Appendix-E: *Basic Statement Keywords*

The following are the statement keywords in C++. Only the highlighted will be describe in detail.

break	**default**	**for**	**switch**
case	**do**	**goto**	
	else	**if**	
continue		**return**	**while**

break

Syntax
break ;

Description
Use the break statement within loops to pass control to the first statement following the innermost enclosing brace.

case

Syntax
switch (<switch variable>){
  case <constant expression> : <statement>; [break;]
    .
    .
    .
  default : <statement>;
}

Description
Use the case statement in conjunction with switches to determine which statements evalute.
The list of possible branch points within <statement> is determined by preceding substatements with

case <constant expression> : <statement>;

where <constant expression> must be an int and must be unique.
The <constant expression> values are searched for a match for the <switch variable>.
If a match is found, execution continues after the matching case statement until a break statement is encountered or the end of the switch statement is reached.
If no match is found, control is passed to the default case.

Note: It is illegal to have duplicate case constants in the same switch statement.

continue

Syntax
continue ;

Description
Use the continue statement within loops to pass control to the end of the innermost enclosing brace; at which point the loop continuation condition is re-evaluated.

default

Syntax
switch (<switch variable>){
  case <constant expression> : <statement>; [break;]
    .
    .
    .
  default : <statement>;
}

Description
Use the default statement in switch statement blocks.

If a case match is not found and the default statement is found within the switch statement, the execution continues at this point.

If no default is defined in the switch statement, control passes to the next statement that follows the switch statement block.

do

Syntax
do <statement> while (<condition>);

Description
The do statement implements a do ... while loop.
<statement> is executed repeatedly as long as the value of <condition> remains non-zero.
Since the conditon is tests after each the loop executes the <statement>, the loop will execute at least once.

else

Syntax
if (<condition>) <statement1>;

if (<condition>) <statement1>;
  else <statement2>;

Description
Use if to implement a conditional statement.
When <condition> evaluates to be nonzero, <statement1> executes.
If <condition> is false (0), <statement2> executes.
Else is optional, but no statements can come between an if statement and an else.
The #if and #else preprocessor statements (directives) look similar to the if and else statements, but have very different effects. They control which source file lines are compiled and which are ignored.

for

Syntax
for ([<initialization>] ; [<condition>] ; [<increment>]) <statement>

Description
The for statement implements a for loop.
<statement> is executed repeatedly UNTIL the value of <condition> is FALSE.

Before the first iteration of the loop, <initialization> initializes variables for the loop.
After each iteration of the loop, <increments> increments a loop counter. (Consequently, j++ is functionally the same as ++j.)

In C++, <initialization> can be an expression or a declaration.
The scope of any identifier declared within the for loop extends to the end of the control statement only.
All the expressions are optional. If <condition> is left out, it is assumed to be always true.

goto

Syntax
goto <identifier> ;

Description
Use the goto statement to transfer control to the location of a local label specified by <identifier>.
Labels are always terminated by a colon.

if

Syntax
if (<condition>) <statement1>;

if (<condition>) <statement1>;
  else <statement2>;

Description
Use if to implement a conditional statement.
When <condition> evaluates to be nonzero, <statement1> executes.
If <condition> is false (0), <statement2> executes.
Else is optional, but no statements can come between an if statement and an else.
The #if and #else preprocessor statements (directives) look similar to the if and else statements, but have very different effects. They control which source file lines are compiled and which are ignored.

return

Syntax
return [<expression>] ;

Description
Use the return statement to exit from the current function back to the calling routine, optionally returning a value.

switch

Syntax
switch (<switch variable>) {
  case <constant expression> : <statement>; [break;]
    .
    .
    .
  default : <statement>;
}

Description
Use the switch statement to pass control to a case which matches the <switch variable>. At which point the statements following the matching case evaluate.
If no case satisfies the condition the default case evaluates.
To avoid evaluating any other cases and reliquish control from the switch, terminate each case with break;.

while

Syntax
while (<condition>) <statement>

Description
Use the while keyword to implement a while loop.
<statement> executes repeatedly until the value of <condition> is zero.
The test takes place before <statement> executes. Thus, if <condition> evaluates to zero on the first pass, the loop does not execute.