CS111

Introduction to Computing Science

The `if` Statement

```
if it's quicker to the candy mountain,
        we'll go that way
else
        we go that way
```
but what about that way?

Multiple Alternatives

Multiple if statements can be combined
to evaluate complex decisions.

Multiple Alternatives

How would we write code to deal with Richter scale values?

Multiple Alternatives

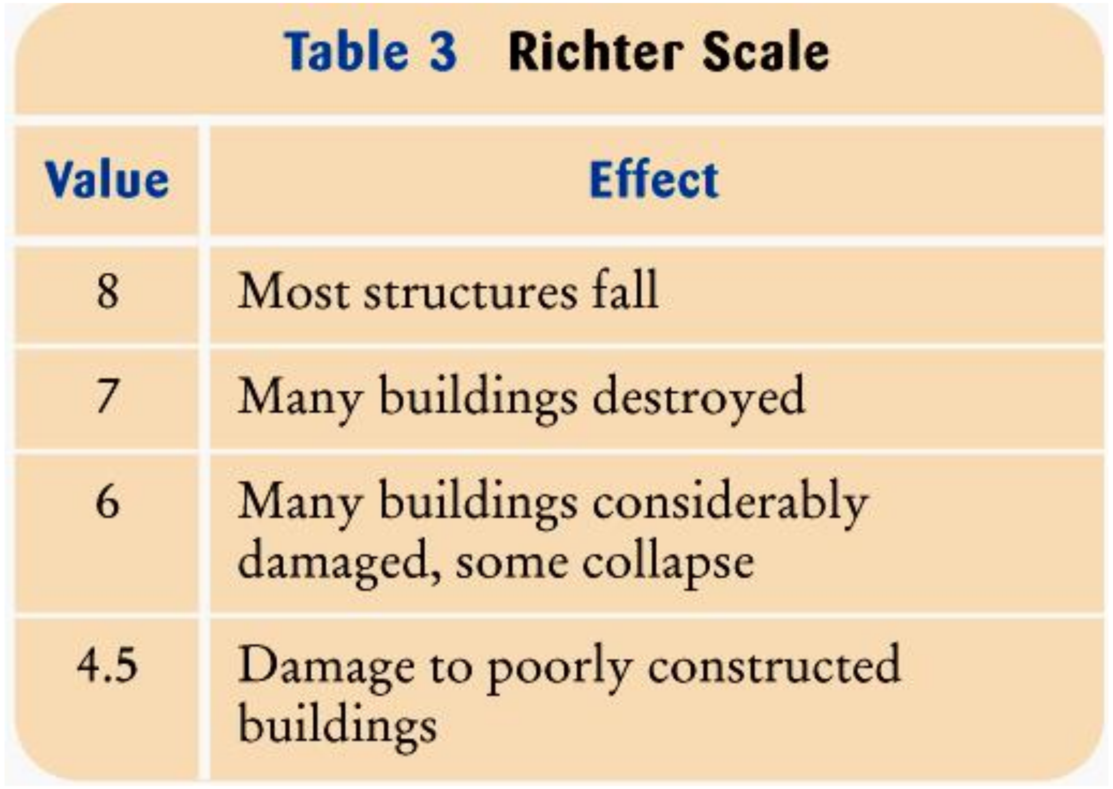


Table 3 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings

Multiple Alternatives

In this case, there are five branches:

one each for the four descriptions of damage,

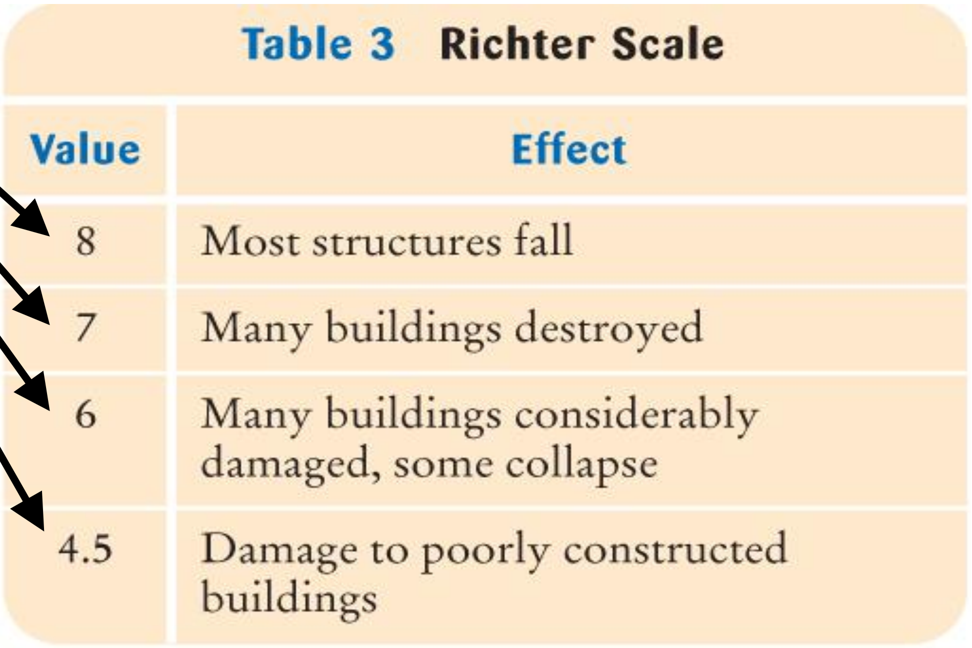


Table 3 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings

and one for no destruction.

Multiple Alternatives

You use multiple if statements
to implement multiple alternatives.

Multiple Alternatives

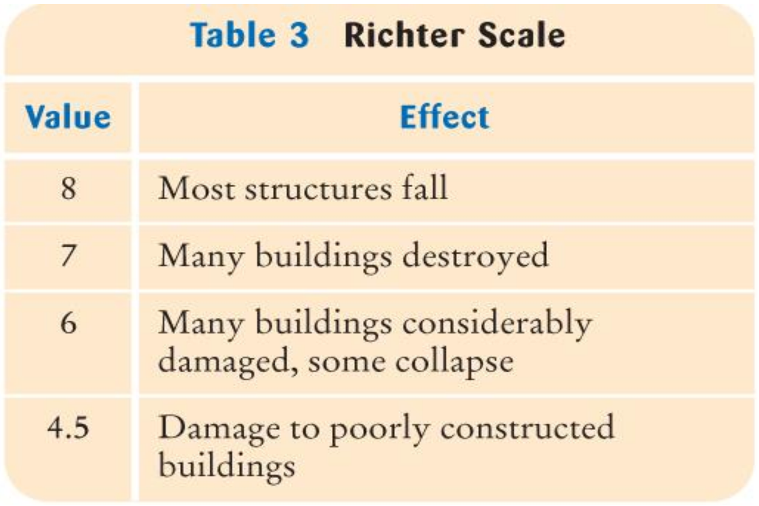
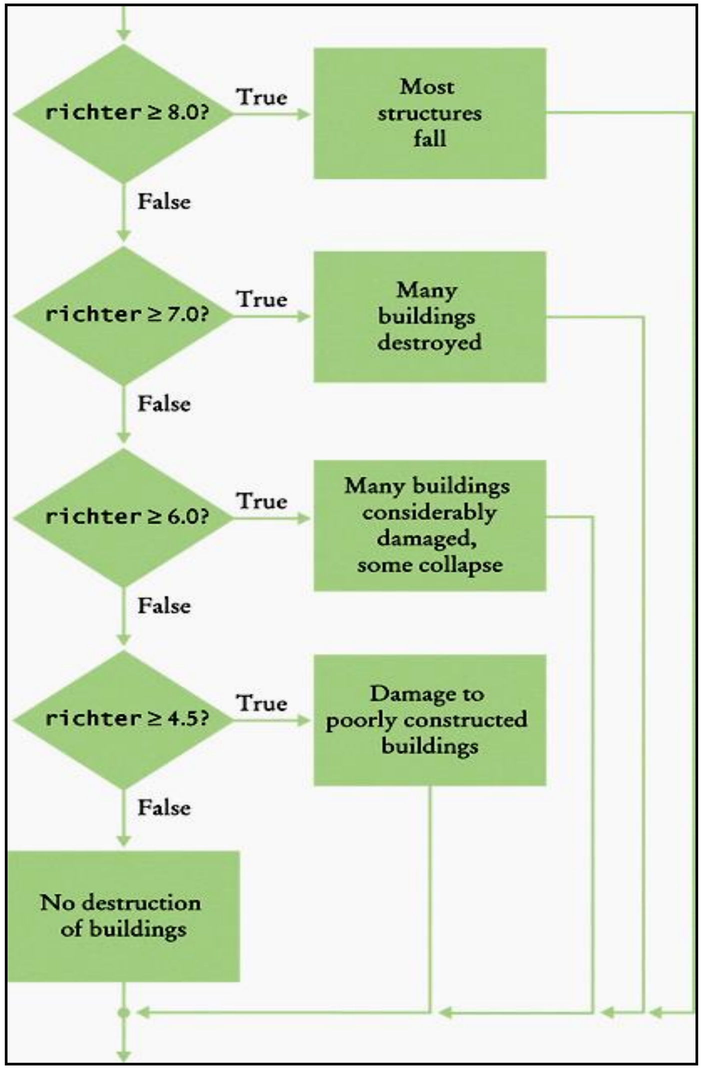


Table 3 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings



Multiple Alternatives

```
if (richter >= 8.0)
{
   cout << "Most structures fall";
}
else if (richter >= 7.0)
{
   cout << "Many buildings destroyed";
}
else if (richter >= 6.0)
{
   cout << "Many buildings considerably damaged, some collapse";
}
else if (richter >= 4.5)
{
   cout << "Damage to poorly constructed buildings";
}
else
{
   cout << "No destruction of buildings";
}
. . .
```

Multiple Alternatives

If a test is `false`,
that block is skipped and
the next test is made.

```
if (richter >= 8.0)
{
   cout << "Most structures fall";
}
else if (richter >= 7.0)
{
   cout << "Many buildings destroyed";
}
else if (richter >= 6.0)
{
   cout << "Many buildings considerably damaged, some collapse";
}
else if (richter >= 4.5)
{
   cout << "Damage to poorly constructed buildings";
}
else
{
   cout << "No destruction of buildings";
}
. . .
```

Multiple Alternatives

```
if (richter >= 8.0)
{
   cout << "Most structures fall";
}
else if (richter >= 7.0)
{
   cout << "Many buildings destroyed";
}
else if (richter >= 6.0)
{
   cout << "Many buildings considerably damaged, some collapse";
}
else if (richter >= 4.5)
{
   cout << "Damage to poorly constructed buildings";
}
else
{
   cout << "No destruction of buildings";
}
. . .
```

As soon as one of the four tests succeeds, that block is executed, displaying the result,

and no further tests are attempted.

Multiple Alternatives – Wrong Order of Tests

Because of this execution order,
when using multiple if statements,
pay attention to the order of the conditions.

The `switch` Statement

This is a bit of a mess to read.

```
int digit;
...
if (digit == 1) { digit_name = "one"; }
else if (digit == 2) { digit_name = "two"; }
else if (digit == 3) { digit_name = "three"; }
else if (digit == 4) { digit_name = "four"; }
else if (digit == 5) { digit_name = "five"; }
else if (digit == 6) { digit_name = "six"; }
else if (digit == 7) { digit_name = "seven"; }
else if (digit == 8) { digit_name = "eight"; }
else if (digit == 9) { digit_name = "nine"; }
else { digit_name = ""; }
```

The `switch` Statement

C++ has a statement that helps a bit with the readability of situations like this:

The switch statement.

ONLY a sequence of if statements that compares a single value against several constant alternatives can be implemented as a switch statement.

Use only with integer or characters.

The `switch` Statement

```
int digit;

switch (digit)
{
   case 1: digit_name = "one"; break;
   case 2: digit_name = "two"; break;
   case 3: digit_name = "three"; break;
   case 4: digit_name = "four"; break;
   case 5: digit_name = "five"; break;
   case 6: digit_name = "six"; break;
   case 7: digit_name = "seven"; break;
   case 8: digit_name = "eight"; break;
   case 9: digit_name = "nine"; break;
   default: digit_name = ""; break;
}
```

The *default branch* is chosen if none of the cases matches.

The `switch` Statement

```
int digit;

switch (digit)
{
   case 1: digit_name = "one"; break;
   case 2: digit_name = "two"; break;
   case 3: digit_name = "three"; break;
   case 4: digit_name = "four"; break;
   case 5: digit_name = "five"; break;
   case 6: digit_name = "six"; break;
   case 7: digit_name = "seven"; break;
   case 8: digit_name = "eight"; break;
   case 9: digit_name = "nine"; break;
   default: digit_name = ""; break;
}
```

'break' means to leave the switch immediately.

continue here!

The `switch` Statement

Break

- Every branch of the switch must be terminated by a break statement.
- If the break is missing, execution falls through to the next branch, and so on, until finally a break or the end of the switch is reached.
- In practice, this fall-through behavior is rarely useful, and it is a common cause of errors.
- If you accidentally forget the break statement, your program compiles but executes unwanted code.

Common Error – Forgotten break

```
int digit;

switch (digit)
{
   case 1: digit_name = "one"; break;
   case 2: digit_name = "two";
   case 3: digit_name = "three"; break;
   case 4: digit_name = "four"; break;
   case 5: digit_name = "five"; break;
   case 6: digit_name = "six"; break;
   case 7: digit_name = "seven"; break;
   case 8: digit_name = "eight"; break;
   case 9: digit_name = "nine"; break;
   default: digit_name = ""; break;
}
```

A forgottten 'break' means you stay in the switch.

You'll go to the next test. It is false.

You'll go to the next test. It is false again.

Etcetera.

You'll end up at the default. It is always true. And you execute also this block.

Oh, no!

Have a break in every case.

Common Error – Forgotten break

Many programmers consider the switch statement somewhat dangerous and prefer the if statement.

If your aren’t sure about a switch, use an if-statement.

Exercise

Normally at USP, grade for a course is determined as follows

Range	Grade
85-100	A+
78-84	A
71-77	B+
64-70	B
57-63	C+
50-56	C
40-49	D
0-39	E

Exercise

Write a variable declaration to hold the score.
What data type could it be?

Exercise

Write a `cout` statement to ask the user to enter score and a `cin` statement to read in the value?

Exercise

Write an if statement to test whether score is greater than or equal to 85 and if true display “You got A+”

Exercise

Write an **`else if`** part to test whether the score is greater than or equal to 78 and if so display “You got A”.

Exercise

After you have tested all the conditions, how will the last part of the **`if`**…**`else if`** look like?