Тема 5-1.

Статические массивы и примеры на C++

для АСУб и ЭВМб

Темы лекции

- Понятие массив
- Статические массивы
- Пример алгоритмов сортировки
- Строки как массивы
- Строки как тип string

Многомерные массивы

- В языке C++ такие массивы рассматриваются как одномерные массивы одномерных массивов
- Поэтому такой массив может быть определен следующим образом:

```
int a[10] [5];
```

Многомерные массивы

- Двумерный массив может инициализироваться как одномерный массив:

```
int a[2] [3] = { 3, 45, 11, -8, 74, -10};
```

или как массив массивов:

```
int a[2] [3] = { {3, 45, 11}, {-8, 74, -10}};
```

- При наличии инициализатора в определении двумерного массива можно не указывать размер по первому измерению, например:

```
int a[ ] [3] = { {3, 45, 11}, {-8, 74, -10}};
```

Многомерные массивы

- Для двумерных массивов каждый из индексов записывается в отдельных квадратных скобках:

```
a[0] [2] = a[1] [2] + 4;
```

- Поскольку элементы двумерного массива располагаются в оперативной памяти в виде непрерывной последовательности, то возможно обращение к элементу массива с использованием одного индексного выражения

Многомерные массивы

- Пусть определение массива имеет вид:

  int a [m] [n],

где m, n – константы

- Тогда эквивалентными являются два обращения: a [i] [j] и a[i*m+j]

Указатели и одномерные массивы

- Переменная **array** содержит адрес первого элемента массива, как если бы это был **указатель**!

```
int array[] = {1, 2, 3, 4, 5};
std::cout << "array [0] = " << *array << std::endl;
// тип array – это int[5]
std::cout << "The array has address: " << array << '\n';
// тип &array[0]  – это int*
std::cout << "Element 0 has address: " << &array[0] << '\n';
```

- Хотя *оба* указывают на первый элемент массива, информация о типе данных у них разная!

Распад массива на указатель (decay)

- При вычислениях, статический массив распадается (неявно преобразовывается) в указатель на первый элемент массива!

```
int array[] = {1, 2, 3, 4, 5};
std::cout << *array<<std::endl; // выведется 1;
std::cout << sizeof(array) << '\n'; // размер массива в байтах, т.е. 20

int *ptr = array;
std::cout << *ptr<< '\n'; ; // выведется  1
std::cout << sizeof(ptr) << '\n'; // размер указателя, т.е. 4
```

- Доступ к элементам по-прежнему осуществляется через указатель, но информация, полученная из типа массива (например, его размер), не может быть доступна из типа указателя.

Передача массива в функцию

- При передаче массива в качестве аргумента в функцию, массив распадается в указатель на массив и этот указатель передается в функцию

```
void sizeByPointer(int *array) // Здесь массив рассматривается как указатель
{std::cout << sizeof(array) << '\n';}

void sizeByBracket (int array[]) //Неявно конвертируется array[] в *array
{std::cout << sizeof(array) << '\n';}

int main(){
    int array[] = {1, 2, 3, 4, 5};
    std::cout << sizeof(array) << '\n'; // размер массива в байтах, т.е. 20
// далее выведется размер указателя, а не размер массива!
    sizeByPointer(array);
    sizeByBracket(array);
}
```

Указатели и одномерные массивы

- С помощью указателей можно **манипулировать элементами массива,** как и с помощью индексов
- Имя массива это не стандартный указатель, и мы не можем изменить его адрес

```
int a[5] = {1, 2, 3, 4, 5};
a++;              // так сделать нельзя
int b = 8;
a = &b;           // так тоже сделать нельзя
```

Арифметика указателей и массивы

- Для перемещения по элементам массива используются отдельные указатели:

```
int a[5] = {1, 2, 3, 4, 5};
int *ptr = a;
int a2 = *(ptr+2);
std::cout <<  "value: " << a2 << std::endl;  // value: 3
```

- С помощью указателей можно перебрать массив:

```
for(int *ptr=a; ptr<=&a[4]; ptr++)
{
    std::cout << "address=" << ptr << "\tvalue=" << *ptr <<
std::endl;
}
```

Указатели и строки

Поскольку **массив символов** может интерпретироваться как **строка**, то указатель на значения типа **char** тоже может интерпретироваться как строка:

```
char letters[] = "hello";

char *p = letters;

std::cout << p << std::endl;        // hello

std::cout << p[4] << std::endl;        // 0
```

Многомерные массивы и указатели

```
int main()
{
    int a[3][4] = { {1, 2, 3, 4} , {5, 6, 7, 8}, {9, 10, 11, 12}};
    int n = sizeof(a)/sizeof(a[0]);          // число строк
    int m = sizeof(a[0])/sizeof(a[0][0]);    // число столбцов

    int *end = a[0] + n * m - 1;     // указатель на самый
    последний элемент 0 + 3 * 4 - 1 = 11
    for(int *ptr=a[0], i=1; ptr <= end; ptr++, i++)
    {
        std::cout << *ptr << "\t";
    }
```

Многомерные массивы и указатели

```
#include <iostream>
using namespace std;
int main() {
    int multi[4][4][3]={}; // Объявление 3-х мерного массива
    int (*p2multi)[3];// указатель на массив
    int (*p1multi);
    cout << multi[3][2][2] << "\n"; // Обращение к элементу.

    p2multi = multi[3]; // p2multi указатель на 4-ую «плоскость»

    p1multi = multi[3][2]; // p1multi указатель на 4-ую «плоскость»
                           и 3-ю строку.
}
```

Многомерные массивы и указатели

```
#include <iostream>
using namespace std;
int main() {
    int multi[4][4][3]={}; // Объявление 3-х мерного массива
    cout << sizeof(multi) << "\n";//192
    cout << sizeof(multi[3]) << "\n";//48
    cout << sizeof(multi[3][2]) << "\n";//12
}
```

Многомерные массивы: статические и динамические

- Статический (фиксированный) – массив массивов, указатели на массив
- Динамический массив – массив указателей, указатель на указатель