Compound if structures

CSC100 Introduction to programming in C/C++

Table of Contents

1 README

- `[X]` Work through this notebook at your own pace. When you're done, check a task off by typing `C-c C-c` on the line with the `[ ]`. Check this task off for practice! You can close bullet points with the `<TAB>` key on your keyboard
- `[X]`

  Make sure that you can run C where you are by executing the code block below (`C-c C-c`) and then saving the file (`C-x C-s`)

  ```
  puts("hello world");
  ```

- `[X]` If this does not lead to the output `hello world`, try to analyze the problem by yourself first. Typical sources of errors:
  - Can you write to the directory?
  - Is this file an Org-file?
  - Did you use the correct key sequence?
  - Do you have the right code block header arguments?
- This section follows chapter 3 in Davenport/Vine (2015) and chapters 4 and 5 in King (2008).

2 Logical operators &&, ||, !

Show the output produced by each of the following expressions with the given values - calculate the result in your head before you run the code!

- `[ ]`

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

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

  ```
  1
  ```

- `[ ]`

  Check the value of `!!i + !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
```

- `[ ]`

  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
```

3 Checking for upper and lower case

- Replace the ??? in the code 1 by the letter a. Run the block and check that the file `ascii` contains the letter a.

```
echo 'a' > ascii
```

- `[ ]`

  Run the code 1 below. You see that a is not recognized as A.

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

if ( letter == 'A' )
  printf("\nOkay! Input %c recognized as a or A.\n", letter);
 else
   printf("\nNot okay! Input %c not recognized as a or A.\n", letter);
```

- [ ]

  Change the code from 1 in 1 so that the input a or A are both recognized as A! If you want, you can change the input by changing and running the code block 1 above.

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

  if ( letter == 'A' || letter == 'a' )
    printf("\nOkay! Input %c recognized as a or A.\n", letter);
   else
     printf("\nNot okay! Input %c not recognized as a or A.\n", letter);
  ```

4 Checking for a range of values

- [ ]

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

  ```
  echo "5 0 10" > num
  ```

- [ ]

  Replace the expression ??? 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);
   }
  ```

  ```
  5 is in the interval [0,10)
  ```

- [ ]

  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.

- [ ]

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);
 }
```

```
5 is in the interval [0,10)
```

5 Caveat: i < j < k

- 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
```

- [ ]

  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
```

6 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: Marcus Birkenkrahe

Created: 2022-03-26 Sat 21:30

Validate