# Лабораторная работа 5

## Цель работы

Изучение условных операторов языка Си. Изучение массивов и основных приемов работы с массивами.

## Задание

Составить программу, выполняющую действия согласно варианту задания

| № | Задание |
|---|---|
| 1 | для заданного натурального числа **n** определить количество цифр в записи числа. Поменяв первую и последнюю цифру в записи числа, определить величину нового числа. |
| 2 | для заданных натуральных чисел **n** и **m** вычислить сумму **m** первых и сумму **m** последних цифр в записи числа **n** и найти знакочередующуюся сумму цифр числа **n**. Если m меньше количество цифр в числе n, недостающие цифры принять равными 0. |
| 3 | для заданного натурального числа **n** поменять все цифры в записи числа **n**; приписав в начале и конце числа единицы, определить величину нового числа. |
| 4 | найти все четверки простых чисел из первых 100 натуральных, принадлежащих одному десятку. |
| 5 | для заданного числа **n** (n<15) построить n строк треугольника Паскаля. |
| 6 | найти первые 120 натуральных чисел, сумма цифр которых равна 10 и вывести их на экран в виде матрицы 10X12. |

## Теоретическая часть

### Массивы в языке Си

Массив - это группа элементов одинакового типа. Форма объявления массива на языке Си следующая:

```
тип_данных имя_массива[размер1][[размер2][размер3]…];
```

"тип_данных" – определяет тип данных, хранящихся в массиве.
"размер1","размер2"… – определяет количество элементов в массиве, т.е. его размерность.

**ВНИМАНИЕ!!!** Элементы массива в языке Си нумеруются с нуля!

Пример описания одномерного массива из 10 целых чисел и двумерного массива вещественных чисел из 10 строк и 10 столбцов.

```
int page[10];
float bigmas[10][10];
```

Обращение к элементам этих массивов происходит следующим образом: page[1] - обращение ко второму элементу массива page, bigmas[0][0] – обращение к первому элементу первой строки массива bigmass.

При задании массива возможна и его инициализация. В этом случае присваиваемые значения указываются в квадратных скобках. Пример инициализации одномерного массива целых чисел:

```
int s[3]={1,2,3};
```

Если размер массива не указан, то он определяется по числу начальных значений. Но рекомендуется всегда указывать размер объявляемого массива.

```
int day[]={31,28,31,30,31,30,31,31,30,31,30,31};
```

При инициализации многомерных массивов начальные значения для каждой новой строки рекомендуется заключать в фигурные скобки. Если отдельных фигурных скобок нет, то инициализация производится по мере возрастания индексов.

Примеры инициализации двумерного массива:

```
int s[2][3]={{4,5,6},{7,8,9}};
int f[2][3]={10,11,12,13,14};
char p[2][2]={{'n'},{'y'}};
```

Массив s инициализируется полностью заданными значениями. В массиве f из его шести значений (размер массива f - 2 строки и 3 столбца) инициализируется только первые 5 элементов (это элементы с индексами 0,0 0,1 0,2 1,0 1,1). В массиве p инициализируются только 2 элемента: p[0][0]='n' и p[1][0]='y'. Если задан размер массива, то значения, не заданные явно, принимают значения, равные 0.

Если не проинициализировать элементы массива перед началом работы с ним, то внешние и статические массивы инициализируются нулем, а автоматические и регистровые будут содержать "мусор", оставшийся в этом участке памяти.

### Указатели и массивы

В языке Си существует сильная взаимосвязь между указателями и массивами, причем эта связь настолько сильна, что указатели и массивы фактически надо рассматривать одновременно.

Любое действие, которое достигается индексированием массива, может быть выполнено и с помощью указателей. Вариант с указателями, в общем-то, будет работать быстрее, но он, по крайней мере, для начинающих, несколько тяжеловат для понимания.

Описание **int a[10]** определяет массив **a** размером в 10 элементов, т.е. это блок из 10 последовательных объектов, именуемых **a[0], a[1],..., a[9].** Запись **a[i]** обозначает элемент в i-й позиции от начала.

Если **pa** – это указатель на целое значение, описанный как **int *pa**, то присваивание **pa=&a[0]** присваивает **pa** адрес нулевого элемента массива **a**, т.е. **pa** содержит адрес **a[0]**. Если теперь выполнить присваивание **x=*pa**, то в переменную **x** будет скопировано содержимое **a[0]**.

Если **pa** указывает на отдельный элемент массива **a**, то по определению **pa+1** указывает на следующий элемент, и, вообще, **pa-i** указывает на i-й элемент перед **pa**, а **pa+i** – на i-й элемент после. Таким образом, если **pa** указывает на **a[0]**, то ***(pa+1)** относится к содержимому **a[1]**, **pa+i** есть адрес **a[i]**, а ***(pa+i)** есть содержимое **a[i]**.

Эти замечания справедливы вне зависимости от типа переменных в массиве a. Определение операции "добавление 1 к указателю" и другой адресной арифметики подразумевает масштабирование, связанное с размером памяти для объекта, на который показывает указатель. Таким образом, при вычислении выражения pa+i значение i, прежде, чем будет добавлено к pa, будет умножено на размер объекта, на который указывает pa.

Фактически любое упоминание массива приводится транслятором к ссылке на начало этого массива, т.е. имя массива есть ссылочное выражение. Это приводит к небольшому числу полезных следствий. Так как имя массива есть синоним для местоположения нулевого элемента, то присваивание **pa=&a[0]** можно записать и в таком виде: **pa=a**.

Не удивительно теперь, по крайней мере, на первый взгляд, что значение **a[i]** можно записать как ***(a+i)**. Вычисляя **a[i]**, транслятор сразу же переводит его в ***(a+i)** – эти две формы полностью эквивалентны. Применяя операцию & к обеим частям этого равенства, получаем, что **&a[i]** и **a+i** также идентичны: a+i - адрес i-го элемента относительно a.

С другой стороны, если **pa** - указатель, то его можно использовать с индексом: **pa[i]** идентично ***(pa+i)**. Короче, любой массив и индексное выражение можно записать как указатель и смещение и, наоборот, причем это можно делать даже в одном операторе.

Однако между именем массива и указателем есть одно различие, о котором следует всегда помнить. Указатель есть переменная, так что **pa=a** и **pa++** есть осмысленные операции. Имя же массива – константа, а не переменная, поэтому операторы типа **a=pa** или **a++**, или **p=&a** недопустимы.

Пример. Программа распечатки содержимого одномерного массива с использованием указателей. Массив предварительно проинициализирован.

```
#include<stdio.h>
void main()
{
int p[5]={1,2,3,4,5};
int *ref;
    ref=p;
    printf("\n");
    for(int i=0;i<5;i++)
         printf("%d\t",*(ref+i));
}
```

Рассмотрим теперь, как получить доступ к элементу многомерного массива, используя указатели. Допустим, в программе описан трехмерный массив и указатель на него:

```
int arr[L][M][K], *ptr;
ptr=&arr[0][0][0];
```

Массив arr состоит из L элементов, каждый из которых – двумерный массив MxN. Каждый массив MxN в памяти располагается по строкам. Необходимо получить доступ к элементу arr[i][j[k]. Последовательно это вычисляется следующим образом:

```
ptr                  - адрес 0-го массива M на N
ptr+i*(M*N)          - адрес i-го массива M на N
ptr+i*(M*N)+j*N      - адрес j-й строки i-го массива M на N
ptr+i*(M*N)+i*N+k    - адрес элемента arr[i][j][k]
*(ptr+i*(M*N)+i*N+k) - значение элемента arr[i][j][k]
```

### Условный оператор

В языке Си условный оператор имеет две формы записи:

```
if (выражение)
      оператор1
оператор3
```

и

```
if (выражение)
      оператор1
else
      оператор2
оператор3
```

Если выражение истинно, то выполняется оператор1, если оно ложно, то при использовании первой формы записи условного оператора управление передается оператору3, а при использовании второй формы записи выполняется оператор2. Если же выражение истинно, то выполняется оператор 1

Если в условном операторе необходимо выполнить несколько действий, то в качестве оператора1 (или оператора2) используется составной оператор { }.

В языке Си допускается использование вложенных операторов if. В этом случае, если используется ключевое слове else и нет фигурных скобок, то else относится к ближайшему ключевому слову if.

Пример. Рассмотрим два примера программы

```
Пример 1

if(a==b)
{
    if(a==0)
       b=2;
}
else a=3
Пример 2

if(a==b)
    if(a==0)
       b=2;

    else a=3
```

В первом примере else относится к первому if, а во втором - ко второму.

### Оператор переключатель switch

Оператор предназначен для организации выбора одного из нескольких вариантов. Общий вид оператора переключателя:

```
switch(выражение)
{
    case метка1:операторы1;
    case метка2:операторы2;
. . .
    case меткаN:операторыN;
    default:операторы;
}
```

Значение "выражения" вычисляется и сравнивается с "метками" (обычно это целые или символьные константы). В случае совпадения выполняется группа операторов, соответствующая метке. Оператор default выполняется, если значение выражения не совпало со значением ни одной метки. Наличие метки default необязательно. Перед группой оператором можно использовать не одну, а несколько меток.

Желательно в конце группы операторов, соответствующих каждой метке, использовать оператор break для завершения выполнения оператора переключателя (см. Рис. 6).

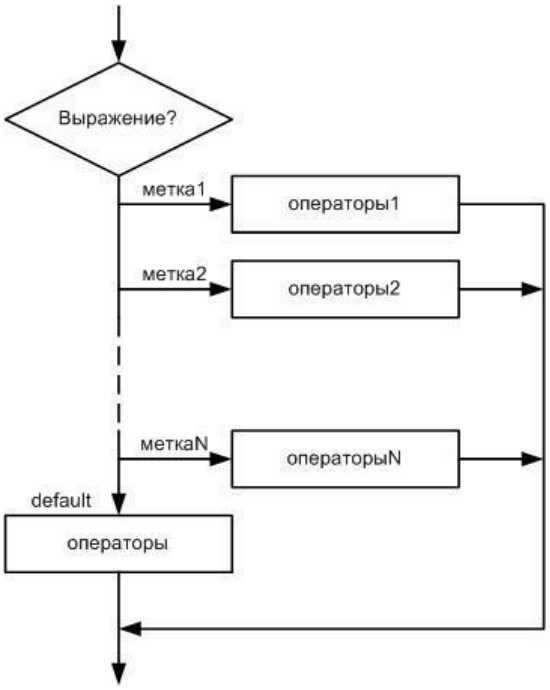


Рис. 6. Алгоритм выполнения оператора switch с оператором break

В случае отсутствия оператора break выполнение операторов, следующих за выбранным по переключателю, будет продолжено. (см. Рис. 7).

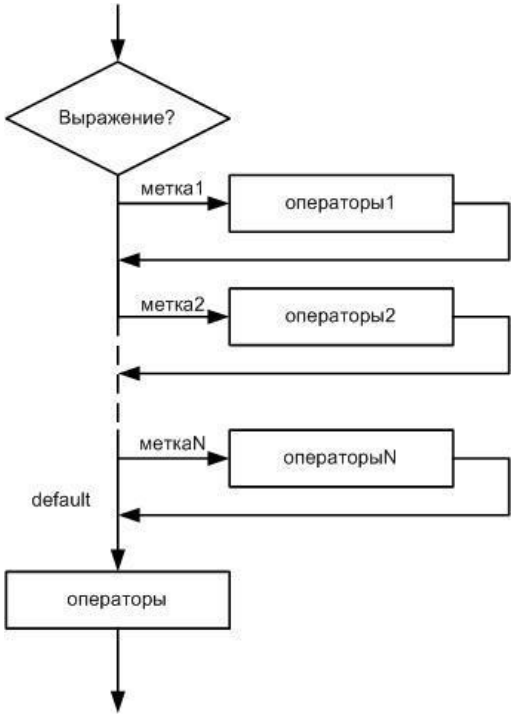


Рис. 7. Алгоритм выполнения оператора switch без оператора break

Оператор switch может быть вложен один в другой, при этом их метки могут совпадать.

### Оператор перехода goto

Оператор перехода имеет следующий вид:

```
goto метка;
```

## Рекомендации по выполнению лабораторной работы

### Определение цифр числа

Любое целое число D, записанное в позиционной системе счисления, выглядит в виде совокупности цифр $d_i$ (i=0…n-1), причем каждая цифра может иметь значение от 0 до p-1, где p – основание системы счисления: $D = d_{n-1}d_{n-2}d_{n-3}\ldots d_2d_1d_0$, (1)

при этом величина W числа D вычисляется следующим образом:

$$W = (((0*p+ d_{n-1})*p+ d_{n-2})*p+ d_{n-3})*p+….+d_2)*p+d_1)*p+d_0 \qquad (2)$$

При решении различных задач программирования возникает задача определения цифр некоторого целого числа. Для этого можно использовать следующий алгоритм, который вытекает из формулы (2):

1. Число D делится на основание системы счисления p.
2. Целая часть от деления принимается за новое значение числа D, а полученный остаток считается очередной цифрой числа D, начиная с младшей.
3. Пункты 1-2 повторяются до тех пор, пока частное от деления не станет равным нулю.

Если известны все цифры числа $d_i$, то величина W числа D определяется на основании формулы (2). Фрагмент программы для определения величины числа в десятичной системе счисления приведен ниже.

```
int digits[32];     /* массив для хранения цифр числа, начиная с младшей */
int i,n;            /* n - переменная для хранения количества цифр числа */
long s=0;           /* переменная для хранения величины числа */
     for(i=n-1;i>=0;i++)
          s = s*10+digits[i];
```

### Поиск простых чисел

Простое число – это число, которое делится только на 1 и на само себя. Для поиска простых чисел можно использовать алгоритм, называемый "решетом Эратосфена":

1. Пусть задано N последовательных натуральных чисел, начиная с 2.
2. Берем первое непомеченное натуральное число
3. Помечаем все числа, кратные выбранному.
4. Переходим к следующему непомеченному числу и повторяем пункт 3 до тех пор, пока не останется непомеченных чисел.

Фрагмент программы, реализующий этот алгоритм, приведен ниже.

```
int data[99];       /* массив для хранения натуральных чисел от 2 до 100 */
int i,j,n;
     for(i=0;i<99;i++)            /* заполним массив цифрами от 2 до 100 */
          data[i] = i+2;
     for(i=0;i<50;i++)            /* будем перебирать числа до 50 включительно */
     {
          if(data[i] != -1)       /* если число не помечено */
          {
               n = data[i];       /* запомним шаг */
               for(j=i+n;j<99;j=j+n)
                    data[j] = -1;    /* пометим все числа, кратные data[i] */
          }
     }
```

### Треугольник Паскаля

Треугольник Паскаля имеет для 8 строк следующий вид:

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
1   7   21  35  35  21  7   1
```

Рис. 8. Пример треугольника Паскаля

Как видно из приведенного рисунка, количество элементов в i-ой строке треугольника Паскаля равно i, причем первый и последний элементы равны 1. Все остальные элементы i-ой строки треугольника Паскаля вычисляются по правилу a[i][j]=a[i-1][j-1]+a[i-1]a[j]. Таким образом, j-й элемент i-ой строки равен сумме элементов предыдущей строки, стоящих над j-м элементом и слева от него.

## Содержание отчета

Отчет по лабораторной работе должен содержать:

- задание лабораторной работы, соответствующее варианту
- структурную схему алгоритма программы
- текст программы
- результаты работы программы