Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: “ЯП”

Тема: “Перегрузка операций. Исключения”

Выполнил:

Студент 2-го курса

Группы ПО-7

Смушко О.Р.

Дата выполнения: 06.10.21

Проверил:

Бойко Д.О.

Брест, 2021

**Вариант 8**

**Цель работы:** Изучение правил перегрузки операций и принципов обработки исключений C++

**Задание:** Написать программу, в которой описана иерархия классов: геометрические фигуры (эллипс, квадрат, трапеция). Описать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором перегрузить операцию [], а также реализовать функцию подсчёта общей площади и периметра. Для базового класса и его потомков перегрузить операции ==, !=, =. Продемонстрировать работу операторов.

В программе используются некоторые основные алгоритмы: ветвление, цикл. Определена иерархия классов согласно заданию.

**Листинг программы:**

**Определение классов:**

**Trapeze.h**

#ifndef LAB\_3\_TRAPEZE\_H  
#define LAB\_3\_TRAPEZE\_H  
#include "Figures.h"  
  
class Trapeze: public Figures{  
 double a, b, c, d, h;  
public:  
 Trapeze(double a, double b, double c, double d, double h);  
 void showSquare() override;  
 void showPerimeter() override;  
 double Read\_a();  
 double Read\_b();  
 double Read\_c();  
 double Read\_d();  
 double Read\_h();  
 bool operator == (const Trapeze& right){ //overloading == operator  
 return square == right.square;  
 }  
 bool operator != (Trapeze \*const right){ //overloading != operator  
 return this != right;  
 }  
 Trapeze &operator = (const Trapeze &right){ //overloading = operator  
 square = right.square;  
 return \*this;  
 }  
};  
#endif //LAB\_3\_TRAPEZE\_H

**Box.h**

#ifndef LAB\_3\_BOX\_H  
#define LAB\_3\_BOX\_H  
#include "Figures.h"  
  
class Box: public Figures{  
 double a;  
public:  
 Box(double a);  
 void showSquare() override;  
 void showPerimeter() override;  
 double Read\_a();  
 bool operator == (const Box& right){ //overloading == operator  
 return square == right.square;  
 }  
 bool operator != (Box \*const right){ //overloading != operator  
 return this != right;  
 }  
 Box &operator = (const Box &right){ //overloading = operator  
 square = right.square;  
 return \*this;  
 }  
};  
#endif //LAB\_3\_BOX\_H

**Ellipse.h**

#ifndef LAB\_3\_ELLIPSE\_H  
#define LAB\_3\_ELLIPSE\_H  
  
#include "Figures.h"  
  
class Ellipse: public Figures{  
 double a, b;  
public:  
 Ellipse(double a, double b);  
 void showSquare() override;  
 void showPerimeter() override;  
 double Read\_a();  
 double Read\_b();  
 bool operator == (const Ellipse& right){ //overloading == operator  
 return square == right.square;  
 }  
 bool operator != (Ellipse \*const right){ //overloading != operator  
 return this != right;  
 }  
 Ellipse &operator = (const Ellipse &right){ //overloading = operator  
 square = right.square;  
 return \*this;  
 }  
};  
  
#endif //LAB\_3\_ELLIPSE\_H

**Figures.h**

#ifndef LAB\_3\_FIGURES\_H  
#define LAB\_3\_FIGURES\_H  
  
#include <iostream>  
#include <locale>  
#define pi 3.14159  
using namespace std;  
  
class Figures{ //abstract class  
protected:  
 double square;  
 double perimeter;  
public:  
 double get\_square() const{ //getting square value  
 return square;  
 }  
 double get\_perimeter() const{ //getting perimeter value  
 return perimeter;  
 }  
 virtual void showSquare() = 0; //pure virtual function for displaying square value  
 virtual void showPerimeter() = 0; //pure virtual function for displaying perimeter value  
};  
  
  
#endif //LAB\_3\_FIGURES\_H

**ArrayFigures.h**

#ifndef LAB\_3\_ARRAYFIGURES\_H  
#define LAB\_3\_ARRAYFIGURES\_H  
#include "Figures.h"  
#include <cassert>  
typedef Figures \*ptr;  
class ArrayFigures{ //collection of objects class  
private:  
 int m\_length;  
 ptr \*m\_data;  
public:  
 ArrayFigures();  
 ArrayFigures(int length);  
 ~ArrayFigures();  
 void erase();  
 void insertBefore(ptr value, int index);  
 void remove(int index);  
 ptr &operator[] (int index){ //overloading [] operator  
 assert(index >= 0 && index < m\_length);  
 return m\_data[index];  
 }  
};  
#endif //LAB\_3\_ARRAYFIGURES\_H

**Trapeze.cpp**

#include "Trapeze.h"  
//constructor with parameters  
Trapeze::Trapeze(double a, double b, double c, double d, double h) {  
 square = (h / 2) \* a \* b;  
 perimeter = a + b + c +d;  
}  
  
double Trapeze::Read\_a() { //entering a value  
 cout << "Enter bigger base: ";  
 cin >> a;  
 return a;  
}  
  
double Trapeze::Read\_b() { //entering b value  
 cout << "Enter smaller base: ";  
 cin >> b;  
 return b;  
}  
  
double Trapeze::Read\_c() { //entering c value  
 cout << "Enter 1st lateral side: ";  
 cin >> c;  
 return c;  
}  
  
double Trapeze::Read\_d() { //entering d value  
 cout << "Enter 2nd lateral side: ";  
 cin >> d;  
 return d;  
}  
  
double Trapeze::Read\_h() { //entering h value  
 cout << "Enter height: ";  
 cin >> h;  
 return h;  
}  
  
void Trapeze::showSquare() { //display square value  
 cout << "Square of trapeze: " << square << endl;  
}  
  
void Trapeze::showPerimeter() { //display perimeter value  
 cout << "Perimeter of trapeze: " << perimeter << endl;  
}

**Box.cpp**

#include "Box.h"  
Box::Box(double a){ //constructor with parameters  
 square = a \* a;  
 perimeter = 4 \* a;  
}  
  
double Box::Read\_a() { //entering a value  
 cout << "Enter side of the square: ";  
 cin >> a;  
 return a;  
}  
  
void Box::showSquare() { //display square value  
 cout << "Square of box: " << square << endl;  
}  
  
void Box::showPerimeter() { //display perimeter value  
 cout << "Perimeter of box: " << perimeter << endl;  
}

**Ellipse.cpp**

#include "Ellipse.h"  
Ellipse::Ellipse(double a, double b) { //constructor with parameters  
 square = pi \* a \* b;  
 perimeter = 4 \* ((pi\*a\*b + (a-b)\*(a-b) / (a+b)));  
}  
  
double Ellipse::Read\_a() { //entering a value  
 cout << "Enter small semi-axis (a): ";  
 cin >> a;  
 return a;  
}  
  
double Ellipse::Read\_b() { //entering b value  
 cout << "Enter big semi-axis (b): ";  
 cin >> b;  
 return b;  
}  
  
void Ellipse::showSquare() { //display square value  
 cout << "Square of ellipse = " << square << endl;  
}  
  
void Ellipse::showPerimeter() { //display perimeter value  
 cout << "Perimeter of ellipse = " << perimeter << endl;  
}

**ArrayFigures.cpp**

#include "ArrayFigures.h"  
ArrayFigures::ArrayFigures() { //constructor without parameters  
 m\_length = 0;  
 m\_data = nullptr;  
}  
  
ArrayFigures::ArrayFigures(int length) { //constructor with parameters  
 m\_length = length;  
 assert(length >= 0);  
 if (length > 0)  
 m\_data = new ptr[length];  
 else  
 m\_data = nullptr;  
}  
  
ArrayFigures::~ArrayFigures() { //destructor  
 delete[] m\_data;  
}  
  
void ArrayFigures::erase() { //function to return the class to its original state  
 delete[] m\_data;  
 m\_data = nullptr;  
 m\_length = 0;  
}  
  
void ArrayFigures::insertBefore(ptr value, int index) {  
 assert(index >= 0 && index <= m\_length); //checking index  
 ptr \*data = new ptr[m\_length+1]; //creating new array  
 for (int before = 0; before < index; ++before) //copying all elements before index  
 data[before] = m\_data[before];  
 data[index] = value; //inserting our new element  
 for (int after = index; after < m\_length; ++after) //copying all elements after inserted element  
 data[after+1] = m\_data[after];  
 delete[] m\_data;  
 m\_data = data;  
 ++m\_length;  
}  
  
void ArrayFigures::remove(int index) {  
 assert(index >= 0 && index < m\_length); //checking index  
 if (m\_length == 1){ //if it's the last element of array  
 erase();  
 return;  
 }  
 ptr \*data = new ptr[m\_length-1]; //creating new array  
 for (int before = 0; before < index; ++before) //copying all elements before index  
 data[before] = m\_data[before];  
 for (int after = index+1; after < m\_length; ++after) //copying all elements after deleted element  
 data[after-1] = m\_data[after];  
 delete[] m\_data;  
 m\_data = data;  
 --m\_length;  
}

**Основная программа:**

**Main.cpp**

#include "Figures.h"  
#include "Ellipse.h"  
#include "Box.h"  
#include "Trapeze.h"  
#include "ArrayFigures.h"  
  
void checkBool (bool value){  
 ///For nice output: True or False  
 if (value == 0)  
 cout << "False" << endl;  
 if (value == 1)  
 cout << "True" << endl;  
}  
  
int main(){  
 setlocale(LC\_ALL, "Russian");  
  
 cout << "Написать программу, в которой описана иерархия классов: геометрические фигуры (эллипс, квадрат, трапеция).\n"  
 "Описать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором перегрузить \n"  
 "операцию [], а также реализовать функцию подсчёта общей площади и периметра. Для базового класса и его \n"  
 "потомков перегрузить операции ==, !=, =. Продемонстрировать работу операторов.\n" << endl;  
  
 //checking overloaded operators for class Ellipse objects  
 cout << "\n==================\n" << "Class Ellipse" << "\n==================\n" << endl;  
 Ellipse ellipse1 = Ellipse(ellipse1.Read\_a(), ellipse1.Read\_b());  
 ellipse1.showSquare();  
 Ellipse ellipse2 = Ellipse(ellipse2.Read\_a(), ellipse2.Read\_b());  
 ellipse2.showSquare();  
 double check\_value1 = 6.28318;  
 double check\_value2 = 11;  
 bool check\_ell = ellipse1.get\_square() == check\_value1;  
 bool check2\_ell = ellipse1.get\_square() != check\_value2;  
 cout << ellipse1.get\_square() << " == " << check\_value1 << " : ";  
 checkBool(check\_ell);  
 cout << ellipse1.get\_square() << " != " << check\_value2 << " : ";  
 checkBool(check2\_ell);  
 cout << "Operator = " << endl;  
 ellipse1.operator=(ellipse2);  
 ellipse1.showSquare();  
  
 //checking overloaded operators for class Box objects  
 cout << "\n==================\n" << "Class Box" << "\n==================\n" << endl;  
 Box box1 = Box(box1.Read\_a());  
 box1.showSquare();  
 double check\_value3 = 25;  
 bool check\_box = box1.get\_square() == check\_value3;  
 cout << box1.get\_square() << " == " << check\_value3 << " : ";  
 checkBool(check\_box);  
  
 //checking overloaded operators for class trapeze objects  
 cout << "\n==================\n" << "Class Trapeze" << "\n==================\n" << endl;  
 Trapeze trap1 = Trapeze(trap1.Read\_a(), trap1.Read\_b(), trap1.Read\_c(), trap1.Read\_d(), trap1.Read\_h());  
 trap1.showSquare();  
 trap1.showPerimeter();  
 double check\_value4 = 12;  
 double check\_value5 = 4;  
 bool check\_trap = trap1.get\_square() != check\_value4;  
 bool check\_trap\_per = trap1.get\_perimeter() == check\_value5;  
 cout << trap1.get\_square() << " != " << check\_value4 << " : ";  
 checkBool(check\_trap);  
 cout << trap1.get\_perimeter() << " == " << check\_value5 << " : ";  
 checkBool(check\_trap\_per);  
  
 //testing array of different classes objects  
 cout << "\n==================\n" << "Testing Array" << "\n==================\n" << endl;  
 Ellipse ellipse = Ellipse(1, 2);  
 Box box = Box(5);  
 Trapeze trap = Trapeze(2, 5, 11, 6, 4);  
  
 ArrayFigures array(3);  
 Figures \*pEllipse = &ellipse;  
 Figures \*pBox = &box;  
 Figures \*pTrapeze = &trap;  
 array[0] = pEllipse;  
 array[1] = pBox;  
 array.insertBefore(pTrapeze, 1); //array[1]  
 array.remove(1);  
  
 for (int i = 0; i < 3; i++) {  
 array[i]->showSquare();  
 array[i]->showPerimeter();  
 }  
 array.erase();  
  
 return 0;  
}

**Вывод:** В ходе лабораторной работы научился перегружать операторы, реализовал иерархию классов, класс коллекций фигур (объектов других классов), а также научился “ловить” исключения.