

## Условия задач заключительного этапа

### Задача 1. Алгоритм

Дана блок-схема алгоритма обработки строки  $STR$  (см. рис. 1), в которой функция  $length$  возвращает количество символов в строке.

Чему была равна переменная  $STR$  перед началом выполнения алгоритма, если при  $N=10$  в результате обработки было получено значение переменной  $STR = 'bdieghtreen tre'?$

#### Решение.

Ниже представлен исходный код программы на языке C, реализующей описанный блок-схемой алгоритм:

```

1 int N = 10;
2 char STR[16];
3 memcpy_s(STR, 16, "a123456789ABCDEF", 16);
4 int i = 1;
5 int m = sizeof(STR) - 1;
6 char C;
7 for(int k = 1 ; k <= N ; k++)
8 {
9     for(int i = 1 ; i <= m - 2 ; i++)
10    {
11        if(k % 2 == 0)
12        {
13            if(i % 2 == 0)
14            {
15                C = STR[i];
16                STR[i] = STR[i + 2];
17                STR[i + 2] = C;
18            }
19        }
20    }
21    else
22    {
23        C = STR[i];
24        STR[i] = STR[i + 1];
25        STR[i + 1] = C;
26    }
27 } }

```

На вход алгоритма подаются следующие данные:

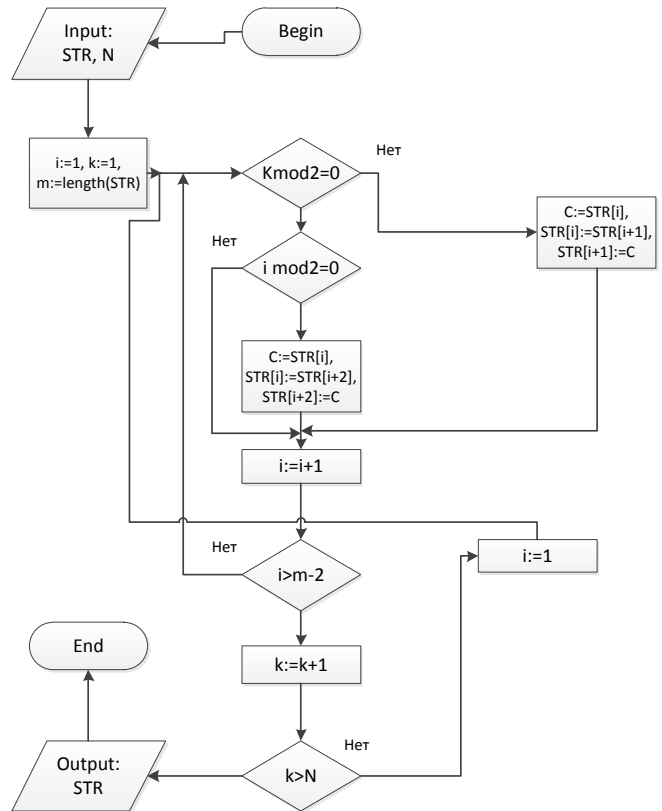


Рис. 1. Блок схема алгоритма обработки строки  $STR$

$N$  — количество шагов алгоритма;  
 $m$  — (количество символов в строке) + 1;  
 $STR$  — строка.

Из анализа данного кода следует, что данный алгоритм реализует перестановку символов в исходной строке  $STR$  следующим образом. Алгоритм состоит из 10 итераций. На каждой итерации алгоритма последовательно просматривается исходная строка  $STR$ . При этом её содержимое изменяется по следующим правилам:

**Случай 1.** Номер итерации алгоритма чётный.

В этом случае каждый символ строки, имеющий чётный индекс  $i$ , последовательно меняется местами с элементом, имеющим индекс  $i+2$  (строки 15, 16, 17 кода).

**Случай 2.** Номер итерации алгоритма нечётный.

В этом случае каждый символ строки последовательно меняется местами с каждым следующим элементом строки (строки 23, 24, 25 кода).

Обозначим через  $B = (b, d, i, e, g, h, t, r, e, e, n, ' ', t, r, e)$  упорядоченный набор символов, соответствующий содержимому переменной  $STR$  после работы алгоритма, описанного блок-схемой, при  $N = 10$ .

Обозначим через  $A = (1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)$  – упорядоченный набор символов, отличных от элементов, входящих в  $B$ .

Пусть на вход алгоритма, заданного блок-схемой, подаётся строка  $STR$ , содержимое которой соответствует упорядоченному набору  $A$ . Тогда после обработки строки  $STR$  с использованием заданного алгоритма при  $N = 10$  её содержимое с одной стороны описывается перестановкой  $C = (A, D, C, 1, E, 3, 2, 5, 4, 7, 6, 9, 8, B, F)$  элементов упорядоченного набора  $A$  и совпадает с упорядоченным набором  $B$  с другой стороны, т. е.  $C = (A, D, C, 1, E, 3, 2, 5, 4, 7, 6, 9, 8, B, F) = (b, d, i, e, g, h, t, r, e, e, n, ' ', t, r, e) = B$ . Сопоставив соответствующие символы этих упорядоченных наборов, отсюда получаем, что в исходном наборе  $1 = e, 2 = t, 3 = h, 4 = e, 5 = r, 6 = n, 7 = e, 8 = t, 9 = ' ', A = b, B = r, C = i, D = d, E = g, F = e$ , следовательно, на вход алгоритма изначально подавалась строка **ethernet bridge**.

**Ответ:** ethernet bridge

## Задача 2. Коллизия

Вводимые пользователями пароли преобразуются функцией, исходный код которой представлен ниже (см. рис. 2). Возвращаемое функцией значение сверяется с эталоном. Известно, что для некоторого логина, пароль «КОНФИДЕНЦИАЛЬНОСТЬ» проходит такую проверку. Существуют ли пароли, которые пройдут описанную проверку и отличные от указанного? Приведите пример такого пароля, состоящего из минимального количества символов.

```

int HASH(char *text)
{
    int k=0, H;
    char temp[6], letter = 'G';
    for(; (k<strlen(text)) && (k<6); ++k) temp[k]=text[k];
    if (k < 6) for(;k<6;k++) temp[k]= letter;
    H = ((temp[0]+temp[1]+temp[2]) &7) *64;
    H += ((temp[0]+temp[2]+temp[4]) &7) *8;
    H += ((temp[1]+temp[3]+temp[5]) &7);
    return H;
}

```

Рис. 2. Исходных код

**Решение.**

Рассмотрим исходный код программы, написанный на языке С. Для удобства пронумеруем строки рассматриваемой функции:

```

int HASH(char *text)
{
    int k=0, H; //1
    char temp[6], letter = 'G'; //2
    for(; (k<strlen(text)) && (k<6); ++k) temp[k]=text[k]; //3
    if (k < 6) for(;k<6;k++) temp[k]= letter; //4
    H = ((temp[0]+temp[1]+temp[2]) &7) *64; //5
    H += ((temp[0]+temp[2]+temp[4]) &7) *8; //6
    H += ((temp[1]+temp[3]+temp[5]) &7); //7
    return H; //8
}

```

В программе использованы следующие переменные:

*text* – входной параметр функции, в которой передается строка, для которой будет вычислено значение функции.

*k* – вспомогательная переменная;

*temp* – символьный массив из 6 элементов, используемый для хранения промежуточных данных при вычислениях;

*H* – целое число типа *int* (для хранения результата используются младшие 9 бит, предназначенное для хранения результирующего значения).

В цикле, реализованном в строках 3, происходит выборка первых 6 символов из входной строки. В строке 4 происходит проверка на длину строки *temp*. Если полученная в результате работы предыдущего цикла строка меньше 6, то она дополняется символами «G». Это преобразование существенно при подборе пароля. Таким образом, после выполнения этой части программы (с учетом того, что по условию задачи преобразуется пароль «КОНФИДЕНЦИАЛЬНОСТЬ») переменная *temp* будет имеет заполнение «КОНФИД».

В строке с номером 5 начинает формироваться выходное значение: ASCII-коды нулевого, первого и второго символов буфера *temp* складываются (по модулю 8), а результата записывается в 6,7,8 биты переменной *H*. В результате выполнения строки 6 аналогично заполняются 3, 4, 5 биты *H*, а в строке 7 – биты с номерами от 0, 1, 2. Таким образом, после выполнения функции, в которой преобразуется пароль «КОНФИДЕНЦИАЛЬНОСТЬ», возвращаемый результат равен  $382_{10} = 101\ 111\ 110_2$ .

Для решения задачи необходимо предложить такой пароль, образ, которого полученный в результате работы функции будет равен 382. Заметим, что из-за процедуры составления строки *temp*, описанной выше, достаточно рассмотреть только пароли длиной менее 7 символов.

Запишем искомый пароль в виде  $a_1 a_2 a_3 a_4 a_5 a_6$ . Учитывая описанный алгоритм построения образа и представление в двоичном виде образ пароля «КОНФИДЕНЦИАЛЬНОСТЬ», запишем систему уравнений, который должен удовлетворять искомый пароль:

$$\begin{cases} a_1 + a_2 + a_3 = 5 \pmod{8} \\ a_1 + a_3 + a_5 = 7 \pmod{8} \\ a_2 + a_4 + a_6 = 6 \pmod{8} \end{cases}$$

Решение будем проводить перебором по размеру пароля.

Предположим, длина искомого пароля 0 или 1, тогда третье уравнение системы принимает вид  $5 = 8 \pmod{8}$ . Таким образом, длина пароля должна быть больше 1.

Пусть длина искомого пароля равна 2, тогда  $temp[6] = "a_1 a_2 GGGG"$  и система принимает вид:

$$\begin{cases} a_1 + a_2 + 71 = 5 \pmod{8} \\ a_1 + 2 * 71 = 7 \pmod{8} \\ a_2 + 2 * 71 = 6 \pmod{8} \end{cases}$$

Данная система несовместна, а значит длина пароля больше 2. Аналогичный результата получается при рассмотрении пароля из трех символов ( $temp[6] = "a_1 a_2 a_3 GGGG"$ ).

Пусть длина искомого пароля равна четырем, тогда  $temp[6] = "a_1 a_2 a_3 a_4 GGGG"$  и система принимает вид:

$$\begin{cases} a_1 + a_2 + a_3 = 5 \pmod{8} \quad (1) \\ a_1 + a_3 + 71 = 7 \pmod{8} \quad (2) \\ a_2 + a_4 + 71 = 6 \pmod{8} \quad (3) \end{cases}$$

Вычитая из уравнения (1) уравнение (2) получаем  $a_2 = 5$ , подставляя это значение в уравнение (3), получаем  $a_4 = 2$ . А из уравнения (2) имеем, например,  $a_1 = 3$ ,  $a_3 = 5$ . Таким образом, один из искомых паролей равен СЕЕВ.

**Ответ.** СЕЕВ (на англ.)

**Задача 3. Язык Орков.**

Саша – любитель фэнтези и регулярный участник ролевых игр по миру Дж. Р. Толкиена, проводимых в Нескучном саду. Любимый Сашин герой – это эльф Леголас, но в этом году ему выпал жребий участвовать в игре на стороне орков. Доспехи у Саши есть, осталось выучить язык. В языке орков пять согласных звуков {h, k, m, r, t} и три гласных {a, o, u}. Каждое слово начинается с согласной буквы, при этом в слове не может быть подряд две гласные или согласные. Длина слов в орочьем языке от трех до шести букв включительно. Сколько слов Саше необходимо выучить.

**Решение.**

Рассмотрим слова из 3-х букв. На первом месте – согласная (5 вариантов), на втором – гласная (3 варианта), на третьем – снова согласная (5 вариантов). Причем согласные могут быть одинаковыми (с повторениями).

Всего таких слов будет  $5 \cdot 3 \cdot 5 = 75$

Аналогично рассмотрим слова из 4-х букв. Всего таких слов будет  $5 \cdot 3 \cdot 5 \cdot 3 = 225$

Аналогично рассмотрим слова из 5-ти букв. Всего таких слов будет  $5 \cdot 3 \cdot 5 \cdot 3 \cdot 5 = 1125$

Аналогично рассмотрим слова из 6-ти букв. Всего таких слов будет  $5 \cdot 3 \cdot 5 \cdot 3 \cdot 5 \cdot 3 = 3375$

Всего слов будет  $75 + 225 + 1125 + 3375 = 4800$ .

**Ответ:** 4800.

**Задача 4. Волшебная гора.**

На волшебной горе расположена система озер и рек. Верхнее озеро образуется из тающего ледника. Из каждого озера вытекает ровно две реки, а впадает одна (за исключением верхнего). Первым на горе возле верхнего озера поселился гном Тим. Каждый новый обитатель горы строил свой дом как можно ближе к Тиму на одном из озер. На каждом озере селился только один гном. Сейчас на горе живут 2014 гномов. Какое количество рек надо проплыть Тиму, чтобы попасть в гости к последнему поселившемуся гному.

**Решение.**

Схему волшебного города можно представить в виде двоичного дерева. В качестве узлов рассматриваются озёра, с построенными на них домами, в качестве входящих и исходящих рёбер – протоки. Корнем дерева (выделенной вершиной) является озеро с домом Тима. Очевидно, что в двоичном дереве число узлов, до которых расстояние от корня равно  $F$  не превосходит  $2^F$ . На расстоянии один может находиться не более двух узлов, на расстоянии 2 – 4-х узлов и т. д. В таблице приведены значения максимального количества домов на расстоянии  $F$ .

Расстояние от корня $F$	0	1	2	3	4	5	6	7	8	9	10	11
Число домов $2^F$	1	2	4	8	16	32	64	128	256	512	1024	2048

Правила застройки города гарантируют появление нового дома на расстоянии  $F+1$  в том и только в том случае, если все возможные места на расстоянии  $F$  уже заполнены. При этом число домов, находящихся не далее, чем на расстоянии  $F$  равно  $1+2+4+\dots+2^F=2^{F+1}-1$ .

В нашем случае могут быть вакантные места для строительства на расстоянии  $F$ , поэтому должно выполняться неравенство  $2^{F+1}-1 \geq 2014$ . Очевидно, что при  $F=9$  выполнено неравенство  $2^{10}-1 < 2014$ , а при  $F=10$   $2^{11}-1 \geq 2014$ . Таким образом, максимальное количество протоков, которое надо проплыть до самого дальнего дома равно 10.

**Ответ:** 10 протоков.

### Задача 5. Лягушка.

Что делает программа (см. Рис.3)?

```

Uses crt;
Const z=10;
hhh=10; y= 2*
hhh+ 1; s= {} hhh- z+
1; Type vec={a -1 ab
x} {} array[s..z] {} of {a ,v
v+1} integer; {} var a,b: {}
vec;v,f: {hh} {xy} {ab}
integer; x: Boolean; {}
Begin clrscr; {a,b,x} {xy}
{ab} Randomize; {}
{hh} {} write( '['= ' );for v:=
s to z do{v x} begin//
f:= random (y);a[v]:= hhh-f+s-1;
write(a[v]:3);End; writeln;b:= a;Repeat{}
x:={x} true;for{a }v:=s to z- 1 do if{}
b[ v] >b[v+1]then Begin{z} f:=b[v];b[
v] := b[ v+1];b[v+1]:=f;{} x:=false;
End; until x;{ab }write(
 '['= ' );for v:=s
to
{v-1}z
do{v+1-2*x}
write{x[a-2]
}(b[v]:3
); End.

```

Рис. 3. Код программы.

#### **Решение.**

1. Как известно, в *Pascal* комментарии могут быть обозначены в фигурных скобках и начинаться со строки `"/"`.

Удалим комментарии и осуществим правильные переносы строк в исходном тексте.

Получим:

```

Uses crt;
Const z = 10;
hhh = 10;
y = 2 * hhh + 1;
s = hhh - z + 1;
Type vec = array[s..z] of integer;
var a, b: vec;
v, f: integer;
x: Boolean;

```

Begin

```

ClrScr;
Randomize;

Write('[ ]=');
for v:= s to z do
begin
  f:= random(y);
  a[v]:= hhh - f + s - 1;
  Write(a[v]: 3);
End;
writeln;

b:= a;
Repeat
  x:= true;
  for v:= s to z - 1 do
  if b[v] > b[v + 1] then
  Begin
    f:= b[v];
    b[v]:= b[v + 1];
    b[v + 1]:= f;
    x:= false;
  End;
Until x;

write('[ ]=');
for v:= s to z
  do Write (b[v]: 3);
End.

```

## 2. Избавимся от запутывающих лишних переменных в начале программы:

```

Uses crt;
Const z = 10;
hhh = 10;
y = 21;
s = 1;
Type vec = array[1..z] of integer;
var a, b: vec;
v, f: integer;
x: Boolean;

```

## 3. Заменяем фиктивные переменные числами и объединим переменные с одинаковыми значениями

```

Uses crt;
Const z = 10;
y = 21;
Type vec = array[1..z] of integer;
var a, b: vec;
v, f: integer;
x: Boolean;

```

```

Begin

```

```

ClrScr;
Randomize;

Write('[]=');
for v:= 1 to z do
begin
    f:= random(y);
    a[v]:= z - f ;
    Write(a[v]: 3);
End;
writeln;

b:= a;
Repeat
    x:= true;
    for v:= 1 to z - 1 do
    if b[v] > b[v + 1] then
    Begin
        f:= b[v];
        b[v]:= b[v + 1];
        b[v + 1]:= f;
        x:= false;
    End;
Until x;

write('[]=');
for v:= 1 to z
do Write(b[v]: 3);
End.

```

#### 4. Переименуем переменные по смыслу и пронумеруем строки

```

Uses crt; //1
Const length = 10; //2
range = 21; //3
Type vector = array[1..length] of integer; //4
var a, b: vector; //5
index, number: integer; //6
flag: Boolean; //7

Begin //8
ClrScr; //9
Randomize; //10

Write('[]='); //11
for index:= 1 to length do //12
begin //13
    number:= random(range); //14
    a[index]:= length - number; //15
    Write(a[index]: 3); //16
End; //17
writeln; //18
b:= a; //19

```



```

Repeat                                     //20
  flag:= true;                             //21
  for index:= 1 to length - 1 do          //22
    if b[index] > b[index + 1] then      //23
      Begin                               //24
        number:= b[index];               //25
        b[index]:= b[index + 1];         //26
        b[index + 1]:= number;           //27
        flag:= false;                   //28
      End;                                //29
  Until flag;                              //30

write('[ ]=');                             //31
for index:= 1 to length
  do Write(b[index]: 3);                 //32
End.                                       //33

```

#### 5. Разбираем код на смысловые части:

- в строках с 1 по 7 происходит Инициализация переменных;
- в строках с 8 по 19 заполняется массив длины 10 числами в диапазоне от -10 до 10 и выводим на экран
- в строках с 20 по 30 сортируется массив методом пузырька
- в строках с 31 по 33 выводиться получившийся массив на экран.

**Ответ:** программа реализует алгоритм сортировки массива.