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: Лабораторная работа №9: «Обработка исключительных ситуаций».

**Вариант 13**

Выполнил студент ИВТ-22-2б:

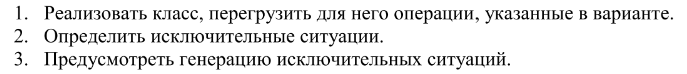
Коняев Александр Сергеевич

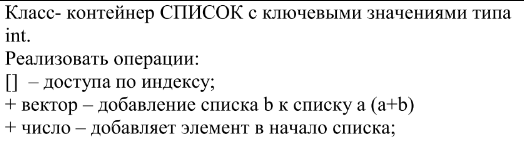
Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**





**Анализ задачи**

Класс Node

Класс LinkedList

* Конструктор с параметрами
* Конструктор без параметров
* Конструктор копирования
* Метод clear
* Метод push\_back
* Метод push\_front
* Метод pop\_back
* Метод pop\_front
* Метод get\_At
* Метод insert
* Метод erase
* Метод print
* Перегруженные операторы: =, [], +, <<, >>

**UML – диаграмма**

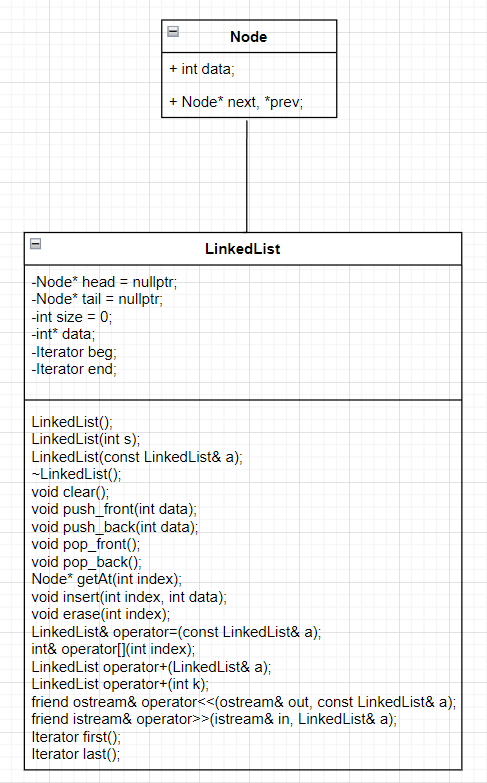


Рисунок 1 – UML-диаграмма.

**Код программы.**

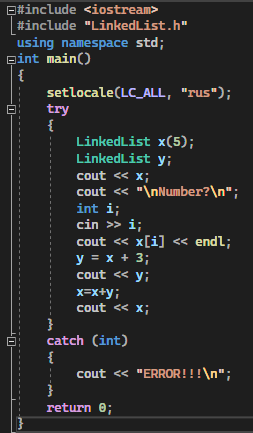


Рисунок 2 – Функция main

**Вывод программы.**

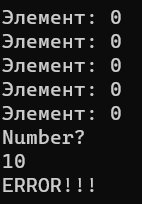
****

Рисунок 3 – Вывод программы

**Ответы на вопросы**

1. Исключение (Exception) в C++ - это механизм обработки ошибок, который позволяет программе отреагировать на неожиданные ситуации в процессе выполнения и перейти к обработке ошибки, не завершая выполнение всей программы.

Когда происходит ошибка в программе, исключение генерируется и передается по стеку вызовов к ближайшему обработчику исключений. Если обработчик не найден, программа аварийно завершается.

Для генерации исключения используется ключевое слово throw, а для обработки исключения - блок try-catch. В блоке try содержится опасный для выполнения код, который может сгенерировать исключение. В блоке catch определяется обработчик исключения, он содержит блок кода, который будет выполнен при возникновении исключения.

Пример кода, иллюстрирующий обработку исключений:

try{

//опасный код, который может сгенерировать исключение

if (x == 0) {

throw "divide by zero"; // генерация исключения

}

int result = y / x;

}

catch(const char\* msg){ // обработчик исключения

std::cerr << msg << std::endl;

}

1. Исключения в C++ обычно делятся на три части:

Генерация исключения - это процесс, когда возникает ошибка и программа создает исключение, чтобы сообщить об этой ошибке.

Выбор обработчика исключения - это процесс выбора блока кода, который будет выполнен при возникновении исключения. Обработчик исключения может быть выбран на основе типа исключения или на основе положения кода.

Обработка исключения - это процесс выполнения блока кода, который был выбран как обработчик исключения. В этом блоке кода может быть предпринята попытка восстановления после возникновения ошибки или же программа может завершиться аварийно.

1. Для генерации исключения в C++ используется оператор throw. Он позволяет создать объект-исключение и передать управление обработчику исключения.
2. Контролируемый блок в C++ представляет собой участок кода, который содержит операции, которые могут вызвать исключение. Этот блок объявляется с помощью ключевого слова try и заканчивается одним или несколькими блоками catch, которые обрабатывают исключения, возникающие внутри блока try.

Контролируемый блок нужен для обработки исключительных ситуаций, которые могут возникнуть при выполнении программы. Использование контролируемого блока позволяет предотвратить аварийное завершение программы и позволяет разработчику обработать возникшее исключение, например, выдав сообщение об ошибке или произвести дополнительную обработку данных.

1. Секция-ловушка (или блок catch) является частью контролируемого блока и предназначена для обработки возникших исключений. Она содержит код, который выполняется в случае, если было выброшено исключение определенного типа.

Секция-ловушка позволяет программисту сделать дополнительную обработку данных, связанных с исключением, и сообщить об ошибке пользователю. Это улучшает общую стабильность и гибкость программы, позволяя ей продолжить работу после возникновения исключительной ситуации.

1. Спецификация исключения в секции ловушки может иметь две формы:

Одно слово «...» - это указывает на то, что секция ловушки обрабатывает любые исключения, которые могут возникнуть в блоке try.

Пример:

try {

// блок кода, который может выбросить исключение

}

catch (...) {

// обрабатываем любые исключения

}

Указание определенного типа исключения, для которого должна быть выполнена обработка.

Пример:

try {

// блок кода, который может выбросить исключение типа std::exception

}

catch (std::exception& e) {

// обрабатываем исключение типа std::exception

}

Первую форму используют, когда необходимо обработать все возможные исключения. Вторую форму используют для выборочной обработки конкретных типов исключений, например, для обработки ошибок ввода-вывода, ошибок работы с памятью и т.д.

1. Для создания собственной иерархии исключений в C++ можно использовать стандартный класс std::exception в качестве базового класса. Данный класс определен в заголовочном файле <exception> и содержит виртуальную функцию what(), возвращающую C-style строку с описанием исключения.

Для создания собственных классов исключений следует унаследовать их от класса std::exception, переопределить метод what(), и, при необходимости, добавить свои собственные поля и методы.

1. Пример кода:

#include <exception>

#include <string>

class MyException: public std::exception {

public:

MyException(const std::string& message): msg(message) {}

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

return msg.c\_str();

}

private:

std::string msg;

};

В данном примере мы создали собственный класс MyException, который наследуется от std::exception. В конструкторе класса мы передаем строку с описанием исключения, которую сохраняем в поле msg. Также мы переопределили метод what(), чтобы он возвращал C-style строку с описанием исключения.

1. Функция f1() может выбрасывать и перехватывать исключения типов int и double. Это означает, что если функция f1() выбрасывает исключение любого другого типа, например std::string, оно не будет поймано в блоке try-catch, и программа завершится аварийно. Также следует заметить, что если функция f1() не выбрасывает исключение типа int или double, но выбрасывает исключение типа производного класса от int или double, это исключение также может быть перехвачено в блоке try-catch.
2. Функция f1() не может выбрасывать исключения. Если она выбросит исключение, то программа будет завершена аварийно.
3. Исключение может быть сгенерировано в любой части программы, в которой происходят ошибки. Это может быть внутри тела функции, в операторах try/except, при работе с файлами, при вводе-выводе данных и т.д. Поэтому важно всегда предусматривать обработку исключений в коде, чтобы избежать неожиданной ошибки и программа могла корректно работать при любых условиях.
4. #include <iostream>

#include <cmath>

#include <stdexcept>

using namespace std;

// Функция без спецификации исключений

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

// Вычисляем полупериметр

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

// Вычисляем площадь по формуле Герона

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

// Функция со спецификацией throw()

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

// Проверяем, что стороны положительные

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

throw "Side length must be positive";

}

// Вычисляем полупериметр

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

// Вычисляем площадь по формуле Герона

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

// Функция с конкретной спецификацией с подходящим стандартным исключением

double triangleAreaStandard(double a, double b, double c) throw(domain\_error) {

// Проверяем, что стороны могут образовать треугольник

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

throw domain\_error("Invalid side lengths");

}

// Вычисляем полупериметр

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

// Вычисляем площадь по формуле Герона

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

// Наше собственное исключение

class MyException : public exception {

public:

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

return "Invalid side lengths";

}

};

// Функция со спецификацией с собственным реализованным исключением

double triangleAreaCustom(double a, double b, double c) throw(MyException) {

// Проверяем, что стороны могут образовать треугольник

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

throw MyException();

}

// Вычисляем полупериметр

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

// Вычисляем площадь по формуле Герона

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}