gradeup

Prep Smart. Score Better.

Programming in C++ (Part-III)

ABOUT ME : NAVNEET GUPTA

- 8 years teaching experience.
- AIR 92 in GATE 2008
- Qualified UGC-NET 2012, Raj.-SET 2012, CSIR-Recruitment-Exam in 2011
- Achieved 3rd Rank in NPTEL-DBMS Course
- Achieved Silver Medal in CSIR on ERP Project in 2013
- Area of Expertise : DBMS, Programming, Algorithms, Discrete Math, Computer Networks, Operating system

gradeup
Prep Smart. Score Better.

C++ Single Level Inheritance Example: Inheriting Methods

```
class animal //base class
{
public:
    void eat(){
        cout<< "Eating..";
    }
};
```

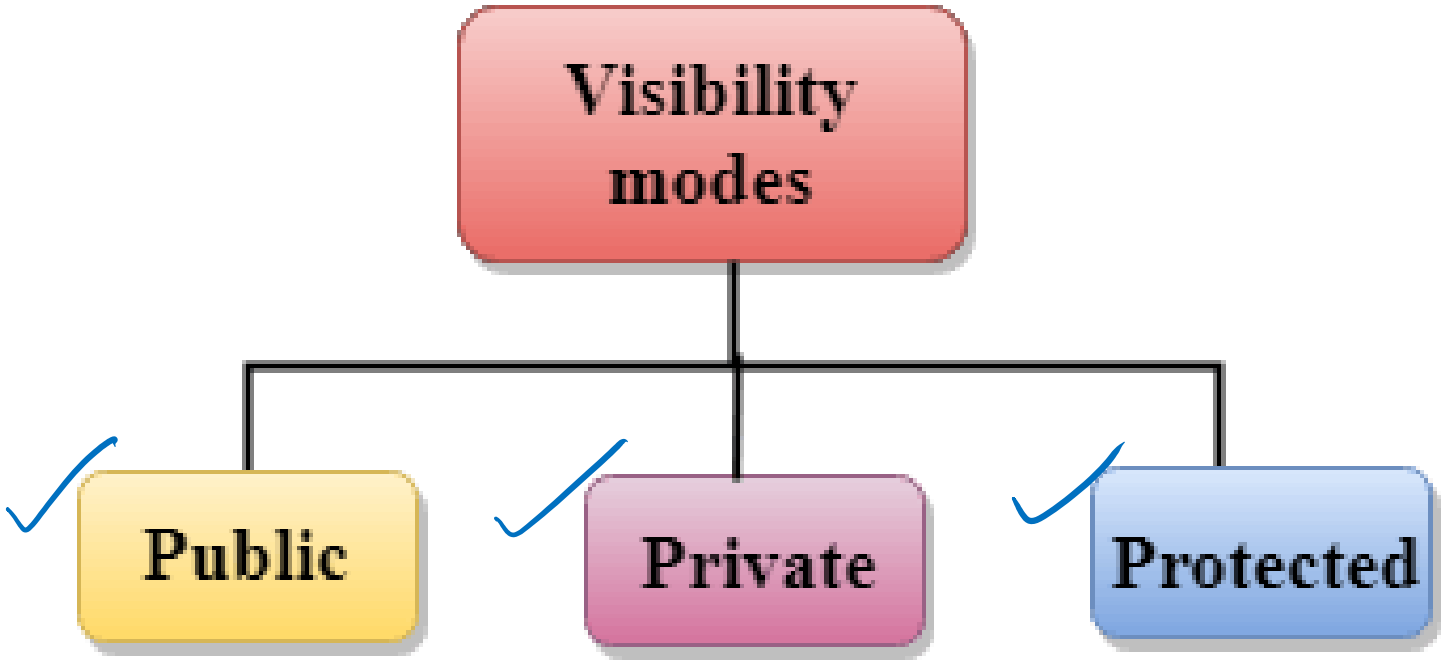
```
class Dog: public animal //derived class
{
public:
    void bark()
    {
        cout<< "barking..";
    }
};
main()
{
    Dog d1;
    d1.eat();
    d1.bark();
}
```

O/P:

Eating...barking..

gradeup
Prep Smart. Score Better.

Visibility modes can be classified into three categories:



gradeup
Prep Smart. Score Better.

Visibility modes can be classified into three categories:

1) Public: When the members are declared as public, it is accessible to all the functions of the program.

2) Private: When the members are declared as private it is accessible within the class only.

3) Protected: When the members are declared as protected, it is accessible within its own class as well as the class immediately derived from it.

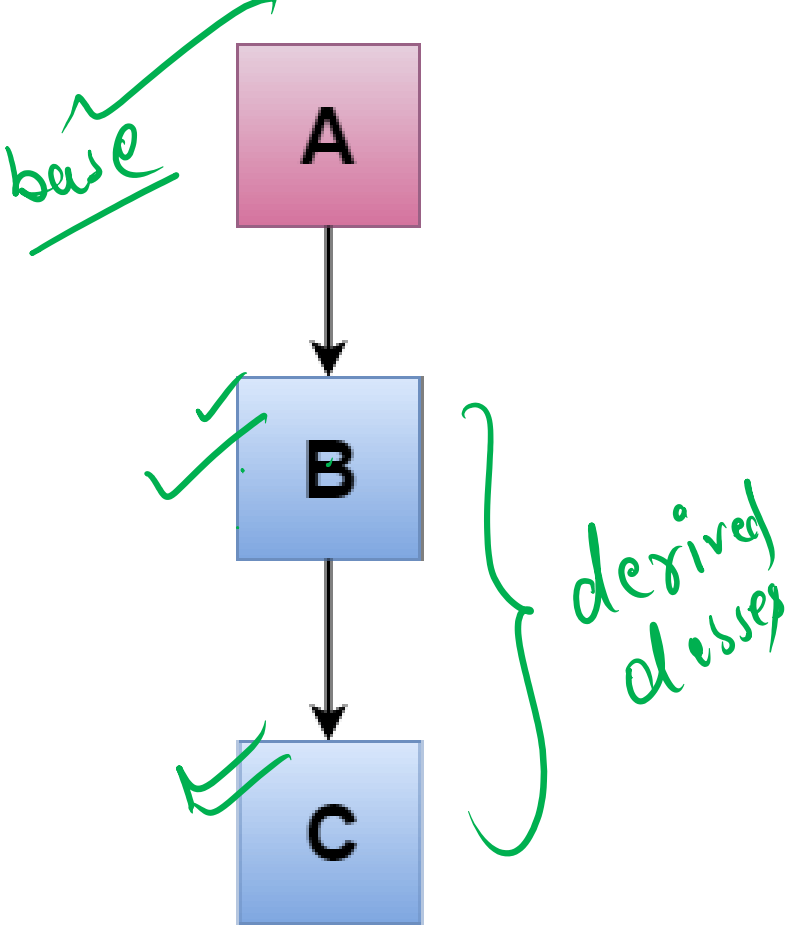
gradeup
Prep Smart. Score Better.

C++ Multilevel Inheritance

(*) Multilevel inheritance is a process of deriving a class from another derived class.

gradeup
Prep Smart. Score Better.

C++ Multilevel Inheritance



gradeup
Prep Smart. Score Better.

C++ Multi Level Inheritance Example:-

```
class Animal
{public:
    void eat (){
    cout << "eating" << endl
};
class Dog : public Animal
    { public:
    void bark(){
        cout << "barking";
    }
};
class cuteDog : public dog
{public:
    void MNG () {
        cout << "Cute";
}};
```

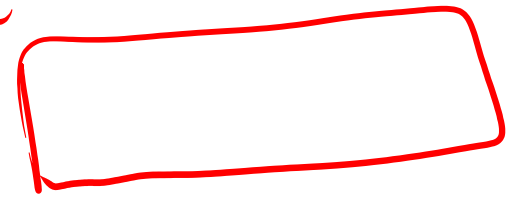
gradeup
Prep Smart. Score Better.

C++ Multi Level Inheritance Example :- (Contd.)

```
main()
{
    CuteDog cd;
    cd.eat();
    cd.bark();
    cd.MNG();
}
```

o/p :-

cd

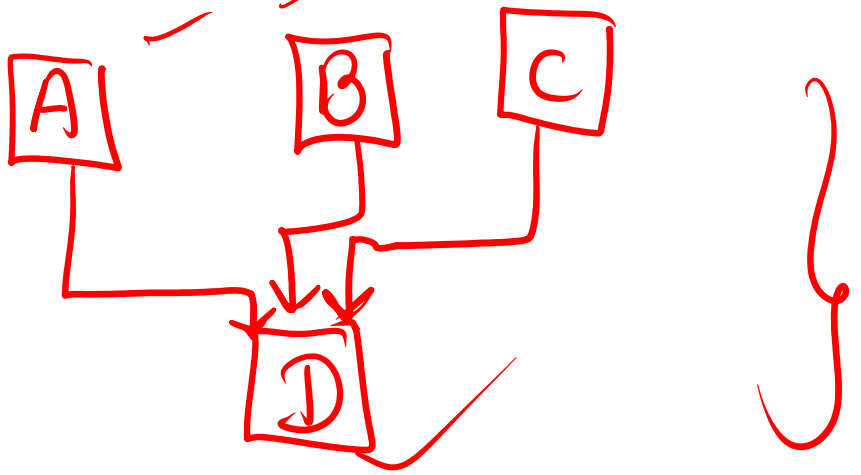


eating... barking... cute

gradeup
Prep Smart. Score Better.

C++ Multiple Inheritance

(*) Multiple inheritance is the process of deriving a new class that inherits the attributes from 2 or more classes.



gradeup
Prep Smart. Score Better.

C++ Multiple Inheritance Example:-

```
class std
{
public:
    string name;
void II (string n)
    {
    name = n;
    }
};
```

```
class grde
{
public:
    string gd;
void PR (string g)
    {
    gd = g;
    }
};
```

gradeup
Prep Smart. Score Better.

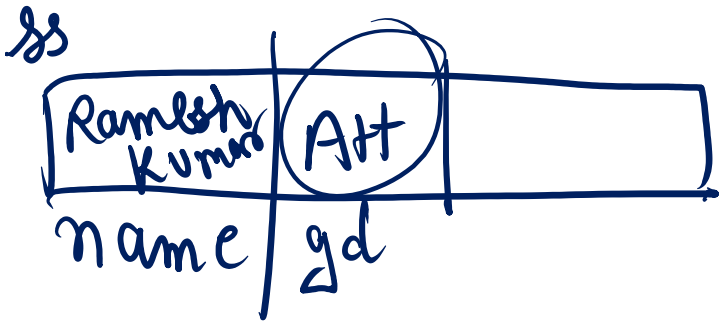
Example:-Contd.:

```
class show : public std, public grade
{
  public:
      void display ()
        {
          cout << "student name is" << name;
          cout << endl << "grade is :" << gd;
        }
};
```

gradeup
Prep Smart. Score Better.

Example:-Contd.:

```
main ()
{
    show ss;
    ss.JJ ("Ramesh Kumar");
    ss.PP ("A++ ");
    ss.display ();
}
```



o/p

student name is Ramesh Kumar

Grade is : A++

gradeup
Prep Smart. Score Better.

Ambiquity Resolution in Inheritance

```
class A
{
public:
    void display()
    {
        cout<< "I am in \"A\" class";
    }
};
```

```
class B
{
public:
    void display()
    {
        cout<< "I m in 'B' class";
    }
};
```

gradeup
Prep Smart. Score Better.

Example Contd:

```
class C : public A, public B
{ public :
    void show()
        { display (); }
}
```

```
main ()
{
    C c1;
    c1. display ();
}
```

o/p

Error: reference to 'display' is ambiguous

```
display ();
```

gradeup
Prep Smart. Score Better.

The above issue can be resolved by using the class resolution operator with the function. In the above example, the derived class code can be rewritten as:

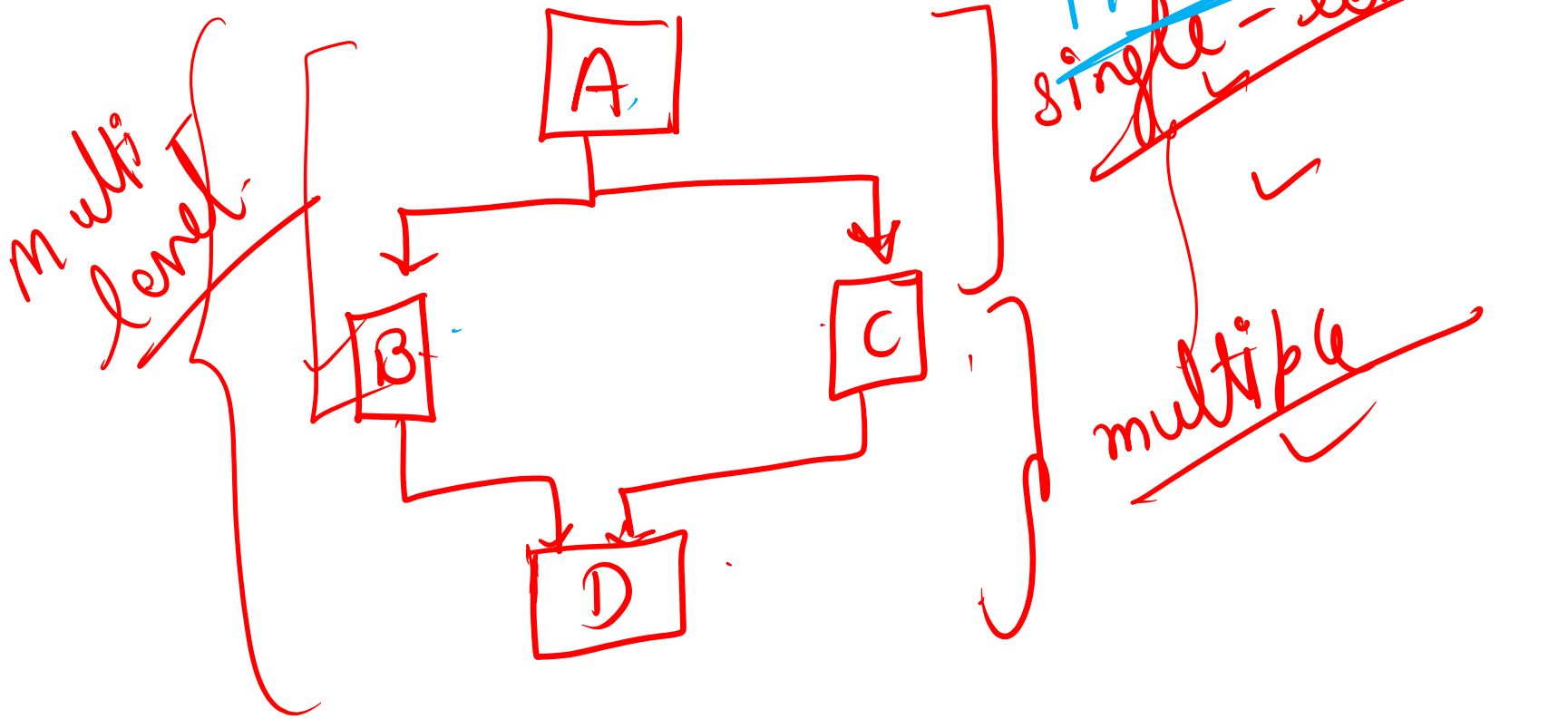
```
class C : public A, public B
{
    void show()
    {
        A :: display ();
        B :: display ();
    }
};
```

```
main()
{
    C c1;
    c1.show();
}
```

gradeup
Prep Smart. Score Better.

C++ Hybrid Inheritance

Hybrid inheritance is a combination of more than one type of inheritance.



gradeup
Prep Smart. Score Better.

Example:

```
class A
{
protected:
    int x;
public: void get()
    {
        cout << "Enter value:";
        cin >> x;
    }
};
```

```
class B: public A
{
protected:
    int y;
public:
    void get_y()
    {
        cout << "Enter value:";
        cin >> y;
    }
};
```

gradeup
Prep Smart. Score Better.

Example:Contd.

```
class C
{
protected:
    int z;
public:
    void get_z()
    {
        cout << "Enter Z: ";
        cin >> z;
    }
};
```

```
cout << "Sum of
    x+y+z = " << x+y+z;
```

```
class D : public B,
          public C
{
protected:
    int p;
public:
    void callme()
    {
        get();
        get_y();
        get_z();
    }
};
```

gradeup
Prep Smart. Score Better.

Example:Contd.

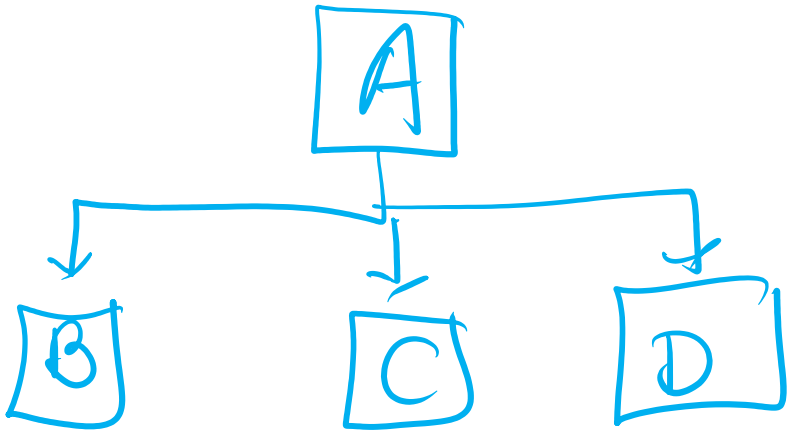
```
main()
{
    D d1;
    d1.callme();
}
```

o/p-

```
Enter value : 15
Enter value: 25
Enter z: 30
Sum of x+y+z=70
```

gradeup
Prep Smart. Score Better.

C++ Hierarchical Inheritance



It is a process of deriving more than one class from a base class

gradeup
Prep Smart. Score Better.

Example:

```
class shape
{
public:
    int a,b;
    void get_data(int n, int m)
    {
        a=n;
        b=m;
    }
};
```

```
class rectangle: public shape
{
public:
    int rect_area()
    {
        int c=a*b;
        return c;
    }
};
```

gradeup
Prep Smart. Score Better.

Example: Contd.:

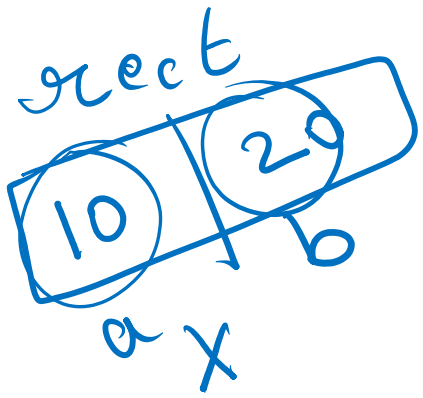
```
class triangle : public shape
{
  public :
  float tri_area ()
  {
    float c = 0.5 * a * b ;
    return c ;
  }
};
```

$\frac{1}{2} * base * height$ $\quad \frac{a * b}{2}$

Area of triangle

gradeup
Prep Smart. Score Better.

Example: Contd.:




```
main()
{
    rectangle rect;
    triangle tri;
    int l, b, base, h;
    cout << "Enter length and breadth of rectangle";
    cin >> l >> b;
    rect.get_data(l,b);
    int area;
    area = rect.rect_area();
    cout << "Area of rectangle =" << area;
    cout << endl << "Enter value of base and height";
    cin >> base >> h;
    tri.get_data(base, h);
    float n = tri.tri_area();
    cout << "Area of triangle =" << n;
}
```

gradeup
Prep Smart. Score Better.

C++ Polymorphism

poly + morphs

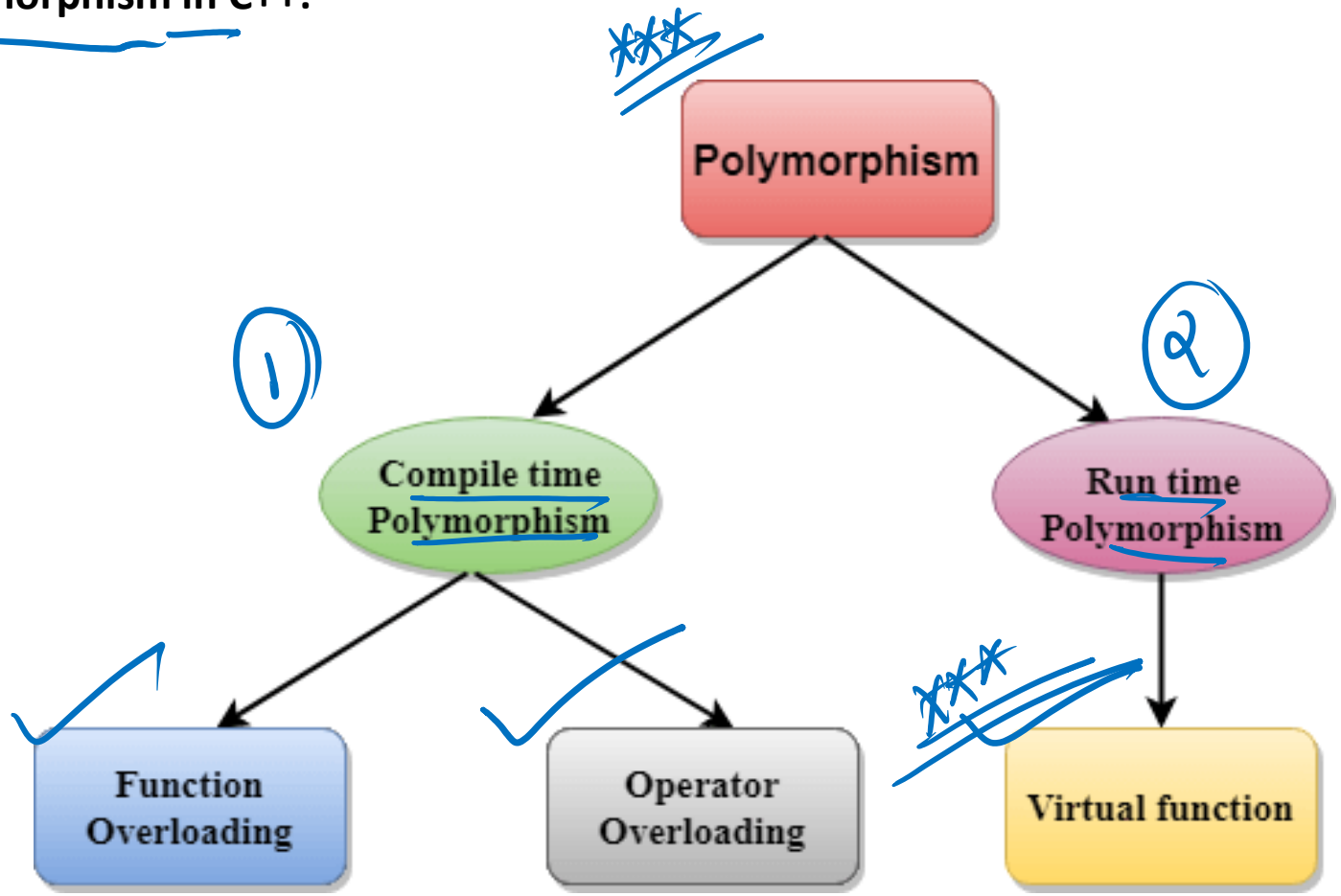
↓ many

↓ to change

one thing has to changed into multiple form.

gradeup
Prep Smart. Score Better.

There are two types of polymorphism in C++:



gradeup
Prep Smart. Score Better.

Compile time polymorphism:

*) The overloaded functions are invoked by matching the type & no. of arguments. This information is available at compile-time and therefore compiler selects the appropriate function at compile time.

gradeup
Prep Smart. Score Better.

Run time polymorphism:

*) Run-time polymorphism is achieved when object's method is invoked at the run time instead of compile-time. It is achieved by "method overriding" which is also known as dynamic binding or late binding.

gradeup
Prep Smart. Score Better.

1 Function Overloading :

*) function overloading is defined as the process of having two or more function with the same name but different in parameters is know as function overloading.

(*) In this, the function is redefined by using either different types of arguments or a diffent no. of arguments.

gradeup
Prep Smart. Score Better.

Function Overloading Example :

```
int   mul(int, int);
float mul(float, int);
int   mul(int a, int b)
{
    return a*b;
}
float mul(float x, int y)
{
    return x*y;
}
```

```
main()
{
    int   r1 = mul(6,7);
    float f1 = mul(2.3, 5);
    cout << r1;
    cout << endl << r2;
}
```

o/p

42
11.5

C++ Operators Overloading

operators that cannot be overloaded :

*) Scope resolution operator (::)

*) sizeof

*) member selector (.)

*) member pointer selector (*)

*) ternary operator (?:)