Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Лабораторная работа №2»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

Пермь 2023

**Постановка задачи**

1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.

**Вариант 8:**

Пользовательский класс КВАРТИРА

Адрес - string

Площадь - double

Количество комнат - int

**Исходный код программы:**

**Файл “Apartment.h”:**

#include <iostream>

#include <string>

using namespace std;

class Apartment {

string adress;

double area;

int number\_of\_rooms;

public:

Apartment(); // Конструктор без параметров

Apartment(string, double, int); // Конструктор с параметрами

Apartment(const Apartment&); // Конструктор копирования

~Apartment(); // Деструктор

string get\_adress(); // Селектор

void set\_adress(string); // Модификатор

double get\_area(); // Селектор

void set\_area(double); // Модификатор

int get\_number\_of\_rooms(); // Селектор

void set\_number\_of\_rooms(int); // Модификатор

void show(); // Просмотр атрибутов

};

**Файл “Apartment.cpp”:**

#include "Apartment.h"

#include <iostream>

#include <string>

using namespace std;

Apartment::Apartment() {

adress = "";

area = 0;

number\_of\_rooms = 0;

cout << "Конструктор без параметров для объекта: " << this << endl;

}

Apartment::Apartment(string adress, double area, int number\_of\_rooms) {

this->adress = adress;

this->area = area;

this->number\_of\_rooms = number\_of\_rooms;

cout << "Конструктор с параметрами для объекта: " << this << endl;

}

Apartment::Apartment(const Apartment &Other) {

adress = Other.adress;

area = Other.area;

number\_of\_rooms = Other.number\_of\_rooms;

cout << "Конструктор копирования для объекта: " << this << endl;

}

Apartment::~Apartment() {

cout << "Деструктор для объекта: " << this << endl;

}

// Селекторы

string Apartment::get\_adress() {

return adress;

}

double Apartment::get\_area() {

return area;

}

int Apartment::get\_number\_of\_rooms() {

return number\_of\_rooms;

}

//Модификаторы

void Apartment::set\_adress(string adress) {

this->adress = adress;

}

void Apartment::set\_area(double area) {

this->area = area;

}

void Apartment::set\_number\_of\_rooms(int number\_of\_rooms) {

this->number\_of\_rooms = number\_of\_rooms;

}

void Apartment::show() {

cout << "Адрес: " << adress << endl;

cout << "Площадь: " << area << endl;

cout << "Количество комнат: " << number\_of\_rooms << endl;

}

**Файл “Lab\_2.cpp”:**

#include "Apartment.cpp"

#include <iostream>

#include <string>

using namespace std;

Apartment make\_apartment() {

string adress;

double area;

int number\_of\_rooms;

cout << "Введите адрес квартиры: ";

cin >> adress;

cout << "Введите площадь квартиры: ";

cin >> area;

cout << "Введите колество комнат в квартире: ";

cin >> number\_of\_rooms;

Apartment apart(adress, area, number\_of\_rooms);

return apart;

}

void print\_apartment(Apartment apart) {

apart.show();

}

int main() {

Apartment apartment\_1; // Конструктор без параметров

apartment\_1.show();

Apartment apartment\_2("Адрес 2-ой квартиры", 32, 2); // Конструктор с параметрами

apartment\_2.show();

Apartment apartment\_3 = apartment\_2; // Конструктор копирования

apartment\_3.set\_adress("Адрес 3-ей квартиры");

apartment\_3.set\_area(20.1);

apartment\_3.set\_number\_of\_rooms(4);

print\_apartment(apartment\_3);

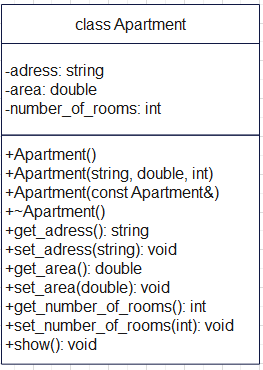
apartment\_1 = make\_apartment();

apartment\_1.show();

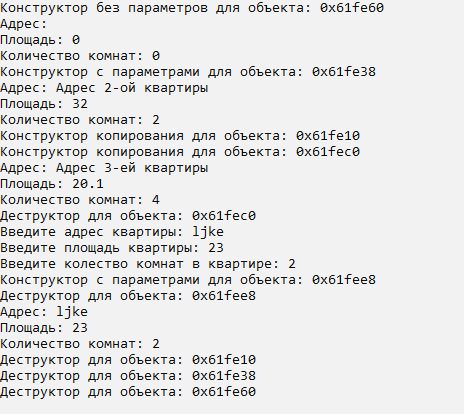
return 0;

}

**UML диаграмма:**



**Скриншот результата выполнения программы**



**Ответы на вопросы**

1. Для чего нужен конструктор?

**Ответ: Конструктор** - это специальный метод, который используется для инициализации объекта и вызывается автоматически при его создании.

1. Сколько типов конструкторов существует в C++?

**Ответ:** Конструктор без параметров, с параметрами и конструктор копирования:

class my\_class {

private:

int element:

my\_class() { // Конструктор без параметров

element = 0;

}

my\_class(int element) { // Конструктор с параметрами

this->element = element;

}

my\_class(const my\_class &other) { // Конструктор копирования

element = other.element;

}

};

1. Для чего используется деструктор? В каких случаях деструктор описывается явно?

**Ответ: Деструктор** - это специальный метод, используемый для освобождения памяти, выделенной объекту. Деструктор указывается явно в случае освобождения динамической памяти для предотвращения утечек памяти.

class my\_class {

public:

int \*array;

my\_class(int \*array) {

this->array = array;

cout << "Вызвался конструктор " << this << endl;

}

~my\_class() {

delete[] array;

cout << "Вызвался деструктор " << this << endl;

}

};

int main() {

int \*arr = new int[3];

int \*arr2 = new int[4];

my\_class ar(arr);

my\_class ar1(arr2);

return 0;

}

1. Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?

**Ответ:** Конструктор без параметров используется для инициализации объекта значениями по умолчанию в случае, если параметры объекта не были указаны.

class my\_class {

int element:

my\_class() { // Конструктор без параметров

element = 0;

}

};

Конструктор с параметрами используется для инициализации объекта параметрами, указанными при создании экземпляра класса.

class my\_class {

public:

int element;

my\_class(int element) { // Конструктор с параметрами

this->element = element;

}

};

int main() {

my\_class el(34);

}

Конструктор копирования используется для присваивания одного объекта класса всех полей другого объекта.

class my\_class {

public:

int element;

my\_class(const my\_class &other) { // Конструктор копирования

element = other.element;

}

};

1. В каких случаях вызывается конструктор копирования?

**Ответ:** Конструктор копирования вызывается в случае, когда в операторе объявления один объект используется для инициализации другого.

1. Перечислить свойства конструкторов.

**Ответ:**

- Конструктор должен иметь то же имя, что и класс

- Класс может иметь несколько конструкторов с разными параметрами

class my\_class {

public:

int element;

my\_class {

element = 0;

};

my\_class(int el) {

element = el;

};

};

- Конструктор не возвращает значение

- Если конструктор не был указан явно, то компилятор создаст пустой конструктор автоматически

1. Перечислить свойства деструкторов

**Ответ:**

- Деструктор должен иметь то же имя, что и класс, но со знаком тильда перед именем

- Деструктор не принимает аргументы

- Деструктор не возвращает значение.

- Деструктор вызывается автоматически, когда объект выходит из области видимости:

- Если деструктор не был указан явно, то компилятор создаст пустой конструктор автоматически

1. К каким атрибутам имеют доступ методы класса?

**Ответ:** Методы класса имеют доступ ко всем атрибутам класса.

class my\_class {

private:

int a;

protected:

int b;

public:

int c;

void show() {

cout << a << b << c;

};

};

1. Что представляет собой указатель this?

**Ответ:** Указатель this - скрытый константный указатель, содержащий адрес объекта, который вызывает метод класса.

class my\_class {

private:

int a;

public:

void set\_a(int a) {

this->a = a;

};

};

1. Какая разница между методами определёнными внутри класса и вне класса?

**Ответ:** Разница между методами, определёнными внутри и вне класса, в объявлении метода в классе.

class my\_class {

private:

int a;

public:

void set\_a(int);

void show\_a() {

cout << "a: " << a;

}

};

void my\_class::set\_a(int a) {

this->a = a;

}

int main() {

my\_class el;

el.set\_a(5);

el.show\_a();

return 0;

}

1. Какое значение возвращает конструктор?

**Ответ:** Конструктор не возвращает значение.

1. Какие методы создаются по умолчанию?

**Ответ:** Конструктор и деструктор.

1. Какое значение возвращает деструктор?

**Ответ:** Деструктор, как конструктор, не возвращает значение.

1. Дано описание класса

class Student {

string name;

int group;

public:

Student(string, int);

Student(const student&)

~Student();

};

Какой метод отсутствует в описании класса?

**Ответ:** В описании класса отсутствует конструктор без параметров: Student();

1. Какой метод будет вызван при выполнении следующих операторов:

Student \*s;

s = new Student;

**Ответ:** Конструктор по умолчанию.

1. Какой метод будет вызван при выполнении следующих операторов:

student s(“Ivanov”, 20);

**Ответ:** Конструктор с параметрами.

1. Какие методы будут вызваны при выполнении следующих операторов:

student s1(“Ivanov”, 20);

s2 = s1;

**Ответ:** Конструктор с параметрами, конструктор копирования.

1. Какие методы будут вызваны при выполнении следующих операторов:

student s1(“Ivanov”, 20);

student s2;

s2 = s1;

**Ответ:** Конструктор с параметрами и без, конструктор копирования.

1. Какой конструктор будет использоваться при передаче параметра в функцию print():

void print(student a)

{a.show();}

**Ответ:** Конструктор по умолчанию.

1. Класс описан следующим образом:

class Student {

string name;

int age;

public:

void set\_name(string);

void set\_age(int);

...

};

Student p;

Каким образом можно присвоить новое значение атрибуту name объекта p?

**Ответ:**

Описать метод set\_name(string) вне класса:

void Student::set\_name(string name) {

this->name = name;

}

Вызвать метод set\_name для объекта p: p.set\_name(“Ivanov”);