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: «ООП. Обработка исключительных ситуаций»

Вариант №15

Выполнил студент ИВТ-22-2б:

Кучугова Яна Александровна

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 15:**

Класс- контейнер СПИСОК с ключевыми значениями типа int. Реализовать операции:

[] – доступа по индексу;

int() – определение размера списка;

\* вектор – умножение элементов списков a[i]\*b[i];

+n - переход вправо к элементу с номером n.

Вариант реализации: 1,3

# Исключительные ситуации

Исключительные ситуации генерируются:

1 – в конструкторе с параметром при попытке создать вектор больше максимального размера;

2, 3 – в операции [] – при попытке обратиться к элементу с номером меньше 0 или больше текущего размера вектора;

В остальных случаях – exception.

# Описание классов

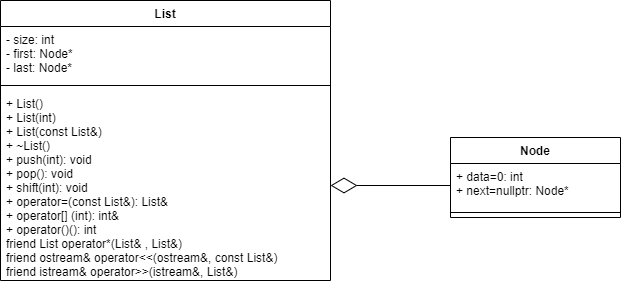
**Вариант 1:**

Информация об исключительных ситуациях передается с помощью стандартного типа данных.

Класс Node (узел списка). Поля с доступом public: int data – поле данных шаблонного типа, Node\* next – указатель на следующий узел.

Класс List (список). Поля с доступом private: int size – размер списка, Node\* first – указатель на первый элемент списка, Node\* last – указатель на последний элемент списка. Методы с доступом public: 3 вида конструкторов и деструктор; методы push() и pop() для добавления и удаления элементов; перегрузка операторов для методов присваивания, доступа по индексу, определения размера списка, умножения элементов списков. Дружественные функции для ввода/вывода списка.

Исключения обозначаются данными типа int и добавляются в файл с описанием методов.

Также описание класса представлено на UML-диаграмме:

**Вариант 3:**

Информация об исключительных ситуациях передается с помощью стандартного типа данных.

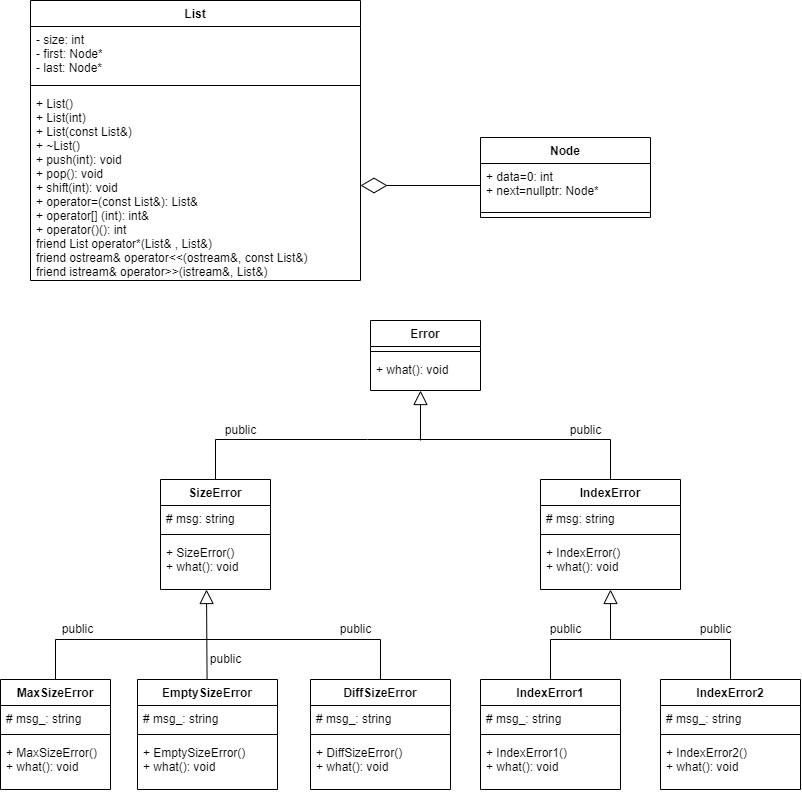
Класс Node (узел списка). Поля с доступом public: int data – поле данных шаблонного типа, Node\* next – указатель на следующий узел.

Класс List (список). Поля с доступом private: int size – размер списка, Node\* first – указатель на первый элемент списка, Node\* last – указатель на последний элемент списка. Методы с доступом public: 3 вида конструкторов и деструктор; методы push() и pop() для добавления и удаления элементов; перегрузка операторов для методов присваивания, доступа по индексу, определения размера списка, умножения элементов списков. Дружественные функции для ввода/вывода списка.

Заголовочный файл Error.h содержащий иерархию классов-ошибок:

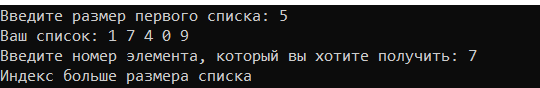
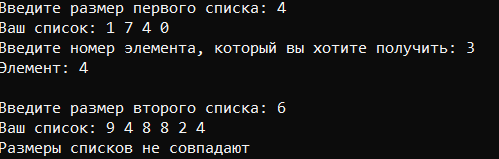
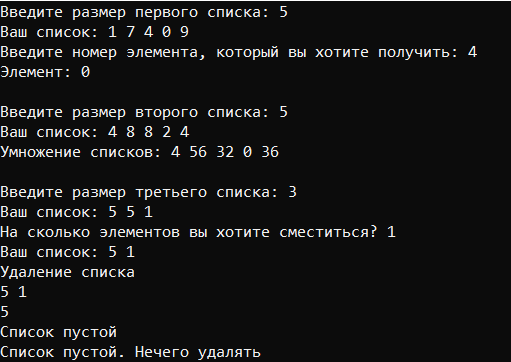
Error – базовый класс; SizeError – ошибка в размере списка, MaxSizeError – превышение максимального размера, EmptySizeError – удаление из пустого списка, DiffSizeError – удаление из пустого списка, IndexError – ошибка в индексе списка, IndexError1 – индекс меньше 0, IndexError2 – индекс больше текущего размера списка.

Также описание классов представлено на UML-диаграмме:



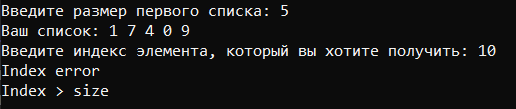
# Результат работы программы

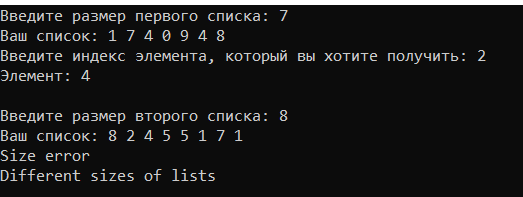
**Вариант 1:**

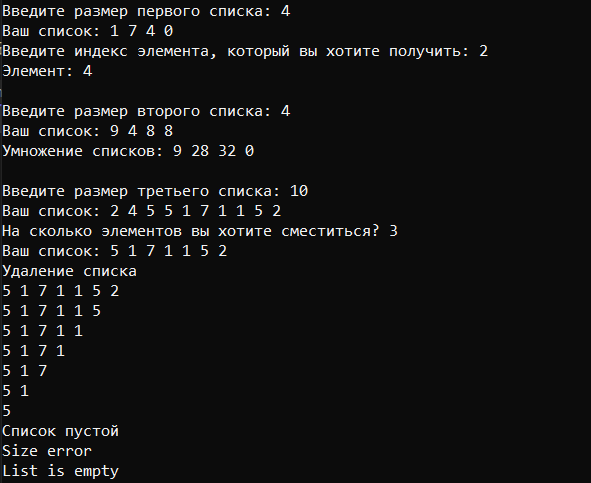
   

**Вариант 3:**









# 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?
2. Обнаружение аварийной ситуации (неизвестно, как обрабатывать).
3. Обработка аварийной ситуации (неизвестно, где возникла).

Достоинства:

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?

throw <выражение>, где <выражение> либо константа, либо переменная некоторого типа, либо выражение некоторого типа.

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

Служебное слово try позволяет выделить в любом месте исполняемого текста программы так называемый контролируемый блок. Блок try используется для проверки возникновения исключения, соответственно в блок try помещается та часть кода, в которой возможно возникновение исключения.

try { операторы }

Свойства блока try:

- Связан с одним или несколькими операторами catch.

- Все переменные объявленные внутри try являются локальными.

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

После блока try обязательно прописывается один или несколько блоков catch, которые обычно называют обработчиками исключений, или секциями-ловушками. Форма записи секции-ловушки следующая:

catch (спецификация\_параметра\_исключения) { /\* блок обработки \*/}

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?

Базовым классом для всех исключений является класс Exception, соответственно для создания своих типов можно унаследовать данный класс.

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?

Исключение генерируется в той части кода, где используется контролируемый блок try(). Однако исключение не локализуется в блоке, где использован оператор его генерации. Исключение как объект возникает в точке генерации, распознается в контролируемом блоке и передается в обработчик исключений. Только после обработки оно может исчезнуть.

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 спецификатором 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;

}