# cc-practice-ops

## Table of Contents

## 1. READ README

- This file is a practice file for logical and compound operators
- Time: approx. 30 min.
- When you're done with a section move the cursor on the section heading and type `S-<right>` (or SHIFT+<right-arrow>).
- This section follows chapter 3 in Davenport/Vine (2015) and chapters 4 and 5 in King (2008).

## 2. TODO Identify yourself

- replace the placeholder `[yourName]` in the header of this file by your name and save the file (`C-x C-s`).

## 3. TODO Logical operators &&, ||, !

1. Complete the `printf` statement in each of the following code blocks.
2. Before you run the code block, guess what the output will be.

   - Check if (NOT `i`) is smaller than `j`, for `i=10` and `j=5`.

     The output should be `1` (TRUE).

     ```
     int i = 10, j = 5;
     printf("%d\n", !i < j); // !10 is 0, and 5 > 0 is TRUE (1)
     ```

     ```
     1
     ```

   - Check the value of NOT(NOT (i)) + NOT(j), for `i=2` and `j=1`.

     ```
     int i = 2, j = 1;
     printf("%d\n", !!i + !j); // !!2 = !0 = 1, !1 = 0, 1 + 0 = 1
     ```

     ```
     1
     ```

   - Using the previous code block 1, check if the following holds: NOT(x + y) = NOT(x) + NOT(y).

```
int i = 2, j = 1;
printf("%d\n", !(!i + j)); // !(!2 + 1) = !!2 + !1 = !(0 + 1) = !1 = 0
```

```
0
```

- Compute `i` AND `j` OR `k`, for `i=5`, `j=0`, `k=-5`.

```
int i = 5, j = 0, k = -5;
printf("%d\n",  i && j || k); // 5 && 0 = 0, 0 || 1 = 1
```

```
1
```

- Compute `i < j` OR `k`, for `i=1`, `j=2`, `k=3`.

```
int i = 1, j = 2, k = 3;
printf("%d\n",  i < j || k); // (i < j) = 1, 3 is TRUE, 1 || 1 is 1
```

```
1
```

## 4. TODO Checking input for upper and lower case

1. Create an input file `ascii` with the letter `b` in it, and check that the file contains the letter.

```
echo 'b' > ./src/ascii
cat ./src/ascii
```

2. Run the code 1 below. Complete the condition for the `IF` statement to check if the input character `letter` is an `B`. When you run the program, you should see that the input is not recognized.

```
char letter;
scanf("%c", &letter);

if ( letter == 'B' )
  printf("Okay! Input %c recognized as 'b' or 'B'.\n", letter);
 else
   printf("Not okay! Input %c not recognized as 'b' or 'B'.\n", letter);
```

```
Not okay! Input b not recognized as 'b' or 'B'.
```

3. Change the code from 1 in 1 so that the input `b` **or** `B` are both recognized. If you want, you can change the input by changing and running the code block 1 above.

```
char letter;
scanf("%c", &letter);

if ( letter == 'B' || letter == 'b' )
```

```
  printf("Okay! Input %c recognized as 'b' or 'B'.\n", letter);
 else
   printf("Not okay! Input %c not recognized as 'b' or 'B'.\n", letter);
```

```
Okay! Input b recognized as 'b' or 'B'.
```

4. What is the ASCII code of the letters `b` and `B`? Write a short program to print out both the character and the ASCII integer value.

   Inputfile:

```
echo 'b B' > ./src/ascii2
cat ./src/ascii2
```

```
 char b, B;
 scanf("%c %c", &b, &B);
printf("The ASCII value of %c is %d\n", b, b);
printf("The ASCII value of %c is %d\n", B, B);
```

```
The ASCII value of b is 98
The ASCII value of B is 66
```

## 5. TODO Checking for a range of values

1. Run the code block 1 below. It creates a file `num` that contains the number 5.

```
echo "5 0 10" > ./src/num
cat ./src/num
```

2. Replace the condition in the code block 1 to check if the input value 5 for `i` is in the interval [m,n) = [0,10).

```
int i, m, n;
scanf("%d %d %d", &i, &m, &n);

if ( i >= m && i < n) {
  printf("%d is in the interval [%d,%d)\n", i, m, n);
 } else {
  printf("%d is NOT in the interval [%d,%d)\n", i, m, n);
 }
```

3. [ ]

   Run 1 for different input values in 1:

| | | |
|---|---|---|
| i = -5 | m = 0 | n = 10 |
| i = 11 | m = 0 | n = 10 |
| i = 0 | m = 0 | n = 10 |

i = 10 m = 0 n = 10

Remember that you have to run 1 with the new values if you want to change the input file.

4. [ ]

   How would you have to change the condition to check if the input variable `i` is OUTSIDE of [m,n)?

   - Change the input values in 1 back to 5 0 10
   - Modify the code in 1 below to test if 5 is outside of the interval [0,10) and run it.

```
int i, m, n;
scanf("%d %d %d", &i, &m, &n);

if ( i < m || i >= n) {
  printf("%d is NOT in the interval [%d,%d)\n", i, m, n);
 } else {
  printf("%d is in the interval [%d,%d)\n", i, m, n);
 }
```

## 6. TODO Caveat: i < j < k

1. In C, the expression `i < j < k` is perfectly legal but it does NOT check if `j` is between `i` and `k`.
2. The relational operator `<` is evaluated from the left. First the Boolean value of `i < j` is computed. It is either 0 or 1.
3. Next, the check `0 < k` or `1 < k` is performed. The following example shows how this can go wrong. Run it for illustration.

```
int i = 5, j = 1, k = 100;
if (i < j < k) {
  printf("TRUE: %d < %d < %d\n", i, j, k);
 } else {
  printf("NOT TRUE: %d < %d < %d\n", i, j, k);
 }
```

```
TRUE: 5 < 1 < 100
```

4. [ ]

   Fix the the code 1 so that the output is correct. Test it for different values of i, j, k.

```
int i = 5, j = 1, k = 100;
if ( i < j && j < k ) {
  printf("TRUE: %d < %d < %d\n", i, j, k);
 } else {
  printf("NOT TRUE: %d < %d < %d\n", i, j, k);
 }
```

```
NOT TRUE: 5 < 1 < 100
```

## 7. References

- Davenport/Vine (2015) C Programming for the Absolute Beginner (3ed). Cengage Learning.
- 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

Author: [yourName] (pledged)

Created: 2022-06-13 Mon 12:00