Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Лабораторная работа №9»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

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

**Вариант 8:**

Класс-контейнер МНОЖЕСТВО с элементами типа int.

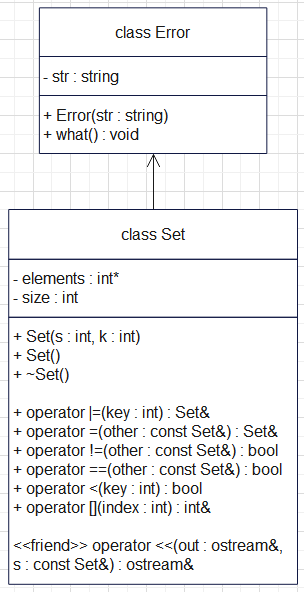
Реализовать операции:

[] -доступа по индексу;

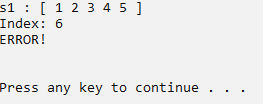
== - проверка на равенство;

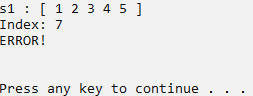
< число - принадлежность числа множеству.

**UML диаграмма:**



**Скриншот результата выполнения программы**





**Ответы на вопросы:**

1. Что представляет собой исключение в C++?

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

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

**Ответ:** Исключения позволяют разделить процесс на 2 части:

1. Обнаружение аварийной ситуации (неизвестно как обрабатывать);
2. Обработка аварийно ситуации (неизвестно, где она возникла).

Достоинства такого подхода:

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?

**Ответ:** Оператор throw <выражение>, где <выражение> -

* либо константа,
* либо переменная некоторого типа,
* либо выражение некоторого типа.

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

**Ответ:**

try {

…

throw выражение

…

Контролируемый блок try{} используется для проверки возникновения исключения.

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

**Ответ:** Секция-ловушка catch( спецификация исключения ) нужна для обработки исключений.

1. Какие формы может иметь спецификация исключения в секции ловушке?

**Ответ:** Спецификация исключения может иметь три формы:

1. (тип имя)
2. (тип)
3. (...)
4. В каких ситуациях используются эти формы?

**Ответ:** Формы 1 и 2 обрабатывают конкретные исключения, а форма 3 перехватывает все исключения.

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

**Ответ:** Можно определять собственные исключения, унаследовав их от класса exception.

1. Каким образом можно создать собственную иерархию исключений?

**Ответ:** 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{};

class BaseException: public exception{};

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

**Ответ:** Функция может генерировать исключения типа int и double.

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

**Ответ:** Функция исключений не генерирует.

1. В какой части программы может генерироваться исключение?

**Ответ:** Исключение генерируется внутри оператора try{};

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

Функцию реализовать в 4 вариантах:

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

**Ответ:**

Без спецификатора исключений:

double Heron(double a, double b, double c) {

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

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

}

Со спецификацией throw():

double Heron(double a, double b, double c) throw() {

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

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

}

С конкретной спецификацей с подходящим стандартным исключением:

#include <stdexcept>

double Heron(double a, double b, double c) throw(invalid\_argument) {

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

throw invalid\_argument(“Invalid triangle sides”);

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

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

}

Спецификация с собственным реализованным исключением:

#include <exception>

class InvalidTriangleException : public std::exception {

public:

virtual const char\* what() const throw() {

return “Invalid triangle sides”;

}

};

double Heron(double a, double b, double c) throw(InvalidTriangleException) {

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

throw InvalidTriangleException();

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

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

}