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: Объектно-ориентированное программирование. Обработка исключительных ситуаций

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

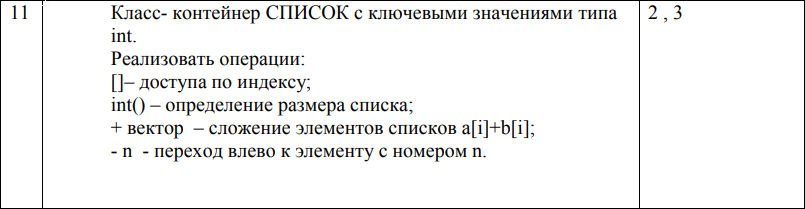
**Пермь, 2021**

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Реализация класса-контейнера.

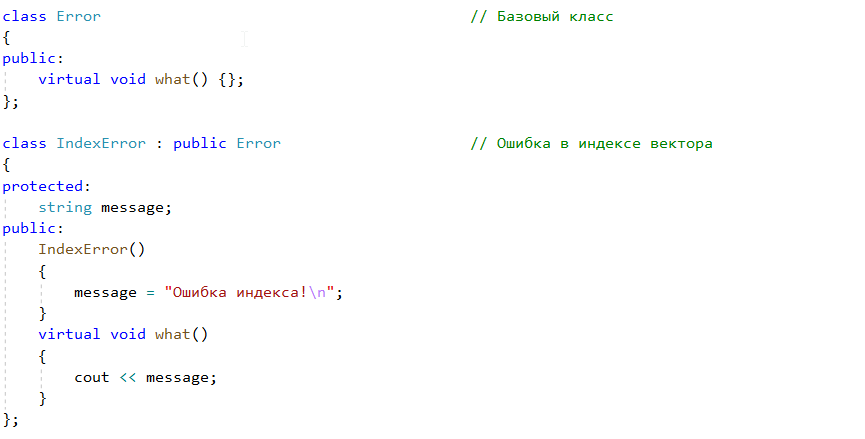
**Постановка задачи**

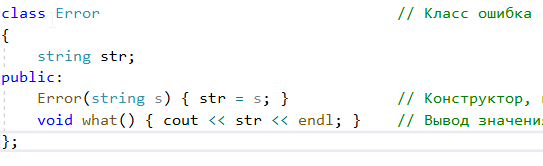
1. Реализовать класс, перегрузить для него операции, указанные в варианте.
2. Определить исключительные ситуации.
3. Предусмотреть генерацию исключительных ситуаций.



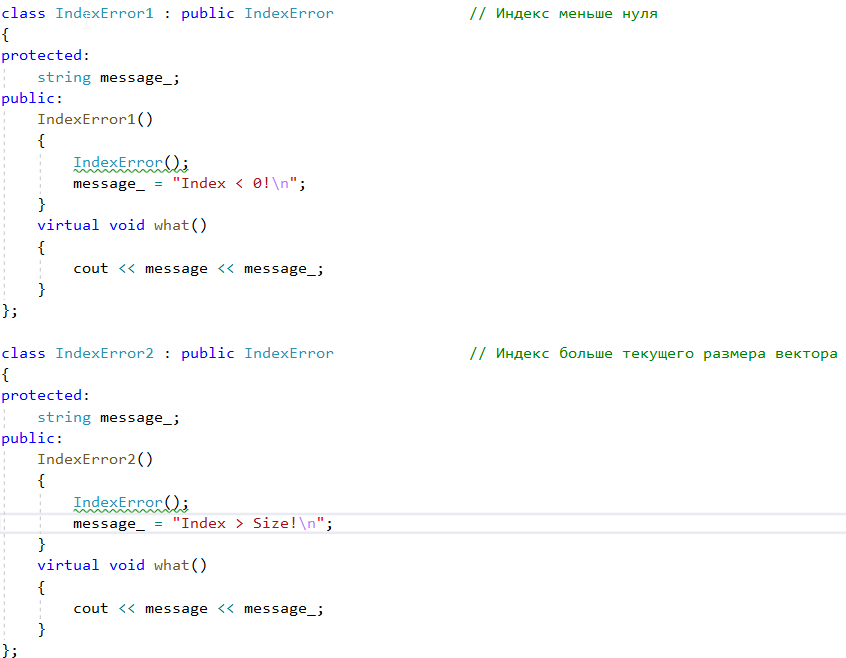
**Анализ задачи**

1. Для решения задачи необходимо:
   1. Реализовать определение классов Error / IndexError и определение необходимых методов данных классов.





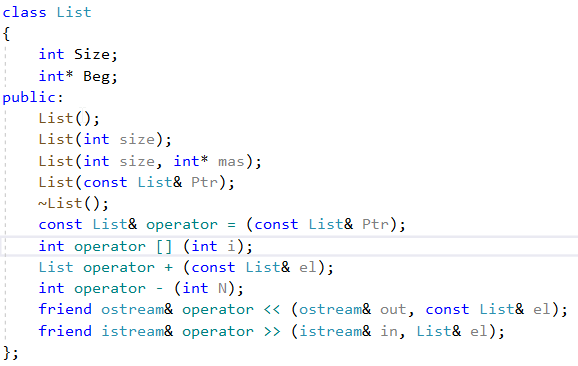
* 1. Реализовать определение классов IndexError1 / IndexError2 и определение необходимых методов и операторов данных классов.



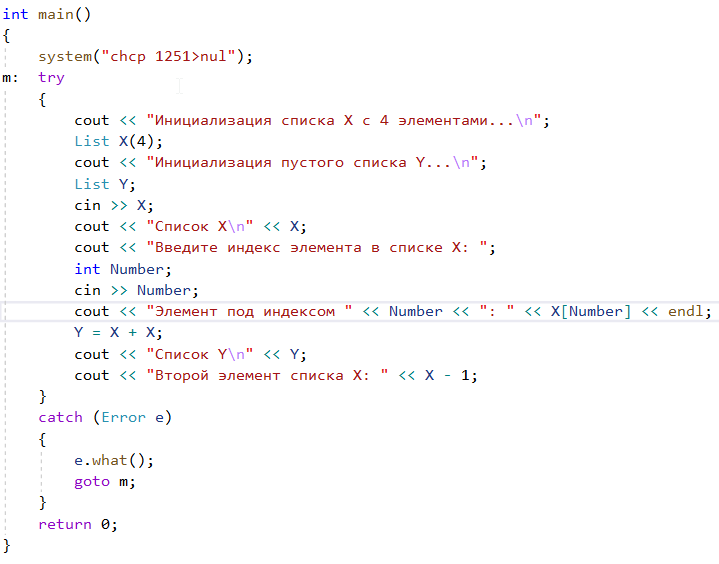
* 1. Реализовать определение производного от классов SizeError / MaxSizeError / EmptySizeError и определение необходимых методов и операторов данных классов.



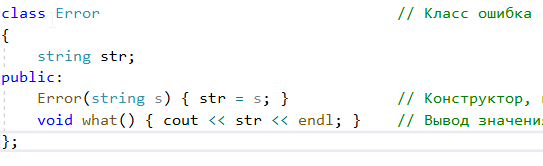
* 1. Реализовать определение класса List и определение необходимых методов и операторов данного класса

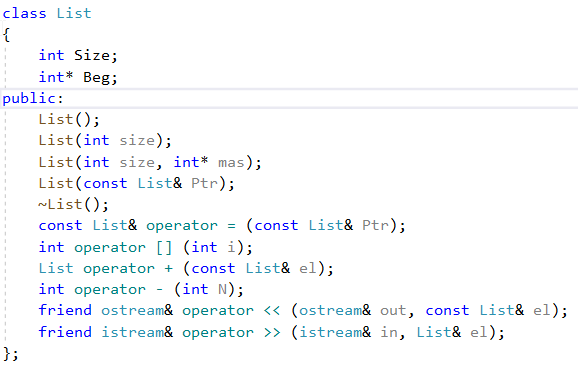


* 1. Реализовать применение этих функций в главной функции.



1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются атрибуты класса.





1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin, используемых при реализации в главной функции и функций перегрузки оператора.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_LSn5FLh1gO.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_hCz8IiLFDx.png

* 1. Вывод данных реализован с помощью оператора cout, используемого при реализации в главной функции и функций перегрузки операторов ввода-вывода.

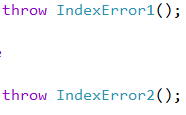
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_Dl9bJ2UEUB.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_CxVnhJnBKk.png

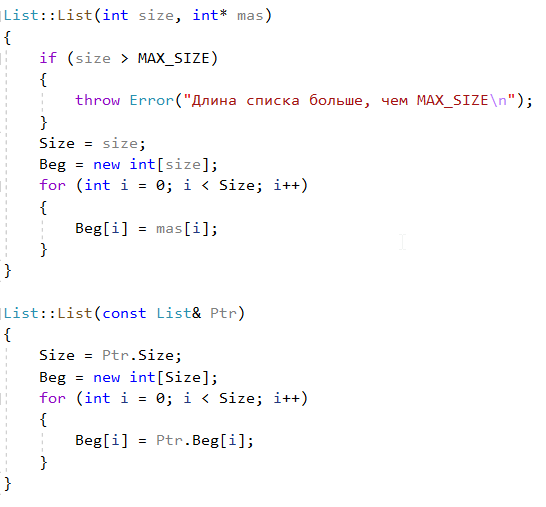
1. Поставленные задачи решены следующими действиями:
   1. В классах Error находится реализация обработки исключительных ситуаций.

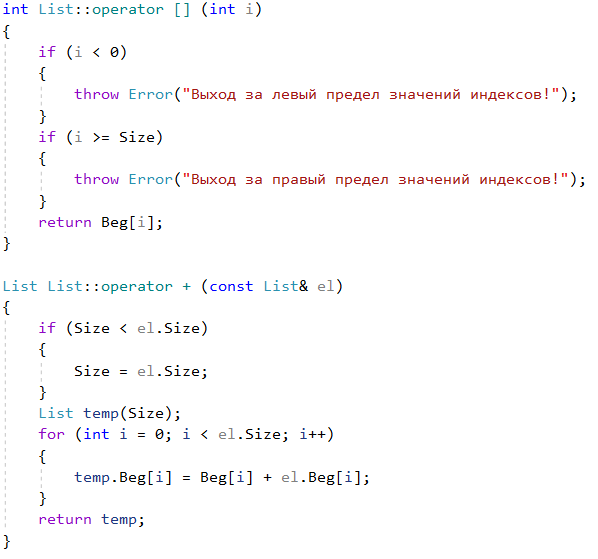
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_XvURaEJhdO.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_jGKuWI0mNv.png

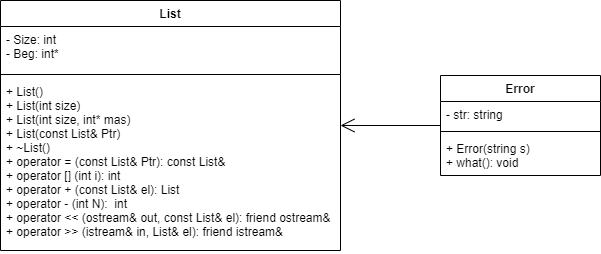


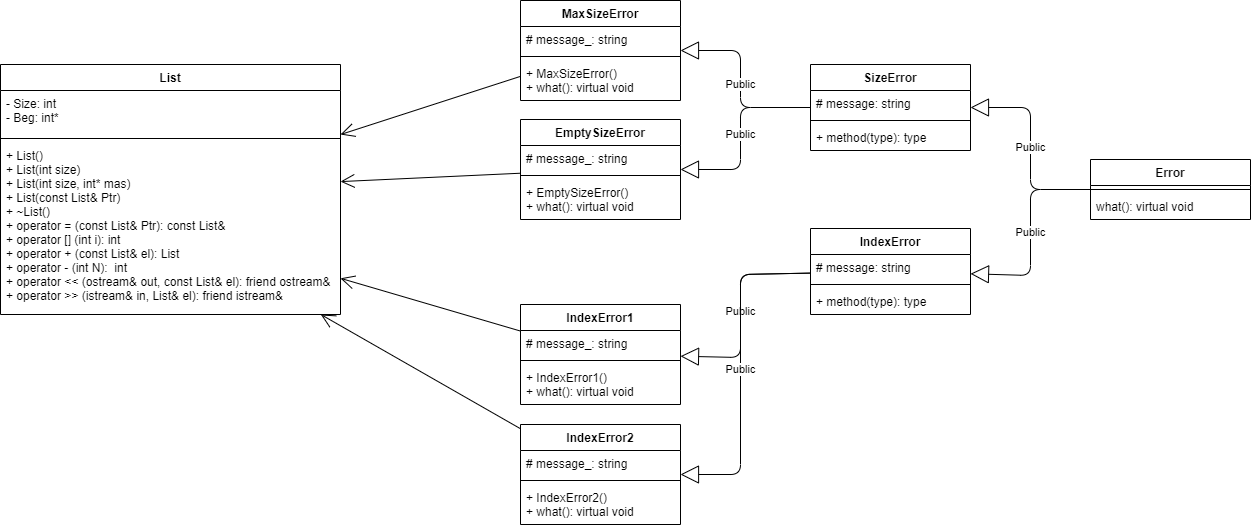
* 1. В классе List находится реализация списка, определение размера списка, доступ элемента списка по индексу, сложение векторов.





**UML-Диаграмма**





**Реализация задачи на языке С++**

**(1)**

**Заголовочный файл error.h**

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Error

{

string str;

public:

Error(string s) { str = s; }

void what() { cout << str << endl; }

};

**Заголовочный файл List.h**

#pragma once

#include <iostream>

using namespace std;

const int MAX\_SIZE = 30;

class List

{

int Size;

int\* Beg;

public:

List();

List(int size);

List(int size, int\* mas);

List(const List& Ptr);

~List();

const List& operator = (const List& Ptr);

int operator [] (int i);

List operator + (const List& el);

int operator - (int N);

friend ostream& operator << (ostream& out, const List& el);

friend istream& operator >> (istream& in, List& el);

};

**Файл с описанием методов класса List.cpp**

#include "List.h"

#include "Error.h"

#include <iostream>

using namespace std;

List::List()

{

Size = 0;

Beg = 0;

}

List::List(int size)

{

if (size > MAX\_SIZE)

{

throw Error("Длина списка больше, чем MAX\_SIZE\n");

}

Size = size;

Beg = new int[size];

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

{

Beg[i] = 0;

}

}

List::List(int size, int\* mas)

{

if (size > MAX\_SIZE)

{

throw Error("Длина списка больше, чем MAX\_SIZE\n");

}

Size = size;

Beg = new int[size];

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

{

Beg[i] = mas[i];

}

}

List::List(const List& Ptr)

{

Size = Ptr.Size;

Beg = new int[Size];

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

{

Beg[i] = Ptr.Beg[i];

}

}

List::~List()

{

if (Beg != 0)

{

delete[] Beg;

}

}

const List& List::operator = (const List& Ptr)

{

if (&Ptr != this)

{

if (Beg != 0)

{

delete[] Beg;

}

Size = Ptr.Size;

Beg = new int[Size];

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

{

Beg[i] = Ptr.Beg[i];

}

}

return \*this;

}

ostream& operator << (ostream& out, const List& el)

{

if (el.Size == 0)

{

out << "Список пуст!\n";

}

else

{

out << "Вывожу элементы списка...\n";

for (int i = 0; i < el.Size; i++)

{

out << el.Beg[i] << " ";

}

out << endl;

}

return out;

}

istream& operator >> (istream& in, List& el)

{

cout << "Введите элементы списка\n";

for (int i = 0; i < el.Size; i++)

{

cout << "> ";

in >> el.Beg[i];

}

return in;

}

int List::operator [] (int i)

{

if (i < 0)

{

throw Error("Выход за левый предел значений индексов!");

}

if (i >= Size)

{

throw Error("Выход за правый предел значений индексов!");

}

return Beg[i];

}

List List::operator + (const List& el)

{

if (Size < el.Size)

{

Size = el.Size;

}

List temp(Size);

for (int i = 0; i < el.Size; i++)

{

temp.Beg[i] = Beg[i] + el.Beg[i];

}

return temp;

}

int List::operator - (int N)

{

if (N < Size || N > -1)

{

return Beg[N];

}

else

{

throw Error("Индекс за пределами размерности списка!\n");

}

}

**Файл с главной программой Лабораторная работа № 18.9.cpp**

#include "List.h"

#include "Error.h"

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251>nul");

m: try

{

cout << "Инициализация списка X с 4 элементами...\n";

List X(4);

cout << "Инициализация пустого списка Y...\n";

List Y;

cin >> X;

cout << "Список X\n" << X;

cout << "Введите индекс элемента в списке X: ";

int Number;

cin >> Number;

cout << "Элемент под индексом " << Number << ": " << X[Number] << endl;

Y = X + X;

cout << "Список Y\n" << Y;

cout << "Второй элемент списка X: " << X - 1;

}

catch (Error e)

{

e.what();

goto m;

}

return 0;

}

**(2)**

**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 message;

public:

IndexError()

{

message = "Ошибка индекса!\n";

}

virtual void what()

{

cout << message;

}

};

class SizeError : public Error

{

protected:

string message;

public:

SizeError()

{

message = "Ошибка размера!\n";

}

virtual void what()

{

cout << message;

}

};

class MaxSizeError : public SizeError

{

protected:

string message\_;

public:

MaxSizeError()

{

SizeError();

message\_ = "Size > MAX\_SIZE!\n";

}

virtual void what()

{

cout << message << message\_;

}

};

class EmptySizeError : public SizeError

{

protected:

string message\_;

public:

EmptySizeError()

{

SizeError();

message\_ = "Список пуст!\n";

}

virtual void what()

{

cout << message << message\_;

}

};

class IndexError1 : public IndexError

{

protected:

string message\_;

public:

IndexError1()

{

IndexError();

message\_ = "Index < 0!\n";

}

virtual void what()

{

cout << message << message\_;

}

};

class IndexError2 : public IndexError

{

protected:

string message\_;

public:

IndexError2()

{

IndexError();

message\_ = "Index > Size!\n";

}

virtual void what()

{

cout << message << message\_;

}

};

**Заголовочный файл List.h**

#pragma once

#include <iostream>

using namespace std;

const int MAX\_SIZE = 30;

class List

{

int Size;

int\* Beg;

public:

List();

List(int size);

List(int size, int\* mas);

List(const List& Ptr);

~List();

const List& operator = (const List& Ptr);

int operator [] (int i);

List operator + (const List& Ptr);

List operator - (int N);

friend ostream& operator << (ostream& out, const List& Ptr);

friend istream& operator >> (istream& in, List& Ptr);

};

**Файл с описанием методов класса List.cpp**

#include "List.h"

#include "Error.h"

#include <iostream>

using namespace std;

List::List()

{

Size = 0;

Beg = 0;

}

List::List(int size)

{

if (size > MAX\_SIZE)

{

throw MaxSizeError();

}

Size = size;

Beg = new int[size];

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

{

Beg[i] = 0;

}

}

List::List(int size, int\* mas)

{

Size = size;

Beg = new int[Size];

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

{

Beg[i] = mas[i];

}

}

List::List(const List& Ptr)

{

Size = Ptr.Size;

Beg = new int[Size];

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

{

Beg[i] = Ptr.Beg[i];

}

}

List::~List()

{

if (Beg != 0)

{

delete[] Beg;

}

}

const List& List::operator = (const List& Ptr)

{

if (&Ptr != this)

{

if (Beg != 0)

{

delete[] Beg;

}

Size = Ptr.Size;

Beg = new int[Size];

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

{

Beg[i] = Ptr.Beg[i];

}

}

return \*this;

}

ostream& operator << (ostream& out, const List& Ptr)

{

if (Ptr.Size == 0) out << "Empty\n";

else

{

for (int i = 0; i < Ptr.Size; i++)

out << Ptr.Beg[i] << " ";

out << endl;

}

return out;

}

istream& operator >> (istream& in, List& Ptr)

{

for (int i = 0; i < Ptr.Size; i++)

{

cout << "> ";

in >> Ptr.Beg[i];

}

return in;

}

int List::operator [] (int i)

{

if (i < 0)

{

throw IndexError1();

}

if (i >= Size)

{

throw IndexError2();

}

return Beg[i];

}

List List::operator + (const List& Ptr)

{

if (Size < Ptr.Size)

{

Size = Ptr.Size;

}

List temp(Size);

for (int i = 0; i < Ptr.Size; i++)

{

temp.Beg[i] = Beg[i] + Ptr.Beg[i];

}

return temp;

}

List List::operator - (int N)

{

if (N < Size && N > -1)

{

return Beg[N];

}

else

{

if (N < 0)

{

throw IndexError1();

}

else

{

throw IndexError2();

}

}

}

**Файл с главной программой Лабораторная работа № 18.9.cpp**

#include "List.h"

#include "Error.h"

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251>nul");

try

{

cout << "Инициализация списка X с 4 элементами...\n";

List X(4);

cout << "Инициализация пустого списка Y...\n";

List Y;

cin >> X;

cout << "Список X\n" << X;

cout << "Введите индекс элемента в списке X: ";

int Number;

cin >> Number;

cout << "Элемент под индексом " << Number << ": " << X[Number] << endl;

Y = X + X;

cout << "Список Y\n" << Y;

cout << "Второй элемент списка X: " << X - 1;

}

catch (Error &e)

{

e.what();

}

return 0;

}

**Ответы на вопросы**

1. Объект, который генерирует система при возникновении исключительной ситуации
2. Поиск исключения и его обработка
3. Throw
4. Try
5. Catch
6. Тип, тип и имя, «…»
7. Exception
8. Унаследовать собственные классы от абстрактного
9. Int и double
10. Никакие
11. В любой