Control Flow

- So far, control flow in our programs has included:
  - sequential processing (stmts 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 (expression)  
statement
```

- while

- the while statement is used to repeat statements

- 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:

```cpp
int number = 1;
while (number <= 3)
{
    cout << “Student” << number << endl;
    number = number + 1;
}
cout << “Done” << endl;
```

• Output:

Student1
Student2
Student3
Done

while structure

• Notice:

```cpp
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?

```cpp
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?

```cpp
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?

```cpp
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.

```cpp
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

```cpp
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.

```cpp
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:

<table>
<thead>
<tr>
<th>Number</th>
<th>Number Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
</tr>
</tbody>
</table>