C OPERATORS - PRACTICE

Marcus Birkenkrahe (pledged)

March 18, 2024

Contents

1 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 Identify yourself

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

3 Logical operators &&, ||, !

- Complete the `printf` statement in each of the following code blocks.

  Example: check if `i` is equal `j` for `i=1` and `j=0`:

  Problem:

```
int i=1, j=0; // declare and initialize variables
printf("%d\n", ???);  // check identity with logical operator ==
```

  Solution:

```
int i=1, j=0; // declare and initialize variables
printf("%d\n", i == j); // 1 = TRUE == FALSE = 0 => FALSE = 0
```

```
0
```

- Before you run the code block, guess what the output will be.

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

```
int i = 10, j = 5;
printf("%d\n", !i < j );
```

```
1
```

  2. 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 );
```

```
1
```

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

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

```
1
```

  4. 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 );
```

```
1
```

5. 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 );
```

```
1
```

4 Checking input for upper and lower case

1. In the shell (`M-x eshell`) or in the code block below, create an input file named `ascii` with the letter `b` in it, and check that the file contains the letter.

```
echo 'b' > ascii
cat ascii
```

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

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

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

```
Input b not recognized as 'b' or 'B'.
```

3. Change the code in the block below so that the input `b` **or** `B` are **both** recognized.

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

if ( letter == 'b' || letter == 'B')
  printf("Input %c recognized as 'b' or 'B'.\n", letter);
 else
   printf("Input %c not recognized as 'b' or 'B'.\n", letter);

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. Put both the lower and the upper case letter into the input file `ascii2`.

   Important: when using `%c` with `scanf`, the empty space is accepted as a character with the ASCII value 32.

   Inputfile (or create this on the shell):

```
echo 'b B' > ascii2
cat ascii2

char c1, c2;
scanf("%c %c", &c1, &c2 );  // accept b and B as input (with space)
printf("%c has ASCII code %i\n", c1,c1);   // print ASCII value of b
printf("%c has ASCII code %i\n", c2,c2);   // print ASCII value of B

b has ASCII code 98
B has ASCII code 66
```

5 Checking for a range of values

1. On the shell, create a file `num` that contains the number 5.

```
echo "10 0 10" > num
cat num
```

2. Define the condition in the code block below 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 ( (m <= i) && (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);
 }
```

```
10 is NOT in the interval [0,10)
```

3. Run **??** for different input values:

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 change the input file to get new input.

   Remember that you need to change the `#+name` of the code block if you want to compare output in the same Org-mode notebook.

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 the input file `num` back to 5 0 10
   - Modify the code 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);
 }
```

```
10 is NOT in the interval [0,10)
```

6 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`:

   - The relational operator `<` is evaluated from the left. First the Boolean value of `i < j` is computed. It is either 0 or 1.
   - 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
```

2. Fix the the code below 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
```