Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**Тема: Классы и объекты. Использование конструкторов.**

**Вариант 5**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Выполнил работу

Студент группы РИС-22-2Б

Бехтольт Дмитрий Андреевич

Проверил

Доцент кафедры ИТАС

Полякова Ольга Андреевна

2023 год

**Постановка задачи**

Пользовательский класс УРАВНЕНИЕ.

Коэффициент A - double.

Коэффициент B - double.

Коэффициент C - double.

**Код**

**KL2.25.cpp**

#include "Uravnenie.h"

#include <iostream>

#include <string>

using namespace std;

Uravnenie make\_uravnenie()

{

double a, b, c;

cout << "Vvod A >> ";

cin >> a;

cout << "Vvod B >> ";

cin >> b;

cout << "Vvod C >> ";

cin >> c;

Uravnenie t(a, b, c);

return t;

}

void print\_uravnenie(Uravnenie t)

{

t.show();

}

void main()

{

// конструктор без параметров

Uravnenie t1;

// конструктор с параметром

Uravnenie t2(2.5, 3.2, 4.8);

Uravnenie t3 = t2;

t3.set\_A(2.0);

t3.set\_B(1.0);

t3.set\_C(4.5);

// вывод

print\_uravnenie(t1);

print\_uravnenie(t2);

print\_uravnenie(t3);

t1 = make\_uravnenie();

t1.show();

}

**Uravnenie.cpp**

#include "Uravnenie.h"

#include <iostream>

#include <string>

using namespace std;

// конструктор без параметров

Uravnenie::Uravnenie()

{

A = 0;

B = 0;

C = 0;

cout << "Constructor bez parametrov dlya objecta " << this << endl;

}

// конструктор с параметрами

Uravnenie::Uravnenie(double a, double b, double c)

{

A = a;

B = b;

C = c;

cout << "Constructor s parametrami dlya objecta " << this << endl;

}

// конструктор копирования

Uravnenie::Uravnenie(const Uravnenie &t)

{

A = t.A;

B = t.B;

C = t.C;

cout << "Constructor copirovaniya dlya objecta " << this << endl;

}

// детструктор

Uravnenie::~Uravnenie()

{

cout << "Destructor dlya objecta " << this << endl;

}

// селекторы

double Uravnenie::get\_A()

{

return A;

}

double Uravnenie::get\_B()

{

return B;

}

double Uravnenie::get\_C()

{

return C;

}

// модификаторы

void Uravnenie::set\_A(double a)

{

A = a;

}

void Uravnenie::set\_B(double b)

{

B = b;

}

void Uravnenie::set\_C(double c)

{

C = C;

}

// просмотр атрибутов

void Uravnenie::show()

{

cout << "A : " << A << endl;

cout << "B : " << B << endl;

cout << "C : " << C << endl;

cout << endl;

}

**uravnenie.h**

#include <iostream>

#include <string>

using namespace std;

class Uravnenie

double A, B, C;

public:

Uravnenie(); // конструктор без параметров

Uravnenie(double, double, double); // конструктор с параметрами

Uravnenie(const Uravnenie&); // консутруктор копирования

~Uravnenie(); // деструктор

double get\_A(); // селектор

void set\_A(double); // модификатор

double get\_B();

void set\_B(double);

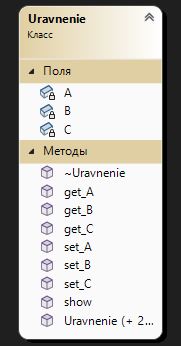
double get\_C();

void set\_C(double);

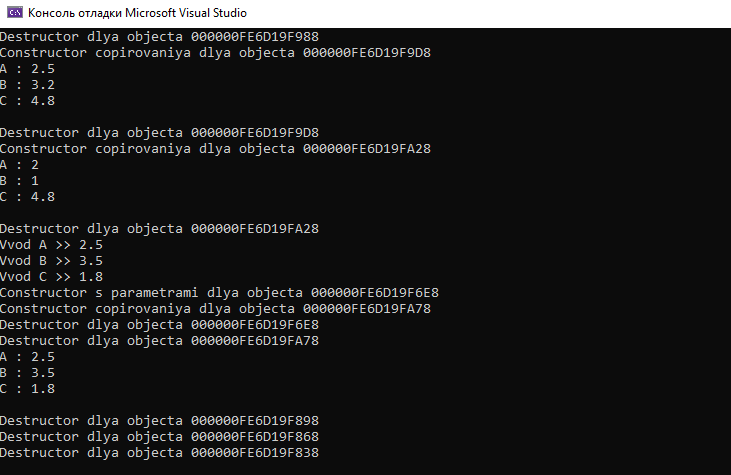
void show(); // просмотр атрибутов

};

**Диаграмма класса**

****

**Результат выполнения программы**

****

Ответы на контрольные вопросы

1. Конструктор необходим для инициализации полей класса.
2. Без параметров, с параметрами, конструктор копирования.
3. Для завершения работы с конструктором класса после выхода из области видимости функции. Если конструктор описан явно, то необходимо и деструктор описать явно.
4. Без параметров – для инициализации полей по умолчанию.

С параметрами – для инициализации полей переданными в конструктор значениями.

Копирования – для инициализации полей с помощью копирования их из другого обьекта.

1. Конструктор копирования вызывается в случае, когда нужно заполнить n обьектов одинаковыми значениями полей.
2. Инициализация полей переданными параметрами, инициализация полей по умолчанию, инициализация полей с помощью копирования, перегрузка конструкторов, конструктор списка по умолчанию, конструктор списка инициализаторов.
3. Не могут иметь аргументов; Не возвращают значение (или void ); Не может быть объявлен как const , volatile или static . Однако они могут вызываться для уничтожения объектов, объявленных как const , volatile или static ; Может быть объявлен как virtual . Используя виртуальные деструкторы, можно уничтожать объекты, не зная их тип — правильный деструктор для объекта вызывается с помощью механизма виртуальных функций. Для абстрактных классов деструкторы также могут объявляться как чисто виртуальные функции; не описаны явно – автоматически вызовутся.
4. Private, Public, Protected;
5. Адрес обьекта класса;
6. Внутри класса можно не передать параметры, вне – обязательно. Использование шаблонов ф-ий порой обязательно.
7. Конструкторы не возвращают значений.
8. Конструктор и деструктор.
9. Аналогично, деструктор не возвращает значений.
10. Сеттер.
11. Никакой.
12. Конструктор с параметрами.
13. Конструктор с параметрами для первого обьекта и конструктор копирования для второго.
14. Аналогично предыдущему, только сначала второй обьект вызовет конструктор без параметров.
15. С параметрами.
16. p.set\_name(“name”);