Программирование на языке C++
Лекция 2

Ссылки

Александр Смаль

Недостатки указателей

- Использование указателей синтаксически загрязняет код и усложняет его понимание. (Приходится использовать операторы * и &.)
- Указатели могут быть неинициализированными (некорректный код).
- Указатель может быть нулевым (корректный код), а значит указатель нужно проверять на равенство нулю.
- Арифметика указателей может сделать из корректного указателя некорректный (легко промахнуться).

Ссылки

- Для того, чтобы исправить некоторые недостатки указателей, в C++ введены ссылки.
- Ссылки являются “красивой обёрткой” над указателями:

```
void swap (int & a, int & b) {
    int t = b;
    b = a;
    a = t;
}

int main() {
  int k = 10, m = 20;
  swap (k, m);
  cout << k << ' ' << m << endl; // 20 10
    return 0;
}
```

Различия ссылок и указателей

- Ссылка не может быть неинициализированной.

```
int * p; // OK
int & l; // ошибка
```

- У ссылки нет нулевого значения.

```
int * p = 0; // OK
int & l = 0; // ошибка
```

- Ссылку нельзя переинициализировать.

```
int a = 10, b = 20;
int * p = &a; // p указывает на a
p = &b;       // p указывает на b
int & l = a;  // l ссылается на a
l = b;        // a присваивается значение b
```

Различия ссылок и указателей

- Нельзя получить адрес ссылки или ссылку на ссылку.

```
int a = 10;
int * p = &a;   // p указывает на a
int ** pp = &p; // pp указывает на переменную p
int & l = a;    // l ссылается на a
int * pl = &l;  // pl указывает на переменную a
int && ll = l;  // ошибка
```

- Нельзя создавать массивы ссылок.

```
int * mp[10] = {};   // массив указателей на int
int & ml[10] = {};   // ошибка
```

- Для ссылок нет арифметики.

lvalue и rvalue

- Выражения в C++ можно разделить на два типа:
  1. **lvalue** — выражения, значения которых являются *ссылкой* на переменную/элемента массива, а значит могут быть указаны слева от оператора =.
  2. **rvalue** — выражения, значения которых являются временными и не соответствуют никакой переменной/элементу массива.
- Указатели и ссылки могут указывать только на lvalue.

```
int a = 10, b = 20;
int m[10] = {1,2,3,4,5,5,4,3,2,1};
int & l1 = a;           // OK
int & l2 = a + b;       // ошибка
int & l3 = *(m + a / 2); // OK
int & l4 = *(m + a / 2) + 1;     // ошибка
int & l5 = (a + b > 10) ? a : b;  // OK
```

Время жизни переменной

Следует следить за временем жизни переменных.

```
int * foo() {
    int a = 10;
    return &a;
}

int & bar() {
    int b = 20;
    return b;
}
```

```
int * p = foo();
int & l = bar();
```