МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Отчет к лабораторной работе № 4

Тема: **Объектно-ориентированный подход. Наследование**

Дисциплина: **«Информационные технологии и программирование»**

**Выполнил:**

Студент 1 курса группы ПИЖ-б-о-23-1(2) направление подготовки: 09.03.04 «Программная инженерия»

Сабодаш Алексей Игоревич

**Проверила:**

Новикова Е. Н.

Ставрополь 2024

**Вариант 8**

**Задание 1:**

Описать базовый класс СТРОКА.

Обязательные поля класса:

• указатель на char — хранит адрес динамически выделенной памяти для размещения символов строки;

• значение типа int — хранит длину строки в байтах.

Обязательные методы:

• конструктор без параметров;

• конструктор, принимающий в качестве параметра С-строку (заканчивается

нулевым байтом);

• конструктор, принимающий в качестве параметра символ;

• конструктор копирования;

• получение длины строки;

• очистка строки (сделать строку пустой);

• деструктор.

Производный от СТРОКА класс КОМПЛЕКСНОЕ\_\_ЧИСЛО.

Строки данного класса состоят из двух нолей, разделенных символом i.

Первое поле задает значение реальной части числа, а второе — мнимой. Каждое

из полей может содержать только символы десятичных цифр и символы - и +,

задающие знак числа. Символы - или + могут находиться только в первой позиции числа, причем символ + может отсутствовать, в этом случае число считается

положительным. Если в составе инициализирующей строки будут встречены любые символы, отличные от допустимых, КОМПЛЕКСНОЕ\_ЧИСЛО принимает

нулевое значение. Примеры строк: 33112, -7il00, +5i-21.

Обязательные методы:

• конструктор без параметров;

• конструктор, принимающий в качестве параметра С-строку (заканчивается

нулевым байтом);

• конструктор копирования;

• деструктор.

Переопределить следующие операции:

• присваивание (=);

• операция (==) — проверка на равенство;

• умножение (\*) — умножение чисел.

Разработчик вправе вводить любое (с обоснованием необходимости) число дополнительных полей и методов.

260 Часть 11. Объектно-ориентированное программирование

Написать тестовую программу, которая:

• динамически выделяет массив указателей на базовый класс (4-6);

• в режиме диалога заполняет этот массив указателями на производные классы, при этом экземпляры производных классов создаются динамически с заданием начальных значений;

• для созданных экземпляров производных классов выполняет проверку всех

разработанных методов с выводом исходных данных и результатов на дисплей.

Для конструкторов копирования каждого класса предусмотреть диагностическую печать количества его вызовов в определенное место дисплея (рекомендуется использовать статические члены класса).

Режим диалога обеспечивается с помощью иерархического меню.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <cstring>

#include <cstdlib>

class Строка {

protected:

char\* data;

int length;

public:

Строка() : data(nullptr), length(0) {}

Строка(const char\* str) {

length = std::strlen(str);

data = new char[length + 1];

std::strcpy(data, str);

}

Строка(char ch) {

length = 1;

data = new char[2];

data[0] = ch;

data[1] = '\0';

}

Строка(const Строка& other) {

length = other.length;

data = new char[other.length + 1];

std::strcpy(data, other.data);

std::cout << "Вызван конструктор копирования Строка\n";

}

int getLength() const {

return length;

}

void clear() {

delete[] data;

data = nullptr;

length = 0;

}

virtual ~Строка() {

delete[] data;

}

};

class КомплексноеЧисло : public Строка {

private:

static int copyCount;

bool validate(const char\* str) {

const char\* p = str;

bool i\_found = false;

if (\*p == '+' || \*p == '-') {

p++;

}

while (\*p != 'i' && \*p != '\0') {

if (!std::isdigit(\*p)) {

return false;

}

p++;

}

if (\*p != 'i') {

return false;

}

i\_found = true;

p++;

if (\*p == '+' || \*p == '-') {

p++;

}

while (\*p != '\0') {

if (!std::isdigit(\*p)) {

return false;

}

p++;

}

return i\_found;

}

public:

КомплексноеЧисло() : Строка("0i0") {}

КомплексноеЧисло(const char\* str) {

if (validate(str)) {

length = std::strlen(str);

data = new char[length + 1];

std::strcpy(data, str);

}

else {

data = new char[4];

std::strcpy(data, "0i0");

length = 3;

}

}

КомплексноеЧисло(const КомплексноеЧисло& other) : Строка(other) {

copyCount++;

std::cout << "Вызван конструктор копирования КомплексноеЧисло: " << copyCount << " раз(а)\n";

}

КомплексноеЧисло& operator=(const КомплексноеЧисло& other) {

if (this != &other) {

delete[] data;

length = other.length;

data = new char[other.length + 1];

std::strcpy(data, other.data);

}

return \*this;

}

bool operator==(const КомплексноеЧисло& other) const {

return std::strcmp(data, other.data) == 0;

}

КомплексноеЧисло operator\*(const КомплексноеЧисло& other) const {

char real1\_str[20], imag1\_str[20], real2\_str[20], imag2\_str[20];

std::sscanf(data, "%[^i]i%s", real1\_str, imag1\_str);

std::sscanf(other.data, "%[^i]i%s", real2\_str, imag2\_str);

int real1 = std::atoi(real1\_str);

int imag1 = std::atoi(imag1\_str);

int real2 = std::atoi(real2\_str);

int imag2 = std::atoi(imag2\_str);

int result\_real = real1 \* real2 - imag1 \* imag2;

int result\_imag = real1 \* imag2 + imag1 \* real2;

char result[50];

std::sprintf(result, "%di%d", result\_real, result\_imag);

return КомплексноеЧисло(result);

}

const char\* toString() const {

return data;

}

~КомплексноеЧисло() {}

};

int КомплексноеЧисло::copyCount = 0;

void menu() {

std::cout << "Меню:\n";

std::cout << "1. Создать КомплексноеЧисло из строки\n";

std::cout << "2. Показать длину строки\n";

std::cout << "3. Очистить строку\n";

std::cout << "4. Проверить равенство двух КомплексноеЧисло\n";

std::cout << "5. Умножить два КомплексноеЧисло\n";

std::cout << "6. Выйти\n";

}

int main() {

setlocale(0, "rus");

const int size = 4;

Строка\* array[size];

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

array[i] = nullptr;

}

int choice = 0;

while (choice != 6) {

menu();

std::cout << "Введите ваш выбор: ";

std::cin >> choice;

switch (choice) {

case 1: {

char input[50];

std::cout << "Введите строку для создания КомплексноеЧисло: ";

std::cin >> input;

array[0] = new КомплексноеЧисло(input);

break;

}

case 2: {

if (array[0]) {

std::cout << "Длина строки: " << array[0]->getLength() << "\n";

}

else {

std::cout << "Строка не создана\n";

}

break;

}

case 3: {

if (array[0]) {

array[0]->clear();

std::cout << "Строка очищена\n";

}

else {

std::cout << "Строка не создана\n";

}

break;

}

case 4: {

char input[50];

std::cout << "Введите строку для создания второго КомплексноеЧисло: ";

std::cin >> input;

КомплексноеЧисло\* secondNumber = new КомплексноеЧисло(input);

КомплексноеЧисло\* firstNumber = dynamic\_cast<КомплексноеЧисло\*>(array[0]);

if (firstNumber && \*firstNumber == \*secondNumber) {

std::cout << "Комплексные числа равны\n";

}

else {

std::cout << "Комплексные числа не равны\n";

}

delete secondNumber;

break;

}

case 5: {

char input[50];

std::cout << "Введите строку для создания второго КомплексноеЧисло: ";

std::cin >> input;

КомплексноеЧисло\* secondNumber = new КомплексноеЧисло(input);

КомплексноеЧисло\* firstNumber = dynamic\_cast<КомплексноеЧисло\*>(array[0]);

if (firstNumber) {

КомплексноеЧисло result = \*firstNumber \* \*secondNumber;

std::cout << "Результат умножения: " << result.toString() << "\n";

}

else {

std::cout << "Строка не создана\n";

}

delete secondNumber;

break;

}

case 6: {

std::cout << "Выход\n";

break;

}

default:

std::cout << "Неверный выбор\n";

break;

}

}

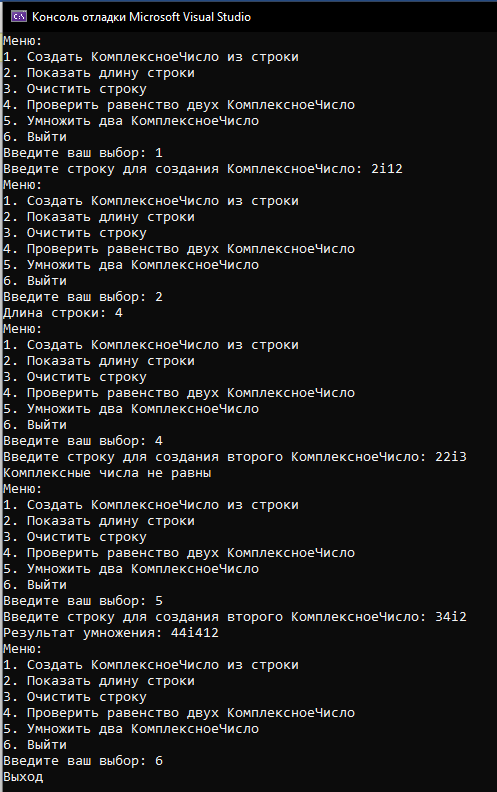
for (int i = 0; i < size; ++i) {

delete array[i];

}

return 0;

}



**https://github.com/strit595/Progrm**