Тема : знакомство с С (Си)

УКАЗАТЕЛИ и МАССИВЫ (динамическое размещение)

Содержание: указатели массивы и указатели динамическое размещение массивов глобальные переменные

- ## Указатели

Указатель – это адрес памяти, распределяемой для размещения идентификатора (в качестве идентификатора может выступать имя переменной, массива, структуры, строкового литерала). В том случае, если переменная объявлена как указатель, то она содержит адрес памяти, по которому может находится скалярная величина любого типа. При объявлении переменной типа указатель, необходимо определить тип объекта данных, адрес которых будет содержать переменная, и имя указателя с предшествующей звездочкой (или группой звездочек).

Указатель объявляется следующим образом.

1. Вначале указывается тип указателя. Это некоторый тип языка **C**. В данном случае он определяет тип объекта, на который указывает указатель.
2. Вслед за этим через пробел ставится звездочка - *. Она обозначает, что следующая за ней переменная является указателем.

Формат объявления указателя:

спецификатор-типа * переменная.

Спецификатор-типа задает тип объекта.

Размер переменной объявленной как указатель, зависит от архитектуры компьютера и от используемой модели памяти, для которой будет компилироваться программа. Указатели на различные типы данных не обязательно должны иметь одинаковую длину.

Примеры:

```
unsigned int * a;      /* переменная  a  представляет собой указатель
                          на тип unsigned int (целые числа без знака) */
double * x;            /* переменная  x  указывает  на  тип  данных с
                          плавающей  точкой  удвоенной  точности     */
char *buf;             /* объявляется  указатель с именем buf
                          который указывает на  переменную типа char */
```

Поясним применение указателей на примере:

```
int main (void)
{

int a=7;

int *p;

     p=&a;
     *p=10;

printf("%4d%4d \n", *p,a);
}
```

Здесь имеется некоторая переменная **a**, которая является целым числом и значение которой равно 7:

```
int a=7;
```

Объявим указатель на **int:**

```
int *p;
```

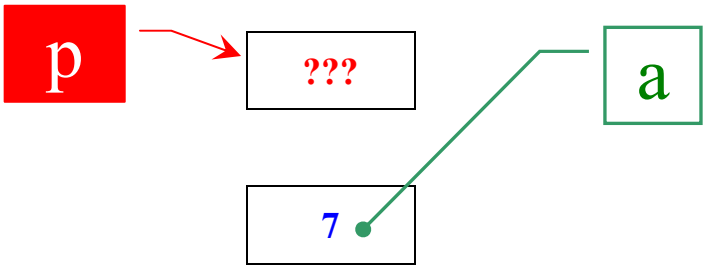


Рис. 1.

Указатель **p** указывает сейчас на неопределенное значение (Рис. 1).
Запишем следующее:

```
p=&a; /* & - операция взятия адреса */
```

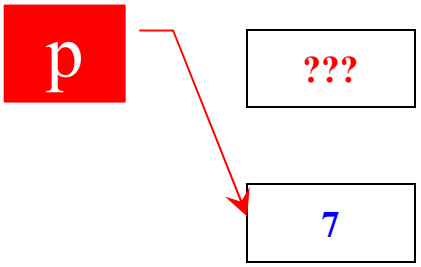


Рис. 2.

Теперь **p** указывает на переменную **a** (Рис. 2)**.**

Если затем напишем

```
*p=10;
```

то переменной **a** присвоится значение **10.**

▪ Массивы и указатели

Поскольку имя массива является указателем, то допустимо, например, такое присваивание:

```
int array[25];
int *ptr;
ptr=array;
```

Указатели на многомерные массивы в языке **C** - это массивы массивов, т.е. такие массивы, элементами которых являются массивы. При объявлении таких массивов в памяти компьютера создается несколько различных объектов. Например при выполнении объявления двумерного массива **int a [4][4]** в памяти выделяется участок для хранения значения переменной **a**, которая является указателем на массив из четырех указателей. Для этого массива из четырех указателей тоже выделяется память. Каждый из этих четырех указателей содержит адрес массива из 4 элементов типа **int**, и, следовательно, в памяти компьютера выделяется четыре участка для хранения четырех массивов чисел типа int, каждый из которых состоит из 4 элементов. Такое выделение памяти показано на схеме на Рис.3.

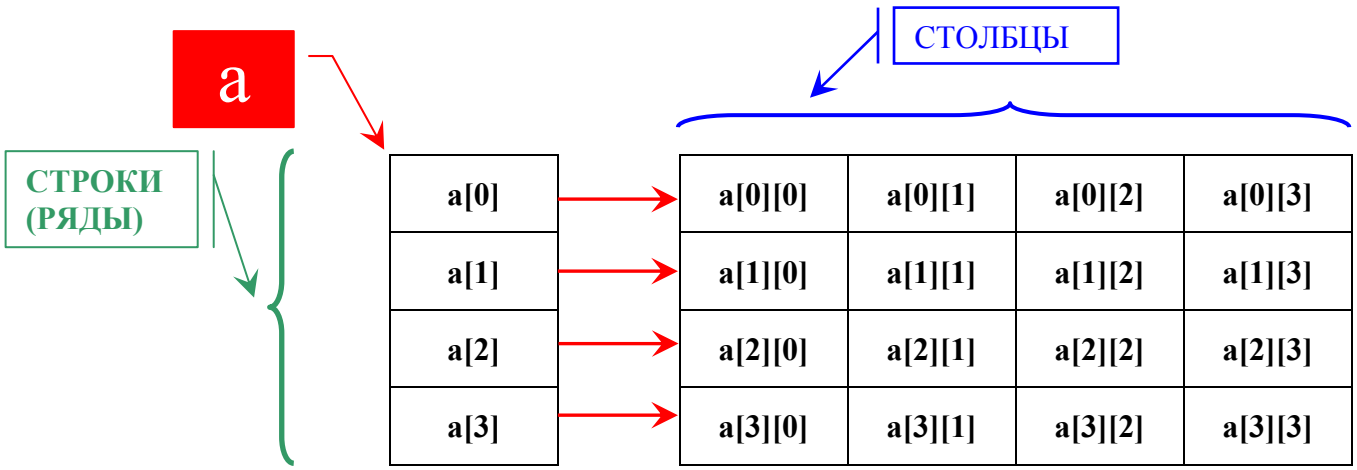


Рис. 3.

Таким образом, объявление **a[4][4]** порождает в программе три разных объекта: указатель с идентификатором **a**, безымянный массив из четырех указателей и безымянный массив из

16 чисел типа int. Доступ к элементам массива указателей может осуществляется с указанием одного индексного выражения в форме **a [2]** или ***(a+2)**.

- ## Динамическое размещение массивов (только для ознакомления)

Динамическое размещение массивов применяется, когда размерность массива планируется задавать в процессе работы программы.

Одномерный массив a**[10]** из элементов типа **float** можно создать следующим образом (**calloc – одна из функций выделения памяти)**:

```
float *a;
a=(float*)(calloc(10,sizeof(float));
```

Для создания двумерного массива вначале нужно распределить память для массива указателей на одномерные массивы, а затем распределять память для одномерных массивов. Пусть, например, требуется создать массив **a[m][n],** это можно сделать при помощи следующего фрагмента программы:

```
#include <stdio.h>
#include <stdlib.h>
....
     int main (void)
     {

          int **a;
          int n,m,i;
     ....
      scanf("%d %d",&n,&m);

                a=(int **)calloc(m,sizeof(int *));
                for (i=0; i<m; i++)
                    a[i]=(int *)calloc(n,sizeof(int));
     ....

       }
```

Следует только помнить, что ненужную для дальнейшего выполнения программы память следует освобождать при помощи функции **free**.

```
     {
     ....
          for (i=0; i<m; i++)
              a[i]=(int *)calloc(n,sizeof(int));

            for (i=0; i<m; i++)
              free(a[i]);
      free(a);
     ....
     }
```

▪ Глобальные переменые

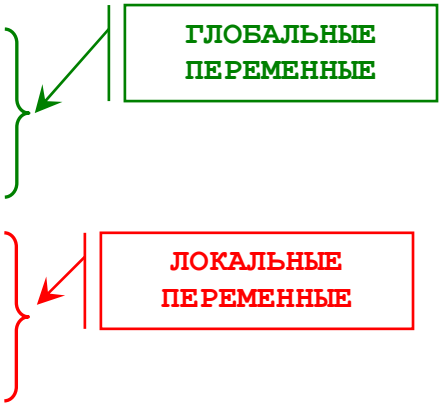
Объявления функций и переменных могут появляться как внутри так и вне определения функции. Любое объявление внутри определения функции считается "внутренним" (локальным). Объявление вне всех определений функций считается "внешним" (глобальным).

```
#include <stdio.h>

char * c="test";
int i;
float l;

int main(void)
{
float f;
double d;
.....

}
```



Пример

В следующем примере показан исходный текст простой программы на **C**. В исходной программе определяется функция main. Программа использует объявления для инициализации глобальных переменных **x** и **y**. Локальные переменные z и w объявляются, но не инициализируются. Память выделяется для всех этих переменных, но только x, y имеют содержательные значения. Значения z и w не имеют смысла до тех пор, пока им не присваиваются значения в выполняемых операторах.

```
.....

int x=1;              /* определяющие объявления для */
int y=2;              /* внешних (глобальных) переменных */


int main (void)                 /* определение функции main */
 {
      int z;          /* определения двух неинициализированных */
      int w;          /* локальных переменных */


      z=y+x;          /* выполняемые операторы  */
      w=y-x;

printf("z=%d w=%d",z,w);

return 0;
 }
```

Вывод программы:

Использованы материалы:

- ***Громов, Титаренко***. Программирование на языке С.
- ***Кернига́н Б., Ритчи Д. Язык С.***

Марина Брик
Составлено: 29.11.2007
Обновлено: 29.10.2008
Обновлено: 16.11.2010