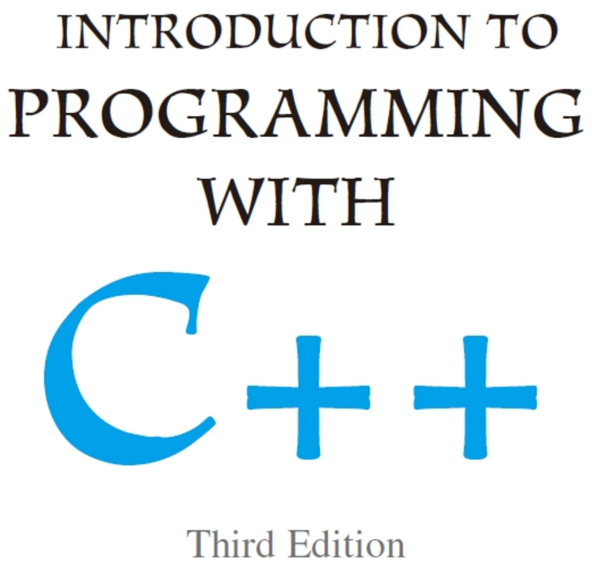
Introduction to Programming with C++

Chieh-Sen (Jason) Huang

Department of Applied Mathematics

National Sun Yat-sen University



INTRODUCTION TO PROGRAMMING WITH C++

Third Edition

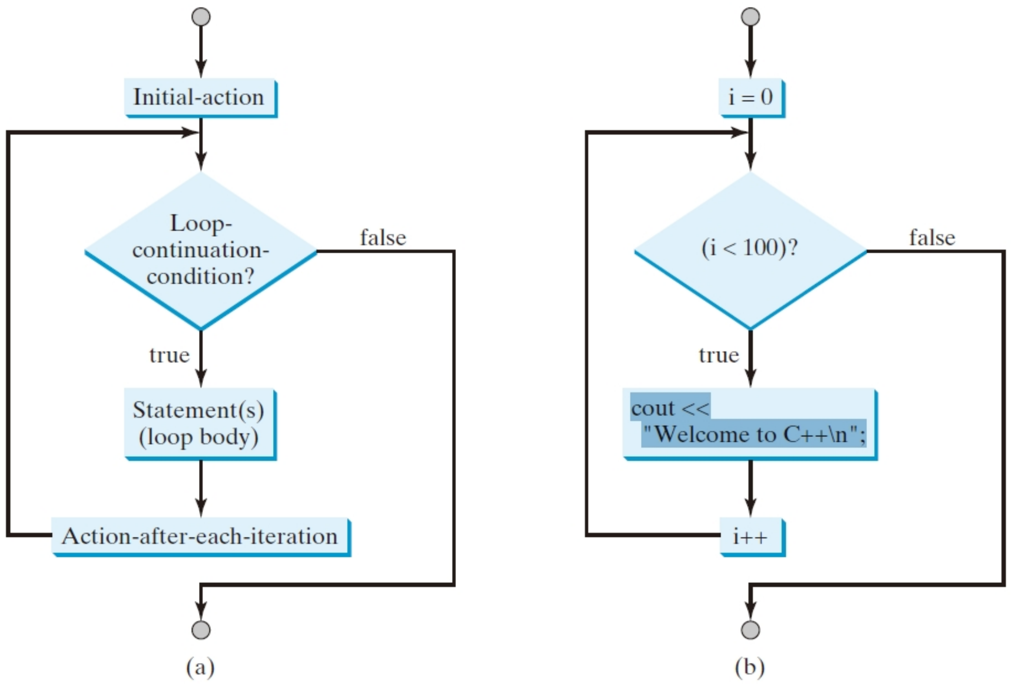
Contents are based on book by Y. Daniel Liang

Loops

- A loop can be used to tell a program to execute statements repeatedly.
- C++ provides a powerful construct called a loop that controls how many times an operation or a sequence of operations is performed in succession.
- The for-loop statement starts with the keyword **for**, followed by a pair of parentheses enclosing **initial-action**, **loop-continuation-condition**, and **action-after-each- iteration**, followed by the loop body enclosed inside braces. initial-action, loop-continuation-condition, and action-after-each-iteration are separated by semicolons.

Loops

```
for (initial-action; loop-continuation-condition;
     action-after-each-iteration)
{
  // Loop body;
  Statement(s);
}
-------------------------------------------------------
for (i = 0; i < 2; i++)
{
  cout << "Welcome to C++\n";
}
cout << "i = " << i << endl;
--------------------------------------------------------
Welcome to C++
Welcome to C++
2
```


Initial-action
Loop-continuation-condition?
false
true
Statement(s) (loop body)
Action-after-each-iteration
(a)
i = 0
(i < 100)?
false
true
cout << "Welcome to C++\n";
i++
(b)

Loops

```
//  1+2+3
{
     int i;
     int Sum = 0; //  set initial value

     for(i = 1; i <= 3; i++)
     {
          Sum += i;
          cout << i << "\t" << Sum << endl;
     }
     return 0;
}
------------------------------------------
1      1
2      3
3      6
```

- We could also do

```
{
     int i = 1;
     int Sum = 0; //  set initial value
     for( ; i <= 3; )
     {
          Sum += i;
          cout << i++ << "\t" << Sum << endl;
     }
     return 0;
}
```

Nested Loops

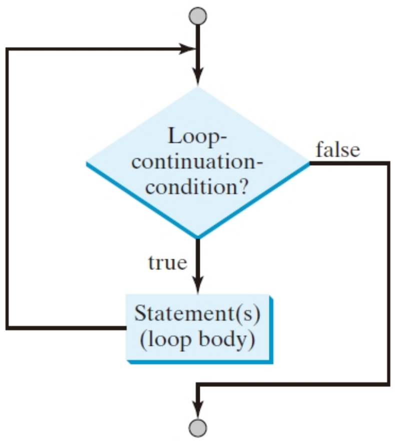
```
for (initial-action; loop-continuation-condition;
     action-after-each-iteration)
{
  for (initial-action; loop-continuation-condition;
     action-after-each-iteration)
  {
    // Statement(s) of inner loop;
  }
  // Statement(s) of outer loop;
}
-------------------------------------------------------
for(i = 1; i <= 9; i++)      //  outer loop (column-wise)
{
  for(j = 1 ; j <= 9; j++)   //  inner loop (row-wise)
    cout << '\t' << i << '*' << j << '=' << i*j;
  cout<<endl;
}
```

Exercise: Write a code to do the upper triangle part of multiplication table.
Homewrok: Write a code to do the diamond part of multiplication table.

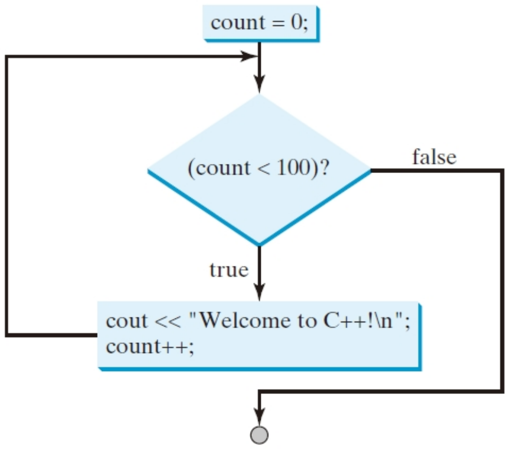
The while Loop

- A while loop executes statements repeatedly while the condition is true.

```
while (loop-continuation-condition)
{
  // Loop body
  Statement(s);
}
-----------------------------------------------------
int i = 0;                   |
while (i < 2){               |for (i = 0; i < 2; i++){
  cout << "Welcome to C++\n";| cout << "Welcome to C++\n";
  i++;                       |
}                            |}
```



(a)



(b)

Caution when using Loop

- Make sure that the **loop-continuation-condition** eventually becomes false so that the loop will terminate.

```
  while (true) // loop-continuation-condition is always true
    cout << "infinite loop...\n";
//-------------------------------------------------------------
  for(;;)
    cout << "infinite loop...\n";
```

- If you are running the program from the command window, press CTRL+C to stop it.

- Using **break** to immediately breaks out the loop.

```
int i = 0;
while (true){ // loop-continuation-condition is always true
   cout << "infinite loop...\n";
   if (i == 5) // when i = 5,
      break;    // immediately breaks out the loop
   i++;
}
```

off-by-one error

- Programmers often make the mistake of executing a loop one more or one less time. This is commonly known as the off-by-one error.
- The following loop displays Welcome to C++ 101 times rather than 100 times. The error lies in the condition, which should be count < 100 rather than count <= 100.

```
int count = 0;
while (count <= 100)
{
  cout << "Welcome to C++!\n";
  count++;
}
cout << i << endl;
```

- What is the value of i at the end?

Controlling a Loop with User Confirmation

- If you want the user to decide whether to continue, you can offer a user confirmation.

```
char continueLoop = 'Y';
while (continueLoop == 'Y')
{
  // Execute the loop body once
  ...
  // Prompt the user for confirmation
  cout << "Enter Y to continue and else to quit: ";
cin >> continueLoop;
}
```

- Controlling a Loop with a Sentinel Value, See List4_5.cpp.

Finding the Greatest Common Divisor

- while is used when the number of iteration is unknown.

```
int a, b = num2, c = num1 % num2;   // c is the reminder
                                 //  after the first division

while (c != 0) {     // when reminder equals to 0 ,
  a = b;             // b is the GCD
  b = c;
  c = a % b;         // Euclidean algorithm,take the reminder
}

cout << "The greatest common divisor for " << a << " and "
     << b << " is " << b << endl;

return 0;
```

- See List5_10.cpp

continue keyword in a loop

- Using **continue** to breaks out the **iteration**.

```
int i = 0;
while (true){ // loop-continuation-condition is always true
   cout << "infinite loop...\n";
   if (i == 5) // when i = 5,
      continue;     // immediately breaks out the loop
   i++;
}
```

- Obviously, the break statement makes this program simpler and easier to read. However, you should use break and continue with caution.

- Too many break and continue statements will produce a loop with many exit points and make the program difficult to read.

goto

- Some programming languages including C++ have a **goto** statement.
- The **goto** statement indiscriminately transfers control to any statement in the program and executes it. This makes your program vulnerable to errors.
- See List5_17.cpp

  Homework: 5.26, 5.27. PE: 5.19, $\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}$.