Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 9

ОСНОВЫ АЛГОРИТМИЗАЦИИ ЗА II СЕМЕСТР

Выполнил студент:

Тимолянов Григорий Констнатинович

Группа РИС-21-1бз

Проверила:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

ПЕРМЬ 2024

# Лабораторная работа №9 "Обработка исключительных ситуаций"

**Вариант №15**

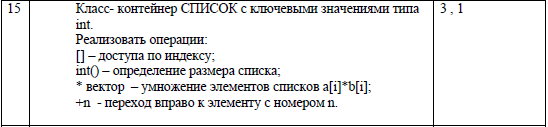
**Цель:**

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Разработка программы, обрабатывающей исключительные ситуации.

## **Задача:**

1. Реализовать класс, перегрузить для него операции, указанные в варианте.
2. Определить исключительные ситуации.
3. Предусмотреть генерацию исключительных ситуаций.



**Решение:**

Файл Исключительные ситуации.cpp (вариант 1)

#include "Listing.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int i;

// Обработка исключительных ситуаций

try {

cout << "Пустой список из двух элементов: \n";

// Список с 2 элементами

Listing x(2);

// Пустой список

Listing y;

cout << x << "\nВведите номер элемента: \n";

cin >> i;

cout << "\nВывод элемента под номером " << i << ":\n";

cout << x[i] << endl;

y = x \* 1;

cout << y;

}

catch(int) {

cout << "Ошибка!\n";

}

return 0;

}

Файл Listing.cpp (вариант реализации 1)

#include "Listing.h"

// Генерация исключения при текущий размер > максимальный размер

Listing::Listing(int s) {

if (s > MAX\_SIZE) { throw 1; }

size = s;

beg = new int[s];

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

beg[i] = 0;

}

}

Listing::Listing(const Listing &l) {

size = l.size;

beg = new int[size];

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

beg[i] = l.beg[i];

}

}

Listing::~Listing() {

if (beg != 0) {

delete[] beg;

}

}

Listing::Listing(int s, int \*mas) {

if (s > MAX\_SIZE) { throw 1; }

size = s;

beg = new int[size];

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

beg[i] = mas[i];

}

}

const Listing& Listing::operator = (const Listing& l) {

if (this == &l) { return \*this; }

if (beg != 0) { delete[]beg; }

size = l.size;

beg = new int[size];

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

beg[i] = l.beg[i];

}

return \*this;

}

ostream& operator << (ostream& out, const Listing& l) {

if (l.size == 0) { out << "Пусто\n"; }

else {

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

out << l.beg[i] << " ";

}

out << endl;

}

return out;

}

istream& operator >> (istream& in, Listing& l) {

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

cout << ">";

in >> l.beg[i];

}

return in;

}

// Если индекс отрицательный, генерируется исключение

int Listing::operator [] (int i) {

if (i < 0) {

throw 2;

}

}

Listing Listing::operator\*(int a) {

Listing temp(size, beg);

size++;

beg = new int[size];

beg[0] = temp.beg[0] \* temp.beg[a];

return \*this;

}

Файл Listing.h (вариант реализации 1)

#pragma once

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

const int MAX\_SIZE = 30;

class Listing {

// Дружественные функции ввода-вывода

friend ostream& operator << (ostream& out, const Listing& l);

friend istream& operator >> (istream& in, Listing& l);

public:

int size;

int\* beg;

Listing() { size = 0; beg = 0; }

Listing(int s);

Listing(int s, int\* mas);

Listing(const Listing& l);

~Listing();

const Listing& operator = (const Listing& l);

int operator[] (int i);

Listing operator\*(int a);

};

# 

Файл Исключ. сит.cpp (вариант реализации 3)

#include "Listing.h"

#include "Error.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

try {

cout << "Список с 2 пустыми элементами:\n";

Listing x(2);

Listing y;

cout << x;

cout << "\nВведите номер элемента:\n";

int i;

cin >> i;

cout << x[i] << endl;

y = x + 3;

cout << y;

//x \* 1;

cout << x;

}

catch (Error& e) {

e.what();

}

return 0;

}

Файл Listing.cpp (вариант реализации 3)

#include "Listing.h"

#include "Error.h"

#include <iostream>

using namespace std;

Listing::Listing(int s) {

if (s > MAX\_SIZE) throw MaxSizeError();

size = s;

beg = new int[s];

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

beg[i] = 0;

}

}

Listing::Listing(const Listing& l) {

size = l.size;

beg = new int[size];

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

beg[i] = l.beg[i];

}

}

Listing::~Listing() {

if (beg != 0) {

delete[] beg;

}

}

Listing::Listing(int s, int\* mas) {

size = s;

beg = new int[size];

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

beg[i] = mas[i];

}

}

const Listing& Listing::operator = (const Listing& l) {

if (this == &l) { return \*this; }

if (beg!= 0) { delete[] beg; }

size = l.size;

beg = new int[size];

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

beg[i] = l.beg[i];

}

return \*this;

}

ostream& operator << (ostream& out, const Listing& l) {

if (l.size == 0) { out << "Пусто\n"; }

else {

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

out << l.beg[i] << " ";

}

out << endl;

}

return out;

}

istream& operator >> (istream& in, Listing& l) {

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

cout << ">";

in >> l.beg[i];

}

return in;

}

int Listing::operator [] (int i) {

if (i < 0) {

throw IndexError1();

}

if (i >= size) {

throw IndexError2();

}

return beg[i];

}

Listing Listing::operator+(int a) {

if (size + 1 == MAX\_SIZE) {

throw MaxSizeError();

}

Listing temp(size + 1, beg);

temp.beg[size] = a;

return temp;

}

Listing Listing::operator\*(int a) {

Listing temp(size, beg);

size++;

beg = new int[size];

beg[0] = temp.beg[0] \* temp.beg[a];

return \*this;

}

Файл Listing.h (вариант реализации 3)

#pragma once

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

const int MAX\_SIZE = 30;

class Listing {

int size;

int\* beg;

friend ostream& operator << (ostream& out, const Listing& l);

friend istream& operator >> (istream& in, Listing& l);

public:

Listing() { size = 0; beg = 0; };

Listing(int s);

Listing(int s, int\* mas);

Listing(const Listing& l);

~Listing();

const Listing& operator = (const Listing& l);

int operator[] (int i);

Listing operator+(int a);

Listing operator\*(int a);

};

Error.h (вариант реализации 3)

#pragma once

#include <string>

#include <iostream>

using namespace std;

// Базовая структура

struct Error {

virtual void what() {};

};

// Ошибка в индексе вектора

struct IndexError : public Error {

IndexError() { msg = "Ошибка индекса\n"; }

virtual void what() { cout << msg; }

protected:

string msg;

};

// Ошибка в размере вектора

struct SizeError : public Error {

SizeError() { msg = "Ошибка длины\n"; }

virtual void what() { cout << msg; }

protected:

string msg;

};

// Превышение макс. размера

struct MaxSizeError : public SizeError {

MaxSizeError() { SizeError(); msg = "Длина > MAXSIZE\n"; }

virtual void what() { cout << msg << msg\_; }

protected:

string msg\_;

};

// Удаление из пустого вектора

struct EmptySizeError : public SizeError {

EmptySizeError() { SizeError(); msg\_ = "Вектор пустой\n"; }

virtual void what() { cout << msg << msg\_; }

protected:

string msg\_;

};

// Индекс < 0

struct IndexError1 : public IndexError {

IndexError1() { IndexError(); msg\_ = "Индекс <0\n"; }

virtual void what() { cout << msg << msg\_; }

protected:

string msg\_;

};

// Индекс больше текущего размера вектора

struct IndexError2 :public IndexError {

IndexError2() { IndexError(); msg\_ = "Индекс > длины\n"; }

virtual void what() { cout << msg << msg\_; }

protected:

string msg\_;

};

