Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине: “Языки программирования”

Тема: «Перегрузка операций. Исключения»

**Вариант №10**

**Выполнил**:

студент 2 курса

группы ПО-7

Курмыса Е.Е.

**Проверила:**

Дряпко А. В.

Брест 2021

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

**Ход работы**

В начале программы вывести задание; в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Иерархию классов следует взять из лабораторной работы No3. После работы программы вся динамически выделенная память должна быть освобождена. Класс коллекция должна иметь методы для изменения количества хранимых объектов: добавление в конец, вставка, усечение, удаление из середины. При обращении к элементам с несуществующим индексом или при некорректном изменении количества должно выбрасываться исключение. Взаимодействие с пользователем организовать в виде простого меню, обеспечивающего возможность переопределения исходных данных и завершение работы программы.

*Написать программу, в которой описана иерархия классов: геометрические фигуры (куб, цилиндр, тетраэдр). Описать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором перегрузить операцию «[ ]», а также реализовать функции подсчёта общего объема и площади поверхности. Для базового класса и его потомков перегрузить операции «==», «!=», «=». Продемонстрировать работу операторов.*

**Иерархия классов-фигур**

**Figure**

Tetrahedron

Cube

Cylinder

**Код**

main.cpp — главная часть программы, в которой демонстрируются все указанные возможности:

#include <iostream>

#include <vector>

#include "figures.h"

#include "array.h"

using namespace std;

int Figure::figures = 0;

int Cube::cubes = 0;

int Cylinder::cylinders = 0;

int Tetrahedron::tetrahedrons = 0;

int main() {

cout << "Написать программу, в которой описана иерархия классов: геометрические фигуры (куб, цилиндр, тетраэдр). Описать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором перегрузить операцию «[ ]», а также реализовать функции подсчёта общего объема и площади поверхности. Для базового класса и его потомков перегрузить операции «==», «!=», «=». Продемонстрировать работу операторов.\n\n";

Cube f1;

Cube f2(10);

Cylinder f3;

Cylinder f4(20, 1.5);

Tetrahedron f5;

Tetrahedron f6(10);

cout << "Операция ==\n";

cout << "Фигуры f1 и f1" << (f1 == f1 ? " " : " не ") << "равны\n";

cout << "Фигуры f1 и f6" << (f1 == f6 ? " " : " не ") << "равны\n";

cout << "\nОперация !=\n";

cout << "Фигуры f5 и f2" << (f5 != f2 ? " не " : " ") << "равны\n";

cout << "Фигуры f6 и f6" << (f6 != f6 ? " не " : " ") << "равны\n";

Array figures;

figures.push\_back(&f1);

figures.push\_back(&f2);

figures.push\_back(&f3);

figures.push\_back(&f4);

figures.push\_back(&f5);

figures.push\_back(&f6);

figures.show\_arr();

Array arr;

while (true) {

cout << "0 - Выход из программы\n1 - Создание фигуры и добавление её в массив\n2 - Добавление существующей фигуры в конец массива\n3 - Вставка существующей фигуры внутрь массива\n4 - Усечение массива до некоторого интервала\n5 - Удаление среднего элемента массива\n6 - Вывести значения всех элементов\n";

int choice;

cin >> choice;

if (choice == 0) {

cout << "Производится выход из программы...\n";

break;

} else if (choice == 1) {

cout << "0 - Выход из функции\n1 - Создать куб\n2 - Создать цилиндр\n3 - Создать тетраэдр\n";

int choice\_2;

cin >> choice\_2;

if (choice\_2 == 0) {

cout << "Производится выход из функции...\n";

continue;

} else if (choice\_2 == 1) {

cout << "Введите длину ребра куба: ";

int a;

cin >> a;

arr.push\_back(new Cube(a));

} else if (choice\_2 == 2) {

cout << "Введите длину высоты цилиндра: ";

int a;

cin >> a;

cout << "Введите длину радиуса основания цилиндра: ";

int r;

cin >> r;

arr.push\_back(new Cylinder(a, r));

} else if (choice\_2 == 3) {

cout << "Введите длину ребра тетраэдра: ";

int a;

cin >> a;

arr.push\_back(new Tetrahedron(a));

} else {

cout << "Некорректное значение!\n";

}

} else if (choice == 2) {

cout << "Введите индекс фигуры (от 0 до 5), которую Вы хотите поместить в массив фигур: ";

int index;

cin >> index;

arr.push\_back(figures[index]);

} else if (choice == 3) {

if (arr.len() == 0) {

cout << "Массив пуст!\n";

}

else {

cout << "Введите индекс фигуры (от 0 до 5), которую Вы хотите поместить в массив фигур: ";

int index;

cin >> index;

cout << "Введите позицию, после которой должна быть размещена данная фигура (от 0 до " << arr.len() - 1 << "): ";

int index\_after;

cin >> index\_after;

arr.insert(figures[index], index\_after);

}

} else if (choice == 4) {

if (arr.len() == 0) {

cout << "Массив пуст!\n";

}

else {

cout << "Введите через пробел начальную и конечную точку интервала от 0 до " << arr.len() - 1 << ": ";

int begin, end;

cin >> begin >> end;

arr.trunc(begin, end);

}

} else if (choice == 5) {

arr.mid\_del();

} else if (choice == 6) {

cout << "Список элементов массива:\n";

arr.show\_arr();

}

}

}

figures.h — объявление класса Figure и соответствующих подклассов с методами:

#ifndef FIGURES\_H

#define FIGURES\_H

using namespace std;

class Figure { // абстрактный класс фигур

protected:

float a; // длина ребра фигуры (либо высота цилиндра,

// потому что почему бы и нет =/)

float r; // длина радиуса окружности основания (только для цилиндра)

string name; // название фигуры

static int figures; // количество фигур

public:

Figure();

Figure(float, float, string);

virtual ~Figure();

virtual const float V() const = 0; // чистая виртуальная функция объёма фигуры

virtual const float S() const = 0; // чистая виртуальная функция площади

// поверхности фигуры

virtual const void info() const = 0; // чистая виртуальная функция вывода

// информации о фигуре

bool operator ==(const Figure&); // перегрузка оператора

// сравнения

bool operator !=(const Figure&); // перегрузка оператора

// антисравнения =P

const Figure& operator =(const Figure&); // перегрузка оператора

// присваивания

};

class Cube: public Figure { // класс Куб

protected:

static int cubes; // количество кубов

public:

Cube();

Cube(float);

~Cube();

const float V() const override;

const float S() const override;

const void info() const override;

};

class Cylinder: public Figure { // класс Цилиндр

protected:

static int cylinders; // количество цилиндров

public:

Cylinder();

Cylinder(float, float);

~Cylinder();

const float V() const override;

const float S() const override;

const void info() const override;

};

class Tetrahedron: public Figure { // класс Тетраэдр (правильный)

protected:

static int tetrahedrons; // количество тетраэдров

public:

Tetrahedron();

Tetrahedron(float);

~Tetrahedron();

const float V() const override;

const float S() const override;

const void info() const override;

};

#endif

figures.cpp — определение методов класса Figure:

#include <iostream>

#include <cmath>

#include "figures.h"

using namespace std;

Figure::Figure() {

a = 0;

r = 0;

name = "Фигура";

cout << "Создан объект класса Figure номер " << ++figures << " через конструктор по умолчанию.\n";

}

Figure::Figure(float a, float r, string name): a(a), r(r), name(name) {

cout << "Создан объект класса Figure номер " << ++figures << " через конструктор с параметрами.\n";

}

Figure::~Figure() {

cout << "Объект класса Figure номер " << figures-- << " успешно уничтожен.\n";

}

bool

Figure::operator ==(const Figure& right) {

return (this->name == right.name) && (this->a == right.a) && (this->r == right.r);

}

bool

Figure::operator !=(const Figure& right) {

return !(\*this == right);

}

const Figure&

Figure::operator =(const Figure& right) {

a = right.a;

r = right.a;

name = right.name;

return \*this;

}

Cube::Cube(): Figure(0, 0, "Куб") {

cout << "Создан объект класса Cube номер " << ++cubes << " через конструктор по умолчанию.\n";

}

Cube::Cube(float a): Figure(a, 0, "Куб") {

cout << "Создан объект класса Cube номер " << ++cubes << " через конструктор с параметрами.\n";

}

Cube::~Cube() {

cout << "Объект класса Cube номер " << cubes-- << " успешно уничтожен.\n";

}

const float

Cube::V() const {

return a \* a \* a;

}

const float

Cube::S() const {

return 6 \* a \* a;

}

const void

Cube::info() const {

cout << "Название: " << this->name << endl;

cout << "Длина ребра a: " << this->a << endl;

cout << "Объём V: " << this->V() << endl;

cout << "Площадь поверхности S: " << this->S() << endl << endl;

}

Cylinder::Cylinder(): Figure(0, 0, "Цилиндр") {

cout << "Создан объект класса Cylinder номер " << ++cylinders << " через конструктор по умолчанию.\n";

}

Cylinder::Cylinder(float a, float r): Figure(a, r, "Цилиндр") {

cout << "Создан объект класса Cylinder номер " << ++cylinders << " через конструктор с параметрами.\n";

}

Cylinder::~Cylinder() {

cout << "Объект класса Cylinder номер " << cylinders-- << " успешно уничтожен.\n";

}

const float

Cylinder::V() const {

return a \* r \* r \* (atan(1) \* 4); // выражение в скобках = Pi :)

}

const float

Cylinder::S() const {

return 2 \* a \* (atan(1) \* 4) \* r;

}

const void

Cylinder::info() const {

cout << "Название: " << this->name << endl;

cout << "Высота a: " << this->a << endl;

cout << "Радиус основания r: " << this->r << endl;

cout << "Объём V: " << this->V() << endl;

cout << "Площадь поверхности S: " << this->S() << endl << endl;

}

Tetrahedron::Tetrahedron(): Figure(0, 0, "Тетраэдр") {

cout << "Создан объект класса Tetrahedron номер " << ++tetrahedrons << " через конструктор по умолчанию.\n";

}

Tetrahedron::Tetrahedron(float a): Figure(a, 0, "Тетраэдр") {

cout << "Создан объект класса Tetrahedron номер " << ++tetrahedrons << " через конструктор с параметрами.\n";

}

Tetrahedron::~Tetrahedron() {

cout << "Объект класса Tetrahedron номер " << tetrahedrons-- << " успешно уничтожен.\n";

}

const float

Tetrahedron::V() const {

return a \* a \* a / (sqrt(2) \* 6);

}

const float

Tetrahedron::S() const {

return a \* a \* sqrt(3);

}

const void

Tetrahedron::info() const {

cout << "Название: " << this->name << endl;

cout << "Длина ребра a: " << this->a << endl;

cout << "Объём V: " << this->V() << endl;

cout << "Площадь поверхности S: " << this->S() << endl << endl;

}

array.h — объявление класса Array – массива фигур:

#ifndef ARRAY\_H

#define ARRAY\_H

class Array {

protected:

const Figure \*\*arr; // массив указателей на фигуры

int count; // количество элементов

public:

Array(); // конструктор пустого массива

~Array(); // деструктор

int len() const; // функция олучения длины массива

void show\_arr(); // вывод информации об элементах

void push\_back(const Figure\*); // вставка в конец

void insert(const Figure\*, int); // вставка

void trunc(int, int); // усечение до интервала [begin, end]

void mid\_del(); // удаление из середины

const Figure\* operator [](int); // оператор "[]"

};

#endif

array.cpp — определение методов класса Array:

#include <iostream>

#include <cmath>

#include "figures.h"

#include "array.h"

using namespace std;

Array::Array(): arr(nullptr), count(0) {

cout << "Создан массив фигур через конструктор.\n";

}

Array::~Array() {

delete[] arr;

cout << "Уничтожен массив фигур через деструктор.\n";

}

int

Array::len() const {

return count;

}

void

Array::show\_arr() {

if (count == 0) {

cout << "Массив пуст!\n";

} else {

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

arr[i]->info();

}

}

}

void

Array::push\_back(const Figure\* figure) {

if (figure != NULL) {

if (count == 0) {

count = 1;

arr = new const Figure\* [count];

arr[0] = figure;

cout << "Фигура успешно помещена внутрь!\n";

} else {

count++;

auto new\_arr = new const Figure\* [count];

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

new\_arr[i] = arr[i];

}

new\_arr[count - 1] = figure;

arr = new\_arr;

cout << "Фигура успешно помещена внутрь!\n";

}

}

}

void

Array::insert(const Figure\* figure, int index) {

if (figure != NULL) {

try {

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

throw "Выход за пределы массива!\n";

}

count++;

auto new\_arr = new const Figure\* [count];

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

new\_arr[i] = arr[i];

}

new\_arr[index + 1] = figure;

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

new\_arr[i + 1] = arr[i];

}

arr = new\_arr;

cout << "Фигура успешно помещена внутрь!\n";

}

catch (char const\* s) {

cout << s << endl;

}

}

}

void

Array::trunc(int begin, int end) {

try {

if (begin > end) {

throw "Начало интервала должно быть не больше конца интервала!\n";

}

if (begin < 0 || end >= count) {

throw "Выход за пределы массива!\n";

}

count = end - begin + 1;

auto new\_arr = new const Figure\* [count];

for (int i = begin; i < end + 1; i++) {

new\_arr[i - begin] = arr[i];

}

arr = new\_arr;

cout << "Массив успешно урезан!\n";

}

catch (char const\* s) {

cout << s << endl;

}

}

void

Array::mid\_del() {

try {

if (count == 0) {

throw "Массив пуст!\n";

}

int del\_index = count / 2;

count--;

auto new\_arr = new const Figure\* [count];

for (int i = 0; i < del\_index; i++) new\_arr[i] = arr[i];

for (int i = del\_index + 1; i < count + 1; i++) new\_arr[i - 1] = arr[i];

arr = new\_arr;

cout << "Средний элемент массива успешно вырезан!\n";

}

catch (char const\* s) {

cout << s << endl;

}

}

const Figure\*

Array::operator [](int index) {

try {

if (index >= count || index < -count) {

throw "Выход за пределы массива!\n";

}

return (index >= 0) ? arr[index] : arr[count + index];

}

catch (char const\* s) {

cout << s << endl;

}

}

**Вывод:**

Написать программу, в которой описана иерархия классов: геометрические фигуры (куб, цилиндр, тетраэдр). Описать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором перегрузить операцию «[ ]», а также реализовать функции подсчёта общего объема и площади поверхности. Для базового класса и его потомков перегрузить операции «==», «!=», «=». Продемонстрировать работу операторов.

Создан объект класса Figure номер 1 через конструктор с параметрами.

Создан объект класса Cube номер 1 через конструктор по умолчанию.

Создан объект класса Figure номер 2 через конструктор с параметрами.

Создан объект класса Cube номер 2 через конструктор с параметрами.

Создан объект класса Figure номер 3 через конструктор с параметрами.

Создан объект класса Cylinder номер 1 через конструктор по умолчанию.

Создан объект класса Figure номер 4 через конструктор с параметрами.

Создан объект класса Cylinder номер 2 через конструктор с параметрами.

Создан объект класса Figure номер 5 через конструктор с параметрами.

Создан объект класса Tetrahedron номер 1 через конструктор по умолчанию.

Создан объект класса Figure номер 6 через конструктор с параметрами.

Создан объект класса Tetrahedron номер 2 через конструктор с параметрами.

Операция ==

Фигуры f1 и f1 равны

Фигуры f1 и f6 не равны

Операция !=

Фигуры f5 и f2 не равны

Фигуры f6 и f6 равны

Создан массив фигур через конструктор.

Фигура успешно помещена внутрь!

Фигура успешно помещена внутрь!

Фигура успешно помещена внутрь!

Фигура успешно помещена внутрь!

Фигура успешно помещена внутрь!

Фигура успешно помещена внутрь!

Создан массив фигур через конструктор.

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

1

0 - Выход из функции

1 - Создать куб

2 - Создать цилиндр

3 - Создать тетраэдр

2

Введите длину высоты цилиндра: 5

Введите длину радиуса основания цилиндра: 9

Создан объект класса Figure номер 7 через конструктор с параметрами.

Создан объект класса Cylinder номер 3 через конструктор с параметрами.

Фигура успешно помещена внутрь!

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

6

Список элементов массива:

Название: Цилиндр

Высота a: 5

Радиус основания r: 9

Объём V: 1272.34

Площадь поверхности S: 282.743

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

3

Введите индекс фигуры (от 0 до 5), которую Вы хотите поместить в массив фигур: 5

Введите позицию, после которой должна быть размещена данная фигура (от 0 до 0): 0

Фигура успешно помещена внутрь!

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

1

0 - Выход из функции

1 - Создать куб

2 - Создать цилиндр

3 - Создать тетраэдр

1

Введите длину ребра куба: 9

Создан объект класса Figure номер 8 через конструктор с параметрами.

Создан объект класса Cube номер 3 через конструктор с параметрами.

Фигура успешно помещена внутрь!

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

2

Введите индекс фигуры (от 0 до 5), которую Вы хотите поместить в массив фигур: 4

Фигура успешно помещена внутрь!

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

6

Список элементов массива:

Название: Цилиндр

Высота a: 5

Радиус основания r: 9

Объём V: 1272.34

Площадь поверхности S: 282.743

Название: Тетраэдр

Длина ребра a: 10

Объём V: 117.851

Площадь поверхности S: 173.205

Название: Куб

Длина ребра a: 9

Объём V: 729

Площадь поверхности S: 486

Название: Тетраэдр

Длина ребра a: 0

Объём V: 0

Площадь поверхности S: 0

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

4

Введите через пробел начальную и конечную точку интервала от 0 до 3: 1 3

Массив успешно урезан!

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

5

Средний элемент массива успешно вырезан!

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

6

Список элементов массива:

Название: Тетраэдр

Длина ребра a: 10

Объём V: 117.851

Площадь поверхности S: 173.205

Название: Тетраэдр

Длина ребра a: 0

Объём V: 0

Площадь поверхности S: 0

0 - Выход из программы

1 - Создание фигуры и добавление её в массив

2 - Добавление существующей фигуры в конец массива

3 - Вставка существующей фигуры внутрь массива

4 - Усечение массива до некоторого интервала

5 - Удаление среднего элемента массива

6 - Вывести значения всех элементов

0

Производится выход из программы...

Уничтожен массив фигур через деструктор.

Уничтожен массив фигур через деструктор.

Объект класса Tetrahedron номер 2 успешно уничтожен.

Объект класса Figure номер 8 успешно уничтожен.

Объект класса Tetrahedron номер 1 успешно уничтожен.

Объект класса Figure номер 7 успешно уничтожен.

Объект класса Cylinder номер 3 успешно уничтожен.

Объект класса Figure номер 6 успешно уничтожен.

Объект класса Cylinder номер 2 успешно уничтожен.

Объект класса Figure номер 5 успешно уничтожен.

Объект класса Cube номер 3 успешно уничтожен.

Объект класса Figure номер 4 успешно уничтожен.

Объект класса Cube номер 2 успешно уничтожен.

Объект класса Figure номер 3 успешно уничтожен.

**Вывод:** я изучил правила перегрузки операций и принципов обработки исключений в C++.