If Statements

2025 Winter APS105: Computer Fundamentals
Jon Eyolfson

Lecture 6
1.0.1

Right Now Your Program Always Does the Same Thing

When you run `main` it runs line by line
  The input may be different (but that's it)

There is a Way to Conditionally Run Code

It is called an if statement and uses a boolean

The syntax of an if statement is:

```
if (<expression>) <statement>
```

However, you should **always** write it like:

```
if (<expression>) {
    <statements>
}
```

Recall: An expression is a combination of operands and operators

The Statements Will Only Run if the Expression is True

```
if (<expression>) {
    <statements>
}
```

C will compute the result of the expression and convert it to a boolean
Recall: a boolean is either `0` (`false`) or `1` (`true`)

If the expression is true, we run the statements between { and }
after those we run the next statement after }

If the expression is false, we immediately run the next statement after }

The Expression in an If Statement Will Convert to a Boolean

There are more operators that result in a boolean
  Relational operators work with numbers

They are the following:
  == meaning $=$ in math (remember = in C is assignment!)
  != meaning $\neq$ in math
  < meaning $<$ in math
  <= meaning $\leq$ in math
  > meaning $>$ in math
  >= meaning $\geq$ in math

The Expression in an If Statement Will Convert to a Boolean

There are more operators that result in a boolean
  Relational operators work with numbers

They are the following:
  == meaning $=$ in math (remember = in C is assignment!)
  != meaning $\neq$ in math
  < meaning $<$ in math
  <= meaning $\leq$ in math
  > meaning $>$ in math
  >= meaning $\geq$ in math

Examples:
  `2 == 2` → `1`
  `3 < 2` → `0`
  `3 >= 3` → `1`

The Relational Operators Follow Similar Type Rules

If both operands are `int` it does the operator using integers

If at least one operand is a `double`, the other will be converted to a `double` if it is not one already
The operator uses real numbers

The result of either case is a boolean!

The Relational Operators Follow Similar Type Rules

If both operands are `int` it does the operator using integers

If at least one operand is a `double`, the other will be converted to a `double` if it is not one already
The operator uses real numbers

The result of either case is a boolean!

Example:
`2 == 2.0` → `1`

There Are Operators That Operate with Booleans

These operators are called logical operators
All operands are booleans and the result is a boolean

The logical operators are the following:
|| means logical or (if either is 1 the result is 1, otherwise 0)
&& means logical and (only if both are 1 the result is 1, otherwise 0)
! (unary) means logical not (flip the value)

There Are Operators That Operate with Booleans

These operators are called logical operators
All operands are booleans and the result is a boolean

The logical operators are the following:
|| means logical or (if either is 1 the result is 1, otherwise 0)
&& means logical and (only if both are 1 the result is 1, otherwise 0)
! (unary) means logical not (flip the value)

Examples:

```
0 || 0 → 0
1 || 0 → 1
0 && 1 → 0
1 && 1 → 1
!1 → 0
!0 → 1
```

Rules for Converting Numbers to Booleans

If C requires a boolean, it will convert (cast) your value

As a general rule, if the number represents 0, it gets converted to `0`
Otherwise it gets converted to `1`

Where the Precedence Rules for Today's Operators Fit

Operator	Associativity
* / %	Left-to-right
+ -	Left-to-right
< <= > >=	Left-to-right
== !=	Left-to-right
&&	Left-to-right
||	Left-to-right
= += -= *= /= %= (assignments)	Right-to-left

Higher Precedence

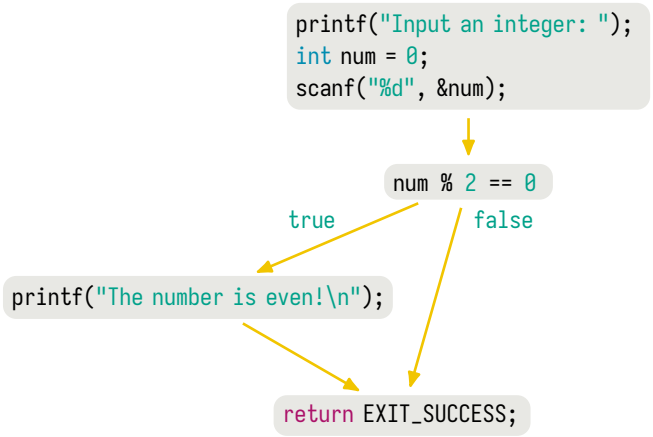
Lower Precedence

A Program That Outputs If a Number is Even

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

int main(void) {
    printf("Input an integer: ");
    int num = 0;
    scanf("%d", &num);
    if (num % 2 == 0) {
        printf("The number is even!\n");
    }
    return EXIT_SUCCESS;
}
```

Which Statements Run in the Previous Program



We Can Run Code Only If the Result is False

Another valid syntax of an if statement is:

```
if (<expr>) <statement>
else <statement>
```

However, you should **always** write it like:

```
if (<expression>) {
    <statements>
}
else {
    <statements>
}
```

We Can Run Code Only If the Result is False

Another valid syntax of an if statement is:

```
if (<expr>) <statement>
else <statement>
```

However, you should **always** write it like:

```
if (<expression>) {
    <statements>
}
else {
    <statements>
}
```

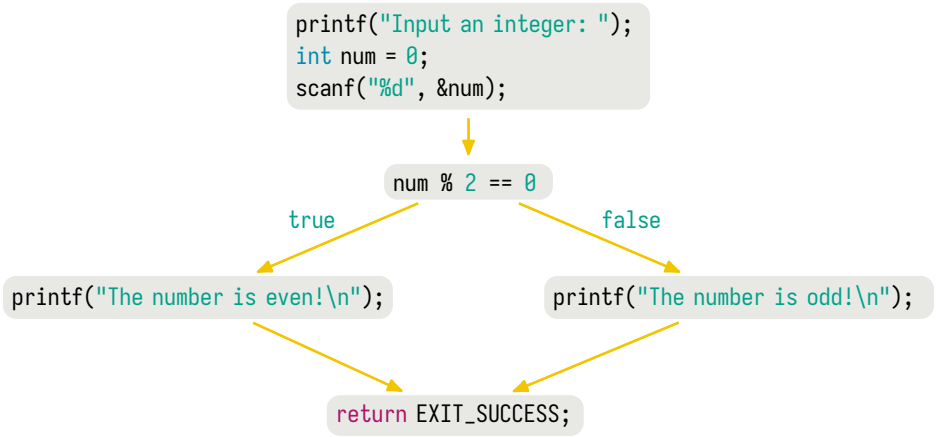
The statements between the { and } after the `else` only run if the result of the expression is false

A Program That Outputs If a Number is Odd or Even

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

int main(void) {
    printf("Input an integer: ");
    int num = 0;
    scanf("%d", &num);
    if (num % 2 == 0) {
        printf("The number is even!\n");
    }
    else {
        printf("The number is odd!\n");
    }
    return EXIT_SUCCESS;
}
```

Which Statements Run in the Previous Program



What Happens With an Else and Two If Statements?

```c
int main(void) {
    printf("Input an integer: ");
    int num = 0;
    scanf("%d", &num);
    if (num % 2 == 0) {
        printf("The number is even!\n");
    }
    if (num >= 10) {
        printf("The number is 10 or greater!\n")  ;
    }
    else {
        /* Does this run if the number is less than 10,
           or if the number is odd? */
    }
    return EXIT_SUCCESS;
}
```

Previous Program is an Example of the Dangling Else Problem

The syntax would be the same for either meaning,
so which `if` does the `else` apply to?

The `else` always applies to the first `if` above it

In the comment, we should put:

```
printf("The number is less than 10!\n");
```

Let's Make a Game by Using a Better Source of Randomness

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

int main(void) {
    /* Seed the random number generator using the current timestamp. */
    srand(time(NULL));
    printf("Enter your bet, even or odd? (0 for even, 1 for odd): ");
    int bet;
    scanf("%d", &bet);
    int dice = rand() % 6 + 1;
    printf("Rolled a %d\n", dice);
    if (dice % 2 == bet) {
        printf("You win!\n");
    }
    else {
        printf("You lose!\n");
    }
    return 0;
}
```