Scope of variables

- **Scope (lifetime) of a variable (local vs. global variable)**
- **Name precedence**
- **Automatic variable :** a variable for which memory is allocated and de-allocated when control enters and exists the block in which it is declared

Example 1:

```
#include <iostream>
using namespace std;
int i;
void ModifyI();
int main()
{
   i=1;

   cout << "before calling ModifyI, i equals to " << i << endl;
   ModifyI();
   cout << "after calling ModifyI, i equals to " << i << endl;

   return 0;
}

void ModifyI()
{
    int i=3;
    return ;
}
```

Example 2:

```
#include <iostream>
using namespace std;
int Fnc(int z);
int x, y;

int main()   {
    int x=3;

    y=x;
    cout << "In main before calling Fnc x = " << x << ", y = " << y << endl;

    x = Fnc(4);
    cout << "In main after calling Fnc x = " << x << ", y = " << y << endl;

     return 0;
}

int Fnc(int z)  {
    x=10;
    y=x+z;
    cout << "In fnc x = " << x << ", y = " << y << "z = " << z << endl;

     return 50;
}
```

Static local variable vs. automatic variable vs. global variable

static local variable :

- used when the value of a local variable of a function needs to be carried over between function calls,
- its scope is local to the function
- exist throughout the program execution

Example 3:

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

int Fnc(int z);
int x, y;

int main()
{
    int x=3;

    y=x;
    cout << "In main before calling Fnc x = " << x << ",  y = " << y << endl;

    x = Fnc(4);
    cout << "In main after calling Fnc x = " << x << ", y = " << y << endl;

     x = Fnc(5);
    cout << "In main after calling Fnc x = " << x << ", y = " << y << endl;

     return 0;
}


int Fnc(int z)
{
    static  int y=10;   // initialized once only
    z = 1 ;    // initialized each time

    x=10+z;
    y=y+z;
    cout << "In fnc x = " << x << ", y = " << y << ", z = " << z << endl;

    return z;
}
```

Example 4:

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

int value=28;

void Func();

int main()
{
   int  value = 8;

   cout << "global value = " << ::value <<endl;
   cout << "main function value = " << value << endl;

   return 0;
}
```

Namespace

- namespace is a mechanism by which the programmer can create **a named scope**.

Example 5:

```
// in header file cstdlib:
namespace  std
{
   …
   int abs(int);
}
```

Three ways to access something defined in a namespace:

First way:

```
#include<cstdlib>
int main()  {
   int alpha;
   int alphb;
   …
   alpha = std::abs(beta);
   std::cout << alpha;
   …
}
```

Second way:

```
#include<cstdlib>
int main()  {
   int alpha;
   int alphb;
   using std::abs;
   using std::cout;
   …
   alpha = abs(beta);
   cout << alpha;
   …
}
```

Third way:

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

int main()  {
   int alpha;
   int alphb;
   …
   alpha = abs(beta);
   cout << alpha;
   …
}
```