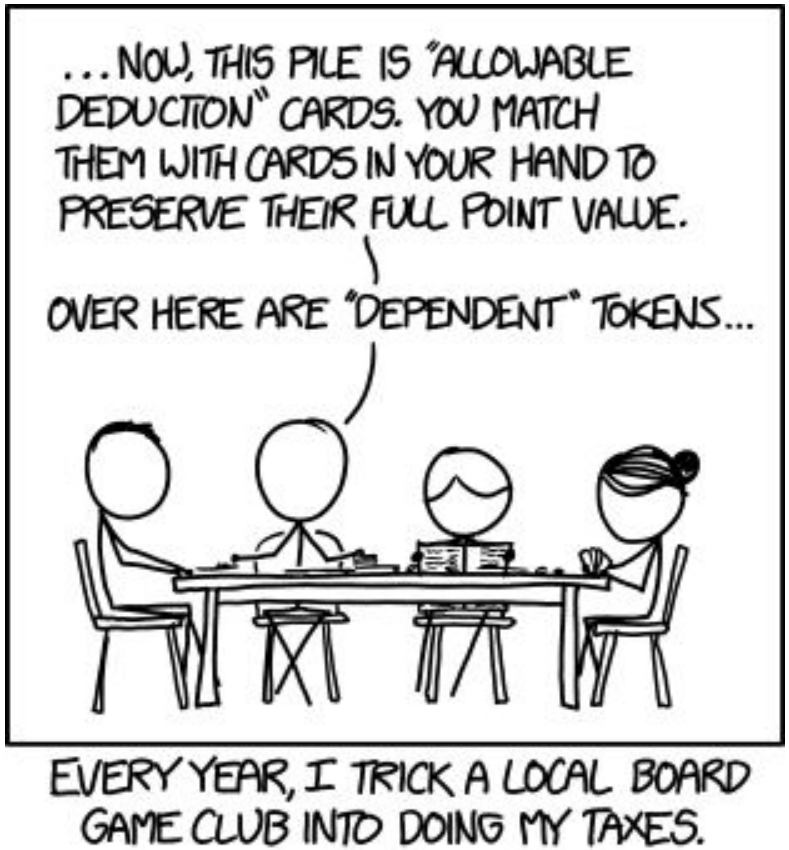
Lecture 10: Nested Decisions



Announcements and reminders

Submissions:

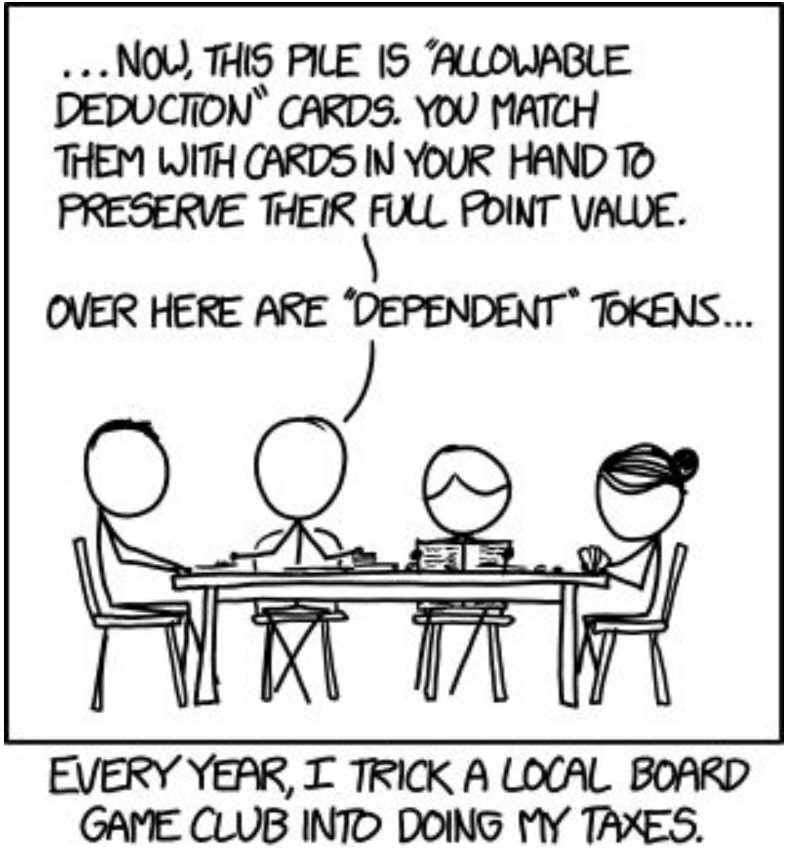
- HW 4 -- due Saturday at 6 PM

Course reading to stay on track:

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

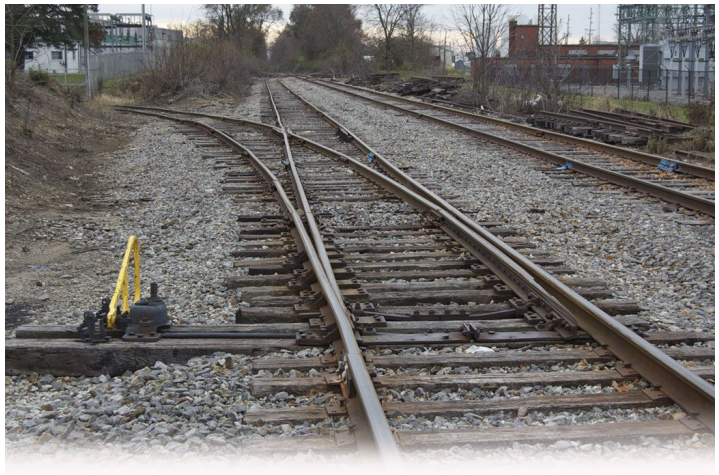
Practicum 1

- Wednesday 20 Feb



Last time on *Intro Computing…*

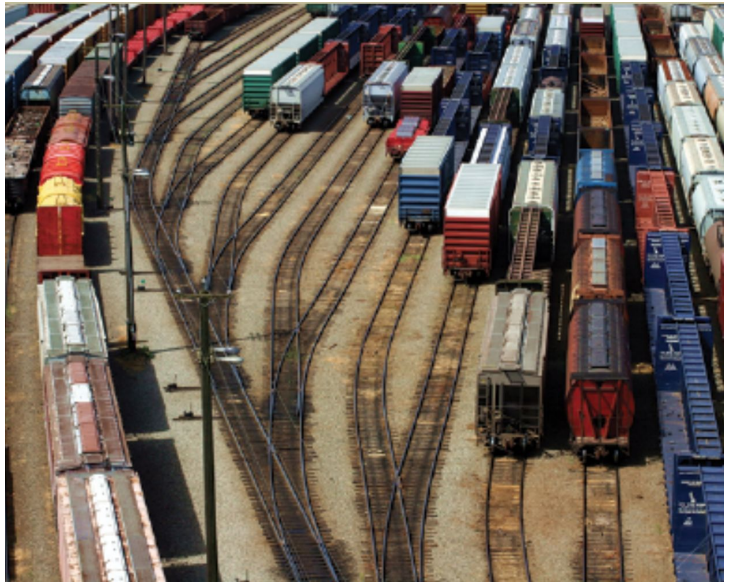
- We learned about **if statements**!
- … and **else** statements!
- We learned about formatting **conventions for braces { } and indentation**!
- We learned about the **do-nothing statement**!
- We dipped our toes in the waters of **Boolean expressions**!
    - Either **true** or **false**



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



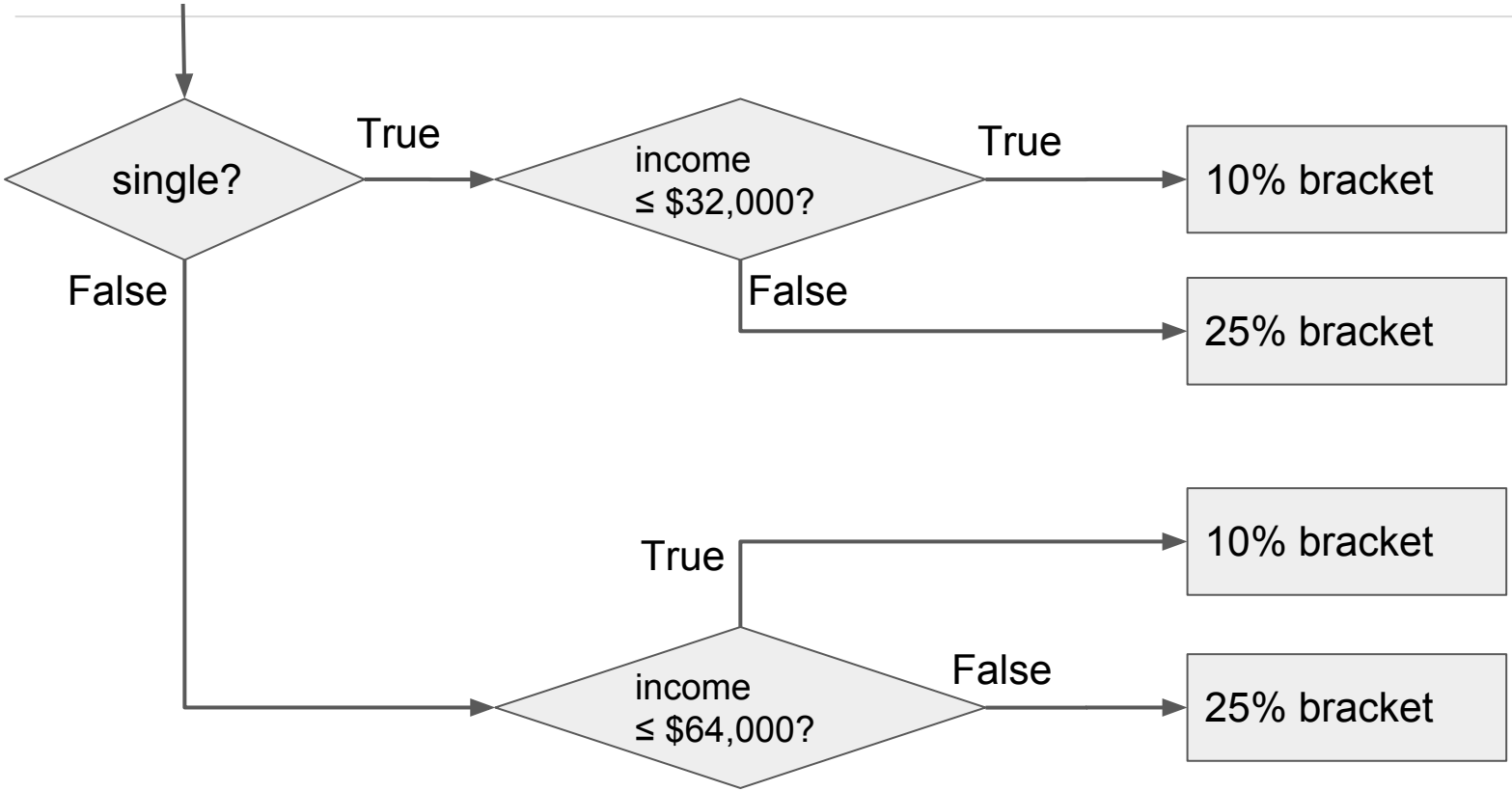
Nested Branches

In the U.S., tax rates depend (among other things) on the taxpayer's marital status

- Single folks have higher tax rates
- Married taxpayers add their income together and pay taxes on the total
- From the IRS in a recent year:

Single and taxable income…	**… the tax rate is…**	**… for the amount over …**
≤ $32,000	10%	$0
> $32,000	$3,200 + 25%	$32,000
Married and taxable income…	**… the tax rate is…**	**… for the amount over …**
≤ $64,000	10%	$0
> $64,000	$6,400 + 25%	$64,000

Example: Tax Table Decisions -- Flowchart



Example: Tax Table Decisions -- coding

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
```

Example: Tax Table Decisions -- coding

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    const double RATE_LO = 0.10;
    const double RATE_HI = 0.25;
    const double HI_SINGLE_LIMIT = 32000;
    const double HI_MARRIED_LIMIT = 64000;

    double tax1 = 0;
    double tax2 = 0;

    double income;
    cout << "Please enter your income: ";
    cin >> income;
```

```
cout << "Please enter filing status (single or married): ";
string marital_status;
cin >> marital_status;

if (marital_status == "single")
{
    if (income <= LO_SINGLE_LIMIT)
    {
        tax1 = RATE_LO * income;
    }
    else
    {
        tax1 = RATE_LO * LO_SINGLE_LIMIT;
        tax2 = RATE_HI * (income - LO_SINGLE_LIMIT);
    }
}
else
```

```
{
    if (income <= LO_MARRIED_LIMIT)
    {
        tax1 = RATE_LO * income;
    }
    else
    {
        tax1 = RATE_LO * LO_MARRIED_LIMIT;
        tax2 = RATE_HI * (income - LO_MARRIED_LIMIT);
    }
}

double total_tax = tax1 + tax2;

cout << "The tax is $" << total_tax << endl;
return 0;
}
```

Example: Comparing 3 Numbers

Question: How can we determine which one is the largest? Or smallest?

Example: Classifying Earthquakes

In the case of the **Richter Scale** for earthquake magnitude, there are 5 branches:

- one for each of the four descriptions here, and
- a fifth “default” for no destruction

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

Example: Classifying Earthquakes -- flowchart

Example: Classifying Earthquakes -- coding

Example: Classifying Earthquakes -- coding

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

Multiple Alternatives: Order of Tests

The **order of execution** of our `if` statements matters a lot here!

Let's see what happens if we screw it up.

Multiple Alternatives: Order of Tests

The **order of execution** of our if statements matters a lot here! → **What if we screw it up?**

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

What happens if
richter = 7.1?

switch Statement vs if Statement

Below is a very complicated block of if (...) statements to choose a text string based on the value of an int variable

```
int digit;
   ...          // digit variable gets set here by some code...
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 = ""; }
```

switch Statement vs if Statement

The **switch** statement is an alternative to nested if/else statements.

… But switch is at least as awkward to code as nested if/else statements:

```
int digit;  // NB: switch can only apply to int/char types
   ...         // digit variable gets set here by some code...
switch(digit)
{




}
```

switch Statement vs if Statement

The **switch** statement is an alternative to nested if/else statements.

… But switch is at least as awkward to code as nested if/else statements:

```
int digit;  // NB: switch can only apply to int/char types
   ...         // digit variable gets set here by some code...
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;       // taken if none of the above apply
}
```

`switch` Statement vs `if` Statement

What's with that `break` ?

- Every branch of the switch must be terminated by a `break` statement (and semicolon ;)
- `break` tells the machine to skip down to the end of the switch statement, because a match was found
- If the `break` is missing, execution falls through to the next branch, and next one, and next one, until finally a `break` or the end of the switch is reached
- In practice, this fall-through behavior is **rarely useful** and **causes lots of errors**!
- If you accidentally forget the `break` statement, program will compile but executes unwanted code.

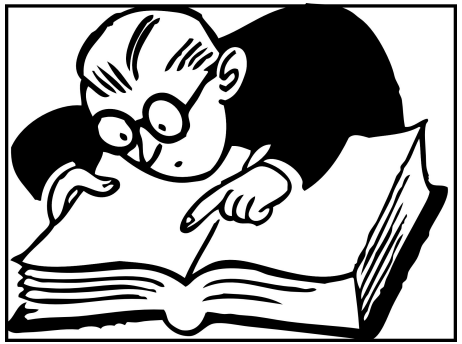
  → Try it and see!

Hand-tracing

- Very useful technique for understanding whether a program works as desired
- You simulate the program's activity on a sheet of paper/whiteboard/by muttering
- Can use this method with pseudocode or real code

How-To: Start by getting out your (pseudo)code

- Use something to mark your current statement
- "Execute" the statements one at a time
- Every time the value of a variable changes (or is initialized), cross out the old value and write in the new one below the old value



Hand-tracing the Tax Example from Earlier

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    const double RATE_LO = 0.10;
    const double RATE_HI = 0.25;
    const double HI_SINGLE_LIMIT = 32000;
    const double HI_MARRIED_LIMIT = 64000;

    double tax1 = 0;
    double tax2 = 0;

    double income;
    cout << "Please enter your income: ";
    cin >> income;
```

S'pose the user is making $80,000 and is married. Nice!

```
cout << "Please enter s for single, m for married: ";
string marital_status;
cin >> marital_status;

if (marital_status == "s")
{
    if (income <= LO_SINGLE_LIMIT)
    {
        tax1 = RATE_LO * income;
    }
    else
    {
        tax1 = RATE_LO * LO_SINGLE_LIMIT;
        tax2 = RATE_HI * (income - LO_SINGLE_LIMIT);
    }
}
else
```

```
    {
        if (income <= LO_MARRIED_LIMIT)
        {
            tax1 = RATE_LO * income;
        }
        else
        {
            tax1 = RATE_LO * LO_MARRIED_LIMIT;
            tax2 = RATE_HI * (income - LO_MARRIED_LIMIT);
        }
    }

    double total_tax = tax1 + tax2;

    cout << "The tax is $" << total_tax << endl;
    return 0;
}
```

What just happened?

- We learned about **nested if/else statements**!
- … and **switch** statements!
- We learned about the dangers of **multiple alternatives!**

  … and how the order of tests often matters
- We learned about **hand-tracing**!
- We even learned a little bit about the U.S. tax code!