Scope considerations

```
using namespace std;
#include <iostream>
  const int orbit =365;
  void mars();
  void mercury();
  void unknown();
  void venus();

  int main ()
  { int error, id;
    cout << "Orbit of Earth "<<orbit<<" days \n";
    mercury();
    venus();
    mars();
    unknown();
    return 0;   }
//------------------------------------------------------------------------------------------------------
  void mars()
  { double orbit = 1.8807*365;
    cout << "Orbit of Mars "<<orbit<<" days \n";
    return;  }

  void venus()
  { int orbit = 225;
    cout << "Orbit of Venus "<<orbit<<" days \n";
    return;  }
//------------------------------------------------------------------------------------------------------
  void mercury()
  { int orbit = 88;
    cout << "Orbit of Mercury "<<orbit<<" days \n";
    return;  }
//------------------------------------------------------------------------------------------------------
  void unknown()
  { cout << "Orbit of unknown "<<orbit<<" days \n";
    return;  }
```

These are the prototypes for the functions of the planets. Notice that there are no parameters or return values.

These are the calls for the functions of the planets. Notice that there are no parameters or return values.

Notice that value of the variable ***orbit*** is different in each these functions. These are considered to be values which are local in scope. They override the global value of ***orbit***.

Since there is no local for the variable ***orbit,*** it uses the global value of ***orbit.*** This is also sometimes referred to as the enclosing scope.

```
using namespace std;
#include <iostream>
void gethours(double & total);
int main ()
  { int id,i ;
    double gross, total, rate, net, fed, state, fica, fedtax, statetax;
//-----------------------------------------------------------------------------
     for (i=0; i<3; i++)
      {  cout << " Enter id ";
        cin >> id;
        gethours(total);
        cout << " Enter Rate ==> ";
        cin >> rate;
 //----------------------------------------------------------------------------
        gross = total*rate;
        net = gross*0.7;
 //----------------------------------------------------------------------------
        cout << "Gross : $ " <<  gross <<  "\t" << "Hours : " <<  total <<  endl;
        cout << "Rate  : " <<  rate <<  endl << "net  : " <<  net <<  endl;
      }
    return (0);
  }
```

This variable I is for the main program and is not changed in any function it calls unless it is passed as a parameter to be changed.

```
void gethours(double & total)
 { int i;
   double hours;
   total=0;
   for (i=0; i< 7; i++)
     {
       cout << " Enter Hours  for day "<<i<<" ==> ";
       cin >> hours;
       total=total+hours;
     }
    return;
 }
```

Notice that the counter I is listed again. The I listed here is the counter I in the function ***gethours.*** If the variable was not declared locally, it would be an invalid reference since the counter ***I*** in the main program is only for that environment.

The function ***gethours*** is considered to be an entirely different program from the main program.

The values for the local variables I and hours only exist while this function is "alive" and are destroyed on exit of the function.