Dr. D. Y. Patil Pratishthan's
Institute for Advanced Computing and Software Development

Sub-c++
Day4

Static Variables

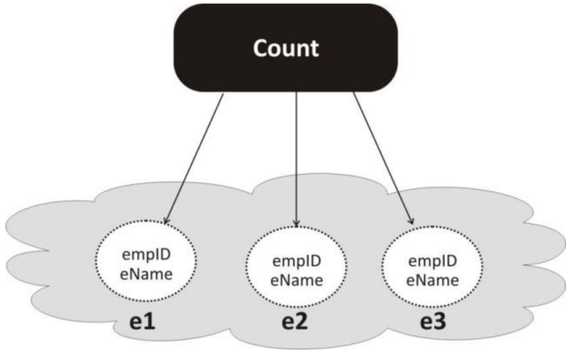
- Some characteristics or behaviors belong to the class rather than a specific instance
  - `interestRate`, `CalculateInterest` method for a `SavingsAccount` class
  - `count` variable in `Employee` to automatically generate employee id
- Such data members are static for all instances
  - Change in static variable value affects all instances
  - Also known as class variable.

Application

.To keep track how many objects created

Static Variables in Memory

```
class Employee
{
    int empId;
    String Nm;
    static int count;
}
```



- Data to be shared by all objects is stored in static data members.
- Only a single copy exists.
- Class scope and lifetime is for entire program.
- How can they be accessed?

  datatype classname::static_varname=value;

Static Member Functions

- Can access static data members only.
- Invoked using class name as:

```
class_name :: functionName();
```

- **`this`** pointer is never passed to a static member function.

```
public class Employee
{
   . . .
   static int count;
   static int showCount()
   {
      return count;
   }
}
```

```
main()
{
   int number =
   Employee::showCount();
   cout<< "Number
   employees are:" <<
   number;
}
```

Destructor

- Destructor is a special member function of the class that is invoked implicitly to release the resources held by the object.

```
~cComplex( ); or ~cString( );
```

~ (tilde)

- Characteristics:
  - Has same name as that of class.
  - Does not have a return type or parameters.
  - Cannot be overloaded. Therefore a class can have only one destructor.
  - Implicitly called whenever an object ceases to exist.

Destructor

- Destructor function de-initializes the objects when they are destroyed.
- It is automatically invoked
  - when object goes out of scope or
  - when the memory allocated to object is de-allocated using the **`delete`** operator.
- It is used to release the resources occupied by the object.
  - If a class contains pointer as a data member then it is mandatory on programmers part to implement a destructor otherwise there is problem of memory leakage.

C++ String

-In C++, string is an object of **std::string** class that represents sequence of characters.
-We can perform many operations on strings such as concatenation, comparison, conversion etc.

-E.g.
```
#include <iostream>
using namespace std;
int main( ) {
    string s1 = "Hello";
        char ch[] = { 'C', '+', '+'};
        string s2 = string(ch);
        cout<<s1<<endl;
        cout<<s2<<endl;  }
```

User Input Strings

It is possible to use the extraction operator >> on cin to display a string entered by a user:

Example

```
string firstName;
cout << "Type your first name: ";
cin >> firstName; // get user input from the keyboard
cout << "Your name is: " << firstName;
```

-- cin considers a space (whitespace, tabs, etc) as a terminating character, which means that it can only display a single word (even if you type many words)

That's why, when working with strings, we often use the getline() function to read a line of text. It takes cin as the first parameter, and the string variable as second:

Example

```
string fullName;
cout << "Type your full name: ";
getline (cin, fullName);
cout << "Your name is: " << fullName;

// Type your full name: IACSD akurdi
// Your name is: IACSD akurdi
```

```
//concatenate two strings
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char key[25], buffer[25];
    cout << "Enter the key string: ";
    cin.getline(key, 25);
    cout << "Enter the buffer string: ";
     cin.getline(buffer, 25);

    strcat(key, buffer);
    cout << "Key = " << key << endl;
    cout << "Buffer = " << buffer<<endl;
    return 0;
}
```

-find out length of string using strlen() function

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char ary[] = "Welcome to C++ Programming";
    cout << "Length of String = " << strlen(ary)<<endl;
    return 0;
}
```

C++ String function

- int compare(const string& str)It is used to compare two string objects.
- int length()It is used to find the length of the string.
- void swap(string& str)It is used to swap the values of two string objects.
- string substr(int pos,int n)It creates a new string object of n characters.
- int size()It returns the length of the string in terms of bytes
- void resize(int n)It is used to resize the length of the string up to n characters.
- string& replace(int pos,int len,string& str)It replaces portion of the string that begins at character position pos and spans len characters.
- string& append(const string& str)It adds new characters at the end of another string object.
- char& at(int pos)It is used to access an individual character at specified position pos.

C++ String compare()

This function compares the value of the string object to the sequence of characters specified by its parameter.

Syntax :

```
int k= str1.compare(str2);
```

k==0 : If k contains value zero, it means both the strings are equal.

k!=0 : If k does contain value zero, it means both the strings are unequal.

```
#include<iostream>
using namespace std;
void main()
{
   string str1="Hello";
   string str2=“IACSD";
    int k= str1.compare(str2);
    if(k==0)
          cout<<"Both the strings are equal";
     else
        cout<<"Both the strings are unequal";

}
```

```
#include<iostream>
using  namespace std;
int main()
{
string s1 = "Welcome to C++";
int len = s1.length();
cout<< "length of the string is : " << len;
return 0;
}
```

C++ String length()-

This function is used to find the length of the string in terms of bytes. This is the actual number of bytes that conform the contents of the string , which is not necessarily equal to the storage capacity.

Syntax-

int len = s1.length();

Parameters-

This function contains single parameter.

Return Value-

This function returns the integer value in terms of bytes.

C++ Math Functions

C++ offers some basic math functions and the required header file to use these functions is <math.h>

cos(x)It computes the cosine of x.
sin(x)It computes the sine of x.
tan(x)It computes the tangent of x.

Exponential functions

exp(x)It computes the exponential e raised to the power x
.frexp(value_type x,int* exp)It breaks a number into significand and 2 raised to the power exponent.
ldexp(float x, int e)It computes the product of x and 2 raised to the power e.
log(x)It computes the natural logarithm of x.
log10(x)It computes the common logarithm of x.
modf()It breaks a number into an integer and fractional part.

Maximum,Minimum and Difference functions-

fdim(x,y)It calculates the positive difference between x and y.

fmax(x,y)It returns the larger number among two numbers x and y.

fmin()It returns the smaller number among two numbers x and y .

Power functions-

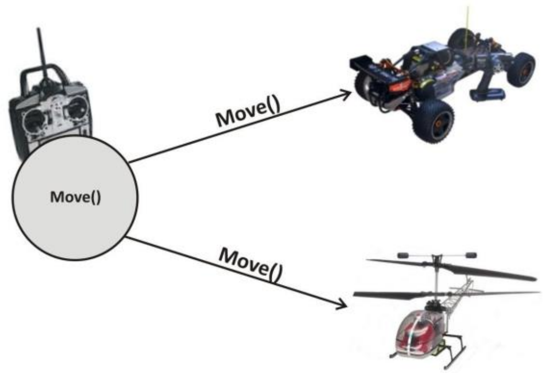
pow(x,y)It computes x raised to the power y.

sqrt(x)It computes the square root of x.

cbrt(x)It computes the cube root of x.

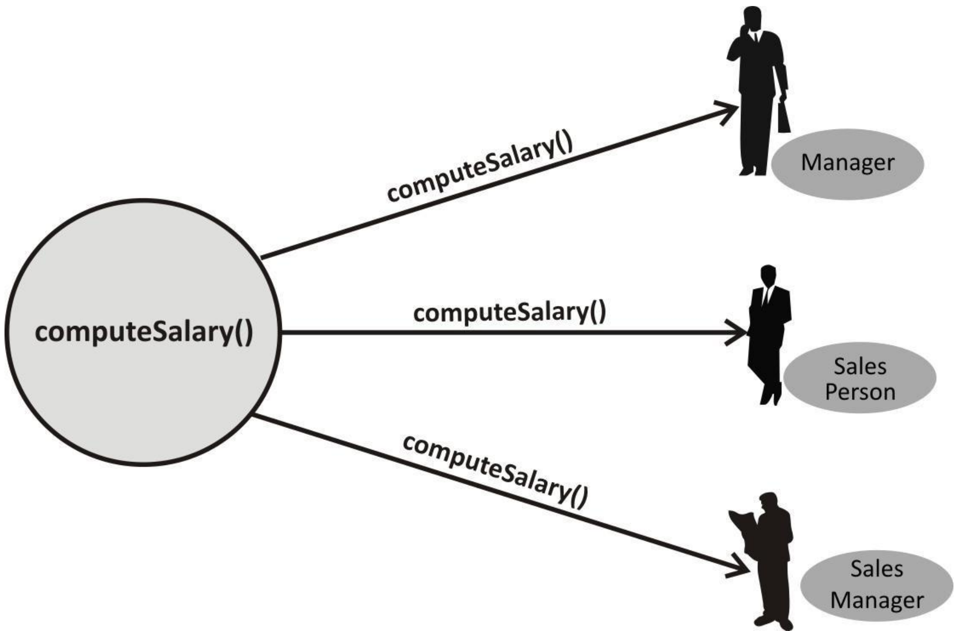
Polymorphism

- The ability of different types of objects to respond to the same message in different ways is called polymorphism.
- Polymorphism helps to :
  - Design extensible software; as new objects can be added to the design without rewriting existing procedures.



Polymorphism

- Ability of different related objects to respond to the same message in different ways is called polymorphism.



Compile-time Binding and Run-time Binding

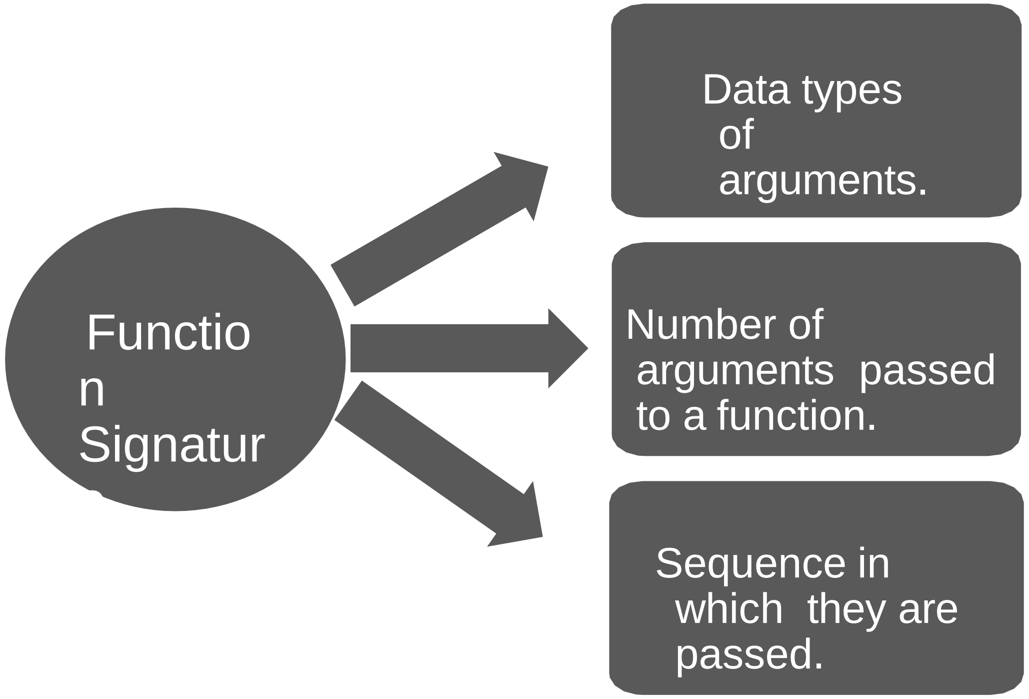
- Binding is an association of function call to an object.
- Compile-time binding
  - The binding of a member function call with an object at compile-time.
  - Also called static type or early binding.
- Run-time binding
  - The binding of the function call to an object at run time.
  - Also called dynamic binding or late binding.
  - Achieved using virtual functions and inheritance.

Function Overloading

- While using function overloading note that:
    - Each function in C++ is name mangled.
    - Name mangling algorithm is different for different compilers, e.g. Microsoft , Borland.
    - Therefore, C++ code compiled under different compilers may not be compatible.
    - Use `extern “C”` directive to suppress name mangling.

Function Overloading

- Using functions with same name but different signatures in the same program is called function overloading.



Function Signatur
Data types of arguments.
Number of arguments passed to a function.
Sequence in which they are passed.

Name Mangling of Overloaded Functions

- Names of overloaded functions are mangled and may look something like this:

```
int sum(int a, int b)     sum@1…….
float sum (float a, float b)   sum@2…….
float sum (int a, float b)          sum@3…….
float sum (float a, int b)          sum@4…….
Void sum(int a, int b, int c)      sum@5……
int sum(int a, int b, int c)      //Not Fun. Overloading
```

Name mangling Example

```
// Name Mangling in function overloading

int f(void) { return 1; }

int f(int) { return 0; }

void g(void) { int i = f(), j = f(0); }
```

Name mangling

```
int___f_v(void) { return 1; }


int___f_i(int) { return 0; }


Void  g_v(void){int i=__f_v(), j =
__f_i(0);}
```

Operator Overloading

The mechanism of giving special meaning to an operator is known as operator overloading.

For example, we can overload an operator ‘+’ in a class like string to concatenate two strings by just using +.

Implementation of Operator overloading:

1. Member function: It is in the scope of the class in which it is declared.

2.Friend function: It is a non-member function of a class with permission to access both private and protected members.

Rule

- To work, at least one of the operand must be a user-defined class object.
- We can only overload the existing operators, Can’t overload new operators.
- Some operators cannot be overloaded using a friend function. However, such operators can be overloaded using the member function.

Which operators Cannot be overloaded?

- Conditional [?:], size of, scope(::), Member selector(.), member pointer selector(.*) and the casting operators.
- We can only overload the operators that exist and cannot create new operators or rename existing operators.

- At least one of the operands in overloaded operators must be user-defined, which means we cannot overload the minus operator to work with one integer and one double. However, you could overload the minus operator to work with an integer and a mystring.
- It is not possible to change the number of operands of an operator supports.
- All operators keep their default precedence and associations (what they use for), which cannot be changed.
- Only built-in operators can be overloaded.

Syntax

```
RT operator Symbol(DT)

{

}
```