ПЗ-06: **Работа с одномерными массивами**

**МАССИВЫ**

В языке Си используется сложный тип данных – ***массив****, представляющий собой упорядоченную конечную совокупность элементов одного типа.*

Число элементов массива называют его размером.

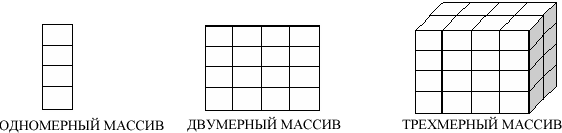
Каждый элемент массива определяется идентификатором массива и своим порядковым номером – индексом.

***Индекс*** *– целое число, по которому производится доступ к элементу массива.*

Индексов может быть несколько. В этом случае массив называют многомерным, а количество индексов одного элемента массива является его размерностью.

Описание массива в программе отличается от описания простой переменной наличием после имени квадратных скобок, в которых задается количество элементов массива.

Например, **double a [10];** // описание массива из 10 вещественных чисел.



**Пример:**

Объявление массива целого типа с инициализацией начальных значений:

**int a[5]={2, 4, 6, 8, 10};**

Если в группе {…} список значений короче, то оставшимся элементам присваивается 0.

* ***Массив*** *– упорядоченная последовательность переменных одного типа, имеющая общее имя*



Использование констант при работе с массивами:

// N обязательно определено ранее как константа!

**const int N = 10;** /\* N – именованная константа целого типа int со значением 10 \*/

**int mas[10];**

**float A[N];**

Еще один способ объявить именованную константу:

**#define <имя\_константы> <значение>**

Пример:

**#define N 50**

**…**

**int A [N];**

**int B [2 \* N];**

|  |  |
| --- | --- |
| Объявление одномерного массива с инициализацией:  **int myMas1[4] = {0, 12, 10, 4};**  **char myMas2[ ] = {‘c’, ‘h’, ‘e’, ‘c’, ‘k’};**  **double myMas3[4] = {1, 2};**  **double myMas4[4] = {0};**  **~~double myMas4[ ];~~** /\* Нельзя!!! Необходимо либо указать количество элементов, либо присвоить значения \*/ |  |

Длина массива – количество байт, отводимое в памяти для хранения всех элементов массива.

**ДлинаМассива = РазмерЭлемента \* КоличествоЭлементов**

Для определения размера элемента массива может использоваться функция

int sizeof(тип);

Например: **sizeof(char) = 1;**

**sizeof(int) = 4;** // для 32-разрядных систем

**sizeof(float) = 4;**

**sizeof(double) = 8;**

|  |  |
| --- | --- |
| #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;  } |  |

Однако 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:

|  |  |
| --- | --- |
| #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  getchar(); getchar();  return 0;  } |  |

**Передача массива в функцию**

Обработку массивов удобно организовывать с помощью специальных функций.

Для обработки массива в качестве аргументов функции необходимо передать:

* адрес массива,
* размер массива.

Исключение составляют функции обработки строк, в которые достаточно передать только адрес.

При передаче переменные в качестве аргументов функции данные передаются как копии. Это означает, что если внутри функции произойдет изменение значения параметра, то это никак не повлияет на его значение внутри вызывающей функции.

Если в функцию передается адрес переменной (или адрес массива), то все операции, выполняемые в функции с данными, находящимися в пределах видимости указанного адреса, производятся над оригиналом данных, поэтому исходный массив (или значение переменной) может быть изменено вызываемой функцией.

**Пример**: Дан массив из 10 элементов. Поменять местами наибольший и начальный элементы массива. Для операций поиска максимального элемента и обмена использовать функцию.

|  |  |
| --- | --- |
| #include <stdio.h>  // Функция обмена  void change(int \*x, int n)  {  // x - указатель на массив (адрес массива)  // n - размер массива  int i;  int max, index;  max = x[0];  index = 0;  // Поиск максимального элемента  for (i = 1; i<n; i++)  {  if (x[i]>max)  {  max = x[i];  index = i;  }  }  // Обмен  x[index] = x[0];  x[0] = max;  }  // Главная функция  int main()  {  int a[10];  int i;  for (i = 0; i<10; i++)  {  printf("a[%d] = ", i);  scanf("%d", &a[i]);  }  change(a, 10); // вызов функции обмена  // Вывод элементов массива  for (i = 0; i<10; i++)  printf("%d ", a[i]);  getchar();  getchar();  return 0;  } |  |

**Пример:** Дан массив размерности n. Вычислить произведение четных элементов

|  |  |
| --- | --- |
| #include <stdio.h>  // Функция вычисления произведения чётных элементов  int func(int \*x, int n) // произведение четных элементов  {  int p = 1; // начальное значение произведения  int i;  for (i = 0; i<n; i++)  {  if (x[i] % 2 == 0) // остаток от деления на 2 равен 0?  p = p \* x[i];  }  return p;  }  // Главная функция  int main()  {  int a[5]; // объявлен массив a из 5 элементов  int i;  int pr;  // Ввод элементов массива  for (i = 0; i<5; i++)  {  printf("a[%d] = ", i);  scanf("%d", &a[i]); // &a[i] - адрес i-го элемента массива  }  pr = func(a, 5); // вычисление произведения  printf("\n pr = %d", pr); // вывод произведения четных элементов  getchar(); getchar();  return 0;  } |  |

|  |  |  |
| --- | --- | --- |
| **Пример**: сложение двух массивов.  **int A[4] = { 2, 3, 4};**  **int B[] = {1, -1, 5};**  **int C[4] = {0};** | Вариант 1  **C[0] = A[0]+B[0];**  **C[1] = A[1]+B[1];**  **C[2] = A[2]+B[2];** | Вариант 2  **for (int i=0; i<3; i++)**  **C[i] = A[i]+B[i];** |

**Задача**: ввести с клавиатуры массив из 5 элементов, умножить все элементы на 2 и вывести полученный массив на экран.

|  |  |
| --- | --- |
| #include <stdio.h>  void main()  {  const int N = 5;  int A[N], i;  // ввод элементов массива  printf("Введите 5 элементов массива:\n");  for( i=0; i < N; i++ ) {  printf ("A[%d] = ", i );  scanf ("%d", & A[i] );  }  // обработка массива  printf("Введите 5 элементов массива:\n");  for( i=0; i < N; i++ ) {  printf ("A[%d] = ", i );  scanf ("%d", & A[i] );  }  // вывод результата  printf("Введите 5 элементов массива:\n");  for( i=0; i < N; i++ ) {  printf ("A[%d] = ", i );  scanf ("%d", & A[i] );  }  getch();  } | **Результат:**  **10 24 68 112 26**  A[0] = 5  A[1] = 12  A[2] = 34  A[3] = 56  A[4] = 13 |

**СТРОКИ КАК ОДНОМЕРНЫЕ МАССИВЫ ДАННЫХ ТИПА char**

В языке Си отдельного типа данных «**строка символов**» нет.

Работа со строками реализована путем использования одномерных массивов типа char, т.е. строка символов – это одномерный массив символов, заканчивающийся нулевым байтом.

Нулевой байт – это байт, каждый бит которого равен нулю, при этом для нулевого байта определена символьная константа ´\0´ (признак окончания строки, или «нуль-символ»).

**char str[10] ={‘V’ , ‘a’, ‘s’, ‘j’ , ‘а’, ‘\0’};**

**char S[ ] = “Работа со строками”;**

Строка может быть объявлена либо как **массив символов**, либо как переменная типа **char\***.

Каждое из двух приведенных ниже эквивалентных объявлений:

**char S[ ] = "строка";**

**char \*Sp = "строка";**

присваивает строковой переменной начальное значение "строка". Первое объявление создает массив из 7 элементов S содержащий символы 'с', 'т', 'р', 'о', 'к', 'а' и '\0'. Второе объявление создает переменную указатель Sp, который указывает на строку с текстом "строка", лежащую где-то в памяти. Но в любом случае число хранимых символов на 1 больше числа значащих символов за счет оконечного нулевого символа.

Доступ к отдельным символам строки осуществляется по индексам, начинающимся с нуля. Например, S[0] и Sp[0] - первые символы объявленных выше строк, S[1] и Sp[1] - вторые и т.д.

В приведенных объявлениях длина строк определялась автоматически компилятором. Можно объявлять строковые переменные заданной длины. Например, оператор char buff[100];

объявляет переменную buff, которая может содержать строку до 99 значащих символов плюс заключительный нулевой символ.

**Резюме**: В программе строки могут определяться следующим образом:

* как строковые константы;
* как массивы символов;
* через указатель на символьный тип;
* как массивы строк.

Кроме того, должно быть предусмотрено выделение памяти для хранения строки.

Любая последовательность символов, заключенная в двойные кавычки «», рассматривается как строковая константа.

**Функции ввода строк**

Для ввода строки может использоваться функция scanf(). Однако функция scanf() предназначена скорее для получения слова, а не строки. Если применять формат "%s" для ввода, строка вводится до (но не включая) следующего пустого символа, которым может быть пробел, табуляция или перевод строки.

Для ввода строки, включая пробелы, используется функция

**char \* gets(char \*);**

**Функции вывода строк**

Для вывода строк можно воспользоваться рассмотренной ранее функцией

**printf("%s", str); // str — указатель на строку**

или в сокращенном формате:

**printf**(str);

Для вывода строк также может использоваться функция

**int puts (char \*s);**

которая печатает строку s и переводит курсор на новую строку (в отличие от printf()). Функция puts() также может использоваться для вывода строковых констант, заключенных в кавычки.

**Функция ввода символов**

Для ввода символов может использоваться функция

**char getchar();**

которая возвращает значение символа, введенного с клавиатуры. Указанная функция использовалась в рассмотренных ранее примерах для задержки окна консоли после выполнения программы до нажатия клавиши.

**Функция вывода символов**

Для вывода символов может использоваться функция

**char putchar(char);**

**Пример:** Посчитать количество введенных символов во введенной строке.

|  |  |
| --- | --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  int main() {  char s[80], sym;  int count, i;  printf("Введите строку : ");  gets(s);  printf("Введите символ : ");  sym = getchar();  count = 0;  for (i = 0; s[i] != '\0'; i++)  {  if (s[i] == sym)  count++;  }  printf("В строке\n");  puts(s); // Вывод строки  printf("символ ");  putchar(sym); // Вывод символа  printf(" встречается %d раз", count);  getchar();  return 0;  } |  |