**Chapter 4 Inserts**

**Insert 161-A**

1 // This program demonstrates how floating-point   
 2 // round-off errors can make equality operations unreliable.  
 3 #include <iostream>  
 4 using namespace std;  
 5   
 6 int main()  
 7 {  
 8 double a = 1.5; // a is 1.5.  
 9 double b = 1.5; // b is 1.5.  
10   
11 a += 0.0000000000000001; // Add a little to a.  
12 if (a == b)  
13 cout << "Both a and b are the same.\n";  
14   
15 if (a != b)  
16 cout << "a and b are not the same.\n";  
17   
18 return 0;  
19 }

**Insert 183-A**

**4.7 The if Statement with Initialization**

*Concept: Beginning with C++17, an if statement can optionally have an initialization clause.*

If you are using C++17 or later, you can write an if statement with an optional initialization clause that is executed before the conditional expression is evaluated. Here is the general format:

if (*initialization*; *expression*)

{

*statement;*

*statement;*

// Place as many statements here as necessary.

}

Here is the general format of an if/else statement with an initialization clause:

if (*initialization*; *expression*)

{

*statement;*

*statement;*

// Place as many statements here as necessary.

}

else

{

*statement;*

*statement;*

// Place as many statements here as necessary.

}

Let's look at an example. In the following code, assume that password is a string object:

if (int len = password.length(); len < MIN\_LENGTH)

{

cout << "Your password is too short." << endl;

}

In this if statement the initialization clause is:

int len = password.length();

The initialization clause defines an int variable named len and assigns the length of the password object to it. Then, the conditional expression len < MIN\_LENGTH is evaluated. If the expression is true, the cout statement is executed.

When a variable is defined in the initialization clause of an if statement, the variable's scope is limited to the if statement and its else block, if it has one. For example, the following code causes an error:

if (int len = password.length(); len < MIN\_LENGTH)

{

cout << "Your password is too short." << endl;

}

// The following statement causes an error.

cout << "Your password is " << len << " characters long." << endl;

The last statement will not compile because it tries to use the len variable, which can be accessed only by code inside the if statement.

Here is an example of an if/else statement that uses an initialization clause:

if (int len = password.length(); len < MIN\_LENGTH)

{

cout << "Your password is too short." << endl;

}

else

{

cout << "Your password has " << len << " characters." << endl;

}

In an if/else if statement, you can have an initialization clause in any of the if statements. Here is an example:

if (int ulen = username.length(); ulen < MIN\_LENGTH)

{

cout << "Your username is too short." << endl;

}

else if (int plen = password.length(); plen < MIN\_LENGTH)

{

cout << "Your password is too short." << endl;

}

else

{

cout << "Your username and password both "

<< "meet the minimum length." << endl;

}

When a variable is defined in the initialization clause of an if/else if statement, the variable's scope begins at the variable's definition, and extends to the end of the if/else if statement.

**Active Code Listing**

Look at the if statement in line 12 of the following program. Remove the comment that reads /\* Enter your code here \*/ and replace it with an initialization clause and a conditional expression. The initialization clause should be:

double a = round(x);

The conditional expression should be:

a == 100;

After you have made this change, do the following:

* Run the program and enter 99.7. The program should display *Perfect score!*
* Run the program and enter 99.2. The program should display *Keep trying!*
* Run the program and enter 100.4. The program should display *Perfect score!*
* Run the program and enter 100.9. The program should display *Keep trying!*

1 #include <iostream>  
 2 #include <cmath>  
 3 using namespace std;  
 4   
 5 int main()  
 6 {  
 7 double x;  
 8   
 9 cout << "Enter a number: ";  
10 cin >> x;  
11   
12 if (/\* Enter your code here \*/)  
13 cout << "Perfect score!" << endl;  
14 else  
15 cout << "Keep trying!" << endl;  
16   
17 return 0;  
18 }

**Checkpoint**

True or False: In an if statement, the initialization clause executes before the conditional expression is evaluated.

**Checkpoint**

If a variable is defined in an if statement's initialization clause, the variable can be accessed by code outside the if statement.

**Checkpoint**

True or False: If a variable is defined in an if/else statement's initialization clause, you can access the variable in the else block.

**Insert 213-A**

**4.16 The switch Statement with Initialization**

*Concept: Beginning with C++17, a switch statement can optionally have an initialization clause.*

If you are using C++17 or later, you can write a switch statement with an optional initialization clause that is executed before the conditional integer expression is evaluated. Here is the general format:

switch (*Initialization*; *IntegerExpression*)

{

case *ConstantExpression*:

// place one or more statements here

case *ConstantExpression*:

// place one or more statements here

// case statements may be repeated as many

// times as necessary

default:

// place one or more statements here

}

Let's look at an example. In the following code, assume that number is a double:

switch (int value = abs(number); value)

{

case 1:

cout << "one" << endl;

break;

case 2:

cout << "two" << endl;

break;

default:

cout << "Invalid value" << endl;

break;

}

In this switch statement the initialization clause is:

int value = abs(number);

The initialization clause defines an int variable named value and assigns the absolute value of number to it. Then, the value variable is evaluated.

When a variable is defined in the initialization clause of a switch statement, the variable's scope is limited to the switch statement. For example, the following code causes an error:

switch (int value = abs(number); value)

{

case 1:

cout << "one" << endl;

break;

case 2:

cout << "two" << endl;

break;

default:

cout << "Invalid value" << endl;

break;

}

// The following statement causes an error.

cout << value << endl;

The last statement will not compile because it tries to use the value variable, which can be accessed only by code inside the switch statement.

**Checkpoint**

In a switch statement, when does the initialization clause execute?

a. Before the conditional integer expression is evaluated

b. After the conditional integer expression is evaluated

**Checkpoint**

True or False: If a variable is defined in a switch statement's initialization clause, the variable may be accessed by code outside the switch statement.

**Checkpoint**

True or False: Beginning with C++17, an initialization clause is required for every if statement and switch statement.