Lesson\_10

|  |  |
| --- | --- |
| Коллеги, здравствуйте. Рад, вас приветствовать на десятом, юбилейном уроке курса основы языка С. На этом уроке нас ждёт продолжение разговора о массивах. |  |
| Массив в языке С может иметь сколько угодно измерений. Самые распространённые – это двумерные и трёхмерные, которые легко себе представить в виде таблицы или куба соответственно. Итак, массив это структура, содержащая элементы. Двумерный массив - это массив из массивов, содержащих элементы. Трёхмерный - это массив из массивов, содержащих массивы. И так далее.  В массиве могут находиться любые типы данных, мы, для удобства, будем рассматривать работу массивов с числами.  *Давайте для удобства попробуем визуализировать двумерный массив. Создадим двумерный массив, например 5\*5, и вот здесь в комментариях я попробую его нарисовать с помощью псевдографики. Массив 5\*5 – это 5 столбцов и 5 строчек. Соответственно, каждая строчка – это будет у нас младший индекс, а каждый столбец – старший индекс.*  *Трехмерный массив… Боюсь, что псевдографики в комментариях для этого не хватит. Но он может быть 3\*3\*3 – это всем известный кубик Рубика. У него есть 6 граней, в каждой из которых 3\*3 квадратика.* | int twoDimensional[5][5];  /\*  \* [][][][][]  \* [][][][][]  \* [][][][][]  \* [][][][][]  \* [][][][][]  \*/  int threeDimensional[3][3][3]; // Rubik’s cube |
| Как работать с многомерными массивами мы рассмотрим на примере двумерного массива. Поставим для себя задачу - сформировать таблицу Пифагора (раньше такие на тетрадях в клетку печатали на обратной стороне). Таблица подразумевает наличие двух измерений - строк и колонок. *Для этого объявим функцию advancedArraysMain. Выведем в консоль фразу Multi-dimensional Arrays! Объявим константы rows и cols и присвоим им значения 10, rows – это количество строк, а cols - соответственно столбцов Создадим двумерный массив, table[rows][cols]. Итак, мы создали массив размером rows, в каждом элементе которого содержится ссылка на массив размером cols - т.е. массив массивов содержащих значения.*  Таблица Пифагора представляет собой таблицу, где строки и столбцы озаглавлены множителями, а в ячейках таблицы находится их произведение. Вот это самое произведение мы и будем выводить.  Заполнение таких массивов значениями ничем не отличается от заполнения одномерных массивов. Мы можем заполнять их посредством конструкции в фигурных скобках, когда мы заранее знаем все значения. Например, вот так:  int arr[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}} - объявляем и инициализируем двумерный массив размером 3х4.  Также мы можем заполнять их поэлементно, как мы сделаем это сейчас.  *Давайте напишем двойной цикл который будет заполнять нашу таблицу.*  *Объявим переменные итераторы. И с помощью внешнего цикла фор пройдемся по всем строкам массива, а с помощью вложенного по всем столбцам массива, при этом будем записывать в каждый элемент массива результат умножения множителей.* *Формула (r + 1) \* (c + 1) позволяет исключить 0 из нашей таблицы. Давайте разберем более подробно работу такой конструкции.* Внешний цикл при каждой итерации перемещает нас на одну строчку вниз. Вложенный, при каждой итерации, перемещает нас на одно значение вправо. Важно понять, что на одну итерацию внешнего цикла приходится cols итераций вложенного. Т.е. с помощью такой конструкции мы поочередно перебираем все элементы массива.  Далее напишем такой же двойной цикл, который будет выводить на экран наши двумерные массивы в удобном для нас виде.  *Запустим нашу программу и увидим, что все отлично работает.* | void advancedArraysMain() {  printf(“Multi-dimensional Arrays! \n”);  const int rows = 10, cols = 10;  int table[rows][cols];  // int arr[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}}  int r, c;  for(r = 0; r < rows; r++) {  for(c = 0; c < cols; c++)  table[r][c] = (r+1)\*(c+1);  }  for(r = 0; r < rows; r++) {  for(c = 0; c < cols; c++)  printf("%3d ", table[r][c]);  printf("\n");  } |
| Как уже говорилось, массивы могут содержать данные любых типов, в том числе и указатели. Именно это позволяет массиву хранить другие массивы, строки и прочие ссылочные типы данных. Используя массивы указателей, мы можем создать, например, массив строк.  Давайте запишем char\* stringArray[3] = {"Hello", "C", "World"};  Это указатели на строки, такой тип данных является указателем. И мы можем создать из этих указателей массив. Используя массивы указателей, мы можем создать, например, двумерный массив, где каждый элемент не обязан быть того же размера, что и остальные.  Но строки и сложно-составленные указатели - это темы, которые очень сильно выходят за рамки Основ языка, но это не помешает нам разобраться со строками на следующих уроках | char\* stringArray[3] = {"Hello", "C", "World"};  for(r = 0; r < 3; r++)  printf("%s ", stringArray[r]); |
| До новых встреч на следующих уроках, коллеги. |  |