
Массивы в языке Си. Многомерные массивы. Массивы и указатели. Массивы и функции. Примеры

Массивы в языке Си

Массив – это составной тип данных в основе которого лежит набор однотипных скалярных(имеющих одно значение) или составных типов данных.

Другое определение:

Массив — это последовательность элементов одного типа, расположенных в памяти *подряд*.

Объявление одномерного массива:

```
int arr[10]; // массив из 10 элементов типа int
```

- Индексация начинается с нуля: `arr[0]` , `arr[1]` , `arr[2]` , ..., `arr[9]` .
- Инициализация:

```
int arr[5] = {1, 2, 3, 4, 5}; // Задаем массиву готовые значения  
int arr[] = {1, 2, 3}; // размер массива определяется автоматически
```

Основные операции:

- Чтение:

```
int a = arr[1];
```

- Запись:

```
arr[3] = 52;
```

Многомерные массивы

Многомерные массивы — массивы *массивов*, по сути - матрица(*Понравилось Шмегуле*).

Объявление и инициализация многомерного массива:

```
int matrix[3][4]; // массив 3x4
int matrix[2][2] = {{1, 2}, {3, 4}}; // массив 2x2 с заданными значениями
```

- Обращение к элементу: `matrix[1][0]` (строка 1, столбец 0). Чтение и запись аналогично одномерному массиву.
- В памяти хранится как один блок данных: строки идут подряд. То есть:
`stroka1|stroka2|stroka3`

Пример:

```
int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};
printf("%d", matrix[1][2]); // Вывод: 6
```

Массивы и указатели

Массивы жестко связаны с указателями, прямая вторая половинка <3.

Имя массива - адрес его первого элемента.

Пример:

```
int arr[5] = {10, 20, 30, 40, 50};
int *ptr = arr; // ptr указывает на arr[0]
printf("%d", *(ptr + 2)); // Вывод: 30
```

- `arr[i]` то же самое, что и `*(arr + i)`
- Адрес элемента: `&arr[i]`.

Многомерные массивы и указатели

Многомерные массивы также можно обрабатывать с использованием указателей.

Как уже было сказано, в памяти многомерный массив хранится как последовательность элементов (в порядке "строки за строками").

Связь указателей и многомерных массивов:

- Имя двумерного массива (`matrix`) - это указатель на первую строку (массив под индексом `[0][0]`).
- `matrix[i]` - указатель на строку `i`.
- Элемент массива можно получить как `*(*(matrix + i) + j)`.

Разбор этой шняги:

1. `*(matrix + i)` - получаем позицию в памяти первого элемента определенной строки, путем прибавления индекса строки к позиции первого элемента первой строки. *Вот как то так. (если не поняли, внизу файла сноска с кратким объяснением от гпт)*
2. `*(*(matrix + i) + j)` - к строке прибавляем индекс нужного элемента и получаем значение.

Пример:

```
int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};

int *row = matrix[0]; // Указывает на первую строку
printf("%d\n", *(row + 1)); // Вывод: 2 (второй элемент первой строки)

int value = (*(matrix + 1) + 2); // Элемент из строки 1, столбца 2
printf("%d\n", value); // Вывод: 6
```

Прочее:

- Обращение к строкам - `matrix[i]` то же самое что и `*(matrix + i)`.
- Обращение к элементам - `matrix[i][j]` эквивалентен `*(*(matrix + i) + j)`.

Массивы и функции

Массивы передаются в функции **по адресу**, а не по значению.

Пример на передачу массива в функцию:

```
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
}

int main() {
    int arr[3] = {1, 2, 3};
    printArray(arr, 3);
}
```

- `arr` - указатель на первый элемент массива.

Многомерные массивы в функции

При передаче многомерного массива нужно указать размеры всех измерений, кроме первого. также необходимо явно указать тип.

Пример:

```

void printMatrix(int matrix[][3], int rows) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

```

Также можно передавать массив так:

```

void printMatrix(int *matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", *(matrix + i * cols + j));
        }
        printf("\n");
    }
}

```

В таком случае, элемент массива получать нужно так:

```

matrix[i][j] = *(matrix + i * cols + j)

```

Примеры

Нахождение суммы элементов массива (явно не используя указатель, т.к. имя массива - указатель на его первый элемент):

```

int sumArray(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

```

Вывод матрицы:

```
void processMatrix(int (*matrix)[3], int rows) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}
```

Транспонирование матрицы через указатели:

```
void transposeMatrix(int (*matrix)[3], int (*transposed)[2], int rows, int
cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transposed[j][i] = matrix[i][j];
        }
    }
}
```

Напоминаю, `transposed[j][i] = matrix[i][j]` будет равно

`*(*(transposed + j) + i) = *(*(matrix + i) + j)`

База

- Имя массива - указатель на его первый элемент.
- Передача массивов в функции осуществляется **по указателю**.
- Индексы массива `i`, `j` не должны быть равны или больше его размерности. если массив `arr[10][5]`, то максимальные `i` и `j` это 9 и 4.

Типо сноски

- `matrix + i` перемещает указатель к строке `i`.
- `*(matrix + i)` разыменовывает этот указатель, чтобы получить массив (строку) `i`.
- `*(matrix + i) + j` смещает указатель внутри строки `i` на элемент `j`.

- `*(*(matrix + i) + j)` возвращает значение этого элемента.