Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**О Т Ч Е Т**

**по лабораторной работе “Классы” №9**

**по дисциплине**

**«Основы алгоритмизации и программирования» семестр 2**

Выполнил студент гр. ИВТ-21-1б

Ипатов Дмитрий Сергеевич

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Д.В.

(оценка) (подпись)

г. Пермь-2022

**Постановка задачи:**

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

**Анализ задачи:**

Исключение – это непредвиденное или аварийное событие.

В С++ исключение – это объект, который система должна генерировать при возникновении исключительной ситуации. Генерация такого объекта и создает исключительную ситуацию.

Исключения позволяют разделить вычислительный процесс на 2 части:

* + 1. обнаружение аварийной ситуации (неизвестно как обрабатывать);
    2. обработка аварийной ситуации (неизвестно, где она возникла). Достоинства такого подхода:

1. удобно использовать в программе, которая состоит из нескольких модулей;

не требуется возвращать значение в вызывающую функцию

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{};

Наследование от стандартных классов позволит использовать метод what для вывода сообщений об ошибках.

Иерархия классов-исключений позволяет вместо нескольких разных блоков-ловушек написать единственный блок с типом аргумента базового класса.

**Код программы (вариант реализации 1):**

#include <iostream>

using namespace std;

const int max\_size = 7;

class List

{

protected:

int size;

int\* data;

public:

List(void);

List(int s, int k);

List(const List&);

~List();

List& operator=(const List&);

int& operator[](int index);

List operator+(int k);

List operator+(const List&);

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

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

};

List::List(int s, int k)

{

if (s > max\_size) throw 1;

size = s;

data = new int [size];

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

data[i] = k;

}

List::List()

{

size = 0;

data = 0;

}

List::List(const List& t)

{

size = t.size;

data = new int[size];

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

{

data[i] = t.data[i];

}

}

List::~List()

{

delete[] data;

data = 0;

}

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

{

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

size = t.size;

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

data = new int[size];

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

{

data[i] = t.data[i];

}

return \*this;

}

int& List::operator[](int index)

{

if (index < 0) throw 2;

if (index >= size) throw 3;

return data[index];

}

List List::operator+(int k)

{

if (size + 1 == max\_size) throw 4;

int newsize = size + 1;

List temp(newsize, 0);

temp.data[0] = k;

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

{

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

}

return temp;

}

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

{

int l;

if (size < t.size) l = t.size;

else l = size;

List temp1(l, 0);

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

{

temp1.data[i] = temp1.data[i] + data[i];

}

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

{

temp1.data[i] = temp1.data[i] + t.data[i];

}

return temp1;

}

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

{

if (p.size == 0) out << "Список пуст!\n";

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

out << p.data[i] << " ";

return out;

}

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

{

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

in >> p.data[i];

return in;

}

int main()

{

setlocale(LC\_ALL, "rus");

try

{

List a(5, 1);

cout << a << '\n';

List b(6, 2); // Размер вектора превышает максимальный, так что копилятор кинет исключение в обработчик

cout << b + a << '\n';

cout << a[3] << '\n';

cout << b + 2;

}

catch (int j)

{

cout << "ОШИБКА!!!";

if (j == 1)

{

cout << "Введённый размер списка превышает максимальный (max:7)!";

}

if (j == 2)

{

cout << "Индекс меньше нуля!!!";

}

if (j == 3)

{

cout << "Индекс превышает максимальный размер списка!";

}

if (j == 4)

{

cout << "При добавлении нового элемента список превысил максимальный размер";

}

}

return 0;

}

**Код программы (вариант реализации 2):**

#include <iostream>

using namespace std;

const int max\_size = 7;

class Exception

{

string str;

public:

Exception(string s) { str = s; }

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

};

class List

{

protected:

int size;

int\* data;

public:

List(void);

List(int s, int k);

List(const List&);

~List();

List& operator=(const List&);

int& operator[](int index);

List operator+(int k);

List operator+(const List&);

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

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

};

List::List(int s, int k)

{

if (s > max\_size) throw Exception("Введённый размер списка превышает максимальный (max:7)!");

size = s;

data = new int[size];

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

data[i] = k;

}

List::List()

{

size = 0;

data = 0;

}

List::List(const List& t)

{

size = t.size;

data = new int[size];

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

{

data[i] = t.data[i];

}

}

List::~List()

{

delete[] data;

data = 0;

}

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

{

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

size = t.size;

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

data = new int[size];

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

{

data[i] = t.data[i];

}

return \*this;

}

int& List::operator[](int index)

{

if (index < 0) throw Exception("Индекс меньше нуля!!!");

if (index >= size) throw Exception("Индекс превышает максимальный размер списка!");

return data[index];

}

List List::operator+(int k)

{

if (size + 1 == max\_size) throw Exception("При добавлении нового элемента список превысил максимальный размер");

int newsize = size + 1;

List temp(newsize, 0);

temp.data[0] = k;

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

{

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

}

return temp;

}

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

{

int l;

if (size < t.size) l = t.size;

else l = size;

List temp1(l, 0);

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

{

temp1.data[i] = temp1.data[i] + data[i];

}

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

{

temp1.data[i] = temp1.data[i] + t.data[i];

}

return temp1;

}

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

{

if (p.size == 0) out << "Список пуст!\n";

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

out << p.data[i] << " ";

return out;

}

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

{

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

in >> p.data[i];

return in;

}

int main()

{

setlocale(LC\_ALL, "rus");

try

{

List a(5, 1);

cout << a << '\n';

List b(6, 2);

cout << b + a << '\n';

cout << a[3] << '\n';

cout << a[-1] << '\n'; // Индекс меньше нуля

/\*cout << b + 2;\*/ // Размер вектора превысит максимальный, так что компилятор кинет исключение в обработчик

}

catch (Exception e)

{

cout << "ОШИБКА!!!";

e.what();

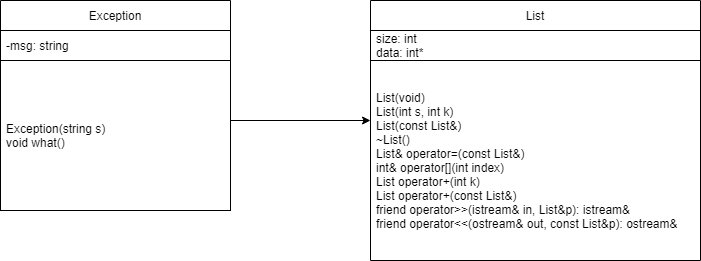
}

return 0;

}

**UML (вариант реализации 1):**

**UML (вариант реализации 2):**

****

**Контрольные вопросы:**

* + 1. Что представляет собой исключение в С++?

Исключение – аварийное или непредвиденное событие.

* + 1. На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?

Обнаружение и обработка. Удобно использовать в программе из нескольких модулей, не требуется возвращать значение в вызывающую ф-ию.

* + 1. Какой оператор используется для генерации исключительной ситуации?

Throw

* + 1. Что представляет собой контролируемый блок? Для чего он нужен?

Часть программы, которая будет выполняться до возникновения исключения.

* + 1. Что представляет собой секция-ловушка? Для чего она нужна?

Это часть программы, которая «отлавливает» исключения и обрабатывает их.

* + 1. Какие формы может иметь спецификация исключения в секции ловушке? В каких ситуациях используются эти формы?

По значению, по ссылке, по константной ссылке, по указателю. Лучше всего передавать объект по ссылке, так как при этом не создается временный объект-исключение.

* + 1. Какой стандартный класс можно использовать для создания собственной иерархии исключений?

Exception

* + 1. Каким образом можно создать собственную иерархию исключений?

Унаследовав стандартную.

* + 1. Если спецификация исключений имеет вид: void f1()throw(int,double); то какие исключения может прождать функция f1()?

Int и double

* + 1. Если спецификация исключений имеет вид: void f1()throw(); то какие исключения может прождать функция f1()?

Любые

* + 1. В какой части программы может генерироваться исключение?

По идее, в любой, может встретиться в блоке try в самом начале, но обрабатывается catch в конце.

* + 1. Написать функцию, которая вычисляет площадь треугольника по трем сторонам (формула Герона).

Функцию реализовать в 4 вариантах:

* + - * без спецификации исключений;
      * со спецификацией throw();
      * с конкретной спецификацией с подходящим стандартным исключением;
      * спецификация с собственным реализованным исключением.

#include<iostream>

#include<cmath>

#include<string>

using namespace std;

class error

{

private:

string msg;

public:

error(string msg)

{

this->msg = msg;

}

void what()

{

cout << "Произошла ошибка: " << msg << endl;

}

};

class triangle

{

protected:

int first; int second; int third;

float s; int p;

public:

triangle(int first, int second, int third)

{

this->first = first;

this->second = second;

this->third = third;

}

~triangle()

{}

void cur\_s\_4()

{

if (first <= 0 or second <= 0 or third <= 0) { throw error("Ошибочкааа"); }

p = first + second + third;

s = sqrt(p\*(p - first)\*(p - second)\*(p - third));

}

void cur\_s\_3()

{

int num = 2;

if (first <= 0 or second <= 0 or third <= 0) { throw num; }

p = first + second + third;

s = sqrt(p\*(p - first)\*(p - second)\*(p - third));

}

void cur\_s\_2()

{

if (first <= 0 or second <= 0 or third <= 0) { throw(1); }

p = first + second + third;

s = sqrt(p\*(p - first)\*(p - second)\*(p - third));

}

void cur\_s\_1()

{

if (first <= 0 or second <= 0 or third <= 0) { throw; }

p = first + second + third;

s = sqrt(p\*(p - first)\*(p - second)\*(p - third));

}

void show()

{

cout << "Площадь - " << s << endl;

}

};

int main()

{

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите вариант исключения: ";

cin >> n;

cout << endl;

switch (n)

{

case 1:

try

{

triangle one(-5, 10, 15);

one.cur\_s\_1();

one.show();

}

catch (...)

{

cout << "Ошибка!!! " << endl;

}

break;

case 2:

try

{

triangle two(-10, 15, 20);

two.cur\_s\_2();

two.show();

}

catch (int)

{

cout << "Ошибка!!! "<< endl;

}

break;

case 3:

try

{

triangle three(-15, 20, 25);

three.cur\_s\_3();

three.show();

}

catch (int num)

{

cout << "Ошибка с кодом: " << num << endl;

}

break;

case 4:

try

{

triangle four(-20, 25, 30);

four.cur\_s\_4();

four.show();

}

catch (error er)

{

er.what();

}

break;

}

return 0;

}