Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине: «Языки программирования»

Тема: «Перегрузка операций. Исключения»

Выполнила:

Студентка 2 курса

Группа ПО-7

Фурсевич Д.С.

Проверил:

Бойко Д. О.

2021

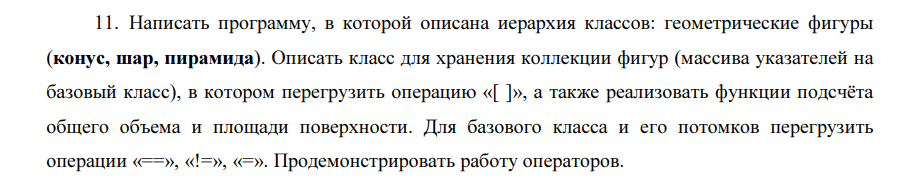
Лабораторная работа №6

**Цель**: изучение правил перегрузки операций и принципов обработки исключений в С++.

**Ход работы**

**Вариант 11**

**Задание:**



**Код программы:**

#define M\_PI 3.14159265358979323846

#include <iostream>

#include <cmath>

#include <windows.h>

using namespace std;

class GeometricShapes {

public:

virtual void Print() = 0;

virtual void Read() = 0;

virtual double Square() = 0;

virtual double Volume() = 0;

};

class IndexError {

protected:

char \*msg;

public:

IndexError() {};

IndexError(const char \*msg) {

this->msg = \_strdup(msg); //дублирование строк с выделением памяти под новую строку

}

char \*get\_msg() {

return msg;

}

~IndexError() {

delete this->msg;

};

};

class CGeometricShapes {

private:

GeometricShapes \*\*arr;

int count;

public:

CGeometricShapes(int count) {

arr = new GeometricShapes \*[count];

for (int i = 0; i < count; i++) {

arr[i] = NULL;

}

this->count = count;

};

~CGeometricShapes() {

for (int i = 0; i < count; i++) {

if (arr[i] != NULL)

delete arr[i];

}

delete[] arr;

};

GeometricShapes\* operator [] (int n) const {

if (n < 0 || n >= count)

{

throw IndexError("Выход за гарницы массива.");

}

return arr[n];

};

GeometricShapes\*& operator [] (int n) {

if (n < 0 || n >= count)

{

throw IndexError("Выход за гарницы массива.");

}

return arr[n];

}

int get\_count() {

return count;

}

void addToEnd(GeometricShapes\* new\_arr) { //добавление в конец списка

GeometricShapes \*\*temp = new GeometricShapes \*[count + 1];

for (int i = 0; i < count; i++) {

temp[i] = arr[i];

}

temp[count] = new\_arr;

arr = temp;

count++;

}

void add(int index, GeometricShapes\* new\_arr) //добавление по заданному индексу

{

if (index<0 || index >count) {

addToEnd(new\_arr);

throw IndexError("Выход за границы массива. Фигура будет добавлена в конец массива");

}

GeometricShapes \*\*temp = new GeometricShapes \*[count + 1];

for (int i = 0; i < index; i++) {

temp[i] = arr[i];

}

temp[index] = new\_arr;

for (int i = index; i < count; i++) {

temp[i + 1] = arr[i];

}

arr = temp;

count++;

}

void deleteFromEnd() { //удаление последнего элемента

GeometricShapes \*\*temp = new GeometricShapes \*[count - 1];

for (int i = 0; i < count - 1; i++) {

temp[i] = arr[i];

}

arr = temp;

count--;

}

void delIndex(int index) {//удаление по заданному индексу

GeometricShapes \*\*temp = new GeometricShapes \*[count - 1];

for (int i = 0; i < index; i++) {

temp[i] = arr[i];

}

for (int i = index + 1; i < count; i++) {

temp[i - 1] = arr[i];

}

arr = temp;

count--;

}

};

class Cone : public GeometricShapes {

private:

int h, r, l;

public:

Cone() {};

Cone(int h, int r, int l) {

this->h = h;

this->r = r;

this->l = l;

}

Cone(const Cone &other) {

cout << "Констурктор копирования класса Cone" << endl;

this->h = other.h;

};

~Cone() {

};

void Print() {

cout << endl << "Конус: " << "R=" << r << " H=" << h << " L=" << l << endl;

cout << "Площадь: " << " S=" << Square() << endl;

cout << "Объём: " << " V=" << Volume() << endl;

};

void Read() {

cout << "Введите радиус основы конуса: ";

cin >> r;

cout << "Введите высоту конуса: ";

cin >> h;

cout << "Введите длину образующей конуса: ";

cin >> l;

cout << endl;

}

double Square() {

return (M\_PI\*r\*l + M\_PI \* r\*r);

};

double Volume() {

return 1/3\*(M\_PI\*h\*r\*r);

};

Cone & operator = (const Cone &other) { //оператор присваивания

this->h = other.h;

this->r = other.r;

this->l = other.l;

return \*this;

};

bool operator == (const Cone &other) { //оператор сравнения

return ( (this->h == other.h) && (this->r == other.r) && (this->l == other.l));

};

bool operator != (const Cone &other) { //оператор сравнения

return !((this->h == other.h) && (this->r ==other.r) && (this->l == other.l));

};

};

class Ball : public GeometricShapes {

private:

int r;

public:

Ball() {};

Ball(int r) {

this->r = r;

};

Ball(const Ball &other) {

this->r = other.r;

};

~Ball() {

};

void Print() {

cout << endl << "Шар: " << " R=" << r << endl;

cout << "Площадь: " << " S=" << Square() << endl;

cout << "Объём: " << " V=" << Volume() << endl;

};

void Read() {

cout << "Введите радиус шара: ";

cin >> r;

cout << endl;

}

double Square() {

return 4\*M\_PI\*r\*r;

};

double Volume() {

return 4/3\*M\_PI\*r\*r\*r;

};

Ball & operator = (const Ball &other) { //оператор присваивания

this->r = other.r;

return \*this;

};

bool operator == (const Ball &other) { //оператор сравнения

return this->r == other.r;

};

bool operator != (const Ball &other) { //оператор сравнения

return !(this->r == other.r);

};

};

class Pyramid : public GeometricShapes {

private:

int a, h;

public:

Pyramid() {};

Pyramid(int a, int h) {

this->a = a;

this->h = h;

};

Pyramid(const Pyramid &other) {

this->a = other.a;

this->h = other.h;

};

~Pyramid() {

};

void Print() {

cout << "Правильная треугольная пирамида: " << " H=" << this->h << " a=" << this->a << endl;

cout << "Площадь: " << " S=" << Square() << endl;

cout << "Объём: " << " V=" << Volume() << endl;

};

void Read() {

cout << "Введите высоту пирамиды: ";

cin >> h;

cout << "Введите сторону основания пирамиды: ";

cin >> a;

cout << endl;

}

double Square() {

return a/4\*(a\*sqrt(3) + 6\*h);

};

double Volume() {

return (a\*a\*sqrt(3)\*h)/12;

};

Pyramid & operator = (const Pyramid &other) { //оператор присваивания

this->a = other.a;

this->h = other.h;

return \*this;

};

bool operator == (const Pyramid &other) { //оператор сравнения

return (this->a == other.a) && (this->h == other.h);

};

bool operator != (const Pyramid &other) { //оператор сравнения

return !((this->a == other.a) && (this->h == other.h));

};

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Задание. Вариант 11." << endl

<< "Написать программу, в которой описана иерархия классов: геометрические фигуры (конус, шар, пирамида)."

<< endl << "Описать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором пергрузить операцию [ ],"

<< endl << "а также реализовать функцию подсчета общей площади и объёма. Для базового класса и его потомков перегрузить"

<< endl << "операции ==, !=, =. Продеманстрировать работу операторов." << endl;

GeometricShapes\* gs = NULL;

CGeometricShapes gsList(0);

int n;

bool ex = false;

while (!ex)

{

cout << "1 - добавить конус" << endl

<< "2 - добавить шар " << endl

<< "3 - добавить пирамиду" << endl

<< "4 - сравнить конусы" << endl

<< "5 - вывести фигуру по индексу" << endl

<< "6 - вывести все фигуры" << endl

<< "7 - удалить последний элемент" << endl

<< "8 - удалить элемент по индексу" << endl

<< "0 - Exit" << endl;

try

{

cin >> n;

switch (n) {

case 1:

{

Cone \*cone1 = new Cone();

cone1->Read();

int k;

bool t = false;

cout << "Добавить в конец(0) или по индексу(1)?";

cin >> t;

if (t) {

int index;

cout << "Введите индекс: ";

cin >> index;

gsList.add(index, cone1);

}

else {

gsList.addToEnd(cone1);

}

break;

}

case 2:

{

Ball \*ball1 = new Ball();

ball1->Read();

bool t = false;

cout << "Добавить в конец(0) или по индексу(1)?";

cin >> t;

if (t) {

int index;

cout << "Введите индекс: ";

cin >> index;

gsList.add(index, ball1);

}

else {

gsList.addToEnd(ball1);

}

break;

}

case 3:

{

Pyramid \*pyramid1 = new Pyramid();

pyramid1->Read();

bool t = false;

cout << "Добавить в конец(0) или по индексу(1)?";

cin >> t;

if (t) {

int index;

cout << "Введите индекс: ";

cin >> index;

gsList.add(index, pyramid1);

}

else {

gsList.addToEnd(pyramid1);

}

break;

}

case 4:

{

Cone objekt1;

objekt1.Read();

Cone objekt2;

objekt2.Read();

if (objekt1 == objekt1)

cout << "Конусы равны\n";

if (objekt1 != objekt1)

cout << "Конусы неравны\n";

break;

}

case 5:

{

int ind;

cout << "Введите индекс элемента: ";

cin >> ind;

if (gsList[ind] != NULL)

gsList[ind]->Print();

break;

}

case 6:

{

cout << "Количество фигур = " << gsList.get\_count() << endl;

for (int i = 0; i < gsList.get\_count(); i++) {

gsList[i]->Print();

}

break;

}

case 7:

{

gsList.deleteFromEnd();

break;

}

case 8:

{

int index;

cout <<"Введите индекс: ";

cin >> index;

gsList.delIndex(index);

}

case 0:

{

ex = true;

}

}

}

catch (IndexError &e)

{

cout << "Ошибка индекса: " << e.get\_msg() << endl;

}

catch (...)

{

cout << "Неизвестная ошибка." << endl;

}

}

}

**Вывод:** изучила правила перегрузки операций и принципов обработки исключений в С++.