Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «Обработка исключительных ситуаций»

Выполнил студент ИВТ-22-2б:

Шилов Валерий Михайлович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

1. Реализовать класс, перегрузить для него операции, указанные в варианте.

2. Определить исключительные ситуации.

3. Предусмотреть генерацию исключительных ситуаций.

**Описание класса**

Класс- контейнер ВЕКТОР с элементами типа int.

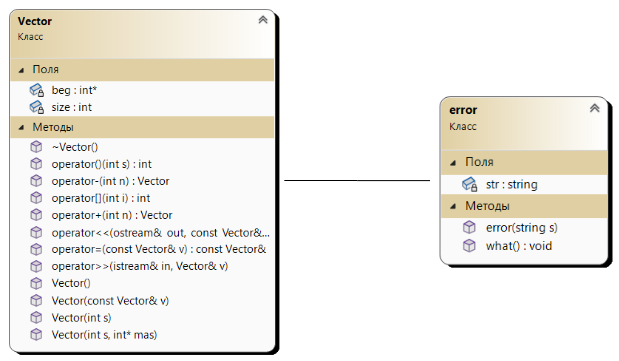
Реализовать операции:

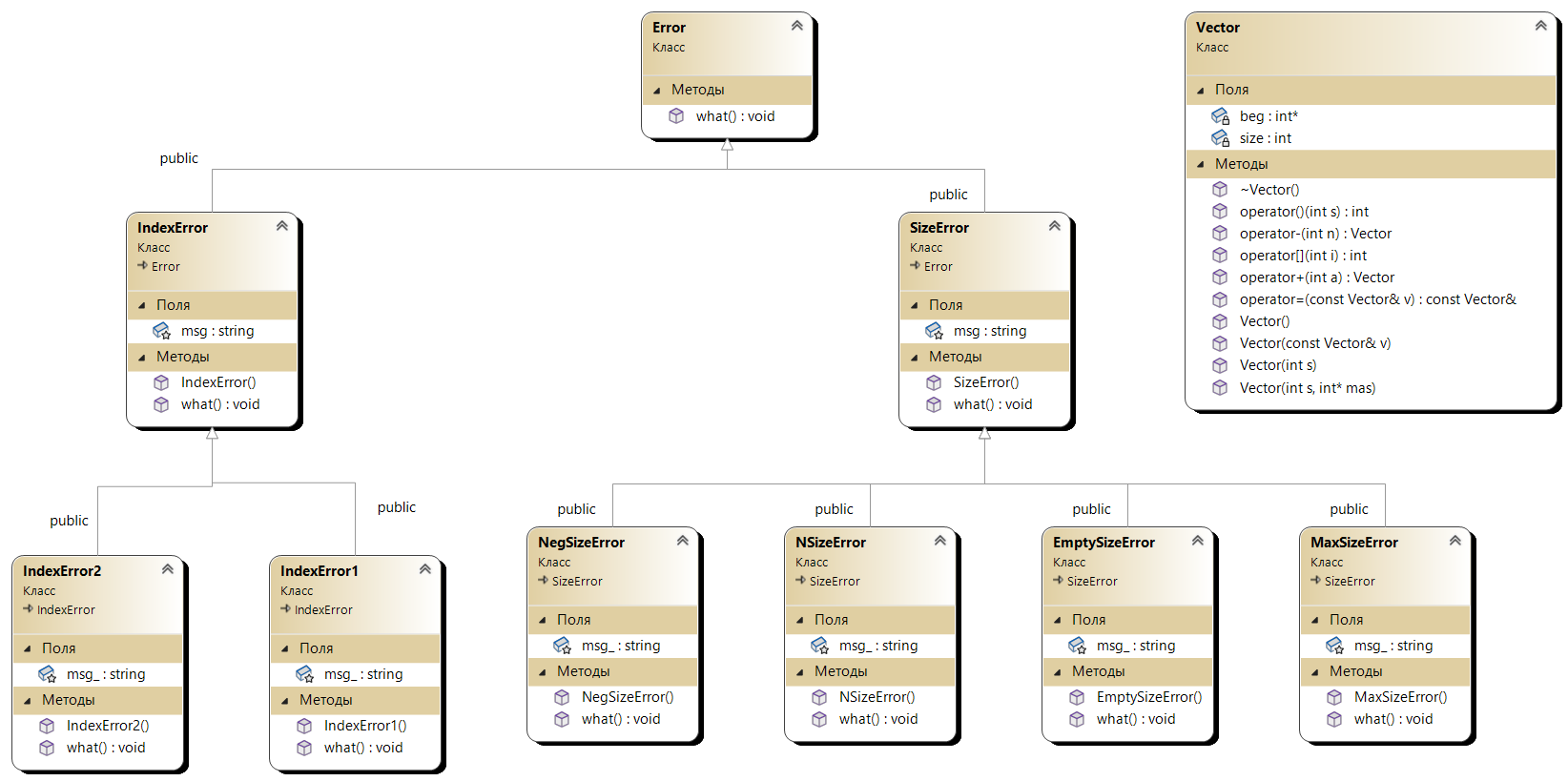
[]– доступа по индексу;

int() – определение размера вектора; - n – удаляет n элементов из конца вектора;

+ n - добавляет n элементов в конец вектора.

**UML-диаграмма**





**Результат работы программы**

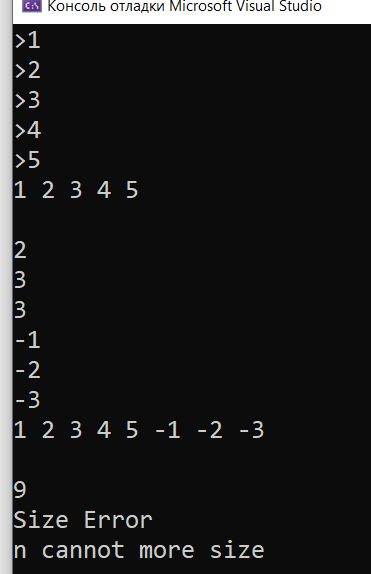
****

Рисунок 2 – результат работы программы

**Контрольные вопросы**

1. Что представляет собой исключение в С++?

Исключение – это непредвиденное или аварийное событие, делающее невозможным дальнейшее выполнение программы по базовому алгоритму.

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

2. На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?

1) обнаружение аварийной ситуации (неизвестно как обрабатывать);

2) обработка аварийной ситуации (неизвестно, где она возникла).

Достоинства:

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

2) не требуется возвращать значение в вызывающую функцию

3. Какой оператор используется для генерации исключительной ситуации?

throw <выражение>, где <выражение> либо константа, либо переменная некоторого типа, либо выражение некоторого типа.

4. Что представляет собой контролируемый блок? Для чего он нужен?

Служебное слово try позволяет выделить в любом месте исполняемого текста программы так называемый контролируемый блок. Блок try используется для проверки возникновения исключения, соответственно в блок try помещается та часть кода, в которой возможно возникновение исключения.

try { операторы }

Свойства блока try:

- Связан с одним или несколькими операторами catch

- Все переменные объявленные внутри try являются локальными

5. Что представляет собой секция-ловушка? Для чего она нужна?

После блока try обязательно прописывается один или несколько блоков catch, которые обычно называют обработчиками исключений, или секциями-ловушками. Форма записи секции-ловушки следующая:

catch (спецификация\_параметра\_исключения) { /\* блок обработки \*/}

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

Форма записи:

сatch(<тип> <имя объекта>) - при такой форме записи оператор принимает сам объект, и переданный объект может использоваться в самом блоке catch. Принять можно по ссылке, копию объекта или указатель на объект.

catch(<тип объекта>) - при такой форме записи оператор не принимает сам объект, соответственно переданный объект не получится использовать в блоке catch.

catch(…) - при такой форме записи оператор принимает любые объекты, передаваемые оператором throw.

7. Какой стандартный класс можно использовать для создания собственной иерархии исключений?

Базовым классом для всех исключений является класс Exception, соответственно для создания своих типов можно унаследовать данный класс.

8. Каким образом можно создать собственную иерархию исключений?

Для этого необходимо:

1. Подключить заголовок #include <expection>

2. Определить собственную функцию с прототипом void F() для подмены стандартной функции terminate().

3. Указать имя этой функции в вызове функции set\_terminate(F);

После этого вместо terminate() будет вызываться наша функция F(). Такая функция не должна возвращать управление оператором return или генерировать исключение throw(),она может только завершить программу

функцией exit() или abort(). Аналогично реализуется подмена стандартной функции unexpected(): set\_ unexpected(F);

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

Если в заголовке спецификация указана, то считается, что функция генерирует те исключения, которые явно указаны в этом списке. Ответ: int, double.

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

Если в заголовке спецификация исключений не указана,

считается, что функция может порождать любое исключение

11. В какой части программы может генерироваться исключение?

Исключение генерируется в той части кода, где используется контролируемый блок try(). Однако исключение не локализуется в блоке, где использован оператор его генерации. Исключение как объект возникает в точке генерации, распознается в контролируемом блоке и передается в обработчик исключений. Только после обработки оно может исчезнуть 12. Написать функцию, которая вычисляет площадь треугольника по трем сторонам (формула Герона).

Функцию реализовать в 4 вариантах:

- без спецификации исключений;

int Geron(int a, int b, int c) {

int p = (a+b+c)/2;

return sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

- со спецификацией throw();

int Geron(int a, int b, int c) throw() {

int p = (a+b+c)/2;

return sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

int main() {

int a, b, c;

cin >> a >> b >> c;

try {Geron(a, b, c);}

catch(...) {cerr << “Exception occured”<<endl;}

- с конкретной спецификацией с подходящим стандартным исключением;

int Geron(int a, int b, int c) throw(invalid\_argument){

if (a==0||b==0||c==0||(a+b)<c||(a+c)<b||(b+c)<a)

throw invalid\_argument("Wrong sides");

int p = (a+b+c)/2;

return sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

- спецификация с собственным реализованным исключением.

#include <stdexcept>

class MyExcept {

string msg;

public:

MyExcept(const string &s) : msg(s) {}

const char \*message() const { return msg.c\_str(); }

};

int Geron(int a, int b, int c) throw(MyExcept) {

if (a==0||b==0||c==0||(a+b)<c||(a+c)<b||(b+c)<a)

throw MyExcept("Wrong sides");

int p = (a+b+c)/2;

return sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

int main() {

int a, b, c;

cin >> a >> b >> c;

try {Geron(a, b, c);}

catch (MyExcept &e) {

cout << e.message() << endl;

}