**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**
**Северо-Кавказский филиал**
**ордена Трудового Красного Знамени федерального государственного**
**бюджетного образовательного учреждения высшего образования**
**«Московский технический университет связи и информатики»**

Кафедра Информатики и вычислительной техники

**МУ к ПЗ с 1 до 4**

**по дисциплине**

# Объектно-ориентированное программирование

Ростов-на-Дону

2019

**МУ к ПЗ с 1 до 4**

**по дисциплине**

# Объектно-ориентированное программирование

Для студентов очной и заочной форм обучения
Направление подготовки - **09.03.01 «**Информатика и вычислительная техника»

Составитель: П.В. Лобзенко, доцент кафедры ИВТ
Рассмотрено и одобрено
на заседании кафедры ИВТ
Протокол от «26» августа 2019 г. № 1

# МУ к ПЗ 1

## Составление программ с инкапсуляцией полей и с перегрузкой (полиморфизмом) методов

**1. Цель занятия:**
Выработать умения и навыки по составлению программ с инкапсулированными данными (инкапсуляция) и перегруженными методами (полиморфизм).

**2. Рекомендации:**
Изучить материалы лекций №№4,5.

**Краткая теория**

### Классы. «Перегруженные» функции.

ООП позволяет разбить задачу на ряд самостоятельных связанных между собой подзадач. Каждая подзадача содержит коды и данные, относящиеся к объекту, что упрощает решение задачи в целом и позволяет решать задачи большего объема.

Понятие объекта тесно связано с понятием класса.

Класс – это дальнейшее развитие понятия структуры. Он позволяет создавать новые типы и определять функции, манипулирующие с этими типами. По сути, класс это тип.

Объект - это представитель определенного класса. В общем, виде объект можно рассматривать как переменную, определенную программистом.

ООП использует механизмы инкапсуляции, полиморфизма и наследования.

Инкапсуляция позволяет создавать объекты - данные, процедуры и функции, манипулирующие с этими данными. Данные, доступные для использования внутри объекта - private, данные доступные извне - public.

Полиморфизм позволяет одно имя функции использовать для решения разных задач (общих для класса действий). В зависимости от данных выполняются те или иные действия.

Наследование позволяет одному объекту наследовать свойства другого объекта, т.е. порожденный класс наследует свойства родительского класса и добавляет собственные свойства.

Класс используется для создания объектов. Основная форма имеет вид:

class имя класса

```
  {
закрытые функции и переменные
public:
открытые функции, функции-члены и переменные
}
список объектов;//не является обязательным
```

Закрытые функции и переменные это члены (members) класса, которые доступны только для других членов этого класса. Открытые функции и переменные доступны для любой части программы, в которой находится класс. Функции, объявленные внутри описания класса, называются функциями членами (member functions).

Для определения функций-членов используется форма:

```
тип имя класса :: имя функции-члена (параметры)
{
тело функции
}
```

Два двоеточия :: после имени класса называются операцией расширения области видимости (scope resolution operator).

Класс только определяет тип объектов, а сами объекты не задает (память для их хранения не выделяется). Для создания объектов имя класса используется как спецификатор типа данных. После создания объекта к открытым членам класса можно обращаться, используя операцию «точка».

Пример.

```
#include <iostream.h>
class class1              //объявлен класс  с именем class1
{
  int a;                   //доступна для функций  членов class1
  public:
  int kwadrat(int b); //функция член класса class1
};
int class1::kwadrat(int b) //определение функции «kwadrat()»-члена класса
class1
{
  a=b*b;
  return a;
}
void main()
{
  class1 c;            //создается объект «C» типа class1
```

```
int z,x;
x=3;
z=c.kwadrat(x);

                                                cout<<"\n"<<z<<"\n";//вычисление и
вывод квадрата числа 3
  }
```

«Перегруженные» функции

Две или более функции называются перегруженными, если они имеют одно и тоже имя.

Обычно функции отличаются количеством и типом аргументов. Транслятор автоматически на основании количества или типов аргументов выберет правильный вариант.

Пример.

```
#include <iostream.h>

void k(int a);              //прототип первой функции
void k(int a, float b); //прототип второй функции

void k(int a)             //описание первой функции
{
 cout <<  a <<"\n";
}

void k(int a, float b)    //описание второй функции
{
 cout <<a<<"\n"<< b <<"\n";
}


main()
{
 k(4);                    //вызов первой функции
 k(5, 10.2);            //вызов второй функции
 return 0;
}
```

**3. Порядок выполнения задания**

3.1. Выбрать 3 варианта задания из перечня вариантов, приведенных ниже по следующему правилу: №по журналу- первая задача; №по журналу +3 – вторая задача; и №по журналу +5 – третья задача (если достигнуто окончание списка вариантов зданий, то перейти в его начало).

3.2. Составить программу по заданию.

3.3. Оттранслировать программу на изучаемом языке программирования и получить решение задачи.

3.4. Оформить отчет для каждой из 3 задач, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrin-shert результата выполнения задания и представить его на проверку.

**4. Варианты заданий:**

**Составить методы с одинаковыми названиями для решения следующих задач**

1. Найти среднее арифметическое положительных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

2. Ввести с клавиатуры N чисел. Найти сумму тех из них, которые принадлежат интервалу (2;9).

3. Для N введенных с клавиатуры чисел найти сумму положительных кратных 3.

4. Для арифметической прогрессии 4, 9, 14, 19... найти первые n членов этой прогрессии.

5. Найти сумму отрицательных значений функции Z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке

[-5,12] с шагом 1.

6. Найти среднее арифметическое отрицательных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

7. Найти среднее арифметическое чисел, принадлежащих отрезку [2,184], кратных 2 и введенных с клавиатуры. Всего ввести N различных чисел.

8.Найти сумму значений функции, больших 2 Z=sin(1/x)+5cos(1/(x-3))+x для x, изменяющегося на отрезке [-3,8] с шагом 1.

9.Найти n членов последовательности x1 =x2 =x3 =1; xk =xk-1 +xk-3.

10. Вычислить последовательность N чисел A0 =x, A1 =2, Ak =Ak-1 - Ak-2.

11. Для x1 =0,3 и x2 =-0,3 найти xk =k+sin(xk-2) для k, изменяющегося следующим образом: k = 3,4,...,14.

12. Составить таблицу перевода дюймов в сантиметры для расстояний от 1 до 13 дюймов с шагом 1.

13. Вывести на печать значения функции, меньшие 2, Z=sin(1/x)+5cos(x-3)+x для x, изменяющегося на отрезке [-7,4] с шагом 1.

14.Напечатать таблицу значений функции Y = tg(x/b)+x/(b-2) для x, изменяющегося от 0 до 10 с шагом 1 (b - произвольное число).

15. Вычислить N -ый член последовательности xk = xk-2 -xk-1, x0 = 2,4 x1 = 3,8.

16. Составить таблицу перевода суток (от 1 до 7) в часы, минуты, секунды.

17. Вычислить N-ый член последовательности

xk = xk-1 + (2/3)xk-2 + 1, x1 =-1, x2 =1,38.

18. Напечатать значения функции

z = 1/(x-2)+1/(x-5)+ln(12,8-X) для x, изменяющегося на отрезке [-4,14] с шагом 1.

19. Вывести на печать отрицательные значения функции

z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-6,13] с шагом 1 (учесть область допустимых значений функции).

20. Из N введенных с клавиатуры чисел напечатать кратные 3 и меньшие 58.

21. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (1,11) и являются четными.

22. Из N введенных с клавиатуры чисел напечатать положительные, кратные 3.

23. Вывести на печать значения функции

z = sin(x/(x-2)), находящиеся в интервале (-0,4;0,8) для x, изменяющегося от 8 до -6 с шагом 1.

24. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (2;9).

25. Для геометрической прогрессии 2, 6, 18, 54, 162 ... определить первые n членов этой прогрессии.

26. Ввести с клавиатуры N чисел. Напечатать те из них, которые не принадлежат интервалу (1;5).

27. Найти n членов последовательности x1 =x2 =x3 =1; x k=xk-1 -2xk-3.

28. Вычислить последовательность N чисел A0 =x,

A1 =2, Ak =Ak-1 + Ak-2

29. Составить таблицу перевода килограммов (от 1 до 13) в граммы с шагом

30. Найти сумму значений функции Y=COS(x/A)+x/(A-2) для x, изменяющегося от 2 до 13 с шагом 1 (A - произвольное число).

# МУ к ПЗ 2

## Создание приложений с различными видами наследования

**1. Цель занятия:**
Выработать умения и навыки по составлению программ, содержащий классы, наследующие другие классы.

**2. Рекомендации:**
Изучить материалы лекций №№4-6.

**Краткая теория**

**Наследование. Виртуальные функции. Указатели на объекты**

Наследование - это механизм посредством которого один класс (производный или потомок) может наследовать свойства другого класса (базового или предка).

Базовый класс определяет все качества, которые являются общими для всех производных классов.

Пример:

```
//Базовый класс
class B {
        int i;
        public:
        void set_i(int n);
        int get_i();
};
//Производный класс D
class D : public B
 {
```

```
    int j;
    public:
    void set_j(int n);
    int mul();
};
```

После имени класса D стоит двоеточие, за которым стоит ключевое слово public и имя класса B. Это означает, что класс D будет наследовать все компоненты класса B. Само ключевое слово public информирует компилятор о том, что т.к. класс B будет наследоваться, то все открытые элементы базового класса будут открытыми элементами производного класса. Однако все закрытые элементы базового класса остаются закрытыми.

Пример:

```
// Простой пример наследования.
#include <iostream.h>
// Задание базового класса
class base {
    int i;
    public:
    void set_i(int n);
    int get_i();
};

// Задание производного класса
class derived : public base
 {
```

```
        int j;
        public:
        void set_j(int n);
        int mul();
};

// Установка значения i в базовом классе
void base::set_i(int n)
{
        i = n;
}

// Возврат значения i в базовом классе
int base::get_i()
{
        return i;
}

// Установка значения j в производном классе
void derived::set_j(int n)
{
        j = n;
}

// Возврат значения i из base и, одновременно, j из derived
```

```
int derived::mul()
{
        // производный класс может вызывать функции-члены базового класса
        return j * get_i();
}

main()
{
        derived ob;

        ob.set_i(10); // загрузка i в base
        ob.set_j(4); // загрузка j в derived

        cout << ob.mul(); // вывод числа 40

        return 0;
}
```

При определении mul() вызывается функция get_i()- базового класса B, а не производного D, что указывает на то, что открытые члены базового класса становятся открытыми членами производного. Но в функции mul() вместо прямого доступа к i, необходимо вызывать get_i(), потому что закрытые члены базового класса(i) остаются закрытыми для производных классов.

Виртуальные функции.

Проблема: как будет вызываться функция производного класса, имеющая такое же название, что функция базового класса.

Пример.

```
#include <stdio.h>
class base
{
public:
  int i;
  base(int x);    //конструктор
  void func()
  {
     printf("Базовая функция  %d",i);
     return;
  };
};
//текст конструктора
base::base(int x)
{
  i=x;
  return;
};
class der1: public base
{
public:
      der1(int x) :base(x) {};  //конструктор
```

```
  void func()
  {
    printf("Функция из производного класса %d", i*i);
    return;
  }
};

main()
{
  base * pc;     //указатель на базовый класс
  base ob(2);   //создать экземпляр объекта базового класса
  der1 ob1(2);  //создать экземпляр объекта производного класса
  pc=&ob;        //указатель на объект базового класса
  pc->func();    //вызов функции базового класса
  pc=&ob1;       //указатель на объект производного класса
  pc->func();    //попытка вызова функции производного класса
  return 0;
 }
```

На первый взгляд кажется, что в первом случае будет вызываться функция базового класса, а во втором функция производного. Однако при проверке Вы легко убедитесь, что и в том и в другом случае будет вызвана функция базового класса. При этом, компилятору трудно «понять», какую реально функцию мы имеем в виду и он на стадии компилирования подставляет во всех тех случаях, где встречается имя func() адрес функции базового класса. Такой процесс установки адресов называется "ранним связыванием" или "статическим связыванием". Если же мы хотим, чтобы во втором случае, т.е. когда указатель pc указывал на производный класс вызывалась функция этого класса, ее еще в базовом

классе следует указать как виртуальную. В нашем случае вместо строки void func() следует написать  virtual void func().

Это напоминает «перегрузку» функций, но «перегружаемые» функции отличаются друг от друга типом или аргументами, здесь же функции должны быть идентичны.

В случае использования виртуальных функций адрес вызываемой функции будет определяться в процессе выполнения кода программы. Такой процесс называется "поздним связыванием" или "динамическим связыванием".

Пример:

```
#include <stdio.h>
class base {
public:
  int i;
  base(int x);    //конструктор
  virtual void func()
  {
     printf("Базовая функция  %d\n",i);
     return;
  };
};
//текст конструктора
base::base(int x)
{
  i=x;
  return;
};
class der1: public base {
```

```
public:
     der1(int x) :base(x) {};  //конструктор
  void func()
  {
    printf("Функция из производного класса %d\n", i*i);
    return;
  }
};
class der2: public base {
public:
     der2(int x) :base(x) {};  //конструктор
};

main()
{
   base * pc;     //указатель на базовый класс
   base ob(2);   //создать экземпляр объекта базового класса
   der1 ob1(2);  //создать экземпляр объекта производного класса 1
   der2 ob2(2);  //создать экземпляр объекта производного класса 2
   pc=&ob;         //указатель на объект базового класса
   pc->func();     //вызов функции базового класса
   pc=&ob1;        //указатель на объект производного класса 1
   pc->func();     //попытка вызова функции производного класса
   pc=&ob2;        //указатель на объект производного класса 2
   pc->func();     //попытка вызова функции производного класса
```

```
  return 0;
}
```

Введен еще один производный класс. В нем функция func() не определена. В этом случае будет вызываться функция класса родителя. Т.е. появится строка: «Базовая функция 2». При этом, принцип прост: если Вы хотите, чтобы вызывалась функция родительского класса, не определяйте ее в производном. Еще один вопрос может возникнуть в связи с данным примером: как быть, если мы хотим, чтобы для класса объектов der2 вызывалась функция класса der1. Решение очень просто - сделайте класс der2 наследником не класса base, а класса der1.

В производных классах функция, определенная в базовом классе как виртуальная может определяться, а может и нет. Если Вы хотите, чтобы во всех производных классах обязательно была определена виртуальная функция, то в базовом классе ее надо определить следующим образом:

```
virtual void func() = 0;
```

В этом случае базовый класс называется агрегатным и от него нельзя будет создавать экземпляры объектов, зато во всех производных классах компилятор «потребует» определить данную виртуальную функцию и, тем самым, уменьшить вероятность ошибок.

## 3. Порядок выполнения задания

3.1. Выбрать 3 варианта задания из перечня вариантов, приведенных ниже по следующему правилу: №по журналу- первая задачи, решение которой производится в методе класса№1; №по журналу +3 – вторая задача, решение которой производится в методе класса№2; и №по журналу +5 – третья задача, решение которой производится в методе класса №3 (если достигнуто окончание списка вариантов зданий, то перейти в его начало).
3.2. Исследовать как меняется код программы в зависимости от последовательности наследования:

- 1,2,3;

- 2,1,3;
- 2,3,1;
- 3,2,1;
- 3,1,2;
- 1,3,2.

Отразить выводы в отчете.

3.3. Составить программу по заданию.

3.4. Оттранслировать программу на изучаемом языке программирования и получить решение задачи.

3.5. Оформить отчет для каждой из 3 задач, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrin-shert результата выполнения задания и представить его на проверку.

## 4. Варианты заданий:

**Составить функции пользователя для следующих задач**

1. Составить программу для перевода длины в метрах в длину в сантиметрах, определив функцию, выполняющую это преобразование и передав длину в метрах в качестве параметра.

2. Составить программу для нахождения суммы элементов каждого из трех массивов, введенных с клавиатуры, определив функцию, выполняющую это действие, и передавая массивы в качестве параметра.

3. Даны числа S, T. Получить с использованием функции пользователя F(T,-2S,1.17)+F(2.2,T,S-T) где F(A, B, C) = (2A-B-sin(C))/(5+C)

4. Составить программу перевода двоичной записи натурального числа в десятичную, описав соответствующую функцию с параметром. Перевод осуществлять для чисел, вводимых с клавиатуры. Признак конца ввода - число 0.

5. Даны числа S, T. Получить с использованием функции пользователя с параметрами G(1,sin(S))+2G(T*S,24)-G(5,-S), где G(A,B)=(2A+B*B)/(A*B*2+B*5).

6. Составить программу для расчета значений гипотенузы треугольника, определив функцию, выполняющую этот расчет. Катеты передаются в качестве параметров.

7. Найти периметр десятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, которые передаются функции в качестве параметров из основной программы.

8. Найти периметр шестиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами. Координаты передаются функции в качестве параметров из основной программы.

9. Найти площадь пятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, и процедуру вычисления площади треугольника по трем сторонам. Описать функции с соответствующими формальными параметрами.

10. Составить программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр. Описать соответствующую функцию, получающую из основной программы в качестве параметра натуральное число и возвращающую TRUE, если оно удовлетворяет указанному условию.

11. Используя подпрограмму - функцию, составить программу для нахождения максимального из трех чисел. Числа передаются функции в качестве параметров.

12. Используя подпрограмму - функцию, составить программу для печати знаков трех чисел, введенных с клавиатуры и передаваемых функции в качестве параметра.

13. Используя подпрограмму - функцию, составить программу для возведения чисел в целую положительную степень. Число передаются функции в качестве параметра из основной программы. Расчет вести для чисел, пока не будет введено число, равное 0.

14. Используя подпрограмму - функцию, составить программу для вычисления функции Z=(X1+Y1)/(X1*Y1), где X1 - первый корень уравнения $X^2$-4*X-1=0; Y1 - первый корень уравнения $2*Y^2 + A*Y - A^2 = 0$ (A - произвольное).

15. Задав функцию, вывести на печать средние арифметические двух массивов, введенных с клавиатуры. Массив передается функции в качестве параметра.

16. Задав функцию, рассчитать и вывести на печать максимальные значения в трех парах чисел, вводимых с клавиатуры. Пара чисел передается функции в качестве параметра.

17. Найти периметр восьмиугольника, координаты вершин которого заданы. Определить функцию вычисления расстояния между двумя точками, заданными своими координатами. Координаты передать функции в качестве параметров.

18. Даны четыре пары чисел. Получить с использованием функции пользователя наибольший общий делитель для каждой пары.

19. Даны числа A, B, C. Получить с использованием функции пользователя наименьшее значение. Числа передаются функции из основной программы в качестве параметров.

20. Даны числа x = 1,2,...,N. Получить с использованием функции пользователя значения 3*P(X+3)*P(X) для заданных x, где $P(X) = 10*X^3 - 14*X^2 + 12*X - 2$.

21. Составить программу для расчета значений катета треугольника, определив функцию, выполняющую этот расчет. Гипотенуза и второй катет передаются в качестве параметров.

22. Даны целые числа a,b,c,d. Проверить с использованием функции пользователя их четность. Число для проверки передается в функцию в качестве параметра из основной программы.

23. Для каждого из 10 введенных с клавиатуры чисел напечатать сообщение: является ли оно простым или нет, описав функцию логического типа, возвращающую значение “ИСТИНА”, если число, переданное ей в качестве параметра, является простым.

24. Даны числа S, T. Получить с использованием функции пользователя Y(T,S)=G(12,S)+G(T,S)-G(2S-1,S*T), где G(A,B)=(2*A+B*B)/(A*B*2+B*5).

25. Определите функцию, определяющую, какой целой степенью числа 2 является ее аргумент (если число не является степенью двойки - выдать соответствующее сообщение).

26. Определите функцию, подсчитывающую сумму N первых элементов целочисленного массива A. N и массив A передать в качестве параметров.

27. Вычислить количество простых чисел, не превосходящих заданного N. Описать функцию логического типа, возвращающую значение true, если число простое и false в противном случае.

28. Используя подпрограмму - функцию с параметрами, составить программу для вычисления функции $F(X,Y) = (2X^3-4*X^2+X+1)/(9*Y^3+Y+4) + 3*Y^2+5*Y$.

29. Составить программу для перевода веса в граммах в вес в килограммах, определив функцию, выполняющую это преобразование. Вес в граммах передается функции в качестве параметра.

30. Даны числа S, T. Получить с использованием функции пользователя G(12, S)+G(T, S)-G(2S-1, S*T) где G(A, B) = (2*A+B*B)/(A*B*2+B*5).

# МУ к ПЗ 3

## Составление программ с инкапсуляцией полей и с перегрузкой (полиморфизмом) методов в Java

**1. Цель занятия:**
Выработать умения и навыки по составлению программ с инкапсулированными данными (инкапсуляция) и перегруженными методами (полиморфизм).

**2. Рекомендации:**
Изучить материалы лекций №№4,5.

**Краткая теория**
**Совмещение методов на примере Java-программы**

Язык Java позволяет создавать несколько методов с одинаковыми именами, но с разными списками параметров. Такая техника называется совмещением методов (**method overloading**).

В качестве примера приведена версия класса Pt, в которой совмещение методов использовано для определения альтернативного конструктора, который инициализирует координаты x и y значениями по умолчанию (-1).

```
class Pt
{
int x, y;
      Pt(int x, int y)
      {
      this.x = x;
      this.y = y;
      }

      Pt()
      {
      x = -1;
      y = -1;
      }
 }

class PtCrAlt
 {
      public static void main(String args[])
      {
      Pt ob = new Pt();
      System.out.println("x = " + ob.x + " y = " + ob.y);
```

```
    }
}
```

В этом примере объект класса Pt создается не при вызове первого конструктора, как это было раньше, а с помощью второго конструктора без параметров. Вот результат работы этой программы:

***x = -1 y = -1***

Решение о том, какой конструктор нужно вызвать в том или ином случае, принимается в соответствии с количеством и типом параметров, указанных в операторе new. **Недопустимо объявлять в классе методы с одинаковыми именами и сигнатурами**. В сигнатуре метода не учитываются имена формальных параметров учитываются лишь их типы и количество.

**this в конструкторах**

Очередной вариант класса Pt показывает, как, используя **this** и совмещение методов, можно строить одни конструкторы на основе других.

```
class Pt
{ int x, y;

    Pt(int x, int y)
    {
    this.x = x;
    this.y = y;
    }

    Pt()
    {
    this(-1, -1);
    }
 }
```

В этом примере второй конструктор для завершения инициализации объекта обращается к первому конструктору.

Методы, использующие совмещение имен, не обязательно должны быть конструкторами. В следующем примере в класс Pt добавлены два метода distance. Функция distance возвращает расстояние между двумя точками. Одному из совмещенных методов в качестве параметров передаются координаты точки x и y, другому же эта информация передается в виде параметра-объекта класса Pt.

```
class Pt
{  double x, y;
```

```
        Pt(double x, double y)
        {
        this.x = x;
        this. y = y;
        }

        double distance (double x, double y)
        { double dx, dy;
         dx = this.x - x;
        dy = this.y - y;
        return Math.sqrt(dx*dx + dy*dy);
        }

        double distance(Pt ob)
        {
        return distance(ob.x, ob.y);
        }
}

class PtDist
{
        public static void main(String args[])
        {
        Pt ob1 = new Pt(0, 0);
        Pt ob2 = new Pt(30, 40);
        System.out.println("ob1 = " + ob1.x + ", " + ob1.y);
        System.out.println("ob2 = " + ob2.x + ", " + ob2.y);
        System.out.println("ob1.distance(p2) = " + ob1.distance(p2));
        System.out.println("ob1.distance(60, 80) = " + ob1.distance(60, 80));
        }
}
```

Во второй форме метода **distance** для получения результата вызывается его первая форма. Ниже приведен результат работы этой программы:

```
ob1 = 0, 0
ob2 = 30, 40
ob1.distance(ob2) = 50.0
ob1.distance(60, 80) = 100.0
```

**Инкапсулирование полей на примере Java-программы**

Данные инкапсулируются в класс путем объявления переменных между открывающей и закрывающей фигурными скобками, выделяющи¬ми в определении класса его тело.

Эти переменные объявляются точно так же, как локальные переменные, которые надо объявлять вне методов, в том числе вне метода main.

```
class Pt
{
int x, y;
}
```

Оператор new

Оператор new создает экземпляр указанного класса и возвращает ссылку на вновь созданный объект.

```
Pt ob = new Pt();
```

Вы можете создать несколько ссылок на один и тот же объект.

Оператор точка используется для доступа к переменным и методам объекта.

```
class PtDva
{
        public static void main(String args[])
       {
Pt ob1 = new Pt();
Pt ob2 = new Pt();
ob1.x = 10;
ob1.y = 20;
```

```
ob2.x = 42;
ob2.y = 99;
System.out.println("x = " + ob1.x + " y = " + ob1.y);
System.out.println("x =  " + ob2.x + " y = " + ob2.y);
}
}
```

В этом примере снова использовался класс Pt, было создано два объекта этого класса, и их переменным x и y присвоены различные зна¬чения. Таким образом мы продемонстрировали, что переменные различ¬ных объектов независимы на самом деле.

Ниже приведен результат, полученный при выполнении этой программы.

```
x = 10 y = 20
x = 42 y = 99
```

**3. Порядок выполнения задания**

3.1. Выбрать 3 варианта задания из перечня вариантов, приведенных ниже по следующему правилу: №по журналу- первая задача; №по журналу +3 – вторая задача;  и №по журналу +5 – третья задача (если достигнуто окончание списка вариантов зданий, то перейти в его начало).

3.2. Составить программу по заданию.

3.3. Оттранслировать программу на изучаемом языке программирования и получить решение задачи.

3.4. Оформить отчет для каждой из 3 задач, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrin-shert результата выполнения задания и представить его на проверку.

**4. Варианты заданий:**

**Составить методы с одинаковыми названиями для решения следующих задач**

1. Найти среднее арифметическое положительных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

2. Ввести с клавиатуры N чисел. Найти сумму тех из них, которые принадлежат интервалу (2;9).

3. Для N введенных с клавиатуры чисел найти сумму положительных кратных 3.

4. Для арифметической прогрессии 4, 9, 14, 19... найти первые n членов этой прогрессии.

5. Найти сумму отрицательных значений функции Z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке

[-5,12] с шагом 1.

6. Найти среднее арифметическое отрицательных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

7. Найти среднее арифметическое чисел, принадлежащих отрезку [2,184], кратных 2 и введенных с клавиатуры. Всего ввести N различных чисел.

8.Найти сумму значений функции, больших 2 Z=sin(1/x)+5cos(1/(x-3))+x для x, изменяющегося на отрезке [-3,8] с шагом 1.

9.Найти n членов последовательности x1 =x2 =x3 =1; xk =xk-1 +xk-3.

10. Вычислить последовательность N чисел A0 =x, A1 =2, Ak =Ak-1 - Ak-2.

11. Для x1 =0,3 и x2 =-0,3 найти xk =k+sin(xk-2) для k, изменяющегося следующим образом: k = 3,4,...,14.

12. Составить таблицу перевода дюймов в сантиметры для расстояний от 1 до 13 дюймов с шагом 1.

13. Вывести на печать значения функции, меньшие 2, Z=sin(1/x)+5cos(x-3)+x для x, изменяющегося на отрезке [-7,4] с шагом 1.

14.Напечатать таблицу значений функции Y = tg(x/b)+x/(b-2) для x, изменяющегося от 0 до 10 с шагом 1 (b - произвольное число).

15. Вычислить N -ый член последовательности xk = xk-2 -xk-1, x0 = 2,4 x1 = 3,8.

16. Составить таблицу перевода суток (от 1 до 7) в часы, минуты, секунды.

17. Вычислить N-ый член последовательности

xk = xk-1 + (2/3)xk-2 + 1, x1 =-1, x2 =1,38.

18. Напечатать значения функции

z = 1/(x-2)+1/(x-5)+ln(12,8-X) для x, изменяющегося на отрезке [-4,14] с шагом 1.

19. Вывести на печать отрицательные значения функции

z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-6,13] с шагом 1 (учесть область допустимых значений функции).

20. Из N введенных с клавиатуры чисел напечатать кратные 3 и меньшие 58.

21. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (1,11) и являются четными.

22. Из N введенных с клавиатуры чисел напечатать положительные, кратные 3.

23. Вывести на печать значения функции

z = sin(x/(x-2)), находящиеся в интервале (-0,4;0,8) для x, изменяющегося от 8 до -6 с шагом 1.

24. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (2;9).

25. Для геометрической прогрессии 2, 6, 18, 54, 162 ... определить первые n членов этой прогрессии.

26. Ввести с клавиатуры N чисел. Напечатать те из них, которые не принадлежат интервалу (1;5).

27. Найти n членов последовательности x1 =x2 =x3 =1; x k=xk-1 -2xk-3.

28. Вычислить последовательность N чисел A0 =x,

A1 =2, Ak =Ak-1 + Ak-2

29. Составить таблицу перевода килограммов (от 1 до 13) в граммы с шагом

30. Найти сумму значений функции Y=COS(x/A)+x/(A-2) для x, изменяющегося от 2 до 13 с шагом 1 (A - произвольное число).

# МУ к ПЗ 4

## Составление программ с наследованием классов в Java

**1. Цель занятия:**
Выработать умения и навыки по составлению программ, содержащий классы, наследующие другие классы.

**2. Рекомендации:**
Изучить материалы лекций №№4-6.

**Краткая теория**

**Реализация множественного наследования на примере Java - программ**

Класс может быть производным не только от одного базового класса, а и от многих. Этот случай называется множественным наследованием. Синтаксис описания множественного наследования похож на синтаксис простого наследования:

```
class A {
}; class B {
};
class C : public A, public B {
};
```

Базовые классы класса C перечислены после двоеточия в строке описания класса и разделены запятыми.

**Методы (функции) классов и множественное наследование**

Рассмотрим пример множественного наследования. Пусть нам для некоторых служащих необходимо указать их образование в программе EMPLOY. Теперь

предположим, что в другой программе у нас существует класс student, в котором указывается образование каждого студента. Тогда вместо изменения класса employee мы воспользуемся данными класса student с помощью множественного наследования.

В классе student содержатся сведения о школе или университете, которые закончил студент, и об уровне полученного им образования. Эти данные хранятся в строковом формате. Методы getedu() и putedu() позволяют нам ввести данные о студенте и просмотреть их.

Информация об образовании нужна нам не для всех служащих. Предполо- жим, что нам не нужны записи об образовании рабочих, а необходимы только записи об ученых и менеджерах. Поэтому мы модифицируем классы manager и scientist так, что они будут яЭта маленькая программа показывает только взаимосвязь между классами:

```
class student

{ };

class employee

{ };

class manager : public employee, private student

{ };

class scientist : private employee, private student

{ };

class laborer : public employee

{ };
```

Теперь мы рассмотрим эти классы более детально в листинге EMPMULT.

```
// empmult.cpp

// множественное наследование

#include <iostream>

using namespace std;
```

```
const int LEN = 80; // максимальная длина имени
//////////////////////////////////////////////////////////
class student // сведения об образовании
{
private:
char school[LEN]; // название учебного заведения
char degree[LEN]; // уровень образования
public:
void getedu()
{
cout << " Введите название учебного заведения: ";
cin >> school;
cout << " Введите степень высшего образования\n";вляться производными клас-
сов employee и student.
cout << " (неполное высшее, бакалавр, магистр, кандидат наук): ";
cin >> degree;
}
void putedu() const
{
cout << "\n Учебное заведение: " << school;
cout << "\n Степень: " << degree;
}
};
//////////////////////////////////////////////////////////
class employee // некий сотрудник
{
```

```
private:
char name[LEN]; // имя сотрудника
unsigned long number; // номер сотрудника
public:
void getdata()
{
cout << "\n Введите фамилию: "; cin >> name;
cout << " Введите номер: "; cin >> number;
}
void putdata() const
{
cout << "\n Фамилия: " << name;
cout << "\n Номер: " << number;
}
};
//////////////////////////////////////////////////////////
class manager : private employee, private student // менеджер
{
private:
char title[LEN]; // должность сотрудника
double dues; // сумма взносов в гольф-клуб
public:
void getdata()
{
employee::getdata();
```

```
cout << " Введите должность: "; cin >> title;
cout << " Введите сумму взносов в гольф-клуб: "; cin >> dues;
student::getedu();
}
void putdata() const
{
employee::putdata();
cout << "\n Должность: " << title;
cout << "\n Сумма взносов в гольф-клуб: " << dues;
student::putedu();
}
};
//////////////////////////////////////////////////////////
class scientist : private employee, private student // ученый
{
private:
int pubs; // количество публикаций
public:
void getdata()
{
employee::getdata();
cout << " Введите количество публикаций: "; cin >> pubs;
student::getedu();
}
void putdata() const
```

```
{
employee::putdata();
cout << "\n Количество публикаций: " << pubs;
student::putedu();
}
};
//////////////////////////////////////////////////////////
class laborer : public employee // рабочий
{
};
//////////////////////////////////////////////////////////
int main()
{
manager m1;
scientist s1, s2;
laborer l1;
// введем информацию о нескольких сотрудниках
cout << endl;
cout << "\nВвод информации о первом менеджере";
m1.getdata();
cout << "\nВвод информации о первом ученом";
s1.getdata();
cout << "\nВвод информации о втором ученом";
s2.getdata();
cout << "\nВвод информации о первом рабочем";
```

```
l1.getdata();
// выведем полученную информацию на экран
cout << "\nИнформация о первом менеджере";
m1.putdata();
cout << "\nИнформация о первом ученом";
s1.putdata();
cout << "\nИнформация о втором ученом";
s2.putdata();
cout << "\nИнформация о первом рабочем";
l1.putdata();
cout << endl;
return 0;
}
```

Функции getdata() и putdata() классов manager и scientist включают в себя такие вызовы функций класса student, как

student::getedu(); и

student::putedu();

Эти методы доступны классам manager и scientist, поскольку названные классы наследуются от класса student.

## 3. Порядок выполнения задания

3.1. Выбрать 3 варианта задания из перечня вариантов, приведенных ниже по следующему правилу: №по журналу- первая задачи, решение которой производится в методе класса№1; №по журналу +3 – вторая задача, решение которой производится в методе класса№2;  и №по журналу +5 – третья задача, решение которой производится в методе класса №3 (если достигнуто окончание списка вариантов зданий, то перейти в его начало).
3.2. Исследовать как меняется код программы в зависимости от последовательности наследования:

- 1,2,3;
- 2,1,3;
- 2,3,1;
- 3,2,1;
- 3,1,2;
- 1,3,2.

Отразить выводы в отчете.

3.3. Составить программу по заданию.

3.4. Оттранслировать программу на изучаемом языке программирования и получить решение задачи.

3.5. Оформить отчет для каждой из 3 задач, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrin-shert результата выполнения задания и представить его на проверку.

## 4. Варианты заданий:

**Составить функции пользователя для следующих задач**

31. Составить программу для перевода длины в метрах в длину в сантиметрах, определив функцию, выполняющую это преобразование и передав длину в метрах в качестве параметра.

32. Составить программу для нахождения суммы элементов каждого из трех массивов, введенных с клавиатуры, определив функцию, выполняющую это действие, и передавая массивы в качестве параметра.

33. Даны числа S, T. Получить с использованием функции пользователя F(T,-2S,1.17)+F(2.2,T,S-T) где F(A, B, C) = (2A-B-sin(C))/(5+C)

34. Составить программу перевода двоичной записи натурального числа в десятичную, описав соответствующую функцию с параметром. Перевод осуществлять для чисел, вводимых с клавиатуры. Признак конца ввода - число 0.

35. Даны числа S, T. Получить с использованием функции пользователя с параметрами G(1,sin(S))+2G(T*S,24)-G(5,-S), где G(A,B)=(2A+B*B)/(A*B*2+B*5).

36. Составить программу для расчета значений гипотенузы треугольника, определив функцию, выполняющую этот расчет. Катеты передаются в качестве параметров.

37. Найти периметр десятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, которые передаются функции в качестве параметров из основной программы.

38. Найти периметр шестиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами. Координаты передаются функции в качестве параметров из основной программы.

39. Найти площадь пятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, и процедуру вычисления площади треугольника по трем сторонам. Описать функции с соответствующими формальными параметрами.

40. Составить программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр. Описать соответствующую функцию, получающую из основной программы в качестве параметра натуральное число и возвращающую TRUE, если оно удовлетворяет указанному условию.

41. Используя подпрограмму - функцию, составить программу для нахождения максимального из трех чисел. Числа передаются функции в качестве параметров.

42. Используя подпрограмму - функцию, составить программу для печати знаков трех чисел, введенных с клавиатуры и передаваемых функции в качестве параметра.

43. Используя подпрограмму - функцию, составить программу для возведения чисел в целую положительную степень. Число передаются функции в качестве параметра из основной программы. Расчет вести для чисел, пока не будет введено число, равное 0.

44. Используя подпрограмму - функцию, составить программу для вычисления функции Z=(X1+Y1)/(X1*Y1), где X1 - первый корень уравнения $X^2$-4*X-1=0; Y1 - первый корень уравнения $2*Y^2 + A*Y - A^2 = 0$ (A - произвольное).

45. Задав функцию, вывести на печать средние арифметические двух массивов, введенных с клавиатуры. Массив передается функции в качестве параметра.

46. Задав функцию, рассчитать и вывести на печать максимальные значения в трех парах чисел, вводимых с клавиатуры. Пара чисел передается функции в качестве параметра.

47. Найти периметр восьмиугольника, координаты вершин которого заданы. Определить функцию вычисления расстояния между двумя точками, заданными своими координатами. Координаты передать функции в качестве параметров.

48. Даны четыре пары чисел. Получить с использованием функции пользователя наибольший общий делитель для каждой пары.

49. Даны числа A, B, C. Получить с использованием функции пользователя наименьшее значение. Числа передаются функции из основной программы в качестве параметров.

50. Даны числа x = 1,2,...,N. Получить с использованием функции пользователя значения 3*P(X+3)*P(X) для заданных x, где $P(X) = 10*X^3 - 14*X^2 + 12*X - 2$.

51. Составить программу для расчета значений катета треугольника, определив функцию, выполняющую этот расчет. Гипотенуза и второй катет передаются в качестве параметров.

52. Даны целые числа a,b,c,d. Проверить с использованием функции пользователя их четность. Число для проверки передается в функцию в качестве параметра из основной программы.

53. Для каждого из 10 введенных с клавиатуры чисел напечатать сообщение: является ли оно простым или нет, описав функцию логического типа, возвращающую значение “ИСТИНА”, если число, переданное ей в качестве параметра, является простым.

54. Даны числа S, T. Получить с использованием функции пользователя Y(T,S)=G(12,S)+G(T,S)-G(2S-1,S*T), где G(A,B)=(2*A+B*B)/(A*B*2+B*5).

55. Определите функцию, определяющую, какой целой степенью числа 2 является ее аргумент (если число не является степенью двойки - выдать соответствующее сообщение).

56. Определите функцию, подсчитывающую сумму N первых элементов целочисленного массива A. N и массив A передать в качестве параметров.

57. Вычислить количество простых чисел, не превосходящих заданного N. Описать функцию логического типа, возвращающую значение true, если число простое и false в противном случае.

58. Используя подпрограмму - функцию с параметрами, составить программу для вычисления функции $F(X,Y) = (2X^3-4*X^2+X+1)/(9*Y^3+Y+4) + 3*Y^2+5*Y$.

59. Составить программу для перевода веса в граммах в вес в килограммах, определив функцию, выполняющую это преобразование. Вес в граммах передается функции в качестве параметра.

60. Даны числа S, T. Получить с использованием функции пользователя G(12, S)+G(T, S)-G(2S-1, S*T) где G(A, B) = (2*A+B*B)/(A*B*2+B*5).