МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cт.преподаватель |  |  |  | Е.О.Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 5 |
| «Множественное наследование в языке С++» |
| по курсу: Объектно-ориентированное программирование |
|  |
|  |

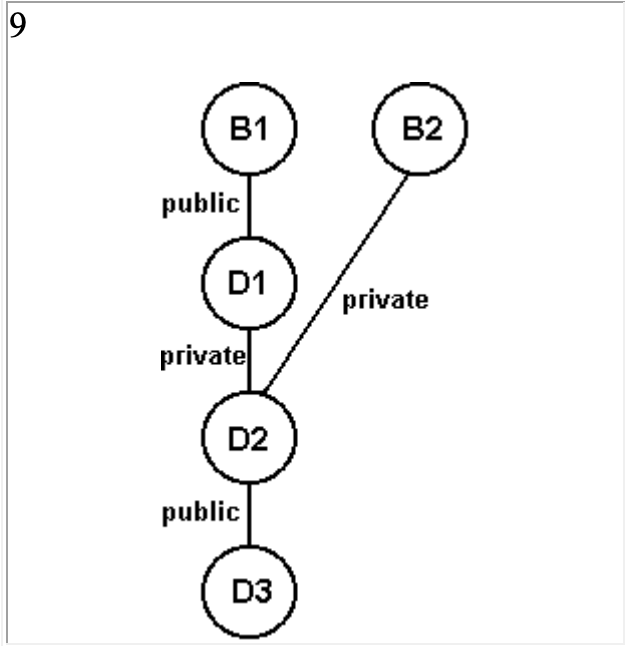
РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Опарин С.Н. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

1. **Задание**

**Вариант №9:**



1. **Листинг программы, реализующей алгоритм:**

#include <iostream>

using namespace std;

class B1{

private:

int b1;

public:

B1(int);

~B1();

void show\_B1();

};

// Базовый клаcс B2

class B2{

private:

int b2;

public:

B2(int);

~B2();

void show\_B2();

};

// D1 наследует публично у B1

class D1 : public B1 {

private:

int d1;

public:

D1(int, int);

~D1();

void show\_D1();

};

// D2 наследует приватно у D1 и B2

class D2 : private B2, private D1 {

private:

int d2;

public:

D2(int, int, int, int);

~D2();

void show\_D2();

};

// D3 наследует публично у D2

class D3 : public D2 {

private:

int d3;

public:

D3(int, int, int, int, int);

~D3();

void show\_D3();

};

B1::B1(int x){

cout << "B1 Created" << endl;

this -> b1 = x;

}

B1::~B1() {

cout << "B1 destroyed" << endl;

}

void B1::show\_B1(){

cout << "B1 = " << this -> b1 << endl;

}

B2::B2(int x){

cout << "B2 Created" << endl;

this -> b2 = x;

}

B2::~B2(){

cout << "B2 destroyed" << endl;

}

void B2::show\_B2(){

cout << "B2 = " << this -> b2 << endl;

}

D1::D1(int x, int y): B1(y){

cout << "D1 created" << endl;

this -> d1 = x;

}

D1::~D1(){

cout << "D1 destroyed" << endl;

}

void D1::show\_D1(){

cout << "D1 = " << this -> d1 << endl;

B1::show\_B1();

}

D2::D2(int x, int y, int i, int j): D1(y, i), B2(j){

cout << "D2 created" << endl;

this -> d2 = x;

}

D2::~D2(){

cout << "D2 destroyed" << endl;

}

void D2::show\_D2(){

cout << "D2 = " << this -> d2 << endl;

B2::show\_B2();

D1::show\_D1();

}

D3::D3(int x, int y, int i, int j, int k): D2(y, i, j, k){

cout << "D3 created" << endl;

this -> d3 = x;

}

D3::~D3(){

cout << "D3 destroyed" << endl;

}

void D3::show\_D3(){

cout << "D3 = " << this -> d3 << endl;

D2::show\_D2();

}

int main(){

cout << "D3 cl(1, 2, 3, 4, 5)\n" << endl;

D3 cl(1, 2, 3, 4, 5);

cout << "\nCледуя иерархии класса D3: \n" << endl;

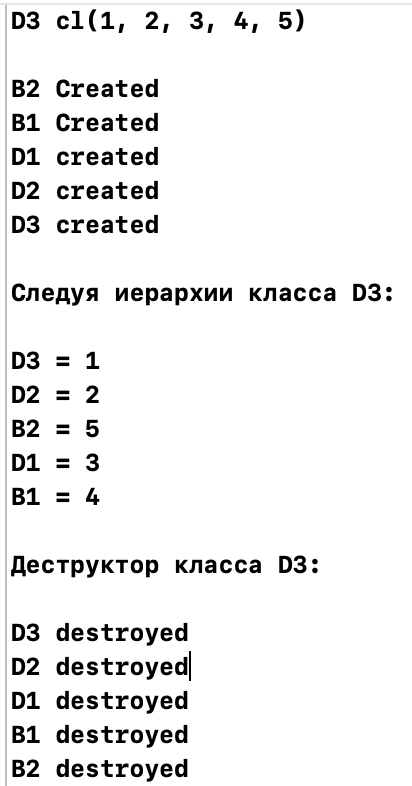
cl.show\_D3();

cout << "\nДеструктор класса D3: \n" << endl;

return 0;

}

Скриншот с результатами:



1. **Вывод**

В ходе лабораторной работы я получил практические навыкы при использовании множественного наследования в языке С++