CSCE 155 - C

Lab 3.0 - Conditionals

Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.
2. Read the required chapters(s) of the textbook as outlined in the course schedule.

Peer Programming Pair-Up

For students in the online section: you may complete the lab on your own if you wish or you may team up with a partner of your choosing, or, you may consult with a lab instructor to get teamed up online (via Zoom).

For students in the face-to-face section: your lab instructor will team you up with a partner.

To encourage collaboration and a team environment, labs are be structured in a *peer programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- How basic control flow works
- When and how to use if, if-then-else, and if-else-if conditional statements in a program

2 Background

The default control flow in a typical program is sequential. That is, each statement is executed one after the other. However, we often have to make decisions based on program state or variable values. This is known as conditional control flow. C, like many programming languages, provides several control structures to alter the flow of control in a program based on the truth-value of some conditional statement.

An if-statement can be used to conditionally execute a block of code. If the condition in an if-statement evaluates to true the code block is executed, otherwise it is not.

```
1 if(x > 0) {
2   printf("x is positive!\n");
3   printf("the value of sqrt(x) = %f\n", sqrt(x));
4 }
```

An if-then-else statement can be used to execute one of two mutually exclusive code blocks. If the condition evaluates to true, then the first block is executed. Otherwise, if the condition is false, the second code block is executed.

```
1 if(x > 0) {
2   printf("x is positive!\n");
3   printf("the value of sqrt(x) = %f\n", sqrt(x));
4 } else {
5   printf("x is not positive, cannot compute its square root\n");
6 }
```

An if-else-if statement can be used to execute one of several mutually exclusive state-

ments. It also requires more than one conditional statement to determine which code block should be executed. The control flow of an if-else-if statement is such that the first condition that evaluates to true is executed. All subsequent statements are skipped. If no condition holds true, then the else code block is executed. However, much like the difference between an if-statement and an if-then-else statement, the final else block is optional.

```
if(x > 0) {
  printf("x is positive!\n");
  printf("the value of sqrt(x) = %f\n", sqrt(x));
} else if(x == 0) {
  printf("x is zero, its square root is zero\n");
} else {
  printf("x is not positive, its square root is complex:\n");
  printf("%f\n", sqrt(x*-1.0));
}
```

Yet another type of conditional control statement is a switch statement. In a switch statement, a single variable's value is used to determine which, among several provided cases gets executed.

```
switch(classLevel) {
  case 1:
    printf("Freshman");
    break;
  case 2:
    printf("Sophomore");
    break;
  case 3:
    printf("Junior");
    break;
  case 4:
    printf("Senior");
    break;
  default:
    printf("ERROR: Unknown value!");
    break;
}
```

In C, the switch-statement can only be used on integer variables (or `char` variables since they are essentially integers). Moreover, the control flow can easily lead to unexpected results if you forget to include the `break` statement at the end of each case.

Operator	C Syntax	Example
Negation	`!`	`!(x > 0)`
Logical And	`&&`	`(x != 0) && (y < 10)`
Logical Or	`||`	`(x < 0) || (x >= 10)`

Table 1: Logical Operators in C

2.1 Compound Statements

Many logical statements require the use of logical operators that can be used to form more complex, compound statements. The three logical operators that we'll focus on are as follows.

- Negation: this is a unary operator that flips the truth value of the statement or variable that it is applied to so that true becomes false and vice versa.
- Logical And: this is a binary operator that is applied to two logical expressions and evaluates to true if and only if both expressions are true
- Logical Or: this is a binary operator that is applied to two logical expressions and evaluates to true if at least one (or both) of the expressions are true

Logical operators are necessary to do more complex statements such as checking for ranges of variable values. For example, to check if a variable `x` lies in the range $[0, 10]$, one would need to use a logical and operator:

```
1 if( x >= 0 && x <= 10 ) {
2   //code
3 }
```

3 Activities

We have provided partially completed programs for each of the following activities. You will need to clone the Lab 03 project from Github using the URL: https://github.com/cbourke/CSCE155-C-Lab03. Refer to previous labs for a step-by-step process.

3.1 Tax Program

The federal income tax for a married couple filing jointly for 2012 is determined by the rules indicated in Table 2. In addition, the total tax is reduced by $1000 for each child that a couple has.

If the AGI is over-	But not over-	The tax is:	Of the amount over-
\$0	\$17,000	10%	\$0
\$17,000	\$69,000	\$1,700 + 15%	\$17,000
\$69,000	\$139,350	\$9,500 + 25%	\$69,000
\$139,350	\$212,300	\$27,087.50 + 28%	\$139,350
\$212,300	\$379,150	\$47,513.50 + 33%	\$212,300
\$379,150	–	\$102,574.00 + 35%	\$379,150

Table 2: 2012 Tax Brackets

AGI	Number of Kids	Taxes
\$4,000	1	\$0
\$20,000	0	\$2,150
\$120,000	3	\$19,250
\$150,000	4	\$26,069.50
\$250,000	0	\$59,954.50
\$500,000	5	\$139,871.50

Table 3: Several example inputs/outputs.

We have provided a partially completed program that reads in a user's Adjusted Gross Income (AGI) and prompts whether or not they have any children. If they do it prompts them for how many. Complete the program by computing the user's total tax based on the user's AGI the number of children they have.

Some example input/output results can be found in Table 3. For example, if the user has an adjusted gross income of \$150,000 and has 4 children, then their tax would be calculated as follows.

- Their AGI falls in the 4th tax bracket and so would be

$$\$27,087.50 + 28\% \times (\$150,000 - \$139,350) = \$30,069.50$$

- With 4 children, they have a \$4,000 tax credit giving a new total of \$26,069.50

Answer the questions on your worksheet (see `worksheet.md`).

3.2 Calculator

In this activity, you will implement a simple menu-based command line calculator. A partially completed program has been provided to you, `calculator.c`. The program prompts the user for two operands and one of several different choices for an operation to be performed on them. Your program should process the input and display the result of the chosen operation. Take care with the following possibilities:

- For division, you should check if $b = 0$ (division by zero is not defined). If it is, an appropriate error message should be output instead.
- For the logarithm operation, you should use the math library's `log` function, which is the natural logarithm (base e). To change to another base, use the formula:

$$\log_a(b) = \frac{\ln(b)}{\ln(a)}$$

  In addition, you should check that both operands are positive, if not then output an appropriate error message.

Reminder: when compiling on CSE you may need to direct `gcc` that it needs to *link* the math library by giving it the `-lm` (**l**ink **m**ath) flag; for example:

```
gcc -lm calculator.c
```

Complete the program and answer the questions on your worksheet.

3.3 Leap Years

Nearly every 4 years is a *leap year* in the Gregorian calendar. In a leap year, there are 366 days (adding February 29th) instead of the usual 365. Specifically a year y is a leap year if it is divisible by 4. However, every year that is divisible by 100 is not a leap year unless it is divisible by 400. Thus, 2000, 2004, 2008 were leaps years but 2001, 2002, 2003 were not. 1900 was not a leap year; though it was divisible by 4 it was divisible by 100 but not 400.

We've provided a partially completed program, `leapYear.c` that tests whether or not various years are leap years.

1. Implement a conditional statement inside the `isLeapYear()` function to determine if the given `year` is a leap year or not. Return true (`1`) if it is, false (`0`) if it is not.
2. Compile and run your program: we've provided 3 hard-coded test cases. Fix any errors in your program until they all *pass*.
3. Using the provided code as an example, add at least 3 more test cases to your program. Repeat your compile/run/test until they all pass.

4 Handin/Grader Instructions

1. Hand in your completed files:
   - `taxes.c`

- `calculator.c`
- `leapYear.c`
- `worksheet.md`

through the webhandin (https://cse-apps.unl.edu/handin) using your cse login and password.

2. Even if you worked with a partner, you *both* should turn in all files.
3. Verify your program by grading yourself through the webgrader (https://cse.unl.edu/~cse155e/grade/) using the same credentials.
4. Recall that both expected output and your program's output will be displayed. The formatting may differ slightly which is fine. As long as your program successfully compiles, runs and outputs the *same values*, it is considered correct.

5 Advanced Activities (Optional)

1. Another conditional operator is the ternary if-then-else operator. It is often used to choose between two values. For example:
`int min = ( (a < b) ? a : b );` The syntax, `X ? Y : Z` is as follows: `X` is any conditional statement; if it evaluates to true, then the expression takes on the value `Y`; otherwise it takes on the value `Z`. Modify your programs to use this ternary operator where appropriate.
2. Change the calculator program as follows: add a menu option so that the user has the option to quit; then add a loop so that the program continues to print the menu. As long as the user performs an operation, it should continue until the user selects the quit option.