UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*

The structured programming theorem

Douglas Wilhelm Harder, M.Math.
Prof. Hiren Patel, Ph.D.
hiren.patel@uwaterloo.ca dwharder@uwaterloo.ca

Outline

- In this lesson, we will:
  - Review the statements we have seen to this point
  - Look at some very ugly flow charts apparently implementable only with a *goto* statement
  - Review theorems and present the structured programming theorem
  - Look at some simple consequences based on flow charts
  - See how to use the structured programming theorem when designing algorithms

Background

- You've seen used and observed numerous applications, some of which may amaze you
  - You ask yourself, how can you do something like this?
  - The answer: Libraries and the *structured programming theorem*

Joust, Williams Electronics

Fortnite, Epic Games

Background

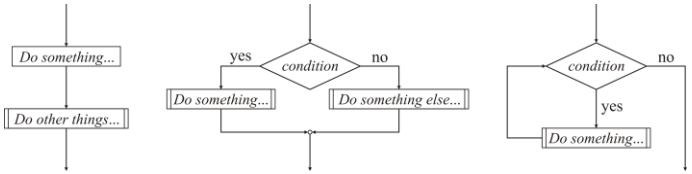
- The C++ language is very complex, and for many and good reasons
  - There are many features designed to support the development of software
  - The C++ standard costs 198 Swiss Francs (~266 CAD) and is over 1600 pages
  - Most of you will never master the language—and that's okay
    - No instructor has, either…

Background

- In your mathematics courses, you have seen a few theorems:
  - A theorem is a statement must be true if a given set of premises are true
- Examples:
  - Euclid's theorem
    There are infinitely many prime numbers
  - The Pythagorean theorem
    If a, b, and c are sides of a right-angled triangle, and c is the side opposite the right angle, then $a^2 + b^2 = c^2$
  - The remainder theorem
    If a polynomial $p(x)$ is divided by $(x - r)$, the remainder is $p(r)$
  - The factor theorem
    A polynomial $p(x)$ has a factor $(x - r)$ if and only if $p(r) = 0$

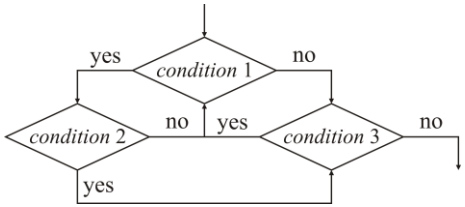
The flow charts so far

- To this point, we have described:
  - Blocks of statements `{statement; …}`
  - Conditional statements `if ( condition ) {…} else {…}`
  - while loops `while ( condition ) {…}`



The flow charts so far

- It is possible, when designing algorithms, to come up with flow charts that cannot be built up from these three statements
- For example, consider:



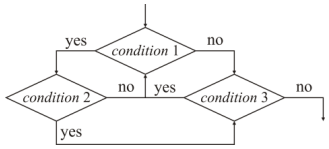
The flow charts so far

- It is always possible to use `goto` statements to implement any flow chart

```
first: if ( condition-1 ) {
    goto second;
} else {
    goto third;
}

second: if ( condition-2 ) {
    goto third;
} else {
    goto first;
}

third: if ( condition-3 ) {
    goto first;
}
```



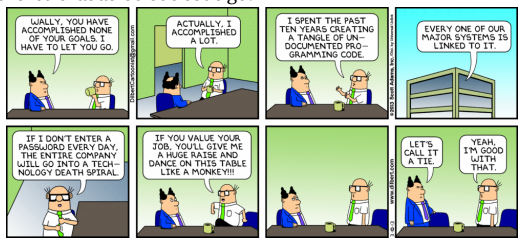
Identifiers can be used as labels

The flow charts so far

- The use of the `goto` statement leads to serious maintenance issues
  - Code becomes more difficult to understand, analyze, debug, extend or otherwise maintain
  - Your costs will go up, your profits will go down

The flow charts so far

- Worse yet: you will never be able to let those programmers who authored that atrocious code go:



- Worst of all: any solution to an assignment or examination question that contains a `goto` statement will receive a grade of 0

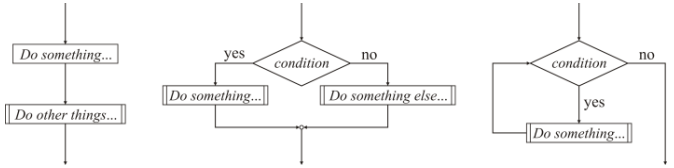
It's rocket science…

- Even the Jet Propulsion Laboratory (JPL) has included this rule:

  Rule 11 (simple control flow)
  The **`goto`** statement ***shall not*** be used.

The structured programming theorem

- Fortunately, the following theorem says we don't need any more:
  - Any program that can be written can be written using flow charts that combine only the three statements we have seen to this point



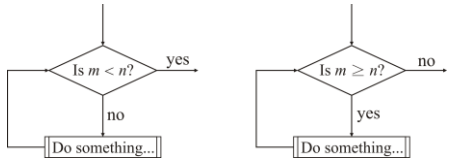
The structured programming theorem

- Consequences:
  - You understand most of the tools already
  - When you learn a new programming language, first understand how to write:
    - conditional statements, and
    - while loops
- When you tackle a problem that require the repetition of a given set of operations, think in terms of a while loop:

  "We must perform this set of instructions while this condition is true"

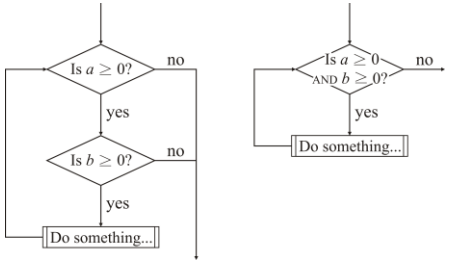
Equivalent flow charts

- These flow charts are equivalent
  - The left cannot be implemented using a while loop, the right can



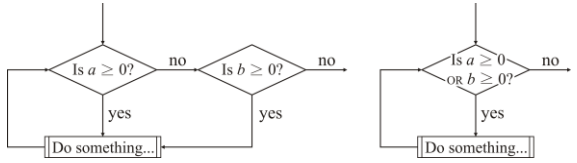
Equivalent flow charts

- These flow charts are also equivalent
  - The left cannot be implemented using a while loop, but using the logical AND operator, the right can



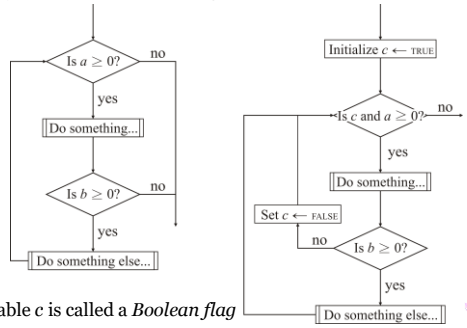
Equivalent flow charts

- These flow charts are also equivalent
  - The left cannot be implemented using a while loop, but using the logical OR operator, the right can



Equivalent flow charts

- These flow charts are also equivalent
  - The left cannot be implemented using a while loop, but using a temporary Boolean variable, the right can



  - The variable *c* is called a *Boolean flag*

Equivalent implementations

- Compare these implementations of an algorithm for sorting an array:

```
void structured( double array[], size_t capacity ) {
    while ( capacity > 0 ) {
        size_t n = 0;
        size_t k = 1;
        double max = array[0];

        while ( k < capacity ) {
            if ( max < array[k] ) {
                array[k - 1] = max;
                max = array[k];
            } else {
                array[k - 1] = array[k];
                n = k;
            }

            ++k;
        }

        array[k - 1] = max;
        capacity = n;
    }
}
```

```
void unstructured( double array[], size_t capacity ) {
    start: if ( capacity == 0 ) {
        return;
    }

    size_t n = 0;
    size_t k = 1;
    double max = array[0];

    next: if ( k == capacity ) {
        array[k - 1] = max;
        capacity = n;
        goto start;
    }

    if ( max < array[k] ) {
        array[k - 1] = max;
        max = array[k];
    } else {
        array[k - 1] = array[k];
        n = k;
    }

    ++k;
    goto next;
}
```


Moral of the story...

- You will not be required to seriously investigate the structured programming theorem or to prove two flow charts are equivalent
- Use this as a guide to designing algorithms:
  - Always think in terms of looping while some condition is true
  - Understand when you need to terminate the loop
  - Always ask Boolean-valued questions, either for while loops or conditional statements

The flow charts so far

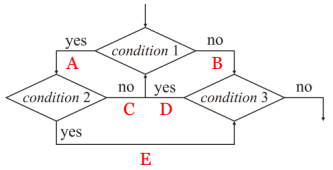
- To go back to our first example:

```
bool flag_1{ true };

while ( flag_1 ) {
    bool flag_2{ false };

    if ( condition-1 ) {
        // Block A
        if ( condition-2 ) {
            // Block E
            flag_2 = true;
        } else {
            // Block C
        }
    } else {
        // Blcok B
        flag_2 = true;
    }

    if ( flag_2 ) {
        if ( condition-3 ) {
            // Block D
        } else {
            flag_1 = false;
        }
    }
}
```




```
first: if ( condition-1 ) {
    // Block A
    goto second;
} else {
    // Block B
    goto third;
}

second: if ( condition-2 ) {
    // Block E
    goto third;
} else {
    // Block C
    goto first;
}

third: if ( condition-3 ) {
    // Block D
    goto first;
}
```

Summary

- Following this lesson, you now:
  - Understand that you should never use a goto statement
  - Understand the consequences of the structured programming theorem
    - Know that conditional and while loops are all you need
  - Understand that flow charts can be rewritten to allow them to be implemented using these statements
    - We saw logical operators and Boolean flags

References

[1] Wikipedia
https://en.wikipedia.org/wiki/Structured_program_theorem

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.



Disclaimer