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: 09.03.04 – «Программная инженерия»

# Лабораторная работа " Обработка исключительных ситуаций."

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

Вариант 8.

Постановка задачи:

Общая:

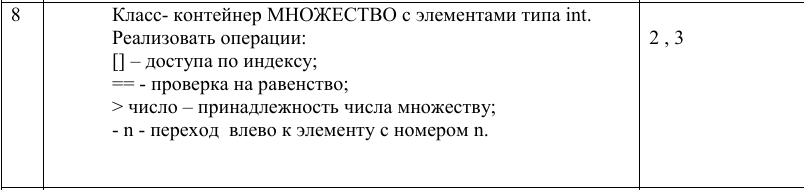
1. Реализовать класс, перегрузить для него операции, указанные в варианте.

2. Определить исключительные ситуации.

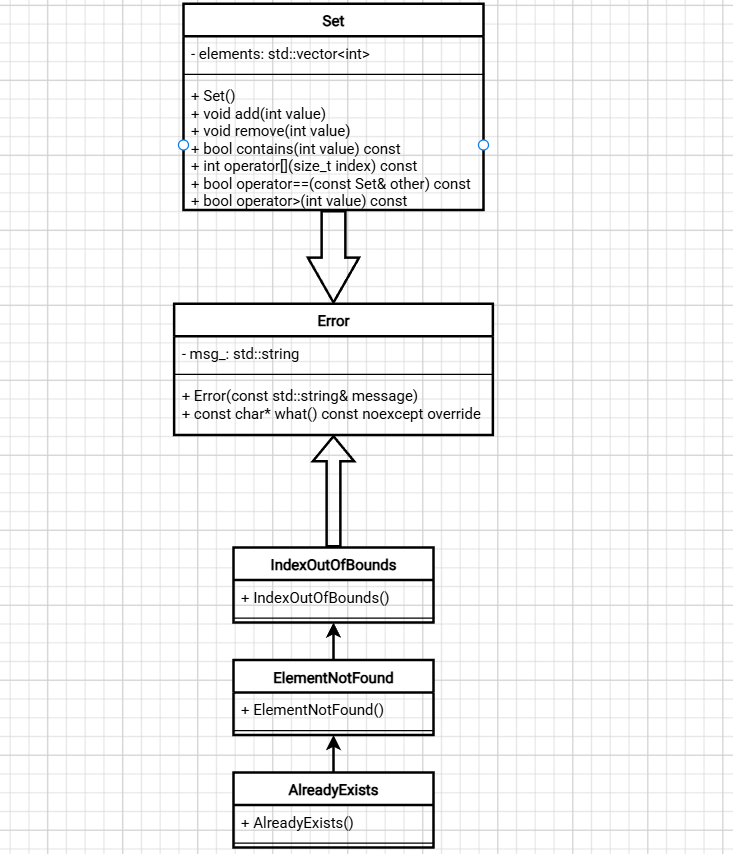
3. Предусмотреть генерацию исключительных ситуаций.

Персональная:





UML – диаграмма:



Программная реализация

**Заголовочный файлы**

**error.h:**

#pragma once

#include <exception>

#include <string>

class Error : public std::exception

{

public:

explicit Error(const std::string& message) : msg\_(message) {}

virtual const char\* what() const noexcept override

{

return msg\_.c\_str();

}

private:

std::string msg\_;

};

class IndexOutOfBounds : public Error

{

public:

IndexOutOfBounds() : Error("Индекс находится за пределами допустимого диапазона") {}

};

class ElementNotFound : public Error

{

public:

ElementNotFound() : Error("Элемент не найден") {}

};

class AlreadyExists : public Error

{

public:

AlreadyExists() : Error("Элемент уже существует в наборе") {}

};

**Set.h:**#pragma once

#include <vector>

#include "error.h"

class Set

{

public:

Set() = default;

void add(int value);

void remove(int value);

bool contains(int value) const;

int operator[](size\_t index) const;

bool operator==(const Set& other) const;

bool operator>(int value) const;

private:

std::vector<int> elements;

};

**Реализация методов**

#include "Set.h"

void Set::add(int value)

{

if (contains(value))

{

throw AlreadyExists();

}

elements.push\_back(value);

}

void Set::remove(int value)

{

auto it = std::find(elements.begin(), elements.end(), value);

if (it != elements.end())

{

elements.erase(it);

}

else

{

throw ElementNotFound();

}

}

bool Set::contains(int value) const

{

return std::find(elements.begin(), elements.end(), value) != elements.end();

}

int Set::operator[](size\_t index) const

{

if (index >= elements.size())

{

throw IndexOutOfBounds();

}

return elements[index];

}

bool Set::operator==(const Set& other) const

{

return elements == other.elements;

}

bool Set::operator>(int value) const

{

return contains(value);

}

**Функция main**

#include <iostream>

#include "Set.h"

int main()

{

Set mySet;

try

{

mySet.add(5);

mySet.add(10);

mySet.add(15);

std::cout << "Элемент с индексом 1: " << mySet[1] << std::endl;

if (mySet > 10)

{

std::cout << "10 входит в множество." << std::endl;

}

mySet.remove(10);

std::cout << "Удален элемент 10." << std::endl;

std::cout << "Доступ к индексу 3: " << mySet[3] << std::endl;

}

catch (const Error& e)

{

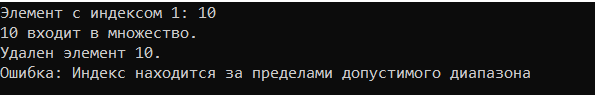
std::cerr << "Ошибка: " << e.what() << std::endl;

}

return 0;

}

Результат работы:



Контрольные вопросы:

1. Что представляет собой исключение в С+?

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

3. Какой оператор используется для генерации исключительной ситуации?

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

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

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

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

8. Каким образом можно создать собственную иерархию исключений?

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

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

11. В какой части программы может генерироваться исключение?

12. 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.

Ответы:

1. **Что представляет собой исключение в C++?**  
   Исключение в C++ — это объект, который генерируется системой при возникновении непредвиденной или аварийной ситуации в программе.
2. **На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?**  
   Исключения позволяют разделить вычислительный процесс на две части:
   1. Обнаружение аварийной ситуации (неизвестно, как обрабатывать).
   2. Обработка аварийной ситуации (неизвестно, где она возникла).  
      Достоинства такого подхода включают удобство использования в многомодульных программах и отсутствие необходимости возвращать значение в вызывающую функцию.
3. **Какой оператор используется для генерации исключительной ситуации?**  
   Исключительная ситуация генерируется с помощью оператора throw.
4. **Что представляет собой контролируемый блок? Для чего он нужен?**  
   Контролируемый блок — это блок кода, обернутый в конструкцию try {}, который используется для проверки возникновения исключений и обработки их в секциях-ловушках catch.
5. **Что представляет собой секция-ловушка? Для чего она нужна?**  
   Секция-ловушка — это часть кода, которая обрабатывает исключения, возникающие в контролируемом блоке. Она нужна для перехвата и обработки исключений, чтобы избежать аварийного завершения программы.
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) — для обработки конкретного исключения.
   2. (тип) — для обработки конкретного типа исключения без доступа к объекту.
   3. (…) — для перехвата всех исключений, должна располагаться последней.
7. **Какой стандартный класс можно использовать для создания собственной иерархии исключений?**  
   Для создания собственной иерархии исключений можно использовать стандартный класс exception.
8. **Каким образом можно создать собственную иерархию исключений?**  
   Для создания собственной иерархии исключений необходимо объявить базовый класс-исключение, а остальные классы наследовать от него. Например:

class BaseException {};

class ChildException1 : public BaseException {};

class ChildException2 : public BaseException {};

1. **Если спецификация исключений имеет вид: void f1() throw(int, double); то какие исключения может порождать функция f1()?**  
   Функция f1() может порождать исключения типов int и double.
2. **Если спецификация исключений имеет вид: void f() throw(); то какие исключения может порождать функция f()?**  
   Функция f() считается такой, что не генерирует исключений.
3. **В какой части программы может генерироваться исключение?**  
   Исключение может генерироваться в любой части программы, где используется оператор throw, например, внутри функций, методов, или в любом блоке кода.

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

### 1). Без спецификации исключений:

#include <cmath>

#include <stdexcept>

double SGeron(double a, double b, double c)

{

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

{

throw std::invalid\_argument("Стороны должны быть положительными и удовлетворять неравенству треугольника.");

}

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

return std::sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

### 2). Со спецификацией throw():

#include <cmath>

#include <stdexcept>

double throwGeron(double a, double b, double c) throw()

{

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

{

throw std::invalid\_argument("Стороны должны быть положительными и удовлетворять неравенству треугольника.");

}

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

return std::sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

### 3). С конкретной спецификацией с подходящим стандартным исключением:

#include <cmath>

#include <stdexcept>

double InvalidGeron(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("Стороны должны быть положительными и удовлетворять неравенству треугольника.");

}

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

return std::sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

### 4). Спецификация с собственным реализованным исключением:

#include <cmath>

#include <iostream>

#include <exception>

// Определение собственного исключения

class TriangleException : public std::exception

{

public:

const char\* what() const throw()

{

return "Ошибка: стороны треугольника должны быть положительными и удовлетворять неравенству треугольника.";

}

};

double EXGeron(double a, double b, double c) throw(TriangleException)

{

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

{

throw TriangleException();

}

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

return std::sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

Ссылка:

**https://github.com/LeonidZhir/-**