Unit III

Constructor

- Special member function.
- Allocate resources.
- Initialize the objects of that class.
- Same name as class.
- Executed automatically.
- Constructor which doesn't take arguments explicitly is called default constructor.
- It has no return type.

Constructor is the first member function to be executed

Syntax of Constructor

```
Class ClassName
{
    ..............//private members
    public:
   ..............//public members
   ClassName( );
};
ClassName::ClassName( )
{
}
```

Constructor prototype

Constructor Definition

Parameterised Constructor

- Constructor with argument.
- Constructor which doesn’t take arguments explicitly is called default constructor.
- Each class can have one and only one default constructor.

Destructor

- Invoked when object is destroyed.
- For local non-static objects, destructor called when function is about to terminate.
- For static or global objects called before program terminates.
- A class can not have more than one destructor.
- Destructor neither take arguments, nor return values.
- Object created most recently is the first one to destroy.

Constructor Overloading

- A class can have multiple constructor.

Destructor Syntax

```
Class ClassName
{
    ..............//private members
    public:
   ..............//public members
   ~ClassName( );
};
ClassName::~ClassName( )
{
}
```

Constructor prototype

Constructor Definition

Difference b/w constructor & destructor

- Arguments cannot be passed to destructor.
- Only one destructor can be declared for a class. Destructors can not be overloaded.
- Destructors can be virtual.

Constructor with default arguments

Nameless Objects

- Unnamed objects can also be created.

Class name

Arg. to constructor

```
classname(arguments);
```

- The scope of a nameless object is limited only to the statement in which it is created.

Dynamic initialization through constructor

- Object's data members can be dynamically initialized during runtime, even after their creation.

Copy Constructor

- Two ways:
  - Classname c1(c2);
  - Classname c1=c2;

Where the copy constructor is invoked

- When we use previously.
- When objects are passed by value.
- When objects are returned from function.

Constant object & Constant member function

- A constant object can call only const member functions.
- Any const member function can’t change the value of data members of a object.

Dynamic Allocation of memory/ Runtime management of memory

- Used when the memory requirement is not known at runtime.
- Memory allocation can be done during execution.
- Two operators are used for this purpose:
  - new
  - delete

- Syntax:

  Datatype * new datatype[size in integer];
- Example:

  int *a;

  a=new int[10];

Note: This is the case of one dimensional integer array.

- Example:

```
int **a;
a=new int*[10];
for (int i=0;i<9;++i)
   a[i]=new int[10];
```

Note: This is the case of two dimensional integer array.

Delete operator

- To free the memory allocated before delete is used.
- **Example**

```
int *a;
a=new int[10];
delete a;
```

- **Example**

```
int **a;
a=new int*[10];
for (int i=0;i<9;++i)
   a[i]=new int[10];
for( i = 0 ; i < 9 ; ++i )
   delete [] a[i] ;
delete [] a ;
```

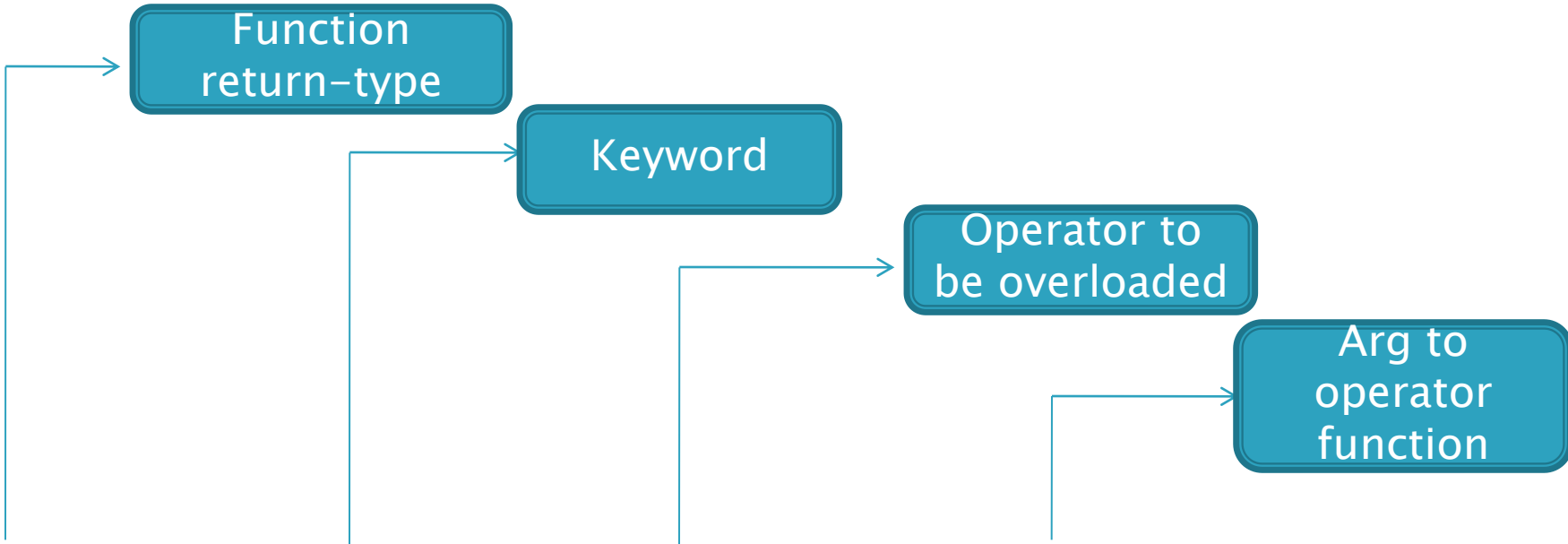
Operator Overloading

- It is another form of polymorphism.
- This concept works in two areas:
  - Extending capability of operators to operate on user defined data.
  - Data conversion.

Operators can't be overloaded

- Class member access operators (. , .*)
- Scope resolution operator (::)
- Size operator (size of)
- Conditional operator(? :)

Syntax




```
Returntype operator operatorsymbol([arg1],[arg2])
{
  //body of overloaded function
}
```

Type Co to nversion

- Conversion from basic data type to class object.
  - A parameterized constructor have to be made in the class.

- Conversion from class object to basic data type.
  - A member function has to be made in the class as:
  - Operator basicDataType()

```
{
}
```

- Conversion from class object to other class object.
  - A member has to be made in the class which is to be converted as:
  - Operator otherClassName()
  - {
  - }