S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, NIDASOSHI.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Permanently Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Subject: Object Oriented Concepts (18CS45)

Module 2: Objects and Arrays (C++ Part)

Dr. Mahesh G Huddar

Dept. of Computer Science and Engineering

Arrays and Objects

- Like Structures, It is possible to have arrays of objects.
- The syntax for declaring and using an object array is exactly the same as it is for any other type of array.
- A array variable of type class is called as an array of objects.

Arrays and Objects - Example

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

class Student
{
        private:
        int rno;
        char name[20];
        double avgmarks;
        public:
```

```
        void print_data()
        {
                cout<<"Roll No is "<<rno<<endl;
                cout<<"Name is "<<name<<endl;
                cout<<"Average marks "<<avgmarks<<endl;
        }
        void read_data()
        {
                cout<<"Enter the Roll No: ";
                cin>>rno;
                cout<<"Enter the name: ";
                cin>>name;
                cout<<"Enter average marks: ";
                cin>>avgmarks;
        }
};
```

Arrays and Objects - Example

```
int main()
{
        Student std[10];
        cout<<"Enter the student data:"<<endl;
        for (int i=0;i<3;i++)
        {
                std[i].read_data();
        }
        for (int i=0;i<3;i++)
        {
                cout<<i+1<<" Student information is:"<<endl;
                std[i].print_data();
        }
        return 0;
}
```

Arrays and Objects - Example

```
#include<iostream>
using namespace std;
class Student
{       private:
                int rno;
                char name[20];
                double avgmarks;
        public:
                void read_data();
                void print_data();
};
```

```
void Student::read_data()
{
        cout<<"Enter the Roll No: ";
        cin>>rno;
        cout<<"Enter the name: ";
        cin>>name;
        cout<<"Enter the average marks: ";
        cin>>avgmarks;
}
void Student::print_data()
{
        cout<<"Roll No is "<<rno<<endl;
        cout<<"Name is "<<name<<endl;
        cout<<"Average marks "<<avgmarks<<endl;
}
```

Arrays and Objects - Example

```
int main()
{	Student std[10];
	cout<<"Enter the student data:"<<endl;
	for (int i=0;i<3;i++)
	{
		std[i].read_data();
	}
	for (int i=0;i<3;i++)
	{
		cout<<i+1<<" Student information is:"<<endl;
		std[i].print_data();
	}
	return 0;
}
```

Array inside objects

- An array can be used as a member variable of class.
- Using the instance or object of that class we can initialize the array elements.

Array inside objects - Example

```
#include<iostream>
using namespace std;
class Employee
{         private:
                   int empid[10];
                   int empsal[10];
          public:
                   void read_data();
                   void print_data();
};
```

```
void Employee::read_data()
{
          for(int i=0; i<3;i++)
          {
                   cout<<"Enter the employee ID: ";
                   cin>>empid[i];
                   cout<<"Enter the Salary: ";
                   cin>>empsal[i];
          }
}
```

Array inside objects - Example

```
void Employee::print_data()
{
        for (int i=0;i<3;i++)
        {
                cout<<"Employee "<<i+1<<" ID is "<< empid[i]<<endl;
                cout<<"Employee "<<i+1<<" Salary is "<< empsal[i]<<endl;
        }
}

int main()
{
        Employee emp;
        emp.read_data();
        emp.print_data();
        return 0;
}
```

Namespace

- Namespace is a new concept introduced by the ANSI C++ standards committee.
- For using identifiers it can be defined in the namespace scope as below.
- **Syntax:**
  - **using namespace std;**
  - In the above syntax "std" is the namespace where ANSI C++ standard class libraries are defined.
  - Even own namespaces can be defined.

Namespace

Syntax:

```
namespace namespace_name
{
    //Declaration of variables, functions, classes, etc.
}
```

Namespace - Example

```
#include<iostream>
using namespace std;
namespace ns1
{
     int a = 10;
}
namespace ns2
{
     float a = 20.5;
}
int main()
{
     cout<<"Namespace example"<<endl;
     cout<<"The value of a in ns1 is "<<ns1::a<<endl;
     cout<<"The value of a in ns2 is "<<ns2::a<<endl;
     return 0;
}
```

Nested Classes in C++

We will learn:

- What are Nested Classes…?
- How to define nested class as private member of enclosing class?
- How to define nested class as public member of enclosing class?
- Define member functions of Nested class outside the enclosing class?
- How to define Nested class outside the enclosing class?

Nested Classes in C++

Definition of Nested Class:

- Nested class is a class defined inside another class. The class which contains the nested class is called as **Enclosing class.**

Nested Classes in C++

How to define nested class as private member of enclosing class?

- The nested class can be defined as private member of enclosing class.

- The object of enclosing class can be used to access the data members and member functions of the nested class.

Nested Classes in C++

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

class A
{
    private:
        class B
        {
                public:
                void display()
                {
                    cout<<"Nested class"<<endl;
                }
        };
```

```
        B in;

public:
                void show()
                {
                        in.display();
                }
};

int main()
{
        A out;
        out.show();
        return 0;
}
```

Nested Classes in C++

How to define nested class as public member of enclosing class?

- We can define the nested class as a public member of enclosing class.

- In this case, the public member function of nested class can be accessed from the object of enclosing class directly.

Nested Classes in C++

```
#include<iostream>
using namespace std;
class A
{
    public:
        class B
        {
            public:
            void display()
            {
            cout<<"Nested class"<<endl;
            }
        };
};
```

```
int main()
{
        A::B in;
        in.display();
        return 0;
}
```

Nested Classes in C++

How to define member functions of Nested class outside the enclosing class?

- The member function of the nested class can be defined outside the enclosing class.
- Here we need to use the scope resolution operator.

Nested Classes in C++

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

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

```
void A::B::display()
{
        cout<<"Nested class"<<endl;
}

int main()
{
        A::B in;
        in.display();
        return 0;
}
```

Nested Classes in C++

How to define Nested class outside the enclosing class?

- The nested class can be defined outside the enclosing class.

- Here we need to use the scope resolution operator.

Nested Classes in C++

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


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

```
class A::B
{
        public:
                void display()
                {
                        cout<<"Nested class"<<endl;
                }
};

int main()
{
        A::B in;
        in.display();
        return 0;
}
```

Constructors in C++

We will Learn:

- What are Constructors?
- What are the types of constructors with simple programming example?

Constructors in C++

Constructors are special class functions which performs initialization of every object.

The Compiler calls the Constructor whenever an object is created.

Constructor’s initialize values to data members after storage is allocated to the object.

While defining a constructor you must remember that the name of constructor will be same as the name of the class, and constructors never have return type.

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

Type of Constructor in C++

Constructors are of four types :

- Default Constructor
- Parameterized Constructor
- Copy Constructor
- Explicit Constructor

Default Constructor in C++

Default constructor is the constructor which doesn't take any argument.

It has no parameters

Syntax :

```
class_name ()

{

Constructor Definition

}
```

Default Constructor in C++

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

class areaRect
{
        private:
                int h, w;
        public:
        areaRect()
        {
                h = 0;
                w = 0;
        }
        int area()
        {
                cout<<"Enter the height of rectangle: ";
                cin>>h;
                cout<<"Enter the width of the rectangle: ";
                cin>>w;
                return h*w;
        }
};
int main()
{
        int result;
        areaRect a1;
        result = a1.area();
        cout <<endl<<"Area of Rectangle is: "<<result;
}
```

Parameterized Constructor in C++

- These are the constructors with parameter.
- Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

Parameterized Constructor in C++

- **Syntax :**

```
class_name (parameters)
{
Constructor Definition
}
```

Parameterized Constructor in C++

```
#include<iostream>
using namespace std;
class areaRect
{
        private:  int h, w;
        public:
        areaRect()
        {
                h = 0;
                w = 0;
        }
        areaRect(int x, int y)
        {
                h = x;
                w = y;
        }
```

```
        int area()
        {
                return h*w;
        }
};
int main()
{
        int result;
        areaRect a1;
        result = a1.area();
        cout <<endl<<"Area of Rectangle is: "<<result;

        areaRect a2(10, 20);
        result = a2.area();
        cout <<endl<<"Area of Rectangle is: "<<result;
}
```

Copy Constructor in C++

Copy constructor is a special type of constructor in which new object is created as a copy of an existing object.

The copy constructor is called whenever a new variable is created from an object.

Syntax :

```
class_name (class_name &obj)
{
Constructor Definition
}
```

Copy Constructor in C++

```
#include<iostream>
using namespace std;
class areaRect
{
        private:  int h, w;
        public:
        areaRect()
        {
                h = 10;
                w = 10;
        }
        areaRect(areaRect &obj)
        {
                h = obj.h;
                w = obj.w;
        }
        int area()
        {
                return h*w;
        }
};
int main()
{
        int result;
        areaRect a1;
        result = a1.area();
        cout <<endl<<"Area of Rectangle is: "<<result;

        areaRect a2(a1);
        result = a2.area();
        cout <<endl<<"Area of Rectangle is: "<<result;
}
```

Explicit Constructor in C++

In C++, compiler is allowed to make one time type conversion to resolve the parameters to functions.

In C++, a constructor with only one required parameter is considered as an implicit conversion function. It converts the parameter type to the class type.

Prefixing explicit keyword prevents the constructor from using the implicit conversion.

Syntax :

```
explicit class_name (parameters)
{
Constructor Definition
}
```

Explicit Constructor in C++

```
#include<iostream>
using namespace std;
class test
{
        private:
          int val;

        public:

        test(int x)
        {
                val = x;
        }
```

```
        void display()
        {
        cout <<"The value of val is "<<val;
        }
};


int main()
{
        test t1=10;
        t1.display();
}
```

Explicit Constructor in C++

```
#include<iostream>
using namespace std;
class test
{
        private:
          int val;

        public:

        explicit test(int x)
        {
                val = x;
        }
```

```
        void display()
        {
        cout <<"The value of val is "<<val;
        }
};


int main()
{
        test t1=10;
        t1.display();
}
```

Constructor Overloading in C++

- Just like other member functions, constructors can also be overloaded.
- In fact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter.
- You can have any number of Constructors in a class that differ in parameter list.

Destructors in C++

- Destructor is a special class function which destroys the object as soon as the scope of object ends.
- The destructor is called automatically by the compiler when the object goes out of scope.
- The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a **tilde ~ sign as prefix to it.**