C Control Structures

Sequence

Compound Statement - {  }

C Control Structures

Decision

If-else

switch-case
C Control Structures

Loop

while  do-while  for

Loops

Directions
1. Wet hair
2. Apply shampoo
3. Lather
4. Rinse
5. Repeat
C Looping Statements

The while statement

\texttt{while(exp) statement}

Operation:
\texttt{The expression is evaluated. If it is "TRUE," the statement is executed, then the expression is evaluated again. If the expression is "FALSE" the statement is NOT executed, and control is passed to the statement following the while.}

while Examples

\texttt{while(ch \neq \textquoteleft \textquoteleft \textquoteleft \textquoteleft ) \texttt{cin} \texttt{\textless\textless} \texttt{ch};}

\texttt{k = 0;}
\texttt{while(k < 10) \texttt{k} = \texttt{k} + 1;}

\texttt{k = 0;}
\texttt{while(k < 10)}
\{
\texttt{cin \textgreater\textgreater} \texttt{ch;}
\texttt{k++;}
\texttt{cout \textless\textless} \texttt{ch;}
\}
\texttt{// END while}
\texttt{cout \textless\textless \textquotemark\{"You input \} \textless\textless k \textless\textless \textquotemark\"characters\textbackslash n\";}
More while Examples

i = 0;
sum = 0.0;
while(i < 10)
{
    cin >> val;
    sum += val;  // sum = sum + val;
    i++;
}  // END while
avg = sum / float(i);

More while Examples

i = 0;
sum = 0.0;
cin >> val;
while(val != 0.0)
{
    sum += val;
    i++;
    cin >> val;
}  // END while
avg = sum / float(i);
C Looping Structures

The do-while statement

\[ \text{do statement while (exp);} \]

Operation:
The statement is executed. Then the expression is evaluated. If the expression is "TRUE," the statement is executed again. If the statement is "FALSE," the statement following the do-while is executed, thereby terminating the loop.

do-while Example

\[ i = 0; \]
\[ \text{sum} = 0.0; \]
\[ \text{do} \]
\[ \{ \]
\[ \text{cin >> val;} \]
\[ \text{sum += val;} \]
\[ i++; \]
\[ \}\text{while (val != 0.0);} \]
\[ \text{avg} = \text{sum} / \text{float} (i-1); \]
Anatomy of a Loop

\begin{itemize}
  \item \textbf{Pre-loop initialization}\n    \begin{itemize}
      \item \texttt{i = 0;}
      \item \texttt{sum = 0.0;}
    \end{itemize}
  \item \textbf{Loop condition}\n    \begin{itemize}
      \item \texttt{while (i < 100)}
    \end{itemize}
  \item \textbf{Loop body}\n    \begin{itemize}
      \item \texttt{// do something}
      \item \texttt{sum += \texttt{val};}
    \end{itemize}
  \item \textbf{Loop "housekeeping"}\n    \begin{itemize}
      \item \texttt{i++;}
      \item \texttt{// END while}
    \end{itemize}
  \item \textbf{Post loop}\n    \begin{itemize}
      \item \texttt{avg = sum / float(i);}\n    \end{itemize}
\end{itemize}

The C for Loop

"The swiss army knife of looping statements"

\texttt{for( expr1; expr2; expr3 ) statement}

\textit{Operation:}

First, \texttt{expr1} is performed (it is the pre-loop initialization).
Then \texttt{expr2} is evaluated; if TRUE, then statement is performed, then \texttt{expr3} is executed and \texttt{expr2} is evaluated again. If \texttt{expr2} is FALSE, control goes to the statement following the \texttt{for} statement.
Comparing while and for

for (expr1; expr2; expr3) statement

is exactly identical to the following while statement:

expr1;
while (expr2)
{
    statement
    expr3;
} //END while

Loop Anatomy with for

for (expr1; expr2; expr3) statement

Pre-loop initialization

Loop condition

"Housekeeping"

Loop body
Some for Examples

```c++
sum = 0.0;
for ( i = 0; i < 10; i++ )
{
    cin >> val;
    sum += val;  // sum = sum + val;
}  // END for
avg = sum / float(i);

sum = 0.0;
cin >> val;
for(i=0; val != 0.0; i++)
{
    cin >> val;
    sum += val;
}  // END for
avg = sum / float(i);
```

More for Examples

```c++
cin >> val;
for(i=0,sum=0.0; val != 0.0; i++,sum+=val)
{
    cin >> val;
}  // END for
avg = sum / float(i);

i=0; sum=0.0;
cin >> val;
for( ; val != 0.0; )
{
    cin >> val;
    i++; sum +=val;
}  // END for
avg = sum / float(i);
```

```c++
for(i=0,sum=0.0,cin>>val; val != 0.0; i++,sum+=val,cin>>val);
avg = sum / float(i);
```
Temperature Conversion

```c++
float fahr, cent, highF, lowF;
cout << "Enter starting and ending value: ";
cin >> lowC >> highC;

cout << "Celsius to Fahrenheit Table\n";
cout << "Celsius    Fahrenheit\n";

for( cent = lowC; cent < highC; cent += 10) 
{
    fahr = cent * 9./5. + 32.;
    cout << cent << "   " << fahr << endl;
} // END for
```

Nested Loops

```c++
for(i = 0; i < 3; i++)
{
    for(j = 0; j < 4; j++)
    {
        cout << i << j << endl;
    } // END for j
} // End for i
```
Multiple Nesting

for(i = 0; i < 3; i++)
{
    for(j = 0; j < 4; j++)
    {
        if(j < 2)
        {
            k = i + j;
            cout << k;
        } // END if
    } // END for j
} // END for i

Auxiliary Control Statements

break;

Causes an exit from the current program block. Execution continues at the statement following the block.

continue;

Causes the rest of the program block to be ignored, but does not exit the block. In loops, this means that the current iteration is terminated and the next one starts.

goto label;

Causes control to transfer to the statement with the specified label.

EX.

    if (error) goto errlabel:
    :
    errlabel: cout << "An error occurred!\n";