C++: Part 4

Vipin Bhatnagar
Dept. of Physics

M.Sc., M.Phil. Courses - 2007

Functions

- We know main function
  - int main(): starting point of the execution of a program
  - Body is { }, the last statement is `return(0);`
- Function Prototyping
  - Prototype describes the function interface to the compiler or it's simply a function declaration statement

type function-name (argument-list);

examples

```
int main()
{
   return (0);
}
float volume(int a, int b, int c)
{
  float v = a*b*c;
  return (v);
}
int main()
{ int a1=2; int b1=3; int c1=4;
  float cubeV = volume(a1,b1,c1);
   return 0;
}
```

In line Functions

- For small functions which are called very frequently by the calling functions
  - Saves lots of execution time in access overheads
  - Function expanded in line when invoked

  inline function-header
  {
  function-body
  }

eg.
```
inline double cubeV(double a)
        {
          return (a*a*a);
        }
```

example

```
inline float mult(float x, float y)
{
  return(x*y);
}
inline double div(double p, double q)
{
  return(p/q);
}
int main()
{
  float a = 12.325; float b = 9.85;
  cout << mult(a,b) << "\n";
  cout << div(a,b) << "\n";
  return(0);
}
```

Default Arguments: functions

```
void println(char ch='*', int len = 40);
// function prototype with default args

//function definition
void println(char ch, int len)
{
  for(int i=1; i<=len; i++) printf("%c",ch);
  printf("\n");
}
int main()
{
  println(); //uses default both args
  println("="); // default for 2nd arg
  return(0);
}
```

Function overloading

- Function polymorphism in OOP
- Overloading means using the same thing for different purposes

```
// function declarations
int add(int a, int b) // prototype1
int add(int a, int b, int c) // type2
```

- Name is same, Number of Args Different
  - Compiler calls the correct function by knowing the argument list

```
cout << add(4,5); // uses type1
cout << add(5,10,15) // uses type2
```

example

```
int volume(int); //prototype decl.
double volume(double, int);
long volume(long, int, int);

// function definitions
int volume(int a)
{ return(a*a*a); } //vol. of a cube
double volume(double r, int h)
{ return(3.141592*r*r*h); } // cylinder
long volume(long l,int b, int h)
{ return(l*b*h); }  // rectangular box

int main()
{
  cout<<volume(10)<<"\n";
  cout<<volume(2.5,8)<<"\n";
  cout<<volume(100L,75,15)<<"\n";
  return(0);
}
```

Friend functions and Virtual functions

Later....when we deal with classes and objects