Тема : знакомство с C (Си)
МАССИВЫ и ФУНКЦИИ, упражнение

Содержание: массивы упражнение

- ## Массивы

Массивы - это группа элементов одинакового типа (double, float, int и т.п.). Из объявления массива компилятор должен получить информацию о типе элементов массива и их количестве. Объявление массива имеет два формата:

```
спецификатор-типа описатель [константное - выражение];

спецификатор-типа описатель [ ];
```

Описатель - это идентификатор массива .

Спецификатор-типа задает тип элементов объявляемого массива. Элементами массива не могут быть функции и элементы типа **void**.

Константное-выражение в квадратных скобках задает количество элементов массива. **Константное-выражение** при объявлении массива может быть опущено в следующих случаях:

- при объявлении массива и его инициализации,
- массив объявлен как формальный параметр функции,
- массив объявлен как ссылка на массив, явно определенный в другом файле.

В языке **C** определены только **одномерные массивы**, но поскольку элементом массива может быть массив, можно определить и **многомерные массивы**, то есть массивы массивов. Они формализуются списком константных-выражений, следующих за идентификатором массива, причем каждое константное-выражение заключается в свои квадратные скобки.

Каждое константное-выражение в квадратных скобках определяет число элементов по данному измерению массива, так что объявление двухмерного массива содержит два константных-выражения, трехмерного - три и т.д. Отметим еще раз, что в языке **C** первый элемент массива имеет индекс равный 0.

Примеры:

```
int a[2][3]; /* представлено в виде матрицы
                a[0][0]  a[0][1] a[0][2]
                a[1][0]  a[1][1] a[1][2]      */

 double b[10];
 /* вектор из 10  элементов имеющих тип double */
```

```
int C[3][3] = { { 2, 3, 4 },
                { 3, 4, 8 },
                { 1, 0, 9 } };
```

В последнем примере объявлен массив **C[3][3]**. Списки, выделенные в фигурные скобки, соответствуют строкам массива, в случае отсутствия скобок инициализация будет выполнена неправильно.

В языке **C** можно использовать сечения массива, как и в других языках высокого уровня, однако на использование сечений накладывается ряд ограничений. Сечения формируются вследствие опускания одной или нескольких пар квадратных скобок. Пары квадратных скобок можно отбрасывать только справа налево и строго последовательно.

Примеры:

```
int s[2][3];
```

Если при обращении к некоторой функции написать **s[0]**, то будет передаваться нулевая строка массива s.

Пример объявления символьного массива.

```
char str[] = "объявление символьного массива";
```

Следует учитывать, что в символьном литерале находится на один элемент больше, так как последний из элементов является управляющей последовательностью '\0'.

В C90 размерности массивов необходимо объявлять при помощи выражений из целых констант, причем размер массива фиксируется во время компиляции. В силу определенных обстоятельств, в C99 (Стандарт языка C 1999 года) это правило было изменено. В C99 можно объявить массив, размерности которого определяются любыми допустимыми целыми выражениями, в том числе и такими, значения которых становятся известны только во время выполнения. Такой массив называется массивом переменной длины. Однако такие массивы объявляются либо в функции, либо как параметры функции. Вот пример массива переменной длины:

```
void f(int diml, int dim2)
{

int matrix[diml][dim2]; /* двумерный массив переменной длины */
/* ... */
}
```

Массивы переменной длины добавлены в C99 главным образом для поддержки численных методов обработки данных. В программировании это средство распространено достаточно широко. Однако следует помнить, что стандарт C90 (и некоторые компиляторы C++) не поддерживает массивы переменной длины.

Упражнение

Написать программу, которая работает следующим образом:

1. У пользователя спрашивается сколько элементов он хочет ввести
2. Пользователь вводит желаемое число (целое)
3. Пользователь вводит элементы вектора (плавающий тип)
4. Вектор выводится на экран

Используются следующие функции:

a. Функция ввода числа элементов вектора
b. Функция ввода одного элемента вектора
c. Функция заполнения вектора (использует функцию ввода одного элемента вектора)
d. Функция вывода числа введенных элементов и вывод самих элементов.

В функции main могут быть:

- Декларирование необходимых переменных
- Вызовы функций

Возможное решение:

```
/*

FILE NAME: vector.c
AUTHOR:
DESCRIPTION: Определяем массив с максимальным количеством элементов – 100. Затем
запрашиваем у пользователя о желаемом количестве элементов. Пользователю
предлагается ввести элементы. Элементы выводятся на экран.

Здесь: double *  − указатель на double.


Можно использовать прототипы:


void taidaVektor(int , double[])
void valjastaVektor (int , double[]))


и соответствующие определения функций:


void taidaVektor(int argPikkus, double argVektor[]) {…}
void valjastaVektor (int argPikkus, double argVektor[])){…}


*/
```

```c
#include <stdio.h>

//прототипы функций
int sisestaVektoriPikkus(void);
double sisestaElement(int);
void taidaVektor(int, double *);
void valjastaVektor(int, double *);

int main (void)
{
   double vektor[100];
 //максимальное число элементов вектора- 100!

   int vektoriPikkus;
    vektoriPikkus=sisestaVektoriPikkus();

    taidaVektor(vektoriPikkus, vektor);
    valjastaVektor(vektoriPikkus, vektor);

    return 0;
}
```

```c
//функция ввода желаемого числа
//количества элементов вектора
int sisestaVektoriPikkus(void)
{
 int vPikkus;
 printf ("Ввести желаемое число количества элементов вектора: ");
 scanf("%d",&vPikkus);
  return(vPikkus);
}
 //функция ввода элемента
double sisestaElement(int argI)
{
        double elem;
        printf("Ввести %d. элемент: ",argI);
        scanf("%lf",&elem);
        return(elem);
}
```

```
//функция заполнения вектора
//аргументы:
//введенная длина и
//указатель на первый элемент массива
void taidaVektor(int argPikkus, double *argVektor)
{
      int i;
      for (i=0; i<argPikkus;i++)
      {
          argVektor[i]=sisestaElement(i);
      }
}

//функция вывода вектора
//введенная длина и указатель на первый элемент массива
void valjastaVektor(int argPikkus, double *argVektor)
{
      int i, vPikkus;
      vPikkus = argPikkus;      // vPikkus здесь использован, чтобы показать присвоение
                                // argPikkus локальной переменной
      printf("\nВы ввели вектор, состоящий из %d элементов: \n",vPikkus);
      for (i=0; i<vPikkus; i++)
      {
          printf("%6.2lf",argVektor[i]);
      }
      printf("\n");
}
```

- В ходе работа программа выводит следующую информацию:

```
Ввести желаемое число количества элементов вектора: 5
Ввести 0. элемент: 1
Ввести 1. элемент: 2
Ввести 2. элемент: 5
Ввести 3. элемент: 3
Ввести 4. элемент: 7

Вы ввели вектор, состоящий из 5 элементов:
  1.00  2.00  5.00  3.00  7.00
```

Задание

Дополнить программу следующим образом:

- Произвести необходимые проверки (например, для количества элементов **,** вводимого пользователем)
- Вектор **argVektor**, заполняемое количество элементов которого определено пользователем ранее, заполняется теперь числами **Fibonacci** (Фибоначчи). **Упростим задание так**: если пользователь ранее ввел число вводимых элементов большее **15**, то присвоим переменной,содержащей количество элементов, значение **15**, уведомив об этом пользователя.

  1, 1, 2, 3, 5, 8, 13, 21, ...

  Для вычисления чисел Фибоначчи использовать **циклы.** Написать **отдельную** функцию.

- Полученный вектор вывести на экран в диапазоне от первого элемента до числа, введенного ранее пользователем, но не более **15** элементов.
- Пользователь вводит 2 индекса выводимых элементов вектора:
  - **argv1** больше или равно, чем нижняя граница вектора! **Проконтролировать!**
  - **argv2** меньше или равно, чем верхняя граница вектора и соответствует количеству (не больше) чисел Фибоначчи ! **Проконтролировать!**
- Выдаются желаемые числа Фибоначчи в диапазоне от **argv1** до **argv2**. Написать **отдельную** функцию.
- Найти сумму этих элементов и их произведение (создать **отдельные функции**!).

Отчет оформляется в MS Word (или в другом текстовом редакторе), представляется в распечатанном виде. Представляемые программы должны быть прокомментированы.

Использованы материалы:

- ***Громов, Титаренко***. Программирование на языке С.
- ***Керниган Б., Ритчи Д. Язык С.***
- ***Материалы*** *Helena Kruus (MSc)*
- ***The cplusplus.com tutorial.*** *Complete C++ language tutorial* *http://www.cplusplus.com/doc/tutorial/index.html [9.11.2010]*
- ***The GNU C library*** *http://www.gnu.org/software/libc/manual/html_node/ [9.11.2010]*

Марина Брик

Составлено: 16.11.2007
Обновлено*: 28.10.2008*
Обновлено: 10.11.2010