Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе 18.9**

Тема: Классы и объекты. Обработка исключительных ситуаций

Выполнил работу

студент группы РИС-20-1б

Томилов В.А.

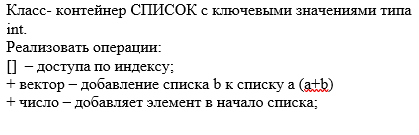
Проверила

Доцент кафедры ИТАС

к.т.н ПоляковаО.А.

Пермь, 2021

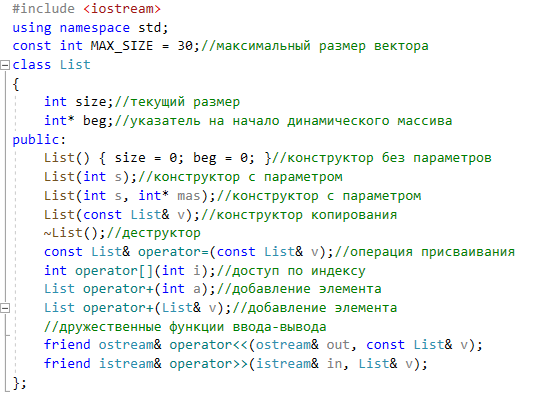
Постановка задачи



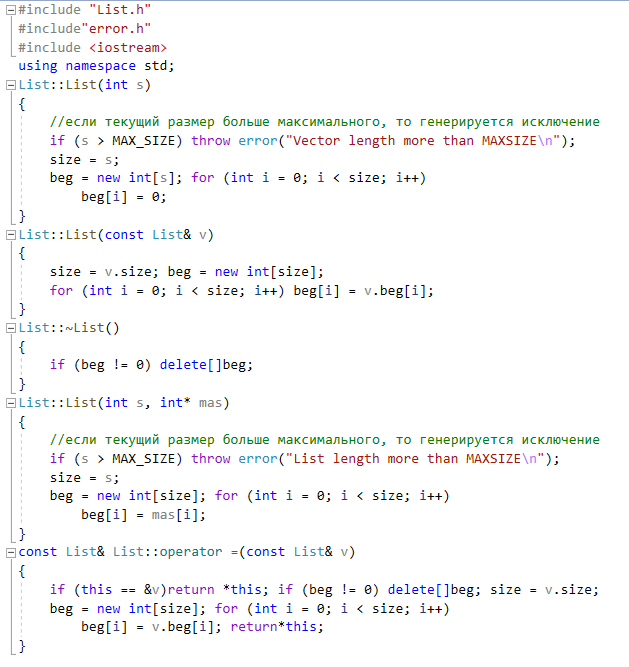
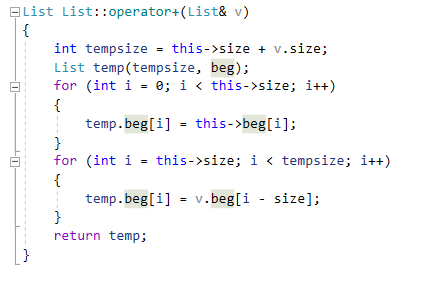
Анализ задачи

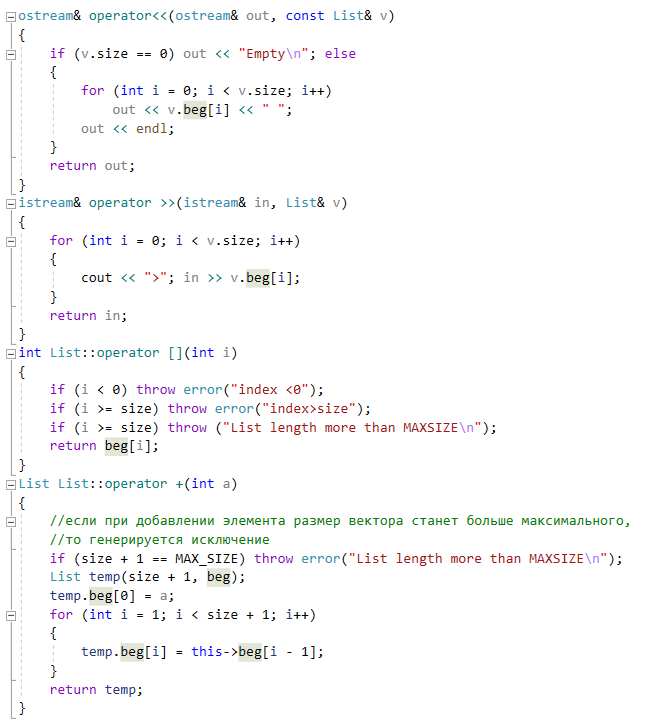
2) Информация об исключительных ситуациях передается с помощью пользовательского класса.

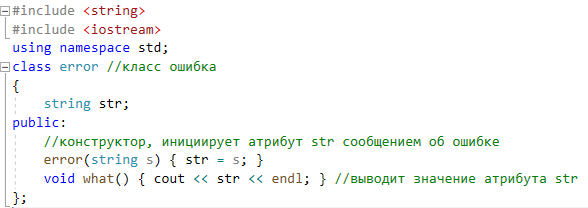
* 1. Для решения задачи необходимо
  2. Добавить в проект файл List.h, содержащий описание класса.

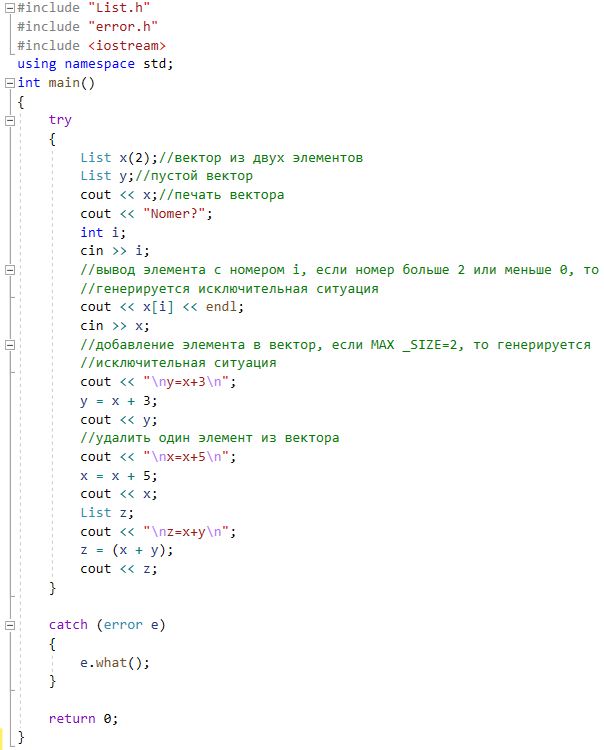


* 1. Добавим в проект файл List.cpp, содержащий описание методов класса List.





* 1. Добавить в проект файл error.h, содержащий описание класса error и описание его методов.
  2. Добавить в проект файл Lab9\_2\_main.cpp, содержащий основную программу.

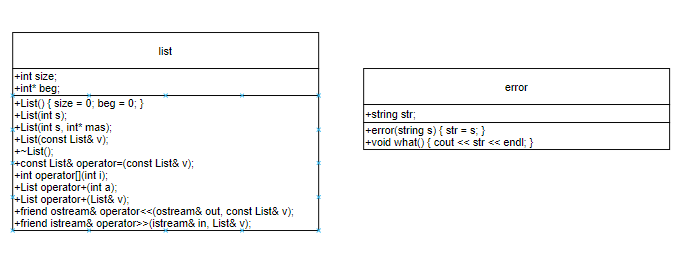


* 1. В ходе работы были использованы следующие типы данных:

2.1 (int) size – текущий размер

2.2 (int\*) beg – указатель на начало динамического массива

2.3 (string) str – описание ошибки

UML – диаграмма

Код программы

Main.cpp

#include "List.h"

#include "error.h"

#include <iostream>

using namespace std;

int main()

{

try

{

List x(2);//вектор из двух элементов

List y;//пустой вектор

cout << x;//печать вектора

cout << "Nomer?";

int i;

cin >> i;

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

//генерируется исключительная ситуация

cout << x[i] << endl;

cin >> x;

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

//исключительная ситуация

cout << "\ny=x+3\n";

y = x + 3;

cout << y;

//удалить один элемент из вектора

cout << "\nx=x+5\n";

x = x + 5;

cout << x;

List z;

cout << "\nz=x+y\n";

z = (x + y);

cout << z;

}

catch (error e)

{

e.what();

}

return 0;

}

List.h

#pragma once

#include <iostream>

using namespace std;

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

class List

{

int size;//текущий размер

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

public:

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

List(int s);

List(int s, int\* mas);

List(const List& v);

~List();

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

int operator[](int i);

List operator+(int a);

List operator+(List& v);

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

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

};

List.cpp

#include "List.h"

#include"error.h"

#include <iostream>

using namespace std;

List::List(int s)

{

//если текущий размер больше максимального, то генерируется исключение

if (s > MAX\_SIZE) throw error("Vector length more than MAXSIZE\n");

size = s;

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

beg[i] = 0;

}

List::List(const List& v)

{

size = v.size; beg = new int[size];

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

}

List::~List()

{

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

}

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

{

//если текущий размер больше максимального, то генерируется исключение

if (s > MAX\_SIZE) throw error("List length more than MAXSIZE\n");

size = s;

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

beg[i] = mas[i];

}

const List& List::operator =(const List& 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 List& 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, List& v)

{

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

{

cout << ">"; in >> v.beg[i];

}

return in;

}

int List::operator [](int i)

{

if (i < 0) throw error("index <0");

if (i >= size) throw error("index>size");

if (i >= size) throw ("List length more than MAXSIZE\n");

return beg[i];

}

List List::operator +(int a)

{

//если при добавлении элемента размер вектора станет больше максимального,

//то генерируется исключение

if (size + 1 == MAX\_SIZE) throw error("List length more than MAXSIZE\n");

List temp(size + 1, beg);

temp.beg[0] = a;

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

{

temp.beg[i] = this->beg[i - 1];

}

return temp;

}

List List::operator+(List& v)

{

int tempsize = this->size + v.size;

List temp(tempsize, beg);

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

{

temp.beg[i] = this->beg[i];

}

for (int i = this->size; i < tempsize; i++)

{

temp.beg[i] = v.beg[i - size];

}

return temp;

}

Error.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class error //класс ошибка

{

string str;

public:

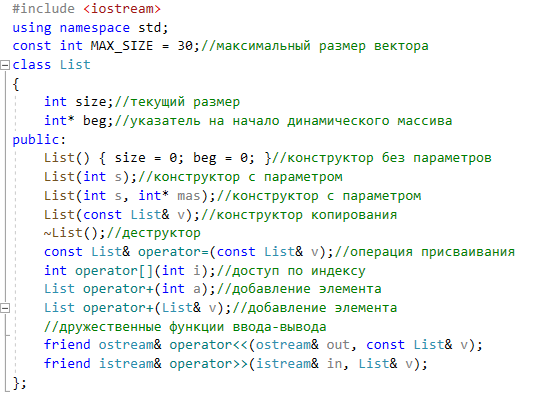
//конструктор, инициирует атрибут str сообщением об ошибке

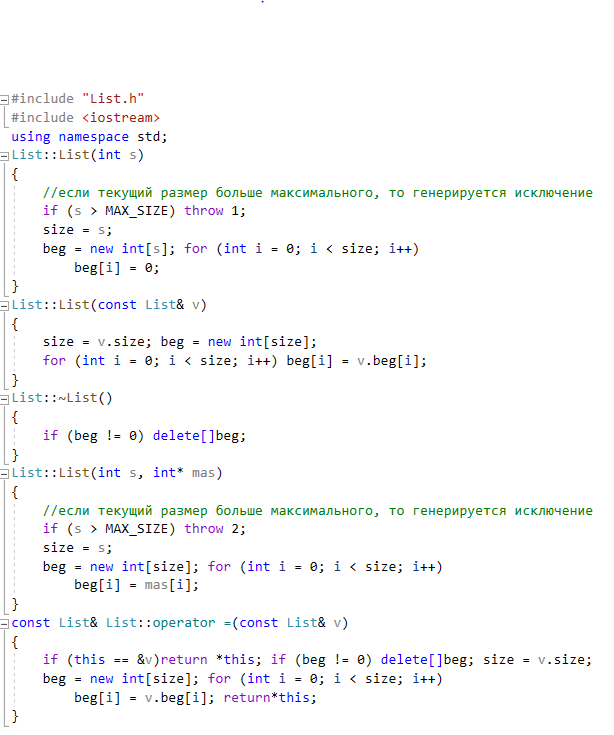
error(string s) { str = s; }

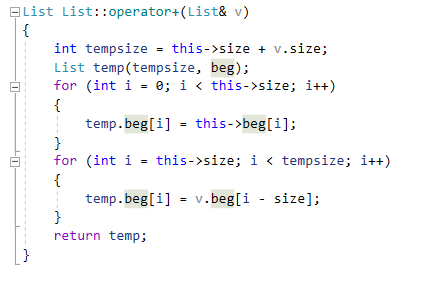
void what() { cout << str << endl; } //выводит значение атрибута str

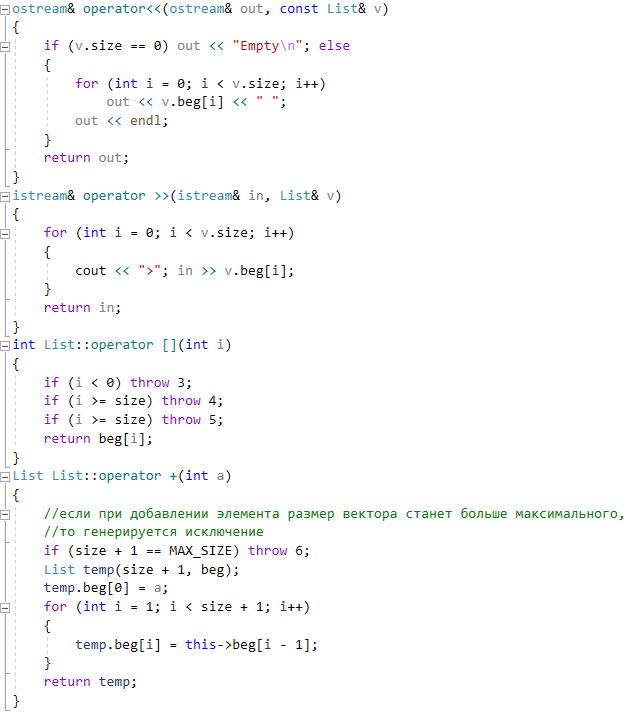
};

1. Информация об исключительных ситуациях передается с помощью стандартного типа данных.
2. Для решения задачи необходимо
   1. Добавить в проект файл List.h, содержащий описание класса.

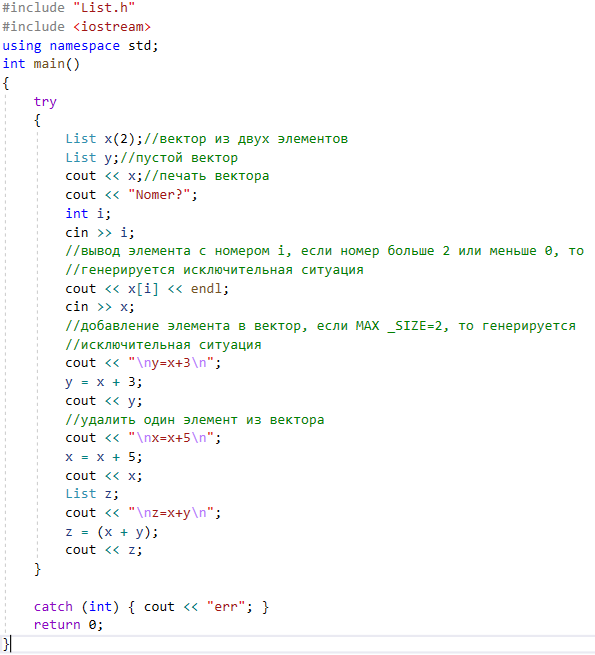


1.2 Добавим в проект файл List.cpp, содержащий описание методов класса List.

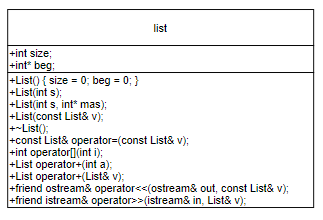




* 1. Добавить в проект файл Lab9\_1\_main.cpp, содержащий основную программу.



UML – диаграмма



Код программы

main.cpp

#include "List.h"

#include <iostream>

using namespace std;

int main()

{

try

{

List x(2);//вектор из двух элементов

List y;//пустой вектор

cout << x;//печать вектора

cout << "Nomer?";

int i;

cin >> i;

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

//генерируется исключительная ситуация

cout << x[i] << endl;

cin >> x;

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

//исключительная ситуация

cout << "\ny=x+3\n";

y = x + 3;

cout << y;

//удалить один элемент из вектора

cout << "\nx=x+5\n";

x = x + 5;

cout << x;

List z;

cout << "\nz=x+y\n";

z = (x + y);

cout << z;

}

catch (int) { cout << "err"; }

return 0;

}

list.h

#pragma once

#include <iostream>

using namespace std;

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

class List

{

int size;//текущий размер

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

public:

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

List(int s);

List(int s, int\* mas);

List(const List& v);

~List();

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

int operator[](int i);

List operator+(int a);

List operator+(List& v);

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

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

};

list.cpp

#include "List.h"

#include <iostream>

using namespace std;

List::List(int s)

{

//если текущий размер больше максимального, то генерируется исключение

if (s > MAX\_SIZE) throw 1;

size = s;

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

beg[i] = 0;

}

List::List(const List& v)

{

size = v.size; beg = new int[size];

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

}

List::~List()

{

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

}

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

{

//если текущий размер больше максимального, то генерируется исключение

if (s > MAX\_SIZE) throw 2;

size = s;

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

beg[i] = mas[i];

}

const List& List::operator =(const List& 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 List& 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, List& v)

{

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

{

cout << ">"; in >> v.beg[i];

}

return in;

}

int List::operator [](int i)

{

if (i < 0) throw 3;

if (i >= size) throw 4;

if (i >= size) throw 5;

return beg[i];

}

List List::operator +(int a)

{

//если при добавлении элемента размер вектора станет больше максимального,

//то генерируется исключение

if (size + 1 == MAX\_SIZE) throw 6;

List temp(size + 1, beg);

temp.beg[0] = a;

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

{

temp.beg[i] = this->beg[i - 1];

}

return temp;

}

List List::operator+(List& v)

{

int tempsize = this->size + v.size;

List temp(tempsize, beg);

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

{

temp.beg[i] = this->beg[i];

}

for (int i = this->size; i < tempsize; i++)

{

temp.beg[i] = v.beg[i - size];

}

return temp;

}

Ответы на вопросы

1. Исключение – это непредвиденное или аварийное событие
2. Исключения позволяют разделить вычислительный процесс на 2 части:

* обнаружение аварийной ситуации (неизвестно как обрабатывать);
* обработка аварийной ситуации (неизвестно, где она возникла).

Достоинства такого подхода:

* удобно использовать в программе, которая состоит из нескольких модулей;
* не требуется возвращать значение в вызывающую функцию

1. Исключение генерируется оператором

throw <выражение>

1. Исключение надо перехватить и обработать. Для проверки возникновения исключения используется контролируемый блок try{}
2. –
3. catch( спецификация исключения ), где спецификация исключения может иметь три формы:

(тип имя)

(тип)

(…)

Формы 1 и 2 обрабатывают конкретные исключения, а форма 3 перехватывает все исключения

1. Можно определять собственные исключения, унаследовав их от класса exception.
2. Для создания собственной иерархии исключений надо объявить свой базовый класс- исключение, например:

class BaseException{};

Остальные классы будут наследниками этого класса, аналогично тому, как это сделано в иерархии стандартных исключений:

class Child\_Exception1:public BaseException{};

class Child\_Exception2:public BaseException{};

Класс BaseException можно унаследовать от стандартного класса exception

class BaseException: public exception{};

1. int,double
2. функция исключений не генерирует
3. В любой
4. -