Local, Global, and static Local variables

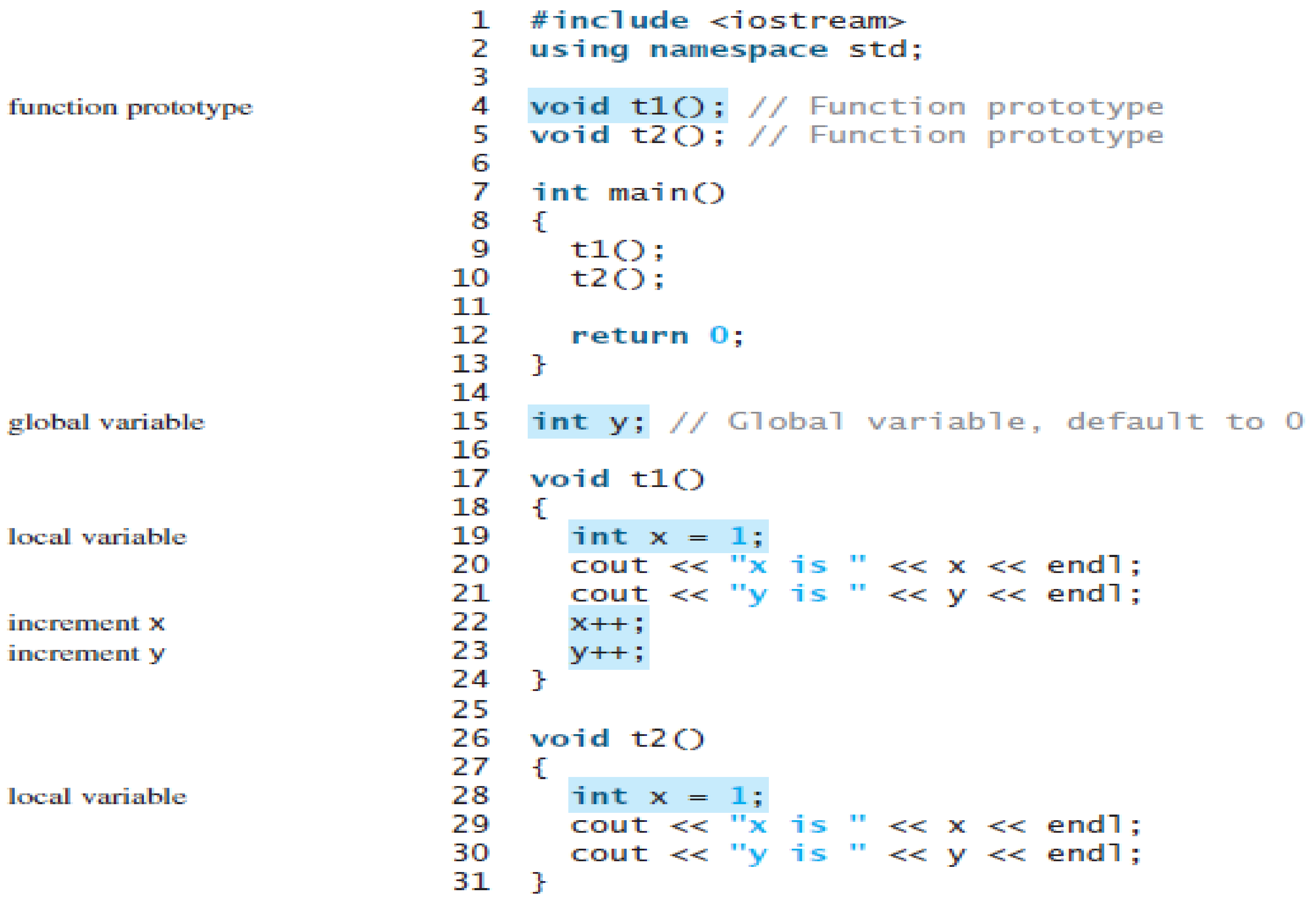
Scope of a Variable

- *A variable can be declared as a local, a global, or a static local in C++.*
- The *scope of a variable* is the part of the program where the variable can be referenced.
- Variable defined inside a function is referred to as a *local variable*.
- C++ also allows you to use *global variables*. They are declared outside all functions and are accessible to all functions in their scope.
- Local variables do not have default values, but global variables are defaulted to zero.

Scope of a Variable

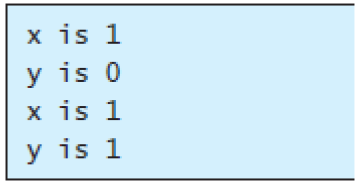
- A variable must be declared before it can be used.
- The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable.
- The scope of a global variable starts from its declaration and continues to the end of the program.
- A parameter is actually a local variable. The scope of a function parameter covers the entire function.

Example demonstrates the scope of local and global variables.



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

void t1(); // Function prototype
void t2(); // Function prototype

int main()
{
  t1();
  t2();

  return 0;
}

int y; // Global variable, default to 0

void t1()
{
  int x = 1;
  cout << "x is " << x << endl;
  cout << "y is " << y << endl;
  x++;
  y++;
}

void t2()
{
  int x = 1;
  cout << "x is " << x << endl;
  cout << "y is " << y << endl;
}
```

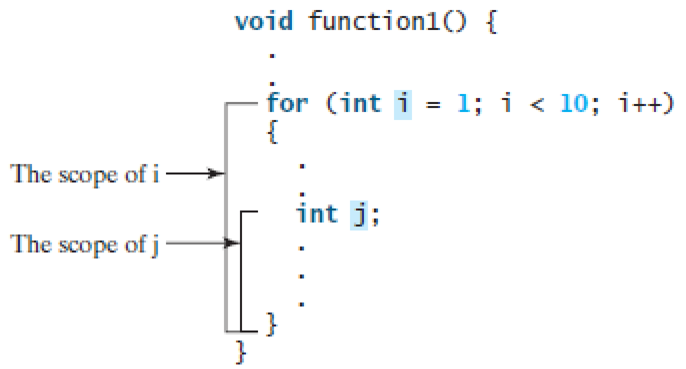
output



```
x is 1
y is 0
x is 1
y is 1
```

The Scope of Variables in a for Loop

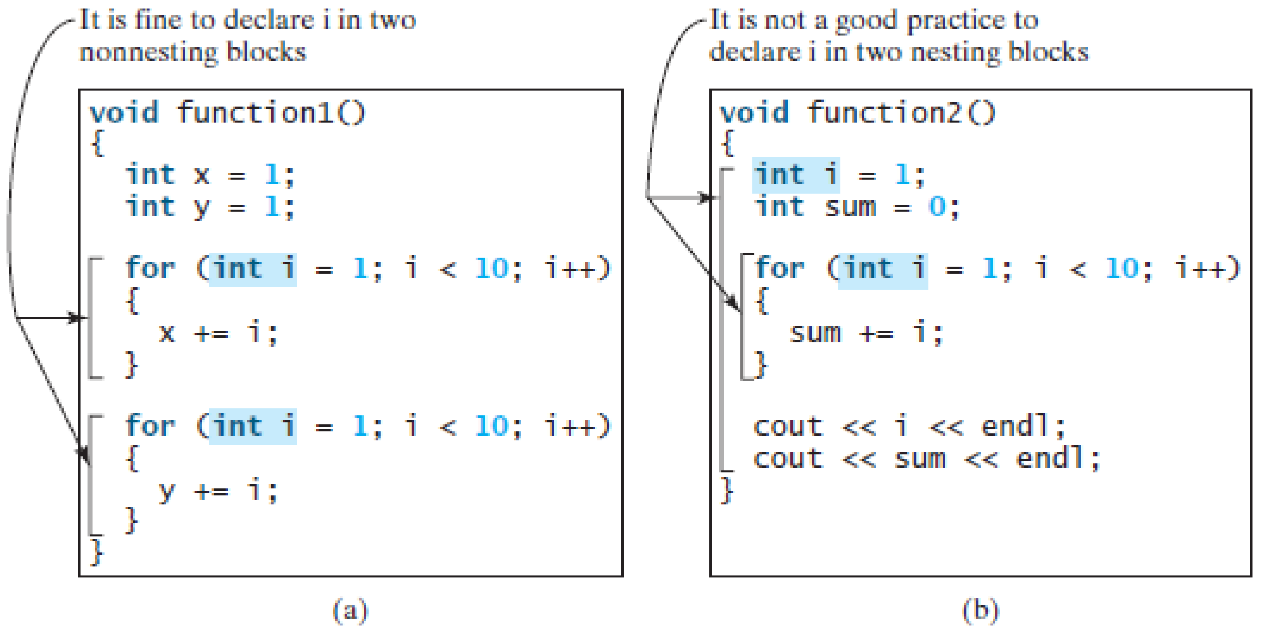
- A variable declared in the initial-action part of a for-loop header has its scope in the entire loop.
- However, a variable declared inside a for-loop body has its scope limited in the loop body from its declaration to the end of the block that contains the variable, as shown in



```
void function1() {
  .
  .
  for (int i = 1; i < 10; i++)
  {
    .
    .
    int j;
    .
    .
    .
  }
}
```

The Scope of Variables in a for Loop

- A variable can be declared multiple times in nonnesting blocks, but you should avoid declaring them in nesting blocks.


It is fine to declare i in two nonnesting blocks
It is not a good practice to declare i in two nesting blocks
(a)
(b)

```
void function1()
{
  int x = 1;
  int y = 1;

  for (int i = 1; i < 10; i++)
  {
    x += i;
  }

  for (int i = 1; i < 10; i++)
  {
    y += i;
  }
}
```

```
void function2()
{
  int i = 1;
  int sum = 0;

  for (int i = 1; i < 10; i++)
  {
    sum += i;
  }

  cout << i << endl;
  cout << sum << endl;
}
```

The Scope of Variables in a for Loop

Caution

- Do not declare a variable inside a block and then attempt to use it outside the block.

Here is an example of a common mistake:

```
for (int i = 0; i < 10; i++)
{
}
cout << i << endl;
```

- The last statement would cause a syntax error, because variable i is not defined outside the for loop.

Static Local Variables

- After a function completes its execution, all its local variables are destroyed. These variables are also known as *automatic variables*. Sometimes it is desirable to retain the values stored in local variables so that they can be used in the next call.
- C++ allows you to declare static local variables. *Static local variables* are permanently allocated in the memory for the lifetime of the program. To declare a static variable, use the keyword static.

Example demonstrates using static local variables.

function prototype

invoke t1

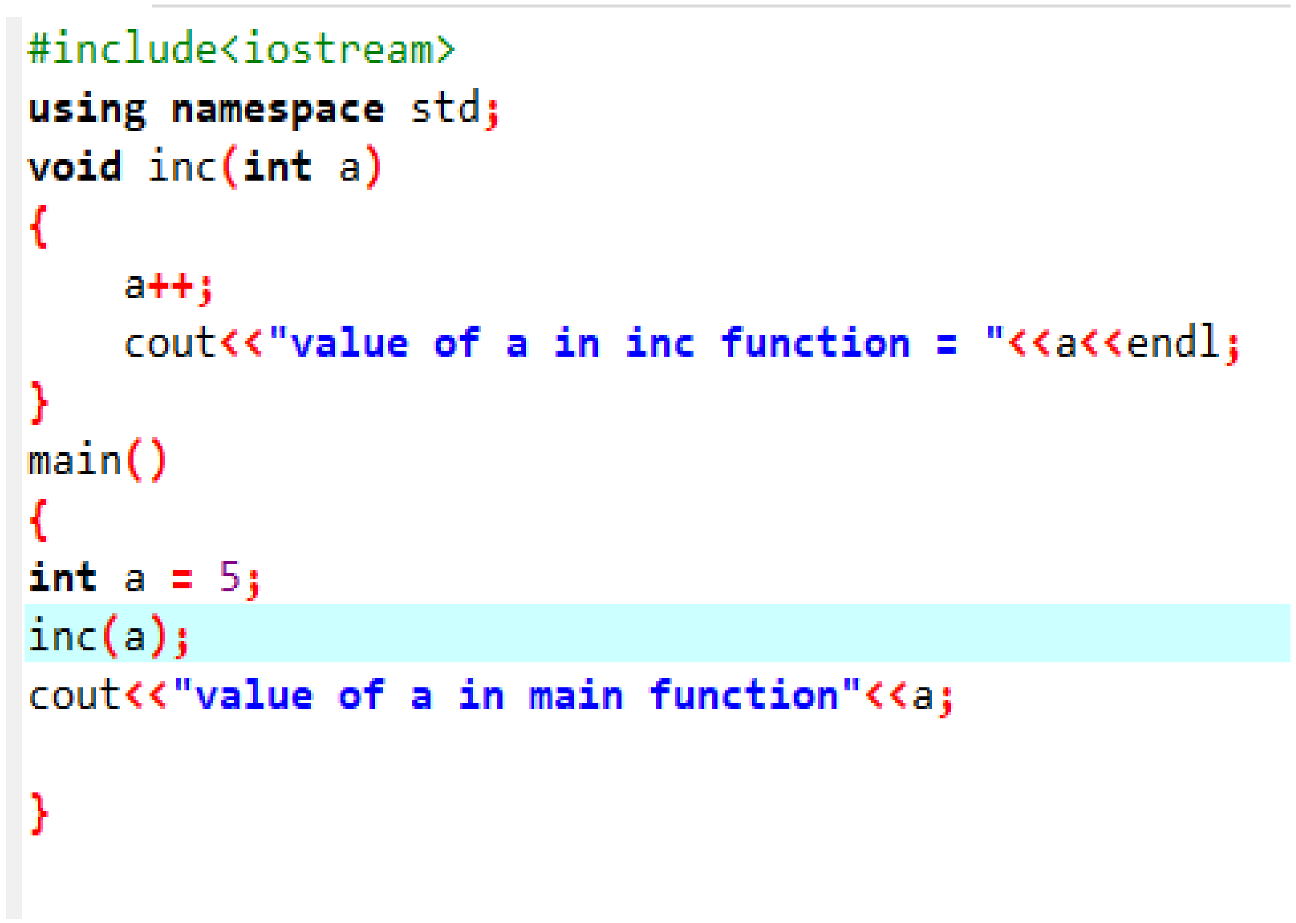
static local variable
local variable
increment x
increment y

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

void t1(); // Function prototype

int main()
{
  t1();
  t1();

  return 0;
}

void t1()
{
  static int x = 1;
  int y = 1;
  x++;
  y++;
  cout << "x is " << x << endl;
  cout << "y is " << y << endl;
}
```

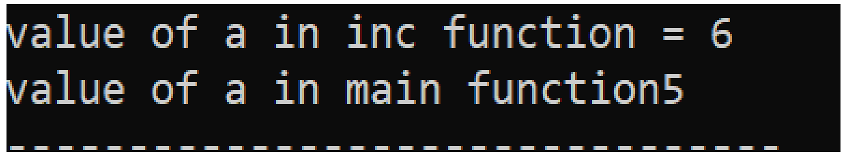
output

```
x is 2
y is 2
x is 3
y is 2
```

Class Activity: Print Output



```
#include<iostream>
using namespace std;
void inc(int a)
{
    a++;
    cout<<"value of a in inc function = "<<a<<endl;
}
main()
{
int a = 5;
inc(a);
cout<<"value of a in main function"<<a;

}
```



```
value of a in inc function = 6
value of a in main function5
-----------------------------------
```