МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет  
по лабораторным работам

по дисциплине «Объектно-ориентированное программирование»

Выполнил Проверил

студент группы ГИ-21 ст. преп. каф. «ИУСиТ»

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Оглавление

[Лабораторная работа № 4 Тема: «наследование, простое наследование» 3](#_Toc54952013)

[1.1 Задания 3](#_Toc54952014)

[1.2 Контрольные вопросы 3](#_Toc54952015)

Лабораторная работа № 4  
Тема: «наследование, простое наследование»

### **Цель**

Изучить принципы наследования.

## Задания

### Задание 1.1

### Условие

Написать программу, в которой описана иерархия классов: базовый класс – «ошибка в программе», производные классы - «ошибка доступа к памяти», «математическая», «деление на ноль», «переполнение». Наследники должны иметь поля, содержащие дополнительные сведения об ошибке, если такие имеются. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа объекта для демонстрации.

Программный код

#include <iostream>

using namespace std;

class Error\_programm

{

public:

string error;

Error\_programm()

{

error="ERROR!!!";

}

virtual void vivod()

{

cout <<error<<endl;

}

};

class Error\_memory : public Error\_programm

{ public:

string error\_inf="An attempt was made to free a heap that was not previously allocated.";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

class Error\_math : public Error\_programm

{ public:

string error\_inf= "Sine and cosine range: -1 to 1";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

class Error\_division : public Error\_programm

{ public:

string error\_inf= "ERROR: You cannot divide by zero :(";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

class Error\_overflow : public Error\_programm

{ public:

string error\_inf= "ERROR: Buffer overflow";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

int main()

{

setlocale(LC\_ALL,"");

int key;

cout <<"1.Error\_programm\n2.Error\_memory\n3.Error\_math\n4.Error\_division\n5.Error\_overflow\n0.Exit\n"<<endl;

do

{

cout << "Enter problem number: ";

cin >> key;

switch(key)

{ case 0: cout <<"Have you problem with your head?"<<endl; break;

case 1:

{Error\_programm error1; error1.vivod();

break;}

case 2:

{Error\_memory error2; error2.vivod();

break;}

case 3:

{Error\_math error3; error3.vivod();

break;}

case 4:

{Error\_division error4; error4.vivod();

break;}

case 5:

{Error\_overflow error5; error5.vivod();

break;}

}

}while(key!=0);

getchar();

return 0;

}

### Результат

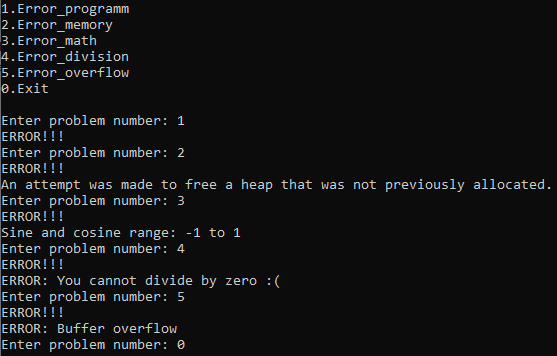


Рисунок 1 – Результат задания 1

## Контрольные вопросы

1. Опишите модификаторы доступа и наследования. Как изменяются атрибуты доступа элементов класса при наследовании?

Атрибут класса (модификатор прав доступа) может задаваться ключевыми словами public и private. Атрибут может опускаться – в этом случае принимается атрибут по умолчанию (для ключевого слова class – private, для struct – public).

|  |  |  |
| --- | --- | --- |
| Ограничения на доступ в базовом классе | Модификатор наследования прав | Ограничения на доступ в производном классе |
| private | private | нет доступа |
| protected | private | private |
| public | private | private |
| private | public | нет доступа |
| protected | public | protected |
| public | public | public |

В производных классах права на доступ к элементам базовых классов могут быть только ограничены, а не расширены.

1. Как работают конструкторы при наследовании?

Конструкторы не наследуются, поэтому производный класс должен иметь собственные конструкторы. Если у базового и произвольного классов имеются конструкторы, то они выполняются в порядке наследования. Если конструктор базового класса требует указания параметров, он должен быть явным образом вызван в конструкторе производного класса в списке инициализации.

1. Как работают деструкторы при наследовании?

Деструкторы не наследуются. Если деструктор не был описан в производном классе, то он формируется по умолчанию и вызывает деструкторы всех базовых классов. В деструкторе производного класса не требуется явно вызывать деструкторы базовых классов, т.к. это будет сделано автоматически. Порядок вызова деструкторов – обратный вызову конструкторов.

1. Какой класс называется производным?

Производным классом называют класс происходящий от базового.

1. Как объявляются производные классы?

ключ\_класса имя\_производного\_класса :

необязательный\_модификатор\_доступа имя\_базового\_класса {

Тело производного класса };

### Вывод

В результате лабораторной работы изучили принципы иерархии классов.