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:**

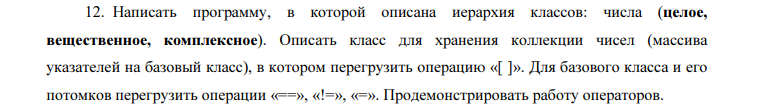
Дряпко А. В.

Брест 2021

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

**Ход работы**

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

**

**Иерархия классов-фигур**

**NUMBER**

Complex

Integer

Real

**Код**

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

#include <iostream>

#include <vector>

#include "number.h"

#include "array.h"

using namespace std;

int Number::numbers = 0;

int Integer::integers = 0;

int Real::reals = 0;

int Complex::complexes = 0;

bool is\_valid(float a, float f = 0) {

return (a == int(a)) && (f == int(f));

}

int main() {

cout << "12 BapuaHT \n\n";

Integer f1;

Integer f2(59);

Real f3;

Real f4(8, 91);

Complex f5;

Complex f6(16, 43, 6);

cout << "OPERATCIA ==\n";

cout << "Number f1 u f1" << (f1 == f1 ? " " : " ne ") << "ravno\n";

cout << "Number f1 u f6" << (f1 == f6 ? " " : " ne ") << "ravno\n";

cout << "\nOperatcia !=\n";

cout << "Number f5 u f2" << (f5 != f2 ? " ne " : " ") << "Ravno\n";

cout << "Number f6 u f6" << (f6 != f6 ? " ne " : " ") << "Ravno\n";

Array numbers;

numbers.push\_back(&f1);

numbers.push\_back(&f2);

numbers.push\_back(&f3);

numbers.push\_back(&f4);

numbers.push\_back(&f5);

numbers.push\_back(&f6);

Array arr;

while (true) {

cout << "0 - Exit\n1 - Sozd i dobavl v massiv\n2 - Dobavl sush number v massiv\n3 - Vstavka sush v massiv\n4 - Usechenie\n5 - Delete\n6 - Vivod vseh elementov\n";

int choice;

cin >> choice;

if (choice == 0) {

cout << "EXIT...\n";

break;

}

else if (choice == 1) {

cout << "0 - Exit iz function\n1 - Cree number full\n2 - Cree VESH number\n3 - Cree complex number\n";

int choice\_2;

cin >> choice\_2;

if (choice\_2 == 0) {

cout << "Exit ix function...\n";

continue;

}

else if (choice\_2 == 1) {

cout << "Vvedite celyu chast: ";

float a;

cin >> a;

if (is\_valid(a)) {

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

}

else {

cout << "celaya chast doljna bit celoy!\n";

}

}

else if (choice\_2 == 2) {

cout << "Vvedite celyu chast: ";

float a;

cin >> a;

cout << "Vvedite vesh chast: ";

float f;

cin >> f;

if (is\_valid(a, f)) {

arr.push\_back(new Real(a, f));

}

else {

cout << "celaya chast i vesh doljna bit celoy!";

}

}

else if (choice\_2 == 3) {

cout << "Vvedite celyu chast: ";

float a;

cin >> a;

cout << "Vvedite vesh chast: ";

float f;

cin >> f;

if (is\_valid(a, f)) {

cout << "Vvedite comlex chast: ";

float i;

cin >> i;

arr.push\_back(new Complex(a, f, i));

}

else {

cout << "celaya chast i vesh doljna bit celoy!!\n";

}

}

else {

cout << "nekkorektnoe znachenie!\n";

}

}

else if (choice == 2) {

cout << "vvedite index (ot 0 do 5), kotoroe hotite dobavit: ";

int index;

cin >> index;

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

}

else if (choice == 3) {

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

cout << "pust!\n";

}

else {

cout << "vvedite index (ot 0 do 5), kotoroe hotite dobavit: ";

int index;

cin >> index;

cout << "vvedite poziciyu posle cotoroy hotite dobavit(ot 0 do " << arr.len() - 1 << "): ";

int index\_after;

cin >> index\_after;

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

}

}

else if (choice == 4) {

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

cout << "Massiv pust!\n";

}

else {

cout << "Vvedite 4erez probel na4alnyu . u kone4nyu ot 0 do " << 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 << "Spisok elemenov:\n";

arr.show\_arr();

}

}

}

NUMBER.h — объявление класса NUMBER и соответствующих подклассов с методами:

#pragma once

#ifndef NUMBER\_H

#define NUMBER\_H

using namespace std;

class Number {

protected:

int a; // целая часть

int f; // вещественная часть

float i; // комплексная часть

string name; // наименование класса числа

static int numbers; //

public:

Number();

Number(int, int, float, string);

virtual ~Number();

virtual const void out() const = 0;

virtual const void info() const = 0;

bool operator ==(const Number&);

bool operator !=(const Number&);

const Number& operator =(const Number&);

};

class Integer : public Number {

protected:

static int integers;

public:

Integer();

Integer(int);

~Integer();

const void out() const override;

const void info() const override;

};

class Real : public Number {

protected:

static int reals;

public:

Real();

Real(int, int);

~Real();

const void out() const override;

const void info() const override;

};

class Complex : public Number {

protected:

static int complexes;

public:

Complex();

Complex(int, int, float);

~Complex();

const void out() const override;

const void info() const override;

};

#endif

NUMBER.cpp — определение методов класса NUMBER:

#include <iostream>

#include <cmath>

#include "number.h"

using namespace std;

Number::Number() {

a = 0;

f = 0;

i = 0;

name = "Chislo";

cout << "Cree obj class Number nomer " << ++numbers << " 4erez constr po ymolch.\n";

}

Number::Number(int a, int f, float i, string name) : a(a), f(f), i(i), name(name) {

cout << "Cree obj class Number nomer " << ++numbers << " 4erez constr s parametrami.\n";

}

Number::~Number() {

cout << "Obj class Number nomer " << numbers-- << " delete.\n";

}

bool

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

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

}

bool

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

return !(\*this == right);

}

const Number&

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

a = right.a;

f = right.f;

i = right.i;

name = right.name;

return \*this;

}

Integer::Integer() : Number(0, 0, 0, "Celoe") {

cout << "Creet jbj class Integer nomer " << ++integers << " 4erez constr po ymolch.\n";

}

Integer::Integer(int a) : Number(a, 0, 0, "Целое") {

cout << "Creet obj class Integer nomer " << ++integers << " 4erez constr s parametrami.\n";

}

Integer::~Integer() {

cout << "Creet obj class Integer nomer " << integers-- << " delete.\n";

}

const void

Integer::out() const {

cout << this->a;

}

const void

Integer::info() const {

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

cout << "Celaya chast: " << this->a << endl;

cout << "Zapis number: "; this->out(); cout << endl << endl;

}

Real::Real() : Number(0, 0, 0, "Veshestvennoe") {

cout << "Creet Obj class REAL nomer " << ++reals << " 4erez constr po ymolch.\n";

}

Real::Real(int a, int f) : Number(a, f, 0, "Veshestvennoe") {

cout << "Creet Obj class REAL nomer " << ++reals << " 4erez constr s parametrami..\n";

}

Real::~Real() {

cout << "OBJ class REAL nomer " << reals-- << " delete.\n";

}

const void

Real::out() const {

cout << this->a << '.' << this->f;

}

const void

Real::info() const {

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

cout << "Celaya chast: " << this->a << endl;

cout << "Veshestvennaya chast: " << this->f << endl;

cout << "Zapis chisla: "; this->out(); cout << endl << endl;

}

Complex::Complex() : Number(0, 0, 0, "Comlex") {

cout << "Creet Obj class Complex nomer " << ++complexes << " 4erez constr po ymolch.\n";

}

Complex::Complex(int a, int f, float i) : Number(a, f, i, "Complex") {

cout << "Creet Obj class Complex nomer " << ++complexes << " 4erez constr s parametrami.\n";

}

Complex::~Complex() {

cout << "Obj class Complex nomer " << complexes-- << " delete.\n";

}

const void

Complex::out() const {

cout << this->a << '.' << this->f << '+' << this->i << 'i';

}

const void

Complex::info() const {

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

cout << "Celaya chast: " << this->a << endl;

cout << "vesh chast: " << this->f << endl;

cout << "complex chast: " << this->i << endl;

cout << "zapis chisla: "; this->out(); cout << endl << endl;

}}

array.h — объявление класса Array–массива чисел:

#pragma once

#ifndef ARRAY\_H

#define ARRAY\_H

class Array {

protected:

const Number\*\* arr;

int count;

public:

Array();

~Array();

int len() const;

void show\_arr();

void push\_back(const Number\*);

void insert(const Number\*, int);

void trunc(int, int);

void mid\_del();

const Number\* operator [](int);

};

#endif

array.cpp — определение методов класса Array:

#include <iostream>

#include <cmath>

#include "number.h"

#include "array.h"

using namespace std;

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

cout << "creet massiv 4erez constr.\n";

}

Array::~Array() {

delete[] arr;

cout << "delete 4erez destr.\n";

}

int

Array::len() const {

return count;

}

void

Array::show\_arr() {

if (count == 0) {

cout << "Pusto!\n";

}

else {

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

arr[i]->info();

}

}

}

void

Array::push\_back(const Number\* number) {

if (number != NULL) {

if (count == 0) {

count = 1;

arr = new const Number \* [count];

arr[0] = number;

cout << "Number vnytri!\n";

}

else {

count++;

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

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

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

}

new\_arr[count - 1] = number;

arr = new\_arr;

cout << "Number vnytri!\n";

}

}

}

void

Array::insert(const Number\* number, int index) {

if (number != NULL) {

try {

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

throw "vihod za predely!\n";

}

count++;

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

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

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

}

new\_arr[index + 1] = number;

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

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

}

arr = new\_arr;

cout << "Number vnytri!\n";

}

catch (char const\* s) {

cout << s << endl;

}

}

}

void

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

try {

if (begin > end) {

throw "Na4alo ne > konca!\n";

}

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

throw "vyhod za predely!\n";

}

count = end - begin + 1;

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

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

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

}

arr = new\_arr;

cout << "Massiv yrezan!\n";

}

catch (char const\* s) {

cout << s << endl;

}

}

void

Array::mid\_del() {

try {

if (count == 0) {

throw "Pusto!\n";

}

int del\_index = count / 2;

count--;

auto new\_arr = new const Number \* [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 << "AVG vyrezan!\n";

}

catch (char const\* s) {

cout << s << endl;

}

}

const Number\*

Array::operator [](int index) {

try {

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

throw "vyhod za predely!\n";

}

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

}

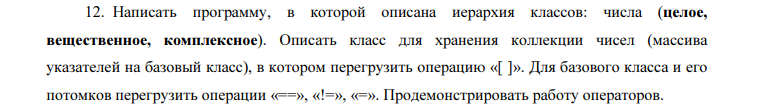
catch (char const\* s) {

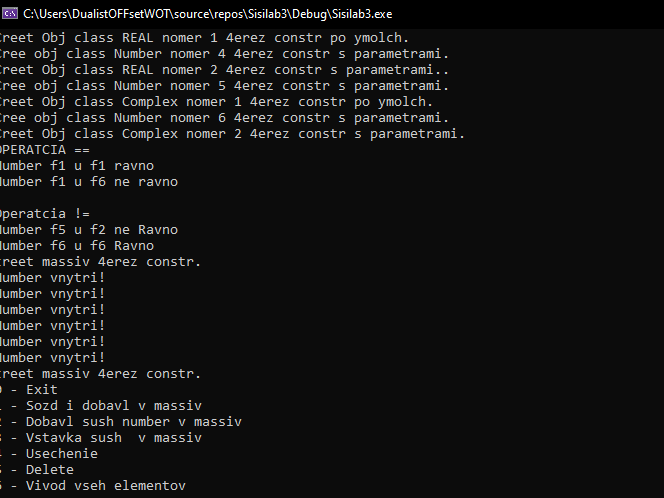
cout << s << endl;

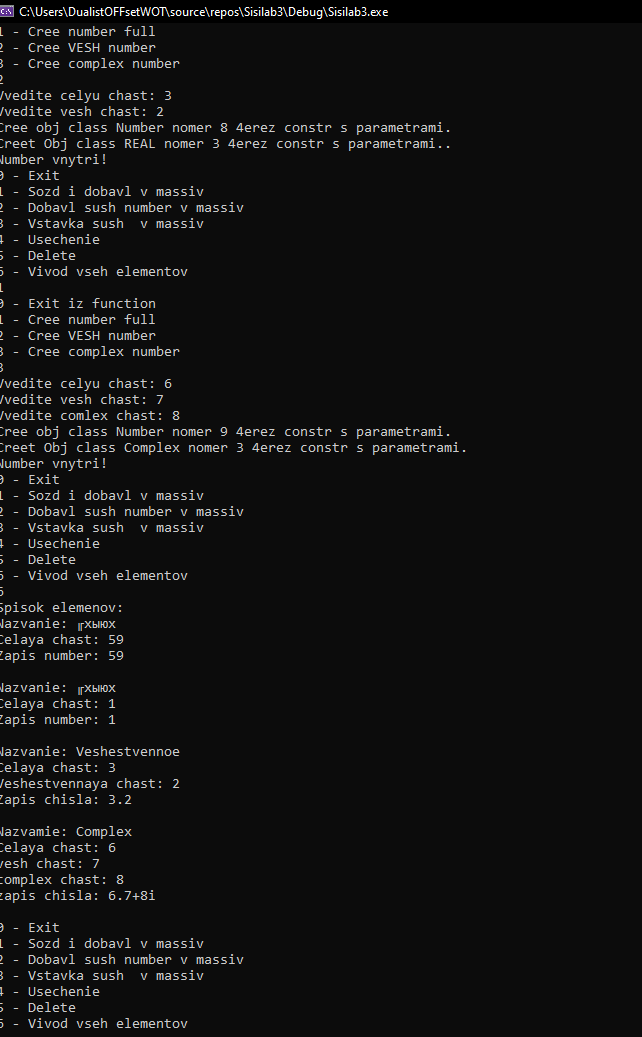
}

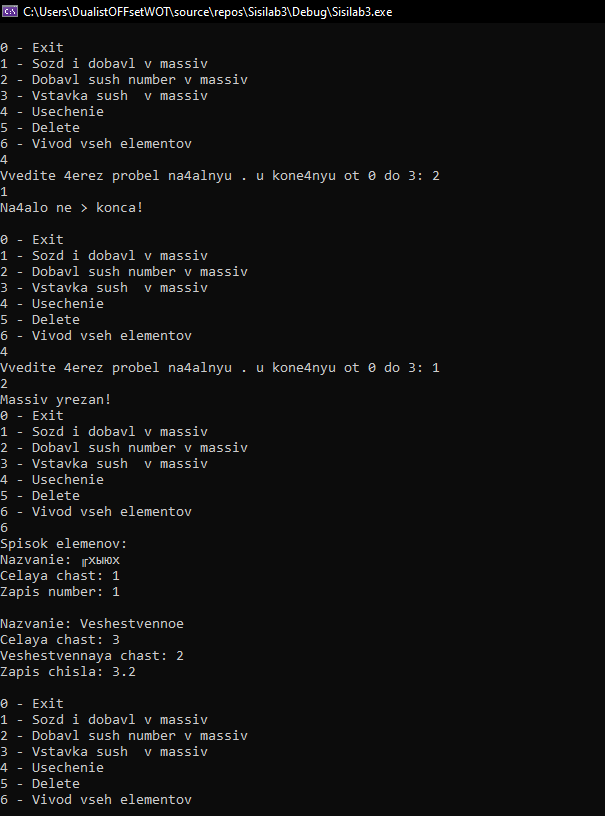
}

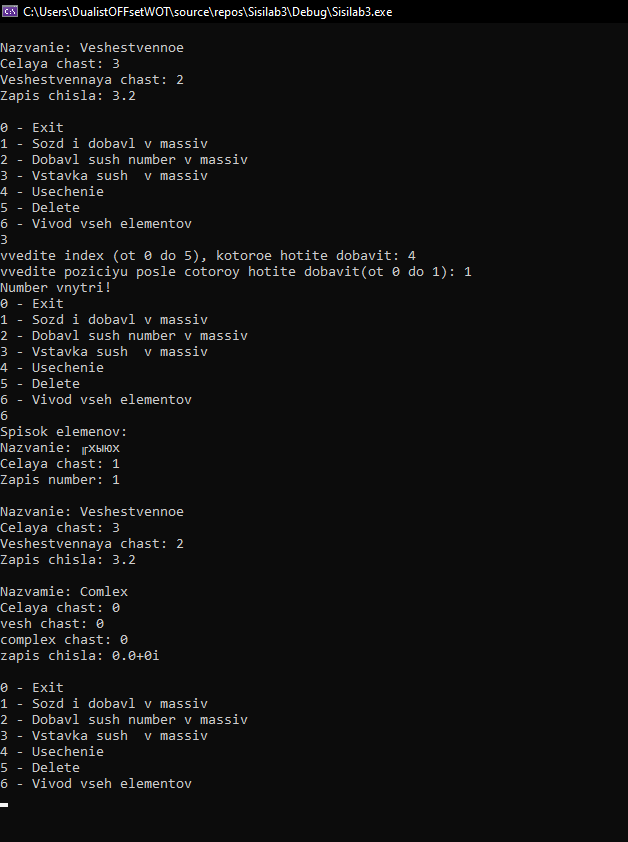
**РЕЗУЛЬТАТЫ ПРОГРАММЫ**







****

****

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