`while` Loops

Lecture 12
Sections 5.8 - 5.9

Robb T. Koether

Hampden-Sydney College

Mon, Sep 23, 2019

Outline

Iterative Structures

- An iterative structure allows a block of statements to be executed repeatedly.
- The iteration continues until a specified condition fails, then it terminates.
- Computers derive their immense computational power through a combination of decision structures, iterative structures, and speed.

The `while` Statement

- The **`while`** statement will repeatedly execute a block of statements as long as a specified boolean expression is true.
- Once the boolean expression is false, execution exits the **`while`** loop and continues with the next statement following the loop.

The `while` Statement

The `while` Statement

```
while (boolean-expression)
{
    action
}
```

- Special situations
  - If the *boolean-expression* is initially false, then the *action* is never executed.
  - If the *boolean-expression* is always true, then the loop never stops.

Examples

- Examples
  - `SumNIntegers.cpp`

Outline

Input Loops

- Often the purpose of a loop is to process a list of numbers as they are read in.
- There are three standard ways to control such a loop.
  - By sentinel value.
  - By end-of-file.
  - By a counter.

Outline

Loops Controlled by a Sentinel Value

- A sentinel value is a special value appended to the input to indicate the end of the list.
- For example, if the data represent test scores, a sentinel value of -1 or 999 may be used.
- Caution
  - The sentinel value must be a value that cannot occur otherwise.
  - The sentinel value should not be processed as regular data.

Unrolling a Loop

- The pattern in the loop is
  1. prompt user for input
  2. read input
  3. test for sentinel value
  4. carry out the action
  5. prompt
  6. read
  7. test
  8. action
  9. ⋮

Unrolling a Loop

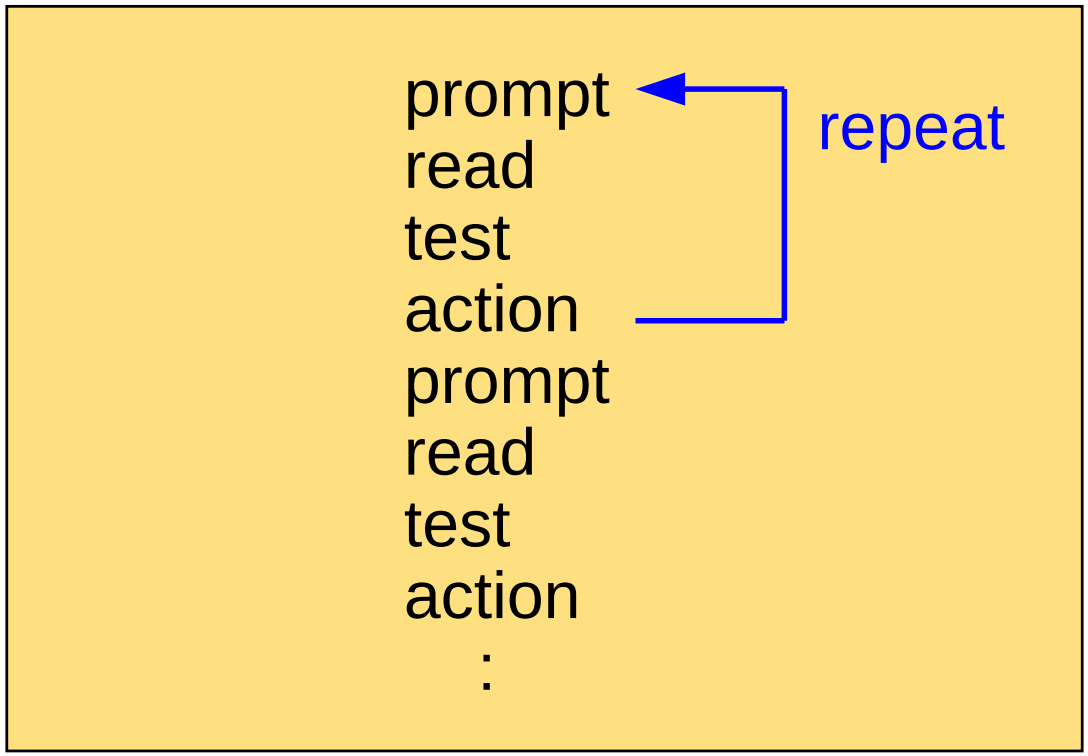
- The pattern begins to repeat with the prompt statement.
- However, the test must occur at the top of the loop, in the **`while`** statement.
- Therefore, the first prompt and read must come before the **`while`** loop.
- And the body of the **`while`** loop must follow the pattern:
  - action
  - prompt
  - read

Unrolling a Loop

prompt
read
test
action
prompt
read
test
action
:

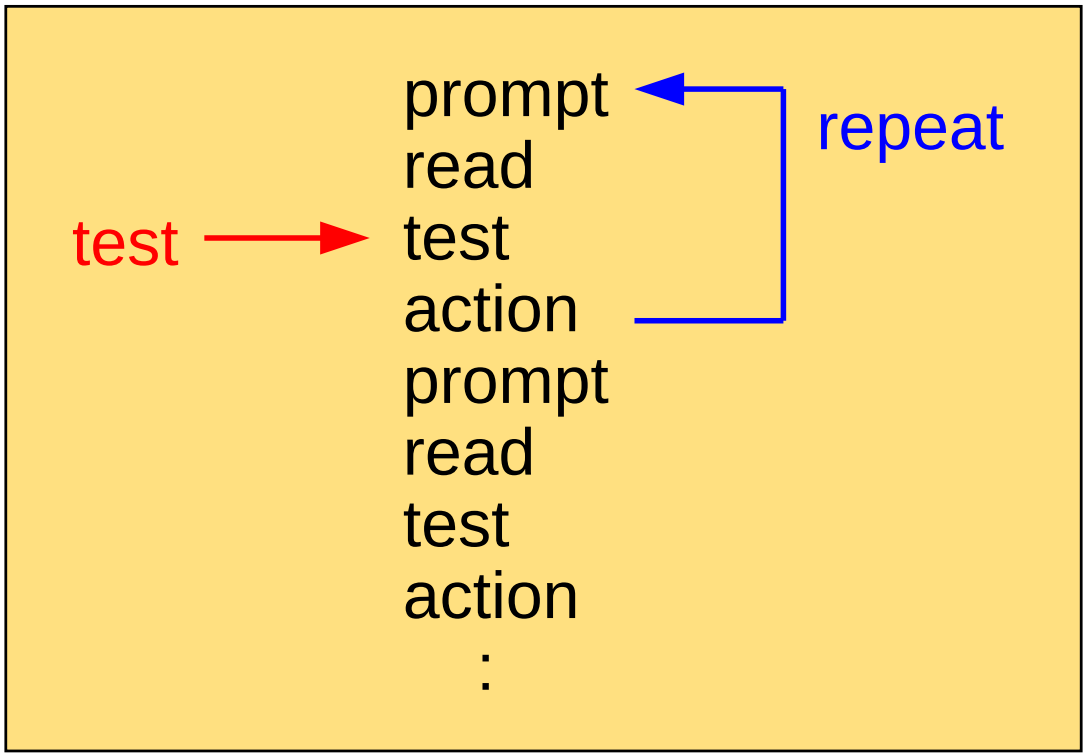
The “unrolled” loop

Unrolling a Loop



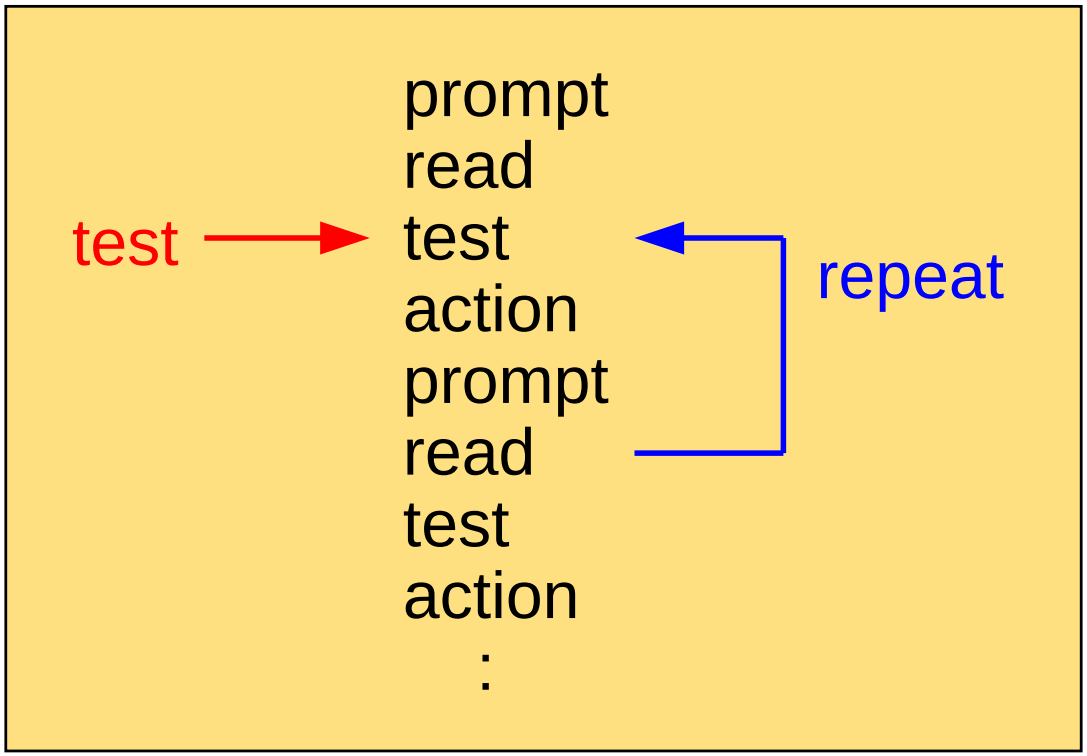
Repeat after the *action*

Unrolling a Loop



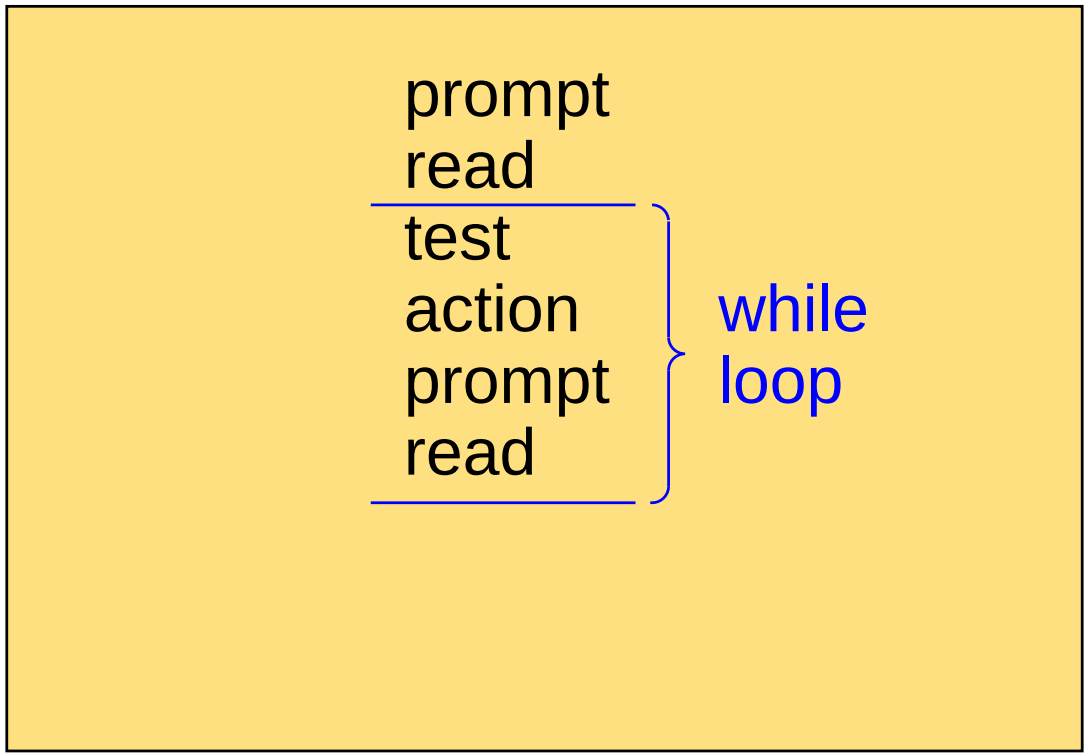
The *test* is here

Unrolling a Loop



The *test* must be in the top of the loop

Unrolling a Loop



This must be the **`while`** loop

Loops Controlled by a Sentinel Value

Loops Controlled by a Sentinel Value

```
const int SENTINEL = value;
int number;
prompt user for input
cin >> number;
while (number != SENTINEL)
{
    action
    prompt user for input
    cin >> number;
}
```

Loops Controlled by a Sentinel Value

- Example
  - `SentinelSum.cpp`

Outline

Detecting End of File (EOF)

- There is an `istream` function `eof()` (end of file) that returns **`true`** when the program attempts to read past the end of a file. Otherwise, it returns **`false`**.
- The **`while`** loop may use the boolean expression

```
!cin.eof()
```

- When input is through the keyboard, there is no "file." In this case, EOF can be simulated by typing `CTRL-Z` (Windows).

Using the `eof()` Function

Using the `eof()` Function

```
int number;
prompt user for input
cin >> number;
while (!cin.eof())
{
    action
    prompt user for input
    cin >> number;
}
```

Example of EOF

- Example
  - `EOFFuncSum.cpp`

Loops Controlled by EOF

- When the input operator `>>` attempts to read past the end of a file, it returns **`false`** (0). Otherwise, it returns *`true`* (nonzero).
- Thus, the expression

  ```
  cin >> number
  ```

  may be used as a boolean expression in a **`while`** statement.
- This expression will both read and test the input.

Loops Controlled by EOF

Loops Controlled by EOF

```
int number;
prompt user for input
while (cin >> number)
{
    action
    prompt user for input
}
```

Example of EOF

- Example
  - `EOFOpSum.cpp`

Outline

Assignment

Assignment

- Read Sections 5.8 - 5.9.