Stacked and Nested If Structures

CSC 100 - Lyon College - Spring 2025

Marcus Birkenkrahe

March 14, 2025

Contents

1 Introduction **1**

2 Sequential `if` structures **2**

3 Stacked `if` (ladder) structures **3**
3.1 Explanation 3
3.2 Example 3
3.3 Practice: Loot by Difficulty 4

4 Nested `if` Structures **6**
4.1 Explanation 6
4.2 Example 6
4.3 Practice: Boss Fight (Bonus Assignment) 7

5 Example with stacked and nested conditional statements **8**

6 What about curly brackets? **9**

7 Conclusion **10**

1 Introduction

- **Objectives**:
  - Sequential `if` structures
  - Stacked `if` structures
  - Nested `if` structures

- **Theme:** Text-based adventure game—dungeon exploration and decision-making.
- **Objective:** Learn sequential, stacked and nested `if` structures for complex decisions in pseudocode and C source code.

2 Sequential `if` structures

- Sequential `if` structures are evaluated independently and sequentially from the top. Multiple conditions can be true, and all matching blocks will execute. The outcome of one does not affect the others.
- Simple sequential `if` statements:

```
if ( i == 1 ) {
   // do one thing
}
if ( i == 2) {
   // do another thing
}
```

- An illustration in BPMN:

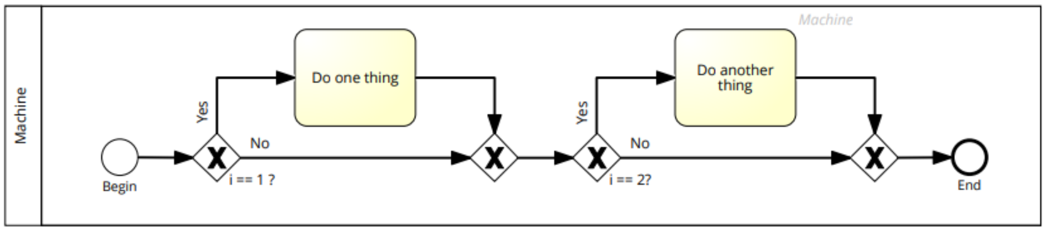

Figure 1: Single IF statements

- Example:
  1. If you have enough gold (> 100), buy a sword.
  2. If you have too little health (< 100), drink a potion.

```
IF gold > 100
   PRINT "Buying a sword!"
```

```
END IF

IF health < 100
   PRINT "Drinking a potion!"
END IF
```

3 Stacked `if` (ladder) structures

3.1 Explanation

- Stacked `if` structures test a sequence of conditions, but only one block executes. Once a condition is true, the rest are skipped.
- Simple stacked `if` statements:

```
if ( i == 1 ) {
   // do one thing
}
else if ( i == 2) {
   // do another thing
}
```

- An illustration in BPMN:

3.2 Example

- Game Context: After defeating an enemy, the player's reward depends on their health.
- Pseudocode:

```
SET health = 60

IF health > 75 THEN
    PRINT "You're in top form! You find a golden shield."
ELSE IF health > 50 THEN
    PRINT "You're scratched but standing. You find a rusty sword."
ELSE IF health > 25 THEN
    PRINT "You're wounded but alive. You find a healing potion."
ELSE
    PRINT "You collapse but grab a small coin."
END IF
```

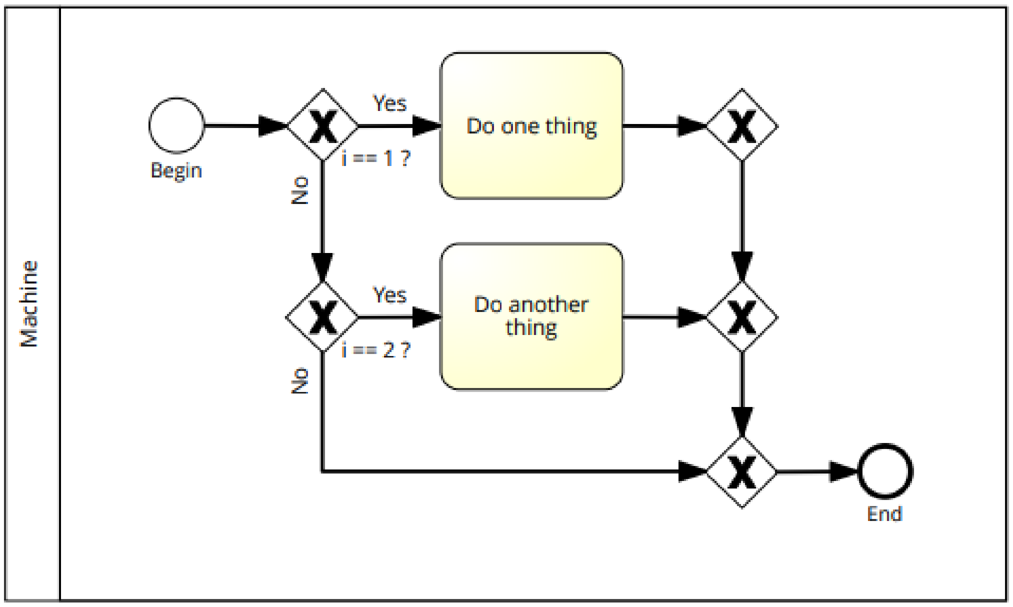

Figure 2: Single IF statements

- Source code: Run this in onecompiler.com/c/43bteavtn

```
int health = 10; // SET health = 60

if (health > 75)  // IF health > 75 THEN
  puts("You're in top form! You find a golden shield.");
 else if (health > 50)  // ELSE IF health > 50 THEN
   puts("You're scratched but standing. You find a rusty sword.");
 else if (health > 25) // ELSE IF health > 25 THEN
   puts("You're wounded but alive. You find a healing potion.");
 else
   puts("You collapse but grab a small coin.");
```

```
You collapse but grab a small coin.
```

3.3 Practice: Loot by Difficulty

- **Task:** Write pseudocode for loot based on enemy difficulty (1-10). Use a stacked `if` structure.

- **Starter Code:**

```
SET enemy_difficulty = 6
PRINT "You defeat a monster!"

// Add stacked if structure here
```

- **Guidance:**
  - $\text{enemy}_{\text{difficulty}}$ 8-10: print "You find a gold coin"
  - $\text{enemy}_{\text{difficulty}}$ 5-7: print "You find an iron ring"
  - $\text{enemy}_{\text{difficulty}}$ 1-4: print "You find a stick"
  - $\text{enemy}_{\text{difficulty}}$ not a number > 0: print "Something went wrong - nothing found."
- **Example Output:** "You defeat a monster! You find an iron ring!"
- **Time:** 10 minutes to write and test.
- **Sample solution:** `https://onecompiler.com/c/43bsst2mg`

```
SET enemy_difficulty = 6
PRINT "You defeat a monster!"

IF enemy_difficulty >= 8
   PRINT "You find a gold coin!"
ELSE IF enemy_difficulty >= 5
   PRINT "You find an iron ring!"
ELSE IF enemy_difficulty >= 1
   PRINT "You find a stick!"
ELSE
   PRINT "Something went wrong - nothing found."
END IF
```

- **Sample source code:**
  onecompiler.com/c/43brbdw4f

```
int enemy_difficulty = 6; // SET enemy_difficulty = 6
printf("You defeat a monster! "); // PRINT "You defeat a monster!"

if (enemy_difficulty >= 8) // 8-10: "gold coin"
  printf("You find a gold coin!");
 else if (enemy_difficulty >= 5) // 5-7: "iron ring"
   printf("You find an iron ring!");
 else if (enemy_difficulty >= 1) // 1-4: "stick"
   printf("You find a stick!");
 else
   puts("Something went wrong - nothing found.");
```

```
You defeat a monster! You find an iron ring!
```

4 Nested `if` Structures

4.1 Explanation

- Nested `if` structures place one or more `if` statements inside another `if` statement, creating layered conditions. The inner `if` is only evaluated if the outer condition is true, allowing for decisions that depend on multiple criteria.

4.2 Example

- Game Context: Opening a vault requires a key AND enough strength.
- Example:

```
SET has_key = true
SET strength = 40

IF has_key THEN
    IF strength >= 50 THEN
        PRINT "You unlock and open the vault!"
    ELSE
        PRINT "You unlock it but can't open it."
    END IF
ELSE
    PRINT "You need a key."
END IF
```

- Key Point: Inner condition (`strength`) depends on outer condition (`has_key`) being true.
- Source code: onecompiler.com/c/43brdaaj4

```
int has_key = 1; // SET has_key = true
int strength = 40; // SET strength = 40

if (has_key) {// IF has_key THEN
  if (strength >= 50) // IF strength >= 50 THEN
    puts("You unlock and open the vault!");
  else
    puts("You unlock it but can't open it.");
 } else {
  puts("You need a key.");
 }
```

```
You unlock it but can't open it.
```

4.3 Practice: Boss Fight (Bonus Assignment)

- **Task:** Write pseudocode for a boss fight. Success requires a sword AND high skill. Use a nested `if` structure. Write C source code and test it.
- **Starter Code:**

```
SET has_sword = true
SET skill = 70
PRINT "You face the boss!"

// Add nested if structure here
```

- **Guidance:**
  - Sword + skill 80: "You win!"
  - Else: "You lose."
- **Example Output:** "You face the boss! You lose!"

5 Example with stacked and nested conditional statements

- Game Context: After defeating an enemy, the player's reward depends on their health. This includes both stacked and nested statements.
- Pseudocode:

```
SET health = 60
SET enemy_defeated = true

IF enemy_defeated THEN
    IF health > 75 THEN
        PRINT "You're in top form! You find a golden shield."
    ELSE IF health > 50 THEN
        PRINT "You're scratched but standing. You find a rusty sword."
    ELSE IF health > 25 THEN
        PRINT "You're wounded but alive. You find a healing potion."
    ELSE
        PRINT "You collapse but grab a small coin."
    END IF
ELSE
    PRINT "You flee with nothing."
END IF
```

- Source code: Run this in onecompiler.com/c/43br7x7b4

```
int health = 10; // SET health = 60
int enemy_defeated = 1; // SET enemy_defeated = 1 (true))

if (enemy_defeated) { // IF enemy_defeated THEN
  if (health > 75)  // IF health > 75 THEN
    puts("You're in top form! You find a golden shield.");
  else if (health > 50)  // ELSE IF health > 50 THEN
    puts("You're scratched but standing. You find a rusty sword.");
  else if (health > 25) // ELSE IF health > 25 THEN
    puts("You're wounded but alive. You find a healing potion.");
  else
    puts("You collapse but grab a small coin.");
 } else {
  puts("You flee with nothing.");
 }
```

```
You collapse but grab a small coin.
```

6 What about curly brackets?

- When should you use curly brackets after the `if`, `else`, and `else if` statements?

  > Curly brackets define a **block of code** that belongs to that `if`, `else if` or `else` condition. You should always use them because they make debugging (and writing code) vastly easier!
  >
  > They are **mandatory** when the block contains more than one statement.
  >
  > They are **optional** when the block contains only one statement.

- Code example:

```
int health = 10;

if (health > 75) {
  puts("You're in top form! You find a golden shield.");
 } else if (health > 50) {
  puts("You're scratched but standing. You find a rusty sword.");
 } else if (health > 25) {
  puts("You're wounded but alive. You find a healing potion.");
 } else {
  puts("You collapse but grab a small coin.");
 }
```

```
You collapse but grab a small coin.
```

```
You collapse but grab a small coin.
```

7 Conclusion

- **Summary:**
  - **Sequential:** Independent checks that can all execute if true (e.g., buy a sword AND drink a potion based on gold and health).
  - **Stacked:** Mutually exclusive choices in a sequence (e.g., one reward based on health after a battle).
  - **Nested:** Layered conditions where inner checks depend on outer ones (e.g., needing a key AND strength to open a vault).
- **Next:** Switch control statement, and loops for repeated actions in the game (e.g., fighting multiple enemies).