МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра АСУ

**Отчет**

По лабораторной работе

по дисциплине   
«Языки программирования высокого уровня»

Тема: «НАСЛЕДОВАНИЕ И ВИРТУАЛЬНЫЕ МЕТОДЫ»

Задание № 6

Вариант № 4

Выполнил:   
 ст. гр. УВА-111:

Букин А. А.

Проверил:

Савушкин С.А.

Москва 2020 г.

**Содержание:**

[**Постановка задачи 3**](#_heading=h.30j0zll)

[**Общие требования 3**](#_heading=h.1fob9te)

[**Задание 3**](#_heading=h.3znysh7)

[**Авторское понимание и формализация задачи**](#_heading=h.2et92p0) **3**

[**Текст программы**](#_heading=h.tyjcwt) **4**

[**Результаты тестирования**](#_heading=h.3dy6vkm) **9**

[**Вывод**](#_heading=h.1t3h5sf) **9**

**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.

**Задание**

Создать класс Money (Класс для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа unsigned char - для копеек.) 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**

Мы создаем класс Array и наследуем Money и BitString для работы с ними. Реализуем виртуальные методы и демонстрируем их работу. И получившийся результат выводим на экран.

**Текст программы**

**Файл – lab6.cpp**

**#include <string>**

**#include <iostream>**

**#include <sstream>**

**using namespace std;**

**#define MAX\_SIZE 255**

**class Array {**

**protected:**

**unsigned char \*array;**

**unsigned char initial\_value; // значение по умолчанию для элемента**

**unsigned char size; // размер массива**

**public:**

**Array(unsigned char arr\_size); // конструктор инициализации**

**Array(unsigned char arr\_size, unsigned char init\_val); // конструктор инициализации**

**Array(const Array &a); // конструктор копирования**

**Array() : size(0), initial\_value(0), array(nullptr) {}; // конструктор без аргументов**

**virtual unsigned char& operator[](int index); // оператор индексирования**

**virtual void sum\_with(Array &a); //функция поэлементного сложения**

**virtual string toString();**

**};**

**Array::Array(unsigned char arr\_size) {**

**array = new unsigned char [arr\_size];**

**size = arr\_size;**

**initial\_value = 0;**

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

**array[i] = initial\_value;**

**}**

**Array::Array(unsigned char arr\_size, unsigned char init\_val) {**

**array = new unsigned char [arr\_size];**

**size = arr\_size;**

**initial\_value = init\_val;**

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

**array[i] = initial\_value;**

**}**

**Array::Array(const Array &a) {**

**array = new unsigned char [a.size];**

**size = a.size;**

**initial\_value = a.initial\_value;**

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

**array[i] = initial\_value;**

**}**

**// оператор индексирования**

**unsigned char& Array::operator[](const int index) {**

**if (index < 0 || index >= size)**

**return (initial\_value);**

**return array[index];**

**}**

**string Array::toString() {**

**ostringstream s;**

**s << "Массив: { ";**

**if (size > 0) {**

**for (int i = 0; i < size - 1; i++)**

**s << (int)array[i] << ", ";**

**s << (int)array[size - 1];**

**}**

**s << " }";**

**return s.str();**

**}**

**void Array::sum\_with(Array &a) {**

**int max\_size = size > a.size ? size : a.size;**

**unsigned char \*max\_arr = new unsigned char [max\_size];**

**for (int i = 0; i < max\_size; i++)**

**{**

**max\_arr[i] = (\*this)[i] + a[i];**

**}**

**size = max\_size;**

**delete[] array;**

**array = max\_arr;**

**}**

**class BitString : public Array {**

**public:**

**BitString(unsigned char arr\_size): Array(arr\_size) {}; // конструктор инициализации**

**BitString() : Array(0) {}; // конструктор без аргументов**

**virtual unsigned char& operator[](int index); // оператор индексирования**

**virtual void sum\_with(Array &a); //функция поэлементного сложения**

**virtual string toString();**

**};**

**string BitString::toString() {**

**ostringstream s;**

**s << "BitString: \'";**

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

**s << (int)array[i];**

**s << "\'";**

**return s.str();**

**}**

**// оператор индексирования**

**unsigned char& BitString::operator[](const int index) {**

**if (index < 0 || index >= size)**

**return (initial\_value);**

**return array[index];**

**}**

**void BitString::sum\_with(Array &a) {**

**Array::sum\_with(a);**

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

**array[i] = array[i] ? 1 : 0;**

**}**

**int main()**

**{**

**cout << "┍━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┑" << endl;**

**cout << "| Демонстрация работы функций |" << endl;**

**cout << "┕━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┙" << endl << endl;**

**Array a1 = Array();**

**Array a2 = Array(4);**

**Array a3 = Array(5, 7);**

**cout << "По умолчанию : " << a1.toString() << endl;**

**cout << "С размером 4 : " << a2.toString() << endl;**

**cout << "С заполнением 7 : " << a3.toString() << endl;**

**BitString b1 = BitString(2);**

**BitString \*b2 = new BitString(2);**

**Array a4 = BitString(2);**

**Array \*a5 = new BitString(2);**

**cout << endl;**

**cout << "b1 : " << b1.toString() << endl;**

**cout << "b2 : " << b2->toString() << endl;**

**cout << "a4 : " << a4.toString() << endl;**

**cout << "a5 : " << a5->toString() << endl; // в объекте типа Array лежит BitString**

**// вызывается виртуальная функция BitString из объекта Array**

**Array \*a6 = new BitString(5);**

**Array \*a7 = new Array(5, 9);**

**(\*a7)[1] = 0;**

**(\*a7)[3] = 0;**

**cout << endl;**

**cout << "a6 : " << a6->toString() << endl;**

**cout << "a7 : " << a7->toString() << endl;**

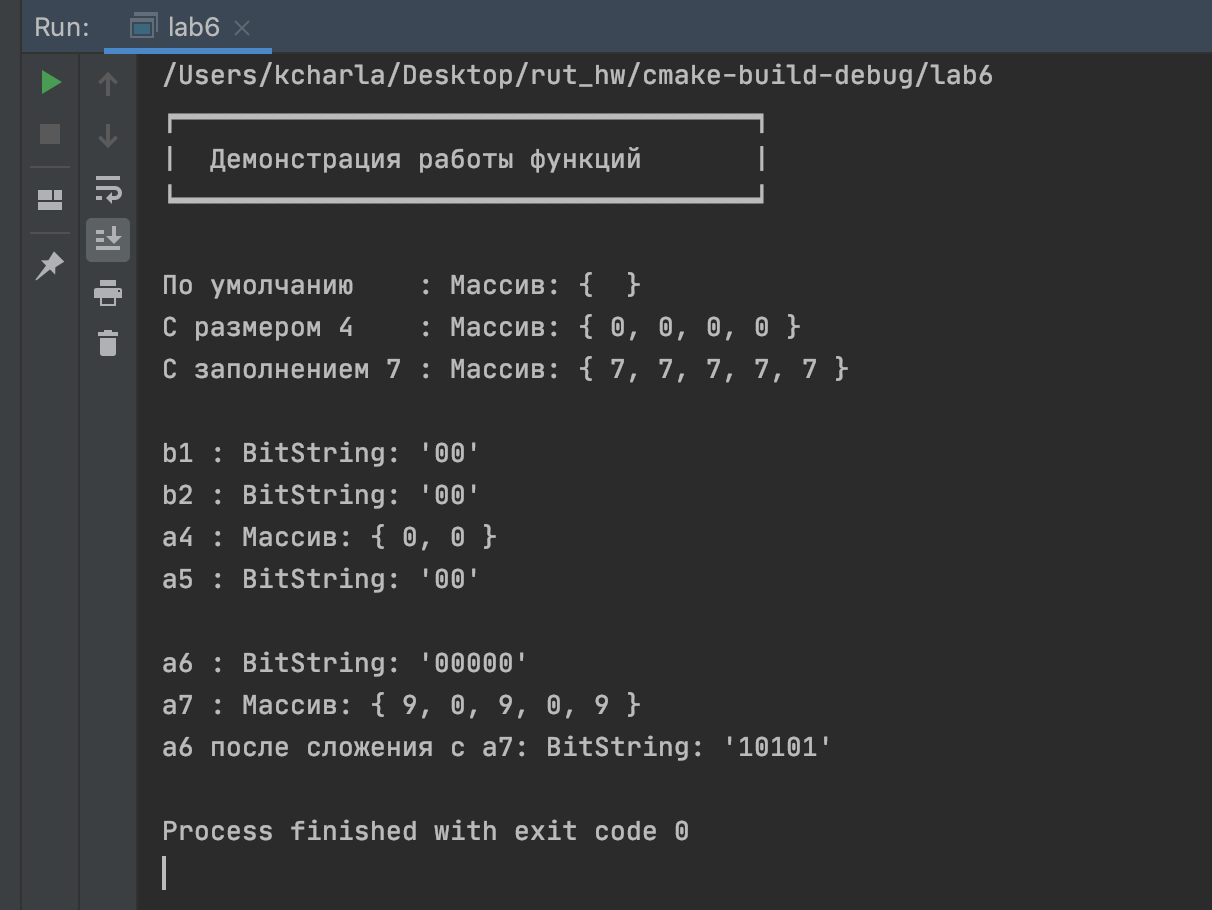
**a6->sum\_with(\*a7); // сработала виртуальная функция сложения из BitString**

**cout << "a6 после сложения с a7: " << a6->toString() << endl;**

**return (0);**

**}**

**Результаты тестирования**

****

**Вывод**

Программа выдает ожидаемые результаты и работает корректно. Поскольку полученные результаты совпали с ожидаемыми, программу можно считать правильной. Я научился создавать в классе объектов виртуальные методы.