Decision Structures
The Turing test, developed by Alan Turing in 1950, is a test of a machine’s ability to exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human.
Figure 1: Alan Turing, one of the great Computer Scientists. You may have heard of something called the Turing Test.

“Sometimes it’s the people no one imagines anything of who do the things that no one can imagine.”

Alan Turing
How do computers think?
Turing Test

Understanding how computers (appear) to make decisions.

- Can a human make the distinction between the difference of talking to a human or a robot?
- A silly question maybe a few years ago, but we’re getting closer.
- So how do computers appear to make decision?
Decision Structures
First Tool: Selection Statements
Can a human make the distinction between the difference of talking to a human or a robot?

A silly question maybe a few years ago, but we’re getting closer.

So how do computers appear to make decision?
if-statement semantics

Listing 1: Conceptual example of an if-statement
#include <iostream>

int main()
{
    int x = 5;
    if (x > 3)
    {
        std::cout << "x is greater than 3";;
    }

    return 0;
}

Listing 2: First if-statement with a greater than comparison operator
#include <iostream>

int main()
{
    if (true)
    {
        std::cout << "the conditional is true";
    }
    return 0;
}

Listing 3: if-statement bool forcing execution of code block
```cpp
#include <iostream>

int main()
{
    if (false)
    {
        std::cout << "the code between the curly braces will not execute";
        std::cout << "this is because the conditional is false";
    }
    return 0;
}
```

**Listing 4:** Not executing code block—essentially dead code in this case
#include <iostream>

int main() {
    bool condition = false;
    if (condition) {
        std::cout << "the code between the curly braces will not execute";
        std::cout << "this is because the conditional is false";
    }
    return 0;
}

Listing 5: Evaluating a variable. Code will execute depending on variable state
Some operators in conditional statements

1. A is greater than B
2. A is less than B
3. A is greater than or equal to B
4. A is less than or equal to B
5. A is equal to B

There are other operators you can look up here:
http://www.cplusplus.com/doc/tutorial/operators/

Listing 6: C++ syntax for expression. Remember C++ is case sensitive.
if (A==B)

== is for equality
It is a very common mistake to write (A=B) within a condition, which evaluates to true, because one equals sign means assignment. Be careful!

= is for assignment
We learned this when we created variables.

1 e.g.
2 int x = 5;
3 x = 5 + 7;
4 x == 5+7; // This evaluates to true, but does not do anything useful alone.

Listing 7: The number of equals signs matters!
if/else statement

Listing 8: Execute 1 block of code or do not execute it

if ( a > b ) {
    Do something
}
// Execute any remaining code

Listing 9: Choose 1 block of code or another

if ( a > b ) {
    Do something
}
else {
    // Always execute this block if the above condition evaluates to false
    // In this case, a <= b must be true
}

// Execute any remaining code
if/else if statement

```python
if ( a > b ){
    Do something, because a is greater than b
}
else if ( a == b )
{
    // Execute this block conditionally. If the above condition evaluates to false
    // In this case, a <= b must be true
}
// execute more code below

Listing 10: Conditionally execute one block or the other
```
if/else if/else statement

```java
if ( a > b ){
    Do something, because a is greater than b
}
else if ( a == b )
{
    // Execute this block conditionally. If the above condition evaluates to false
    // In this case, a <= b must be true
}
else{
    // Always execute this code
}
```

**Listing 11:** Three possible blocks of code could execute
Second Tool: Iteration
while

```
#include <iostream>

int main()
{
    int x = 10;
    while (x > 0)
    {
        std::cout << x << " \n";
        x = x - 1;
    }
}
```

**Listing 12:** While loop example
#include <iostream>

int main()
{
    int x = 10;
    while (x > 0)
    {
        std::cout << x << "\n";
        // x = x - 1; What happens if I do not decrement this value?
    }
}

Listing 13: Infinite Loop
Break and Continue

Listing 14: Stopping and continuing control flow

```cpp
#include <iostream>

int main()
{
    int x = 10;
    while (x > 0)
    {
        std::cout << x << " \n";
        if (x == 5)
        {
            break;
        }
    }
    return 0;
}
```
range based for-loop - iterate through entire collection

```cpp
#include <iostream>
#include <string>
int main() {
    std::string s = "abcdefg";
    for (char c : s)
        std::cout << c << "\n";
    return 0;
}

Listing 15: Iteration over a collection
```
for (a subset of data)

```cpp
#include <iostream>
#include <string>

int main()
{
    std::string s = "abcdefg";
    for (int i = 0; i < 3; ++i)
    {
        std::cout << s.at(i) << "\n";
    }
    return 0;
}
```

**Listing 16:** iteration using an index
The AND and OR statements in C++ are important for creating more powerful control flow.

- AND
- OR
P: I have a drivers license
Q: I am at least 16 years old.
You can drive if you have a drivers license and are at least 16 years old.

\[
\begin{array}{c|c|c}
P & Q & P \text{ And } Q \\
\hline
T & T & T \\
T & F & F \\
F & T & F \\
F & F & F \\
\end{array}
\]
P: I am over 65 years old Q: I have a Student ID
You can get a discount at the movie theaters if you are over 65 OR have a student ID.

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P OR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
And and Or Example

```cpp
if ( x > 1 && x < 10)
{
    std::cout << "x is greater than 1 and less than 10 \n";
}

if ( x > 0 || x < 0)
{
    std::cout << "x can not be 0 \n";
}
```

Listing 17: AND and OR example for making more complex conditional statements
Scope
• It is worth noting that we have seen a lot of curly braces.
• The curly braces show the scope of our variables.
• The curly braces also separate what code gets executed, based on a conditional statement.
#include <iostream>

int main() {
    int x = 0;
    if (x == 0) {
        // i is locally declared and defined in a block
        int i = 5;
        std::cout << "x is equal to 0, and i = " << i << std::endl;
    }
    return 0;
}

Listing 18: Creating a new variable local to a block of code
#include <iostream>

int main()
{

    int x = 0;
    if (x==0){
        // i is locally defined
        int i = 5;
        std::cout << "x is equal to 0, and i = " << i << std::endl;
    }

    std::cout << "x = " << x << std::cout;
    // Is i within scope at this point?
    std::cout << "i = " << i << std::cout;

    return 0;
}

Listing 19: i is not defined in scope
Scope Example 3 – Fixing error

#include <iostream>

int main()
{
    int x = 0;
    // Move i's declaration outside of control block
    int i = 5;
    if (x == 0)
    {
        std::cout << "x is equal to 0, and i = " << i << std::endl;
    }
    std::cout << "x = " << x << std::cout;
    std::cout << "i = " << i << std::cout;

    return 0;
}

Listing 20: TBD
Activity Discussion
Review of what we learned

- (At least) Two students
- Tell me each 1 thing you learned or found interesting in lecture.
5-10 minute break
To the lab!


\footnote{You should have gotten an e-mail and hopefully setup an account at https://www.eecs.tufts.edu/~accounts prior to today. If not—no worries, we'll take care of it during lab!}