МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ:

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

«Множественное наследование в языке С++»

ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4831 |  |  |  | К.А. Корнющенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

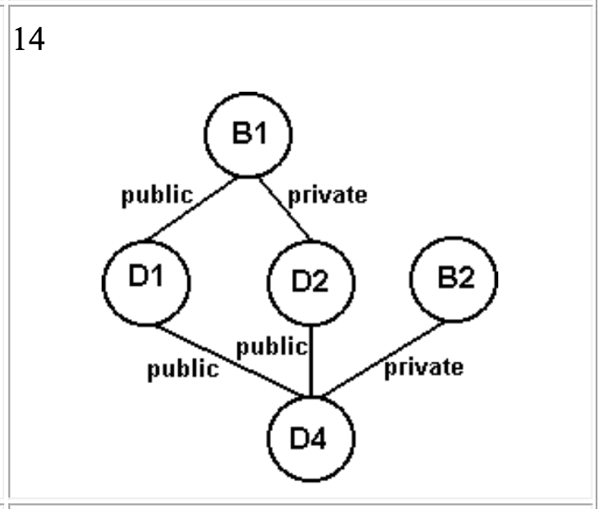
Санкт-Петербург 2020

Задание:

Получение практических навыков при использовании множественного наследования в языке С++.

В работе необходимо построить иерархию классов согласно схеме наследования, приведенной в варианте задания.

Каждый класс должен содержать:  
- инициализирующий конструктор  
- функцию show для вывода значений.  
- деструктор  
Функция main должна иллюстрировать иерархию наследования.



Листинг программы:

Main.cpp

#include <iostream>

#include "B1.hpp"

#include "B2.hpp"

#include "D1.hpp"

#include "D2.hpp"

#include "D4.hpp"

using namespace std;

int main() {

D4 temp(1, 2, 3, 4, 5);

temp.showD4();

}

B1.cpp

#include "B1.hpp"

B1::B1(int b1){

this->b1 = b1;

cout << "Работает конструктор B1: параметр = " << b1 << endl;

};

B1::~B1(){

cout << "~B1 сработал" << endl;

};

void B1::showB1(){

cout << "B1 = " << b1 << endl;

};

B1.hpp

#ifndef B1\_hpp

#define B1\_hpp

#include <iostream>

using namespace std;

class B1{

private:

int b1;

public:

B1(int b1);

~B1();

void showB1();

};

#endif /\* B1\_hpp \*/

B2.cpp

#include "B2.hpp"

B2::B2(int b2){

this->b2 = b2;

cout << "Работает конструктор B2: параметр = " << b2 << endl;

};

B2::~B2(){

cout << "~B2 сработал" << endl;

};

void B2::showB2(){

cout << "B2 = " << b2 << endl;

};

B2.hpp

#ifndef B2\_hpp

#define B2\_hpp

#include <iostream>

using namespace std;

class B2{

private:

int b2;

public:

B2(int b2);

~B2();

void showB2();

};

#endif /\* B2\_hpp \*/

D1.cpp

#include "D1.hpp"

D1::D1(int b1,int d1) : B1(b1){

this->d1 = d1;

cout << "Работает конструктор D1: параметр = " << d1 << endl;

};

D1::~D1(){

cout << "~D1 сработал" << endl;

};

void D1::showD1(){

cout << "D2 = " << d1 << endl;

showB1();

};

D1.hpp

#ifndef D1\_hpp

#define D1\_hpp

#include <iostream>

#include "B1.hpp"

using namespace std;

class D1:virtual public B1{

private:

int d1;

public:

D1(int b1,int d1);

~D1();

void showD1();

};

#endif /\* D1\_hpp \*/

D2.cpp

#include "D2.hpp"

D2::D2(int b1,int d2) : B1(b1){

this->d2 = d2;

cout << "Работает конструктор D2: параметр = " << d2 << endl;

};

D2::~D2(){

cout << "~D2 сработал" << endl;

};

void D2::showD2(){

cout << "D2 = " << d2 << endl;

showB1();

};

D2.hpp

#ifndef D2\_hpp

#define D2\_hpp

#include <iostream>

#include "B1.hpp"

using namespace std;

class D2:virtual private B1{

private:

int d2;

public:

D2(int b1,int d2);

~D2();

void showD2();

};

#endif /\* D2\_hpp \*/

D4.cpp

#include "D4.hpp"

D4::D4(int b1,int b2,int d1,int d2,int d4) : D2(b1,d2),B2(b2),D1(b1,d1),B1(b1){

this->d4 = d4;

cout << "Работает конструктор D3: параметр = " << d4 << endl;

};

D4::~D4(){

cout << "~D2 сработал" << endl;

};

void D4::showD4(){

cout << "D4 = " << d4 << endl;

showD2();

showD1();

showB2();

};

D4.hpp

#ifndef D4\_hpp

#define D4\_hpp

#include <stdio.h>

#include "D1.hpp"

#include "D2.hpp"

#include "B2.hpp"

#include "B1.hpp"

using namespace std;

class D4:public D1,public D2, private B2{

private:

int d4;

public:

D4(int b1,int b2,int d1,int d2,int d4);

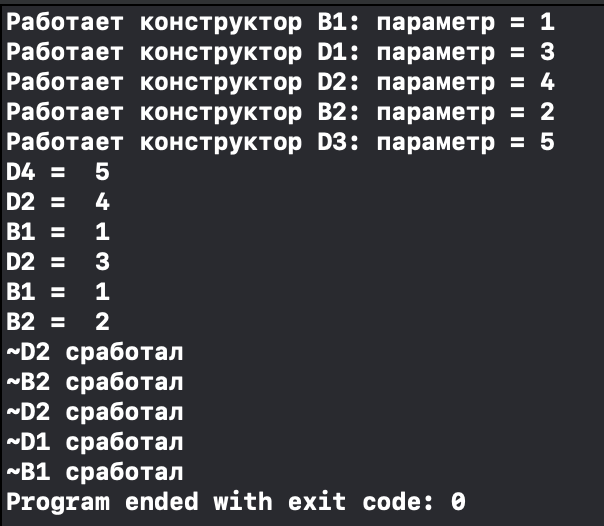
~D4();

void showD4();

};

#endif /\* D4\_hpp \*/

Скриншоты с результатами:



Вывод:

В ходе выполнения лабораторной работы получили практические навыки использования множественного наследования.