Лабораторная работа №10

Двумерные массивы.

Цель работы: Изучение теоретических сведений и получение практических навыков по работе с многомерными массивами.

Теоретические сведения

Язык программирования Си позволяет создавать многомерные массивы. В самой общей форме объявление многомерного массива будет выглядеть следующим образом:

```
type Array[size1][size2]...[sizeN];
```

где `type` - это один из зарезервированных типов языка Си, `Array` - имя массива, `size1,...,sizeN` - размерности массива по каждому измерению. Например, следующая конструкция позволяет создать целочисленный трехмерный массив:

```
int Threedim[5][10][4];
```

Двумерные массивы

Простейшей формой многомерных массивов являются двумерные массивы. Любой двумерный массив представляет собой массив одномерных массивов. Для объявления двумерного массива размерности `N×M` необходимо использовать следующую конструкцию

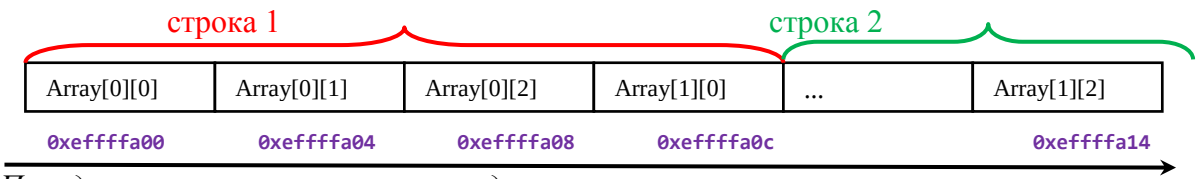
```
type Array[N][M];
```

Двумерный массив можно наглядно представить в виде таблицы значений с `N` строк и `M` столбцов (см. рис. 1).

	столбец 0	столбец 1	столбец 2	столбец 3	столбец M
строка 0	Array[0][0]	Array[0][1]	Array[0][2]	Array[0][3]	Array[0][M]
строка 1	Array[1][0]	Array[1][1]	Array[1][2]	Array[1][3]	Array[1][M]
строка 2	Array[2][0]	Array[2][1]	Array[2][2]	Array[2][3]	Array[2][M]
строка N	Array[N][0]	Array[N][1]	Array[N][2]	Array[N][3]	Array[N][M]

Рисунок 1 - Представление двумерного массива в виде таблицы.

Таким образом, каждый элемент массива `Array` определяется двумя индексами `i` и `j`, являющимися уникальными для каждого элемента: `Array[i][j]`. Расположение двумерного массива в памяти компьютера показано на рисунке 2. Все ячейки, хранящие значения элементов массива, являются смежными и располагаются в памяти последовательно друг за другом (элементы 1-ой строки, элементы 2-ой строки и т.д.).



Порядок расположения элементов двумерного массива в памяти компьютера

Рисунок 2 - Расположение двумерного массива типа int в памяти компьютера. Разница в 4 байта в адресах массива связана с тем, что каждый элемент занимает в памяти компьютера 4 байта.

Инициализация двухмерных массивов

Многомерные массивы можно инициализировать при определении, используя фигурные скобки:

```
int A[3][4]={
{0,1,2,3},      /* инициализация первой строки - индекс 0 */
{4,5,6,7},     /* инициализация второй строки - индекс 1 */
{8,9,10,11}   /* инициализация третьей строки - индекс 2*/
};
```

Отметим, что внутренние скобки, разбивающий набор элементов на отдельные строки, являются необязательными. К примеру, массив A, инициализированный выше, можно также инициализировать следующим образом:

```
int A[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Работа с элементами двумерного массива

Для доступа к элементу двухмерного массива необходимо использовать индексы, соответствующие строке и столбцу

```
int val = A[2][3];
```

Такая форма записи элемента соответствует 4-му элементу из третьей строки. Приведем пример программы, которая выводит на экран элементы массива:

```
#include <stdio.h>
int main () {
   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;
```

```
    /* output each array element's value */
   for ( i = 0; i < 5; i++ ) {
      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i, j, a[i][j] );
      }
   }

   return 0;
}
```

Указатели и двумерные массивы. Динамическое выделение памяти

Рассмотрим двухмерный следующий массив:

```
int Array[N][M];
```

Каждую строку двухмерного массива `Array` можно рассматривать как указатель на одномерный массив `*Array[i]`.

Для доступа к ячейке двухмерного массива можно также использовать аппарат указателей:

```
*(*(Array+i)+j)
```

Данная запись полностью эквивалентна `Array[i][j]`. Таким образом, имя массива является указателем на массив. Отметим также, что не обязательно каждый указатель `*(Array+i)` должен указывать на область памяти одинакового размера. В результате каждый столбец массива может иметь разный размер.

Следующий пример показывает один из способов динамического выделения памяти под двухмерный массив, который, однако, не гарантирует, что ячейки памяти будут соприкасающимися (см. на рис. 2):

```
int rows = 2;
int columns = 5;
int i;
int **matrix = (int **) malloc(rows * sizeof(int *));
for (i = 0; i < rows; i++)
{
        matrix[i] = (int *) malloc(columns * sizeof(int));
}
```

Сначала, используя функцию `malloc`, память выделяется динамически под каждую строку (первый "внешний" массив). Фактически мы выделили память для хранения `rows` указателей, доступ к которым можно получить через `*(matrix+i)` или `*matrix[i]`. Затем память выделяется внутри строк отдельно под каждый элемент. Теперь каждый из указателей `*(matrix+i)` указывает на область памяти размером `columns*sizeof(int)`. Поскольку две функции `malloc` используется отдельно – это приводит к тому, что выделяемые из кучи[1] ячейки памяти не обязательно будут соседствующими. Рисунок 3 показывает, как настроены указатели и что происходит в памяти компьютера с адресами.

Освобождение динамически выделенной памяти будет выглядеть следующим образом:

```
for (i = 0; i < rows; i++)
{
    free(matrix [i]);
}
free(matrix);
```

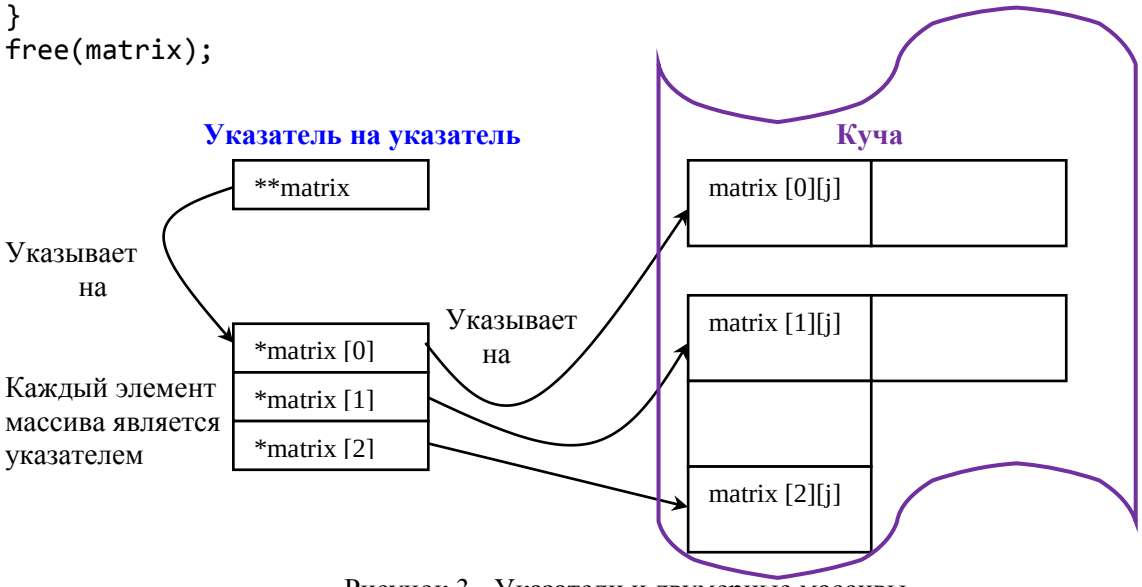


Рисунок 3 - Указатели и двумерные массивы.

Двумерные массивы как параметры функций

Если в программе используются динамические многомерные массивы, возможно использование «неоднородных» функций, которые не специфицируют в момент компиляции, насколько велики передаваемые в качестве аргумента массивы. Другими словами, функция может оперировать массивами разных размеров. В этом случае функция будет выглядеть следующим образом:

```
double func(double **array, int nrows, int ncolumns)
```

[1] Куча (eng. heap) - название структуры данных, с помощью которой реализована динамически распределяемая память приложения, а также памяти зарезервированной под эту структуру. Куча - длинный отрезок адресов памяти, поделенный на подряд идущие блоки различных размеров.

```
{
        int i,j;
        for(i = 0; i < nrows; i++)
        {
                for(j = 0; j < ncolumns; j++)
                        array[i][j] = 0;
        }
}
```

Данная функция в качестве параметра принимает указатель на указатель на `double`.

Контрольные вопросы

1. Двумерный массив. Как объявить двухмерный массив? Приведите примеры.
2. Сколько измерений могут иметь массивы в языке Си.
3. Как выделить память под многомерный массив динамически? Приведите пример.
4. Как освободить память, выделенную под многомерный массив динамически? Приведите пример. Что произойдет с памятью, выделенной под массив динамически, если она не будет освобождена в программе?
5. Как работать с указателями на двухмерные массивы? Приведите примеру? Какой смысл имеет конструкция: **p.
6. Как передать в функцию двухмерный массив?