Branching (if) Part 2 Outline

1. Branching (if) Part 2 Outline
2. A Sequence of Statements to Execute When the if Condition Fails
3. Flowchart for When the if Condition Fails
4. The else Clause
5. The Meaning of else
6. Flowchart for if - else
7. if - else Example
8. if - else Flowchart
9. Indenting Inside if Blocks
10. Multiple, Related Conditions
11. The else if Clause
12. if - else if Example
13. if - else if Flowchart
14. Mixing Branching Clauses
15. if - else if - else Example
16. if - else if - else Flowchart
17. Multiple else if Clauses
18. Multiple else if Clauses (continued)
19. Multiple if - else if Example
20. Multiple if - else if Flowchart
21. if, Plus Multiple else if, Plus else
22. if, Multiple else if, else Example
23. Nested if Blocks
24. Nested if Block Example
25. How Nested if Blocks Work
26. Nested if Indentation
27. Nested if Block Example
28. Nested if Block Example (continued)

A Sequence of Statements to Execute When the if Condition Fails

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

Well, we could simply use another if block:

```c
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) || ... */
```

But that’s kind of cumbersome. Plus, it increases the chances of a mistake, since we might mistype the second condition, or we might decide to change the first but forget to change the second.
The else Clause

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, called an else clause:

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

In such a case, the sequence of statements that execute when the if condition evaluates to true (1) is called the if clause, and the sequence of statements that execute when the if condition evaluates to false (0) is called the else clause.

Notice that the else statement does not 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.

Also notice that the presence of the else clause guarantees that at exactly one of the clauses of this if block will be executed.
The Meaning of else

The statements inside the if clause are executed if and only if the condition in the if statement evaluates to true (1).

By contrast, the statements inside the else clause are executed if and only if the if condition evaluates to false (0).

So, in programming, the keyword else means “otherwise.”

Thus, exactly one of these two clauses will be executed.

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 if statement’s condition, the statements after the last block close are always executed.
### if - else Example

```c
#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) {
        /* 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 {
        /* if (users_number < computers_number) */
        printf("Wow, you picked a number that isn’t\n");
        printf("less than mine.  Good work!\n");
        /* if (users_number < computers_number) */
        printf("And now I’m sick of you.\n");
        printf("Bye!\n");
    } /* main */

    % 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!
```

Notice that the else clause must occur **AFTER** the if clause. That is, **EVERY if block MUST begin with an if clause.**
Indenting Inside if Blocks

The if statement, the else statement and the block close of the if block are indented the same amount as other statements — e.g., declarations, printf statements — that are inside the main function.

However, the statements inside an if clause or inside an else clause are indented some extra space.

Specifically, however much the if statement is indented past the block open of the function, the statements inside the if block — that is, not including the if statement, the else statement and the block close of the if block — are indented that much again.

So, in CS1313 programming projects, the if statement, the else statement and the block close of the if block should be indented four spaces farther than the block open and block close of the function, but the statements inside the if clause and the else clause should be indented eight spaces farther than the block open and block close of the function (i.e., 4 + 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) || ... */
if (abs(users_number - computers_number) <=
    close_distance) {
    printf("Close, but no cigar.\n");
} /* if (abs(users_number - computers_number) <= ... */

Multiple, Related Conditions

What if we have multiple, related conditions and we want to be able to handle each?

Well, we could simply use multiple if blocks:

```c
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 (abs(users_number - computers_number) <=
    close_distance) {
    printf("Close, but no cigar.\n");
} /* if (abs(users_number - computers_number) <= ... */
```

That's not too cumbersome.

But notice that there's a case where both printf statements might be executed: in the event that both

- users_number is less than minimum_number, and
- users_number is within close_distance of computers_number.

In that case, both outputs will be printed, which is not what we want; we want either to be told that we're outside the range, or to be told that we're close. We definitely don’t want both.
The else if Clause

C allows us to set up another special clause of statements attached to the first if clause, called an else if clause:

```c
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 if (abs(users_number - computers_number) <=
    close_distance) {
    printf("Close, but no cigar.\n");
} /* if (abs(users_number - computers_number) <= ... */
```

As usual, the statements inside the if clause are executed if and only if the condition in the if statement evaluates to true (1).

By contrast, the statements inside the else if clause are executed if and only if both of the following occur:

1. The if condition evaluates to false (0), and
2. the else if condition evaluates to true (1).

Note: in the case that the if condition evaluates to true (1), it is also the case that the else if condition isn’t evaluated at all. Why? Because in that case the statements inside the else if clause will be skipped regardless of the value of the else if condition, so the evaluation of the else if condition would be irrelevant. Why do work that isn’t going to help? This is another instance of short circuiting.

By the way, notice that it could be the case that no clause of this if block gets executed, if both conditions evaluate to false (0).

Also, notice that the indenting rules that apply to if clauses and else clauses also apply to else if clauses.

if - else if Example

```c
% cat islesselseif.c
#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 less than mine!\n");
        printf("Well, okay, maybe it’s believable.\n");
    } /* if (users_number < computers_number) */
    else if (users_number > computers_number) {
        printf("Surprise, surprise! Your number is greater than mine!\n");
        printf("And now I’m sick of you.\n");
    } /* if (users_number > computers_number) */
    printf("Bye!\n");
}
/* main */
% gcc -o islesselseif islesselseif.c
% islesselseif
Pick an integer:
6
Surprise, surprise! Your number is greater than mine!
And now I’m sick of you.
Bye
% islesselseif
Pick an integer:
5
And now I’m sick of you.
Bye
% islesselseif
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!
```

Notice that the else if clause must occur AFTER the if clause. That is, EVERY if block MUST begin with an if clause.
### Mixing Branching Clauses

Not only can we have an `else if` clause, we can also have an `else` clause as well, as the final clause of the entire `if` block.

```c
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 if (users_number == computers_number) {
    printf("That's amazing!\n");
} /* if (users_number == computers_number) */
else {
    printf("Bzzzt! Not even close.\n");
} /* if (users_number == computers_number)...else */
```

The statements inside the `else` clause are executed if and only if BOTH the `if` condition and the `else if` condition evaluate to false (0).

Notice that the presence of the `else` clause guarantees that at exactly one of the clauses of this `if` block will be executed. If the `else` clause were absent, then it might be that no clause is executed, if both of the conditions evaluated to false (0).

Again, notice that each clause has its own block open and block close.

Also, notice that the indenting rules that apply to `if` clauses and `else` clauses also apply to `else if` clauses.

Again, notice that, if it is the case that an `if` block has an `else` clause, then the `else` clause MUST be the final clause of the `if` block.
if - else if - else Example

```c
#include <stdio.h>

int 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 less than mine!\n");
        printf("Well, okay, maybe it's believable.\n");
    } /* if (users_number < computers_number) */
    else if (users_number > computers_number)
    {
        printf("Surprise, surprise! Your number is greater than mine!\n");
    } /* if (users_number > computers_number) */
    else
    {
        printf("Yowza! Your number is equal to mine!\n");
    } /* if (users_number > computers_number)...else */
    printf("And now I'm sick of you.\n");
    printf("Bye!\n");
} /* main */
```

% gcc -o islesselseifelse islesselseifelse.c
% islesselseifelse
Pick an integer:

6
Surprise, surprise! Your number is greater than mine!
And now I'm sick of you. Bye!
% islesselseifelse
Pick an integer:

5
Yowza! Your number is equal to mine!
And now I'm sick of you. Bye!
% islesselseifelse
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 if - else Flowchart

```
if (users_number < computers_number) {
    printf("That's unbelievable! Your number is less than mine!\n");
    printf("Well, okay, maybe it's believable.\n");
} /* if (users_number < computers_number) */
else if (users_number > computers_number) {
    printf("Surprise, surprise! Your number is greater than mine!\n");
} /* if (users_number > computers_number) */
else {
    printf("Yowza! Your number is equal to mine!\n");
} /* if (users_number > computers_number)...else */
printf("And now I'm sick of you.\n");
printf("Bye!\n");
```

Prompt for user's number.

Input user's number.

user's < computer's? False
True

user's > computer's? False
True

Output less. Output greater. Output equal.

Output goodbye.
Multiple else if Clauses

We don’t have to stop at just one else if clause; we can have as many as we like:

```c
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 if (users_number == computers_number) {
    printf("That's amazing!\n");
} /* if (users_number == computers_number) */
else if (abs(users_number - computers_number) <=
        close_distance) {
    printf("Close, but no cigar.\n");
} /* if (abs(users_number - computers_number) <= ... */
```

As usual, the statements inside the if clause are executed if and only if the condition (a Boolean expression completely enclosed in parentheses) in the if statement evaluates to true (1).

Also as usual, the statements inside the first else if clause are executed if and only if both of the following occur:

1. The if condition evaluates to false (0), and
2. the first else if condition evaluates to true (1).

As for the second else if clause, its statements are executed if and only if all of the following occur:

1. the if condition evaluates to false (0), and
2. the first else if condition evaluates to false (0), and
3. the second else if condition evaluates to true (1).

We can generalize this principle for an arbitrary number of else if clauses.

**General Rule for Multiple else if Clauses**

For a given else if clause, the statements inside it are executed if and only if all of the following occur:

1. The if condition evaluates to false (0), and
2. all prior else if conditions within the entire if block (in the event that there are any) evaluate to false (0), and
3. the given else if condition evaluates to true (1).

Again, the conditions (Boolean expressions completely enclosed in parentheses) in the if and else if statements are evaluated until one of them results in true (1); the conditions in subsequent else if statements within the if block are skipped.
Multiple if - else if Example

```c
#include <stdio.h>

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

    printf("Pick an integer:
");
    scanf("%d", &users_number);
    if (users_number < computers_number) {
        printf("That's unbelievable! Your number is
");
        printf("less than mine!
");
        printf("Well, okay, maybe it's believable.
");
    } /* if (users_number < computers_number) */
    else if (users_number > computers_number) {
        printf("Surprise, surprise! Your number is
");
        printf("greater than mine!
");
    } /* if (users_number > computers_number) */
    else if (users_number == computers_number) {
        printf("Yowza! Your number is equal to mine!
");
    } /* if (users_number == computers_number) */
    printf("And now I'm sick of you.
");
    printf("Bye!
");

    return 0;
}
```

```
gcc -o islesselseifs islesselseifs.c
islesselseifs
Pick an integer: 6
Surprise, surprise! Your number is greater than mine!
And now I'm sick of you.
Bye!
islesselseifs
Pick an integer: 5
Yowza! Your number is equal to mine!
And now I'm sick of you.
Bye!
islesselseifs
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!
```

Multiple if - else if Flowchart

1. Prompt for user's number.
2. Input user's number.
3. If user's number is less than computer's number, output less.
4. If user's number is greater than computer's number, output greater.
5. If user's number is equal to computer's number, output equal.
6. Output goodbye.
if, Plus Multiple else if, Plus else

Not surprisingly, we can not only have as many else if clauses as we like, we can also have an else clause as well, as the FINAL clause of the entire if block.

```c
if ((users_number < minimum_number) ||
    (users_number > maximum_number))
    printf("Hey! That's not between %d and %d!\n", minimum_number, maximum_number);
else if (users_number == computers_number)
    printf("That's amazing!\n");
else if (abs(users_number - computers_number) <= close_distance)
    printf("Close, but no cigar.\n");
else
    printf("Bzzzt! Not even close.\n");
```

The statements inside the else clause are executed if and only if the if condition and all of the else if conditions in the block evaluate to false (0).

Notice that the else clause MUST be the FINAL clause of the entire if block, and the statements inside the else clause will be executed only if all of the conditions within the entire if block evaluate to false (0).

Again, the else clause guarantees that exactly one of the clauses of this if block will be executed. If the else clause were absent, and all of the if block’s conditions evaluated to false (0), then no clause would be executed.

As usual, notice that each clause has its own block open and block close.

if, Multiple else if, else Example

```c
#include <stdio.h>

int main ()
{
    int users_number;
    printf("Pick an integer: \n");
    scanf("%d", &users_number);
    if (users_number < computers_number)
        printf("That's unbelievable! Your number is less than mine!\n");
    else if (users_number > computers_number)
        printf("Surprise, surprise! Your number is greater than mine!\n");
    else if (users_number == computers_number)
        printf("Yowza! Your number is equal to mine!\n");
    else
        printf("This statement will never be executed.\n");
    printf("And now I'm sick of you. \n");
    printf("Bye!\n");
}
```

% cat islesselseifselse.c
% gcc -o islesselseifselse islesselseifselse.c
% islesselseifselse
Pick an integer:
6
Surprise, surprise! Your number is greater than mine!
And now I'm sick of you.
Bye!
% islesselseifselse
Pick an integer:
5
Yowza! Your number is equal to mine!
And now I'm sick of you.
Bye!
% islesselseifselse
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!
Nested if Blocks

Inside each clause of an if block, we can nest more if blocks:

```c
if ( condition ) {
    statement
    ...
} else if ( condition ) {
    statement
    ...
} else {
    statement
    ...
}
```

Nested if Block Example

```c
#include <stdio.h>

int main ()
{
    /* main */
    const int minimum_number = 1;
    const int maximum_number = 10;
    const int computers_number = 5;
    const int close_distance = 1;
    int users_number;

    printf("I'm thinking of a number between %d and %d.\n", minimum_number, maximum_number);
    printf("What number am I thinking of?\n");
    scanf("%d", &users_number);
    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 if (users_number == computers_number) {
        printf("That's amazing!\n");
    } /* if (users_number == computers_number) */
    else {
        printf("Well, at least you were within the range!\n");
        if (abs(users_number - computers_number) <=
            close_distance) {
            printf(" and you were close!\n");
        } /* if (abs(users_number - computers_number) <= ...) */
        else if (users_number < computers_number) {
            printf(" but you were way too low.\n");
        } /* if (users_number < computers_number) */
        else {
            printf(" but you were way too high.\n");
        } /* if (users_number < computers_number)...else */
        printf("My number was %d.\n", computers_number);
    } /* if (users_number == computers_number)...else */
} /* main */
```
How Nested if Blocks Work

Suppose that an if block is nested inside another if block. What will happen?

Well, a statement inside a clause of an if block is executed if and only if the clause’s condition evaluates to true (1) and all prior conditions within the if block evaluate to false (0) — or, in the case that the clause is an else clause, it is executed if and only if all of the if block’s conditions evaluate to false.

On the other hand, an if statement is a normal executable statement (more or less).

So, the inner if statement will be reached, and therefore executed, if and only if the outer clause that contains it has a condition that evaluates to true (1) and if all of the outer if block’s prior clauses have conditions that evaluate to false (0) — or, in the case of an outer else clause, if all of the conditions of the outer if block’s prior clauses evaluate to false (0).

Once the inner if block is reached, it will be executed exactly like any other if block.

Nested if Indentation

Notice that the statements inside the nested if blocks are indented several extra spaces, so that it’s obvious which statements are inside which blocks.

In CS1313 programming projects, statements should be indented an extra four spaces for each block that they are inside.

We’ll see later that this rule applies not only to if blocks but to other kinds of blocks as well (e.g., while loops).
Nested if Block Example

```c
#include <stdio.h>

int main ()
{
    /* main */
    const int minimum_number = 1;
    const int maximum_number = 10;
    const int computers_number = 5;
    const int close_distance = 1;
    int users_number;

    printf("I'm thinking of a number between %d and %d.\n", minimum_number, maximum_number);
    printf("What number am I thinking of?\n");
    scanf("%d", &users_number);
    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 if (users_number == computers_number) {
        printf("That's amazing!\n");
    } /* if (users_number == computers_number) */
    else {
        printf("Well, at least you were within the range\n");
        if (abs(users_number - computers_number) <=
            close_distance)
        {
            printf(" and you were close!\n");
        } /* if (abs(users_number - computers_number) <= ... */
        else if (users_number < computers_number) {
            printf(" but you were way too low.\n");
        } /* if (users_number < computers_number) */
        else 
        {
            printf(" but you were way too high.\n");
        } /* if (users_number < computers_number) */
    } /* if (users_number == computers_number)...else */
}
/* main */
```

Nested if Block Example (continued)

```c
% gcc -o nestedif nestedif.c
% nestedif
I'm thinking of a number between 1 and 10.
What number am I thinking of?
0
Hey! That's not between 1 and 10!
% nestedif
I'm thinking of a number between 1 and 10.
What number am I thinking of?
11
Hey! That's not between 1 and 10!
% nestedif
I'm thinking of a number between 1 and 10.
What number am I thinking of?
5
Well, at least you were within the range but you were way too high.
My number was 5.
% nestedif
I'm thinking of a number between 1 and 10.
What number am I thinking of?
4
Well, at least you were within the range and you were close!
My number was 5.
% nestedif
I'm thinking of a number between 1 and 10.
What number am I thinking of?
6
Well, at least you were within the range and you were close!
My number was 5.
% nestedif
I'm thinking of a number between 1 and 10.
What number am I thinking of?
5
That's amazing!
```