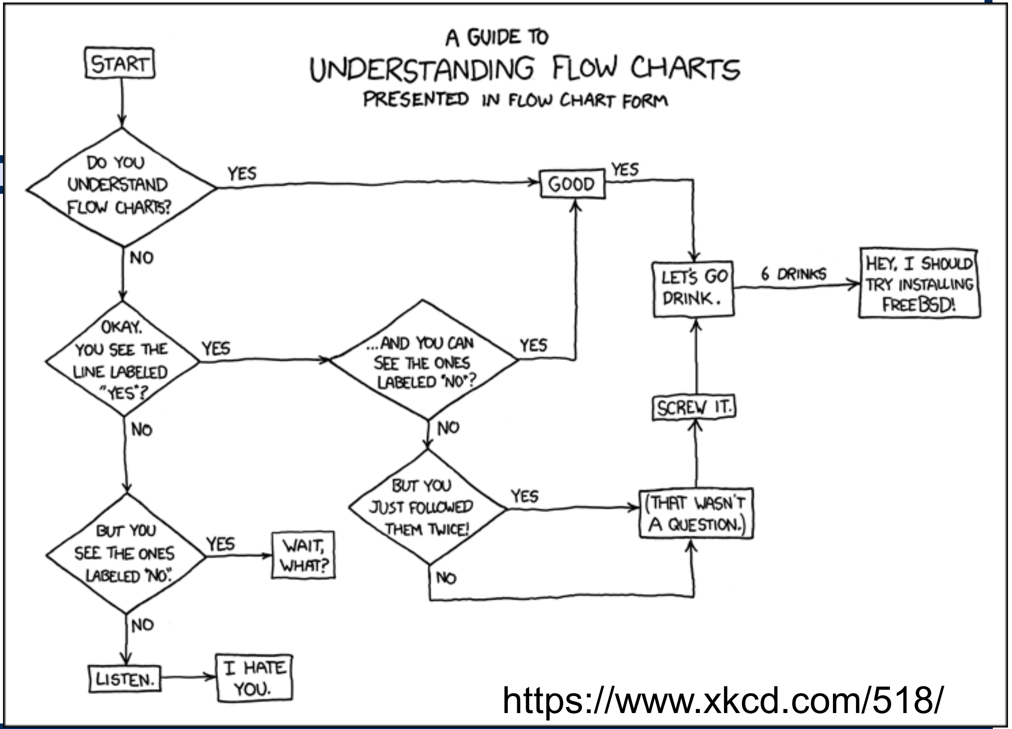
CSCI 200: Foundational Programming Concepts & Design
Lecture 04


A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
NO
NO
SCREW IT.
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
NO
NO
LISTEN.
I HATE YOU.

https://www.xkcd.com/518/

Conditionals

Find a Partner!

Open Canvas, VS Code & iClicker Open

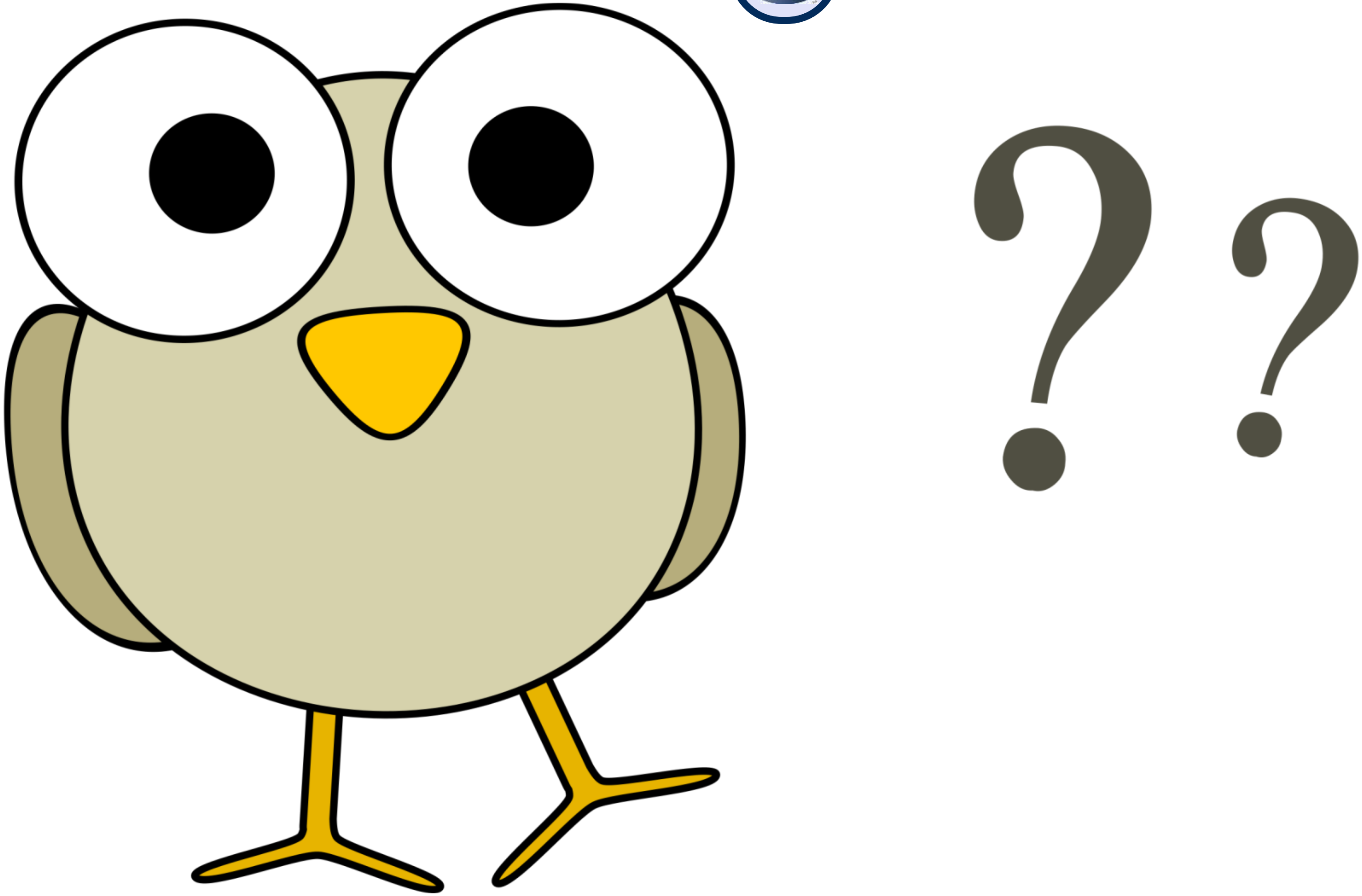
Download today’s starter code

Previously in CSCI 200

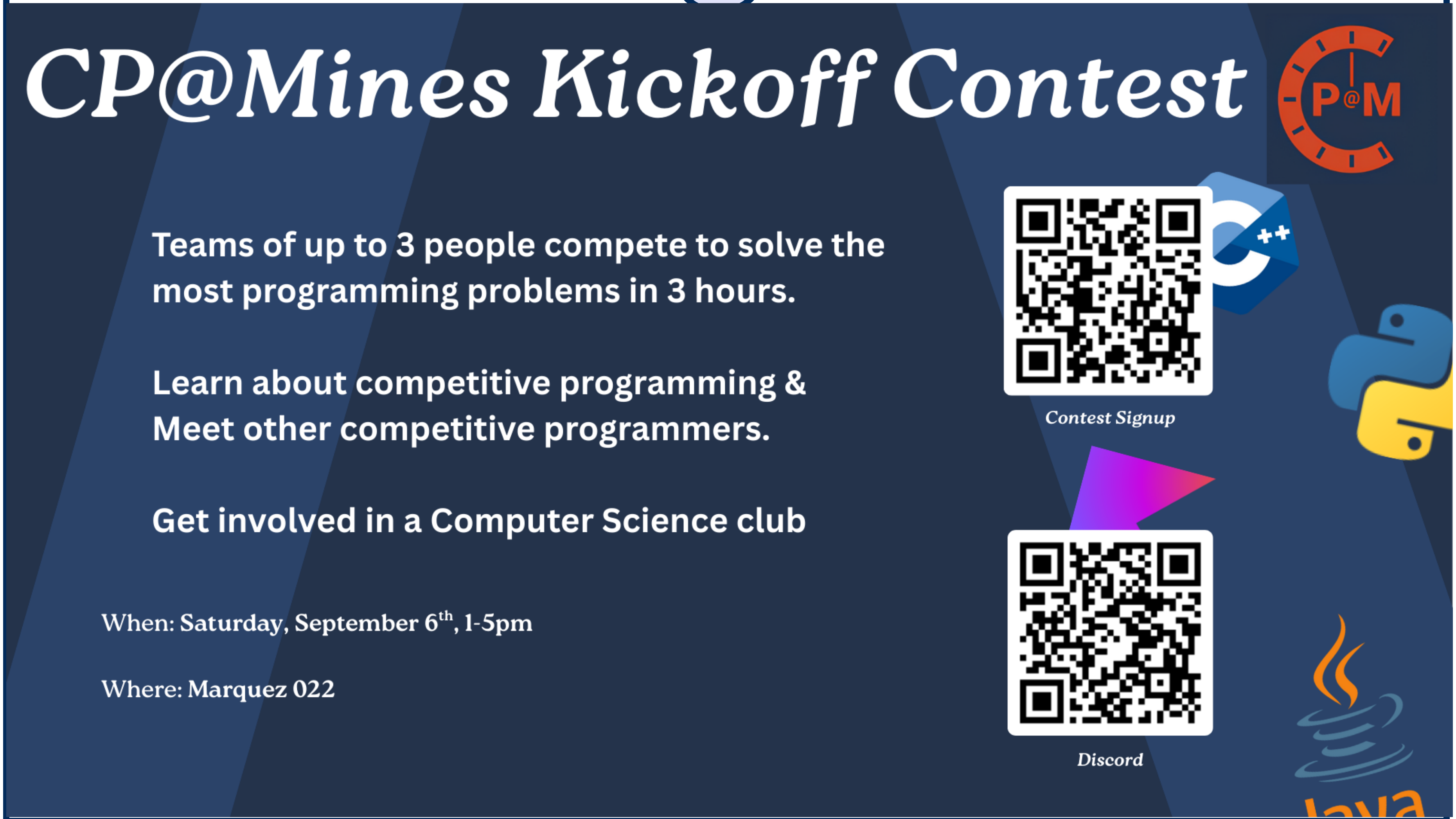
- Build Process
  - Compile each cpp file separately
  - Link all object files and libraries

- Makefile
  - Batches build commands
  - Only rebuilds files that have changed since last build

Questions?



Announcement



CP@Mines Kickoff Contest

Teams of up to 3 people compete to solve the most programming problems in 3 hours.

Learn about competitive programming & Meet other competitive programmers.

Get involved in a Computer Science club

When: Saturday, September 6th, 1-5pm

Where: Marquez 022

Contest Signup

Discord

Random Quiz

- Make Canvas Full Screen
- Put everything else away
- Access Code:
- 4 Minutes

Learning Outcomes For Today

- Identify C++ control structures and conclude which branch a sample program will execute.
- List C++ logic operators and evaluate Boolean expressions consisting of multiple logic operators.
- Evaluate the resultant output of a code block containing a control structure.

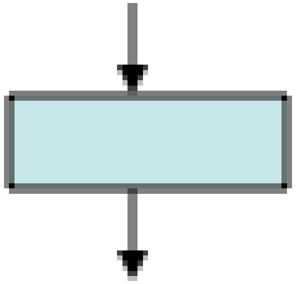
On Tap For Today

- Program Flow
- Programming Tips
- Scope
- Practice

On Tap For Today

- Program Flow
- Programming Tips
- Scope
- Practice

Statement Types



Sequential

Imperative Programming

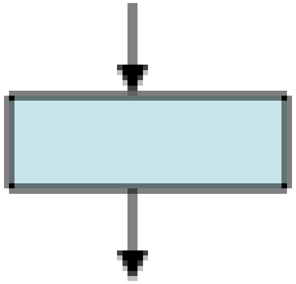
- Explicit sequence of steps to perform one at a time
  - Shows how the computation takes place

- Each step changes the **state** of the program
  - **state** comprised of stack information
    - Current line of execution
    - Variables that are in scope

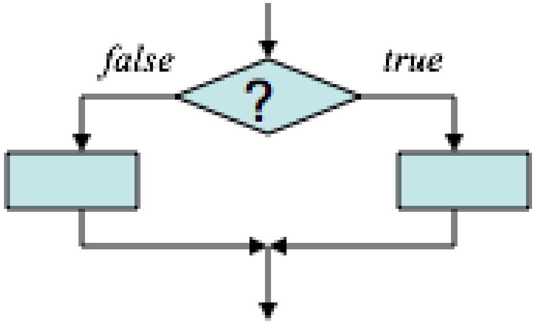
Imperative Programming

```
int main() {
  int sum = 0;
  sum += 1;
  sum += 2;
  sum += 3;
  sum += 4;
  sum += 5;
  sum += 6;
  sum += 7;
  sum += 8;
  sum += 9;
  sum += 10;
  cout << "The sum is: " << sum << endl;
  return 0;
}
```

Statement Types



Sequential



Conditional

Structured Programming

- Imperative Programming where flow is defined by control structures (e.g. conditionals, loops)

```
int main() {
  int sum = 0;
  cout << "Enter sum:  ";
  cin >> sum;
  if(sum > 0) {
    cout << "Sum is positive" << endl;
  } else {
    cout << "The sum is:  " << sum << endl;
  }
  return 0;
}
```

Turing Machine

- Given infinite time and memory, if a machine has the following features:
  1. Sequence
  2. Control
  3.
  4. Output
  5. Input
  6. Variables
- It can solve any mathematical problem

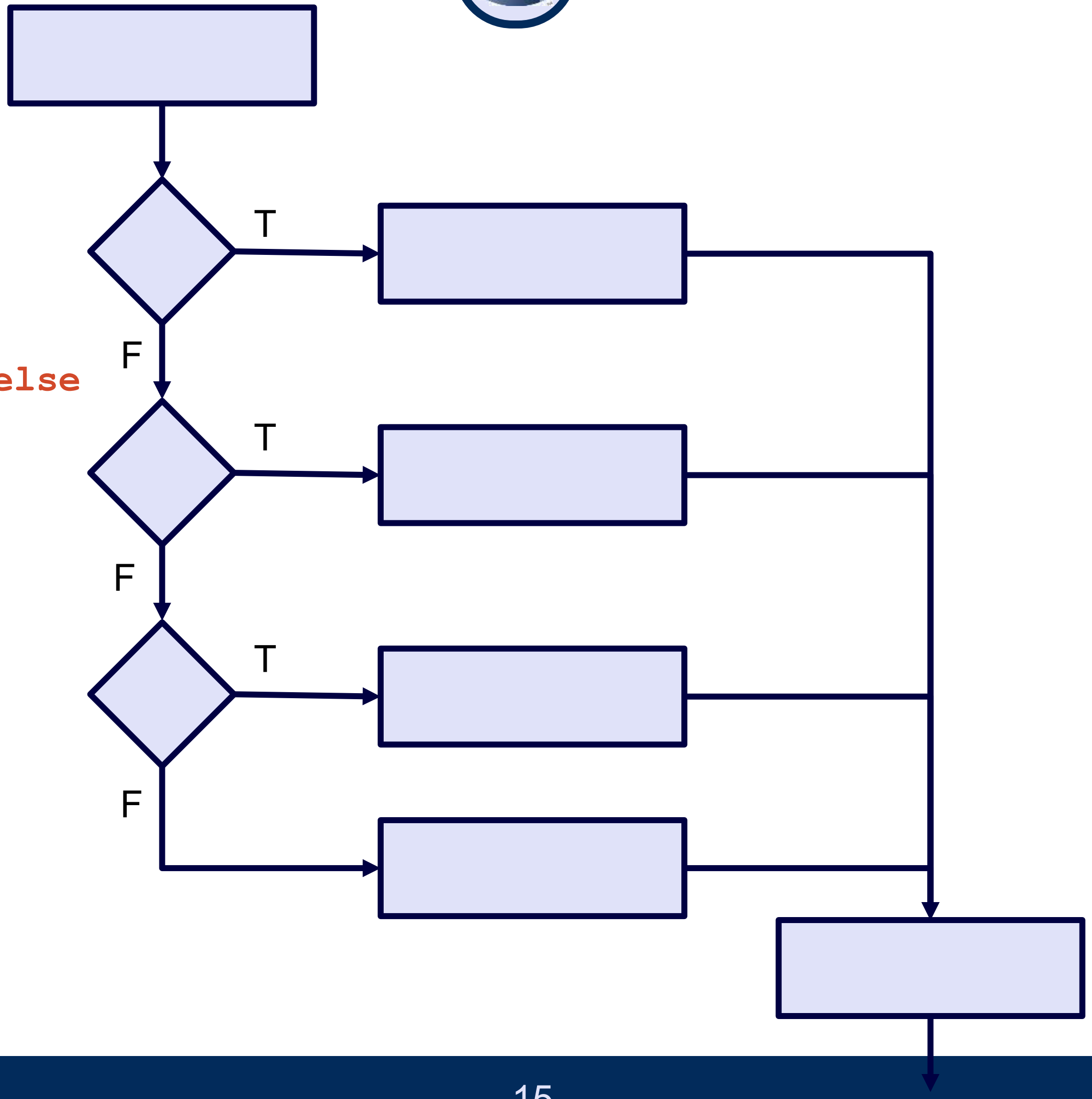
Program Flow

A. `if`

B. `if` / `else`

C. `if` / `else if` / `else`

D. `switch`



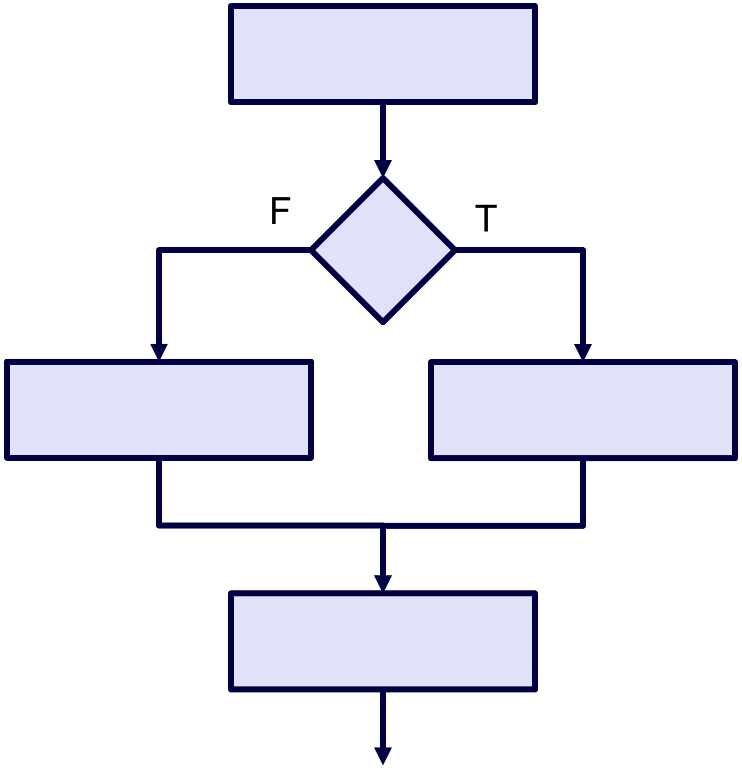
Program Flow

A. if

B. if / else

C. if / else if / else

D. switch



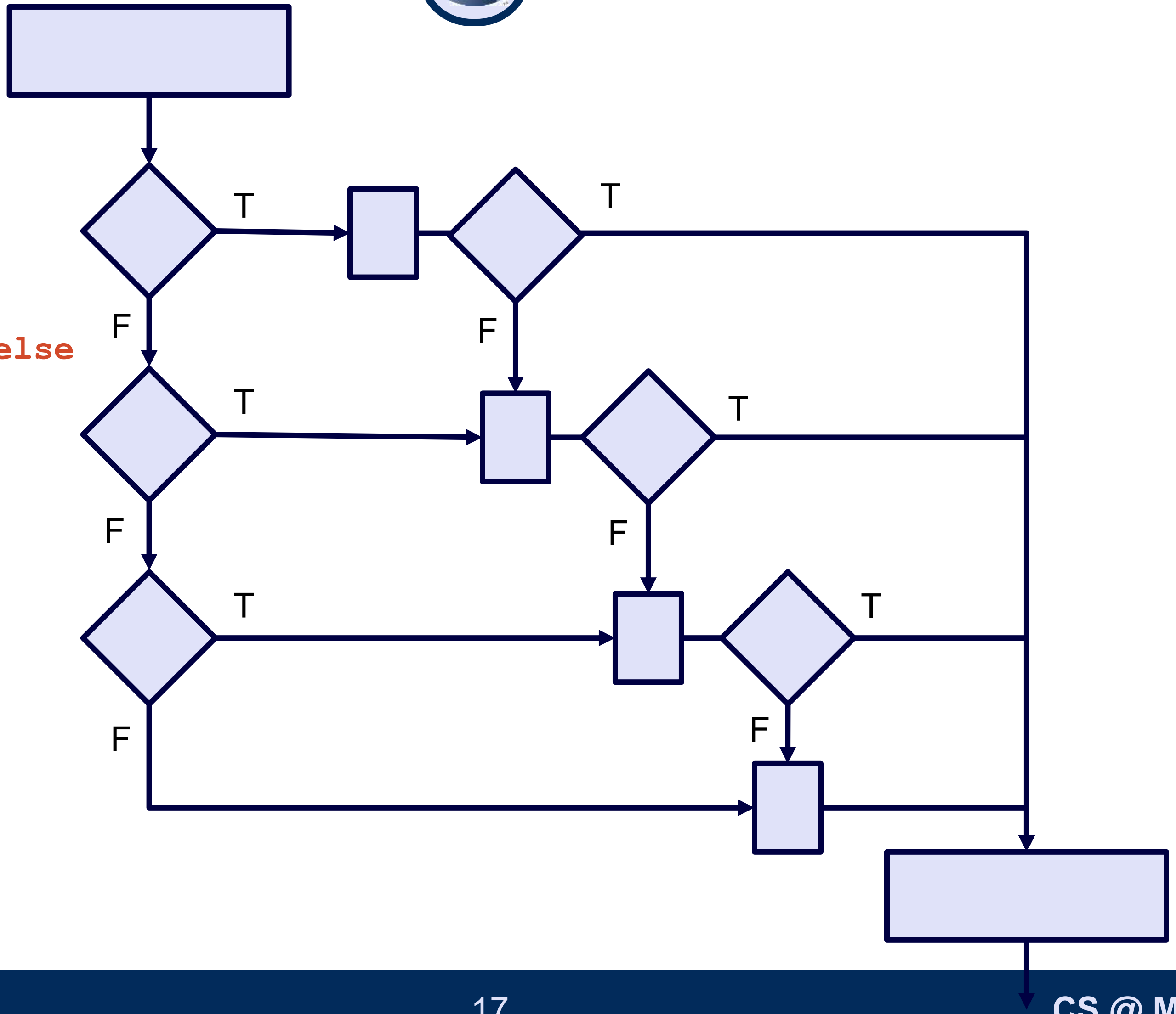
Program Flow

A. `if`

B. `if` / `else`

C. `if` / `else if` / `else`

D. `switch`



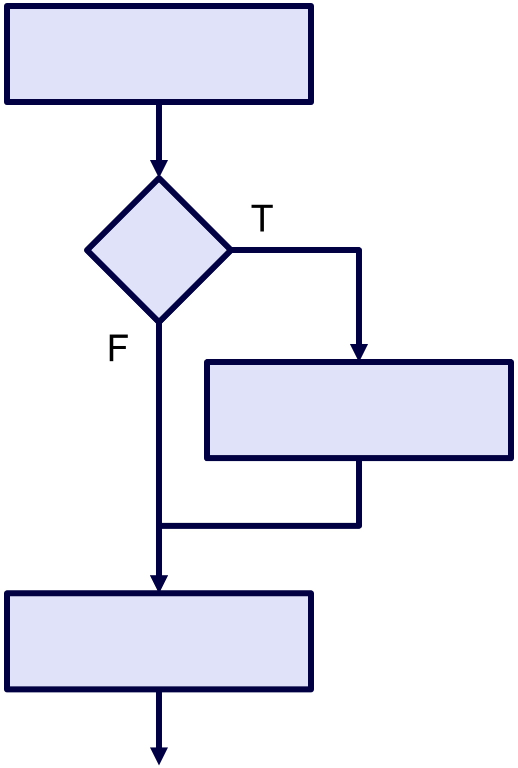
Program Flow

A. `if`

B. `if` / `else`

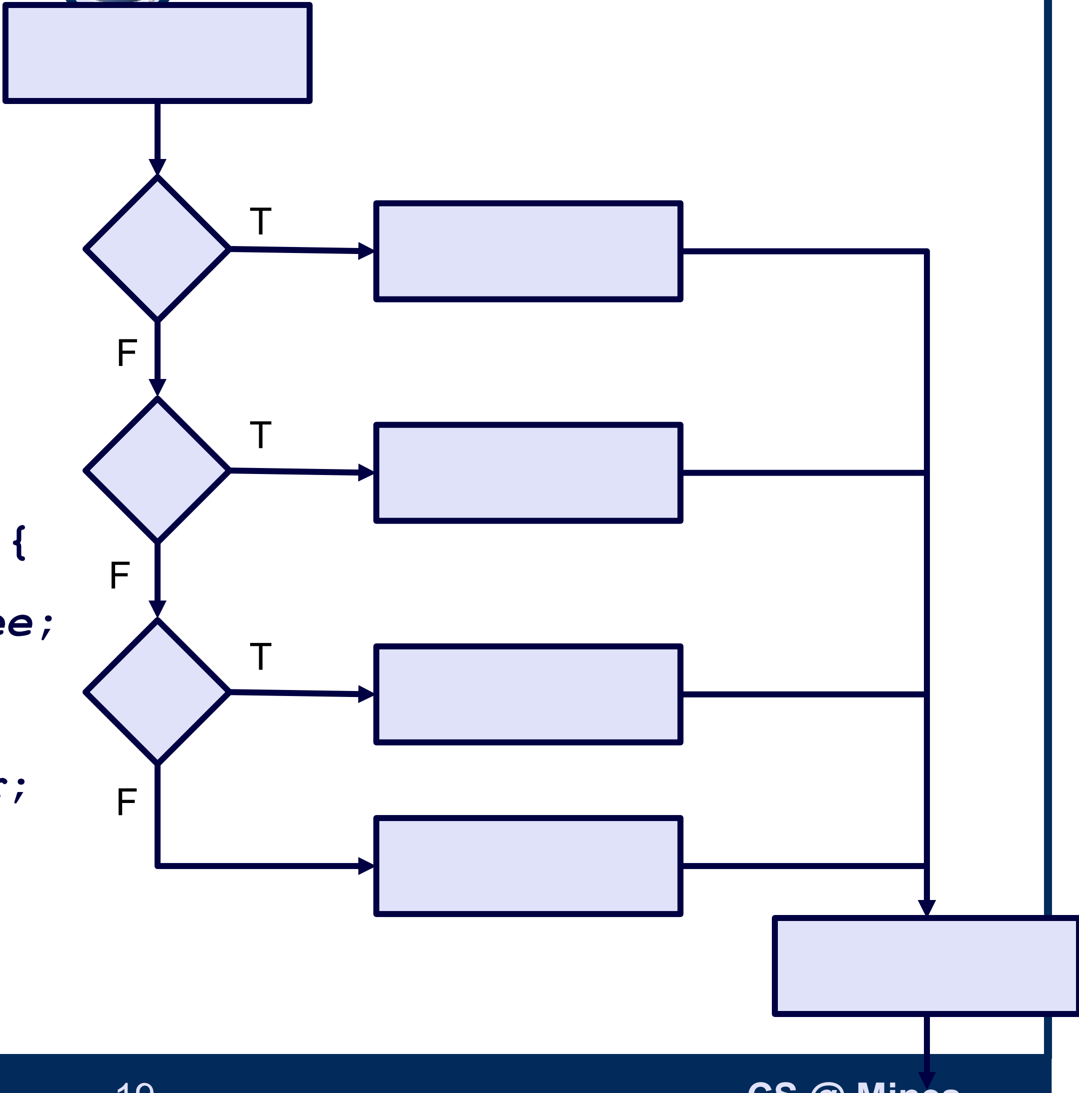
C. `if` / `else if` / `else`

D. `switch`



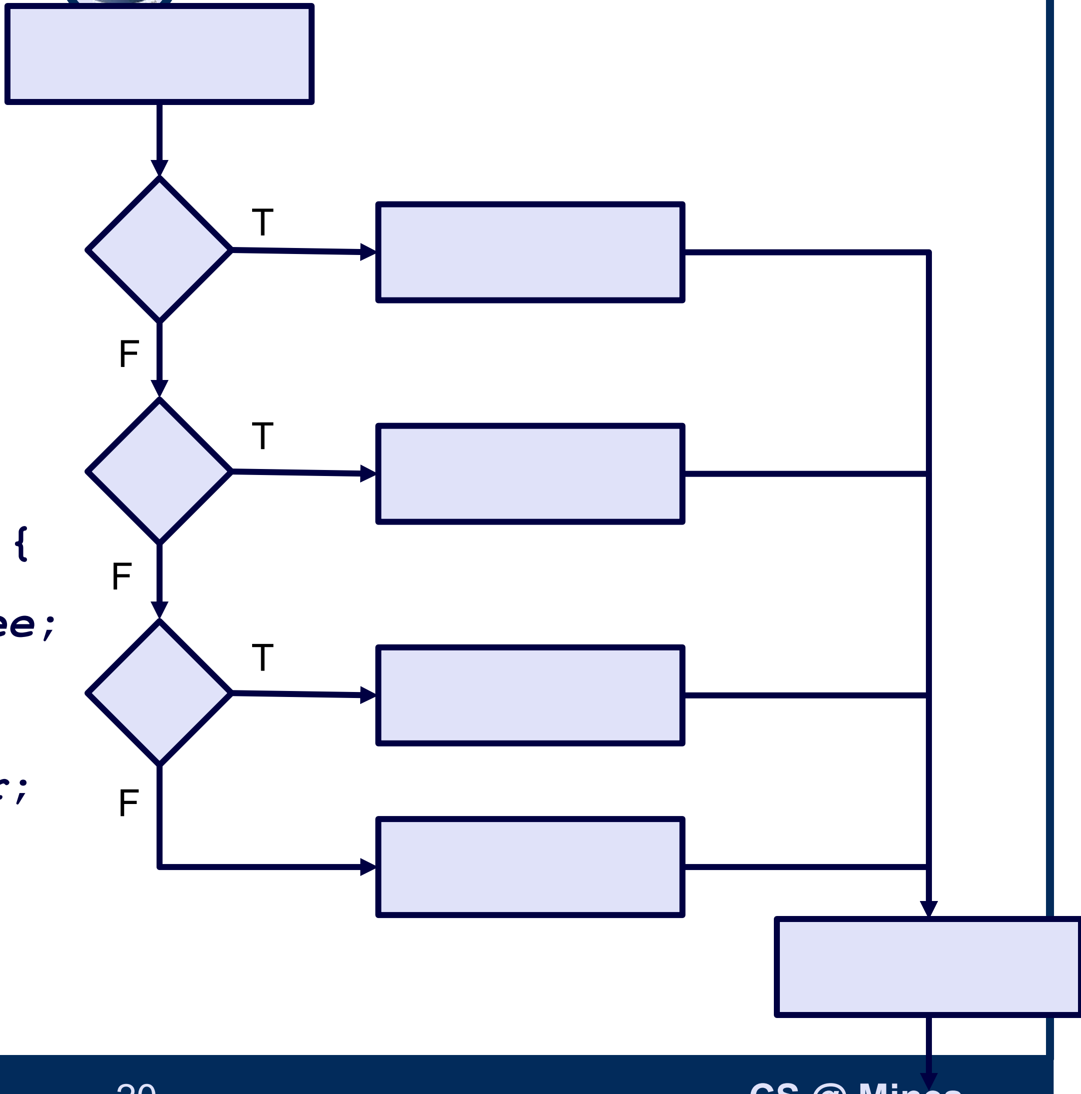
else if Fun Fact!

```
if( condition1 ) {
    statementOne;
} else {
    if( condition2 ) {
        statementTwo;
    } else {
        if( condition7 ) {
            statementThree;
        } else {
            statementFour;
        }
    }
}
```



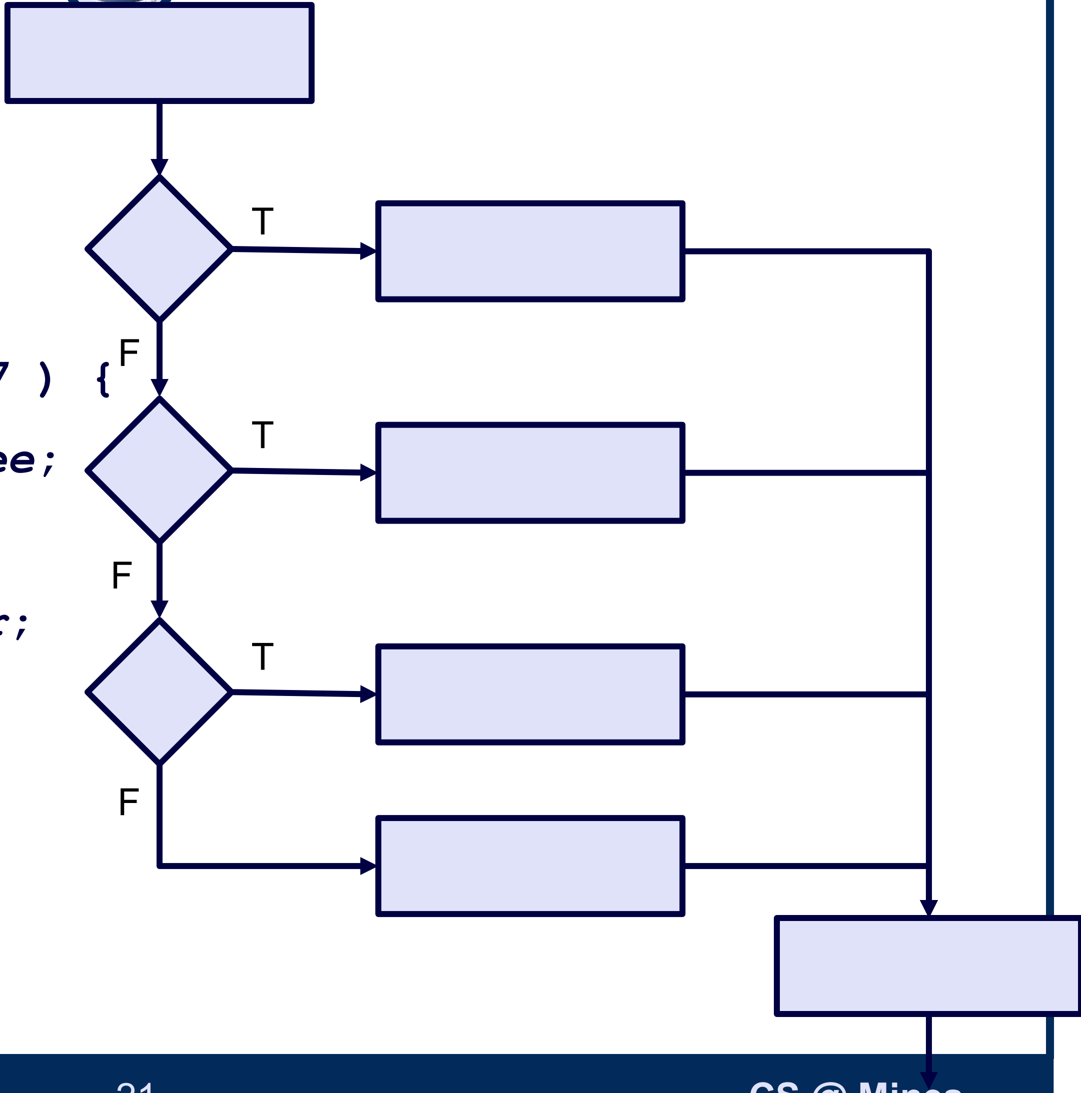
else if Fun Fact!

```
if( condition1 ) {
    statementOne;
} else
    if( condition2 ) {
        statementTwo;
    } else
        if( condition7 ) {
            statementThree;
        } else {
            statementFour;
        }
```



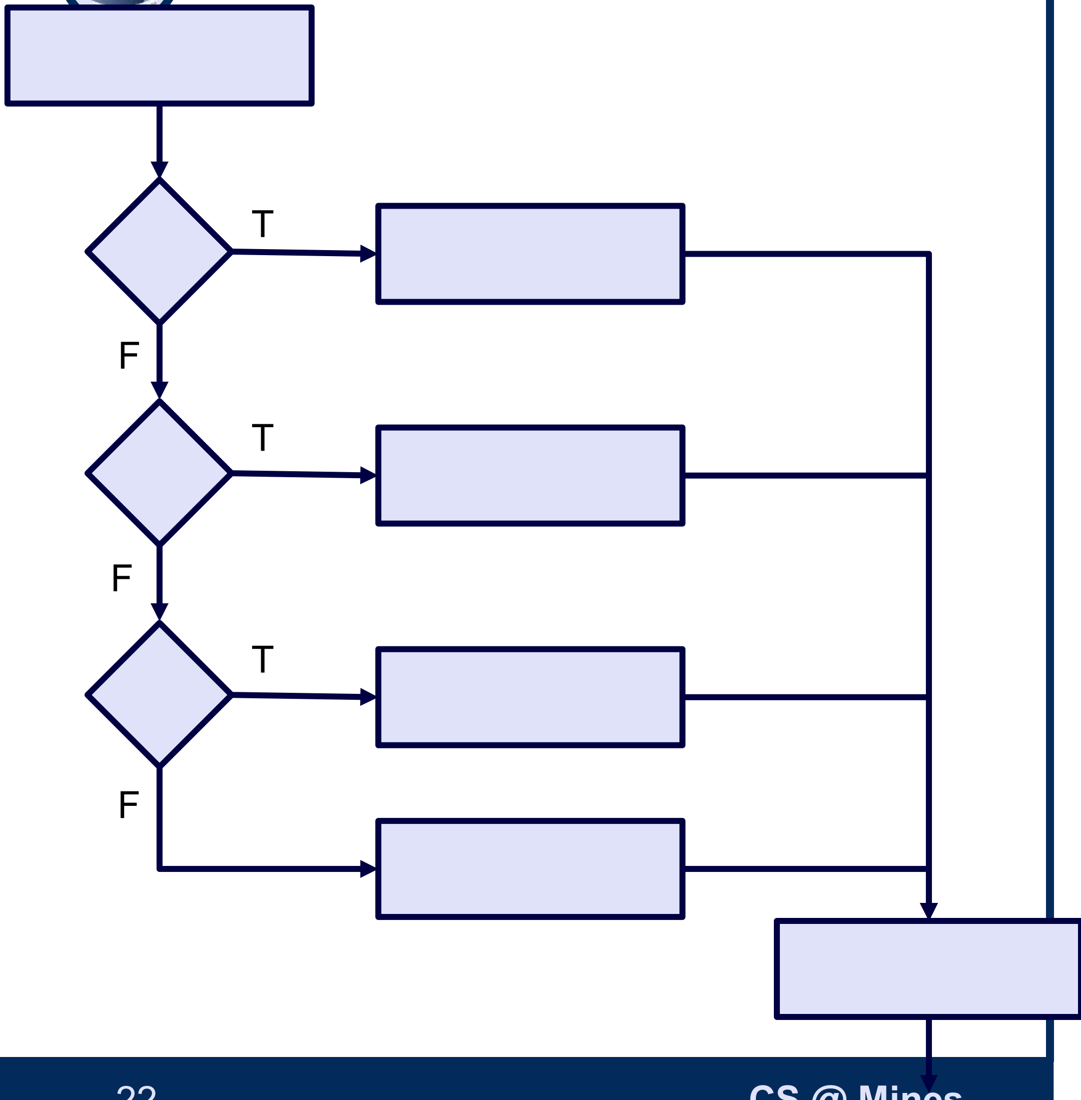
else if Fun Fact!

```
if( condition1 ) {
    statementOne;
} else if( condition2 ) {
        statementTwo;
    } else if( condition7 ) {
            statementThree;
        } else {
            statementFour;
        }
```



else if Fun Fact!

```
if( condition1 ) {
    statementOne;
} else if( condition2 ) {
    statementTwo;
} else if( condition7 ) {
    statementThree;
} else {
    statementFour;
}
```



On Tap For Today

- Program Flow
- Programming Tips
- Scope
- Practice

Practice!

```
int x = 0, y = 3, z = 3;
```

`1. y <= 3 + x;`	T
`2. z != y - x;`	F
`3. x < y && x >= z;`	F
`4. (x < z) && (x = y);`	T
`5. x > y || y >= z;`	T
`6. y == z && !(x < y);`	F
`7. !y == 0 || z != 3;`	T
`8. !(y == 0 || z == 3);`	F

Programming Task #1

- Complete Programming Task #1
  - User enters the price of a pie
  - If price is greater than $10, then discount by $3.14
  - Otherwise, discount by $1
  - Display the final discounted price

Sample Input	Expected Out
9	8
10	9
10.25	7.11
15	11.86

if / else Syntax

```
if( piePrice > 10.0 )
    discountedPrice = piePrice - 3.14;
else
    discountedPrice = piePrice - 1.0;


// one statement each
// no code blocks needed
```

if / else Syntax

```
if( piePrice > 10.0 ) {
    discountedPrice = piePrice - 3.14;
} else {
    discountedPrice = piePrice - 1.0;
}
// but is recommended
```

Tip #1!

Style Requirement:

Always Use Braces

if / else Syntax

```
if( piePrice > 10.0 ) {
    discountedPrice = piePrice - 3.14;
} else {
    discountedPrice = piePrice - 1.0;
}
// but is recommended
```

Tip #2!

Style Requirement:

Indent code blocks

Practice - What Gets Printed?

```
int x = 15;
if( x < 20 )
    cout << "apple";
    if( x < 10 )
        cout << "banana";
else
    cout << "orange";
```

White Space Doesn’t Matter

- Dangling `else`
  - `else` always belongs to the closest `if` behind it

```
int x = 15;
if( x < 20 )
    cout << "apple";
    if( x < 10 )
        cout << "banana";
else
    cout << "orange";
```

White Space Doesn’t Matter

- Dangling else
  - else always belongs to the closest if behind it

```
int x = 15;
if( x < 20 )
    cout << "apple";
if( x < 10 )
    cout << "banana";
else
    cout << "orange";
```

Very Important Point

- else always belongs to the closest if behind it
  - Use code blocks { } to denote sections

```
int x = 15;
if( x < 20 ) {
    cout << "apple";
    if( x < 10 ) {
        cout << "banana";
    }
} else {
    cout << "orange";
}
```

Tip #1!

Style Requirement:

Always Use Braces

if / else Syntax

```
if( piePrice > 10.0 ) {
    discountedPrice = piePrice - 3.14;
    cout << discountedPrice;
} else {
    discountedPrice = piePrice - 1.0;
    cout << discountedPrice;
}
```

if / else Syntax

```
if( piePrice > 10.0 ) {
    discountedPrice = piePrice - 3.14;
} else {
    discountedPrice = piePrice - 1.0;
}
cout << discountedPrice;
```

Tip #3!

Avoid duplicated code

Programming Task #2

- Complete Programming Task #2
  - If age within [13, 19], print "Teenager!"
  - Otherwise, print "Not a teen"

Sample Input	Expected Out
`12`	`Not a teen`
`13`	`Teenager!`
`15`	`Teenager!`
`19`	`Teenager!`
`23`	`Not a teen`

Checking Ranges

```
if( 13 <= age <= 19 ) {
    cout << "Teenager!" << endl;
} else {
    cout << "Not a teen" << endl;
}
```

Checking Ranges

```
if( 13 <= age && age <= 19 ) {
    cout << "Teenager!" << endl;
} else {
    cout << "Not a teen" << endl;
}
```

Tip #4!

Compare lower and upper bounds separately

Other Common Errors

```
int a = 27, b = 27, c = 27;

if( a == b == c ) {
    cout << "True!" << endl;
} else {
    cout << "false" << endl;
}
```

~~Other Common Errors~~ FIXED

```
int a = 27, b = 27, c = 27;


if( a == b && a == c ) {
    cout << "True!" << endl;
} else {
    cout << "false" << endl;
}
```

Programming Task #3

- Complete Programming Task #3
  - Run program with sample inputs below to verify results
  - Correct the condition, using `1e-6` as a tolerance
  - See https://bit.ly/comparefloats for reference

Sample Input	Expected Out
`2`	`sqrt(2) squared is 2`
`3`	`sqrt(3) squared is 3`
`4`	`sqrt(4) squared is 4`

Comparing Floats Error

```
double num1 = 0.3 * 3.0 + 1.0;
double num2 = 1.0;


if( num1 == num2 ) {
    cout << "it's true! :)" << endl;
} else {
    cout << "it's false :(" << endl;
}
```

Comparing Floats FIXED

```
double num1 = 0.3 * 3.0 + 1.0;
double num2 = 1.0;
const double EPSILON = 1e-6;

if( fabs(num1 - num2) < EPSILON ) {
    cout << "it's true! :)" << endl;
} else {
    cout << "it's false :(" << endl;
}
```

Tip #5!

Compare floats for equality against an error tolerance

On Tap For Today

- Program Flow
- Programming Tips
- Scope
- Practice

Local Scope

- { } denote a code block
- Variables only exist within that code block
  - Concept of “scope”

Scope Notes

```
int main() {
  int x = 4;, z = 6;
  cout << x << endl;   // prints 4
  int x = 5;           // error! redefinition of x
  if( true ) {
    int z = 2;         // warning, "shadows" prior declaration
    int y = 3;
    cout << z << endl; // prints 2
  }
  cout << z << endl;   // prints 6
  cout << y << endl;   // error! y undeclared
  return 0;
}
```

On Tap For Today

- Program Flow
- Programming Tips
- Scope
- Practice

To Do For Next Time

- Start on Lab1B and A1
- Keep watching pre-class videos