CONDITIONAL IF ELSE STATEMENTS

CSC100 Introduction to programming in C/C++ (Spring 2023)

Marcus Birkenkrahe

April 1, 2024

1 Simple and nested 'if' structures

- In this section of the course, we go beyond simple statements and turn to program flow and evaluation of logical conditions
- This section follows chapter 3 in Davenport/Vine (2015) and chapters 4 and 5 in King (2008)
- All available material is in GitHub.

2 Overview and example

- `If` statement structure in C is very similar to pseudocode `"If"`
- The code block 2 is the C version of the pseudocode:

```
// checks if health is smaller or equal 100
if (health <= 100)
  // drink health potion if condition is TRUE
else
  //resume battle if condition is FALSE
```

- Differences: condition needs *parentheses* `(...)`; no `"end if"` statement
- The `health` check results in a *Boolean* answer: true (`1`) or false (`0`)
- Unlike other languages, there are no Boolean types `TRUE`, `FALSE`
- To run, the program needs a *declaration* of the `health` variable

- *Multiple statements* need to be included in braces `{...}`
- The source code ?? will run. The variable has been declared and initialized:

```
int health = 99;

if (health <= 100) {
  // drink health potion
  puts("This is what you do:");
  printf("Drinking health potion!\n");
 }
 else {
   // resume battle
   puts("This is what you do:");
   printf("Resuming battle!\n");
 }
```

```
This is what you do:
Drinking health potion!
```

3 Stacked vs. nested IF structures

- In the example 3, the stacked `if` statements are evaluated independently, case by case. It does not matter if any of them fails. The `switch` control structure (next) is built this way.

```
if ( i == 1 )
// do one thing

if ( i == 2)
// do another thing
```

  The figure 1 shows the BPMN model for this program:

- In the example 3, the second part of the IF statement is entered only if the first condition fails.

```
if ( i == 1 ) {
   // do one thing
```

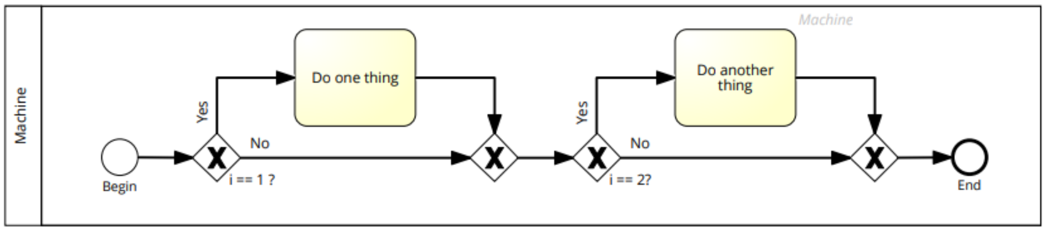

Figure 1: Single IF statements

```
}
else if ( i == 2) {
   // do another thing
}
```

- The figure 2 shows the BPMN model for this program:

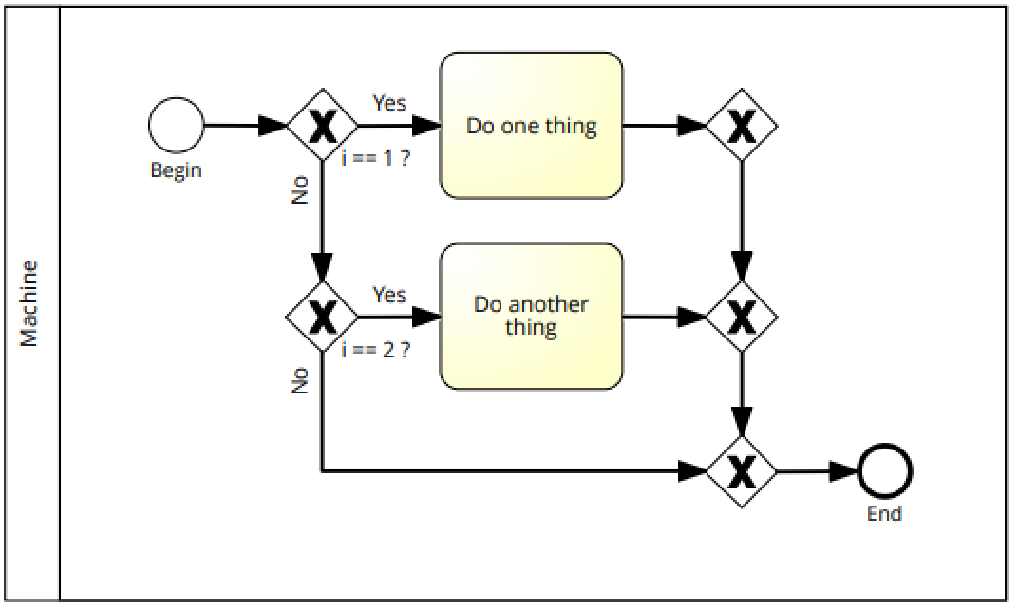

Figure 2: Single IF statements

- Which one of these you implement, depends strongly on the problem and on your performance requirements (they're quite different in speed

- which you do you think performs better?[1])

4 Let's practice!

- Open the Emacs browser with `M-x eww` and enter the URL tinyurl.com/ifelse-org
- Save the file with `C-x C-w` as `ifelse.org`
- Close the buffer with `C-x k` and reopen the file with `C-x C-f`
- Add your own name at the top and pledge
- Complete the file
- Submit the completed file to Canvas

5 References

- Davenport/Vine (2015) C Programming for the Absolute Beginner (3ed). Cengage Learning.
- GVSUmath (Aug 10, 2012). Proving Logical Equivalences without Truth Tables [video]. URL: youtu.be/iPbLzl2kMHA.
- Kernighan/Ritchie (1978). The C Programming Language (1st). Prentice Hall.
- King (2008). C Programming - A modern approach (2e). W A Norton.
- Orgmode.org (n.d.). 16 Working with Source Code [website]. URL: orgmode.org

[1]Answer: in this case (with mutually exclusive conditions), the nested statement (`if =... =else`) is generally more efficient because only one of the statements has to be checked, not both.