Control Flow
(order of execution)

• So far, control flow in our programs has included:
  ‣ sequential processing (1st statement, then 2nd statement…)
  ‣ branching (conditionally skip some statements).

• Chapter 5 introduces loops, which allow us to conditionally repeat execution of some statements.
  ‣ while loop
  ‣ do-while loop
  ‣ for loop

5.2 The **while** loop

• As long as the relational expression is true, repeat the statement

```
while (expression)
  statement
```

• How it works:
  ‣ expression is evaluated:
  ‣ If it is true, then statement is executed, then it starts over (and expression is evaluated again).
  ‣ If (when) it is false, then statement is skipped (and the loop is done).
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

5.3 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
```

Input Validation

• Checking 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
```

5.4 Counters

• **Counter**: a variable that is incremented (or decremented) each time a loop repeats.
• Used to keep track of the number of iterations (how many times the loop has repeated).
• Must be initialized before entering loop!!!
Counters

- Example (how many times do they enter an invalid number?):

```cpp
int number;
int count = 0;

cout << "Enter a number between 1 and 10: ";
cin >> number;

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

cout << count << " invalid numbers entered " << endl;

// Do something with number here
```

Counters

- Example, using the counter to control how many times the loop iterates:

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

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

Output:

5.5 The do-while loop

- Execute the statement(s), then repeat as long as the relational expression is true.

```
Statement(s)
------------
Expression
```

```cpp
do statement
while (expression);
```

Yes, it always executes at least once.

do-while syntax and semantics

- The do-while loop has the test expression at the end:

```
do statement
while (expression);
```

- How it works:
  - statement is executed.
  - expression is evaluated:
    - If it is true, then it starts over (and statement is executed again).
    - If (when) it is false, the loop is done.

- statement always executes at least once.
do-while example

• Example:

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

• Output

| Student1 | Student2 | Student3 | Done |

Different ways to control the loop

• Conditional loop: body executes as long as a certain condition is true
  ‣ input validation: loops as long as input is invalid

• Count-controlled loop: body executes a specific number of times using a counter
  ‣ actual count may be a literal, or stored in a variable.

• Count-controlled loop follows a pattern:
  ‣ initialize counter to zero (or other start value).
  ‣ test counter to make sure it is less than count.
  ‣ update counter during each iteration.

5.6 The for loop

• The for statement is used to easily implement a count-controlled loop.

```cpp
for (expr1; expr2; expr3)
statement
```

• How it works:
  ‣ expr1 is executed (initialization)
  ‣ expr2 is evaluated (test)
  ‣ If it is true, then statement is executed, then expr3 is executed (update), then start over.
  ‣ If (when) it is false, then statement is skipped (and the loop is done).
The for loop flow chart

for (expr1; expr2; expr3)
statement

expr1

expr2

expr3

True

False

The for loop and the while loop

• The for statement

for (expr1; expr2; expr3)
statement

• is equivalent to the following code using a while statement:

expr1; // initialize
while (expr2) { // test
    statement
    expr3; // update
}

for loop example

• Example:

```cpp
int number;
for (number = 1; number <= 3; number++)
{
    cout << "Student" << number << endl;
}
cout << "Done" << endl;
```

• Output

```
Student1
Student2
Student3
Done
```

Counters: redo

• Example, using the counter to control how many times the loop iterates:

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

• Rewritten using a for loop:

```cpp
cout << "Number Number Squared" << endl;
cout << "------  --------------" << endl;
int num;
for (num = 1; num <= 8; num++)
    cout << num << " " << (num * num) << endl;
```
Define variable in init-expr

- You may define the loop counter variable inside the for loop’s initialization expression:

```cpp
for (int x = 10; x > 0; x=x-2)
    cout << x << endl;
```

- Do NOT try to access x outside the loop (the scope of x is the for loop statement ONLY)
- What is the output of the for loop?

User-controlled count

- You may use value input by the user to control the number of iterations:

```cpp
int maxCount;
cout << “How many squares do you want?” << endl;
cin >> maxCount;

for (int num = 1; num <= maxCount; num++)
    cout << num << “ “ << (num * num) << endl;
```

- How many times does the loop iterate?

The exprs in the for are optional

- You may omit any of the three exprs in the for loop header

```cpp
int value, incr;
cout << “Enter the starting value: “;
cin >> value;

for ( ; value <= 100; )
{
    cout << “Please enter the next increment amount: “;
cin >> incr;
    value = value + incr;
    cout << value << endl;
}
```

- Style: use a while loop for something like this.
- When expr2 is missing, it is true by default.