Лабораторная работа №4

Одномерные массивы

Цель работы: усвоение принципов работы с одномерными массивами; построение программ, оперирующих одномерными массивами данных.

Краткие теоретические сведения

Массив – это переменная, в которой можно хранить множество значений одного и того же типа. Для объявления массива необходимо указать требуемый тип данных и размер массива. Для указания размера массива количество его элементов помещается в квадратные скобки. Количество квадратных скобок указывает мерность массива. Массивы можно объявлять глобально и локально, а также статически и динамически.

Поскольку размер оперативной памяти ограничен моделью памяти, необходимо быть уверенным, что модель памяти позволит разместить массив в оперативной памяти. Для этого используется оператор sizeof.

Инициализация массива для символьных массивов может быть проведена с помощью строки.

Для инициализации массива любого типа необходимо указать присвоение заключенного в фигурных скобках значений. Инициализатор для объектов составных типов (каким является массив) состоит из списка инициализаторов, разделенных запятыми и заключенных в фигурные скобки. Каждый инициализатор в списке представляет собой либо константу соответствующего типа, либо, в свою очередь, список инициализаторов. Эта конструкция используется для инициализации многомерных массивов.

Наличие списка инициализаторов в объявлении массива позволяет не указывать число элементов по его первой размерности. В этом случае количество элементов в списке инициализаторов и определяет число элементов по первой размерности массива. Тем самым определяется размер памяти, необходимой для хранения массива. Число элементов по остальным размерностям массива, кроме первой, указывать обязательно.

Если в списке инициализаторов меньше элементов, чем в массиве, то оставшиеся элементы неявно инициализируются нулевыми значениями. Если же число инициализаторов больше, чем требуется, то выдается сообщение об ошибке.

Пример 1:

```
int    a[3]   = {0, 1, 2};  // Число инициализаторов равно числу элементов
double b[5]   = {0.1, 0.2, 0.3}; // Число инициализаторов меньше числа элементов
int    c[ ]   = {1, 2, 4, 8, 16}; // Число элементов массива опред. по числу инициализаторов
int    e[3]   = {0, 1, 2, 3}; // Ошибка – число инициализаторов больше числа элементов
```

Обратите внимание, что не существует присваивания массиву, соответствующего описанному выше способу инициализации.

```
int a[3] = {0, 1, 2};          // Объявление и инициализация
a = {0, 1, 2};                 // Ошибка
```

Доступ к элементам массива указывается имя массива, а затем номер требуемого элемента в квадратных скобках (для многомерных массивов указываются индексы по всем измерениям в отдельных квадратных скобках). Нумерация элементов массива начинается с нуля и номер максимального элемента на единицу меньше числа элементов по данному измерению.

Для доступа к конкретному элементу массива используются так называемые индексные выражения:

***<имя массива>*[*<целочисленное выражение>*]**

Здесь квадратные скобки являются требованием синтаксисам языка, а не признаком необязательности конструкции.

Индекс массива может быть не только константой, но и выражением, которое имеет целочисленный тип, например, ***a***[***i*** + 1] (здесь ***a*** должно быть именем ранее объявленного массива, а ***i*** – переменной целого типа).

Объявление массива и индексное выражение, используемое для доступа к элементу массива, имеют схожий синтаксис. Различаются они по месту в программе. Это особенно важно, когда мы определяем индекс последнего элемента массива. **Индексы элементов** массива в языке **Си** начинаются с **0**, и номер последнего элемента на **1** меньше количества элементов массива. Поэтому если Вы объявили массив ***x*** из 10 элементов, то не можете написать индексное выражение ***x***[10], т.к. в этом случае пытаетесь обратиться к элементу с индексом 10, которого нет в данном массиве. Компилятор не выдаст сообщения об ошибке, но результаты работы такой программы будут непредсказуемы.

Имя массива является адресом его начала. Оно имеет тип ***константный указатель*** *на* ***<тип элементов массива>*.**

Конструкция ***a*[*i*]** эквивалентна ***(*a* + *i*)**

Пример 2:

```
#include <stdio.h>

void main(void)
{
  int nArr[3];
  nArr[0] = 1;
  nArr[1] = 2;
  nArr[2] = 3;

  printf("\n\tArray\n\n");
  printf("nArr[0]\t=\t%d\n", nArr[0]);
  printf("nArr[1]\t=\t%d\n", nArr[1]);
  printf("nArr[2]\t=\t%d\n", nArr[2]);

  return 0;
}
```

Для доступа к массиву можно использовать указатель (объявление int *p;). Операция нахождения адреса переменной.

Пример 3:

```
int p=5;
int *q;
q=&p;
```

Доступ с помощью указателя применяется операция звездочка (*): p = *q; или *q = 7;

Арифметические операции над указателями приведут к увеличению значения на размер типа данных, на которые указывает указатель.

Для обработки элементов массива обычно используется оператор пошагового цикла *for*.

Для обработки многомерного массива используется соответствующее количество циклов.

Массивы не самодостаточны в том смысле, что не гарантируется хранение информации о количестве элементов вместе с самим массивом. В большинстве реализаций Си отсутствует проверка диапазона индексов для массивов.

При обращении к элементам массива индекс требуемого элемента указывается в квадратных скобках [].

Пример 4:

```
#include <stdio.h>
int main() {
  int a[]={5, 4, 3, 2, 1}; // массив a содержит 5 элементов
  printf("%d %d %d %d %d\n",a[0], a[1], a[2], a[3], a[4]);
  getchar();
  return 0;
}
```

Часто требуется задавать значения элементов массива в процессе выполнения программы. При этом используется объявление массива без инициализации. В таком случае указание количества элементов в квадратных скобках обязательно.

int a[10];

Для задания начальных значений элементов массива очень часто используется параметрический цикл:

Пример 5:

```
#include <stdio.h>
int main() {
  int a[5]; // объявлен массив a  из 5 элементов
  int i;
  // Ввод элементов массива
  for(i=0; i<5; i++)  {
     printf("a[%d] = ", i);
     scanf("%d", &a[i]); // &a[i] - адрес i-го элемента массива
    }
  // Вывод элементов массива
  for(i=0;i<5;i++)  {
      printf("%d ",a[i]); // пробел в формате печати обязателен
    }
  return 0;
}
```

Пример 6. Нахождение максимального элемента массива:

```
#include <conio.h>
#include <stdio.h>

void main() {
   int a[10] = {1, 2, 5, 3, 9, 6, 7, 7, 2, 4};
   unsigned i;
   int max;

   max = a[0];
   for (i = 1; i<10; i++) {
      if (a[i] > max) {
         max = a[i];
      }
   }

   printf("max element is %d", max);
   getch();
}
```

Пример 7. Удаление из одномерного массива всех отрицательных элементов :

```
…
for (i=0; i<n; i++)
        if (a[i]<0){                    //если найден отрицательный элемент, то
                for (j=i+1; j<n; j++)   //сдвинуть все элементы, стоящие после
                                        //удаляемого на одну позицию
                        a[j-1]=a[j];
                n--;                    //уменьшение размера поиска
                i--;                    //возврат к предыдущему индексу
}
```

Практическая часть

Упражнение 1

Создать файл проекта и разработать Си-программу в соответствии с вариантом, составить графическую диаграмму алгоритма программы.

1. Преобразовать массив следующим образом: все отрицательные элементы массива перенести в начало, сохранив исходное взаимное расположение, как среди отрицательных, так и среди остальных элементов массива.
2. Расположить элементы массива в обратном порядке.
3. Найти и поменять местами элементы, имеющие минимальное и максимальное значения в массиве.
4. Определить, упорядочены ли элементы массива по убыванию.
5. Вывести все неповторяющиеся элементы массива.
6. Сдвинуть элементы массива циклически на n позиций влево.
7. Сдвинуть элементы массива циклически на n позиций вправо.
8. Удалить минимальный и максимальный элементы массива.
9. Сформировать два новых массива: в первый записать отрицательные элементы исходного массива, во второй – все остальные.
10. Определить, симметричен ли массив, т.е. читается ли он одинаково слева направо и справа налево.
11.

Упражнение 2

Создать файл проекта и разработать Си-программу в соответствии с вариантом, составить графическую диаграмму алгоритма программы.

1. Найти количество элементов массива, отличающихся от среднего значения элементов массива не более чем на 3.
2. Определить количество инверсий в массиве (таких пар элементов, в которых большее значение находится слева от меньшего).
3. Определить количество элементов, значение которых больше среднего значения всех элементов массива.
4. Удалить элементы, значение которых меньше среднего значения всех элементов массива.
5. Удалить из массива повторяющиеся элементы.
6. Задан массив из k элементов. Определить количество различных элементов в массиве.
7. Задан массив из k символов латинского алфавита. Вывести на экран в алфавитном порядке все символы, которые входят в этот массив по одному разу.
8. Какая сумма элементов массива больше – с первого до элемента с номером К или от элемента с номером К+1 до последнего.
9. Дан массив из 10 элементов. Первые 4 упорядочить по возрастанию, последние 4 по убыванию.
10. Дан массив, например, состоящий только из 0 и 1. Определить самое большое количество подряд идущих единиц и вывести на экран индексы начала и конца этого диапазона.