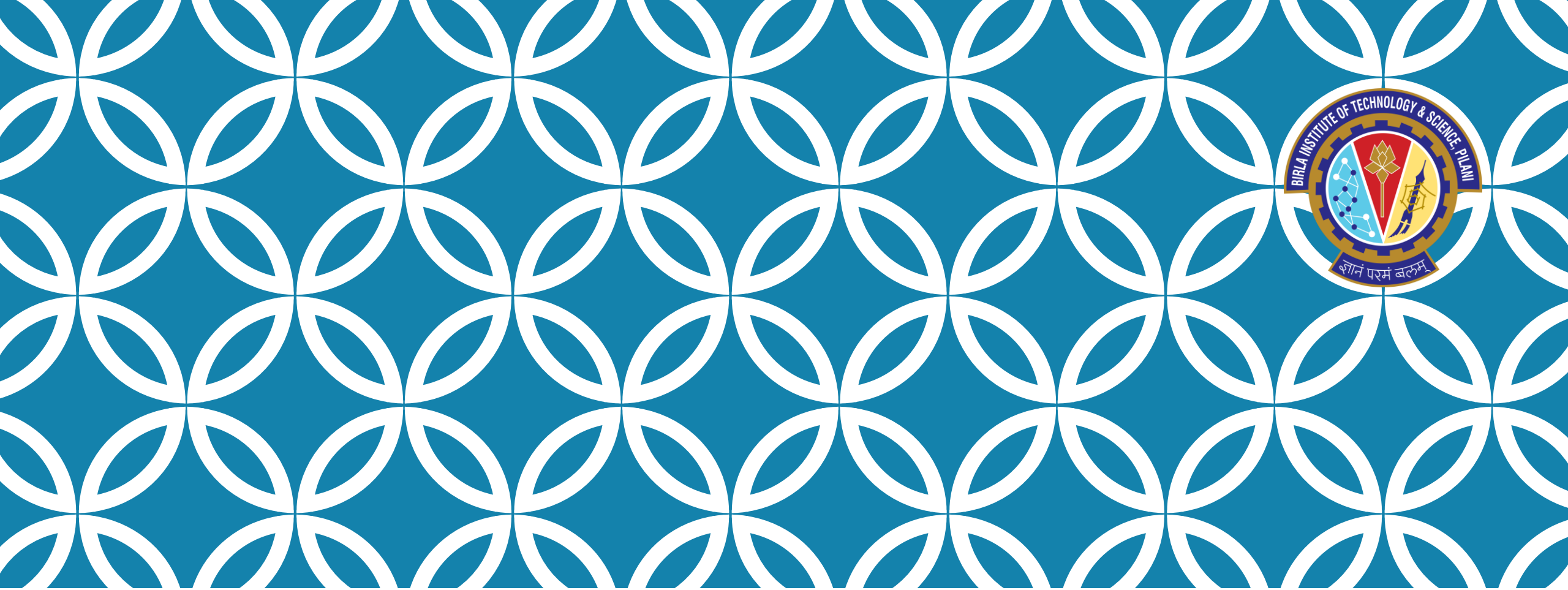


BITS F232: FOUNDATIONS OF DATA STRUCTURES & ALGORITHMS (1ST SEMESTER 2023-24) C++ CONTINUED…

Chittaranjan Hota, PhD
Sr. Professor of Computer Sc.
BITS-Pilani Hyderabad Campus
hota[AT]hyderabad.bits-pilani.ac.in

POLYMORPHISM: FUNCTION OVERLOADING

```cpp
#include <iostream>
using namespace std;

int square ( int x ) {
    return x * x;
}
double square( double y ) {
    return y * y;
}

int main()
{
    int i = square (6);
    cout << i << endl;

    double f = square (5.5);
    cout << f << endl;
    return 0;
}
```

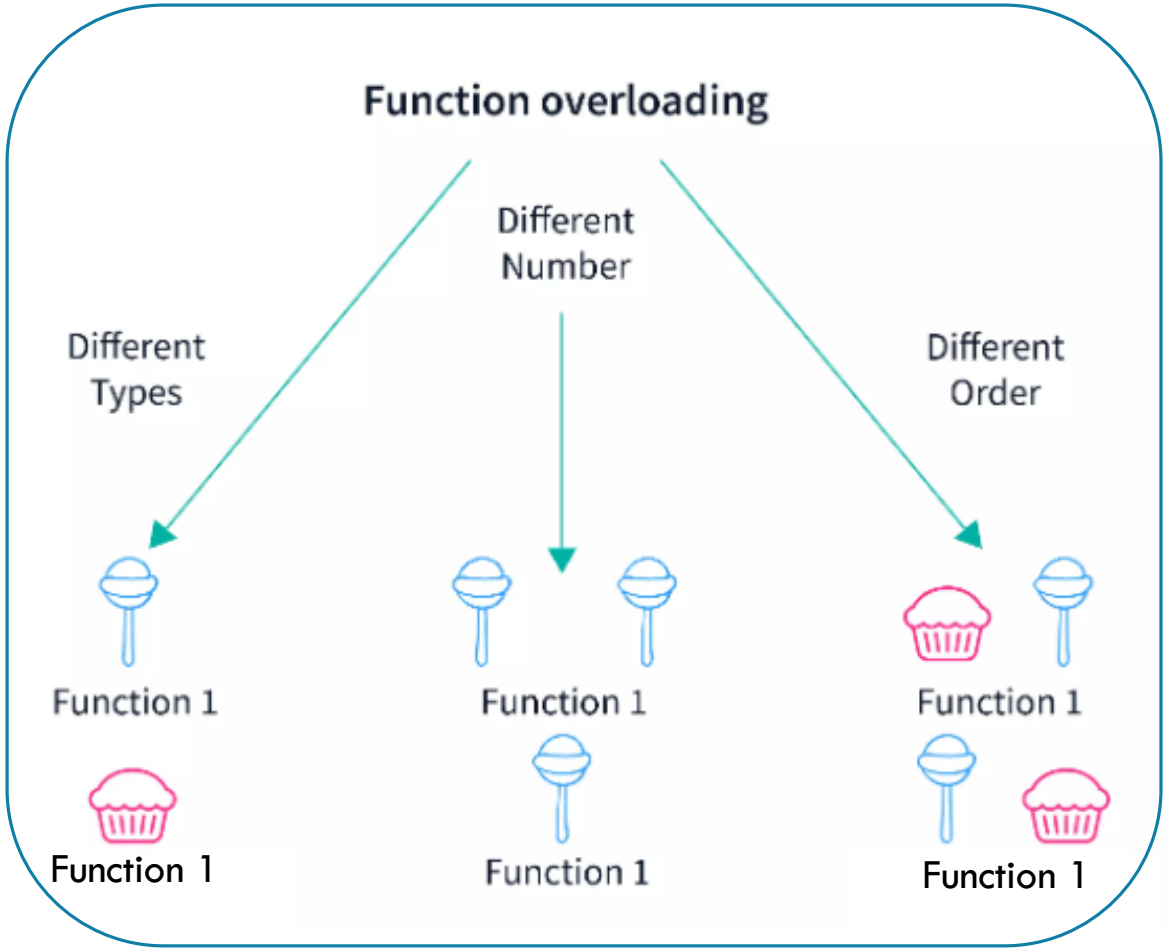
main.cpp

```cpp
#include <iostream>
using namespace std;
class Add
 {
    public:
    int sum(int a,int b)
    {
        return (a+b);
    }
    int sum(int a,int b, int c)
    {
       return (a+b+c);
    }
 };
int main()
{
    Add obj;
    cout<<obj.sum(35, 10)<<endl;
    cout<<obj.sum(100, 50, 50);
    return 0;
}
```

```
45
200

...Program finished with exit code 0
Press ENTER to exit console.
```

CONTINUED...




```cpp
#include<iostream>
using namespace std;

int area(int length) {
    return length * length;
}

int area(int length, int breadth = 1) {

    return length * breadth;
}

int main() {

    int area_ = area(20);
    cout<<"Area square(length = 10) = "<<area_<<endl;

    return 0;
}
```

input

Compilation failed due to following error(s).

```
main.cpp: In function 'int main()':
main.cpp:15:21: error: call of overloaded 'area(int)' is ambiguous
   15 |     int area_ = area(20);
      |                 ~~~~^~~~
main.cpp:4:5: note: candidate: 'int area(int)'
    4 | int area(int length) {
      |     ^~~~
main.cpp:8:5: note: candidate: 'int area(int, int)'
    8 | int area(int length, int breadth = 1) {
      |     ^~~~
```

OPERATOR OVERLOADING

```
#include <iostream>
using namespace std;
int main()  {
    int a = 45;
    int b;
    b = a;
    cout << “value of b: “ <<b;
    return 0;
}
```

```
value of b: 45
```

```
#include <iostream>
using namespace std;
class employee {
public:
    int empno;
    float salary;
};
int main() {
    employee e1= {123, 60000.50}, e2;
    e2 = e1;
    cout << e1.empno << ‘\t’ <<  e2.salary;
    return 0;
}
```

```
123 60000.5
```

CONTINUED ...

```
#include <iostream>
using namespace std;
int main()  {
    int a = 45;
    int b;
    b = a;
    int c = a + b;
    cout << "value of c: " << c;
    return 0;
}
```

```
value of c: 90
```

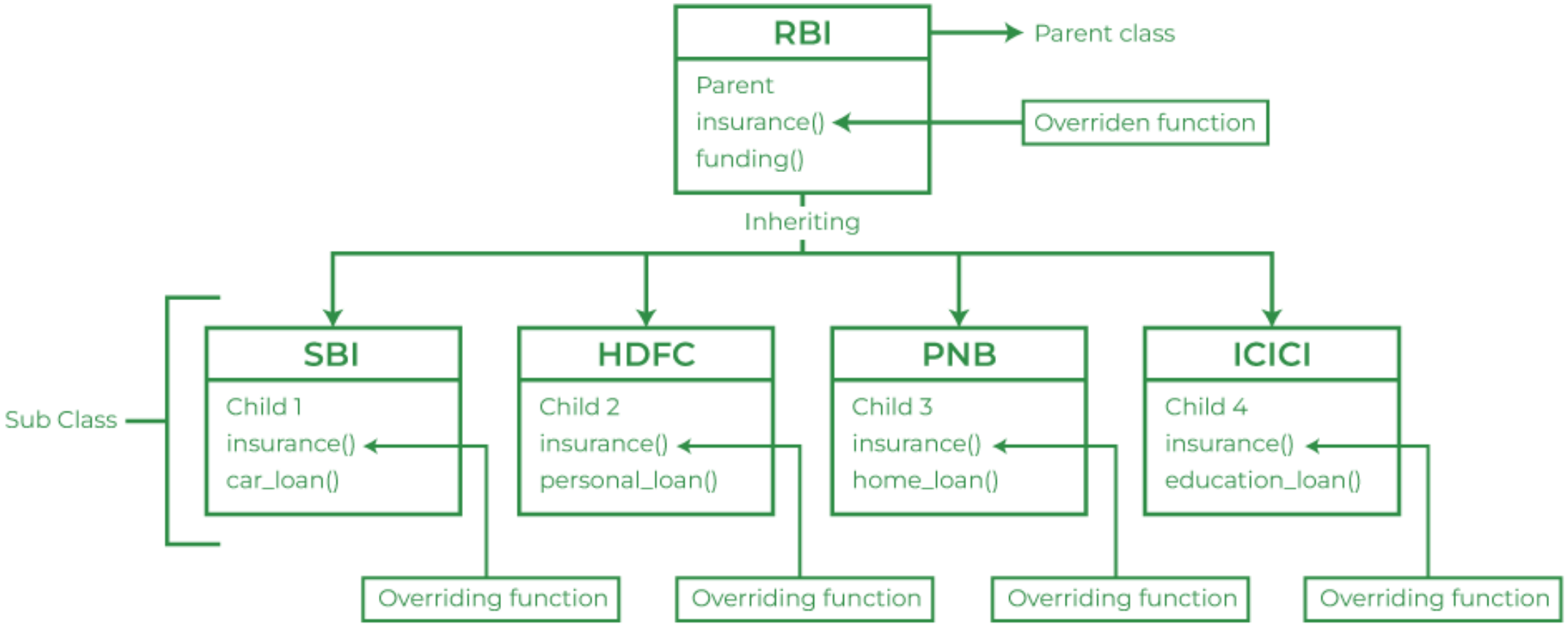
```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  class AddString {
5  public:
6      char str[50];
7      AddString() {}
8      AddString(char str[]) { strcpy(this->str, str); }
9      AddString operator+ (AddString& S2) {
10         AddString S3;
11         strcat(this->str, S2.str);
12         strcpy(S3.str, this->str);
13         return S3;
14     }
15 };
16 int main() {
17     char str1[] = "BITS,Pilani\t";
18     char str2[] = "Hyderabad Campus";
19     AddString a(str1);
20     AddString b(str2);
21     AddString c;
22     c = a + b;
23     cout << c.str;
24     return 0;
25 }
```

Operator Overloading

```
BITS,Pilani Hyderabad Campus
```

Runtime Polymorphism

FUNCTION OVERRIDING IN C++



Img. Source: www.geeksforgeeks.org

main.cpp

```cpp
#include <iostream>
using namespace std;

class Base {
   public:
    void print() {
        cout << "In Base Function" << e
    }
};

class Derived : public Base {
   public:
    void print() {
        cout << " In Derived Function"
    }
};

int main() {
    Derived d;
    d.print();
    return 0;
}
```

```
In Derived Function


...Program finished with exit code 0
Press ENTER to exit console.
```

```cpp
#include <bits/stdc++.h>
using namespace std;
class base
{
public:
    virtual void print ()
    { cout<< "Inside base class's print function" <<endl; }

    void show ()
    { cout<< "Inside base class" <<endl; }
};
class child:public base
{
public:
    void print ()
    { cout<< "Inside child class's print function" <<endl; }

    void show ()
    { cout<< "Inside derived class" <<endl; }
};
int main() {
    base *b;
    child c;
    b = &c;
    //virtual function, bound at runtime (Runtime polymorphism)
    b->print();
    // Non-virtual function, bound at compile time
    b->show();
    return 0;
}
```

```
Inside child class's print function
Inside base class
```

FRIEND CLASS IN C++

- ✓ A friend class is a class whose members have access to the private members of another class.
- ✓ Friendship is NOT transitive.
- ✓ Can a friend not access protected members?

Result

CPU Time: 0.00 sec(s), Memory: 3424 kilobyte(s)

```
49
```

```
1  #include <iostream>
2  using namespace std;
3  class Square;
4  class Rectangle {
5      int width, height;
6    public:
7      int area (){return (width * height);}
8      void convert (Square a);
9  };
10 class Square {
11   friend class Rectangle;
12   private:
13     int side;
14   public:
15     Square (int a):side(a) {}
16 };
17
18 void Rectangle::convert (Square a) {
19   width = a.side;
20   height = a.side;
21 }
22 int main () {
23   Rectangle rect;
24   Square sqr (7);
25   rect.convert(sqr);
26   cout << rect.area();
27   return 0;
28 }
```

FRIEND FUNCTION IN C++

```
#include<iostream>
using namespace std;
class B;
class A
{
    int x;
    public:
        void setdata (int i) {
            x = i;
        }
    friend void min (A, B);
} ;
class B
{
     int y;
     public:
          void setdata (int i) {
              y = i;
          }
     friend void min (A, B);
};
```

```
void min (A a, B b)
{
   if (a.x < b.y)
         cout<< a.x << std::endl;
   else
         cout<< b.y << std::endl;
}
  int main ()
{
   A a;
   B b;
    a. setdata (100);
    b. setdata (250);
    cout << "Min:";
    min (a, b);
    return 0;
}
```

```
Min:100
```

PROTECTED ACCESS SPECIFIER

```cpp
#include <iostream>
using namespace std;
class BitsPilani {
    private:string Museum;
    public: int YearEst;
    BitsPilani (){
        Museum = "BirlaMuseum";
        YearEst = 0;
        FootballGround = "Nil";
    }
    protected: string FootballGround;
};
class BitsHyd : public BitsPilani {
    public: void DisplayGround(){
        FootballGround = "Grass";
        cout <<"Football Ground is made up of:"<<FootballGround<<endl;
    }
    void DisplayEst () {
        cout << "BITS Pilani was established in:" << YearEst << endl;
    }
};
int main () {
    BitsHyd obj;
    obj.YearEst = 1964;
    obj.DisplayGround();
    obj.DisplayEst();
    return 0;
}
```

Access Specifier	Same Class	Outside Class	Derived Class
Public Modifier	YES	YES	YES
Private Modifier	YES	NO	NO
Protected Modifier	YES	NO	YES

CPU Time: 0.00 sec(s), Memory: 3436 kilobyte(s)

```
Football Ground is made up of:Grass
BITS Pilani was established in:1964
```

Can you access Museum within BitsHyd?

Car and Will of Parents: who can access in what mode?

ABSTRACT CLASSES IN C++

```cpp
#include <iostream>
using namespace std;

class Parent  //Base class
{
    public:
    virtual void show() = 0;    // Pure Virtual Function
};

class Child:public Parent //Derived class
{
    public:
    void show()
    {
        cout << "Implementation of Virtual Function in Child cla
    }
};

int main()
{
    Parent *b;
    Child c;
    b = &c;
    b->show();
}
```

```
Implementation of Virtual Function in Child class
```

main.cpp

```cpp
#include <iostream>
using namespace std;
class Shape {
   public:
      virtual int Area() = 0;
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
         height = h;
      }
   protected:
      int width;
      int height;
};
class Rectangle: public Shape {
   public:
      int Area() {
         return (width * height);
      }
};
class Triangle: public Shape {
   public:
      int Area() {
         return (width * height)/2;
      }
};
int main() {
  Rectangle R;
  Triangle T;

  R.setWidth(3);
  R.setHeight(10);

  T.setWidth(10);
  T.setHeight(4);

  cout << "The area of the rectangle is: " << R.Area() << endl;
  cout << "The area of the triangle is: " << T.Area() << endl;
}
```

```
The area of the rectangle is: 30
The area of the triangle is: 20
```