Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ C++ С ИСПОЛЬЗОВАНИЕМ КЛАССОВ

Отчет по лабораторной работе по дисциплине «Объектно-ориентированное программирование»

Студент группы 549
 Баулин С.К.
«__» _____ 2020 г.

Кандидат физико-математических наук, доцент кафедры ЭМИС
_____ Шельмина Е. А.
«__» _____ 2020 г.

оценка

Томск 2020

Лабораторная работа № 5

Программирование на языке C++ с использованием классов

Цель работы

Получение навыков применения конструкторов и деструкторов.

Краткий теоретический материал

Определение класса.

Определение класса идентично определению структуры в C++, за исключением того, что:

- оно обычно содержит одну или несколько спецификаций доступа (public, protected, private);
- вместо ключевого слова struct используется слово class;
- оно обычно включает в себя функции (функции-элементы или методы) наряду с данными-элементами;
- обычно в нем имеются некоторые специальные функции, такие как конструктор (функция с тем же именем, что и сам класс) и деструктор (функция, именем которой является имя класса с префиксом - тильдой (~)).

Управление доступом.

В C++ можно ограничить видимость данных и функций класса при помощи меток public, protected, private. Метка-спецификатор доступа применяется ко всем элементам класса, следующим за ней, пока не встретится другая метка или кончится определение класса.

Метка-спецификатор public (открытый) используется тогда, когда элементы-данные и функции-элементы класса должны быть доступны для функций-элементов и других функций программы, в которой имеется представитель класса.

Метка-спецификатор protected (защищенный) используется в том случае, когда элементы данных и функции-элементы должны быть доступны для функций-элементов данного класса и классов производных от него.

Метка-спецификатор private (закрытый) используется, если элементы-данные и функции-элементы должны быть доступны только для функций-элементов данного класса.

В классе элементы по умолчанию являются закрытыми.

Элементы класса.

Элементы класса делятся на две основные категории:

- данные, называемые элементами-данными;
- код, называемый элементами-функциями или методами.

Данные-элементы.

Данные-элементы классов C++ идентичны элементам структур языка C++ с некоторыми дополнениями:

- данными-элементами могут быть перечислимые типы, битовые поля или представители ранее объявленного класса. Также допускается вложенное объявление перечислимого типа данных и создание псевдонимов с помощью typedef;
- данное-элемент класса может быть указателем или ссылкой на представитель этого класса.

Элементы-функции.

Функция-элемент является функцией, объявленной (описанной) внутри определения класса. Тело функции может также определяться внутри определения класса, в этом случае функция называется встроенной (inline) функцией-элементом. Когда тело функции определяется вне тела класса, перед именем функции ставится префикс из имени класса и операции разрешения видимости (::).

Доступ к данным-элементам

Функции-элементы находятся в области действия класса, в котором они определены. Т.о. они могут обращаться к любому элементу класса, используя просто имя переменной. Обычные функции или функции-элементы другого

класса могут получить доступ к элементам-данным с помощью операции . или ->, применяемых к представителю или указателю на представитель класса.

Конструктор

Конструктор инициализирует представитель класса (объект) и является функцией-элементом с тем же именем, что и класс. Конструктор вызывается компилятором всегда, когда создается представитель класса. Объект считается созданным в тот момент, когда завершил работу конструктор объекта.

Для конструкторов выполняются следующие правила:

- для конструктора не указывается возвращаемый тип;
- конструктор не может возвращать значение;
- конструктор не наследуется;
- для одного класса может существовать один или несколько конструкторов;
- если конструктор не задан явным образом, то автоматически создаётся пустой конструктор.

Конструктор копирования

Конструктором копирования называется специальный конструктор в языке программирования C++, применяемый для создания нового объекта как копии уже существующего. Такой конструктор принимает как минимум один аргумент: ссылку на копируемый объект.

Обычно компилятор автоматически создает конструктор копирования для каждого класса (известные как неявные конструкторы копирования, т.е. конструкторы копирования, заданные неявным образом), но в некоторых случаях программист создает конструктор копирования, называемый в таком случае явным конструктором копирования (или "конструктором копирования, заданным явным образом"). В подобных случаях компилятор не создает неявные конструкторы.

Копирование объектов выполняется за счет использования конструктора копирования и оператора присваивания. Конструктор копирования в качестве

первого параметра (типа const или volatile) принимает ссылку на собственный тип класса. Кроме этого параметра, он может иметь еще дополнительные параметры, при условии, что для таких дополнительных параметров заданы значения по умолчанию.

Существует четыре случая вызова конструктора копирования:

- когда объект возвращает значение;
- когда объект передается (функции) по значению в качестве аргумента;
- когда объект конструируется на основе другого объекта (того же класса);
- когда компилятор генерирует временный объект (как в первом и втором случаях выше; как явное преобразование и т.д.).

Деструктор

Деструктор является дополнением конструктора. Он имеет то же имя, что и класс, но с префиксом - тильдой (~). Он вызывается всякий раз, когда уничтожается представитель класса. Объект считается уничтоженным, когда завершил работу деструктор объекта. Для деструктора существуют следующие правила:

- деструктор не может иметь аргументов;
- деструктор не может возвращать значения;
- деструктор не наследуется (исключением является виртуальный деструктор);
- для одного класса может существовать только один деструктор;
- если деструктор не задан явным образом, то автоматически создаётся пустой деструктор.

Задания

Задание 1. Определить класс-строку. В класс включить два конструктора: для определения класса строки строкой символов и путем

копирования другой строки (объекта класса строки). Предусмотреть функции слияния двух строк и функцию подсчёта предложений в строке.

Задание 2. Создать класс с полями a и b и функциями-членами, в теле которых будут реализованы арифметические операции согласно варианту. Вывести на экран результат арифметических операций.

Вариант: 2.

Тип полей a b: float.

Выражения: /=, +.

Ход работы

Задание 1. Код работы и результат выполненного задания показаны на рисунках 1-2.

```
class Cstr
{
    char *ss;

public:
    Cstr(const char *);
    Cstr(const Cstr &h)
    {
        for (int i = 0; i < strlen(ss) + 1; i++)
        {
            ss[i] = h.ss[i];
        }
        cout << "Вызван второй конструктор (копирование)" << endl;
    };

    const char *get_ss() const { return ss; }

    void strcanc(const Cstr &h)
    {
        cout << strcat(ss, h.ss) << endl;
    };
    void count_sentence()
    {
        int count = 0;
        for (int i = 0; i < strlen(ss) + 1; i++)
        {
            if ((ss[i] == '.' || ss[i] == '?' || ss[i] == '!') && (ss[i + 1] == ' ' || ss[i + 1] == '\0'))
            {
                count++;
            }
        }
        if (count > 0)
        {
            cout << "В строке \"" << ss << "\" " << count << " предложение(ий)" << endl;
        }
        else
            cout << "Предложений не найдено..." << endl;
    };
};
```

Рисунок 1 – Скриншот начала кода задания 1

```
47  Cstr::Cstr(const char *_ss)
48  {
49      ss = new char[strlen(_ss) + 1];
50      strcpy(ss, _ss);
51      cout << "Вызван первый (определяющий) конструктор" << endl;
52  }
53
54  int main()
55  {
56      SetConsoleCP(1251);
57      SetConsoleOutputCP(1251);
58      setlocale(LC_ALL, "Russian");
59      Cstr str1 = "Hello. World!";
60      cout << "Строка 1: " << str1.get_ss() << endl;
61      Cstr str2(str1);
62      cout << "Строка 2: " << str2.get_ss() << endl;
63      cout << "Слияние двух строк: ";
64      str1.strcanc(str2);
65      str2.count_sentence();
66
67      return 0;
68  }
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  КОНСОЛЬ ОТЛАДКИ

PS C:\Users\seron\Desktop\study\3sem\OOP\laba5> .\laba5_1.exe
Вызван первый (определяющий) конструктор
Строка 1: Hello. World!
Вызван второй конструктор (копирование)
Строка 2: Hello. World!
Слияние двух строк: Hello. World!Hello. World!
В строке "Hello. World!" 2 предложение(ий)
```

Рисунок 2 – Скриншот конца кода и результата выполненного задания

Задание 2. Код работы и результат выполненного задания показан на рисунке 3.

Вывод

Получены навыки применения конструкторов и деструкторов.

Рисунок 3 – Скриншот кода и результата выполненного задания 2