k+1;
    x:=x div 10;
 until x=0;
 m:=k div 2;
 d:=1;
 for i:=1 to m do d:=d*10;
   if n div d=n mod d then
        writeln('симетричне ')
   else writeln('несиметричне
');
end.
```

Приклад 8

Надрукувати перші 5 шестизначних симетричних чисел.

В цій задачі використовуються такі алгоритми:

- Пошук декількох чисел (5), що задовольняють деякій умові (в нашій задачі: чи є число симетричним?);
- Визначення, чи є число симетричним?

Зрозуміло, що другий алгоритм вкладений у першій.

Змінні:

Вхідних даних немає.

Вихідні:

- *n* – шукані числа (цілого типу)

Проміжні:

- *t* – лічильник шуканих чисел (цілого типу)
- *x* – копія числа *n* (цілого типу)
- *i* – параметр циклу (цілого типу)
- *k* – кількість цифр числа *n* (цілого типу)
- *m* – степінь числа 10, половина від *k* кількості цифр числа (цілого типу)
- *d* – 10^m (цілого типу).

Алгоритм

1. Якщо згадати перший алгоритм, то у ньому:
    - До циклу задаються початкові значення числу *n* та кількості чисел *t* .
    - В тілі циклу *repeat*:
        - Збільшується значення числа *n*;
        - Число *n* перевіряється на якусь умову: якщо умова вірна, то число друкується та збільшується лічильник таких чисел *t*.
    - Перевіряється умова завершення циклу: чи надрукована потрібна кількість таких чисел?
        - Якщо так, то цикл завершується і кінець програми;
        - Якщо ні, то перехід на початок тіла циклу.
2. Зрозуміло, що цей алгоритм буде зовнішнім циклом у нашій програмі і перебір та перевірку чисел потрібно починати з числа 100000, бо нам потрібні шестизначні числа.
3. Умова, на яку у цьому циклі перевіряється число *n*, це – чи є це число симетричним? Алгоритм, який це визначає буде вкладений в перший.
4. Тепер згадаємо другий алгоритм, який перевіряє, чи є число n симетричним:
    - Обчислюємо *k* кількість цифр у числі;
    - Обчислюємо *m* половину кількості цифр у числі;
    - Обчислюємо d =10^m.
    - Перевіряємо рівність *n div d=n mod d*:
        - Якщо рівність вірна, то число *n* - симетричне, ми його друкуємо та збільшуємо лічильник таких чисел.
        - Якщо рівність не вірна, то число *n* - несиметричне.
5. Другий алгоритм вкладений у перший перед перевіркою числа *n* (виділено жирним).

Програма

```
 var n,t,k,x,m,d,i:longint;
begin
 n:=100000;t:=0;
 repeat
   n:=n+1;
   k:=0; x:=n;
   repeat
    k:=k+1;
    x:=x div 10;
   until x=0;
   m:=k div 2;
   d:=1;
   for i:=1 to m do d:=d*10;
    if n div d=n mod d then
    begin
        write(n,' ');
t:=t+1;
    end;
 until t=5;
end.
```

Результат роботи програми

Відповідь
100100 101101 102102 103103 104104

Варіанти задач

1. Надрукувати всі числа з інтервалу [100, 200], цифровий корінь яких кратний 3 (3, 6, 9).

2. Дано число *a*. Зайдіть *a* перших простих чисел.
3. Дано число *a*. Знайти просте число, що більше *a*.
4. Дано число *a*. Знайдіть 5 простих чисел, більших *a*.
5. Дано число *a*. Знайти найближче до нього просте число.
6. Знайдіть всі числа з інтервалу (100, 200), цифровий корінь яких є простим числом (1, 2, 3, 5, 7).
7. Знайдіть всі числа з інтервалу (100, 200), які кратні своєму цифровому кореню.
8. Знайдіть цифрові корені чисел Фібоначчі, що належать інтервалу (100, 1000).
9. Обчисліть та надрукувати цифрові корені досконалих чисел, що належать діапазону (1; 10000). Натуральне число називається досконалим, якщо воно дорівнює сумі своїх дільників, із урахуванням 1 і за винятком самого числа. Наприклад, число 6 - досконале (6=1+2+3).
10. Знайдіть всі симетричні числа з інтервалу [10000, 1000000].
11. Знайдіть всі симетричні паліндроми з інтервалу [1000000, 1000000000]. Пояснення: *паліндром* - це число, яке читається однаково справа наліво та зліва направо, тобто саме число дорівнює перевернутому числу.
12. Надрукувати з чисел Фібоначчі в інтервалі від 1 до 100, тільки прості числа, а також їх порядкові номери в ряду Фібоначчі.
13. Для всіх чисел, що належать діапазону [100; 120] обчислити та надрукувати ті дільники, що є членами послідовності Фібоначчі.
14. Надрукувати всі чотирьохзначні симетричні числа та знайдіти їх кількість.
15. Знайдіть цифрові корені всіх симетричних чисел, що належать інтервалу (10000, 100000).
16. Надрукувати перші 10 п’ятизначних паліндромів, що є простими числами. Пояснення: *паліндром* - це число, яке читається однаково справа наліво та зліва направо, тобто саме число дорівнює перевернутому числу.

- **4. Підсумки уроку.**
  - Чи може оператор, повторюваний у циклі, сам бути циклом?
  - Що таке вкладені цикли?
  - У чому полягає правило вкладення циклів ?

- Наведіть приклади задач, при розв'язуванні яких виникає необхідність використання вкладених циклів.
- **Проаналізуйте циклічну конструкцію:**

```
For i:=1 To 2 Do
        For j:=1 To 3 Do
                For k:=-1 To 3 Do
                        WriteLn (i, j, к);
```

а) назвіть тіло циклу по i, j, к;

б) скільки разів буде виконаний цикл по i, j, к?

в) що надрукує програма ?

- **Проаналізуйте циклічну програму:**

```
Program P5;
 var i, y, k: Integer;
Begin
  For i:=1 To 3 Do Begin
      j:=2;
      While j<=8 Do Begin
          k:=i+j;
            WriteLn (i: 4, j: 4, k: 4);
            j:=j+2;
      End
   End
End.
```

а) Назвіть тіло циклу по i, j.

б) Скільки разів буде виконаний зовнішній цикл? Внутрішній?

в) Заповніть таблицю зміни значень змінних i, j, k.

- **Проаналізуйте циклічну програму:**

```
Program P6;
```

```
var i,j:integer;
Begin
  i:=0;
  While I<10 Do Begin
    {*} J:=0;
    While J<0 Do Begin
      {**}  Write(i*10 + j: 4);
      j:=j+1;
    End {while J};
    WriteLn;
  End{while i};
End.
```

а) скільки разів виконається рядок {*}? рядок {**}?

б) що виведе на екран програма ?

в) перепишіть програму за допомогою FOR.

- **Запишіть циклічну конструкцію, в результаті виконання якої на екрані буде надруковано:**

0 0

0 1

1 0

1 1

---

5. Домашнє завдання.

1. Вивчити конспект.

2. Знайти і вивести на екран всі тризначні числа, які діляться на кожну із своїх цифр. Примітка: памятайте, що на нуль ділити не можна!