Двумерные массивы

С позволяет создавать многомерные массивы. Простейшим видом многомерного массива является двумерный массив. Двумерный массив - это массив одномерных массивов. Двумерный массив объявляется следующим образом:

тип имя_массива[размер второго измерения][размер первого измерения];

Следовательно, для объявления двумерного массива целых с размером 10 на 20 следует написать:

```
int d[10] [20] ;
```

Посмотрим внимательно на это объявление. В противоположность другим компьютерным языкам, где размерности массива отделяются запятой, С помещает каждую размерность в отдельные скобки.

Для доступа к элементу с индексами 3, 5 массива d следует использовать

```
d[3] [5]
```

В следующем примере в двумерный массив заносятся числа от 1 до 12, после чего массив выводится на экран.

```
#include <stdio.h>
int main(void)
{
int t,i, num[3][4];
/* загрузка чисел */
for(t=0; t<3; ++t)
for (i=0; i<4; ++i)
num[t][i] = (t*4)+i+1;

/* вывод чисел */
for (t=0; t<3; ++t)
{
for (i=0; i<4; ++i)
printf("%d  ",num[t][i]);
printf ("\n");
}
return 0;
}
```

В данном примере num[0][0] имеет значение 1, num[0][1] имеет значение 2, num[0][2] - 3 и так далее. num[2][3] имеет значение 12.

Двумерные массивы сохраняются в виде матрицы, где первый индекс отвечает за строку, а второй -за столбец. Это означает, что правый индекс изменяется быстрее левого, если двигаться по массиву в порядке расположения элементов в памяти. На рис. показано

графическое представление двумерного массива в памяти. Левый индекс можно рассматривать как указатель на строку.

Число байт в памяти, требуемых для размещения двумерного массива, вычисляется следующим образом:

*число байт = размер второго измерения * размер первого измерения * sizeof (базовый тип)*

Предполагая наличие в системе 2-байтных целых, целочисленный массив с размерностями 10 на 5 будет занимать 10 * 5 * 2, то есть 100 байт.

Когда двумерный массив используется как аргумент функции, передается указатель на первый элемент. Функция, получающая двумерный массив, должна, как минимум, определять размер первого измерения, поскольку компилятору необходимо знать длину каждой строки для корректной индексации массива. Например, функция, получающая двумерный целочисленный массив с размерностями 5, 10, будет объявляться следующим образом:

```
void func1 (int x[] [10])
{
...
}
```

Можно определить размер второго измерения, но это не обязательно. Компилятору нужно знать размер первого измерения для правильного выполнения операторов типа

```
x [2] [4]
```

в функции. Если длина строки не известна, невозможно узнать, где начинается следующая строка.

Краткая программа, приведенная ниже, использует двумерный массив для хранения оценок каждого студента в классах учителя. Программа предполагает, что учитель имеет три класса и в каждом классе может быть максимум 30 студентов. Обратим внимание, как осуществляется доступ к grade из каждой функции:

```
#include <conio.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#define CLASSES 3
#define GRADES 30
int grade[CLASSES][GRADES];
void disp_grades(int g[] [GRADES]), enter_grades(void);
int get_grade(int num);
int main(void) /* программа для хранения оценок */
{
char ch;
for (;;) {
do {
printf("(E)nter grades\n");
printf("(R)eport grades\n");
printf("(Q) uit\n");
ch = toupper(getche());
}
while(ch!='E' && ch!='R' && ch!='Q');
```

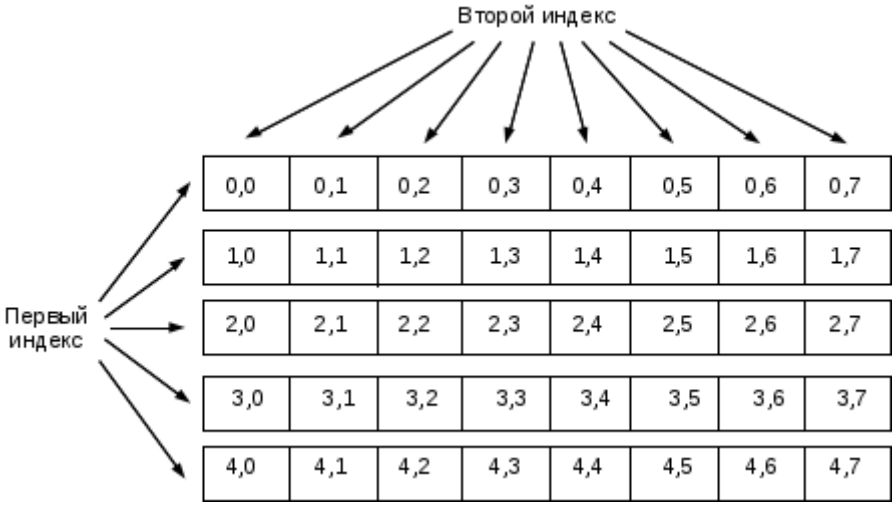
```
switch(ch)
{
case 'E':
enter_grades();
break;
case 'R':
disp_grades(grade);
break;
case 'Q':
return 0;
}
}
}

/* ввод каждой оценки студентов */
void enter_grades(void)
{
int t, i;
for(t=0; t<CLASSES; t++) {
printf("Class # %d:\n", t+1);
for(i=0; i<GRADES; ++i)
grade[t][i] = get_grade (i);
}
}

/* реальный ввод оценки */
int get_grade(int num) {
char s [80];
printf("enter grade for student # %d:\n", num+1);
gets (s);
return(atoi(s));
}

/* вывод оценок класса */
void disp_grades(int g[] [GRADES])
{
int t, i;
for(t=0; t<CLASSES; ++t)
{
printf("Class # %d:\n", t+1);
for(i=0; i<GRADES; ++i)
printf("grade for student #%d is %d\n", i+1, g[t][i]);
}
}
```

Рисунок. Размещение двумерного массива в памяти


Второй индекс
Первый индекс
0,0 0,1 0,2 0,3 0,4 0,5 0,6 0,7
1,0 1,1 1,2 1,3 1,4 1,5 1,6 1,7
2,0 2,1 2,2 2,3 2,4 2,5 2,6 2,7
3,0 3,1 3,2 3,3 3,4 3,5 3,6 3,7
4,0 4,1 4,2 4,3 4,4 4,5 4,6 4,7