**Lecture 4 - Chapter 3: Structured Program Development in C– Wed Aug 30 or Thurs Aug 31**

**Announcements**

Reading:

* Chapter 3

Assignments:

* Due: Assignment #1
* Assign: Assignment #2 (If Statements) – due on **Wed Sept 6** (MW class) or **Thurs Sept 7** (T/R class)
* No late assignments accepted

Tutor Schedule:

Monday:

             9:30am – 12pm

Tuesday:

             12:30 – 4:30

Wednesday:

             9:30 – 12:00 pm

Thursday:

             12:30 – 4:30

Friday:

            None

**Today’s Goals**

1. Algorithms and Pseudocode
2. Control Structures
3. If Statements
4. Examples from book

**Today’s Terminology**

**Terminology**

* Algorithm
  + An ordered set of steps to solve a problem
  + For example, the steps to sort a list of numbers from ascending to descending
* Pseudocode
  + English mixed with programming code
  + You represent an algorithm in pseudocode
* Flow of Control
  + The order in which statements are executed in a program
* Selection Statement
  + Way to make a decision
  + If statements
  + Switch statements
* Iteration Statement
  + Way to repeatedly execute code
  + While loops
  + Do-while loops
  + For loops
* Flowchart
  + Graphical representation of code flow (algorithm)
* Equality Operators
  + == Equal
  + != Not equal
* Relational operators
  + < Less than
  + <= Less than or equal
  + > Greater than
  + >= Greater than or equal
* Boolean expression
  + An expression that evaluates to true or false

**Algorithms and Pseudocode**

**Algorithm vs Pseudocode**

* Algorithm is the set of steps needed to solve a problem
* Pseudocode is the language to represent the algorithm

**Why Pseudocode?**

* Working with pseudocode first, allows you to focus on the algorithm without being distracted by the details of a programming language.
* Forces you to sit down and **think about the problem** you are trying to solve before you jump on the computer and start coding.

Pseudocode examples will be presented in next lecture (lecture 5)

**Control Structures**

Three control structures in programming

* Sequence
  + Statements are executed one after another
* Selection
  + Based on a condition, one direction or another is executed
  + Provided by **if-statements**, **if-else-statements**, **switch statements**
* Iteration
  + Statements are executed over and over
  + Provided by **while loops**, **do-while loops**, **for loops**

Chapter 3 - if statements, while loop

Chapter 4 - switch statement, do-while, for loops

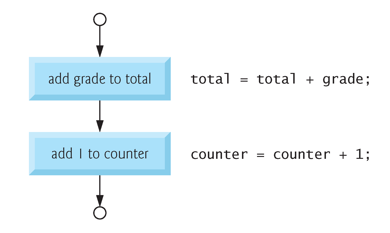
**Selection Statements**

**Purpose**

* Allows different paths within code to be taken based on a condition

**Flowcharts**

* Graphical representation of code flow
* Different symbols are used to represent different things
  + Rectangle - Statements
  + Diamond - Decisions
  + Parallelogram - Input/output
  + Flow line - Flow of control



**Types of Selection Statements**

* If statements
* Switch Statement

**If Statements**

**Purpose**

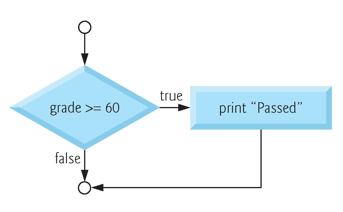
* Provides ability to select a path of execution based on a condition
* “If statements” implement conditional execution - alters flow of control
  + If it is snowing today, then I'm not going to class

**Condition**

* Create a condition using **equality** and **relational operators**
* Conditions evaluate to true if the condition evaluates to **non-zero**
* Conditions evaluate to false if the condition evaluates to **zero**

**How it Works**

* A condition is evaluated (i.e. boolean expression)
* If the condition is true
  + Statements in "then" part are executed
* If the condition is false
  + The statements in "then" part are not executed and
  + The statements in "else" part are executed (if there is an "else" part)



**General Form**

* **Single-Selection Statement (One-Way If)**
  + Executes statement(s) in if-block if the condition is true otherwise skips statements in if-block

if (boolean expression) {

statement(s); //Execute these statements if condition is true

}

if (numOfStudents <= 30) {

puts ("Class is available - register student for class");

}

* **Double-Selection Statement (Two-Way If)**
  + Executes statement (s) in "then" block if condition evaluates to true or

executes statement(s) in "else" block if condition evaluates to false

if (boolean expression) {

statements(s); // Execute these statements if condition is true

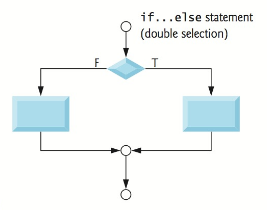
}

else {

statement(s); // Execute these statements if condition is false

}

* + Flowchar**t**



* + **Example #1**

if (numOfStudents <= 30)

puts ("Class is available - register student for class");

else

puts ("Class is full - place student on waitlist");

**Note on style:**

* + - **There are no curly braces for the then-part or the else-part in Example #1**
    - **Recommendation is to use curly braces and write the above code as in Example #2**
  + **Example #2**

if (numOfStudents <= 30) {

puts ("Class is available - register student for class");

}

else {

puts ("Class is full - place student on waitlist");

}

* **Nested If Statements**
  + Placing if-statements inside if-statements

if (boolean expression) {

if (boolean expression) { Always indent code in the

statement(s); the body of an if-statement

} // end inner if

} // end outer if

* + **Example #1**

if (numOfStudents <= 30) {

puts ("Class is available - register student for class");

if (studentType == MATH\_STUDENT) {

puts ("Placing student in math group");

}

else {

puts ("Placing student in non-math group");

}

} // end numberOfStudents <= 30

else **{**

puts ("Class is full - place student on waitlist");

} // end else

* + **Example #2**

Note: notice how the indentation is starting to get deeper with each level of nesting

if (grade >= 90) {

puts ("You got an A");

}

else { 1st level of indentation

if (grade >= 80) {

puts ("You got a B");

}

else { 2nd level of indentation

if (grade >= 70) {

puts ("You got a C");

}

else { 3rd level of indentation

if (grade >=D)

puts ("You got a D");

}

} // end else

} // else grade < 90)

* + **Example #3 – Preferred way to write above example**

Levels of indentation is reduced to one – easier to read – less confusing

Notice that curly braces are used but are not necessary since single statements

Including the curly braces helps reduce dangling-else error

if (grade >= 90) {

puts ("You got an A"); Combined **if** & **else** keywords on one line

}

else if (grade >= 80) {

puts ("You got a B");

}

else if (grade >= 70) {

puts ("You got a C");

}

else if (grade >=D) {

puts ("You got a D");

}

else {

puts ("Take the class over");

**}**

**Trace If Statement**

* Let's trace what happens in a multi-way IF statement

int score = 70;

if (score >= 90) { // This condition is false

puts ("You got an A");

}

else if (score >= 80) { // This condition is evaluated next - but it is false

puts ("You got a B");

}

else if (score >= 70) { // Next this condition is evaluated - it is true

puts ("You got a C"); // This statement is executed and control drops out of if

}

else if (score >= 60) {

puts ("You got a D");

}

else {

puts ("Take the class over");

}

puts ("Good Bye"); // This statement is always executed

**Notes:**

* Note on style:
  + Block statements can be omitted if there is **only one** statement in the "then part or "else" part
  + **Recommendation - use curly braces - it helps make the code easier to modify - less error prone**
  + Indent the body statements for if-statements to increase readability (in code and pseudocode)
  + Avoid deeply nested if statements
  + An "else" clause always belongs with the most recent if clause

**Conditional Expressions**

**Purpose**

* Shortcut way to write a two-way if statement (if-then-else)
* Consists of the symbols **?** and **:**
* Also called the "ternary" operator - takes 3 operands (ternary operator)
  + 1st – condition
  + 2nd – value if condition is true
  + 3rd – value if condition is false

**General Form**

* Expression can be either a boolean value or a statement that evaluates to a boolean value

result = expression ? value1 : value2;

**How it Works**

* The conditional "expression" is evaluated
* If the expression is true, value1 is returned
* If the expression is false, value2 is returned

**Example #1**

if (a < 0) {

absValue = -a;

}

else {

absValue = a;

}

Can be re-written as:

absValue = (a < 0) ? -a : a;

**Example #2**

**puts** (numOfStudents <= 30 ? "Available" : "Waitlist");

**Common Errors and Pitfalls**

**Logic errors**

* Error that causes your program to produce the wrong result.
* If-statements are an area where logic errors are easy to create!

**Forgetting Necessary Braces**

* Causes flow of control to be incorrect - leads to logical errors
* Indentation does not dictate flow of control – its only for readability

if (numStudents <= 30)

puts ("Class is available - register student for class");

else

puts (“Class is not available”);

puts (“You will be placed on the waitlist”);

* Incorrect output if numStudents = 15

Class is available - register student for class

You will be placed on the waitlist

* To fix, add in parentheses

if (numStudents <= 30) { Not necessary but good practice

puts ("Class is available - register student for class");

}

else { These are necessary

puts (“Class is not available”); to prevent logic error

puts (“You will be placed on the waitlist”);

}

**Wrong Placement of Semicolon**

* Causes flow of control to be incorrect - leads to logical error
* Don't place a semicolon at the end of **if (boolean expression)**
* If you say **if (boolean expression);**

if (numStudents <= 30); {

puts ("Class is available - register student for class");

}

* + Telling the complier that there is **nothing to do** in the "then-part", same as

if (boolean expression) {

// do nothing

}

* + Can be sneaky error since hard to see that you've placed the **";"** on the line
  + The curly braces don't do anything except define a block so the following code

if (numStudents <= 30); { // Unwanted semicolon!!!

puts ("Class is available - register student for class");

}

* + Is equivalent to

if (numStudents <= 30) **{**

// No statements so do nothing

**}**

puts ("Class is available - register student for class");

**Using assignment and not equality**

* Common issue!!!
* Complier warning – possible assignment in condition
* This is an assignment statement not a boolean expression

**int** mathGrade = 80;

**if** (mathGrade = 90) {

**puts** ("Assignment vs Equality");

} // end if

**printf** ("Match grade = %d\n", mathGrade);

* This code
  + DID NOT test to see if the math grade was 90, it changed it to 90!
  + Always displays print statement!

**Redundant Condition**

* In some cases, not an error situation - just poor programming

bool snowing = false;

if (snowing == true) { // Remember this is a boolean expression

puts ("It is snowing!");

}

else {

puts ("It is not snowing");

}

* Correct way to write above code

bool snowing = false;

if (snowing) { // **snowing** is a boolean and boolean expression is correctly written

}

else {

}

* But in next example we have a logic error
  + In this case, boolean expression contains an assignment statement so it assigns then evaluated

bool snowing = false; // This is an assignment statement

if (snowing = true) { // Assignment statement again - assigning new value

puts ("It is snowing");

}

else {

puts ("It is not snowing");

}

Snowing has a different value (true) going forward!

**Dangling else**

* Happens with nested Ifs
* When there are multiple "else" clauses and it's not clear which "if" the "else" belongs to
* Indentation is not an indication which "if" the "else" belongs to

**Example with bad indentation:**

bool snowing = true;

bool roadsIcy = true;

bool stayHome;

if (snowing)

If (roadsIcy)

stayHome = true;

else

stayHome = false;

* You might think the **else** belongs to the if (snowing) but it belongs to the if (roadsIcy)
* Better - proper indentation

if (snowing)

if (roadsIcy)

stayHome = true;

else

stayHome = false;

* Even better - use curly braces

if (snowing) {

if (roadsIcy) {

stayHome = true;

}

else {

stayHome = false;

}

}

**Equality testing of floats**

* Floating point numbers are numbers with decimal points, like 34.5
* Cannot reliably test **equality** of two floating-point numbers
* Do **not** perform equality checking with floating point numbers
* You can only test if the difference is within a threshold
* More in the next lecture – lecture 5

**Class Time for Exercises:**

**Go over example on p.76**