Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.9**

Дисциплина: «Информатика»

Тема: Обработка исключительных ситуаций

Вариант 4

Выполнил:

Студент группы РИС-20-1б

Еске Вячеслав Сергеевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель задачи**

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

2) Разработка программы, обрабатывающей исключительные ситуации.

Для достижения поставленной цели необходимо решить следующие задачи:

* Проанализировать теоретические сведения о ООП
* Абстрактные типы данных. Контейнеры
* Реализовать алгоритм поставленной задачи
* Учесть все исключительные ситуации на ввод элементов
* Создать отдельные файлы: .cpp для описания методов класса, .cpp для описания главной функции, .h для описания класса

Постановка задачи

Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора;

- - – удаляет элемент из вектора (постфиксная операция удаляет элемент в конец вектора, префиксная – в начало)

Анализ задачи

1. Определить какие операции должны быть выполнены по заданию:

* Создание класса

class Vector

* Создание множества и заполнение класса
* Создание методов получения размера множества, пересечений и доступа по индексу
* Отчистка памяти

1. Для решения задачи используются переменные:

Несколько переменных целочисленного значения для ввода чисел и реализации меню

int s, in, menu = 4;

Указатель для работы с динамической памятью

1. Ввод данных осуществляется посредством функции

cin >> rub;

Вывод данных осуществляется посредством функции

cout << "Искомый элемент = " << a.give(in);

1. Создание метода класса , который ищет пересечения класса:

void Vector::cross()

{

int k = 0;

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

{

for (int j = i + 1; j < size; j++)

{

if (data[i] == data[j])

k++;

}

if (k != 0)

cout << "\nЭлемент " << data[i] << " встречается " << k + 1 << " раз";

k = 0;

}

1. Создание методов класса для нахождения размера и доступа по индексу

int give(int index);

int givesize();

Создание контейнера

Vector::Vector(int s)

{

int k;

size = s;

data = new int[size];

cout << "Введи элементы множества" << endl;

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

{

cin >> data[i];

}

}

1. Создание инструмента обработки исключительных ситуаций. Для этого была использована логика try:

try

{

cout << "Искомый элемент: " << a.give(in);

}

catch (const int in)

{

cout << "Ошибка ввода";

}

}

Код

#include "Vector.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

//контролируемый блок

try {

Vector x(2);//вектор из двух элементов

cout<<"Вектор: "<<x<<"\n";//печать вектора х

cout << "Введите значения вектора: ";

cin >> x;

cout<<"Введите номер элемента, который нужно вывести: ";

int i;

cin>>i; //вывод элемента с номером i, если номер больше 2 или меньше 0, то

//генерируется исключительная ситуация

cout << "Элемент под номером " << i<<endl;

cout<<x[i]<<endl;

//добавление элемента в вектор, если MAX\_SIZE=2, то генерируется исключительная ситуация

//удалить один элемент из вектора

cout << "Операция декремента постфиксная"<<endl;

x--;

cout << x;

//удалить один элемент из вектора

cout << "Операция декремента префиксная"<<endl;

--x;

cout << x;//вектор пустой

cout << "Операция декремента постфиксная"<<endl;

x--;

cout << x;

//удалить один элемент из вектора

//генерируется исключительная ситуация

}

//обработчик исключения

catch (int) {cout<<"Ошибка! Текущий размер больше максимального!"<<endl;}//сообщение об ошибке

catch (double) { cout << "Ошибка! Индекс отрицательный!" << endl; }

catch (char) { cout << "Оишбка! Индекс больше размера вектора!" << endl; }

catch (float) { cout << "Ошибка! Вектор пустой!" << endl; }

return 0;

}

#pragma once

#include <iostream>

using namespace std;

const int MAX\_SIZE=30;//максимальный размер вектора

class Vector {

int size;//текущий размер

int \*beg;//указатель на начало динамического массива

public:

Vector() { size = 0; beg = 0; }//конструктор без параметров

Vector(int s);//конструктор с параметром

Vector(int s,int\* mas);//конструктор с параметром

Vector(const Vector&v);//конструктор копирования

~Vector();//деструктор

const Vector& operator=(const Vector&v);//операция присваивания

int operator[](int i);//доступ по индексу

Vector& operator--();//удаление элемента

Vector operator--(int);

//дружественные функции ввода-вывода

friend ostream& operator<<(ostream&out,const Vector&v);

friend istream& operator>>(istream& in, Vector&v);

};

#include "Vector.h"

int a;

double b;

char c;

float d;

Vector::Vector(int s)

{

//если текущий размер больше максимального, то генерируется исключение

if(s>MAX\_SIZE) throw a;

size=s;

beg=new int [s];

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

beg[i]=0;

}

Vector::Vector(const Vector& v)

{

size = v.size;

beg = new int[size];

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

beg[i] = v.beg[i];

}

Vector::~Vector()

{

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

}

Vector::Vector(int s, int\* mas)

{

//если текущий размер больше максимального, то генерируется исключение

if(s>MAX\_SIZE) throw a;

size=s;

beg=new int[size];

for(int i=0;i<size;i++) beg[i]=mas[i];

}

const Vector& Vector::operator =(const Vector& v)

{

if (this == &v)return \*this;

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

size = v.size;

beg = new int[size];

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

beg[i] = v.beg[i];

return\*this;

}

ostream& operator<<(ostream& out, const Vector& v)

{

if (v.size == 0) out << "Empty\n";

else

{

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

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

out << endl;

}

return out;

}

istream& operator >>(istream& in, Vector& v)

{

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

{

cout << ">";

in >> v.beg[i];

}

return in;

}

int Vector::operator [](int i)

{

if (i < 0)throw b;//если индекс отрицательный, то генерируется исключение

//если индекс больше размера вектора, то генерируется исключение

if (i >= size) throw c;

return beg[i];

}

Vector& Vector::operator --()

{

//если вектор пустой, то удалить элемент нельзи и генерируется исключение

if(size==0) throw d;

if (size==1)//если в вектор один элемент

{

size = 0;

delete[]beg;

beg = 0;

return \*this;

};

Vector temp(size, beg);

delete[]beg;

size--;

beg = new int[size];

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

beg[i] = temp.beg[i+1];

return\*this;

}

Vector Vector::operator--(int)

{

if (size == 0) throw d;

if (size == 1)//если в вектор один элемент

{

size = 0;

delete[]beg;

beg = 0;

return \*this;

};

size--;

return\*this;

}

#include <iostream>

#include "Vector.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

//контролируемый блок

try {

Vector x(2);//вектор из двух элементов

cout << "Вектор: " << x << "\n";//печать вектора х

cout << "Введите значения вектора: ";

cin >> x;

cout << "Введите номер элемента, который нужно вывести: ";

int i;

cin >> i; //вывод элемента с номером i, если номер больше 2 или меньше 0, то

//генерируется исключительная ситуация

cout << "Элемент под номером " << i << endl;

cout << x[i] << endl;

//добавление элемента в вектор, если MAX\_SIZE=2, то генерируется исключительная ситуация

//удалить один элемент из вектора

cout << "Операция декремента постфиксная " << endl;

x--;

cout << x;

//удалить один элемент из вектора

cout << "Операция декремента префиксная " << endl;

--x;

cout << x;//вектор пустой

cout << "Операция декремента постфиксная " << endl;

x--;

cout << x;

//удалить один элемент из вектора

//генерируется исключительная ситуация

}

catch (error e) { e.what(); }

return 0;

}

#pragma once

#include <string>

#include <iostream>

using namespace std;

class error //класс ошибка

{

string str;

public:

//конструктор, инициирует атрибут str сообщением об ошибке

error(string s){str=s;}

void what(){cout<<str<<endl;} //выводит значение атрибута str

};

#pragma once

#include <iostream>

#include "error.h"

using namespace std;

const int MAX\_SIZE = 30;//максимальный размер вектора

class Vector {

int size;//текущий размер

int\* beg;//указатель на начало динамического массива

public:

Vector() { size = 0; beg = 0; }//конструктор без параметров

Vector(int s);//конструктор с параметром

Vector(int s, int\* mas);//конструктор с параметром

Vector(const Vector& v);//конструктор копирования

~Vector();//деструктор

const Vector& operator=(const Vector& v);//операция присваивания

int operator[](int i);//доступ по индексу

Vector& operator--();//удаление элемента

Vector operator--(int);

//дружественные функции ввода-вывода

friend ostream& operator<<(ostream& out, const Vector& v);

friend istream& operator>>(istream& in, Vector& v);

};

#include "Vector.h"

Vector::Vector(int s)

{

//если текущий размер больше максимального, то генерируется исключение

if (s > MAX\_SIZE) throw error("Ошибка! Текущий размер больше максимального!\n");

size = s;

beg = new int[s];

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

beg[i] = 0;

}

Vector::Vector(const Vector& v)

{

size = v.size;

beg = new int[size];

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

beg[i] = v.beg[i];

}

Vector::~Vector()

{

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

}

Vector::Vector(int s, int\* mas)

{

//если текущий размер больше максимального, то генерируется исключение

if (s > MAX\_SIZE) throw error("Ошибка! Текущий размер больше максимального!\n");

size = s;

beg = new int[size];

for (int i = 0; i < size; i++) beg[i] = mas[i];

}

const Vector& Vector::operator =(const Vector& v)

{

if (this == &v)return \*this;

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

size = v.size;

beg = new int[size];

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

beg[i] = v.beg[i];

return\*this;

}

ostream& operator<<(ostream& out, const Vector& v)

{

if (v.size == 0) out << "Empty\n";

else

{

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

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

out << endl;

}

return out;

}

istream& operator >>(istream& in, Vector& v)

{

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

{

cout << ">";

in >> v.beg[i];

}

return in;

}

int Vector::operator [](int i)

{

if (i < 0)throw error("Ошибка! Индекс отрицательный!\n");//если индекс отрицательный, то генерируется исключение

//если индекс больше размера вектора, то генерируется исключение

if (i >= size) throw error("Ошибка! Индекс больше размера вектора!\n");

return beg[i];

}

Vector& Vector::operator --()

{

//если вектор пустой, то удалить элемент нельзи и генерируется исключение

if (size == 0) throw error("Ошибка! Вектор пустой!\n");

if (size == 1)//если в вектор один элемент

{

size = 0;

delete[]beg;

beg = 0;

return \*this;

};

Vector temp(size, beg);

delete[]beg;

size--;

beg = new int[size];

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

beg[i] = temp.beg[i + 1];

return\*this;

}

Vector Vector::operator--(int)

{

if (size == 0) throw error("Ошибка! Вектор пустой!\n");

if (size == 1)//если в вектор один элемент

{

size = 0;

delete[]beg;

beg = 0;

return \*this;

};

size--;

return\*this;

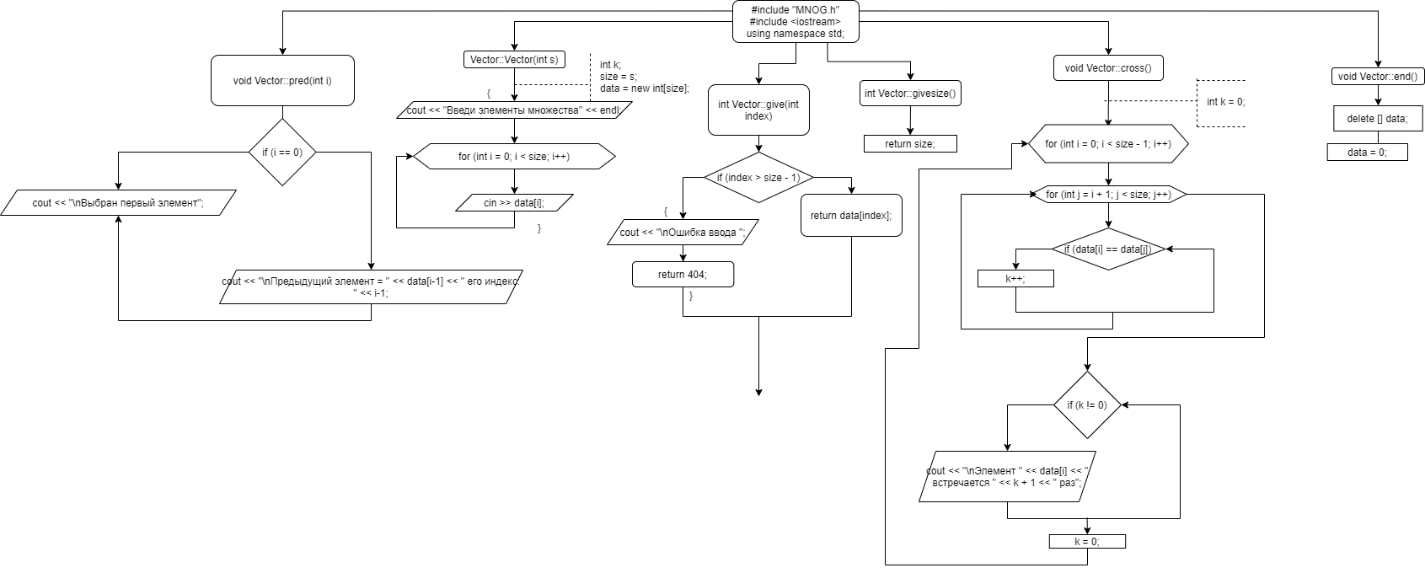
}

Блок-схема:

Main.cpp:



Pair.cpp:



Pair.h:



Работа кода

