Ch 5. Looping
Part 1

CS 1428
Fall 2011

Jill Seaman

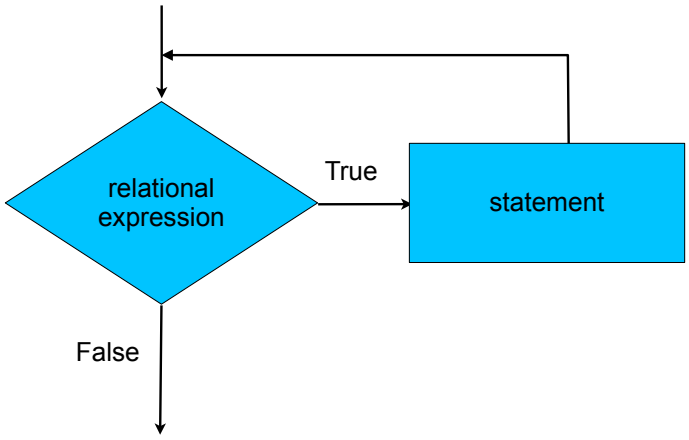
Lecture 12

Control Flow

- So far, control flow in our programs has included:
  - sequential processing (stmnts done in order)
  - branching (conditionally skip some statements)
- Chapter 5 introduces loops, which allow us to conditionally repeat execution of a set of statements.
  - while loop
  - do-while loop
  - for loop

The while loop

- The statement is repeated as long as the relational expression is true.



while

- the while statement is used to repeat statements

```
while (expression)
   statement
```

- expression is evaluated:
  - If it is true, then statement is executed, and expression is re-evaluated
  - If/when it is false, then statement is skipped, and the loop is exited.

while example

- Example:

```
int number = 1;

while (number <= 3)
{
   cout << "Student" << number << endl;
   number = number + 1;
}

cout << "Done" << endl;
```

- Output:

```
Student1
Student2
Student3
Done
```


while structure

- Notice:

```
while (number <= 3)
{
   cout << "Student" << number << endl;
   number = number + 1;
}
```

- relational expression in parentheses.
- NO semi-colon after relational expression.
- Good style: indent the statements in the body.
- The body can be a block.
- The body can be one statement.

Watch out

- What is output?

```
int x = 13;

while (x <= 10) {
   cout << "Repeat!" << endl;
   x = x + 1;
}

cout << "Done!" << endl;
```

- If the condition is false the first time, the body is NEVER executed.

Watch out

- What is output?

```
int x = 1;

while (x <= 10)
   cout << "Repeat!" << endl;

cout << "Done!" << endl;
```

- Something inside the body must eventually make the condition false.
- If not, you have an infinite loop.
  - try ctrl-c to exit

Watch out

- What is output?

```
int x = 1;

while (x <= 10)
   cout << "Repeat!" << endl;
   x = x + 1;

cout << "Done!" << endl;
```

- Don't forget the braces!!
- Another watchout:
  - don't use = for ==

Using while for Input Validation

- Inspect user input values to make sure they are valid.
- If not valid, ask user to re-enter value.

```
int number;

cout << "Enter a number between 1 and 10: ";
cin >> number;
while (number < 1 || number > 10) {
   cout << "Please enter a number between 1 and 10: ";
   cin >> number;
}

// Do something with number here
```

- What is another way to write the relational expression?

Using while for Input Validation

- Can check for valid characters

```
char answer;

cout << "Enter the answer to question 1 (a,b,c or d): ";
cin >> answer;
while (answer != 'a' && answer != 'b' &&
       answer != 'c' && answer != 'd')
{
   cout << "Please enter a letter a, b, c or d: ";
   cin >> answer;
}

// Do something with answer here
```


Counters

- A counter is a variable used to keep track of loop iterations.

```
cout << "Number  Number Squared" << endl;
cout << "------  --------------" << endl;

int  num = 1;
while (num <= 8)
{
   cout << num << "           " << (num * num) << endl;
   num = num + 1;  // increment the counter
}
```

- Output:

```
Number  Number Squared
------  --------------
1            1
2            4
3            9
4            16
5            25
6            36
7            49
8            64
```