Муниципальное бюджетное общеобразовательное учреждение
Павловская средняя общеобразовательная школа
с углублённым изучением отдельных предметов
Павловского муниципального района
Воронежской области

Экзаменационные билеты

для 10 класса

1 (базовый уровень, время – 3 мин)

Тема: Анализ таблиц истинности логических выражений.

Что проверяется:

Умение строить таблицы истинности и логические схемы.

1.5.1. Высказывания, логические операции, кванторы, истинность высказывания

1.1.6. Умение строить модели объектов, систем и процессов в виде таблицы истинности для логического высказывания

Про обозначения

К сожалению, обозначения логических операций И, ИЛИ и НЕ, принятые в «серьезной» математической логике ($\wedge, \vee, \neg$), неудобны, интуитивно непонятны и никак не проявляют аналогии с обычной алгеброй. Автор, к своему стыду, до сих пор иногда путает $\wedge$ и $\vee$. Поэтому на его уроках операция «НЕ» обозначается чертой сверху, «И» – знаком умножения (поскольку это все же логическое умножение), а «ИЛИ» – знаком «+» (логическое сложение).

В разных учебниках используют разные обозначения. К счастью, в начале задания ЕГЭ приводится расшифровка закорючек ($\wedge, \vee, \neg$), что еще раз подчеркивает проблему.

Что нужно знать:

- условные обозначения логических операций

¬ A, $\overline{A}$	не A (отрицание, инверсия)
A ∧ B, $A \cdot B$	A и B (логическое умножение, конъюнкция)
A ∨ B, $A + B$	A или B (логическое сложение, дизъюнкция)
A → B	импликация (следование)
A ≡ B	эквивалентность (равносильность)

- операцию «импликация» можно выразить через «ИЛИ» и «НЕ»:

  A → B = ¬ A ∨ B или в других обозначениях **A → B =** $\overline{A} + B$

- иногда для упрощения выражений полезны формулы де Моргана:

¬ (A ∧ B) = ¬ A ∨ ¬ B	$\overline{A \cdot B} = \overline{A} + \overline{B}$
¬ (A ∨ B) = ¬ A ∧ ¬ B	$\overline{A + B} = \overline{A} \cdot \overline{B}$

- если в выражении нет скобок, сначала выполняются все операции «НЕ», затем – «И», затем – «ИЛИ», «импликация», и самая последняя – «эквивалентность»
- таблица истинности выражения определяет его значения при всех возможных комбинациях исходных данных
- если известна только часть таблицы истинности, соответствующее логическое выражение однозначно определить нельзя, поскольку частичной таблице могут соответствовать несколько *разных* логических выражений (не совпадающих для других вариантов входных данных);
- количество *разных* логических выражений, удовлетворяющих неполной таблице истинности, равно 2^k, где k – число *отсутствующих* строк; например, полная таблица истинности выражения с тремя переменными содержит 2^3=8 строчек, если заданы только 6 из них, то можно найти $2^{8-6}=2^2$=4 *разных* логических выражения, удовлетворяющие этим 6 строчкам (но отличающиеся в двух оставшихся)

- логическая сумма A + B + C + … равна 0 (выражение ложно) тогда и только тогда, когда все слагаемые одновременно равны нулю, а в остальных случаях равна 1 (выражение истинно)
- логическое произведение A · B · C · … равно 1 (выражение истинно) тогда и только тогда, когда все сомножители одновременно равны единице, а в остальных случаях равно 0 (выражение ложно)
- логическое следование (импликация) A→B равна 0 тогда и только тогда, когда A (посылка) истинна, а B (следствие) ложно
- эквивалентность A≡B равна 1 тогда и только тогда, когда оба значения одновременно равны 0 или одновременно равны 1

Пример задания:

Р-22 (демо-2021). Логическая функция F задаётся выражением

$$(x \vee y) \wedge \neg(y \equiv z) \wedge \neg w.$$

На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z, w.

?	?	?	?	F
1		1		1
0	1		0	1
	1	1	0	1

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Вопросы:

1. Таблицы истинности
2. Формулы преобразования
3. Правила построения таблиц истинности

2 (базовый уровень, время – 4 мин)

Тема: Анализ программы с циклом.

Что проверяется:

Знание основных конструкций языка программирования, понятия переменной, оператора присваивания.

1.7.2. Основные конструкции языка программирования. Система программирования.

1.1.4. Читать и отлаживать программы на языке программирования.

Что нужно знать:

- основные конструкции языка программирования:
  - объявление переменных
  - оператор присваивания
  - оператор вывода
  - циклы
- уметь выполнять ручную прокрутку программы
- уметь выделять переменную цикла, от изменения которой зависит количество шагов цикла

- уметь определять количество шагов цикла
- уметь определять переменную, которая выводится на экран
- формулу для вычисления n -ого элемента арифметической прогрессии:

$$a_n = a_1 + d(n-1)$$

- формулу для вычисления суммы первых n членов арифметической прогрессии:

$$S_n = \sum_{i=1}^{n} a_i = a_1 + a_2 + \square + a_n = \frac{a_1 + a_n}{2} \cdot n = \frac{2a_1 + d(n-1)}{2} \cdot n$$

где a_i – i -ый элемент последовательности, d – шаг (разность) последовательности

Пример задания:

P-04. *Определите, при каком наименьшем введённом значении переменной* **s** *программа выведет число 64.*

```
var s, n: integer;
begin
  readln (s);
  n := 1;
  while s < 51 do begin
    s := s + 5;
    n := n * 2
  end;
  writeln(n)
end.
```

Вопросы:

1. Циклы
2. Тестирование программы
3. Решение программы с циклами через систему

3 (повышенный уровень, время – 3 мин)

Тема: Позиционные системы счисления.

Что проверяется:

Знание позиционных систем счисления.

1.4.1. Позиционные системы счисления.

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов(?).

Что нужно знать:

- принципы кодирования чисел в позиционных системах счисления
- чтобы перевести число, скажем, 12345_N, из системы счисления с основанием N в десятичную систему, нужно умножить значение каждой цифры на N в степени, равной ее разряду:

$\mathbf{N^0 = 1}$

4 3 2 1 0 ← разряды

$\mathbf{1\ 2\ 3\ 4\ 5_N = 1 \cdot N^4 + 2 \cdot N^3 + 3 \cdot N^2 + 4 \cdot N^1 + 5 \cdot N^0}$

- последняя цифра записи числа в системе счисления с основанием N – это остаток от деления этого числа на N
- две последние цифры – это остаток от деления на N^2, и т.д.
- число 10^N записывается как единица и N нулей: $10^N = 10\underbrace{\ldots}_{N}0$
- число 10^N-1 записывается как N девяток: $10^N - 1 = \underbrace{9\ldots}_{N}9$
- число $10^N-10^M = 10^M \cdot (10^{N-M} - 1)$ записывается как N-M девяток, за которыми стоят M нулей: $10^N - 10^M = \underbrace{9\ldots}_{N-M}\underbrace{90\ldots}_{M}0$
- число 2^N в двоичной системе записывается как единица и N нулей: $2^N = 10\underbrace{\ldots}_{N}0_2$
- число 2^N-1 в двоичной системе записывается как N единиц: $2^N - 1 = \underbrace{1\ldots}_{N}1_2$
- число $2^N–2^K$ при $K < N$ в двоичной системе записывается как $N–K$ единиц и K нулей: $2^N - 2^K = \underbrace{1\ldots}_{N-K}\underbrace{10\ldots}_{K}0_2$
- поскольку $2^N + 2^N = 2 \cdot 2^N = 2^{N+1}$, получаем $2^N = 2^{N+1} - 2^N$, откуда следует, что $-2^N = -2^{N+1} + 2^N$
- число 3^N записывается в троичной системе как единица и N нулей: $3^N = 10\underbrace{\ldots}_{N}0_3$
- число 3^N-1 записывается в троичной системе как N двоек: $3^N - 1 = \underbrace{2\ldots}_{N}2_3$
- число $3^N – 3^M = 3^M \cdot (3^{N-M} – 1)$ записывается в троичной системе как N-M двоек, за которыми стоят M нулей: $3^N - 3^M = \underbrace{2\ldots}_{N-M}\underbrace{20\ldots}_{M}0_3$
- можно сделать аналогичные выводы для любой системы счисления с основанием a:
  - число a^N в системе счисления с основанием a записывается как единица и N нулей: $a^N = 10\underbrace{\ldots}_{N}0_a$
  - число a^N-1 в системе счисления с основанием a записывается как N старших цифр этой системы счисления, то есть, цифр $(a-1)$: $a^N - 1 = \underbrace{(a-1)(a-1)\ldots}_{N}(a-1)_a$

- число $a^N - a^M = a^M \cdot (a^{N-M} - 1)$ записывается в системе счисления с основанием a как N-M старших цифр этой системы счисления, за которыми стоят M нулей:

$$a^N - a^M = \underbrace{(a-1)\ldots(a-1)}_{N-M}\underbrace{0\ldots0}_{M}{}_a$$

Пример задания:

Р-25. (**демо-2021**) Значение арифметического выражения: $49^7 + 7^{21} - 7$ – записали в системе
счисления с основанием 7. Сколько цифр 6 содержится в этой записи?

Вопросы:

1. Системы счисления. Сравнение чисел в различных системах счисления
2. Перевод из одной системы счисления в другую
3. Арифметические операции в различных системах счисления

4 (повышенный уровень, время – 14 мин)

Тема: Перебор последовательности целых чисел. Проверка делимости

Что проверяется:

Умение создавать собственные программы (20–40 строк) для обработки целочисленной информации.

1.7.2. Основные конструкции языка программирования. Система программирования.

1.1.5. Умение создавать программы на языке программирования по их описанию.

Что нужно знать:

- в известных задачах этого типа (не олимпиадных) нет ограничения на время выполнения, по крайней мере, оно несущественно для отрезков, заданных для перебора; поэтому можно использовать простой перебор без оптимизации;
- задачи этого типа предлагается решать с помощью электронных таблиц или собственной программы; как правило, написать правильную программу значительно проще
- пусть необходимо перебрать все целые числа на отрезке [a; b] и подсчитать, для скольких из них выполняется некоторое условие; общая структура цикла перебора записывается так (Python):

```
count = 0
for n in range(a, b+1):
  if условие выполнено:
    count += 1
print( count )
```

Pascal:

```
count := 0;
for n:=a to b do
  if условие выполнено then
    count := count + 1;
writeln(count);
```

C++:

```
int count = 0;
for(int n = a; n <= b; n++)
```

```
  if( условие выполнено )
    count += 1;
std::cout << count;
```

- проверку условия удобно оформить в виде функции, возвращающей логическое значение (True/False), но можно этого и не делать
- проверить делимость числа n на число d можно с помощью операции взятия остатка от деления n на d: если остаток равен 0, число n делится на d нацело
- проверка на языке Python выглядит так:

```
if n % d == 0:
      print("Делится")
else: print("Не делится")
```

- тоже самое на Pascal

```
if n mod d = 0 then
     writeln('Делится')
else writeln('Не делится')
```

- то же самое на C++

```
if( n % d == 0 )
     std::cout << "Делится";
else std::cout << "Не делится";
```

Пример задания:

> **P-01 (демо-2021).** *Рассматривается множество целых чисел, принадлежащих числовому отрезку [1016; 7937], которые делятся на 3 и не делятся на 7, 17, 19, 27. Найдите количество таких чисел и максимальное из них. В ответе запишите два целых числа: сначала количество, затем максимальное число. Для выполнения этого задания можно написать программу или воспользоваться редактором электронных таблиц.*

Вопросы:

1. Условия кратности
2. Подсчет количества делителей
3. Проверка числа на простоту

5 (высокий уровень, время – 18 минут)

Тема: Обработка символьных строк

Что проверяется:

Умение создавать собственные программы (10–20 строк) для обработки символьной информации.

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности.

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Что нужно знать:

- сначала нужно прочитать строку из файла; эта задача в разных языках программирования решается несколько по-разному
- в языке Python удобнее всего использовать такую конструкцию:

```
with open("k7.txt", "r") as F:
```

```
    s = F.readline()
```

конструкция **with-as** – это *контекстный менеджер*, в данном случае он открывает указанный файл в режиме чтения (второй аргумент «**r**» при вызове функции **open**), записывает ссылку на него в файловую переменную **F**, выполняет тело блока (читает первую строку файла в переменную **s**) и закрывает (освобождает) файл

- в языке PascalABC.NET можно выполнить перенаправление потока ввода:

```
assign( input, 'k7.txt' );
readln(s);
```

программа будет «думать», что читает данные, введённые с клавиатуры (с консоли), а на самом деле эти данные будут прочитаны из файла **k7.txt**

- в языке FreePascal также можно выполнить перенаправление потока ввода, но нужно дополнительно открывать входной поток:

```
assign( input, 'k7.txt' );
reset( input );            { для FreePascal!!! }
readln(s);
```

- при работе в среде FreePascal нужно убедиться, что в параметрах компилятора включена поддержка **длинных символьных строк**; на всякий случай стоит добавить в первой строке программы директиву

```
{$H+}
```

- Среда PascalABC НЕ ПОДДЕРЖИВАЕТ работу с **длинными символьными строками**, поэтому для решения задачи использовать версию PascalABC.NET, которую можно бесплатно скачать с сайта автора www.pascalabc.net.

- в языке C++ используем потоки:

```
#include <fstream>
#include <string>
using namespace std;
int main()
{
  ifstream F("k7.txt");
  string s;
  getline( F, s );
  ...
}
```

Самая длинная цепочка символов «С»

- пусть требуется найти самую длинную цепочку символов С (или каких-то других, в соответствии с заданием) в символьной строке **s**;
- можно использовать такой алгоритм:

```
while не конец строки:
  найти очередную букву С
  длина := длина текущей цепочки букв С
  if длина > максимальной длины:
    максимальная длина := длина
```

однако этот алгоритм содержит вложенный цикл и при составлении программы легко запутаться и не учесть какой-то особый случай (например, когда строка состоит только из букв С)

- лучше применить однопроходный алгоритм без вложенного цикла

```
for c in s:
```

```
    обработать символ c
```

- будем использовать переменные

  cLen – длина текущей цепочки букв C

  maxLen – максимальная длина цепочки букв C на данный момент

- рассмотрим очередной символ строки; если это буква C, увеличиваем **cLen** на 1 и, если нужно запоминаем новую максимальную длину; если это не буква C, просто записываем с **cLen** ноль:

```
maxLen = 0
cLen = 0
for c in s:
  if c == 'C':
    cLen += 1           # ещё одна буква C
    if cLen > maxLen:   # возможно, новая максимальная длина
       maxLen = cLen
  else:
    cLen = 0            # цепочка букв C кончилась
```

- проверим правильность работы алгоритма в особых случаях:

1) если вся строка состоит из букв C, значение переменной **cLen** постоянно увеличивается и в конце станет равно длине символьной строки; то же значение окажется и в переменной **maxLen**;
2) если в строке нет символов C, переменная **cLen** всегда равна 0, такое же значение будет и в переменной **maxLen**

Самая длинная цепочка любых символов

- теперь поставим задачу найти самую длинную цепочку символов в символьной строке **s**; сложность состоит в том, что мы (в отличие от предыдущей задачи) не знаем, из каких именно символов состоит самая длинная цепочка
- если символов в алфавите немного (скажем, A, B и C), то можно с помощью описанного выше алгоритма найти самые длинные цепочки из букв A, B и C, а затем выбрать из них «длиннейшую»; такая идея может сработать при аккуратной реализации, но плохо обобщается на случай, когда возможных символов много (например, используются все заглавные латинские буквы и цифры)
- поэтому лучше применить однопроходный алгоритм без вложенного цикла
- будем использовать переменные

  curLen – длина текущей цепочки одинаковых символов

  maxLen – максимальная длина цепочки одинаковых символов на данный момент

  c – символ, из которого строится самая длинная подцепочка

- в начальный момент рассмотрим один первый символ (цепочка длины 1 есть всегда!):

```
maxLen = 1
curLen = 1
c = s[0]
```

- будем перебирать в цикле все символы, начиная с **s[1]** (второго по счёту) до конца строки, постоянно «оглядываясь назад», на предыдущий символ

```
for i in range(1,len(s)):
  обработать пару символов s[i-1] и s[i]
```

- если очередной символ **s[i]** такой же, как и предыдущий, цепочка одинаковых символов продолжается, и нужно увеличить значение переменной curLen; если

значение curLen стало больше **maxLen**, обновляем **maxLen** и запоминаем новый базовый символ в переменной **c**:

```
if s[i] == s[i-1]:      # цепочка продолжается
  curLen += 1           # увеличиваем длину
  if curLen > maxLen:   # если цепочка побила рекорд
    maxLen = curLen     # запоминаем её длину...
    c = s[i]            # и образующий символ
else:
  curLen = 1            # началась новая цепочка
```

если очередной символ не совпал с предыдущим, началась новая цепочка, и её длина пока равна 1 (это значение записывается в переменную curLen)

- получается такой цикл обработки строки:

```
maxLen, curLen, c = 1, 1, s[0]
for i in range(1, len(s)):
  if s[i] == s[i-1]:
    curLen += 1
    if curLen > maxLen:
      maxLen = curLen
      c = s[i]
  else:
    curLen = 1
```

- проверим правильность работы алгоритма в особых случаях:
  3) если вся строка состоит из одинаковых символов, значение переменной **curLen** постоянно увеличивается и в конце станет равно длине символьной строки; то же значение окажется и в переменной **maxLen**;
  4) если в строке нет пар одинаковых символов, переменная **curLen** всегда равна 1, такое же значение будет и в переменной **maxLen**

Пример задания:

P-07 (демо-2021). Текстовый файл **24.txt** состоит не более чем из 10^6 символов X, Y и Z. Определите максимальное количество идущих подряд символов, среди которых каждые два соседних различны. Для выполнения этого задания следует написать программу.

Вопросы:

1. Работа с файлами. Чтение и запись.
2. Поиск в файле
3. Работа с числами в файле

6 (повышенный уровень, время – 3 мин)

Тема: Вычисление информационного объема сообщения.

Что проверяется:

Умение подсчитывать информационный объём сообщения.

1.1.3. Дискретное (цифровое) представление текстовой, графической, звуковой информации
и видеоинформации. Единицы измерения количества информации.
1.3.1. Умение оценивать объём памяти, необходимый для хранения информации.

Что нужно знать:

- с помощью K бит можно закодировать $Q = 2^K$ различных вариантов (чисел)
- таблица степеней двойки, она же показывает, сколько вариантов Q можно закодировать с помощью K бит:

K, бит	1	2	3	4	5	6	7	8	9	10
Q, вариантов	2	4	8	16	32	64	128	256	512	1024

- при измерении количества информации принимается, что в одном байте 8 бит, а в одном килобайте (1 Кбайт) – 1024 байта, в мегабайте (1 Мбайт) – 1024 Кбайта[1]
- чтобы найти информационный объем сообщения (текста) I, нужно умножить количество символов (отсчетов) N на число бит на символ (отсчет) K: $I = N \cdot K$
- две строчки текста не могут занимать 100 Кбайт в памяти
- мощность алфавита M – это количество символов в этом алфавите
- если алфавит имеет мощность M, то количество всех возможных «слов» (символьных цепочек) длиной N (без учета смысла) равно $Q = M^N$; для двоичного кодирования (мощность алфавита M – 2 символа) получаем известную формулу: $Q = 2^N$

Пример задания:

Р-10 (демо-2021). При регистрации в компьютерной системе каждому объекту сопоставляется
идентификатор, состоящий из 15 символов и содержащий только символы из 8-символьного набора: A, B, C, D, E, F, G, H. В базе данных для хранения сведений о каждом объекте отведено одинаковое и минимально возможное целое число байт. При этом используют посимвольное кодирование идентификаторов, все символы кодируют одинаковым и минимально возможным количеством бит. Кроме собственно идентификатора, для каждого объекта в системе хранятся дополнительные сведения, для чего отведено 24 байта на один объект. Определите объём памяти (в байтах), необходимый для хранения сведений о 20 объектах. В ответе запишите только целое число – количество байт.

Вопросы:

1. Формула нахождения информационного объема.
2. Единцы измерения информации

[1] Часто килобайт обозначают «Кб», а мегабайт – «Мб», но в демо-тестах разработчики ЕГЭ привели именно такие обозначения.