More Flow of Control

Adam Sweeney

Engineering Educator, EECS

Fall 2017

Introduction

- Most of Chapter 3 from textbook
- Talk more about branching and loops
  - Introduce some new methods

Agenda

- Boolean Expression review
- Multiway branches
- More on loops

BOOLEAN EXPRESSION REVIEW

Boolean Expression Review

- Evaluate the following: `!(false || true)`

Boolean Expression Review (cont.)

- Evaluate the following: `!(false || true)`
  - (false || true) → true
  - !(true)
  - false

Boolean Expression Review (cont.)

- Precedence

The unary operators +, - ++, --, and !	Highest Precedence (done first)
The binary arithmetic operations *, /, %	
The binary arithmetic operations +, -	
The Boolean operations <, >, <=, >=	
The Boolean operations ==, !=	
The Boolean operation &&	
The Boolean operation ||	Lowest Precedence (done first)

Boolean Expression Review (cont.)

- Consider the following:
  - An if-else block that requires checking a timer, and ensuring it is under a certain limit
  - Implemented like so:

```
if (!time > limit)
    [Something]
else
    [Something_else]
```

  - Assume time is 36 and limit is 60
    - How does the Boolean expression evaluate? Why?

Boolean Expression Review (cont.)

- Consider the precedence
  - ! is evaluated first
  - !36 evaluates to false
    - 36 evaluates to true, since it is non-zero
  - false is converted to 0 for the integer comparison
  - 0 > 60 is false
  - So, (!time > limit) evaluates to false
- How can we fix it?

Boolean Expression Review (cont.)

- Ensure proper precedence is used
  - (!time > limit) → (!(time > limit))

Boolean Expression Review (cont.)

- Ensure proper precedence is used
  - (!time > limit) → (!(time > limit))
- Avoid ! altogether
  - (time <= limit)
    - This is easier to read and understand as well

MULTIWAY BRANCHES

Multiway Branches

- Last week covered if-else branching
- What if there are more than three branches?

Multiway Branches

- Last week covered if-else branching
- What if there are more than three branches?
  - We can use nested if-else blocks

Multiway Branches (cont.)

- See the nested example of a four branch scenario
- This is cumbersome to read
- Takes up a lot of space

```
if (BOOLEAN_EXPRESSION) {
    STEPS
}
else {
    if (BOOLEAN_EXPRESSION) {
        STEPS
    }
    else {
        if (BOOLEAN_EXPRESSION) {
            STEPS
        }
        else {
            STEPS
        }
    }
}
```

Multiway Branches (cont.)

- Remember the properties of if-else
- if and else will execute a single statement
  - Without the need for braces
  - What would it look like if the else's single statement was an if statement?

Multiway Branches (cont.)

- Better way to implement nested if-else
- Much easier to read
- Takes up less space
- Preferred style is shown

```
if (BOOLEAN_EXPRESSION) {
    STEPS
} else if (BOOLEAN_EXPRESSION) {
    STEPS
} else if (BOOLEAN_EXPRESSION) {
    STEPS
} else {
    STEPS
}
```

Multiway Branches (cont.)

- Switch
  - Another way to handle multiway branching
  - Some restrictions in comparison to if-else
  - More elegantly handles certain scenarios

Multiway Branches (cont.)

- Demo time

MORE ON LOOPS

More on Loops

- The while loop is really all we need
  - Once a while and do-while are in the middle of their iterations, they are essentially indistinguishable
  - But sometimes the syntax doesn't gel as cleanly as we would like
  - These extra ways to do the same thing are called 'syntactic sugar'
  - I prefer to take advantage if I feel it suits me
- Discuss a new loop syntax

More on Loops (cont.)

- The for loop

```
for (INITIALIZATION_ACTION; BOOLEAN_EXPRESSION; UPDATE_ACTION)
    [Statement];
```

- Invoke with ‘for’, three expressions in parentheses
  - Note where a semicolon is and is not used
- By default, for loop can execute a single statement
  - Use { } to create a compound statement

More on Loops (cont.)

- Demo!