Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.9**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Обработка исключительных ситуаций

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

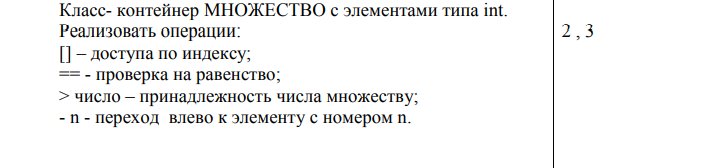
# **Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, обрабатывающей исключительные ситуации.

# **Постановка задачи**

1. Реализовать класс, перегрузить для него операции, указанные в варианте.
2. Определить исключительные ситуации.
3. Предусмотреть генерацию исключительных ситуаций.
4. Написать демонстрационную программу, иллюстрирующую выполнение задачи.

(8 вариант)



# **Анализ задачи**

1. Определить какие действия предстоит выполнить:

* Разработать класс Set с полями data, size, max\_size, begin, end.
* Организовать методы для ввода данных, конструкторы и деструкторы, перегрузки операторов для класса Set.

class Set {

private:

int\* data;

int size;

public:

Set();

Set(Set&);

~Set();

int get\_size() {

return size;

}

Set& operator = (const Set&);

void Insert(int);

int operator [] (int);

bool operator == (const Set&);

bool operator > (int);

int operator - (int);

friend istream& operator >> (istream&, Set&);

friend ostream& operator << (ostream&, const Set&);

};

* Разработать классы ошибок, которые могу возникнуть в результате работы программы в файле Error.

class Error {

public:

virtual void what() { };

};

class IndexError : public Error {

protected:

string type\_error;

public:

IndexError() {

type\_error = "\nОшибка индекса!";

}

void what() {

cout << type\_error;

}

};

class IndexError1 : public IndexError {

protected:

string error\_message;

public:

IndexError1() {

IndexError();

error\_message = "\nПроизошла попытка обращения к элементу с отрицательный индексом\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class IndexError2 : public IndexError {

protected:

string error\_message;

public:

IndexError2() {

IndexError();

error\_message = "\nПроизошла попытка обращения к элементу, индекс которого больше максимального индекса\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class IndexShiftError1 : public IndexError {

protected:

string error\_message;

public:

IndexShiftError1() {

IndexError();

error\_message = "\nПроизошла попытка сдвига индекса на значение большее длины множества\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class IndexShiftError2 : public IndexError {

protected:

string error\_message;

public:

IndexShiftError2() {

IndexError();

error\_message = "\nПроизошла попытка сдвига индекса на отрицательное значение\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class SizeError : public Error {

protected:

string type\_error;

public:

SizeError() {

type\_error = "\nОшибка размера!";

}

void what() {

cout << type\_error;

}

};

class SizeError1 : public SizeError {

protected:

string error\_message;

public:

SizeError1() {

SizeError();

error\_message = "\nПроизошла попытка создания множества, кол-во элементов которого больше максимального кол-ва элементов\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class SizeError2 : public SizeError {

protected:

string error\_message;

public:

SizeError2() {

SizeError();

error\_message = "\nПроизошла попытка создания множества с отрицательным кол-вом элементов\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class EmptySizeError : public SizeError {

protected:

string error\_message;

public:

EmptySizeError() {

SizeError();

error\_message = "\nПроизошла попытка обращения к пустому множеству\n";

}

void what() {

cout << type\_error << error\_message;

}

};

* Протестировать обработку исключений в функции main.

int main() {

system("chcp 1251");

Set A, B;

try {

cout << "\nЗаполнение множеста A:\n"; cin >> A;

}

catch (Error &e) {

e.what();

}

try {

cout << "\nЗаполнение множеста B:\n"; cin >> B;

}

catch (Error &e) {

e.what();

}

cout << "\nСодержимое множества A: " << A;

cout << "\nСодержимое множества B: " << B;

if (A == B) {

cout << "\nМножества равны\n";

}

else {

cout << "\nМножества не равны\n";

}

int value;

try {

cout << "\nПроверка оператора '[]' на множестве А\nВведите индекс:\n> "; cin >> value; cout << A[value - 1] << endl;

}

catch (Error &e) {

e.what();

}

cout << "\nПроверка оператора '>' на множестве А\nВведите элемент:\n> "; cin >> value;

if (A > value) {

cout << "\nЭлемент " << value << " во множестве есть!\n";

}

else {

cout << "\nЭлемента " << value << " во множестве нет!\n";

}

try {

int size; size = A.get\_size();

cout << "\nПроверка оператора '-' на множестве А\nВведите сдвиг:\n> "; cin >> value;

cout << A[size - 1] << endl << A - value << endl;

}

catch (Error &e) {

e.what();

}

system("pause");

return 0;

}

1. C какими типами данных предстояло работать:

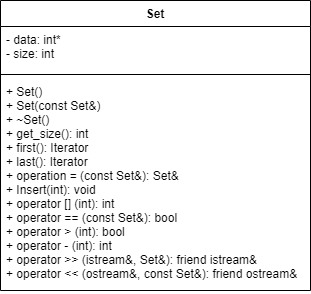
* Для хранения данных реализована переменная data типа int\*.

int\* data;

* Для хранения длины множества реализована переменная size типа int.

int size;

# **UML диаграмма**



# **Код программы на языке C++**

Заголовочный файл Set.h

#pragma once

#include <iostream>

using namespace std;

const int MAX\_SIZE = 20;

class Set {

private:

int\* data;

int size;

public:

Set();

Set(Set&);

~Set();

void Insert(int);

int get\_size();

Set& operator = (const Set&);

int operator [] (int);

bool operator > (int);

bool operator == (const Set&);

int operator - (int);

friend ostream& operator << (ostream&, const Set&);

friend istream& operator >> (istream&, Set&);

};

Заголовочный файл Error.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Error {

public:

virtual void what() { };

};

class IndexError : public Error {

protected:

string type\_error;

public:

IndexError() {

type\_error = "\nОшибка индекса!";

}

void what() {

cout << type\_error;

}

};

class IndexError1 : public IndexError {

protected:

string error\_message;

public:

IndexError1() {

IndexError();

error\_message = "\nПроизошла попытка обращения к элементу с отрицательный индексом\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class IndexError2 : public IndexError {

protected:

string error\_message;

public:

IndexError2() {

IndexError();

error\_message = "\nПроизошла попытка обращения к элементу, индекс которого больше максимального индекса\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class IndexShiftError1 : public IndexError {

protected:

string error\_message;

public:

IndexShiftError1() {

IndexError();

error\_message = "\nПроизошла попытка сдвига индекса на значение большее длины множества\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class IndexShiftError2 : public IndexError {

protected:

string error\_message;

public:

IndexShiftError2() {

IndexError();

error\_message = "\nПроизошла попытка сдвига индекса на отрицательное значение\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class SizeError : public Error {

protected:

string type\_error;

public:

SizeError() {

type\_error = "\nОшибка размера!";

}

void what() {

cout << type\_error;

}

};

class SizeError1 : public SizeError {

protected:

string error\_message;

public:

SizeError1() {

SizeError();

error\_message = "\nПроизошла попытка создания множества, кол-во элементов которого больше максимального кол-ва элементов\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class SizeError2 : public SizeError {

protected:

string error\_message;

public:

SizeError2() {

SizeError();

error\_message = "\nПроизошла попытка создания множества с отрицательным кол-вом элементов\n";

}

void what() {

cout << type\_error << error\_message;

}

};

class EmptySizeError : public SizeError {

protected:

string error\_message;

public:

EmptySizeError() {

SizeError();

error\_message = "\nПроизошла попытка обращения к пустому множеству\n";

}

void what() {

cout << type\_error << error\_message;

}

};

Обьявление класса в Set.cpp

#include "Set.h"

#include "Error.h"

#include <algorithm>

Set::Set() {

size = 0;

data = new int[MAX\_SIZE];

}

Set::Set(Set& temp\_Set) {

size = temp\_Set.size;

delete[] data;

data = new int[MAX\_SIZE];

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

data[i] = temp\_Set.data[i];

}

}

Set::~Set() {

delete[] data;

}

void Set::Insert(int value) {

int i;

for (i = 0; i < size; i++) {

if (data[i] == value) {

i = size;

}

}

if (i != size + 1) {

data[size] = value;

if (i > 0) {

sort(data, data + size + 1);

}

size++;

}

}

int Set::get\_size() {

return size;

}

Set& Set::operator=(const Set& temp\_Set) {

if (\*this == temp\_Set) {

return \*this;

}

size = temp\_Set.size;

delete[] data;

data = new int[MAX\_SIZE];

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

data[i] = temp\_Set.data[i];

}

return \*this;

}

int Set::operator[](int index) {

if (size == 0) {

throw EmptySizeError();

}

if (index < 0) {

throw IndexError1();

}

if (index >= size) {

throw IndexError2();

}

cout << "\nСодержимое ячейки множества с номером " << index+1 << " = ";

return data[index];

}

bool Set::operator>(int value) {

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

if (data[i] == value) {

return true;

}

}

return false;

}

bool Set::operator==(const Set& temp\_Set) {

if (size == temp\_Set.size) {

int i;

for (i = 0; i < size; i++) {

if (data[i] != temp\_Set.data[i]) {

i = size;

}

}

if (i == size) {

return true;

}

}

return false;

}

int Set::operator-(int shift) {

if (size == 0) {

throw EmptySizeError();

}

if (shift >= size) {

throw IndexShiftError1();

}

if (shift < 0) {

throw IndexShiftError2();

}

cout << "\nСодержимое ячейки множества с номером " << size-shift+1 << " = ";

return data[size - shift];

}

ostream& operator << (ostream& out, const Set& temp\_Set) {

out << "\n{ ";

for (int i = 0; i < temp\_Set.size; i++) {

out << temp\_Set.data[i] << " ";

}

out << "}\n";

return out;

}

istream& operator >> (istream& in, Set& temp\_Set) {

int number, quantity;

cout << "\nКол-во элментов - "; in >> quantity;

if (quantity > MAX\_SIZE) {

throw SizeError1();

}

if (quantity < 0) {

throw SizeError2();

}

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

cout << i + 1 << ") ";

in >> number;

temp\_Set.Insert(number);

}

return in;

}

Main файл

#include <iostream>

#include "Set.h"

#include "Error.h"

using namespace std;

int main() {

system("chcp 1251");

Set A, B;

try {

cout << "\nЗаполнение множеста A:\n"; cin >> A;

}

catch (Error &e) {

e.what();

}

try {

cout << "\nЗаполнение множеста B:\n"; cin >> B;

}

catch (Error &e) {

e.what();

}

cout << "\nСодержимое множества A: " << A;

cout << "\nСодержимое множества B: " << B;

if (A == B) {

cout << "\nМножества равны\n";

}

else {

cout << "\nМножества не равны\n";

}

int value;

try {

cout << "\nПроверка оператора '[]' на множестве А\nВведите индекс:\n> "; cin >> value; cout << A[value - 1] << endl;

}

catch (Error &e) {

e.what();

}

cout << "\nПроверка оператора '>' на множестве А\nВведите элемент:\n> "; cin >> value;

if (A > value) {

cout << "\nЭлемент " << value << " во множестве есть!\n";

}

else {

cout << "\nЭлемента " << value << " во множестве нет!\n";

}

try {

int size; size = A.get\_size();

cout << "\nПроверка оператора '-' на множестве А\nВведите сдвиг:\n> "; cin >> value;

cout << A[size - 1] << endl << A - value << endl;

}

catch (Error &e) {

e.what();

}

system("pause");

return 0;

}

# **Скриншоты тестов**

