

# **if Lesson 2 Outline**

1. `if` Lesson 2 Outline
2. Absolute Value
3. Absolute Value Definition
4. Absolute Value Implementation
5. What Does This Mean?
6. Examples of `if-else`
7. What If the Condition Fails? #1
8. What If the Condition Fails? #2
9. What If the Condition Fails? #3
10. What If the Condition Fails? #4
11. What If the Condition Fails? #5
12. Condition Fails Flowchart
13. The `else` clause #1
14. The `else` clause #2
15. The `else` clause #3
16. The `else` clause #4
17. The Meaning of `else`
18. How Many Clauses Will Execute?
19. Order of Clauses
20. Block Opens and Block Closes
21. `if-else` Flowchart
22. `if-else` Example #1
23. `if-else` Example #2
24. `if-else` Example Flowchart
25. `if-else` Indentation #1
26. `if-else` Indentation #2
27. Examples of `if-else`



# Absolute Value

Consider the function

$$a(y) = |y|$$

So we know that

$$a(-2.5) = | -2.5 \dots | = +2.5$$

$$a(-2) = | -2 \quad | = +2$$

$$a(-1) = | -1 \quad | = +1$$

$$a(0) = | 0 \quad | = 0$$

$$a(+1) = | +1 \quad | = +1$$

$$a(+2) = | +2 \quad | = +2$$

$$a(+2.5) = | +2.5 \quad | = +2.5$$

...



# Absolute Value Definition

How is  $|y|$  defined?

Well, you could always define it as  
the nonnegative square root of  $y^2$ :

$$|y| = \sqrt{y^2}$$

But here's another definition:

$$|y| = \begin{cases} -y, & \text{if } y \text{ is negative} \\ y, & \text{otherwise} \end{cases}$$



# Absolute Value Implementation

$$|y| = \begin{cases} -y, & \text{if } y \text{ is negative} \\ y, & \text{otherwise} \end{cases}$$

Here's an implementation of absolute value in C:

```
if (y < 0) {  
    absolute_value_of_y = -y;  
} /* if (y < 0) */  
else {  
    absolute_value_of_y = y;  
} /* if (y < 0)...else */
```



# What Does This Mean?

```
if (y < 0) {  
    absolute_value_of_y = -y;  
} /* if (y < 0) */  
else {  
    absolute_value_of_y = y;  
} /* if (y < 0)...else */
```

1. Evaluate the **condition** ( $y < 0$ ), which is a Boolean expression, resulting in either true (1) or false (0).
2. If the condition evaluates to true, then execute the statement inside the `if` clause.
3. Otherwise, execute the statement inside the `else` clause.



# Examples of `if-else`

```
if (a > b) {
    printf("Wow, a is greater than b!\n");
} /* if (a > b) */
else {
    printf("Loser, a is not greater than b!\n");
} /* if (a > b)...else */
```

```
if (my_height < your_height) {
    shortest_height = my_height;
} /* if (my_height < your_height) */
else {
    shortest_height = your_height;
} /* if (my_height < your_height)...else */
```



# What If the Condition Fails? #1

What if we have something that we want executed only in the event that the Boolean expression in the if condition **fails**? That is, when the condition evaluates to false (0).



# What If the Condition Fails? #2

If there's something that we want to do when the `if` condition fails, we could simply use another `if` block with the exact opposite condition:

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
if (!(users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if (!(users_number < minimum_number) || ... */
```





# What If the Condition Fails? #3

Using another `if` block with the exact opposite condition is cumbersome:

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
if (!(users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if (!(users_number < minimum_number) || ... */
```



# What If the Condition Fails? #4

Using another `if` block with the exact opposite condition is cumbersome:

(a) It increases the likelihood of bugs, since you're typing twice as much.

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
if (!(users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if (!(users_number < minimum_number) || ... */
```



# What If the Condition Fails? #5

Using another `if` block with the exact opposite condition is cumbersome:

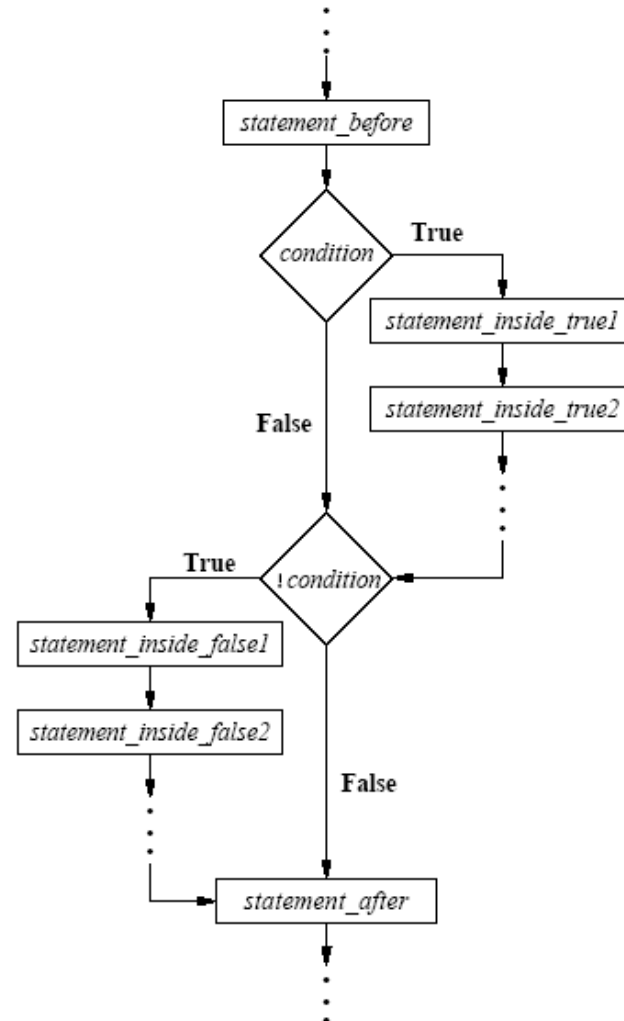
(b) If we later change the first condition, but we forget to change the second, that'll introduce a hard-to-find bug.

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
           minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
if (!(users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Woohoo! That's between %d and %d!\n",
           minimum_number, maximum_number);
} /* if (!(users_number < minimum_number) || ... */
```



# Condition Fails Flowchart

```
...  
statement_before;  
if (condition) {  
    statement_inside_true1;  
    statement_inside_true2;  
} /* if (condition) */  
if (!condition) {  
    statement_inside_false1;  
    statement_inside_false2;  
} /* if (!condition) */  
statement_after;  
...
```



# The `else` clause #1

Often, we want to have not only

- a sequence of statements to execute in the event that the `if` condition evaluates to true (1),

but **also**

- a sequence of statements to execute in the event that the `if` condition evaluates to false (0).

So, C (like most programming languages) allows us to set up a special group of statements within the `if` block, known as an ***else clause***.



# The else clause #2

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

- The sequence of statements to execute when the `if` condition evaluates to **true** (1) is known as the *if clause*.
- The sequence of statements to execute when the `if` condition evaluates to **false** (0) is known as the *else clause*.



# The else clause #3

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
        minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
        minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

**NOTICE:** The **else statement DOESN'T have a condition** of its own: it's simply the keyword `else`, with its condition implied by the `if` statement.

That is, the `else` clause's condition is the opposite of the `if` clause's condition, and is **IMPLIED** instead of stated explicitly.



# The else clause #4

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

Notice that the presence of the `else` clause guarantees that, for this `if` block, **EXACTLY ONE of the clauses will be executed.**

(For an `if` block **WITHOUT** an `else` clause, **AT MOST** one of the clauses would be executed, or none.)





# The Meaning of `else`

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

The statements inside the `if` clause are executed only in the event that the `if` condition evaluates to true (1).

The statements inside the `else` clause are executed only in the event that the `if` condition evaluates to false (0).

So, in programming, the keyword `else` means **otherwise**.



# How Many Clauses Will Execute?

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

When executing an `if` block that has **BOTH** an `if` clause and an `else` clause, **EXACTLY ONE** clause will be executed:

- either the condition will evaluate to true (1), in which case the `if` clause will execute,

**OR**

- the condition will evaluate to false (0), in which case the `else` clause will execute.



# Order of Clauses

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

Notice that the `else` clause comes **AFTER** the `if` clause. That is, **EVERY** `if` block **MUST** begin with an `if` clause. Having an `else` clause is **OPTIONAL**.

In the event that an `if` block has an `else` clause, then the `else` clause comes at the **END** of the `if` block.



# Block Opens and Block Closes

```
if ((users_number < minimum_number) ||
    (users_number > maximum_number)) {
    printf("Hey! That's not between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ... */
else {
    printf("Woohoo! That's between %d and %d!\n",
          minimum_number, maximum_number);
} /* if ((users_number < minimum_number) || ...else */
```

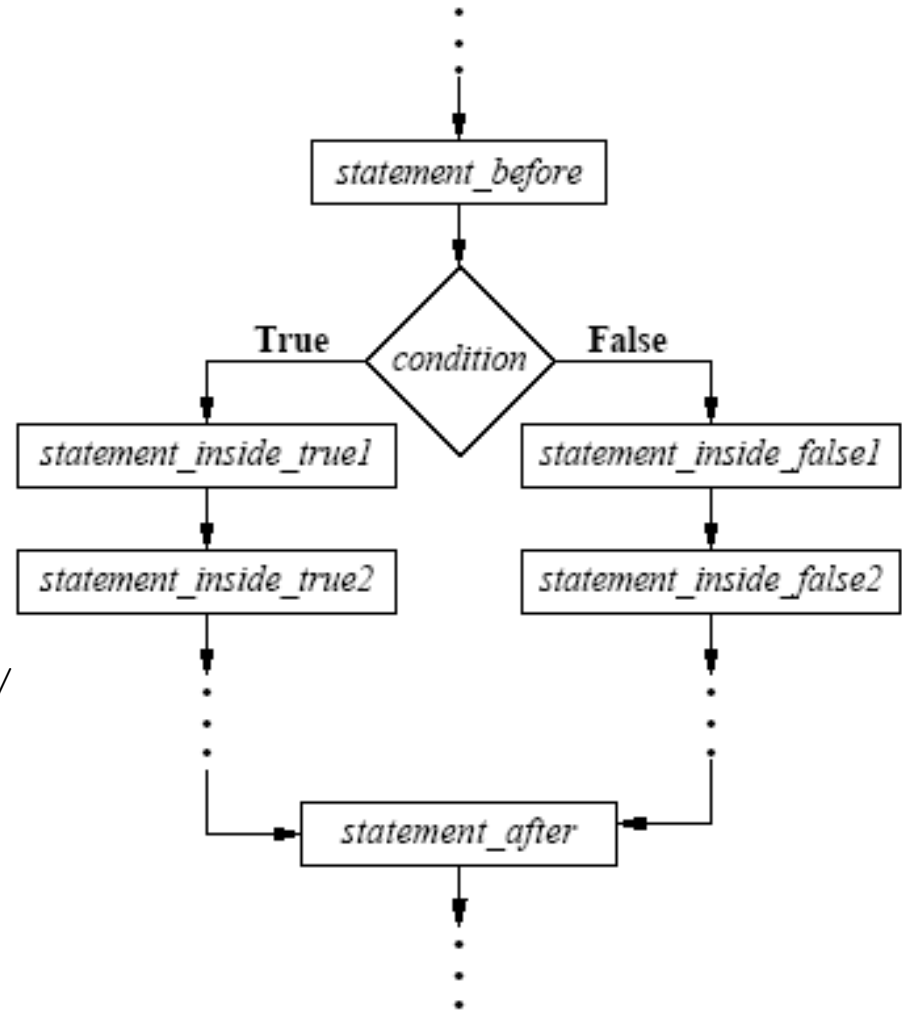
Notice that each of the clauses – the `if` clause and the `else` clause – has its own block open and its own block close.

Again, regardless of the value of the Boolean expression in the condition of the `if` statement, any statements after the last block close are always executed.



# if-else Flowchart

```
...  
statement_before;  
if (condition) {  
    statement_inside_true1;  
    statement_inside_true2;  
} /* if (condition) */  
else {  
    statement_inside_false1;  
    statement_inside_false2;  
} /* if (condition...else) */  
statement_after;  
...
```



# if-else Example #1

```
#include <stdio.h>

int main ()
{ /* main */
    const int computers_number = 5;
    int users_number;

    printf("Pick an integer:\n");
    scanf("%d", &users_number);
    if (users_number < computers_number) {
        printf("That's unbelievable! Your number is\n");
        printf("  less than mine!\n");
        printf("Well, okay, maybe it's believable.\n");
    } /* if (users_number < computers_number) */
    else {
        printf("Wow, you picked a number that isn't\n");
        printf("  less than mine. Good work!\n");
    } /* if (users_number < computers_number)...else */
    printf("And now I'm sick of you.\n");
    printf("Bye!\n");
} /* main */
```



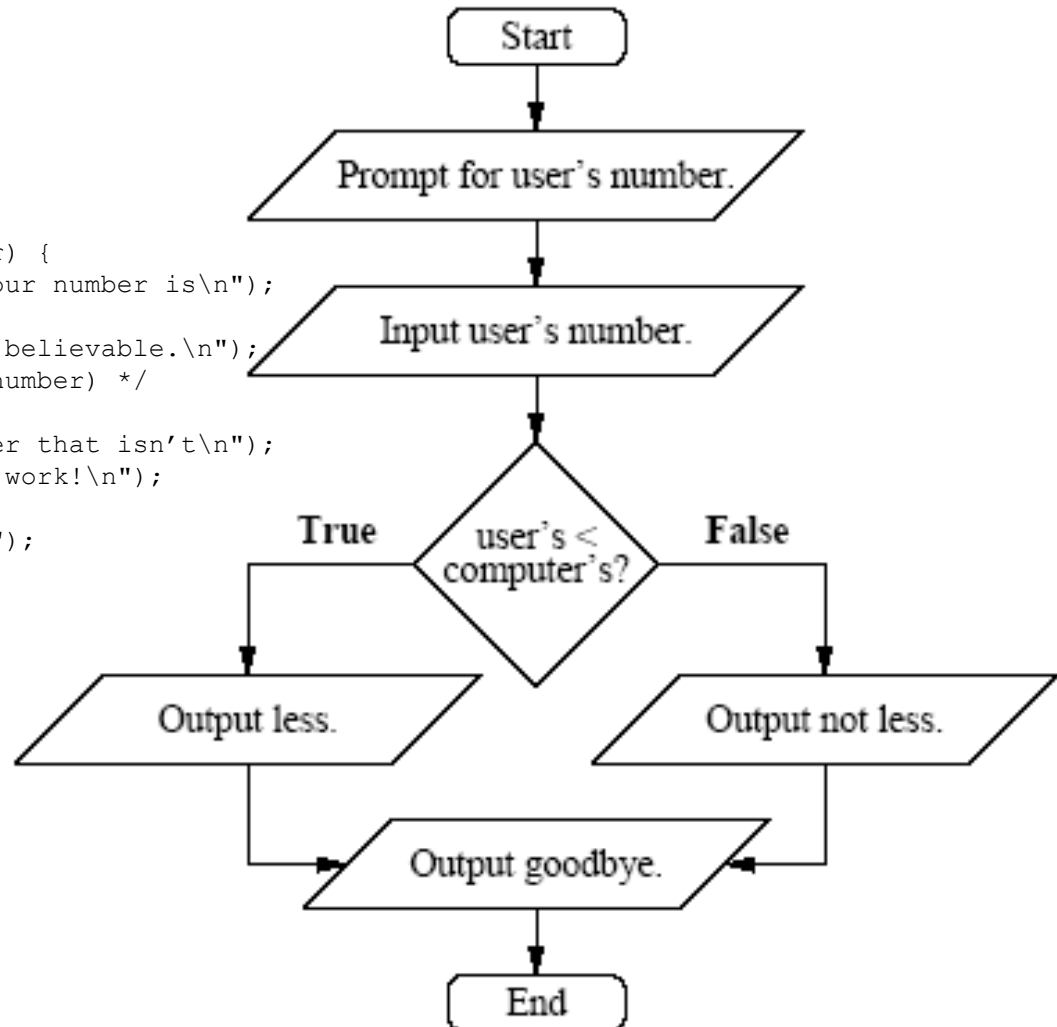
# if-else Example #2

```
% gcc -o islesselse islesselse.c
% islesselse
Pick an integer:
6
Wow, you picked a number that isn't
  less than mine. Good work!
And now I'm sick of you.
Bye!
% islesselse
Pick an integer:
5
Wow, you picked a number that isn't
  less than mine. Good work!
And now I'm sick of you.
Bye!
% islesselse
Pick an integer:
4
That's unbelievable! Your number is
  less than mine!
Well, okay, maybe it's believable.
And now I'm sick of you.
Bye!
```



# if-else Example Flowchart

```
printf("Pick an integer:\n");
scanf("%d", &users_number);
if (users_number < computers_number) {
    printf("That's unbelievable! Your number is\n");
    printf(" less than mine!\n");
    printf("Well, okay, maybe it's believable.\n");
} /* if (users_number < computers_number) */
else {
    printf("Wow, you picked a number that isn't\n");
    printf(" less than mine. Good work!\n");
} /* if (users_number < ...else */
printf("And now I'm sick of you.\n");
printf("Bye!\n");
```





# **if-else Indentation #1**

```
if (condition) {  
    statement_true1;  
    statement_true2;  
    ...  
}  
else {  
    statement_false1;  
    statement_false2;  
}
```

Statements inside the `if` clause are indented additionally, beyond the indentation of the `if` statement and its associated block close.

Statements inside the `else` clause are indented the same amount as statements inside the `if` clause.



## **if-else Indentation #2**

```
if (condition) {  
    statement1;  
    statement2;  
    ...  
}  
else {  
    statement_false2;  
    statement_false2;  
}
```

In CS1313, the statements inside the `if` clause are indented an additional **4 spaces** beyond the `if` statement and its associated block close, and likewise for the `else` clause.

In CS1313, you are **ABSOLUTELY FORBIDDEN** to use tabs for indenting in your source code.



# Examples of `if-else`

```
if (a > b) {
    printf("Wow, a is greater than b!\n");
} /* if (a > b) */
else {
    printf("Loser, a is not greater than b!\n");
} /* if (a > b)...else */
```

```
if (my_height < your_height) {
    shortest_height = my_height;
} /* if (my_height < your_height) */
else {
    shortest_height = your_height;
} /* if (my_height < your_height)...else */
```

