C++ 순환문과 조건문

금융공학 프로그래밍

If statement

```
statement1
if (test_expr)
      statement2
statement3
```

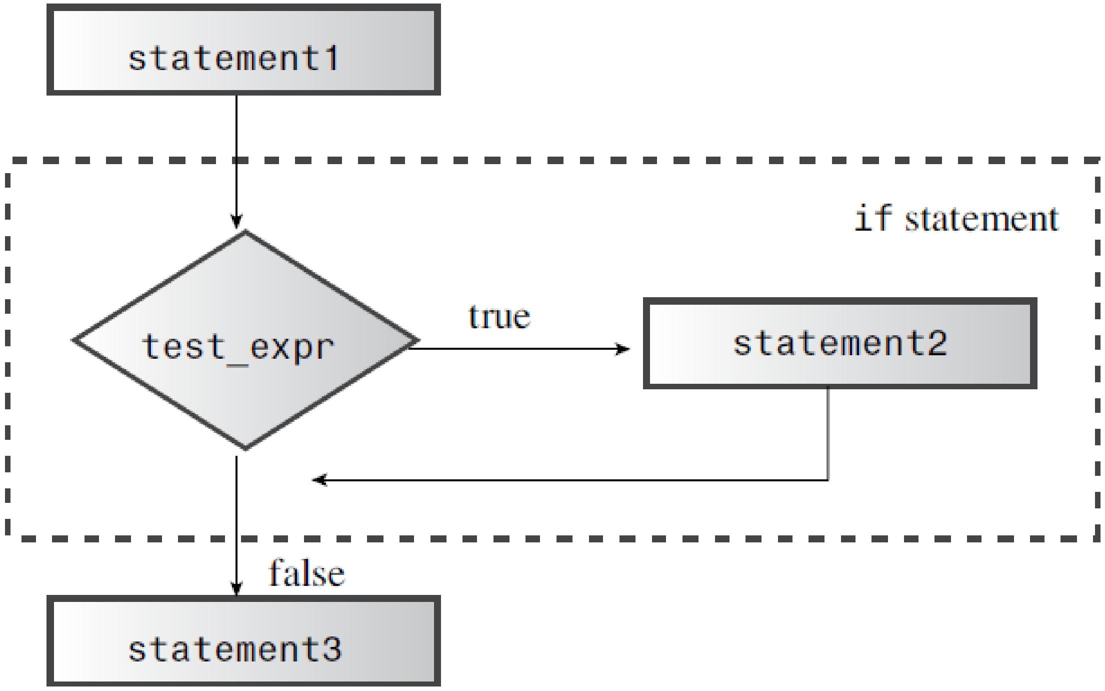


Figure 6.1 The structure of `if` statements.

Listing 6.1 **if.cpp**

```
// if.cpp -- using the if statement
#include <iostream>
int main()
{
    using std::cin;     // using declarations
    using std::cout;
    char ch;
    int spaces = 0;
    int total = 0;
    cin.get(ch);
    while (ch != '.')   // quit at end of sentence
    {
        if (ch == ' ')  // check if ch is a space
            ++spaces;
        ++total;        // done every time
        cin.get(ch);
    }
    cout << spaces << " spaces, " << total;
    cout << " characters total in sentence\n";
    return 0;
}
```

If-else statement

```
statement1
if (test_expr)
      statement2
else
      statement3

statement4
```

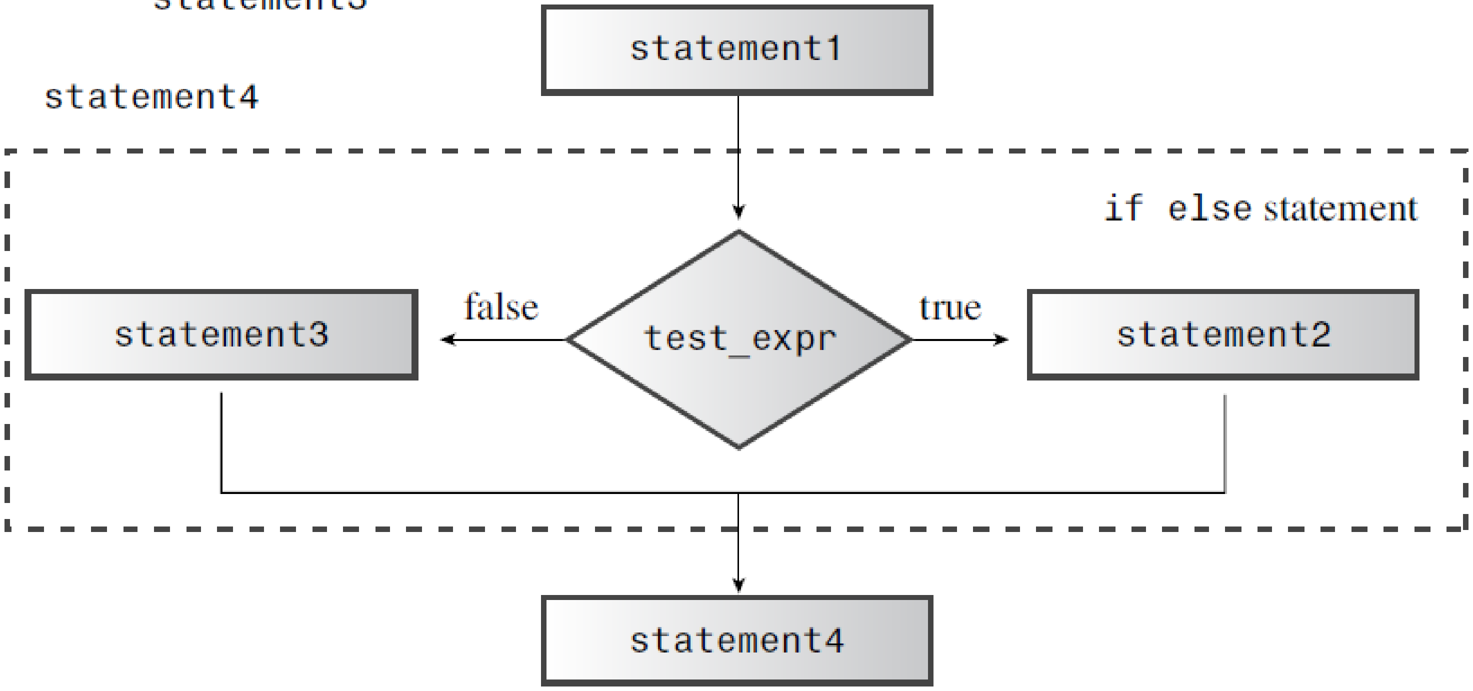


Figure 6.2 The structure of `if else` statements.

Listing 6.3 **ifelseif.cpp**

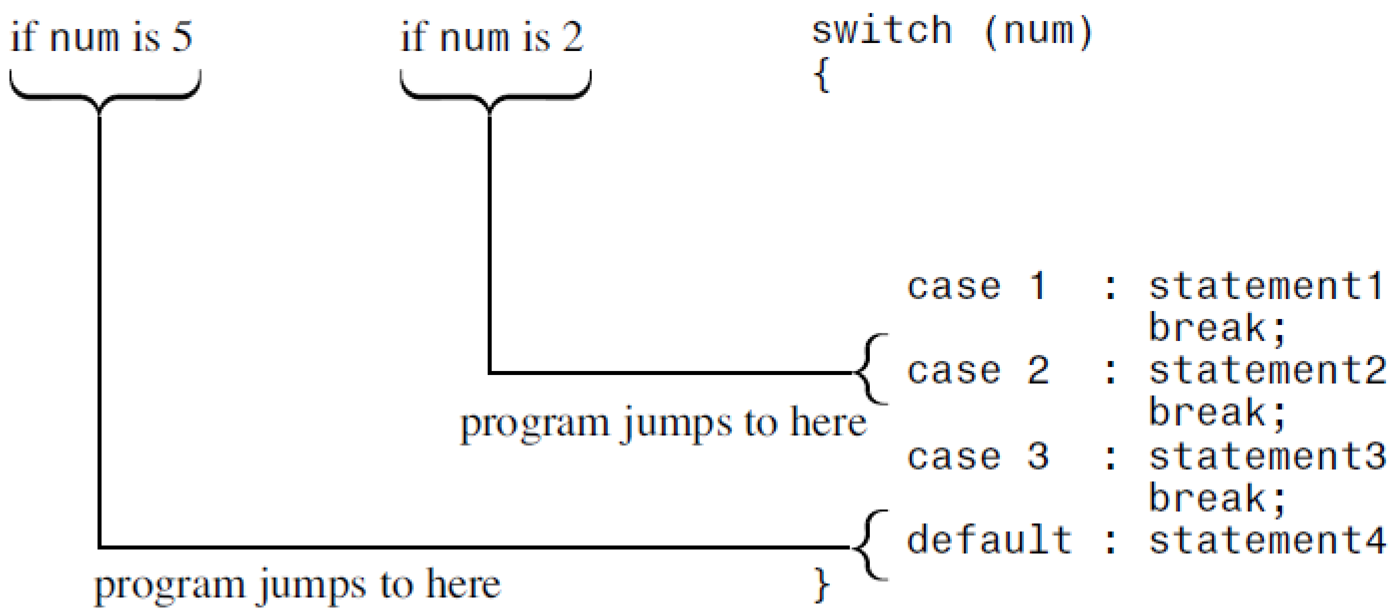
```
// ifelseif.cpp -- using if else if else
#include <iostream>
const int Fave = 27;
int main()
{
    using namespace std;
    int n;

    cout << "Enter a number in the range 1-100 to find ";
    cout << "my favorite number: ";
    do
    {
        cin >> n;
        if (n < Fave)
            cout << "Too low -- guess again: ";
        else if (n > Fave)
            cout << "Too high -- guess again: ";
        else
            cout << Fave << " is right!\n";
    } while (n != Fave);
    return 0;
}
```

? operator

Listing 6.9 **`condit.cpp`**

```
// condit.cpp -- using the conditional operator
#include <iostream>
int main()
{
    using namespace std;
    int a, b;
    cout << "Enter two integers: ";
    cin >> a >> b;
    cout << "The larger of " << a << " and " << b;
    int c = a > b ? a : b;   // c = a if a > b, else c = b
    cout << " is " << c << endl;
    return 0;
}
```

Switch statement



```
switch (num)
{
        case 1   : statement1
                   break;
        case 2   : statement2
                   break;
        case 3   : statement3
                   break;
        default  : statement4
}
```

Figure 6.3 The structure of `switch` statements.

Listing 6.11 **enum.cpp**

```
// enum.cpp -- using enum
#include <iostream>
// create named constants for 0 - 6
enum {red, orange, yellow, green, blue, violet, indigo};

int main()
{
    using namespace std;
    cout << "Enter color code (0-6): ";
    int code;
    cin >> code;
    while (code >= red && code <= indigo)
    {
        switch (code)
        {
            case red     : cout << "Her lips were red.\n"; break;
            case orange  : cout << "Her hair was orange.\n"; break;
            case yellow  : cout << "Her shoes were yellow.\n"; break;
            case green   : cout << "Her nails were green.\n"; break;
            case blue    : cout << "Her sweatsuit was blue.\n"; break;
            case violet  : cout << "Her eyes were violet.\n"; break;
            case indigo  : cout << "Her mood was indigo.\n"; break;
        }
        cout << "Enter color code (0-6): ";
        cin >> code;
    }
    cout << "Bye\n";
    return 0;
}
```

For loop

```
statement1
for (int_expr; test_expr; update_expr)
     statement2
statement3
```

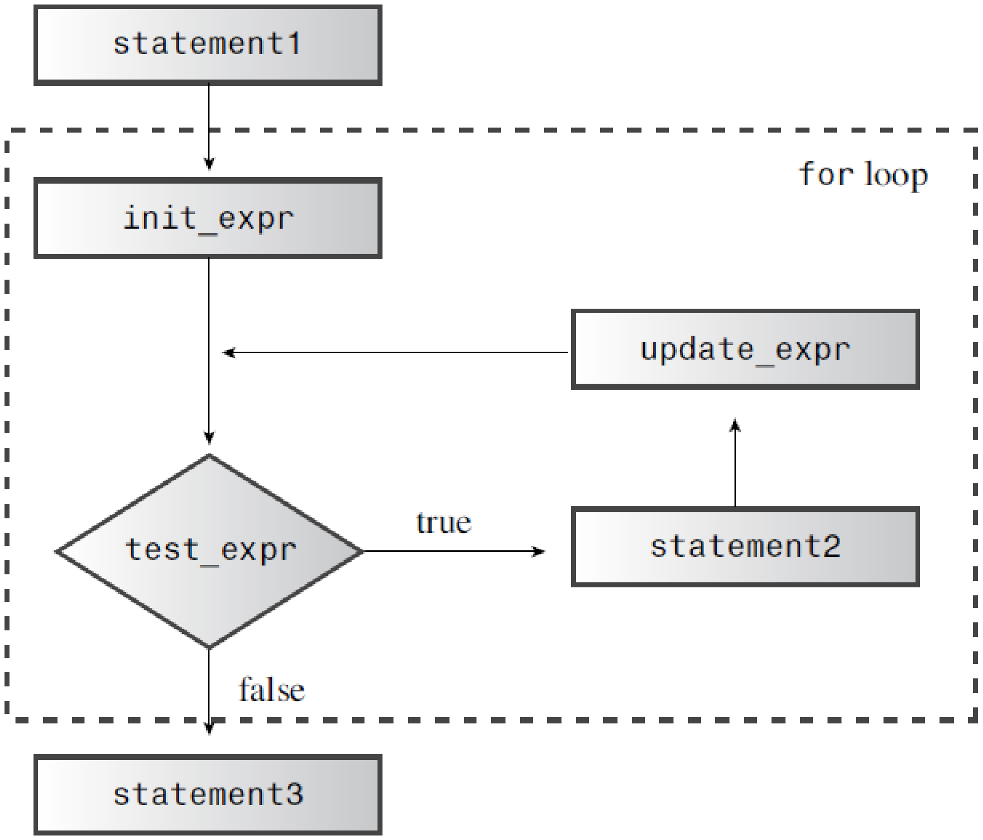


Figure 5.1 The design of `for` loops.

For loop

Listing 5.9 **forstr2.cpp**

```
// forstr2.cpp -- reversing an array
#include <iostream>
#include <string>
int main()
{
    using namespace std;
    cout << "Enter a word: ";
    string word;
    cin >> word;

    // physically modify string object
    char temp;
    int i, j;
    for (j = 0, i = word.size() - 1; j < i; --i, ++j)
    {                       // start block
        temp = word[i];
        word[i] = word[j];
        word[j] = temp;
    }                       // end block
    cout << word << "\nDone\n";
    return 0;
}
```

For loop

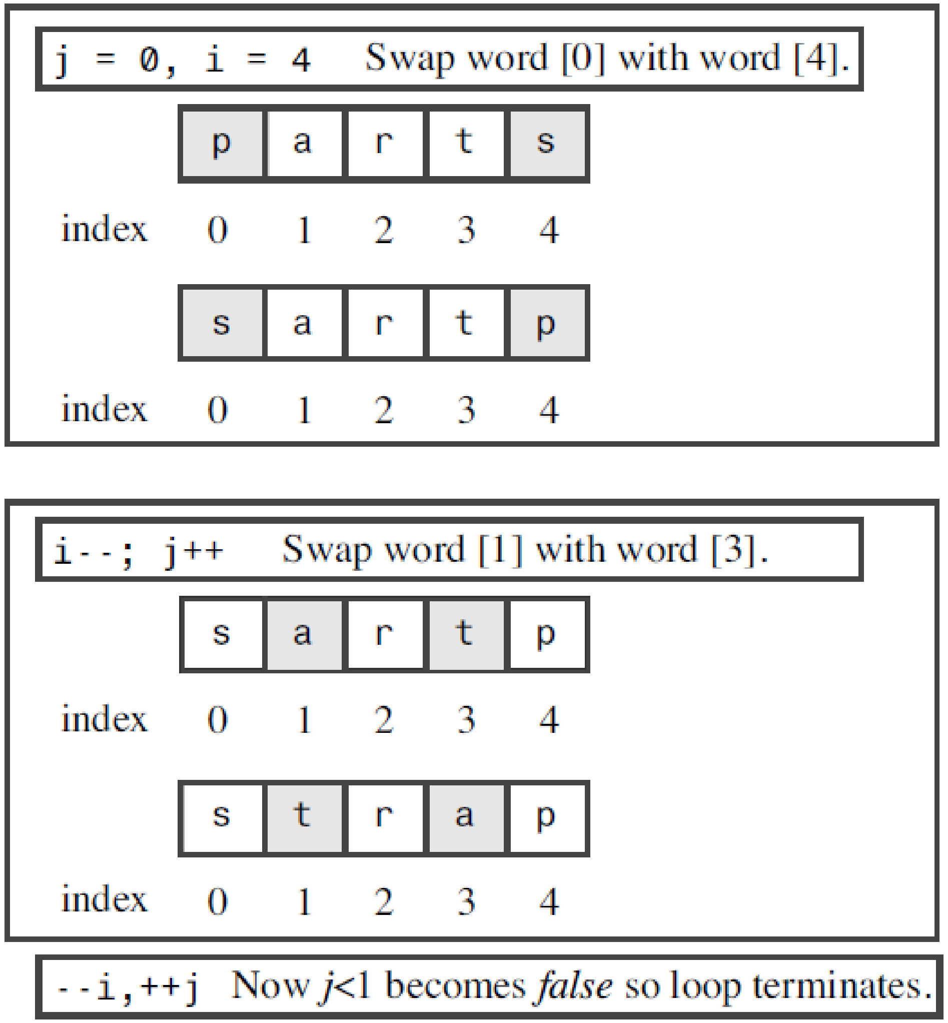

j = 0, i = 4 Swap word [0] with word [4].
p a r t s
index 0 1 2 3 4
s a r t p
index 0 1 2 3 4
i--; j++ Swap word [1] with word [3].
s a r t p
index 0 1 2 3 4
s t r a p
index 0 1 2 3 4
--i,++j Now j<1 becomes false so loop terminates.

Figure 5.2 Reversing a string.

While loop

```
statement1
while (test_expr)
      statement2
statement3
```

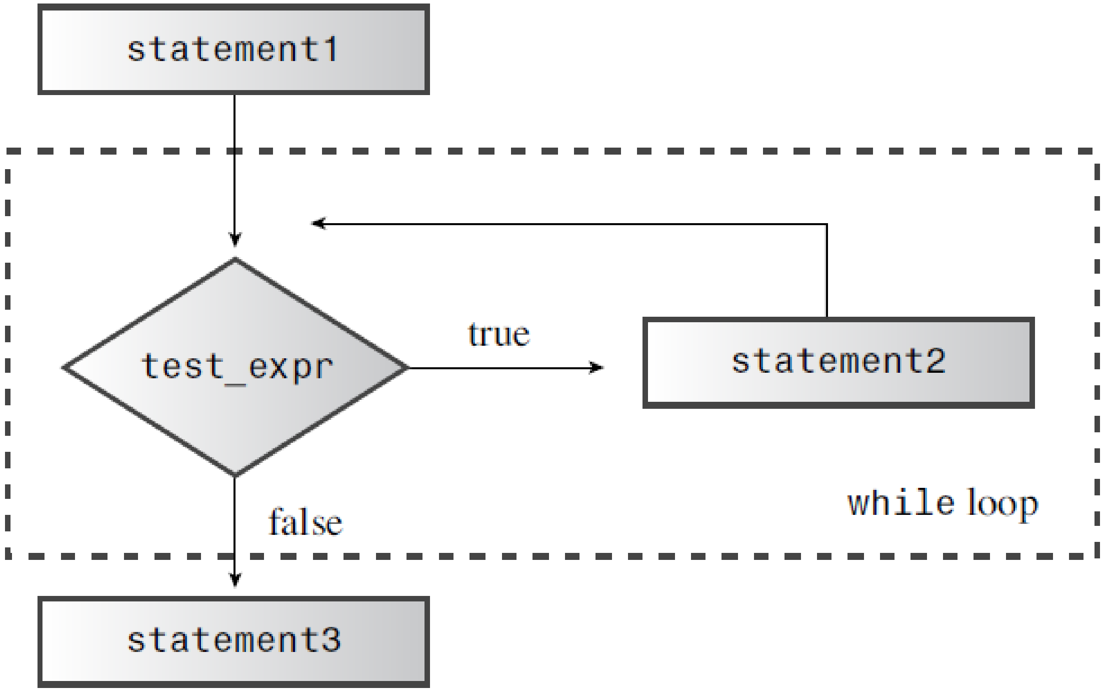


Figure 5.3 The structure of `while` loops.

Do-while loop

```
statement1
do
        statement2
while (test_expr);
statement3
```

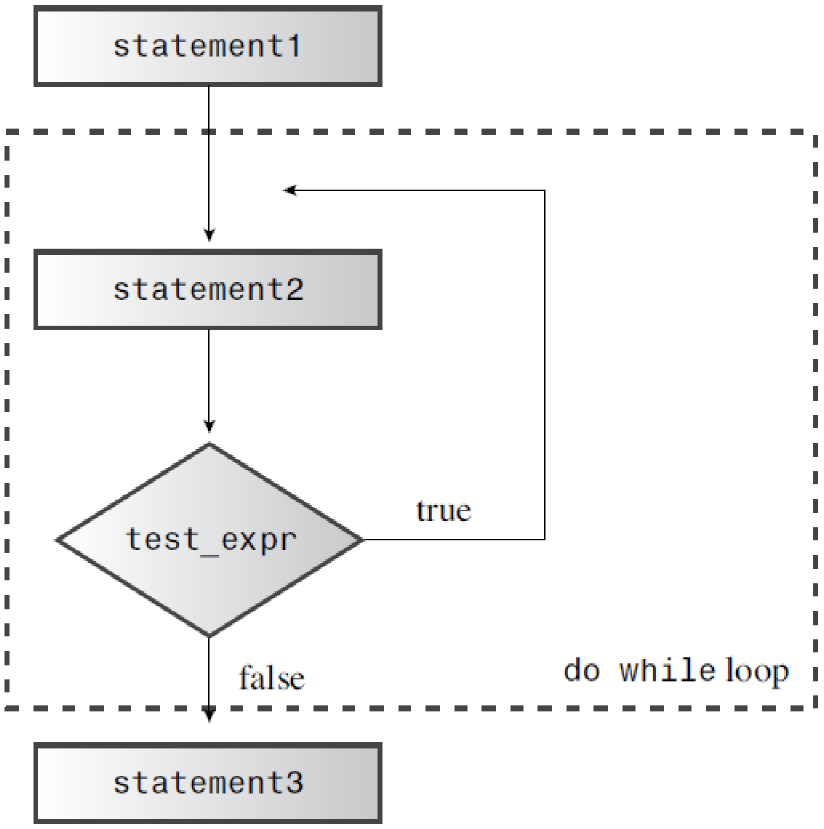


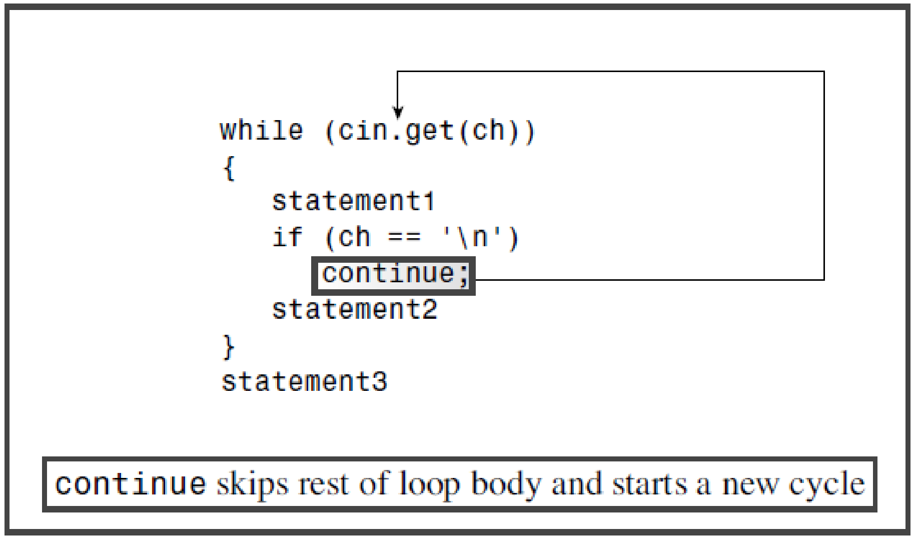
Figure 5.4 The structure of `do while` loops.

Do-while loop

Listing 5.15 **dowhile.cpp**

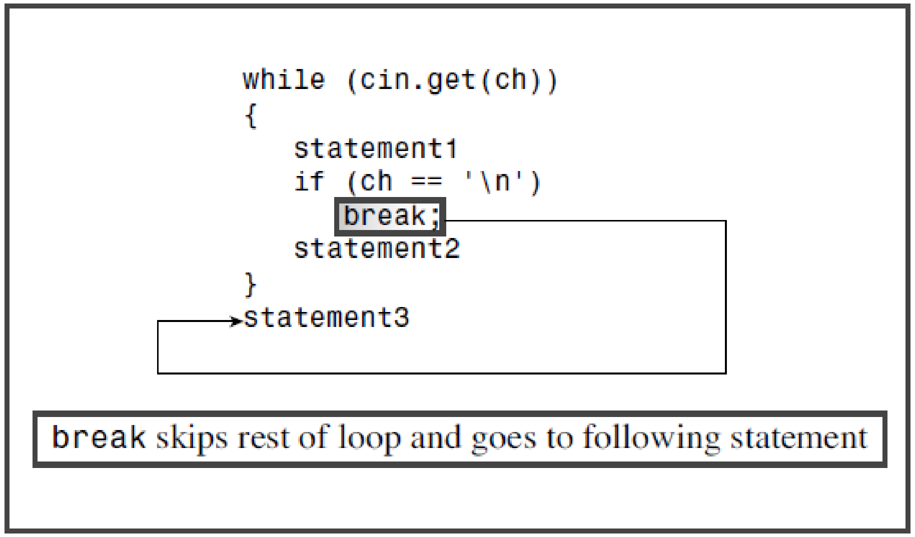
```
// dowhile.cpp -- exit-condition loop
#include <iostream>
int main()
{
    using namespace std;
    int n;

    cout << "Enter numbers in the range 1-10 to find ";
    cout << "my favorite number\n";
    do
    {
        cin >> n;       // execute body
    } while (n != 7);   // then test
    cout << "Yes, 7 is my favorite.\n" ;
    return 0;
}
```

Continue & Break



```
while (cin.get(ch))
{
   statement1
   if (ch == '\n')
      continue;
   statement2
}
statement3
```

continue skips rest of loop body and starts a new cycle



```
while (cin.get(ch))
{
   statement1
   if (ch == '\n')
      break;
   statement2
}
statement3
```

break skips rest of loop and goes to following statement

Figure 6.4 The structure of continue and break statements.

Q & A