CS118 – Programming Fundamentals

Lecture # 20
Tuesday, October 30, 2019
FALL 2019
FAST – NUCES, Faisalabad Campus

Zain Iqbal

Local and Global Variables

- **Local variable:** Defined within a function or block; accessible only within the function or block
- Other functions and blocks can define variables with the **same name**
- When a function is called, local variables in the calling function are **not accessible** from within the called function
- C++ **does not** allow the **nesting of functions**. That is, you cannot include the definition of one function in the body of another function.

Local and Global Variables

- **Global variable:** A variable defined outside all functions; it is accessible to all functions within its scope
- Easy way to **share large amounts of data** between functions
- **Scope of a global variable** is from its point of definition to the program end
  - Use cautiously

Local Variable Lifetime

- A local variable only **exists** while its defining function is executing
- Local variables are **destroyed** when the function terminates
- Data **cannot be retained** in local variables defined in a function between calls to the function

Initializing Local and Global Variables

- **Local** variables **must be initialized** by the programmer
- **Global** variables are initialized to **0 (numeric)** or NULL (character) when the variable is defined

Local and Global Variable Names

- Local variables **can** have **same names** as global variables
- When a function contains a local variable that has the same name as a global variable, the global variable is unavailable from within the function
- The local definition "hides" or "shadows" the global definition

If Local and Global Variable have different name


```cpp
#include <iostream>
using namespace std;
int t;
void funOne(int& a);
int main()
{
int x = 15; //Line 1
cout << "Line 2: In main: t = " << t << endl; //Line 2
funOne(x); //Line 3
cout << "Line 4: In main after funOne: "
<< " t = " << t << endl; //Line 4
return 0; //Line 5
}
void funOne(int& a)
{
cout << "Line 6: In funOne: a = " << a
<< " and t = " << t << endl; //Line 6
a = a + 12; //Line 7
cout << "Line 8: In funOne: a = " << a
<< " and t = " << t << endl; //Line 8
t = t + 13; //Line 9
cout << "Line 10: In funOne: a = " << a
<< " and t = " << t << endl; //Line 10
}
```

```
Line 2: In main: t = 0
Line 6: In funOne: a = 15 and t = 0
Line 8: In funOne: a = 27 and t = 0
Line 10: In funOne: a = 27 and t = 13
Line 4: In main after funOne:  t = 13
```

If Local and Global have same name


```
#include <iostream>
using namespace std;
int t;
void funOne(int& a);
int main()
{
t = 15; //Line 1
cout << "Line 2: In main: t = " << t << endl; //Line 2
funOne(t); //Line 3
cout << "Line 4: In main after funOne: "
<< " t = " << t << endl; //Line 4
return 0; //Line 5
}
void funOne(int& a)
{
cout << "Line 6: In funOne: a = " << a
<< " and t = " << t << endl; //Line 6
a = a + 12; //Line 7
cout << "Line 8: In funOne: a = " << a
<< " and t = " << t << endl; //Line 8
t = t + 13; //Line 9
cout << "Line 10: In funOne: a = " << a
<< " and t = " << t << endl; //Line 10
}
```

```
Line 2: In main: t = 15
Line 6: In funOne: a = 15 and t = 15
Line 8: In funOne: a = 27 and t = 27
Line 10: In funOne: a = 40 and t = 40
Line 4: In main after funOne:  t = 40
```

Static Local Variables

- **Local variables**
  - Only exist while the function is executing
  - Are redefined each time function is called
  - Lose their contents when function terminates
- **static local variables**
  - Are defined with key word static

    static int counter;
  - Are defined and initialized **only the first time** the function is executed
  - Retain their contents between function calls
  - Better to initialize when declared

    static int counter = 0 ;

static **variable illustrated**


```
//Program: Static and automatic variables

#include <iostream>

using namespace std;

void test();

int main()
{
    int count;

    for (count = 1; count <= 5; count++)
        test();

    return 0;
}

void test()
{
    static int x = 0;
    int y = 10;

    x = x + 2;
    y = y + 1;

    cout << "Inside test x = " << x << " and y = "
         << y << endl;
}
```

Sample Run:

```
Inside test x = 2 and y = 11
Inside test x = 4 and y = 11
Inside test x = 6 and y = 11
Inside test x = 8 and y = 11
Inside test x = 10 and y = 11
```

Questions