**Министерство транспорта Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования «Российский университет транспорта»**

**(РУТ(МИИТ)**

**Институт управления и цифровых технологий**

**Кафедра «Цифровые технологии управления транспортными процессами»**

**Отчёт**

По лабораторной работе

по дисциплине «Программирование»

Тема: «Виртуальные методы»

Задание № 4

Вариант № 5 (25)

Выполнил:   
 ст. гр. УИС-111:

Чаругин А. М.

Проверил:

Савушкин С. А.

Москва 2022 г.

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc104157202)

[Общие требования 3](#_Toc104157203)

[Вариант задания 3](#_Toc104157204)

[Авторское понимание и формализация задачи 3](#_Toc104157205)

[Текст программы 4](#_Toc104157206)

[Lab4\_work\_4.cpp 4](#_Toc104157207)

[Decimal.h 5](#_Toc104157208)

[Decimal.cpp 6](#_Toc104157209)

[BitSrting.cpp 7](#_Toc104157210)

[BitString.h 8](#_Toc104157211)

[Результаты тестирования 8](#_Toc104157212)

[Выводы 8](#_Toc104157213)

# Постановка задачи

## Общие требования

В каждом упражнении требуется реализовать в том или ином виде определение нового класса.

Создать базовый класс Array с полями: массив типа unsigned char и поле для хранения количества элементов у текущего объекта-массива. Максимально возможный размер массива задается статической константой. Реализовать конструктор инициализации, задающий количество элементов и начальное значение (по умолчанию 0). Реализовать метод доступа к элементу, перегрузив операцию индексирования []. При этом должна выполняться проверка индекса на допустимость.

Реализовать в классе Array виртуальную функцию поэлементного сложения массивов. Реализовать два класса, переопределив виртуальную функцию сложения. Вызывающая программа должна продемонстрировать все варианты вызова виртуальных функций.

## Вариант задания

5) Создать класс Decimal (Класс для работы со знаковыми целыми десятичными числами, использует для представления числа массив из 100 элементов типа unsigned char, каждый из которых является десятичной цифрой. Младшая цифра имеет меньший индекс (единицы - в нулевом элементе массива). Знак представляется отдельным полем sing.) и класс BitString (Класс для работы с битовыми строками не более чем из 100 бит. Битовая строка должна быть представлена массивом типа unsigned char, каждый элемент которого принимает значение 0 или 1.).

## Авторское понимание и формализация задачи

Необходимо реализовать класс Decimal (класс для работы с элементами типа unsigned char. Максимальный размер массива задается статической константой, а также реализовать конструктор инициализации, задающий количество элементов и начальное значение (по умолчанию 0)), продемонстрировать переопределенный оператор сложения на примере инициализации объекта, являющимся суммой двух других объектов этого же класса Decimal. Перегрузить оператор индексирования [], сделать проверку на допустимость (для класса Decimal и BitString). Реализовать класс BitString, также продемонстрировать переопределенный оператор сложения на примере суммы двух других объектов.

Класс Decimal в конструкторе принимает значение типа char(число в десятичной системе счисления), чтобы в дальнейшем работать как со значением типа int. Таким образом, можно создавать объекты класса Decimal и оперировать операциями сложения и присваивания как с обычными десятичными числами.

Класс BitString представляет собой класс, способный принимать в конструктор символы типа string, также переопределены операторы сложения и присваивания. Оператор сложения представляет собой сложение элементов char напрямую (используется их значения в Int).

# Текст программы

## Lab4\_work\_4.cpp

#include "Decimal.h"

#include "BitString.h"

#include <ctime>

**int** main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

std::cout << "\t Лабораторная работа #4,Тема: «�'иртуальные методы»\n";

Decimal firstDecimal("10");

std::cout << "The first object of Decimal class(number): " << firstDecimal;

Decimal secondDecimal("20");

std::cout << "The second object of Decimal class(number): " << secondDecimal;

Decimal sumOfFirstAndSecondDecimal = firstDecimal + secondDecimal;

std::cout << "The sum of first and second object of Decimal class(number): " << sumOfFirstAndSecondDecimal << "\n";

BitString oneBitString("a");

std::cout << "The first object of BitString class(char symbol(number)): " << oneBitString;

BitString twoBitString("b");

std::cout << "The second object of BitString class(char symbol(number)): " << twoBitString;

BitString sumOfBitString = oneBitString + twoBitString;

std::cout << "The sum of objects of BitString class(char symbol(number)): " << sumOfBitString;

std::cout << "End.\n";

**return** 0;

}

## Decimal.h

#ifndef DECIMAL\_H

#define DECIMAL\_H

#include <iostream>

#include <string>

**class** Decimal

{

**static** **const** **int** MAXsize = 100;

**unsigned** **char** val[MAXsize];

**unsigned** size;

**public**:

Decimal(**const** std::string& value);

Decimal& **operator**=(**const** std::string& value);

**unsigned** **operator**[](**unsigned** i);

Decimal **operator**+(**const** Decimal& dec);

**unsigned** GetSize();

**friend** std::ostream& **operator** << (std::ostream& os, **const** Decimal& digit);

};

#endif

#**pragma once**

## Decimal.cpp

#include "Decimal.h"

Decimal::Decimal(**const** std::string& value = "") : size(0)

{

\***this** = value;

}

Decimal& Decimal::**operator**=(**const** std::string& value)

{

size = value.size();

**for** (**unsigned** i = 0; i < size; ++i)

{

val[i] = value[size - i - 1] - 48;

}

**return** \***this**;

}

**unsigned** Decimal::**operator**[](**unsigned** i)

{

**unsigned** **char** digit = 0;

**if** (i < size)

{

digit = val[size - i];

}

**return** digit;

}

Decimal Decimal::**operator**+(**const** Decimal& dec)

{

Decimal result;

**unsigned** mod = 0;

result.size = (size < dec.size) ? dec.size : size;

**for** (**unsigned** i = 0; i < result.size; ++i)

{

result.val[i] = val[i] + dec.val[i] + mod;

mod = result.val[i] / 10;

result.val[i] %= 10;

}

**if** (mod)

{

result.val[result.size++] = 1;

}

**return** result;

}

**unsigned** Decimal::GetSize()

{

**return** size;

}

std::ostream& **operator** << (std::ostream& os, **const** Decimal& digit)

{

**for** (**unsigned** i = digit.size - 1; i < digit.size; --i)

{

os << (**unsigned**)digit.val[i];

}

os << "\n";

**return** os;

}

## BitSrting.cpp

#include "BitString.h"

BitString::BitString(**const** std::string& value = "") : size(0)

{

\***this** = value;

}

BitString& BitString::**operator**=(**const** std::string& value)

{

size = value.size();

**for** (**unsigned** i = 0; i < size; ++i)

{

val[i] = value[size - i - 1] - 48;

}

**return** \***this**;

}

**unsigned** BitString::**operator**[](**unsigned** i)

{

**unsigned** **char** digit = 0;

**if** (i < size)

{

digit = val[size - i];

}

**return** digit;

}

BitString BitString::**operator**+(**const** BitString& dec)

{

**int** max\_len = std::max(GetSize(), dec.GetSize());

**int** min\_len = std::min(GetSize(), dec.GetSize());

**for** (**int** index = 0; index < max\_len; index++)

{

val[index] = (index < min\_len) ? (val[index] + dec.val[index]) :

((min\_len == GetSize()) ? dec.val[index] : val[index]);

}

**return** \***this**;

}

**unsigned** **int** BitString::GetSize() **const**

{

**return** size;

}

std::ostream& **operator** << (std::ostream& os, **const** BitString& digit)

{

**for** (**unsigned** i = digit.size - 1; i < digit.size; --i)

{

os << (**unsigned**)digit.val[i];

}

os << "\n";

**return** os;

}

## BitString.h

#ifndef BITSTRING\_H

#define BITSTRING\_H

#include <iostream>

#include <string>

**class** BitString

{

**static** **const** **int** MAXsize = 100;

**unsigned** **char** val[MAXsize];

**unsigned** **int** size = 0;

**public**:

BitString(**const** std::string& value);

BitString& **operator**=(**const** std::string& value);

**unsigned** **operator**[](**unsigned** i);

BitString **operator**+(**const** BitString& dec);

**unsigned** **int** GetSize() **const**;

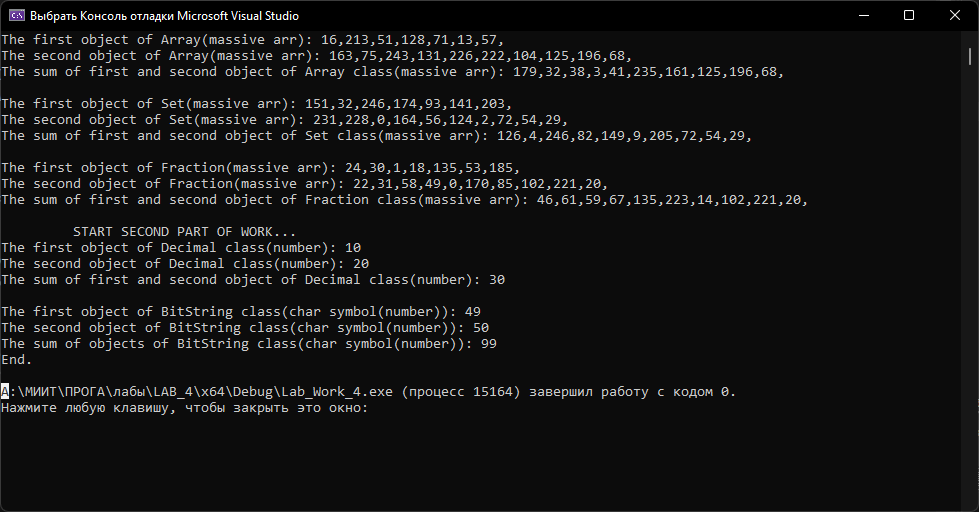
**friend** std::ostream& **operator** << (std::ostream& os, **const** BitString& digit);

};

#endif

#**pragma once**

# Результаты тестирования



# Выводы

В ходе выполнения Четвертой лабораторной работы я узнал, как устроены и как применять виртуальные методы в языке С++.