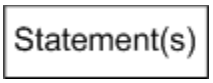
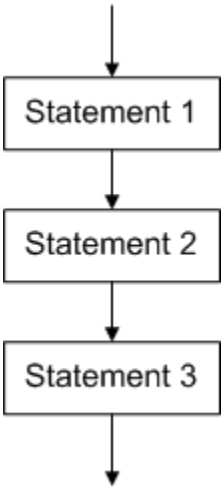
For the purposes of this course, we are only going to use the following flowcharting symbols:

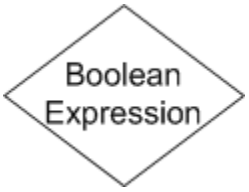
Rectangle



This symbol is used to enclose one or more program statements. The key feature of this symbol is that it will have exactly one arrow leading in and one arrow leading out. As such, multiple rectangles can be combined to indicate linear processing. For example, a program with three steps performed in sequential order might be shown like this:

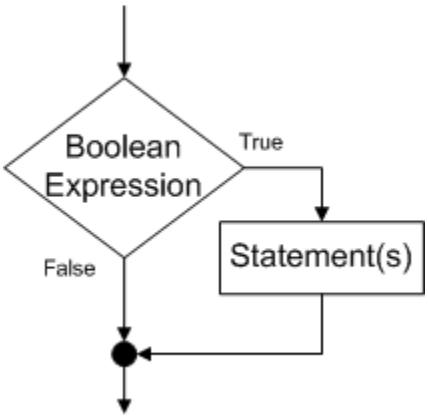


Diamond



This symbol is used to indicate a decision, a choice between one of two possible paths. Accordingly, it will have one arrow entering but two arrows leaving. The decision is based on the Boolean expression, and the arrows leaving a diamond should be labeled "True" & "False" to indicate the path taken by each Boolean result.

An "if" Statement can be represented by the following flowchart:



In this example, the Boolean expression within the diamond is evaluated. If the result is true, the path is taken which leads to the statements within the rectangle. In the false case, the other path is taken.

Notice the solid dot symbol where the two paths merge. The book does not use this symbol, but I think it's important to clearly show where program paths come back together. This symbol essentially serves as the inverse of a diamond. It should have exactly two arrows leading in and one leading out. The previous example can be implemented using the following C++ code:

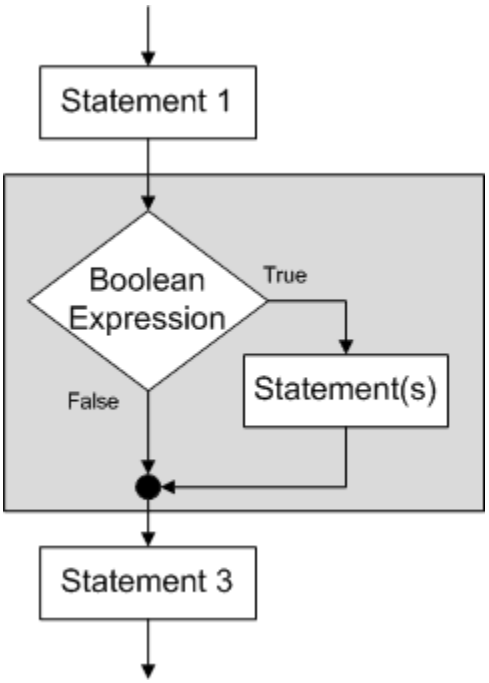
```
if (Boolean Expression)
{
     Statement(s)

}
```

The first line (if …) implements the decision (diamond symbol). The body of the statement, (i.e. the stuff to do if the expression is true) is indented. When the indenting ends, the if statement is over. Anything following (not indented) is executed regardless of how the if statement went.

Combining If Statements with Other Statements

A critical concept in this material is to realize that an If statement is indeed a statement, and can be substituted for "statement" in any of these diagrams. Visualize this as enclosing the decision structure within a box. So long as the box has one arrow entering and one arrow leaving, it is interchangeable with a statement box. Consider the following example:



This example is similar to the previous three-step sequential example, but "Statement 2", which is the grey box, has been implemented as an If statement. Although the grey box encloses conditional logic, the box itself has one arrow entering and one arrow leaving.

The program flow in this example is:

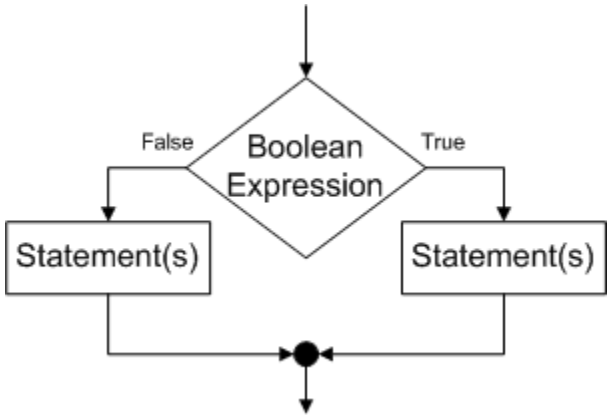
1. Execute Statement 1
2. Evaluate the Boolean Expression
3. If true, Execute the Indicated Statement(s)
4. Execute Statement 3

Statements 1 will always be executed in this example, as it is encountered prior to any conditional logic. The subsequent conditional logic is complete (the divergent paths have converged) prior to statement 3, so it is also always executed. In other words, Statements 1 and 3 are not part of the If statement. The grey box represents the if statement. This flowchart can be represented with the following C++ code:

```
Statement 1
if (Boolean_Expression)
{
     Statement(s)
}
Statement 3
```

If-Else Statements

A C++ "If-Else" Statement can be represented by the following flowchart:



This construct differs from the previously-discussed If statement in that there is an additional statement block on the false path. Thus with this statement, we are choosing between two possible options. The statement(s) for the true option should appear between the colon and the else, indented. The statement(s) for the false option should follow the else:, also indented: The C++ code:

```
if (Boolean_Expression)
{
     Statement(s)
}
else
{
     Statement(s)
}
```

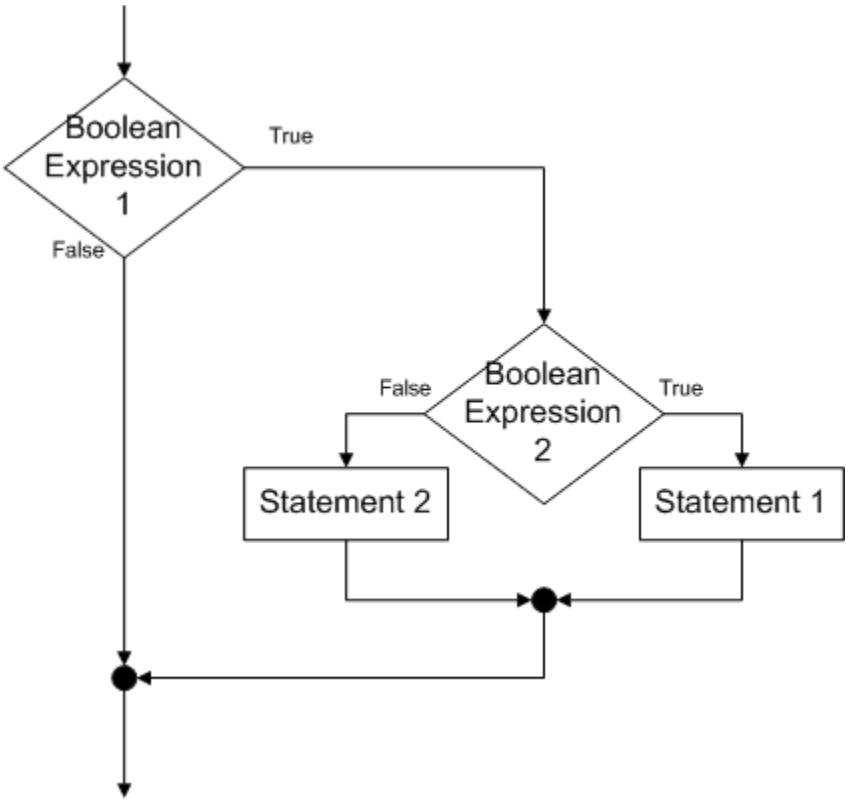
Simple Rules For Using Flowcharts

Given a valid flowchart, it should be a simple and mechanical process to convert that flowchart into C++ code. A flowchart is valid if it complies with these requirements:

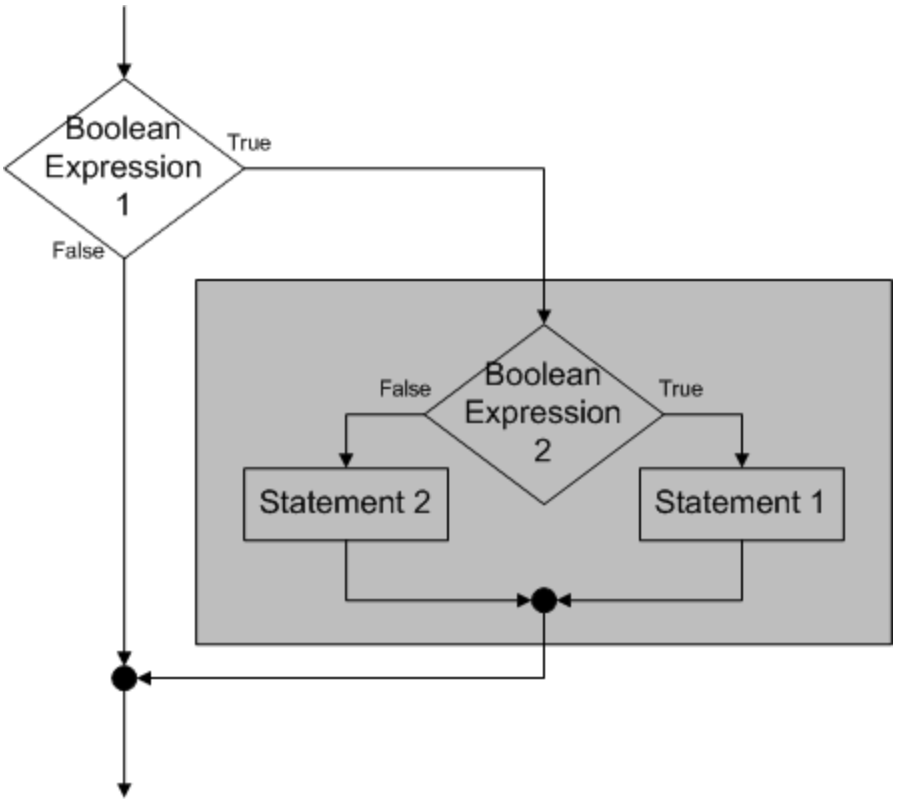
- All statement blocks (rectangles) have one way in and one way out
- Decision blocks (diamonds) have one way in and two ways out
- The dot symbols (End If's) have two ways in and one way out
- Every diamond has a matching dot
- All decision structures can be enclosed in a block that has one way in & one way out
- Don't cross any lines

Nesting If Statements

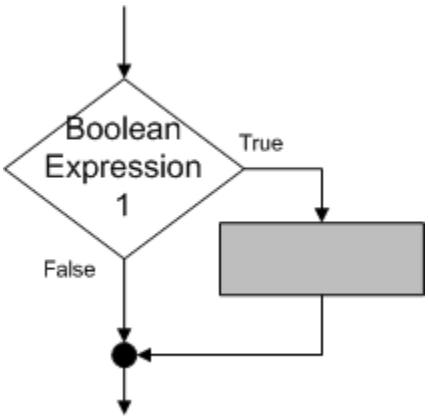
So long as the above rules are followed, If statements can be nested within other if statements. Consider the following example:



The key to understanding compound if statements is to recognize which dots go with which diamonds. You should be able to enclose individual if statements within rectangles to reduce the flowchart to layers, with each layer being a simple statement. For example, if we recognize that the diamond containing Boolean expression 2 matches with the dot directly below it, we can mentally enclose that if statement within a rectangle, as shown below:



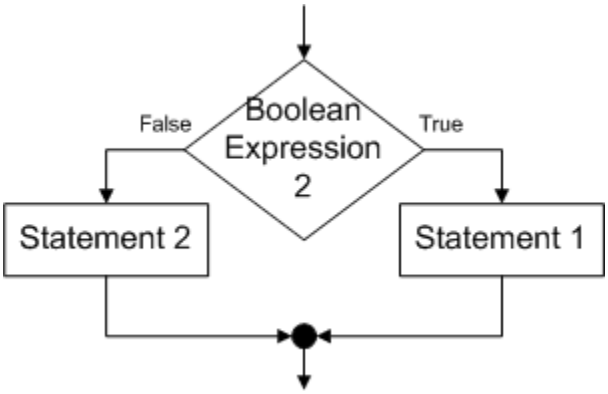
Since the grey box conforms to our rectangle rules, i.e. it has one arrow leading in and one arrow leading out, we can treat this box as a statement. By ignoring the specific contents of this statement, we are left with:



We should recognize this as a simple if statement, where the grey box represents the statement. Therefore, the C++ code for this is:

```
if (Boolean_Expression_1)
{
     //Grey Box Code Goes Here
}
```

Now we can implement the grey box as a smaller problem, knowing that all its code will go where we placed the #grey box code goes here comment. The contents of the grey box appear as follows:



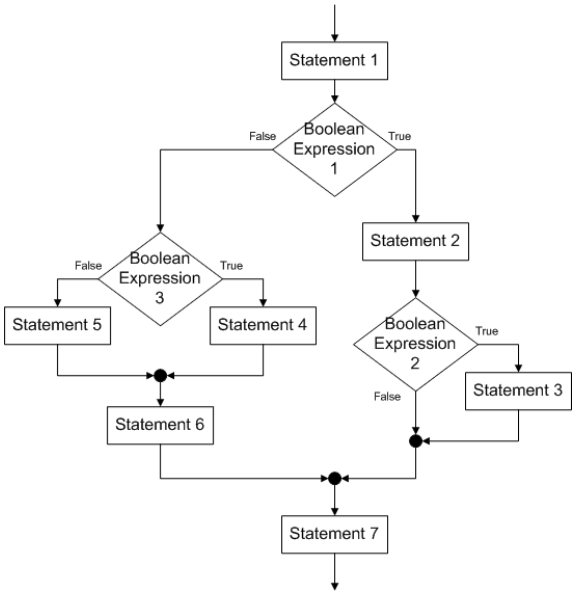
We should recognize this as an If-Else statement, and know that the C++ code to implement is:

```
if (Boolean_Expression_2)
{
     Statement 1
}
else
{
     Statement 2
}
```

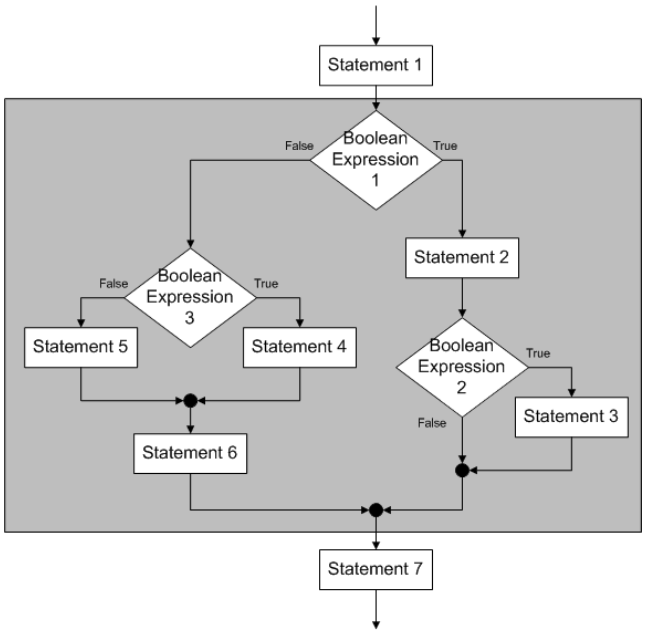
Placing this code in the proper location for the grey box, we wind up with the final code of:

```
if (Boolean_Expression_1)
{
     if (Boolean_Expression_2)
     {
          Statement 1
     }
     else
     {
          Statement 2
     }
}
```

A More Complex Example



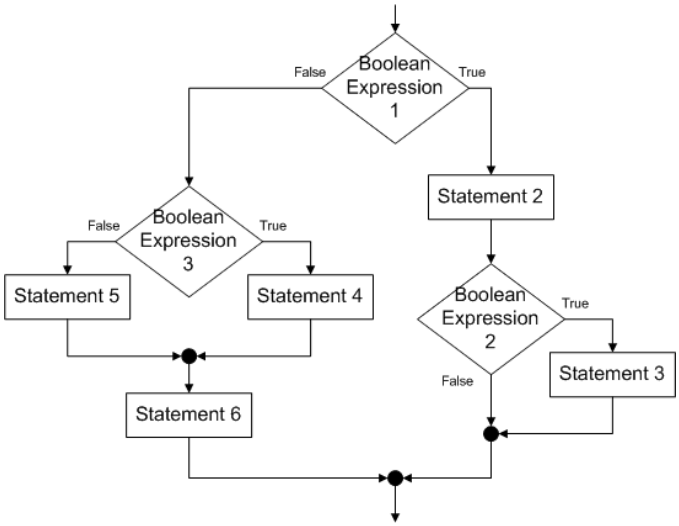
As before, the key is to recognize the layers. If this example, we identify the first decision symbol (containing Boolean Expression 1) and locate the dot symbol which completes it, in this case the one directly above statement 7.



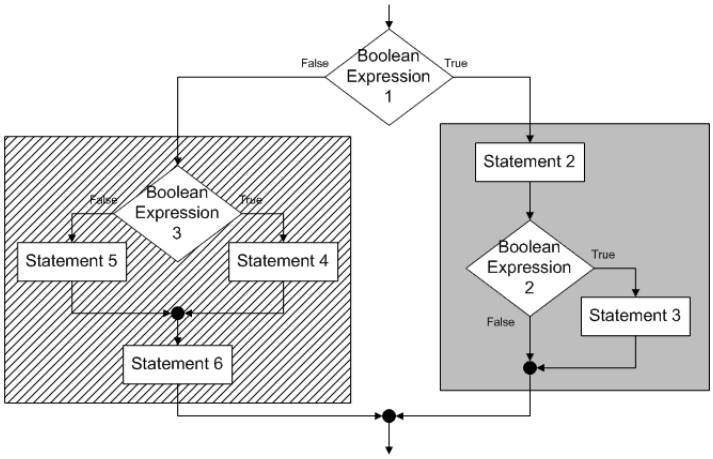
Enclosing this in a grey box illustrates that this box indeed has one arrow in and one arrow out. We can now see that at the outermost layer, the program consists of Statement 1, followed by the grey box code, followed by Statement 7:

```
Statement 1
Grey Box
Statement 7
```

Now we begin work on the grey box, knowing that all the code we create in this step will go between statements 1 & 7. The portion of the flowchart in the grey box follows:



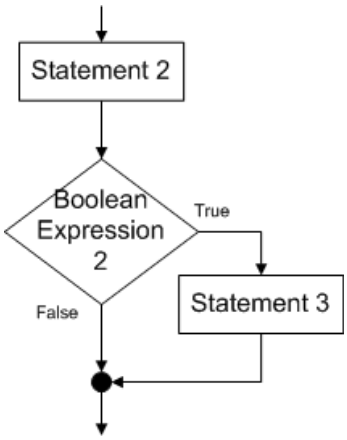
As before, we now need to recognize the statements. If we place boxes around the following sections:



We can see that this flowchart is essentially an if-else statement. If we add the code for a C++ if-else statement to the code we had from the previous step, we get:

```
Statement 1
if (Boolean_Expression_1)
{
     //Grey Box
}
else
{
     //Striped Box
}
Statement 7
```

Now we continue this process for the two boxes. Taking the grey box first, the flowchart within the grey box is:



Hopefully it is clear that this is Statement 2 followed by a simple If statement. The C++ code for this would be:

```
Statement 2
if (Boolean_Expression_2)
{
     Statement 3
}
```

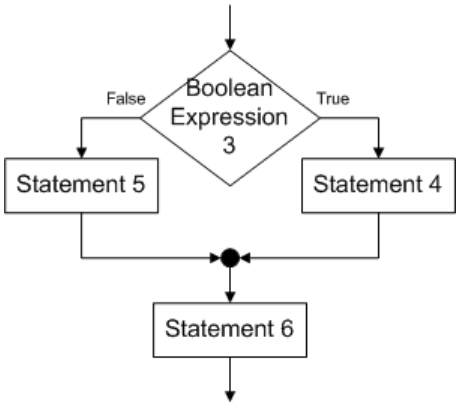
Inserting this code in the grey box location of the previous code, we get:

```
Statement 1

if (Boolean_Expression_1)
{
     Statement 2
     if (Boolean_Expression_2)
     {
          Statement 3
     }
}
else
{
     //Striped Box
}

Statement 7
```

Now we proceed with the striped box, which enclosed the following flowchart:



Hopefully you recognize this as an if-else statement, followed by statement 6. The C++ code for this would be:

```
if (Boolean_Expression_3)
{
     Statement 4
}
else
{
     Statement 5
}
Statement 6
```

Adding this to our previous code in the proper location for "striped box" gives us the final code of:

```
Statement 1

if (Boolean_Expression_1)
{
     Statement 2
     if (Boolean_Expression_2)
     {
          Statement 3
     }
}
else
{
     if (Boolean_Expression_3)
     {
          Statement 4
     }
     else
     {
          Statement 5
     }
     Statement 6
}

Statement 7
```