Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

=========================================================

Факультет «Специальное машиностроение»

Кафедра «Автономные информационные и управляющие системы»

**В.В. Глазков, М.В. Глазкова**

**ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ C++**

*Методические указания к выполнению лабораторных работ*

*по дисциплине «Информатика»*

Москва,

2019 г.

УДК

ББК

*Рекомендовано Редакционно-издательским советом*

*МГТУ им. Н. Э. Баумана в качестве учебного пособия.*

Рецензент

кандидат технических наук, доцент

В.В. Вельтищев

**Глазков, В.В., Глазкова, М.В.**

Программирование на языке С++: Электронное учебное издание / В.В. Глазков, М.В. Глазкова. - издательство МГТУ имени Н.Э. Баумана, 2019. – 42 с.

**ISBN**

Настоящее издание содержит методические указания к лабораторным работам по информатике, целью которых является приобретение студентами знаний по программированию на базе языка C++. Методическое указание состоит из 5 лабораторных работ с 25 вариантами задач. Для каждой лабораторной работы представлены теоретические сведения по изучаемому материалу и примеры решения аналогичных задач.

УДК

ББК

© В.В. Глазков, М.В. Глазкова, 2019

© МГТУ им. Н.Э. Баумана, 2019

© Оформление. Издательство МГТУ

им. Н.Э. Баумана, 2019

Оглавление

[Введение 4](#_Toc126874597)

[1. Лабораторная работа № 1 «Классы и объекты в С++» 5](#_Toc126874598)

[1.1. Краткие теоретические сведения 5](#_Toc126874599)

[1.2. Практическое задание 10](#_Toc126874600)

[1.3. Контрольные вопросы 12](#_Toc126874601)

**Введение**

Методические указания предназначены для выполнения студентами лабораторных работ по дисциплине «Информатика» самостоятельно и под руководством преподавателя.

Язык программирования С, разработанный в США сотрудниками фирмы Bell Laboratories в начале 70-х годов для разработки операционной системы UNIX, вместе с разработанным вслед за ним языком С++, который дополнительно к возможностям языка С включает объектно- ориентированные средства, является одним из наиболее распространенных языков программирования в мире. Первое описание языка дано его авторами Б. Керниганом и Д. Ритчи. Язык С иногда называют языком программирования «среднего» уровня. С одной стороны, язык поддерживает операции «низкого» уровня (операции над битами), а базовые типы отражают те же объекты, что и язык Ассемблера (байты, машинные слова, символы, строки). С другой стороны — имеет основные управляющие конструкции, присущие языкам «высокого» уровня. Таким образом, язык С может использоваться как для решения системных задач, так и для решения прикладных задач (хотя язык создавался, прежде всего, для системного программирования). Язык программирования С++ является расширением языка С, т.е. включает в себя основные средства языка С и новые возможности, в частности, поддерживает объектно- ориентированный подход в программировании.

Важность изучения языка С++ состоит также в том, что многие языки программирования, созданные позже для других целей, например, для разработки Интернет-приложений, наследовали основные элементы синтаксиса языка С, так называемые, С- подобные языки. К таким языкам можно отнести: Java, C#, PHP. Зная синтаксис языка С, освоить данные языки можно достаточно быстро.

Методические указания к лабораторным работам включают в себя работы по решению прикладных задач на языке С++.

# **1. Лабораторная работа № 1 «Классы и объекты в С++»**

Цель работы: Получить практические навыки реализации классов на С++.

## **1.1. Краткие теоретические сведения**

***Класс***

Класс − фундаментальное понятие С++, он лежит в основе многих свойств С++. Класс предоставляет механизм для создания объектов. В классе отражены важнейшие концепции объектно-ориентированного программирования: инкапсуляция, наследование, полиморфизм.

С точки зрения синтаксиса, класс в С++ − это структурированный тип, образованный на основе уже существующих типов. В этом смысле класс является расширением понятия структуры. В простейшем случае класс можно определить с помощью конструкции:

тип\_класса имя\_класса{список\_членов\_класса};

где тип\_класса – одно из служебных слов class, struct, union;

имя\_класса – идентификатор;

список\_членов\_класса – определения и описания типизированных данных и принадлежащих классу функций.

Методы – это функции класса, определяющие операции над объектом.

Поля – это данные объекта, образующие его структуру. Значения полей определяет состояние объекта.

Для описания объекта класса (экземпляра класса) используется конструкция имя\_класса имя\_объекта;

Например:

date today,my\_birthday;

date \*point = &today; // указатель на объект типа date

date clim[30]; // массив объектов

date &name = my\_birthday; // ссылка на объект

В определяемые объекты входят данные, соответствующие членам-данным класса. Функции-члены класса позволяют обрабатывать данные конкретных объектов класса. Обращаться к данным объекта и вызывать функции для объекта можно двумя способами. Первый с помощью “квалифицированных” имен:

имя\_объекта.имя\_поля

имя\_объекта.имя\_метода()

Например:

complex x1,x2;

x1.re = 1.24;

x1.im = 2.3;

x2.set(5.1,1.7);

x1.print();

Второй способ доступа использует указатель на объект

указатель\_на\_объект–>имя\_компонента

Например:

complex \*point = &x1; // или point = new complex;

point –>re = 1.24;

point –>im = 2.3;

point –>print();

***Доступность компонентов класса***

В рассмотренных ранее примерах классов компоненты классов являются общедоступными. В любом месте программы, где “видно” определение класса, можно получить доступ к компонентам объекта класса. Тем самым не выполняется основной принцип абстракции данных – инкапсуляция (сокрытие) данных внутри объекта. Для изменения видимости компонент в определении класса можно использовать спецификаторы доступа: public, private, protected.

Общедоступные (public) компоненты доступны в любой части программы. Они могут использоваться любой функцией как внутри данного класса, так и вне его. Доступ извне осуществляется через имя объекта:

имя\_объекта.имя\_члена\_класса

ссылка\_на\_объект.имя\_члена\_класса

указатель\_на\_объект->имя\_члена\_класса

Собственные (private) компоненты локализованы в классе и не доступны извне. Они могут использоваться функциями – членами данного класса и функциями – «друзьями» того класса, в котором они описаны.

Защищенные (protected) компоненты доступны внутри класса и в классах-наследниках.

***Конструктор***

Недостатком рассмотренных ранее классов является отсутствие автоматической инициализации создаваемых объектов. Для каждого вновь создаваемого объекта необходимо было вызвать функцию типа set, либо явным образом присваивать значения данным объекта. Однако для инициализации объектов класса в его определение можно явно включить специальную компонентную функцию, называемую конструктором. Формат определения конструктора следующий:

имя\_класса(список\_форм\_параметров){операторы\_тела\_конструктора}

Имя этой компонентной функции по правилам языка С++ должно совпадать с именем класса. Такая функция автоматически вызывается при определении или размещении в памяти с помощью оператора new каждого объекта класса.

Пример.

сomplex(double re1 = 0.0,double im1 = 0.0

{re = re1;

im = im1;}

Конструктор выделяет память для объекта и инициализирует данные − члены класса.

Конструктор имеет ряд особенностей:

- для конструктора не определяется тип возвращаемого значения; даже тип void не допустим;

- указатель на конструктор не может быть определен, и соответственно нельзя получить адрес конструктора;

- конструкторы не наследуются;

- конструкторы не могут быть описаны с ключевыми словами virtual, static, const, mutuable, valatile.

Конструктор всегда существует для любого класса, причем, если он не определен явно, он создается автоматически. По умолчанию создается конструктор без параметров и конструктор копирования. Если конструктор описан явно, то конструктор по умолчанию не создается. По умолчанию конструкторы создаются общедоступными (public).

Параметром конструктора не может быть его собственный класс, но может быть ссылка на него (T&). Без явного указания программиста конструктор всегда автоматически вызывается при определении (создании) объекта. В этом случае вызывается конструктор без параметров. Для явного вызова конструктора используются две формы:

имя\_класса имя\_объекта (фактические\_параметры);

имя\_класса (фактические\_параметры);

Первая форма допускается только при не пустом списке фактических параметров. Она предусматривает вызов конструктора при определении нового объекта данного класса:

complex ss (5.9,0.15);

Вторая форма вызова приводит к созданию объекта без имени:

complex ss = complex (5.9,0.15);

Существуют два способа инициализации данных объекта с помощью конструктора. Ранее мы рассматривали первый способ, а именно, передача значений параметров в тело конструктора. Второй способ предусматривает применение списка инициализаторов данного класса. Этот список помещается между списком параметров и телом конструктора. Каждый инициализатор списка относится к конкретному компоненту и имеет вид:

имя\_данного\_компонента (выражение)

По умолчанию создается также конструктор копирования вида

T::T(const T&),

где Т – имя класса.

Конструктор копирования вызывается всякий раз, когда выполняется копирование объектов, принадлежащих классу. В частности, он вызывается:

- когда объект передается функции по значению;

- при построении временного объекта как возвращаемого значения функции;

- при использовании объекта для инициализации другого объекта.

Если класс не содержит явным образом определенного конструктора копирования, то при возникновении одной из этих трех ситуаций производится побитовое копирование объекта. Побитовое копирование не во всех случаях является адекватным. Именно для таких случаев и необходимо определить собственный конструктор копирования.

Можно создавать массив объектов, однако при этом соответствующий класс должен иметь конструктор по умолчанию (без параметров).

Массив объектов может инициализироваться либо автоматически конструктором по умолчанию, либо явным присваиванием значений каждому элементу массива.

class demo{

int x;

public:

demo(){x=0;}

demo(int i){x=i;}

};

void main(){

class demo a[20]; //вызов конструктора без параметров(по умолчанию)

class demo b[2]={demo(10),demo(100)};//явное присваивание

***Деструктор***

Динамическое выделение памяти для объекта создает необходимость освобождения этой памяти при уничтожении объекта. Например, если объект формируется как локальный внутри блока, то целесообразно, чтобы при выходе из блока, когда уже объект перестает существовать, выделенная для него память была возвращена. Желательно, чтобы освобождение памяти происходило автоматически. Такую возможность обеспечивает специальный компонент класса – деструктор класса. Его формат:

~имя\_класса(){операторы\_тела\_деструктора}

Имя деструктора совпадает с именем его класса, но предваряется символом “~” (тильда).

Деструктор не имеет параметров и возвращаемого значения. Вызов деструктора выполняется не явно (автоматически), как только объект класса уничтожается.

Например, при выходе за область определения или при вызове оператора delete для указателя на объект.

string \*p=new string (“строка”);

delete p;

Если в классе деструктор не определен явно, то компилятор генерирует деструктор по умолчанию, который просто освобождает память, занятую данными объекта. В тех случаях, когда требуется выполнить освобождение и других объектов памяти, необходимо определить деструктор явно.

Так же, как и для конструктора, не может быть определен указатель на деструктор.

## **1.2. Практическое задание**

1) Определить пользовательский класс в соответствии с вариантом задания, место под массивы выделять в динамической памяти.

2) Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3) Определить в классе деструктор.

4) Определить в классе методы для просмотра и установки полей данных.

5) Написать демонстрационную программу, в которой создаются и разрушаются объекты пользовательского класса и каждый вызов конструктора и деструктора сопровождается выдачей соответствующего сообщения (какой объект какой конструктор или деструктор вызвал).

6) Показать в программе использование указателя на объект.

***Варианты заданий***

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Класс – студент; поля: имя – char[30], курс – int, пол – int(bool) |
| 2 | Класс – служащий; поля: имя – char[30], возраст – int, рабочий стаж – int |
| 3 | Класс – кадры; поля: имя – char[30], номер цеха – int, разряд – int |
| 4 | Класс – изделие; поля: имя – char[30], шифр – char[20], количество – int |
| 5 | Класс – библиотека; поля: имя – char[30], автор – char[20], стоимость – float |
| 6 | Класс – экзамен; поля: имя – char[30], дата – int, оценка – int |
| 7 | Класс – адрес; поля: имя – char[30], улица – char[20], номер дома – int |
| 8 | Класс – товар; поля: имя – char[30], количество – int, стоимость – float |
| 9 | Класс – корабль; поля: имя – char[30], водоизмещение – int, тип – char[20] |
| 10 | Класс – цех; поля: имя – char[30], начальник – char[20], количество работающих – int |
| 11 | Класс – персона; поля: имя – char[30], возраст – int, пол – int(bool) |
| 12 | Класс – автомобиль; поля: марка – char[30], мощность – int, стоимость – float |
| 13 | Класс – страна; поля: имя – char[30], форма правления – char[20],, площадь – float |
| 14 | Класс – животное; поля: имя – char[30], класс – char[20], средний вес – int |
| 15 | Класс – преподаватель; поля: имя – char[30], предмет – char[20], рабочий стаж – int |
| 16 | Класс – самолет; поля: имя – char[30], скорость – float, тип – char[20] |
| 17 | Класс – город; поля: имя – char[30], население – int, площадь – float |
| 18 | Класс – рыба; поля: имя – char[30], место обитания – char[20], средний вес – float |
| 19 | Класс – игра; поля: имя – char[30], максимальное количество игроков – int, тип – char[20] |
| 20 | Класс – станция метро; поля: имя – char[30], год основания – int, количество эскалаторов – int |
| 21 | Класс – магазин; поля: имя – char[30], название торгового центра – char[20], этаж – int |
| 22 | Класс – растение; поля: имя – char[30], месяц цветения – char[20], средняя высота стебля – float |
| 23 | Класс – отдел; поля: имя – char[30], руководитель – char[20], количество работающих – int |
| 24 | Класс – тест; поля: имя – char[30], количество баллов – int, оценка – int |
| 25 | Класс – книга; поля: имя – char[30], автор – char[20], стоимость – float |

## **1.3. Контрольные вопросы**

1. Что значит в ООП понятие «класс», и какой формат его объявления в программе?
2. Что такое объект класса, что он содержит?
3. Какие существуют уровни доступа к объектам и методам класса? (Дать характеристику каждому).
4. Что такое «конструктор», формат объявления, его особенности?
5. Формат объявления деструктора, его назначение.