Nested `if` Lesson Outline

1. Nested `if` Lesson Outline
2. A Complicated `if` Example #1
3. A Complicated `if` Example #2
4. A Complicated `if` Example #3
5. A Complicated `if` Example #4
6. A Complicated `if` Example #5
7. A Complicated `if` Example #6
8. A Complicated `if` Example Run #1
9. A Complicated `if` Example Run #2
10. A Complicated `if` Example Run #3
11. Nested `if` Blocks
12. Nesting
13. Nested `if` Block Example
14. Nested `if` Block Example #2
15. How Nested `if` Blocks Work #1
16. How Nested `if` Blocks Work #2
17. Nested `if` Indentation
18. Nested `if` Block Example #1
19. Nested `if` Block Example #2
20. Nested `if` Block Example #3
21. Nested `if` Block Example #4

A Complicated `if` Example #1

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{ /* main */

    const int int_code                 =  1;
    const int float_code               =  2;
    const int minimum_for_not_negating =  0;
    const int program_failure_code     = -1;
    const int program_success_code     =  0;
    float float_input_value, float_output_value;
    int   int_input_value,   int_output_value;
    int   data_type_code;
```

A Complicated **if** Example #2

```
printf("I'm going to calculate the absolute value\n");
printf("  of a number that you input.\n");
```

A Complicated `if` Example #3

```
printf("Would you like to input an int or a float?\n");
printf("  (Enter %d for an int or %d for a float.)\n",
    int_code, float_code);
scanf("%d", &data_type_code);
if ((data_type_code != int_code) &&
    (data_type_code != float_code)) {
    printf("ERROR: I don't recognize data type code %d.\n",
        data_type_code);
    exit(program_failure_code);
} /* if ((data_type_code != int_code) ... */
```

Idiotproofing

A Complicated `if` Example #4

```
if (data_type_code == int_code) {
    printf("What is the int value?\n");
    scanf("%d", &int_input_value);
} /* if (data_type_code == int_code) */
else if (data_type_code == float_code) {
    printf("What is the float value?\n");
    scanf("%f", &float_input_value);
} /* if (data_type_code == float_code) */
```

A Complicated `if` Example #5

```
if (data_type_code == int_code) {
    if (int_input_value < minimum_for_not_negating) {
        int_output_value = -int_input_value;
    } /* if (int_input_value < …) */
    else {
        int_output_value = +int_input_value;
    } /* if (int_input_value < …)...else */
} /* if (data_type_code == int_code) */
else if (data_type_code == float_code) {
    if (float_input_value < minimum_for_not_negating) {
        float_output_value = -float_input_value;
    } /* if (float_input_value < ...) */
    else {
        float_output_value = +float_input_value;
    } /* if (float_input_value < ...)...else */
} /* if (data_type_code == float_code) */
```

Note that we're using an `int` variable, `data_type_code`, to ***encode*** a **quality** rather than a **quantity**.

A Complicated `if` Example #6

```
    if (data_type_code == int_code) {
        printf("The absolute value of %d is %d.\n",
            int_input_value, int_output_value);
    } /* if (data_type_code == int_code) */
    else if (data_type_code == float_code) {
        printf("The absolute value of %f is %f.\n",
            float_input_value, float_output_value);
    } /* if (data_type_code == float_code) */

    return program_success_code;

} /* main */
```

A Complicated `if` Example Runs #1

```
% gcc -o absvalbytype absvalbytype.c
% absvalbytype
I'm going to calculate the absolute value
  of a number that you input.
Would you like to input an int or a float?
  (Enter 1 for an int or 2 for a float.)
0
ERROR: I don't recognize data type code 0.
```

A Complicated `if` Example Runs #2

```
% absvalbytype
I'm going to calculate the absolute value
  of a number that you input.
Would you like to input an int or a float?
  (Enter 1 for an int or 2 for a float.)
1
What is the int value?
5
The absolute value of 5 is 5.
% absvalbytype
I'm going to calculate the absolute value
  of a number that you input.
Would you like to input an int or a float?
  (Enter 1 for an int or 2 for a float.)
1
What is the int value?
-5
The absolute value of -5 is 5.
```

A Complicated `if` Example Runs #3

```
% absvalbytype
I'm going to calculate the absolute value
  of a number that you input.
Would you like to input an int or a float?
  (Enter 1 for an int or 2 for a float.)
2
What is the float value?
5.5
The absolute value of 5.500000 is 5.500000.
% absvalbytype
I'm going to calculate the absolute value
  of a number that you input.
Would you like to input an int or a float?
  (Enter 1 for an int or 2 for a float.)
2
What is the float value?
-5.5
The absolute value of -5.500000 is 5.500000.
```

Nested `if` Blocks

```
if (condition) {
    if (condition) {
        statement;
        statement;
    } /* if (condition) */
    else if (condition) {
        statement;
    } /* if (condition) */
    else {
        statement;
        statement;
    } /* if (condition)...else */
} /* if (condition) */
else if (condition) {
    statement;
    if (condition) {
        statement;
        statement;
    } /* if (condition) */
    statement;
} /* if (condition) */
else if (condition) {
    statement;
} /* if (condition) */
else {
    if (condition) {
        statement;
    } /* if (condition) */
    else {
        statement;
        statement;
    } /* if (condition)...else */
} /* if (condition)...else */
```

Nesting

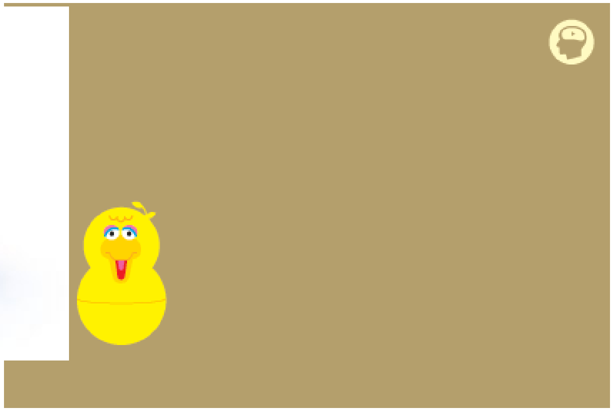
Nesting means putting something inside something else.

For example, one `if` block can be nested inside another `if` block.

We refer to the inner `if` block as “the inner `if` block,” and we refer to the outer `if` block as “the outer `if` block.” Go figure.

http://www.londonshakespeare.org.uk/prisondiaries/images/russian_doll_bigger_to_smaller_md_wht.gif

http://grassreport.files.wordpress.com/2011/10/yellow-russian-doll_b_grassreport.gif



http:/ /1SAI/AAAAAAAAAWE/KgtS2M3unp8/s400/geekyRussianDolls1.JPG

Nested `if` Block Example #1

```
#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);
```

Nested `if` Block Example #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 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 #1

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

Well, the sequence of statements **inside** a clause of an `if` block is executed only in the event that:

- 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,

its sequence of statements will be executed only in the event that **all** of the `if` block's conditions evaluate to false (`0`).

How Nested `if` Blocks Work #2

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

So, in order for the inner `if` statement to be reached, and therefore executed:

- the outer clause that contains it must have a condition that evaluates to true (`1`), and
- all of the outer `if` block's prior clauses must have conditions that evaluate to false (`0`).

Or, in the case of an outer `else` clause,
all of the conditions of the outer `if` block's prior conditions must 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 (for example, `while` loops).

Nested `if` Block Example #1

```
#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);
```

Nested `if` Block Example #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 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 */
```

Nested `if` Block Example #3

```
% 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?
10
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?
1
Well, at least you were within the range
  but you were way too low.
My number was 5.
```

Nested `if` Block Example #4

```
% 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!
```