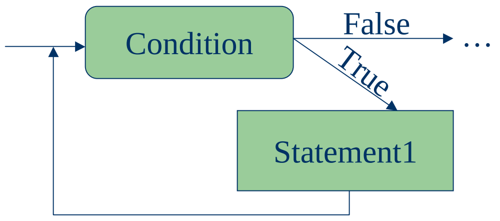
The C++ Language

Loops

Loops

- Recall that a loop is another of the four basic programming language structures
  - Repeat statements until some condition is false.



Loops in C++

- The `while` loop in C++ is the most generic form
- Syntax

```
while (Expression)
   Statement
```

- Semantics
  - Executes *Statement* as long as *Expression* evaluates to `true`

While Loop (Example)

- Recall from our Giving Change Algorithm 1

  2.2 If the value of `change` is >= 100, then perform the following steps.

  2.2.1 Add 1 to the value of `dollars`.

  2.2.2 Subtract 100 from the value of `change`.

  2.2.3 Repeat step 2.2
- This is a loop!

While Loop (Example)

```
const int ONE_DOLLAR = 100; // one dollar in cents

int dollars = 0;  // number of dollars in change
int change = -1;  // the amount of change

cin >> change;

while (change >= ONE_DOLLAR)
{
    dollars = dollars + 1;
    change = change - ONE_DOLLAR;
}
```

5

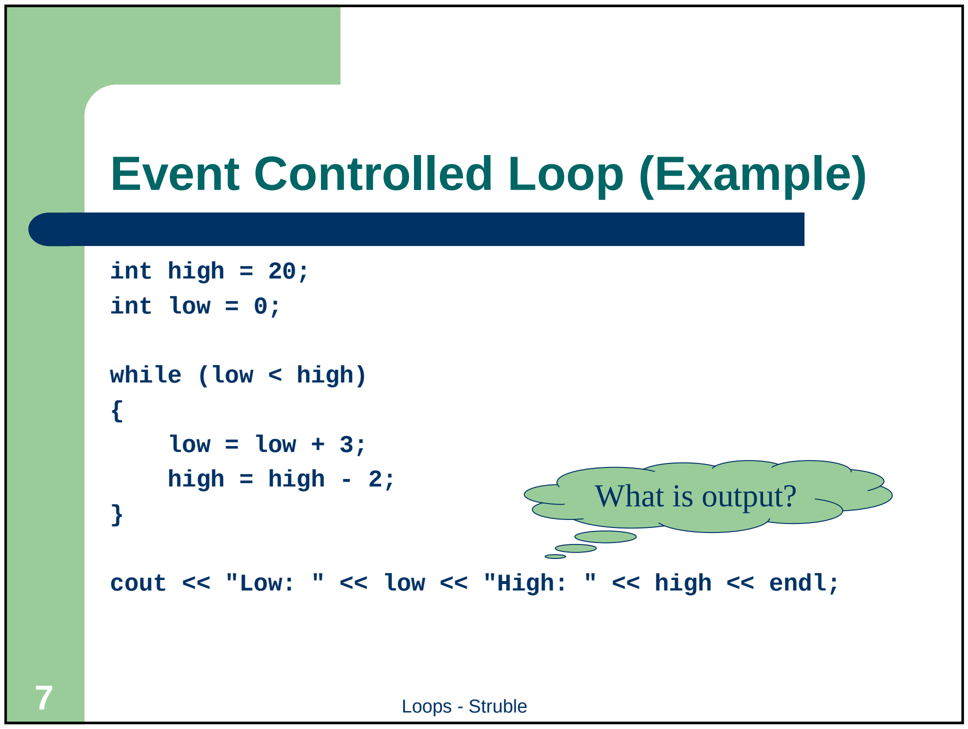
Loops - Struble

Kinds of Loops

- Event Controlled Loop
  - Executes until a specified situation
    - Describes all types of loops
- Count Controlled Loop
  - Executes a specified number of times
- Sentinel Controlled Loop
  - Executes until a dummy value is encountered in the input

6

Loops - Struble



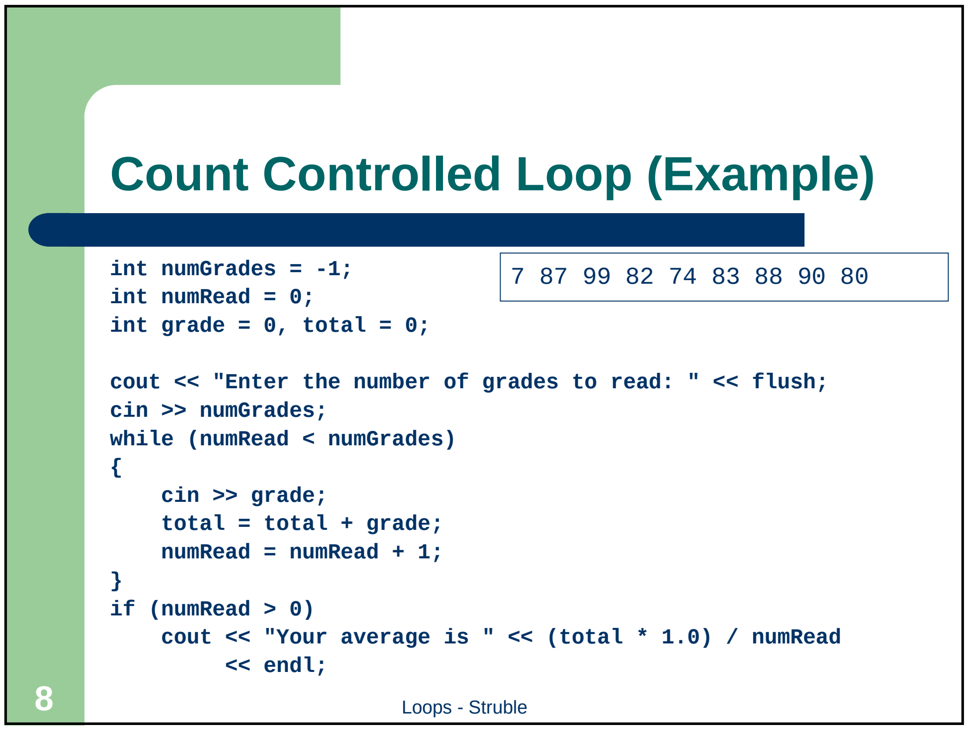
Event Controlled Loop (Example)

```
int high = 20;
int low = 0;

while (low < high)
{
    low = low + 3;
    high = high - 2;
}

cout << "Low: " << low << "High: " << high << endl;
```

What is output?

7

Loops - Struble



Count Controlled Loop (Example)

```
7 87 99 82 74 83 88 90 80
```

```
int numGrades = -1;
int numRead = 0;
int grade = 0, total = 0;

cout << "Enter the number of grades to read: " << flush;
cin >> numGrades;
while (numRead < numGrades)
{
    cin >> grade;
    total = total + grade;
    numRead = numRead + 1;
}
if (numRead > 0)
    cout << "Your average is " << (total * 1.0) / numRead
         << endl;
```

8

Loops - Struble

Sentinel Controlled Loop

```
87 99 82 74 83 88 90 -1 80
```

```
const int SENTINEL = -1;
int numRead = 0;
int grade = 0, total = 0;

cout << "Enter grades or " << SENTINEL << " to quit." << endl;
cin >> grade;                // priming read
while (grade != SENTINEL)
{
    total = total + grade;
    numRead = numRead + 1;
    cin >> grade;            // read the next grade
}
if (numRead > 0)
    cout << "Your average is " << (total * 1.0) / numRead
         << endl;
```

9

Exercises

- Write a loop to do the following:
  1. Read in 10 integers.
  2. Find the maximum value.
  3. Find the minimum value.
- Hints
  - Don't store all 10 values at once, calculate the maximum/minimum so far as you go.
  - Use INT_MIN and INT_MAX to initialize maximum/minimum value so far.
- Trace using the following input.

```
30 -209 45 827 -93 101 -445 79 827 83
```

10

Input Failure

- Each input stream (e.g., `cin` or input file stream) has a *state*
  - The state is *good* if every operation has succeeded
  - The state is *bad* if some operation has failed
    - Couldn't open a file
    - Couldn't read in a value of the expected type
    - Couldn't find delimiting character for `ignore` or `getline`
- We can test input streams for their state
  - `true` is good
  - `false` is bad

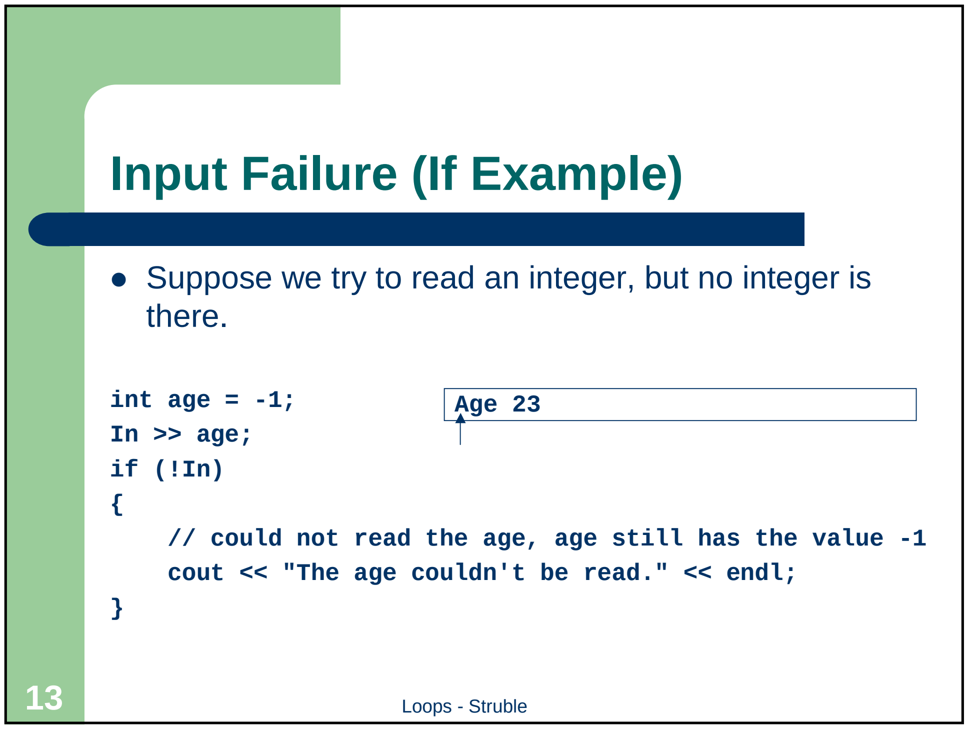
11

Input Failure (If Example)

- Use the stream variable as part of a boolean expression

```
ifstream In;
In.open("Data.txt");
if (In)
{
    cout << "The file opened properly." << endl;
}
else
{
    cout << "The file was not opened." << endl;
}
```

12

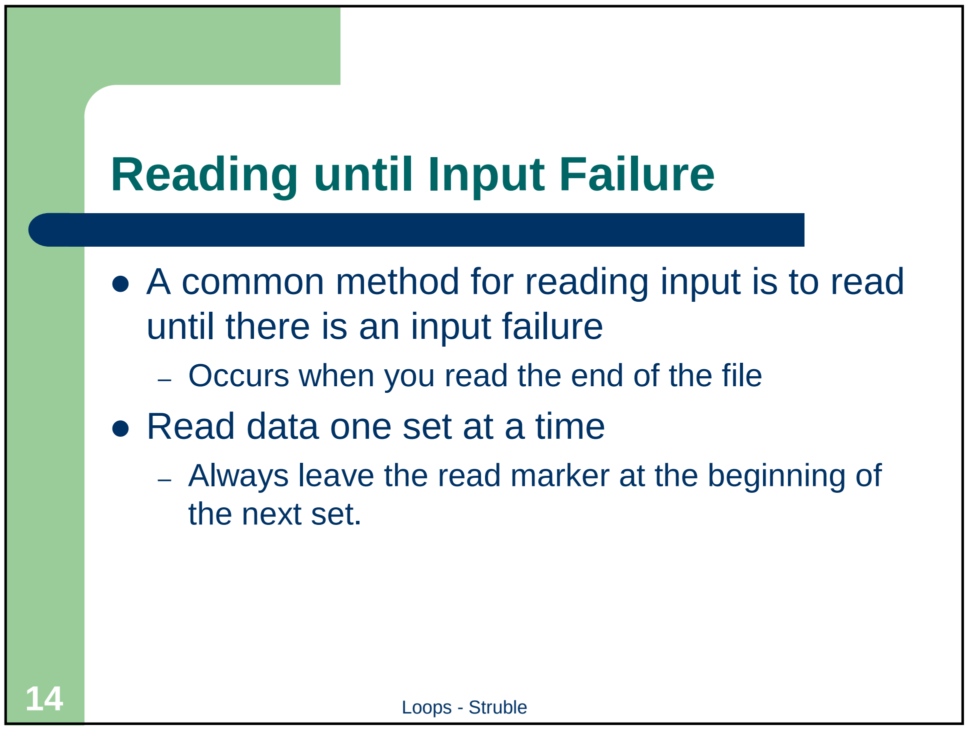


Input Failure (If Example)

- Suppose we try to read an integer, but no integer is there.

```
int age = -1;
In >> age;
if (!In)
{
    // could not read the age, age still has the value -1
    cout << "The age couldn't be read." << endl;
}
```

```
Age 23
↑
```



Reading until Input Failure

- A common method for reading input is to read until there is an input failure
  - Occurs when you read the end of the file
- Read data one set at a time
  - Always leave the read marker at the beginning of the next set.

Read until Input Failure (Correct Example)

```
const char DELIMITER = '|';
string name = "";
int age = -1;

ifstream In("Data.txt");

// Priming read for one data set
getline(In, name, DELIMITER);
In >> age;
In.ignore(INT_MAX, '\n');

while (In)
{
    cout << "Name: " << name << "\tAge: " << age << endl;
    // Read the next data set
    getline(In, name, DELIMITER);
    In >> age;
    In.ignore(INT_MAX, '\n');
}
```

```
Joe Missouri|32
Sally White|27
Missy Green|24
```

There's an invisible end of file character here.

```
Name: Joe Missouri   Age: 32
Name: Sally White    Age: 27
Name: Missy Green    Age: 24
```

Read until Input Failure (Incorrect Example)

```
const char DELIMITER = '|';
string name = "";
int age = -1;

ifstream In("Data.txt");


// No priming read
while (In)
{
    // Read a data set
    getline(In, name, DELIMITER);
    In >> age;
    In.ignore(INT_MAX, '\n');

    cout << "Name: " << name << "\tAge: " << age << endl;
}
```

```
Joe Missouri|32
Sally White|27
Missy Green|24
```

```
Name: Joe Missouri   Age: 32
Name: Sally White    Age: 27
Name: Missy Green    Age: 24
Name: Missy Green    Age: 24
```

Recovering from Input Failure

- When input failure occurs, you cannot read anything else until you reset the state of the stream
- Use the `clear()` function for input streams
- Syntax

```
InputStreamVariable.clear();
```

Recovering from Input Failure (Example)

```
ifstream In("Data.txt");
int anInt = -1;
int total = 0;
// read integers and total them
In >> anInt;
while (In) {
    total = total + anInt;
    In >> anInt;
}
// read the name that follows the integers
// Note that final extraction does skip the whitespace!
string name = "";
In.clear();              // reset the flags
getline(In, name);
```

```
25 15 30 45 Michael Jordan
```

Exercise

- Modify your minimum and maximum calculations to read until input failure instead of reading a fixed number of integers.
- Trace through your code with the input

```
205 -90 -103 199 76 823 -205 133 144 150
```

19

Loops - Struble

End-Of-File Handling

- Input streams support the `eof()`function
  - `true` if last input operation read end-of-file mark
  - `false` if last input operation did not read end-of-file mark
- Syntax

```
InputStreamVariable.eof()
```

20

Loops - Struble

End-Of-File Example

```
int anInt = 0;
char c = '\0';
ifstream In("Input.txt");

In >> anInt;
In.get(c);
while (!In.eof())
{
    cout << anInt << endl;
    In >> anInt;
    In.get(c);
}
```

Input

```
1
2
3

```

With newline

```
1
2
3
```

Without newline

21 Loops - Struble

Other Loops

- C++ provides alternative syntax for loops
  - `for` loops
  - `do` … `while` loops
- These alternatives can always be rewritten as while loops
  - *Syntactic sugar*

22 Loops - Struble

For Loops

- A `for` loop is the preferred syntax for count controlled loops.
- Syntax

```
for (Initialization; TestExpression; Update)
   Statement
```

- Initialization is often used to initialize variables
- TestExpression determines loop termination
- Update executes once an iteration
  - Updates values used in test expression

23

For Loop (Example)

```
int total = 0;              int x;
int x = 0;                  int total = 0;

while (x < 10)              for (x = 0; x < 10; x = x + 1)
{                               total = total + x;
   total = total + x;
   x = x + 1;
}
```

24

Increment and Decrement

- *post-increment* (++) and *post-decrement* (- -) operators are used frequently with for loops
- Examples

```
i++; // i = i + 1;
i--; // i = i - 1;
```

- Can be used in expressions, but should be avoided for readability.

25

For Loop and Increment (Example)

```
7 87 99 82 74 83 88 90 80
```

```
int numGrades = -1;
int numRead = 0;
int grade = 0, total = 0;

cout << "Enter the number of grades to read: " << flush;
cin >> numGrades;
for (numRead = 0; numRead < numGrades; numRead++)
{
    cin >> grade;
    total = total + grade;
}
if (numRead > 0)
    cout << "Your average is " << (total * 1.0) / numRead
         << endl;
```

26

For Loop and Decrement (Example)

```
const char STAR='*';  // what to draw
int numPrint;

cout << "How many stars do you want to print? " << flush;
for (cin >> numPrint; numPrint > 0; numPrint--)
    cout << STAR;

cout << endl;
```

Notice the use of an input statement here!

27

Loops - Struble

Pitfall

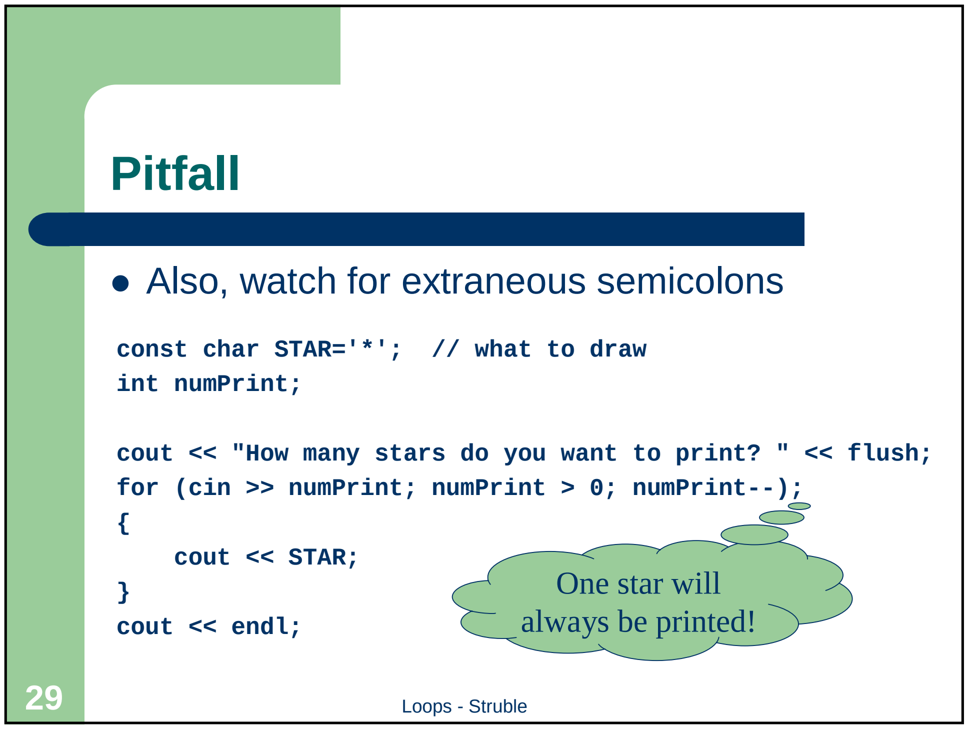
- Be careful about identifying the statements be executed in the for loop

```
const char STAR='*';  // what to draw
int numPrint;

cout << "How many stars do you want to print? " << flush;
for (cin >> numPrint; numPrint > 0; numPrint--)
    cout << STAR;
    cout << endl;
```

Only printing the stars is in the loop.

28

Loops - Struble



Pitfall

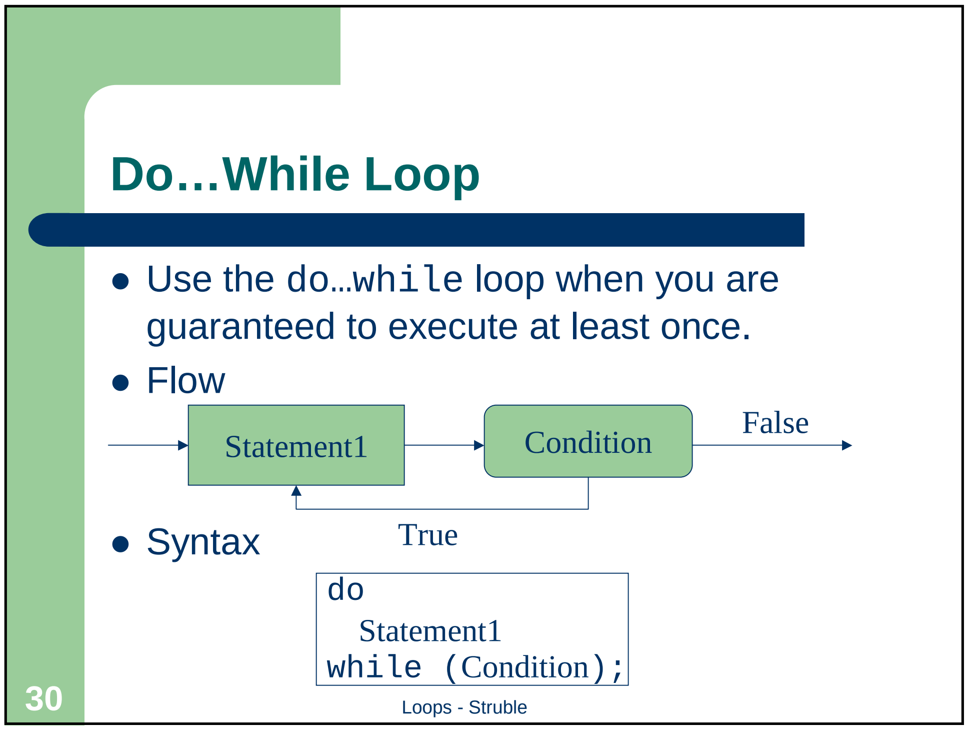
- Also, watch for extraneous semicolons

```
const char STAR='*';  // what to draw
int numPrint;

cout << "How many stars do you want to print? " << flush;
for (cin >> numPrint; numPrint > 0; numPrint--);
{
    cout << STAR;
}
cout << endl;
```

One star will always be printed!

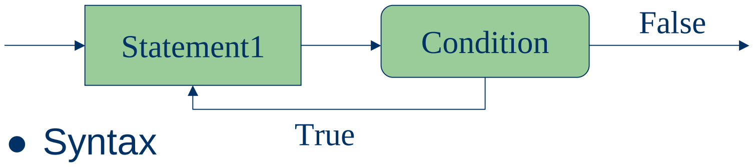
29

Loops - Struble



Do…While Loop

- Use the do…`while` loop when you are guaranteed to execute at least once.
- Flow

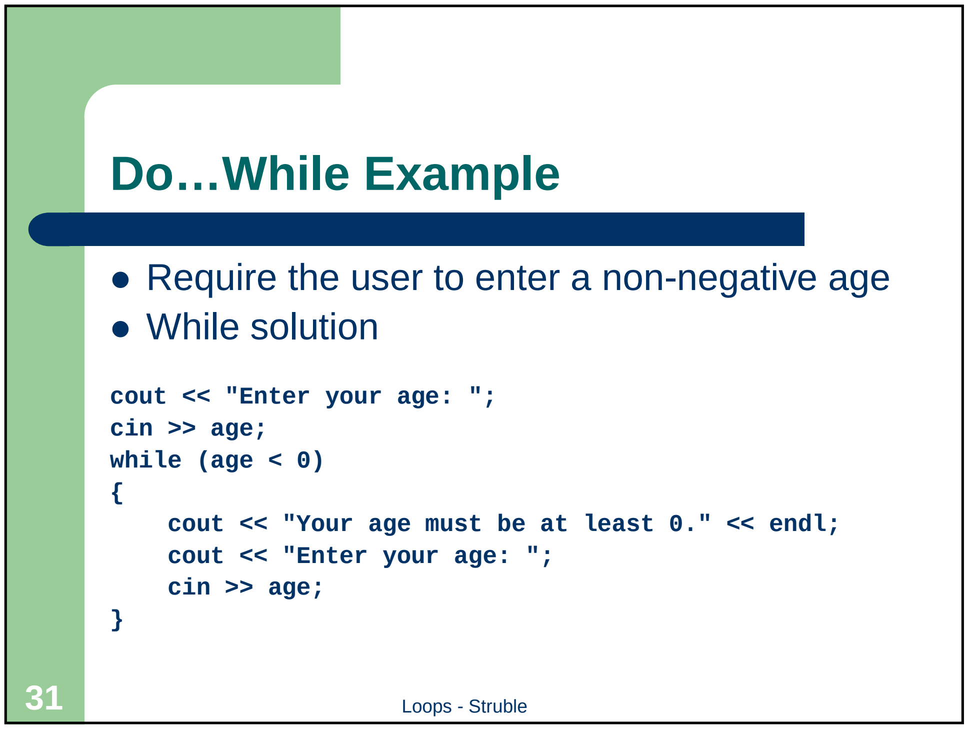


- Syntax

```
do
   Statement1
while (Condition);
```

30

Loops - Struble



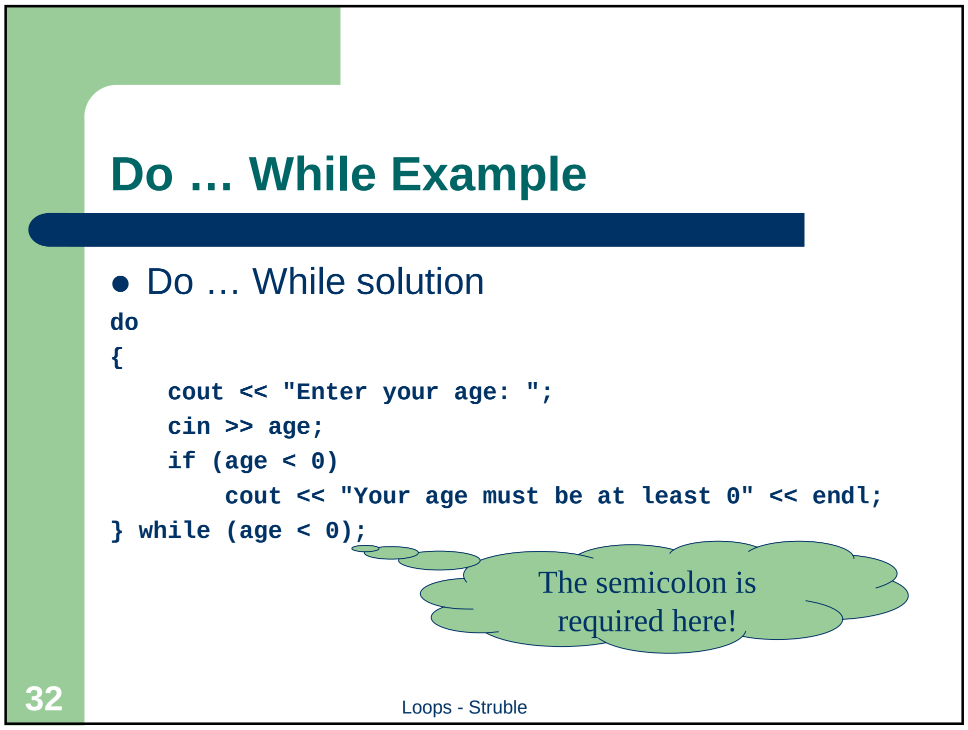
Do…While Example

- Require the user to enter a non-negative age
- While solution

```
cout << "Enter your age: ";
cin >> age;
while (age < 0)
{
    cout << "Your age must be at least 0." << endl;
    cout << "Enter your age: ";
    cin >> age;
}
```

31

Loops - Struble



Do … While Example

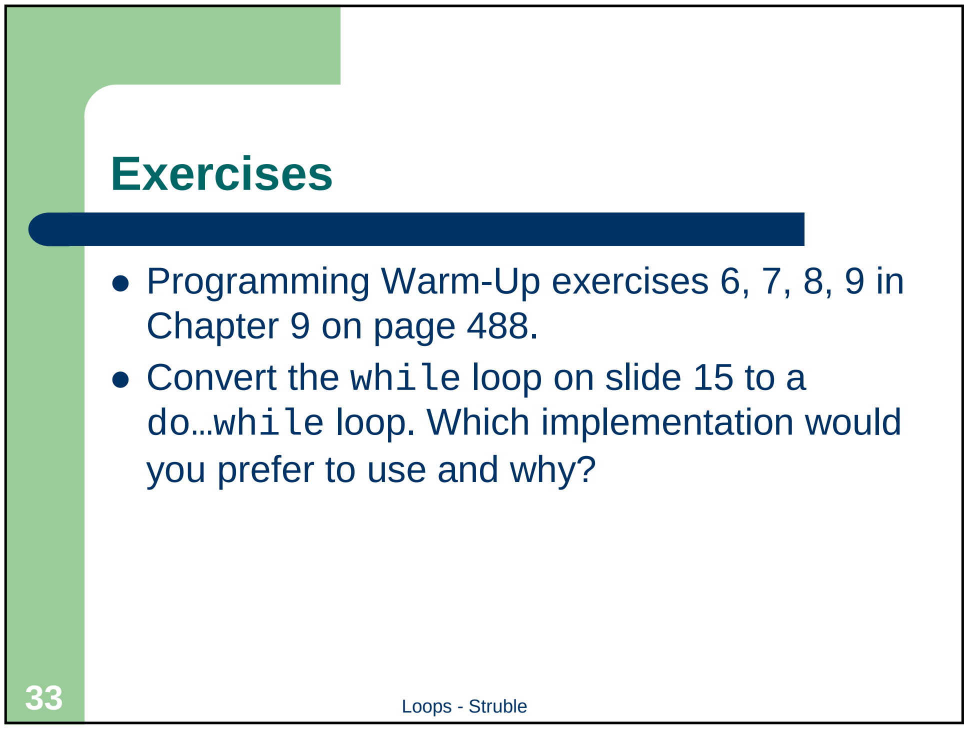
- Do … While solution

```
do
{
    cout << "Enter your age: ";
    cin >> age;
    if (age < 0)
        cout << "Your age must be at least 0" << endl;
} while (age < 0);
```

The semicolon is required here!

32

Loops - Struble

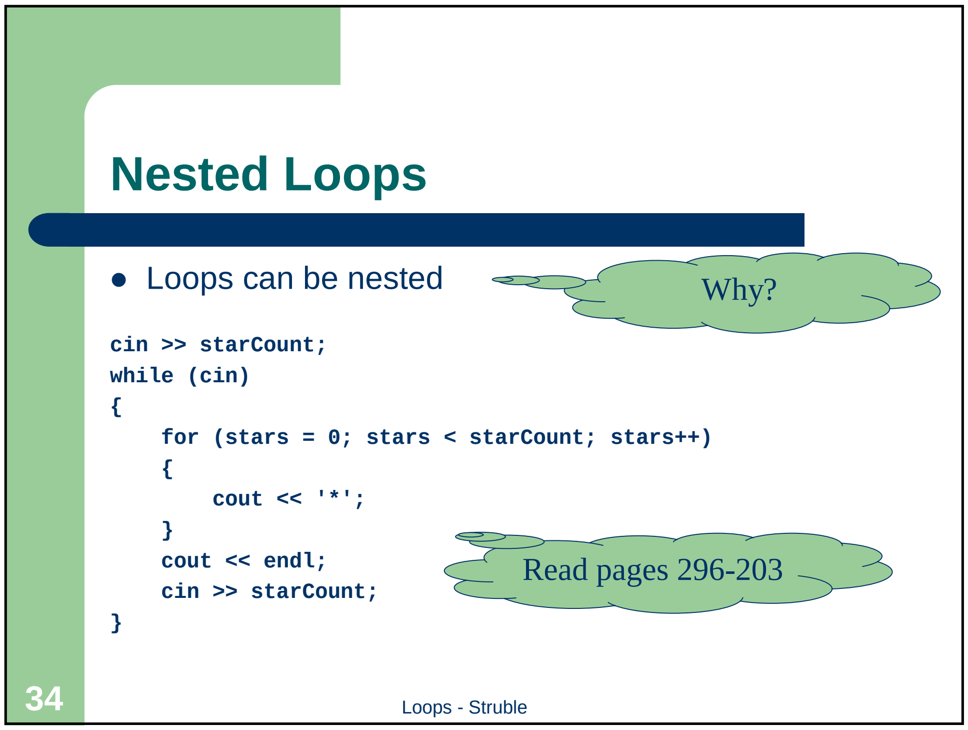


Exercises

- Programming Warm-Up exercises 6, 7, 8, 9 in Chapter 9 on page 488.
- Convert the `while` loop on slide 15 to a do…`while` loop. Which implementation would you prefer to use and why?

33

Loops - Struble



Nested Loops

- Loops can be nested

Why?

```
cin >> starCount;
while (cin)
{
    for (stars = 0; stars < starCount; stars++)
    {
        cout << '*';
    }
    cout << endl;
    cin >> starCount;
}
```

Read pages 296-203

34

Loops - Struble

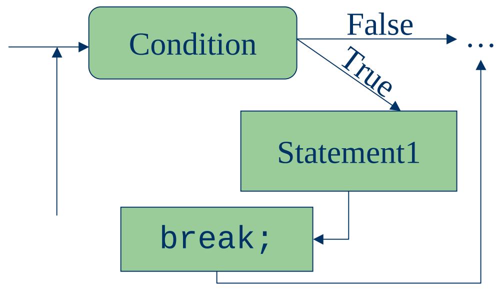
Other Control Statements

- Two other control statements
  - `break`
  - `continue`
- Change control flow in loops and `switch` statements

35

Break Statement

- Stops executing the innermost loop containing the `break` statement
- Flow (`while` loop)



36

Break Example

- Loop testing for input failure and sentinel

1 2 3 -1 4 5 6

```
const int SENTINEL = -1;
cin >> anInt;
while (In)
{
    if (anInt == SENTINEL)
        break;
    cout << anInt << endl;
    cin >> anInt;
}
```

37

Loops - Struble

Better implementation

```
const int SENTINEL = -1;

cin >> anInt;
while (In && anInt != SENTINEL)
{
    cout << anInt << endl;
    cin >> anInt;
}
```

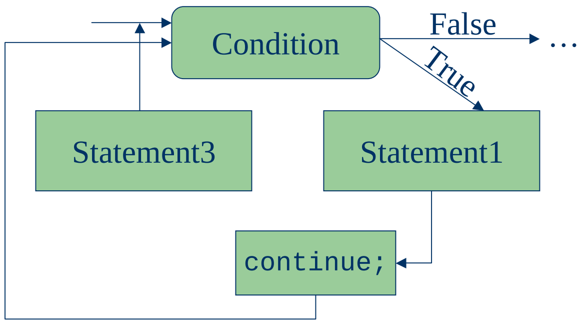
Why is this considered better?

38

Loops - Struble

Continue Statement

- Skips the rest of an iteration
- Flow (`while` loop)



39

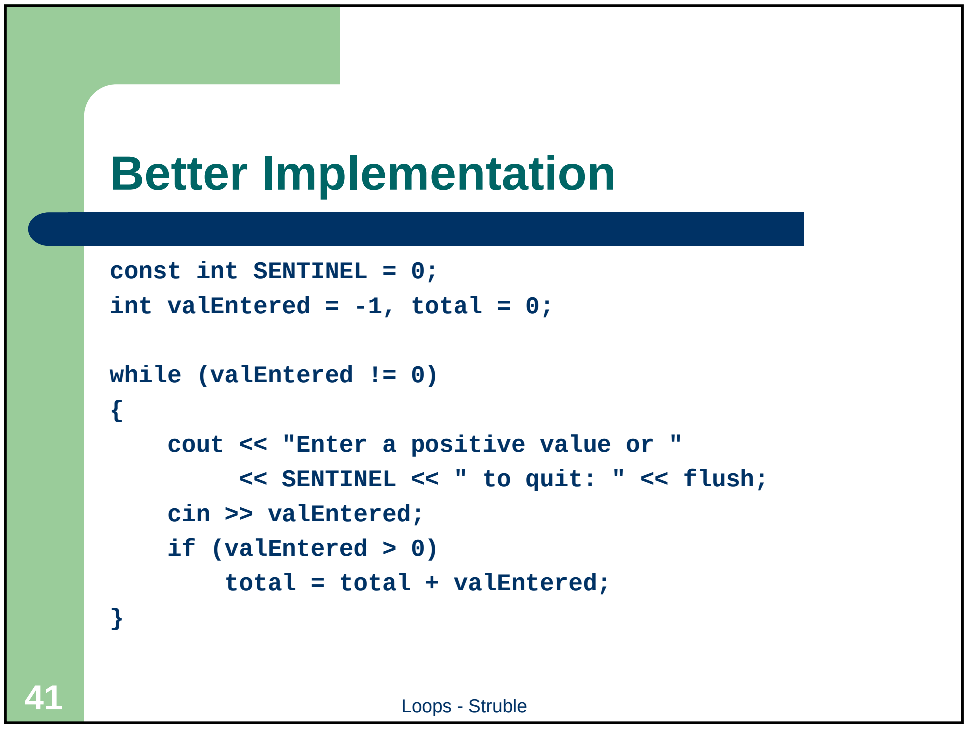
Continue Example

- Total positive values

1 -2 3 0 4 5 6

```
const int SENTINEL = 0;
int valEntered = -1, total = 0;

while (valEntered != 0)
{
    cout << "Enter a positive value or "
         << SENTINEL << " to quit: " << flush;
    cin >> valEntered;
    if (valEntered < 0)
        continue;
    total = total + valEntered;
}
```

40

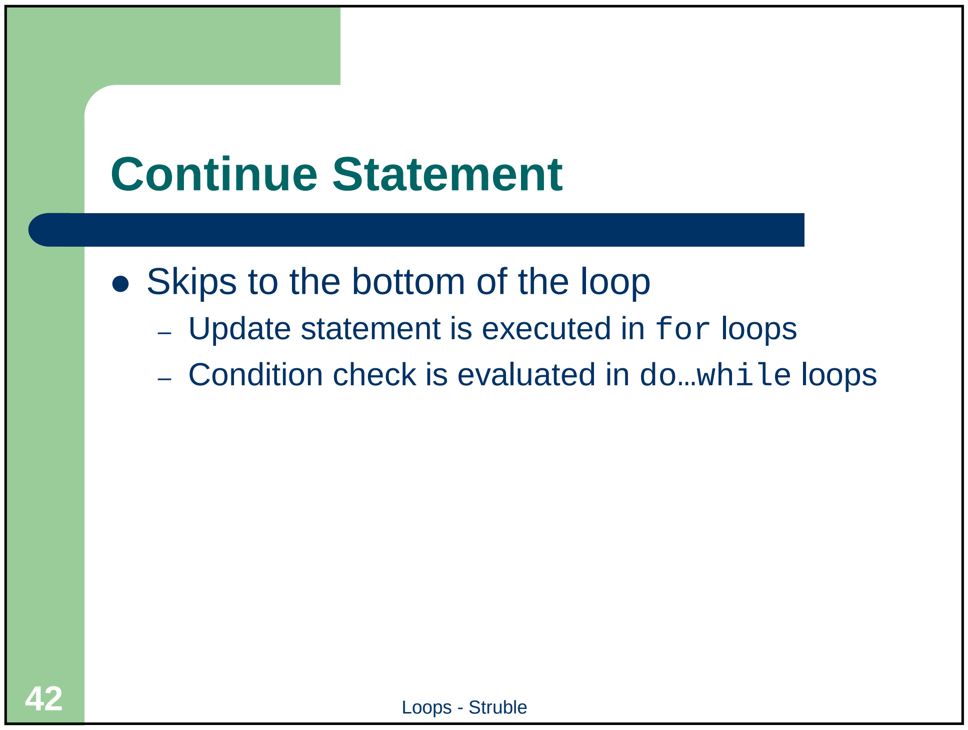


Better Implementation

```
const int SENTINEL = 0;
int valEntered = -1, total = 0;

while (valEntered != 0)
{
    cout << "Enter a positive value or "
         << SENTINEL << " to quit: " << flush;
    cin >> valEntered;
    if (valEntered > 0)
        total = total + valEntered;
}
```

41

Loops - Struble



Continue Statement

- Skips to the bottom of the loop
  - Update statement is executed in `for` loops
  - Condition check is evaluated in do…`while` loops

42

Loops - Struble