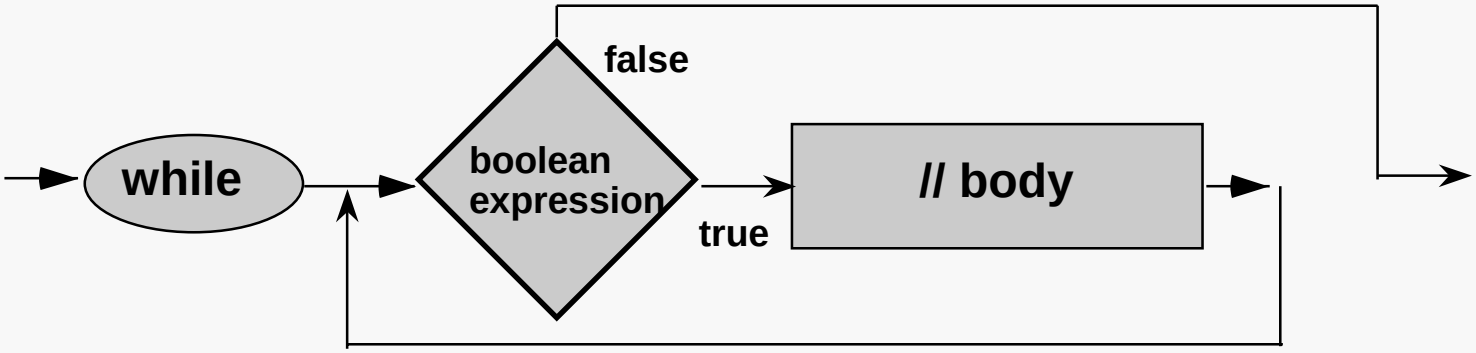
While Loop

Loop - a control structure that causes a set of statements to be executed repeatedly, (reiterated).

While statement - most versatile type of loop in C++



The statement can be a compound statement, called the body of the loop.

As long as the Boolean-expression is true the statement is executed repeatedly.

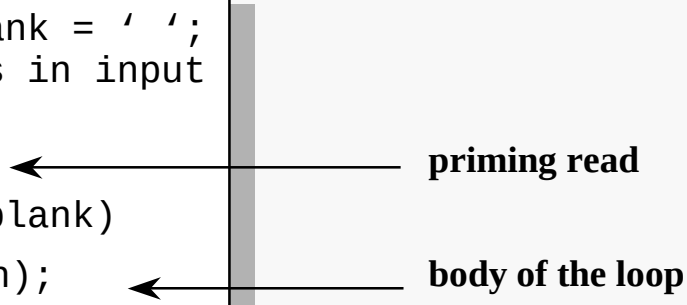
loop entry	- the point where flow of control passes to the body
iteration	- one pass or repetition through the loop
loop test	- the point at which the Boolean expression is evaluated and the decision is made to (re)enter the body
loop exit	- the point at which the repetition ends and control passes to the next statement after the loop
termination condition	- the condition that causes the looping to stop

©1995-1999 Barnette ND, McQuain WD, Keenan MA

While Loop (cont)

Example:

```
const char blank = ' ';
// Skip blanks in input

cin.get(ch);
while (ch == blank)
    cin.get(ch);
```



- Note that the statement(s) in the body of the loop will never be executed if the loop condition is initially false.
- There must be at least one statement in the body of the loop that affects the value of the boolean expression, otherwise if the condition is true initially then the loop will never be exited (an “infinite” loop).

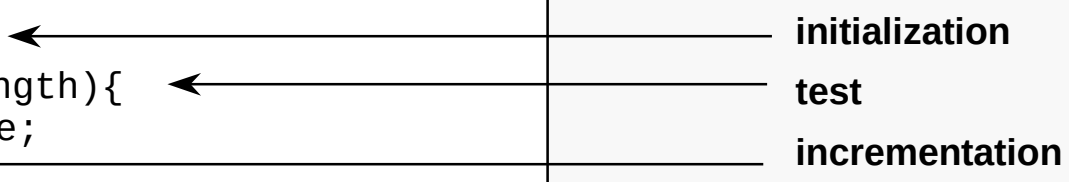
©1995-1999 Barnette ND, McQuain WD, Keenan MA

Count Controlled Loop

Count Controlled Loop - a loop that executes a specified number of times

lcv (loop control variable) - an integer variable in the Boolean expression
- initialized before the loop,
- incremented (or decremented) inside the loop

```
// output a line of delimiting chars

const char toWrite  = '*' ;
const int Length = 80 ;

int Column = 0 ;
while (Column < Length){
    cout << toWrite;
    Column++;
} // end while
```



©1995-1999 Barnette ND, McQuain WD, Keenan MA

Count Controlled Loop (continued)

Example:

```
const int NumGrades = 25;
int SumGrades = 0;
int NumRead   = 0;                    // initialize
int Grade;

while (NumRead < NumGrades )  {       // test
   cin >> Grade ;                     // process the
   SumGrades = SumGrades + Grade;     // grades
   NumRead   = NumRead + 1;           // increment lcv
}
cout << (SumGrades/NumGrades) << endl;
```

How are SumGrades and NumRead used?

_____ and _____

What is the termination condition?

What is the value of NumRead after the loop has terminated?

if the "increment lcv" step is omitted?

For efficiency considerations, count-controlled loops should be implemented using the for statement (see following slides).

©1995-1999 Barnette ND, McQuain WD, Keenan MA

Event Controlled Loop

Event-Controlled Loop: loop that executes until a specified situation arises to signal the end of the repetition

Sentinel loop: loop that terminates when a dummy data value is input, used to signal the end of a list of data, the sentinel is not part of the raw data

```
int Month;
bool newMonth = true;

while (newMonth) {
   cin >> Month ;
   newMonth = (Month > 0 && Month < 13);
   if (newMonth)
      cout << "It's a month" << endl;
}
```

Note that the loop control variable could be replaced by the condition used to set it, (assuming a priming value other than zero is assigned to month), but the resulting code would perhaps not be as readable.

The loop above is a combination of a sentinel loop & a flag (event), controlled loop.

©1995-1999 Barnette ND, McQuain WD, Keenan MA

Event Controlled Loop (cont)

Example:

```
int NumPeople   = 0;
int SumHeights  = 0;
int Height;
    cin >> Height ;
    while (Height != 0 ) {
      NumPeople++;
      SumHeights = SumHeights + Height;
      cin >> Height  ;
}
float AvgHeight = SumHeights / NumPeople;
```

DATA FILE:

```
61   57   67   70
57   56   59
77    0   72   68
71   69   60    0
```

While could be coded:

```
   while (Height) {
```

The first cin is a priming read. Note the positioning of the other read.
The last height read (sentinel 0) will not be processed like the other data items.

Poor Usage:

```
bool Done = false;
while (!Done) {
   cin >> Height;
   cout << Height;
   if (Height == 0)
      Done = true;
}
```

Without a priming read the EOF should be checked initially. An empty data file would result in a runtime "attempt to access beyond end of file" error.

©1995-1999 Barnette ND, McQuain WD, Keenan MA

EOF Controlled Loop

Example

```
0       int NumBreaks = 0;
1       int HiTemp    = 0, Temperature;
2       inData >> Temperature;
3       while (inData)  {

4           if (Temperature > HiTemp) {
5              HiTemp    = Temperature;
6              NumBreaks = NumBreaks + 1;
            }
7           inData >> Temperature;
8       }
9       cout >> HiTemp;
```

DATA

```
32¶
45¶
78¶
62¶
78¶§
```

¶	represents the return char
§	represents the end of file char

How is hitemp used? (note that it was initialized)

What about numbreaks?

Trace the program segment:

NumBreaks	eof	Temperature	if-condition	HiTemp	after statement

©1995-1999 Barnette ND, McQuain WD, Keenan MA

End-of-line Controlled Loop

Example:

```
// Count the number of chars and
// the number of lines in the input
char achar       =   ' ';
int numchars =   0;
int numlines =   0;
cin.get(achar) ;     // priming read
while (!cin.eof()) {
    // nested loop
    while (achar != '\n') {
       numchars++;
       cout << achar ;
       cin.get(achar) ;
    }
    numlines++;
    cout << achar << endl ;
    cin.get(achar);
}
```

DATA

```
Now is the¶
time for¶
all good men¶
to come to the¶
aid of their party !¶§
```

¶	represents the return char
§	represents the end of file char

©1995-1999 Barnette ND, McQuain WD, Keenan MA

Sentinel-Controlled Loop

Flag (sentinel) controlled loops require dummy data values that signify the end of the actual data to be imbedded in the input file.

Example

```
const int DUMMYHOURS = -1 ;
          •   •   •
cin >> HoursWorked ;
bool MoreData = (HoursWorked != DUMMYHOURS );
while (MoreData ) {

  // process the data

  cin >> HoursWorked ;
  MoreData = (HoursWorked != DUMMYHOURS ) ;

 }
```

Requires a priming read, as other event loops, to initialize the flag (moredata).

The flag variable must be updated as soon as the condition changes.

Note how the boolean variable, moredata, and the constant, DUMMYHOURS, are used to self-document the condition and code.

©1995-1999 Barnette ND, McQuain WD, Keenan MA

Loop Design Considerations

Questions that one should consider carefully when coding a loop:

- What is the condition that ends the loop?
- How should the condition be initialized?
- How should the condition be updated?
- What is the process to be repeated?
- How should the process be initialized?
- How should the process be updated?
- What is the state of the program on exiting the loop?

©1995-1999 Barnette ND, McQuain WD, Keenan MA

For Loop

For loops are used whenever the number of times a loops needs to execute is known or can be calculated beforehand.

```
for (initial expression; test expression; update expression)
    <statement>;
```

The initial expression is performed just once, prior to loop execution. The initial expression may declare variables as well as specify initializations for them.

The test expression is evaluated and if false then the next statement after the for loop is executed.

If the test expression evaluates to true, then the <statement> of the for loop is executed, the update expression is performed, and test expression is evaluated again.

©1995-1999 Barnette ND, McQuain WD, Keenan MA

For Loop

```
char toWrite  = '*';
const int Length = 80;
int Column;
for (Column = 0; Column < Length; Column++) {
   cout << toWrite;
}
```

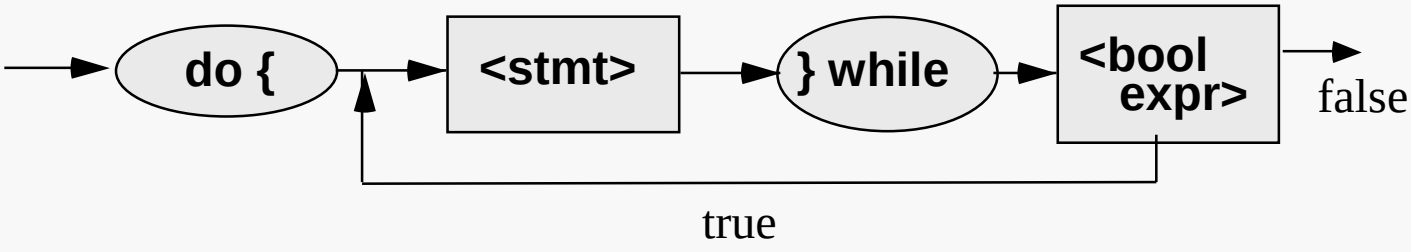
```
int sum;
for (int scan = 1, sum = 0; scan <= 10; scan++)
    sum  =  sum  +  scan  *  scan;

cout << "Sum of squares = " << sum;
```

```
int sweep, high = 34, total = 0;
for (sweep = -40; sweep < high + 7; sweep = sweep + 10)
{
    cout << "in the loop again" << endl;
    total  =  total + sweep / 12;
}
```


©1995-1999 Barnette ND, McQuain WD, Keenan MA

Do-While Loop

Syntax:



Example: skip blanks on input line

```
do{
  cin.get(ch);
}while (ch == ' ' );
```

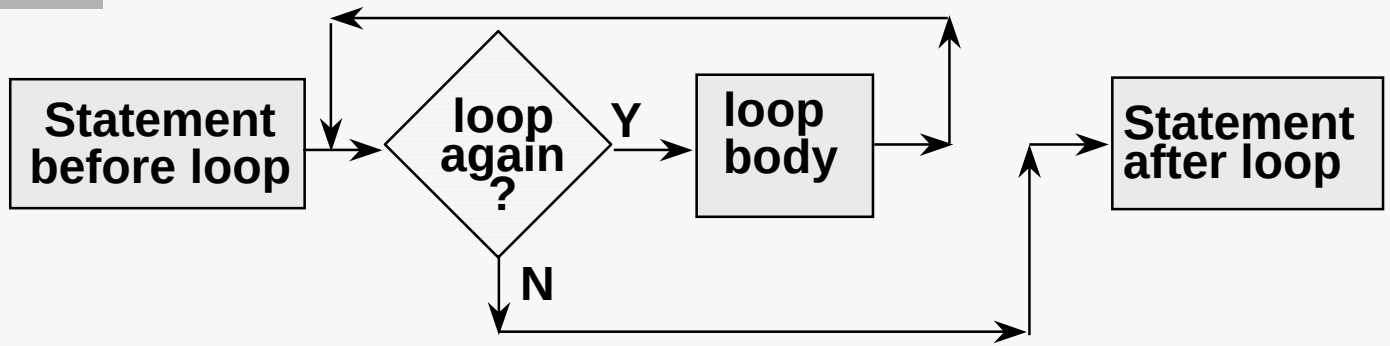
loop body

The loop body is always executed at least one time.

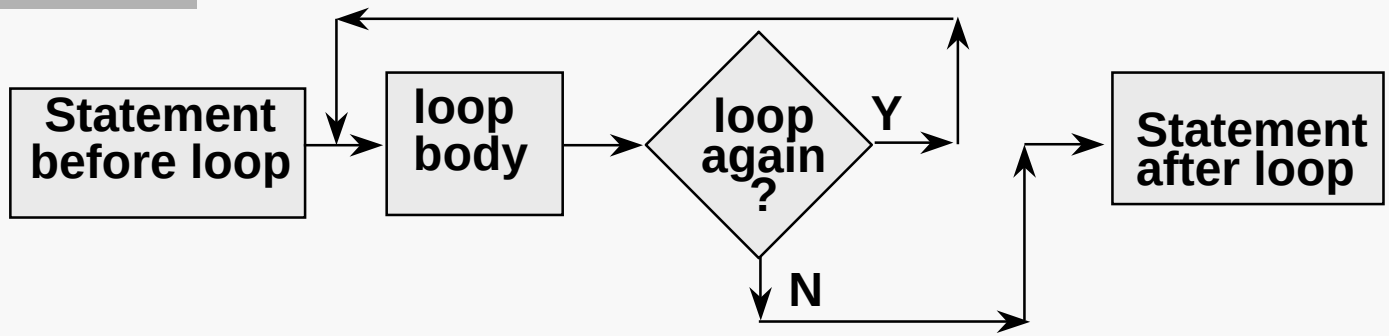
©1995-1999 Barnette ND, McQuain WD, Keenan MA

While vs Do-While

while



do-while



©1995-1999 Barnette ND, McQuain WD, Keenan MA

Break Statement

We have seen the break statement used with the switch statement. A break statement causes an immediate exit from the ***innermost*** switch, while, do-while, or for statement in which it appears. If a break statement is in a loop nested inside another loop, control exits the inner loop but not the outer loop.

```
#include <iostream.h>

void main( ) {
   int valEntered;
   int SumSoFar = 0;

   while (true) {
      cout << "Enter a positive integer (or 0 to quit): " << flush;
      cin  >> valEntered;
      if (valEntered == 0)
         break;
      SumSoFar = SumSoFar + valEntered;
      cout << "Sum so far is: " << setw(8) << SumSoFar << endl;
   }
}
```

However, see the next slide for an arguably superior alternative...

©1995-1999 Barnette ND, McQuain WD, Keenan MA

Break Statement

The preceding example would be improved if it were converted to use a while loop instead of the break, because the loop termination condition would be clearer:

```
#include <iostream.h>

void main( ) {
   int valEntered;
   int SumSoFar = 0;

   cout << "Enter a positive integer (or 0 to quit): " << flush;
   cin  >> valEntered;

   while (valEntered != 0) {
      SumSoFar = SumSoFar + valEntered;
      cout << "Sum so far is: " << setw(8) << SumSoFar << endl;

      cout << "Enter a positive integer (or 0 to quit): " << flush;
      cin  >> valEntered;
   }
}
```

```
While could be coded:

  while(valEntered ) {
```


©1995-1999 Barnette ND, McQuain WD, Keenan MA

Continue Statement

The continue statement is used only for loops and allows the programmer to terminate the current loop iteration but not the entire loop. It causes an immediate branch to the bottom of the loop, thereby skipping the rest of the statements in the body of the loop. Make sure you understand the distinction between the continue and break statements.

```
#include <iostream.h>

void main( ) {
   int valEntered = -1;
   int SumSoFar = 0;

   while (valEntered != 0) {
      cout << "Enter a positive integer (or 0 to quit): " << flush;
      cin  >> valEntered;
      if (valEntered <= 0)
         continue;
      SumSoFar = SumSoFar + valEntered;
      cout << "Sum so far is: " << setw(8) << SumSoFar << endl;
   }
}
```

However, see the next slide for an arguably superior alternative...

©1995-1999 Barnette ND, McQuain WD, Keenan MA

Continue Statement

The preceding example could be improved by using an if statement that made the last two statements in the while loop explicitly conditional on the value entered:

```
#include <iostream.h>

void main( ) {
   int valEntered = -1;
   int SumSoFar = 0;

   while (valEntered != 0) {
      cout << "Enter a positive integer (or 0 to quit): " << flush;
      cin  >> valEntered;
      if (valEntered > 0) {
         SumSoFar = SumSoFar + valEntered;
         cout << "Sum so far is: " << setw(8) << SumSoFar << endl;
      }
   }
}
```


©1995-1999 Barnette ND, McQuain WD, Keenan MA