SC/CE/CZ2002 Tutorial 8
Q1: C→C++
Q2: OO in C++

```c
#include<stdio.h>

void bubble(int a[],int n)
{
     int i,j,t;
   for(i=n-2;i>=0;i--)
      {
       for(j=0;j<=i;j++)
          {
             if(a[j]>a[j+1])
                    {
                 t=a[j];
                 a[j]=a[j+1];
                 a[j+1]=t;
              }
          }
       }
  }//end function.

 int main()
  {
     int a[100],n,i;

     printf("\n\n Enter number of Integer elements to be sorted: ");
     scanf("%d",&n);

     for( i=0;i<=n-1;i++)
     {
         printf("\n\n Enter integer value for element no.%d : ",i+1);
         scanf("%d",&a[i]);
     }

     bubble(a,n);

     printf("\n\n Finally sorted array is: ");
     for( i=0;i<=n-1;i++)
            printf("%d  ",a[i]);
  } //end program.
```

```java
import java.util.Scanner;

class BubbleSort {

        public static void bubble(int a[],int n)
        {
                int i,j,t;
                for(i=n-2;i>=0;i--)
                {
                    for(j=0;j<=i;j++)
                    {
                                if(a[j]>a[j+1])
                                {
                                    t=a[j];
                                    a[j]=a[j+1];
                                    a[j+1]=t;
                                }
                    }
                }
            }//end function.

    public static void main(String args[])
    {
            int n,i;
            int a[] = new int[100];

        System.out.print("\n\n Enter number of Integer elements to be sorted: ");
        Scanner in = new Scanner(System.in);
        n = in.nextInt();
        for( i=0;i<=n-1;i++)
        {
         System.out.print("\n\n Enter integer value for element no. " + (i+1) + " : " );
                        a[i] = in.nextInt();
        }
            bubble(a,n);

            System.out.print("\n\n Final sorted array is : ");
                for( i=0;i<=n-1;i++)
                        System.out.print(a[i] + "   " );

                System.out.println() ; // optional
            } //end program.
}
```

3

C vs. C++: Simple Comparison

```c
#include <stdio.h>

int getFact(int n) {
  int c, fact = 1;
  if ( n < 0 )
      printf("Number should be non-negative.");
  else
      for (c = 1; c <= n; c++)
          fact = fact * c;
  return fact;
}

int main()
{
  int n = 1;

  printf("Enter a number to calculate it's factorial\n");
  scanf("%d", &n);

  printf("Factorial of %d = %d\n", n, getFact(n));

  return 0;
}
// save as <anyname>.c
```

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

int getFact(int n) {
  int c, fact = 1;
  if ( n < 0 )
  std:: cout << "Number should be non-negative." << endl ;
  else
      for (c = 1; c <= n; c++)
          fact = fact * c;
  return fact;
}

int main()
{
  int n = 1;

  cout << "Enter a number to calculate it's factorial > ";
  cin >> n;

  cout << "Factorial of " << n << " = " << getFact(n) << endl;

  return 0;
}
// save as <anyname>.cpp
```

4

```c
1    #include<stdio.h>
2
3    void bubble(int a[],int n)
4    {
5         int i,j,t;
6      for(i=n-2;i>=0;i--)
7          {
8          for(j=0;j<=i;j++)
9              {
10                 if(a[j]>a[j+1])
11                     {
12                   t=a[j];
13                   a[j]=a[j+1];
14                   a[j+1]=t;
15                 }
16             }
17         }
18    }//end function.
19
20   int main()
21    {
22         int a[100],n,i;
23
24         printf("\n\n Enter number of Integer elements to be sorted: ");
25         scanf("%d",&n);
26
27         for( i=0;i<=n-1;i++)
28         {
29             printf("\n\n Enter integer value for element no.%d : ",i+1);
30             scanf("%d",&a[i]);
31         }
32
33         bubble(a,n);
34
35         printf("\n\n Finally sorted array is: ");
36         for( i=0;i<=n-1;i++)
37             printf("%d  ",a[i]);
38    } //end program.
```

Q1a. Identify the line number of the code to be changed

```cpp
1    #include <iostream>
2    // using namespace std;
3    void bubble(int a[],int n)
4    {
5         int i,j,t;
6       for(i=n-2;i>=0;i--)
7          {
8          for(j=0;j<=i;j++)
9             {
10               if(a[j]>a[j+1])
11                    {
12                 t=a[j];
13                 a[j]=a[j+1];
14                 a[j+1]=t;
15               }
16           }
17         }
18    }//end function.
19
20   int main()
21    {
22        int a[100],n,i;
23
24      std::cout << "\n\n Enter number of Integer elements to be sorted: ";
25      std::cin >> n;
26
27      for( i=0;i<=n-1;i++)
28      {
29          std::cout << "\n\n Enter integer value for element no." << i+1 << " ";
30          std::cin >> a[i];
31      }
32
33      bubble(a,n);
34
35       std::cout << "\n\n Finally sorted array is: ";
36      for( i=0;i<=n-1;i++)
37           std::cout << a[i] << " " ;
38    } //end program.
```

```cpp
#include <iostream>
using namespace std;
void bubble(int a[],int n)
{
int i,j,t;
for(i=n-2;i>=0;i--)
{
for(j=0;j<=i;j++)
{
if(a[j]>a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
}
}
}
}//end function.
int main()
{
int a[100],n,i;
cout << "\n\n Enter number of Integer elements to be sorted: ";
cin >> n;
for( i=0;i<=n-1;i++)
{
cout << "\n\n Enter integer value for element no." << i+1 << " ";
cin >> a[i];
}
bubble(a,n);
cout << "\n\n Finally sorted array is: ";
for( i=0;i<=n-1;i++)
cout << a[i] << " " ;
} //end program.
```

Main function

- A program shall contain a global function named main, which is the designated start of the program in hosted environment. It shall have one of the following forms:

- `int main() { body }`
- `int main(int argc, char* argv[]) { body }`
- `main() {body}`
- `int main(void) {body}`

Q1c

- Write the code as a C++ <u>class</u>.

C++ class

keyword   user-defined name

```
class ClassName

{  Access specifier:        //can be private,public or protected

   Data members;            // Variables to be used

   Member Functions() { }   //Methods to access data members

};                          // Class name ends with a semicolon
```

https://www.geeksforgeeks.org/c-classes-and-objects/

11

```c
void bubble(int a[],int n)
{
int i,j,t;
for(i=n-2;i>=0;i--)
{
for(j=0;j<=i;j++)
{
if(a[j]>a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
}
}
}
}//end function.
```

Please write out the data members in the BubbleSort class in C++.

ⓘ Start presenting to display the poll results on this slide.

slido

Please write the constructor method header of the BubbleSort class in C++.

ⓘ Start presenting to display the poll results on this slide.

```cpp
void bubble(int a[],int n)
{
int i,j,t;
for(i=n-2;i>=0;i--)
{
for(j=0;j<=i;j++)
{
if(a[j]>a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
}
}
}
}//end function.
```

```cpp
class BubbleSort {
    private :
        int _size ;
        int *_numArray ;

    public :
        BubbleSort(int nums[], int size)
```

```
class BubbleSort {
    private :
        int _size ;
        int *_numArray ;

    public :
        BubbleSort(int nums[], int size) : _size(size), _numArray(nums){}
```

Constructor

Initializer List is used to initialize data members of a class. The list of members to be initialized is indicated with constructor as a comma separated list followed by a colon.

slido

Based on the code of BubbleSort we have wrote so far, please complete the destructor of it in C++.

ⓘ Start presenting to display the poll results on this slide.

19

```cpp
public :
    BubbleSort(int nums[], int size) : _size(size), _numArray(nums){}

    void sort()
    {
        int t ;
        for(int i=_size-2;i>=0;i--)
        {
            for(int j=0;j<=i;j++)
            {
                if(_numArray[j]>_numArray[j+1])
                {
                    t=_numArray[j];
                    _numArray[j]=_numArray[j+1];
                    _numArray[j+1]=t;
                }
            }
        }
    }//end function.

    ~BubbleSort() { delete _numArray ; }  // destructor
```

20

Stack allocation version

```cpp
int main()
{
int a[100],n,i;
cout << "\n\n Enter number of Integer elements to be sorted: ";
cin >> n;
for( i=0;i<=n-1;i++)
{
cout << "\n\n Enter integer value for element no." << i+1 << " ";
cin >> a[i];
}

BubbleSort b(a,n);
b.sort() ;

cout << "\n\n Finally sorted array is: ";
for( i=0;i<=n-1;i++)
cout << a[i] << " " ;

} //end program.
```

21

Heap allocation version

```cpp
int main()
{
int a[100],n,i;
cout << "\n\n Enter number of Integer elements to be sorted: ";
cin >> n;
for( i=0;i<=n-1;i++)
    {
    cout << "\n\n Enter integer value for element no." << i+1 << " ";
    cin >> a[i];
    }

BubbleSort *b = new BubbleSort(a,n);
b->sort();

cout << "\n\n Finally sorted array is: ";
for( i=0;i<=n-1;i++)
cout << a[i] << " " ;

delete b ;
} //end program.
```

Q2

Referring to Tutorial 5 Question 3(v) and its solution, implement the codes for :

a. the Polygon class,

b. its subclasses Rectangle and Triangle class AND

c. a printArea function demonstrating dynamic binding of the calArea implementation.

```java
public class Polygon {
    public enum KindofPolygon { POLY_PLAIN, POLY_RECT,  POLY_TRIANG};
    protected String name;
    protected float width;
    protected float height;
    protected KindofPolygon polytype;

    public Polygon(String theName, float theWidth, float theHeight) {
                name = theName;
                width = theWidth;
                height = theHeight;
                polytype = KindofPolygon.POLY_PLAIN;
    }
    public KindofPolygon getPolytype() {
                return polytype;
    }
    public void setPolytype(KindofPolygon value) {
        polytype = value;
    }
    public String getName()  { return name; }
    public abstract float calArea();
    public void printWidthHeight( ) {
System.out.println("Width = " + width +  " Height = " +   height);
    }
}
```

38

Q2

```java
public class Polygon {
    public enum KindofPolygon { POLY_PLAIN, POLY_RECT,  POLY_TRIANG};
    protected String name;
    protected float width;
    protected float height;
    protected KindofPolygon polytype;

    public Polygon(String theName, float theWidth, float theHeight) {
                name = theName;
                width = theWidth;
                height = theHeight;
                polytype = KindofPolygon.POLY_PLAIN;
    }
    public KindofPolygon getPolytype() {
                return polytype;
    }
    public void setPolytype(KindofPolygon value) {
        polytype = value;
    }
    public String getName()  { return name; }
    public abstract float calArea();
    public void printWidthHeight( ) {
System.out.println("Width = " + width +  " Height = " +   height);
    }
}
```

39

Enumeration in C++

- Enum is a user defined data type where we specify a set of values for a variable and the variable can only take one out of a small set of possible values. We use enum keyword to define a Enumeration.

Simple enum Example

```cpp
enum direction {East, West, North, South}dir
```

```cpp
#include<iostream>
using namespace std;
enum direction {East, West, North, South}dir;
int main()
{
    dir = West;
    cout<<dir;
    return 0;
}
```

https://beginnersbook.com/2017/09/cpp-enumeration/

```cpp
enum KindofPolygon { POLY_PLAIN, POLY_RECT, POLY_TRIANG};


class Polygon {

protected :
        string name;
        float width;
        float height;
        KindofPolygon polytype;

public :
        Polygon(string theName, float theWidth, float theHeight):name(theName) {
                width = theWidth;
                height = theHeight;
                polytype = POLY_PLAIN;
        }
    KindofPolygon getPolytype() {
                return polytype;
        }

        void setPolytype(KindofPolygon value) {
                polytype = value;
        }
```

Q2

```java
public class Polygon {
    public enum KindofPolygon { POLY_PLAIN, POLY_RECT,  POLY_TRIANG};
    protected String name;
    protected float width;
    protected float height;
    protected KindofPolygon polytype;

    public Polygon(String theName, float theWidth, float theHeight) {
                    name = theName;
                    width = theWidth;
                    height = theHeight;
                    polytype = KindofPolygon.POLY_PLAIN;
    }
    public KindofPolygon getPolytype() {
                    return polytype;
    }
    public void setPolytype(KindofPolygon value) {
        polytype = value;
    }
    public String getName()  { return name; }
    public abstract float calArea();
    public void printWidthHeight( ) {
System.out.println("Width = " + width +  " Height = " +   height);
    }
}
```

Please write the calArea menthod in Polygon class in C++.

```cpp
#include  <iostream>
#include <string>

using namespace std ;

enum KindofPolygon { POLY_PLAIN, POLY_RECT, POLY_TRIANG};
string StringKindofPolygon[] = { "POLY_PLAIN", "POLY_RECT", "POLY_TRIANG"};

class Polygon {

protected :
        string name;
        float width;
        float height;
        KindofPolygon polytype;

public :
        Polygon(string theName, float theWidth, float theHeight):name(theName) {
                width = theWidth;
                height = theHeight;
                polytype = POLY_PLAIN;
        }
    KindofPolygon getPolytype() {
                return polytype;
        }

        void setPolytype(KindofPolygon value) {
                polytype = value;
        }
        string getName() { return name; }
        virtual float calArea() = 0 ;
        void printWidthHeight( ) {
                cout << "Width = " << width << " Height = " << height << endl;
        }
};
```

44

Q2b

- Implement the codes for its subclasses Rectangle and Triangle class

```java
public class Rectangle extends Polygon {
    public Rectangle(String theName, float theWidth,
            float theHeight)
    {
        super(theName, theWidth, theHeight) ;
        this.polytype =   KindofPolygon.POLY_RECT;
    }
    public float calArea() { return width * height; }
}
```

Class Inheritance

```
class Point3D : public Point {
    int _z;
public:
    Point3D( const int x, const int y, const int z) : Point(x, y)
        { _z = z; }
.........
};
```

Using Base class constructor

`: Point(x, y) , _z(z) { }`

Specify the desired base constructors <u>after</u> a **single colon** just <u>before</u> the body of constructor.

Class Multiple Inheritance

```
class DrawableString : public Point, public DrawableObject
{

.....
}
```

Please write the class header of Rectangle class in C++.

ⓘ Start presenting to display the poll results on this slide.

```java
public class Rectangle extends Polygon {
    public Rectangle(String theName, float theWidth,
                     float theHeight)
    {
        super(theName, theWidth, theHeight) ;
        this.polytype =   KindofPolygon.POLY_RECT;
    }
    public float calArea() { return width * height; }
}
```



```cpp
class Rectangle : public Polygon {
public :

                {
                    polytype = POLY_RECT;
                }

    float calArea() { return width * height; }
};
```

51

Please write the constructor method header of Rectangle class in C++.

ⓘ Start presenting to display the poll results on this slide.

52

```java
public class Rectangle extends Polygon {
    public Rectangle(String theName, float theWidth,
                     float theHeight)
    {
        super(theName, theWidth, theHeight) ;
        this.polytype =   KindofPolygon.POLY_RECT;
    }
    public float calArea() { return width * height; }
}
```

```cpp
class Rectangle : public Polygon {
public :
      Rectangle(string theName, float theWidth,
                  float theHeight) : Polygon(theName, theWidth, theHeight)
    {
          polytype = POLY_RECT;
    }

    float calArea() { return width * height; }
};
```

```java
public class Triangle extends Polygon {
    public Triangle (String theName, float theWidth,
                float theHeight
    {
        super(theName, theWidth, theHeight) ;
        this.polytype = KindofPolygon.POLY_TRIANG;
    }
    public float calArea() { return 0.5f * width * height; }
}
```

⬇

```cpp
class Triangle : public Polygon  {
public :
    Triangle (string theName, float theWidth,
                    float theHeight) : Polygon(theName, theWidth, theHeight)
    {
            polytype = POLY_TRIANG;
    }

    float calArea() { return 0.5f * width * height; }
};
```

54

Q2c

Implement the codes for a *printArea* function demonstrating dynamic binding of the *calArea* implementation.

```java
public class TestPolygon {   // (ii)
    public static void printArea(Polygon poly) {
        float area = poly.calArea( );
            System.out.println("The area of the " +
    poly.getPolytype()  + " is " + area);
    }


public static void main(String[] args ) {
        Rectangle rect1 = new Rectangle("Rect1", 3.0f, 4.0f);
        printArea(rect1);
        rect1.printWidthHeight();

        Triangle triang1= new Triangle("Triang1", 3.0f, 4.0f);
        printArea(triang1);
        triang1.printWidthHeight();
    }
}
```

Please fill up the header of printArea method to complete the C++ code.

```cpp
class TestPolygon {

public :
        _____{
            float area = poly.calArea( );
            cout <<"The area of the " << StringKindofPolygon[poly.getPolytype()]  <<  " is " << area << endl;
        }
};

int main() {

        Rectangle rect1("Rect1", 3.0f, 4.0f);
        rect1.printWidthHeight();
        TestPolygon::printArea(rect1);


        Triangle triang1("Triang1", 3.0f, 4.0f);
        triang1.printWidthHeight();
        TestPolygon::printArea(triang1);
}
```

Please fill up the header of printArea method to complete the C++ code.

```cpp
class TestPolygon {

public :
        static void printArea(Polygon& poly) {
                float area = poly.calArea( );
                cout <<"The area of the " << StringKindofPolygon[poly.getPolytype()]  <<  " is " << area << endl;
        }
};

int main() {

        Rectangle rect1("Rect1", 3.0f, 4.0f);
        rect1.printWidthHeight();
        TestPolygon::printArea(rect1);


        Triangle triang1("Triang1", 3.0f, 4.0f);
        triang1.printWidthHeight();
        TestPolygon::printArea(triang1);
}
```

Static in class

- The static data members are basically same in Java and C++. The static data members are the property of the class, and it is shared to all of the objects.

- In C++ and Java, the static members can be accessed without creating some objects

```cpp
class TestPolygon {

public :

        _____{

            float area = poly->calArea( );
            cout <<"The area of the "
                    << StringKindofPolygon[poly->getPolytype()]
                    <<  " is " << area << endl;
            }
};
int main() {

                Rectangle rect1("Rect1", 3.0f, 4.0f);
                    rect1.printWidthHeight();
                    TestPolygon::printArea(&rect1);

            Triangle triang1("Triang1", 3.0f, 4.0f);
                    triang1.printWidthHeight();
                    TestPolygon::printArea(&triang1);

        }
```

slido

Please fill up the header of printArea method to complete the C++ code.

ⓘ Start presenting to display the poll results on this slide.

```cpp
class TestPolygon {

public :
    static void printArea(Polygon* poly) {
        float area = poly->calArea( );
    cout <<"The area of the " << StringKindofPolygon[poly->getPolytype()]
      <<  " is " << area << endl;
    }
};
int main() {

        Rectangle rect1("Rect1", 3.0f, 4.0f);
            rect1.printWidthHeight();
            TestPolygon::printArea(&rect1);


        Triangle triang1("Triang1", 3.0f, 4.0f);
            triang1.printWidthHeight();
            TestPolygon::printArea(&triang1);

    }
```

63

Polymorphism

- **Virtual**
  - To force method evaluation to be based on <u>object type</u> rather than <u>reference type</u>. [<ref type> <name> = new <obj type>(..)]
  - Without **virtual =>** non polymorphic (no dynamic binding)
  - **Example : virtual** void area() { cout << "........" << endl ; }
  - **Virtual** function magic only operates on pointers(*) and references(&).
  - If a method is declared **virtual** in a class, it is **automatically virtual** in all **derived** classes.
- **Pure method => abstract** method (pure virtual)
  - By placing **"= 0"** in its declaration
  - **Example : virtual** void area() **= 0** ; // abstract method
  - **The class becomes an abstract** class

https://www.learncpp.com/cpp-tutorial/126-pure-virtual-functions-abstract-base-classes-and-interface-classes/