# Класс

# Операции класса

## Весь встроенный функционал класса

class X {

X(Sometype); //‘‘ordinary constructor’’: create an object

X(); //default constructor

X(const X&);

// copy constructor

X(X&&); //move constructor

X& operator=(const X&); // copy assignment: clean up target and copy

X& operator=(X&&);

// move assignment: clean up target and move

˜X(); //destructor:clean up

// ...

};

## Перегрузка операторов

Присваивание (=)

#include <iostream>

#include <cstring>

class MyString {

char\* data;

public:

// Конструктор

MyString(const char\* str = "") {

data = new char[strlen(str) + 1];

strcpy(data, str);

}

// Конструктор копирования

MyString(const MyString& other) {

data = new char[strlen(other.data) + 1];

strcpy(data, other.data);

}

// **Перегрузка копирующего присваивания**

MyString& operator=(const MyString& other) {

if (this == &other) return \*this; // защита от самоприсваивания

delete[] data; // освобождаем старую память

data = new char[strlen(other.data) + 1];

strcpy(data, other.data);

return \*this;

}

// **Перегрузка перемещающего присваивания**

MyString& operator=(MyString&& other) noexcept {

if (this == &other) return \*this;

delete[] data; // освобождаем старую память

data = other.data; // забираем указатель

other.data = nullptr; // "обнуляем" источник

return \*this;

}

// Деструктор

~MyString() {

delete[] data;

}

void print() const {

std::cout << (data ? data : "empty") << "\n";

}

};

int main() {

MyString a("Hello");

MyString b("World");

b = a; // копирующее присваивание

b.print();

MyString c("Move");

b = std::move(c); // перемещающее присваивание

b.print();

c.print(); // будет "empty"

}

## Пример

#include <string>

using namespace std;

string ident(string arg) // <-- copy constructor: s1 копируется в arg

{

return arg; // <-- move constructor: перемещаем arg в временный объект для результата

}

int main ()

{

string s1 {"Adams"};

// ^^^ constructor from string literal (инициализация из "Adams")

s1 = ident(s1);

// 1. copy constructor: s1 -> arg (внутри ident)

// 2. move constructor: arg -> временный объект результата ident()

// 3. move assignment: временный объект -> s1

// После этого s1 = "Adams"

string s2 {"Pratchett"};

// ^^^ constructor from string literal (инициализация из "Pratchett")

s1 = s2;

// ^^^ copy assignment: копируем содержимое s2 в s1

} // <-- destructor: освобождает ресурсы у s1 и s2