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

по лабораторной работе №9

Тема: «Программа, управляемая событиями.»

Выполнил

Студент группы РИС-22-2б

Чиртулов М.В.

Проверил доц. 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 14:

Класс- контейнер СПИСОК с ключевыми значениями типа

int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера списка;

\* число – умножает все элементы списка на число;

- n – переход влево к элементу с номером n.

# Контрольные вопросы

1. Что представляет собой исключение в С++?

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

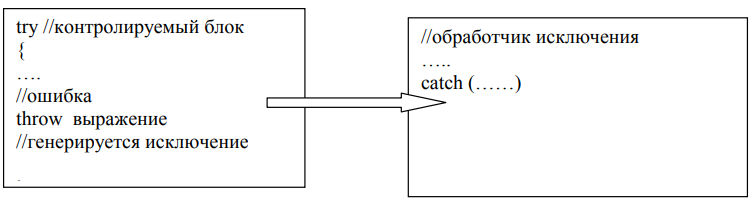
2. На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?

**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.**

**Общая схема:  
**

3. Какой оператор используется для генерации исключительной ситуации?

**Исключение генерируется оператором throw <выражение>, где <выражение> -**

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

**Тип объекта-исключения может быть как встроенным, так и определяемым пользователем. Для представления исключений часто используют пустой класс:**

**class ZeroDevide{};**

**Генерация исключения будет выглядеть:**

**throw ZeroDevide();//вызывается конструктор без параметров**

**или**

**throw new ZeroDevide();**

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

**Проверка возникновения исключения делается с помощью оператора try, с которым неразрывно связаны одна или несколько блоков обработки исключений — catch. Оператор try объявляет в любом месте программы контролируемый блок, который имеет следующий вид:**

**try { */\* контролируемый блок \*/* }**

**Контролируемый блок, помимо функции контроля, обладает функциями обычного блока: все переменные, объявленные внутри него, являются локальными в этом блоке и не видны вне его.**

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

**После блока try обязательно прописывается один или несколько блоков catch. Блок кода после catch предложения является обработчиком исключений. Это обработчик, который перехватывает исключение, которое возникает, если типы в throw выражениях и catch совместимы. 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?

**Спецификация исключения может иметь три формы:**

**1) (тип имя)**

**2) (тип)**

**3) (…)**

**Тип – это встроенный тип или тип, определенный программистом.**

**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.**

**Форма 1 означает, что объект передается в блок обработки, чтобы его каким-то образом там использовать, например, для вывода информации в сообщении об ошибке. Примеры:**

* **catch( exception e) // по значению**
* **catch( exception &e) // по ссылке**
* **catch( const exception &e) // по константной ссылке**
* **catch( exception \*e) //по указателю**

**Лучше всего передавать объект по ссылке, т. к. при этом не создается временный объект-исключение.**

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

**В составе стандартной библиотеки С++ реализован ряд стандартных исключений, которые организованы в иерархию классов.**

**Эта иерархия может служить основой для создания собственных классов исключений и иерархии исключений. Можно определять собственные исключения, унаследовав их от класса *exception*.**

8. Каким образом можно создать собственную иерархию исключений?

**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.**

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

**Для каждой функции, метода, конструктора или деструктора можно в заголовке указать спецификацию исключений. Если в заголовке спецификация исключений не указана, считается, что функция может порождать любое исключение, если указана, то считается, что функция генерирует те исключения, которые явно указаны в этом списке.**

**Следовательно, функция f1() может генерировать исключения типа int и double.**

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

**Если спецификация имеет вид такой вид, то считается, что функция исключений не генерирует.**

11. В какой части программы может генерироваться исключение?

**Исключение могут генерируется внутри оператора try { }.**

12. Написать функцию, которая вычисляет площадь треугольника по трем сторонам

(формула Герона).

Функцию реализовать в 4 вариантах:

**без спецификации 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 triangleArea(double a, double b, double c) throw() {**

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

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

**}**

**с конкретной спецификацией с подходящим стандартным исключением;**

**#include <stdexcept>**

**double triangleArea(double a, double b, double c) throw(std::invalid\_argument) {**

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

**throw std::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 triangleArea(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));**

**}**

**UML-диаграмма**

