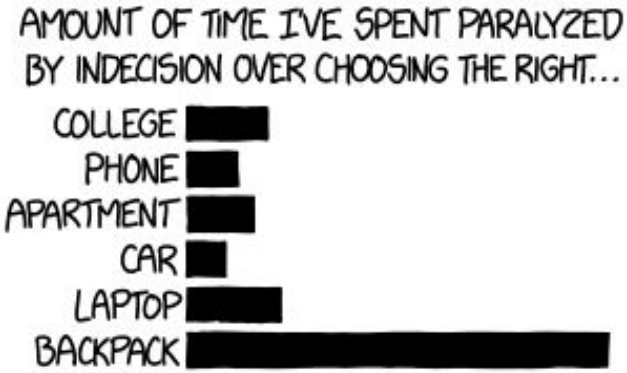
Lecture 9: Decisions



Announcements and reminders

Submissions:

- HW 4 -- due Saturday at 6 PM

Course reading to stay on track:

- 3.1-3.2, 3.7-3.8 by today
- 3.3-3.6 by Wednesday
- Might start Ch. 4 (**loops**) Friday lecture, continue next week

Practicum 1

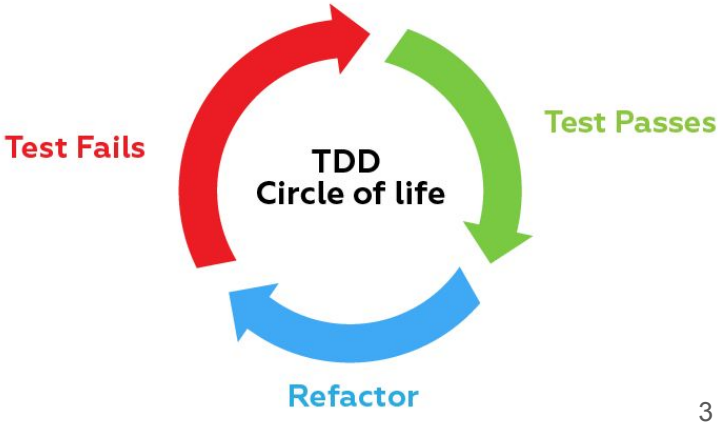
- Wednesday 20 Feb

Last time on *Intro Computing…*

We learned how to build our functions from the bottom-up!

We learned how to test our software!

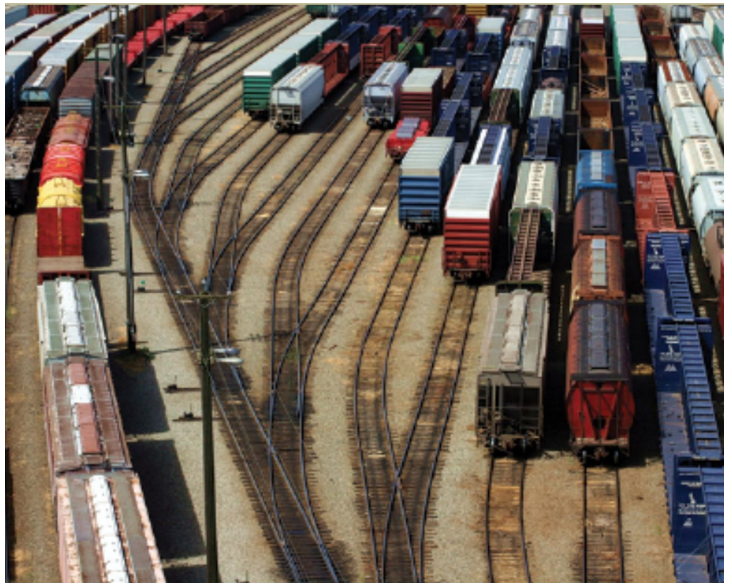
… and how to build our software and test it as we go!



Chapter 3: Decisions

Chapter Topics

- **The if statement**
- Comparing numbers and strings
- Multiple alternatives
- Nested branches
- Problem-solving: flowcharts
- Problem-solving: test cases
- Boolean variables and operators
- Application: input validation

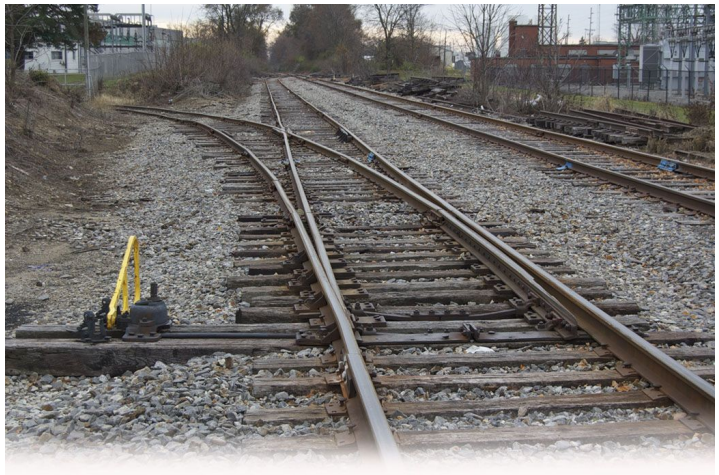


The if statement

The if statement is used to implement a decision

- ***IF*** the condition is fulfilled, then one set of statements is executed
- Otherwise, another set of statements is executed

Example: Syntax - pseudocode and trains



The if statement

The if statement is used to implement a decision

- ***IF*** the condition is fulfilled, then one set of statements is executed
- Otherwise, another set of statements is executed

Example: Syntax - pseudocode and shapes

The if statement

General syntax:

```
if (condition)        // never put a ; after these parentheses
{
   statement1;    // executed if condition is true
}
else                  // optional - what to do if condition is false?
{
   statement2;    // executed if condition is false
}                     // braces are optional but recommended (nicer to read by humans)
```

Boolean expressions

Data type: `bool`

- Can define variables, same as `int` or `double`
- 2 possible values: `true` or `false` (or think of as 1 or 0, respectively)
- `true`/`false` are predefined library constants

Two flavors:

1) **Comparison operators:** `==` `<` `>` `!=` `<=` `>=`

2) **Logical operators:** logical AND: `&&`

logical OR: `||`

Boolean expressions

Data type: `bool`

- Can define variables, same as `int` or `double`
- 2 possible values: `true` or `false` (or think of as 1 or 0, respectively)
- `true`/`false` are predefined library constants

Two flavors:

1) **Comparison operators:** `==` `<` `>` `!=` `<=` `>=`

2) **Logical operators:** logical AND: &&

   logical OR: ||

Boolean expressions - some examples

Math symbol	English	C++ notation	C++ example	Math equivalent
=	Equal to	==	x + 7 == y	
≠	Not equal to	!=	shape != "tri"	
<	Less than	<	count < m + 3	
≤	Less than or equal to	<=	time <= limit	
>	Greater than	>	time > limit	
≥	Greater than or equal to	>=	age >= 21	

Boolean expressions -- elevator code example

Example: Many buildings do not include a 13th floor, due to superstition. Can we write code to calculate the *actual* floor, depending on whether or not the floor in question is above/below the 13th? (For example, if `floor = 16`, we are on `actual_floor = 15`)

Boolean expressions -- elevator code example

Example: Many buildings do not include a 13th floor, due to superstition. Can we write code to calculate the *actual* floor, depending on whether or not the floor in question is above/below the 13th? (For example, if `floor = 16`, we are on `actual_floor = 15`)

```
int floor = -1;
cin >> floor;
int actual_floor;
if (floor > 13)
{
   actual_floor = floor - 1;
}
else
{
   actual_floor = floor;
}
```

Boolean expressions -- elevator code example

Example, rebooted: What if we wanted to write this a bit more compactly? Namely, let's write the same function but without the `else` statement.

Boolean expressions -- elevator code example

Example, rebooted: What if we wanted to write this a bit more compactly? Namely, let's write the same function but without the `else` statement.

```
int floor = -1;
cin >> floor;
int actual_floor = floor;
if (floor > 13)
{
   actual_floor--;
}
```

(We also used the decrement operator!)

Boolean expressions -- elevator code example

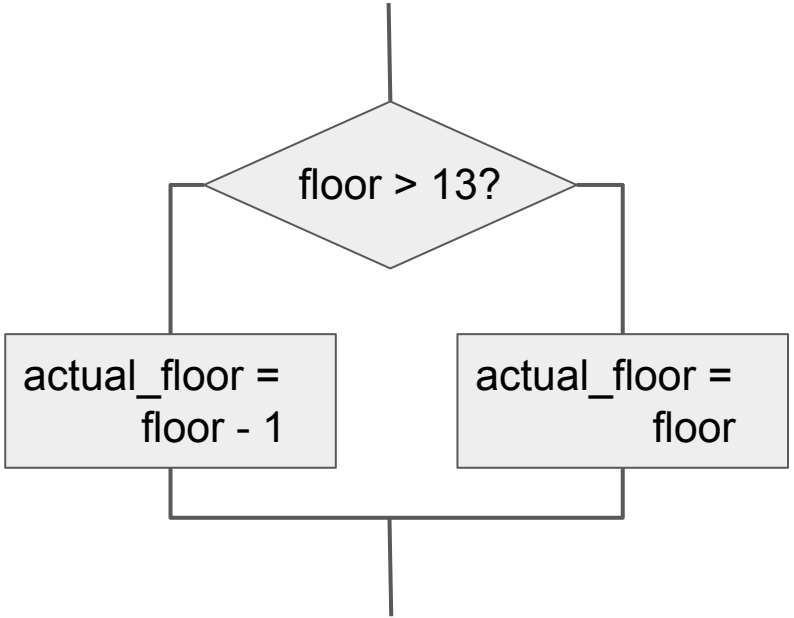
Example, rebooted again: Using a **logical** operator, let's account for the fact that the people who own the building must **be superstitious** in order to skip the 13th floor.

Boolean expressions -- elevator code example

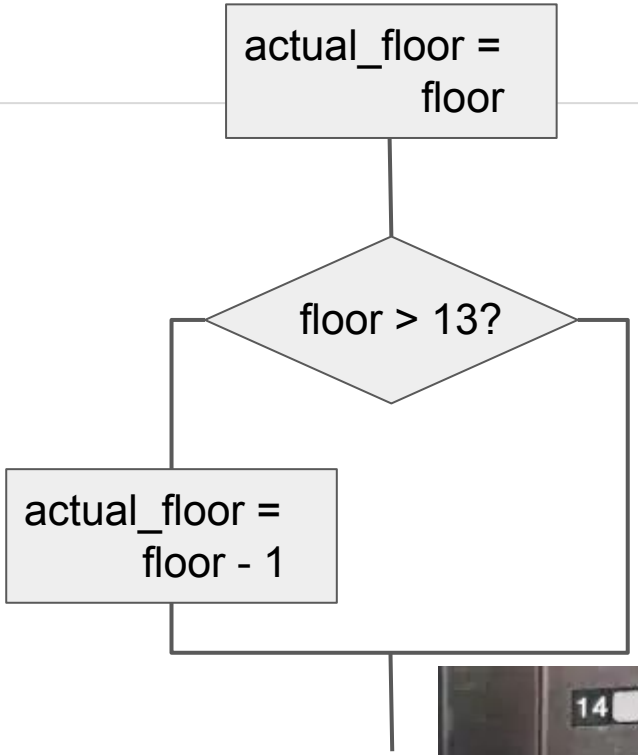
Example, rebooted again: Using a **logical** operator, let's account for the fact that the people who own the building must **be superstitious** in order to skip the 13th floor.

```
int floor = -1;
bool is_superstitious;
cout << "What floor are you on? ";
cin >> floor;
int actual_floor = floor;
cout << "Are you superstitious? (0=no, 1=yes) ";
cin >> is_superstitious;
if (floor > 13 && is_superstitious)
{
   actual_floor--;
}
```

if statement flowcharts



With else



Without else

if statement -- bracket layout

It’s good coding practice to make your code easier to read

Lining up braces/brackets vertically helps

Examples:

```
if (floor > 13)
{
   actual_floor--;
}
```

if statement -- bracket layout

It's good coding practice to make your code easier to read

Lining up braces/brackets vertically helps

Examples:

```
if (floor > 13)
{
   actual_floor--;
}
```

```
if (floor > 13) {
   actual_floor--;
}
```

→ Some programmers prefer this style

→ Saves a line of code!

Rule: Always align the ending brace to clearly show what it is closing, so there is no confusion.

if statement -- bracket layout

When the body of an if statement consists on a single statement, you do not need braces:

Examples:

```
if (floor > 13)
{
   actual_floor--;
}
```

```
if (floor > 13)
   actual_floor--;
```

However, it is a good idea to always include the braces:

- Makes your code easier to read
- Less likely to commit errors such as…

if statement -- common error: the statement

Example: First: what's wrong here?
Second, what will happen in each of these two cases?

```
if (floor > 13);
{
   actual_floor--;
}
```

```
if (floor > 13);
   actual_floor--;
```

if statement -- common error: the do-nothing statement

Example: First: what's wrong here?
Second, what will happen in each of these two cases?

```
if (floor > 13);
{
   actual_floor--;
}
```

```
if (floor > 13);
   actual_floor--;
```

- If `floor > 13`, execute the **do-nothing statement**, (a semicolon by itself)
- Then, execute the code after the do-nothing line
  - The code in braces is no longer part of the `if` statement

if statement -- indentation when nesting

Block-structured code has the property that **nested statements** are indented by one or more levels. Makes things easier to read and keep track of.

```
int main()
{
   int floor;
   ...
   if (floor > 13)
   {
      floor--;
   }
   ...
   return 0;
}
```

0 1 2 ← indentation level

? The conditional operator

C++ has a conditional operator ? of the form:

`[condition] ? value1 : value2`

The value of this expression is `value1` if `[condition]` is true, and is `value2` if `[condition]` is false.

Example: How can we rewrite our `actual_floor` correction using the conditional operator?

? The conditional operator

C++ has a conditional operator ? of the form:

```
[condition] ? value1 : value2
```

The value of this expression is `value1` if `[condition]` is true, and is `value2` if `[condition]` is false.

Example: How can we rewrite our `actual_floor` correction using the conditional operator?

```
actual_floor = floor > 13 ? floor - 1 : floor
```

if statement -- remove redundancy!

Example: Can you find anything redundant in this code? How can we make the code leaner and meaner?

```
if (floor > 13)
{
   actual_floor = floor - 1;
   cout << "Actual floor: " << actual_floor << endl;
}
else
{
   actual_floor = floor;
   cout << "Actual floor: " << actual_floor << endl;
}
```

if statement -- remove redundancy!

Example: Can you find anything redundant in this code? How can we make the code leaner and meaner?

```
if (floor > 13)
{
   actual_floor = floor - 1;
}
else
{
   actual_floor = floor;
}
cout << "Actual floor: " << actual_floor << endl;
```

→ We can save an identical line of code by moving the `cout` statement to the outside of the `if/else` statement (and deleting one)

What just happened?

- We learned about **if statements**!
- … and **else** statements!
- We learned about formatting **conventions for braces { } and indentation**!
- We learned about the **do-nothing statement**!
- We dipped our toes in the waters of **Boolean expressions**!
    - Either **true** or **false**