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:

Бобриков Михаил Александрович

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.

Вариант 3.

Класс - контейнер ВЕКТОР с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

++ – добавляет элемент в вектор (постфиксная операция добавляет элемент в конец, префиксная в начало)

Варианты реализации 3, 1.

**Словесное описание исключительных ситуаций**

Исключительные ситуации генерируются:

1, 2 – в операции [] – при попытке обратиться к элементу с номером меньше 0 или больше текущего размера множества;

3 – +n – при попытке перейти к элементу, который находится вне границ множества.

**Определение класса *error* или иерархии пользовательских классов для определения исключительных ситуаций**

Класс Error

Методы типа public:виртуальная функция what().

Класс IndexError

Поля типа private: msg-строчная переменная.

Методы типа public:Функция IndexError(), и виртуальная функция what().

Класс SizeError

Поля типа private: msg-строчная переменная.

Методы типа public:Функция SizeError(), и виртуальная функция what().

Класс MaxSizeError

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция MaxSizeError(), и виртуальная функция what().

Класс EmptySizeError

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция EmptySizeError(), и виртуальная функция what().

Класс IndexError1

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция IndexError1(), и виртуальная функция what().

Класс IndexError2

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция IndexError2(), и виртуальная функция what().

**Класс контейнер**

Класс Vector:

Поля типа private: size: int, beg: \*int.

Методы типа public: 2 вида конструкторов (с параметрами, копирования), деструктор, перегруженные операторы присваивания(=), доступа по индексу([]), оператор, возвращающий длину вектора(()), перегруженный оператор для добавления к вектору числа(+), перегруженный оператор инкремента(++), перегруженные операторы ввода и вывода.

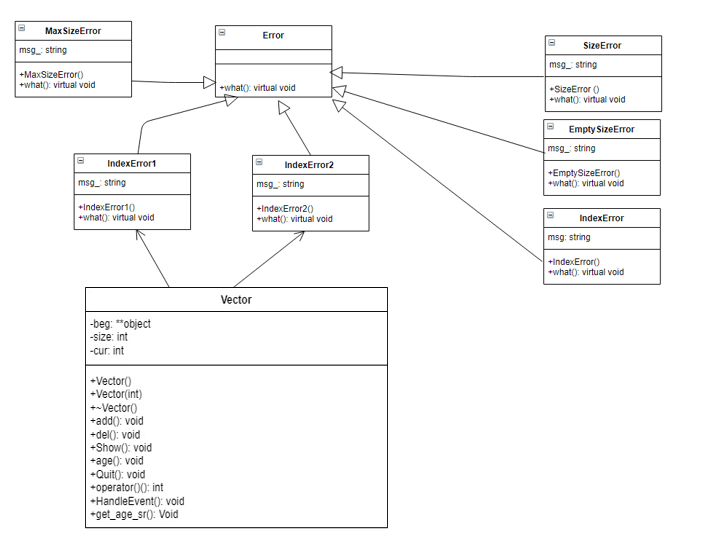
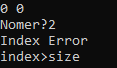
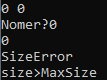


Рис. 1. UML-диаграмма классов

**Вывод программы**

****

****

****

**Контрольные вопросы**

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

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.

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

Обнаружение аварийной ситуации.

Обработка аварийной ситуации.

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

throw.

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

try{

//Блок программы

}

Именно в нем должны создаваться исключения, поэтому генератор исключений throw работает только в нем.

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

сatch () *{*

//Блок программы

*}*

Нужен для обработки исключений.

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

Сatch (спецификация исключения), где спецификация исключения может иметь три формы:

1) (тип имя)

2) (тип)

3) (…)

Формы 1 и 2 обрабатывают конкретные исключения, а форма 3 перехватывает все исключения, такую ловушку надо помещать последней, тогда она будет обрабатывать все исключения, которые еще не были обработаны.

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

Class exception

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

Для создания собственной иерархии исключений надо объявить свой базовый класс исключение, например: class Error{};

Остальные классы будут наследниками этого класса, аналогично тому, как это сделано в иерархии стандартных исключений: class Child\_Error:public Error{}; class Parents\_Error:public Error{};

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

Только исключения типа int и double.

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

Исключения абсолютно любого типа.

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

В операторе throw.

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

* Без спецификации исключений

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

int S, P;

P = (a + b + c) / 2;

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}

* Со спицификатором throw

double geron(int a, int b, int c) {

if (a \* b \* c < 0) throw 1;

double P;

double S;

P = ((a + b + c) / 2);

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}

* C собственным реализованным исключением.

void F() {

exit(1);

}

void set\_terminate() {

F();

}

double geron(int a, int b, int c) {

if (a \* b \* c < 0) set\_terminate();

double P;

double S;

P = ((a + b + c) / 2);

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}

* С конкретной спецификацией с подходящим стандартным исключением.

double geron(int a, int b, int c) {

if (a \* b \* c < 0) terminate();

double P;

double S;

P = ((a + b + c) / 2);

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}