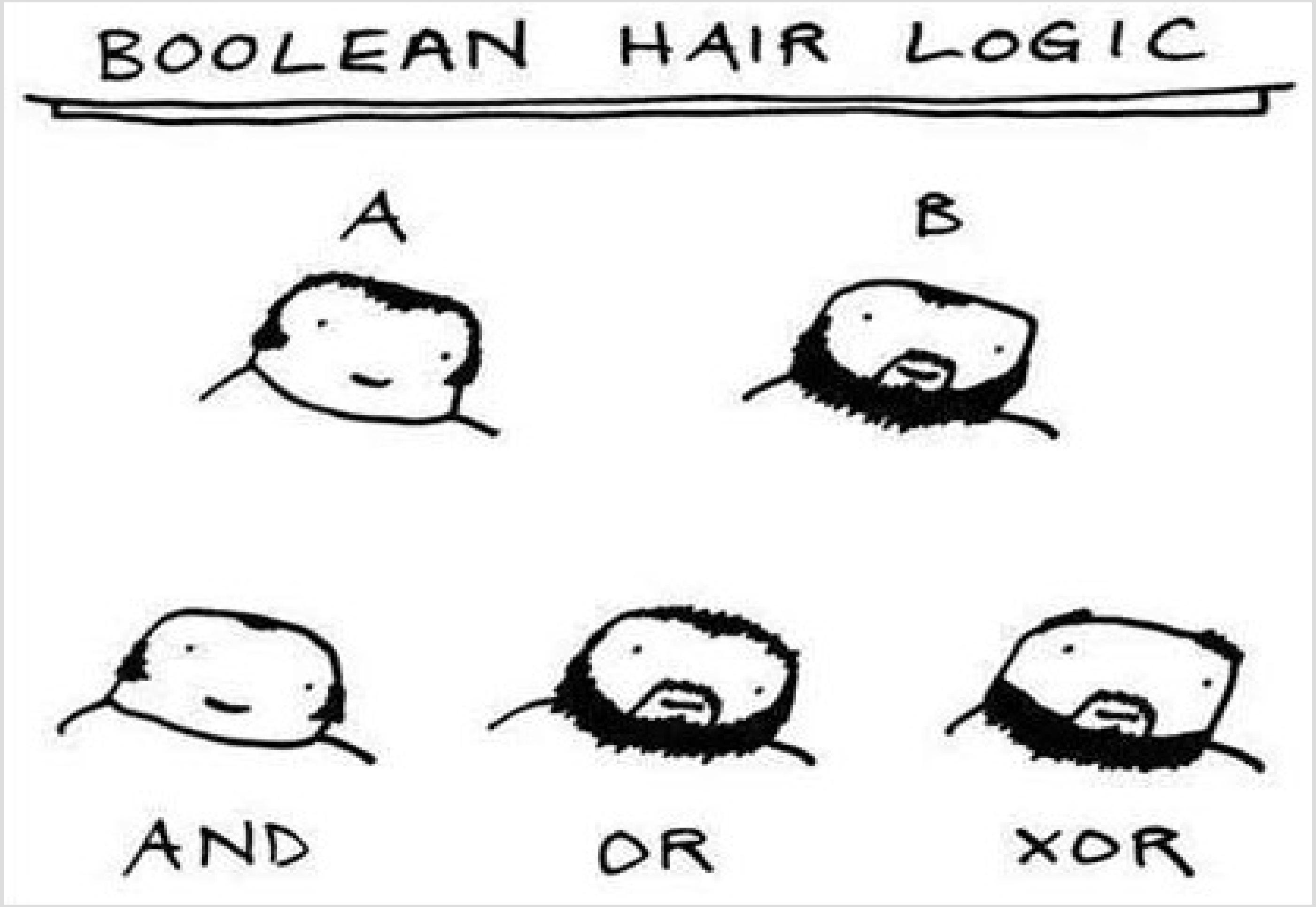

BOOLEAN HAIR LOGIC
A
B
AND
OR
XOR

; and if

Please always put {} after if-statements

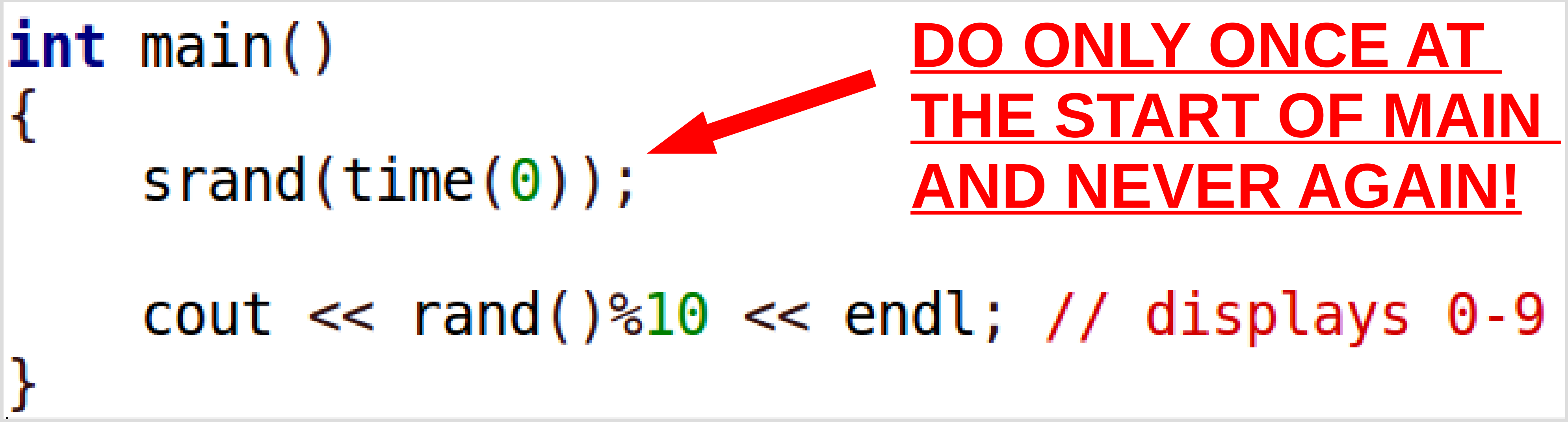
The compiler will let you get away with not putting these (this leads to another issue)

If you do not put {} immediately after an if, it will only associate the first command after with the if-statement
(see: ifAndSemi.cpp)

Random numbers

To use random numbers, you need to do:

1. Run srand(time(0)) once
2. Use rand() to actually generate a number



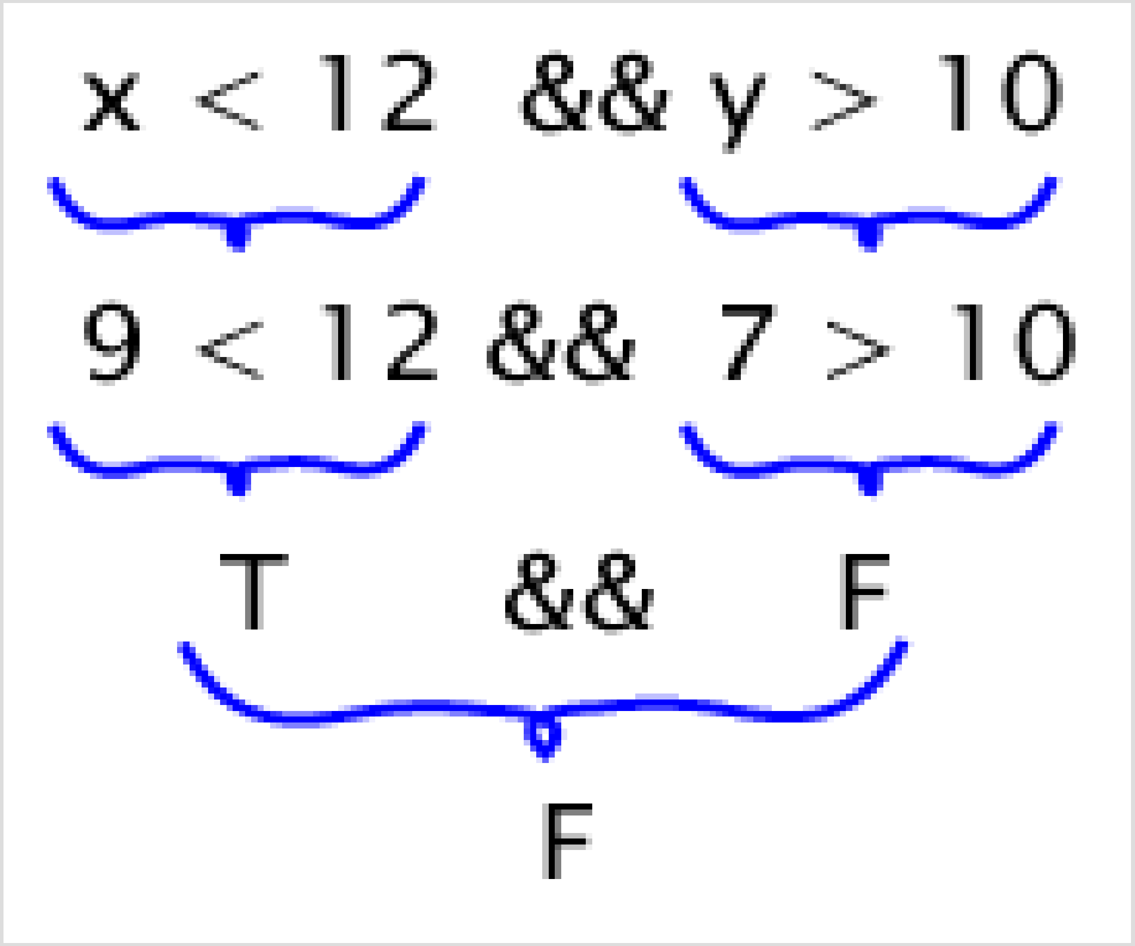
```
int main()
{
    srand(time(0));

    cout << rand()%10 << endl; // displays 0-9
}
```

<u>DO ONLY ONCE AT THE START OF MAIN AND NEVER AGAIN!</u>

(See: rng.cpp)

Complex expressions

```
int x = 9, y = 7;
```



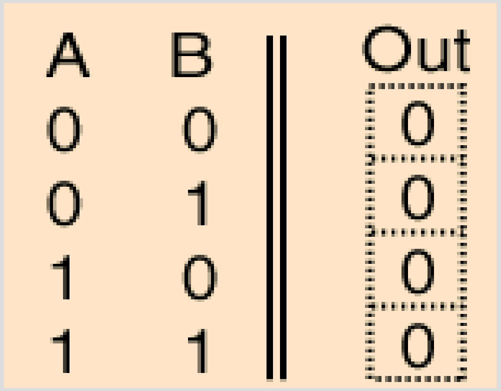
$$\underbrace{x < 12}\ \&\&\ \underbrace{y > 10}$$

$$\underbrace{\underbrace{9 < 12}_{T}\ \&\&\ \underbrace{7 > 10}_{F}}_{F}$$

Complex expressions

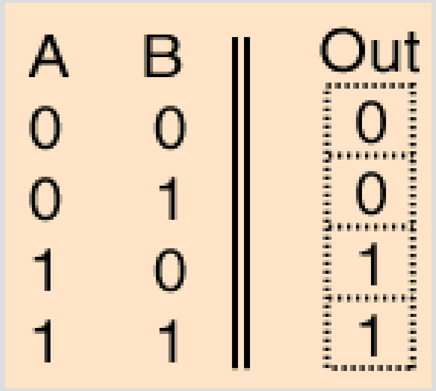
Write boolean expressions for each of the following truth tables:

1.



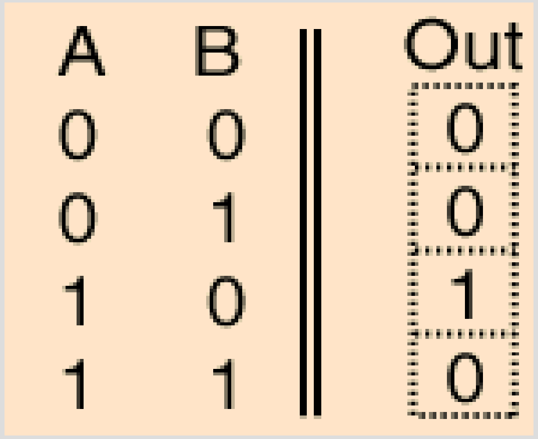
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	0

2.



A	B	Out
0	0	0
0	1	0
1	0	1
1	1	1

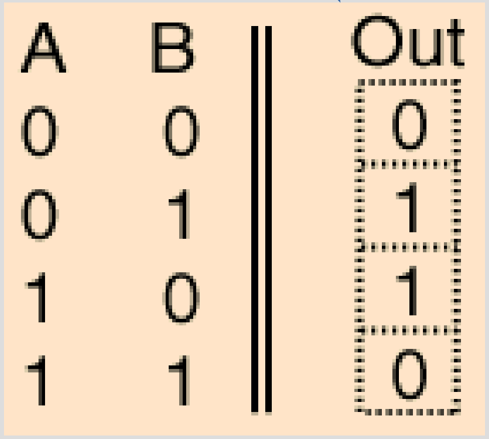
3.



A	B	Out
0	0	0
0	1	0
1	0	1
1	1	0

4.

XOR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Complex expressions

Humans tend to use the english word OR to describe XOR (exclusive or)

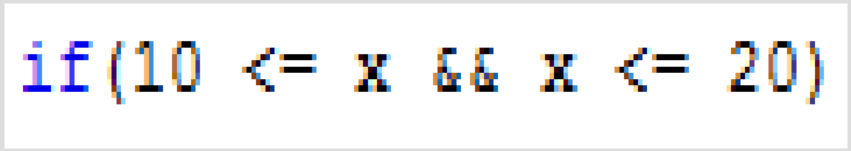
“We can have our final exam on the scheduled day (May 13) or the last day of class (May 6).”

Did you think the statement above meant final exams on both days was a possibility?

Complex expressions

If statements for when x...

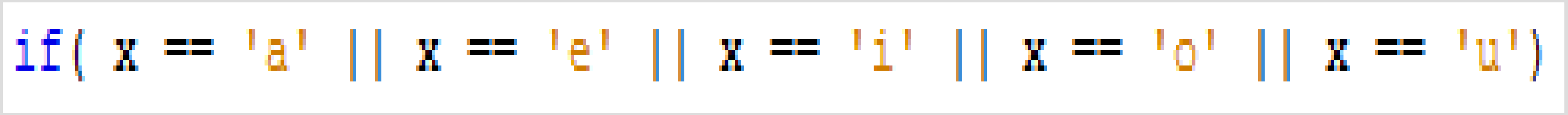
... is between 10 and 20 (inclusive)



```
if(10 <= x && x <= 20)
```

Cannot say: 10 <= x <= 20 (why?)

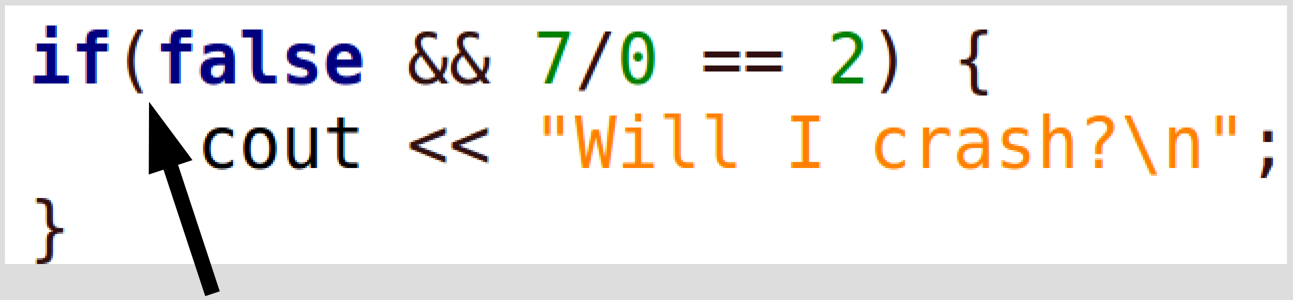
... is a vowel (x is type char)



```
if( x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u')
```

Short-circuit evaluation

Short-circuit evaluation is when you have a complex bool expression (&& or ||) but you don't need to compute all parts.



```
if(false && 7/0 == 2) {
    cout << "Will I crash?\n";
}
```

If this is false, then it will not check next

(See: shortCircuit.cpp)

Short-circuit evaluation

Simple cases of short-circuit:

When you have a bunch of ORs

```
if( expression || exp || exp || exp )
```

Once it finds any true expression,
if statement will be true

When you have a bunch of ANDs

```
if( expression && exp && exp && exp )
```

Once it finds any false expression,
if statement will be false

Complex expressions

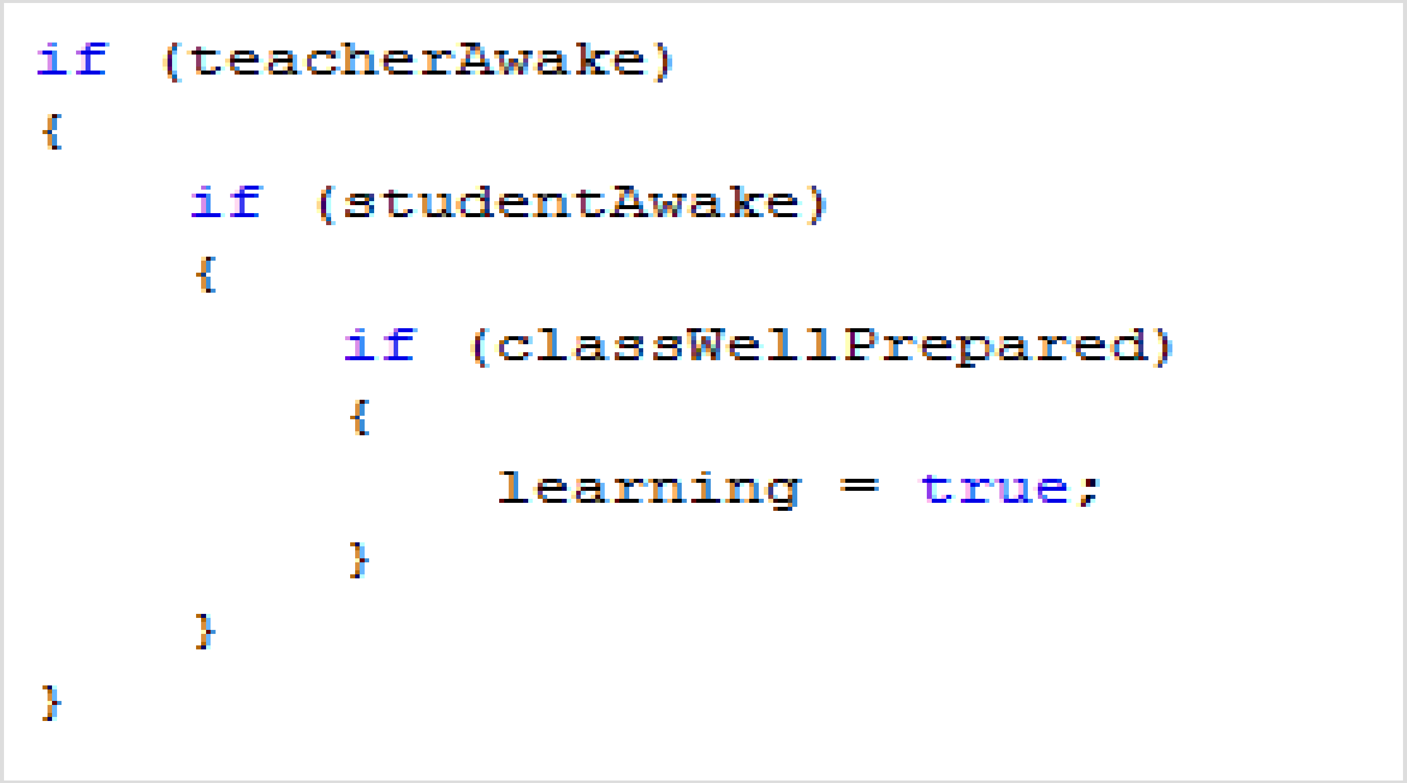
Be careful when negating, that you follow De Morgan's Law:

```
bool a, b;
!(a OR b) is equivalent to (!a) AND (!b)
!(a AND b) is equivalent to (!a) OR (!b)
```

“Neither rainy or sunny” means
“Both not rain and not sunny”

Nested if statements

You can have as many if statements inside each other as you want.



```
if (teacherAwake)
{
    if (studentAwake)
    {
        if (classWellPrepared)
        {
            learning = true;
        }
    }
}
```

Nested if statements

From a truth table perspective, nested loops are similar to AND

The previous if code is equivalent to:

```
if(teacherAwake && studentAwake && classWellPrepared)
{
    learning = true;
}
```

However, sometimes you want to do other code between these evaluations

Nested if statements

RUT-
ROH
WHAT IS AN INTERNET
LAWYER?
MONTY PYTHON and the HOLY GRAIL
Keeper of the Bridge of Death

(See: bridgeOfDeath.cpp)

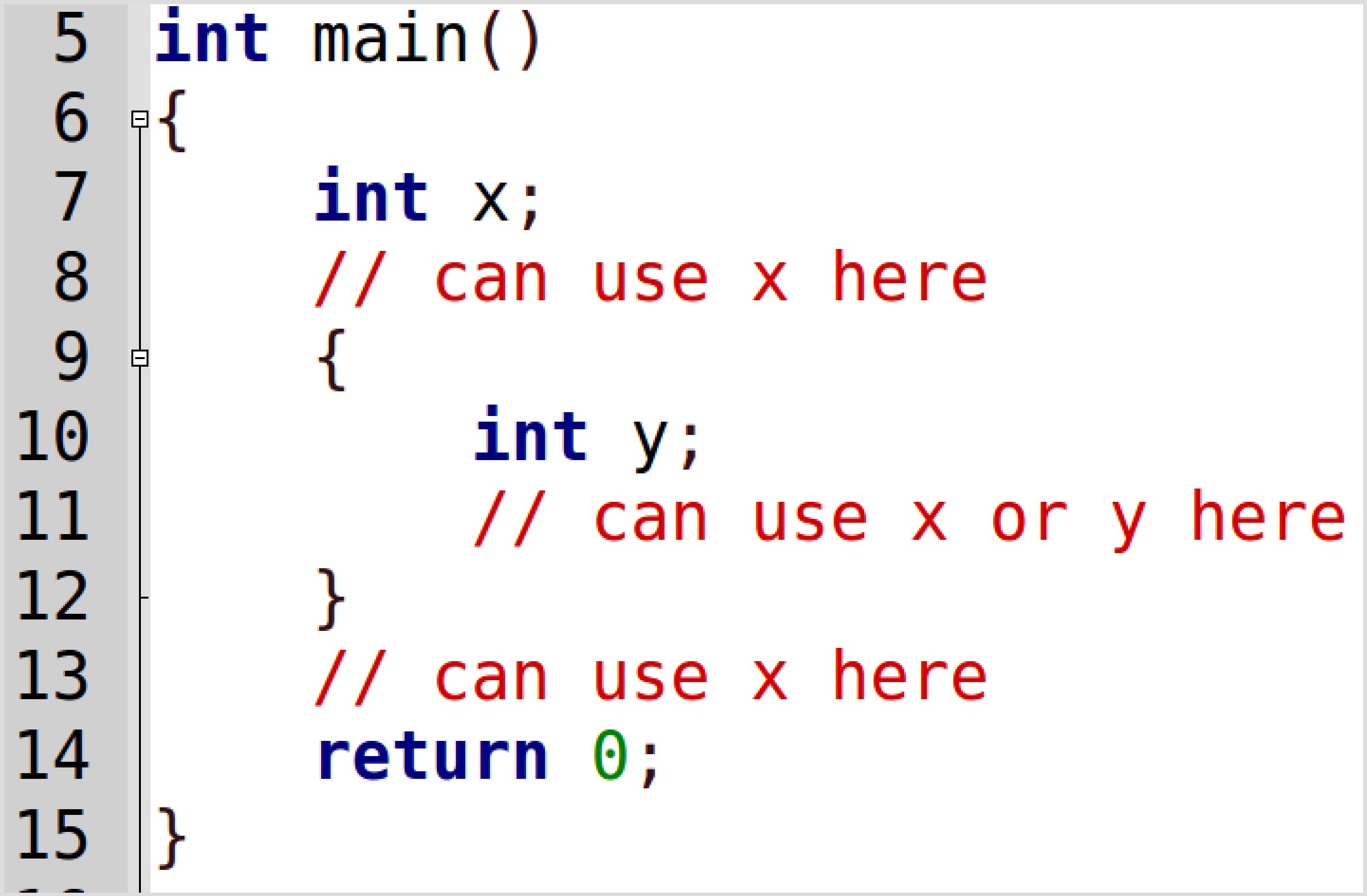
Scope

Where a variable is visible is called its <u>scope</u>

Typically variables only live inside the block (denoted with matching { and })

A variable lives until the block is closed, so inner blocks can see everything from the block it was created inside

Scope



```
int main()
{
    int x;
    // can use x here
    {
        int y;
        // can use x or y here
    }
    // can use x here
    return 0;
}
```

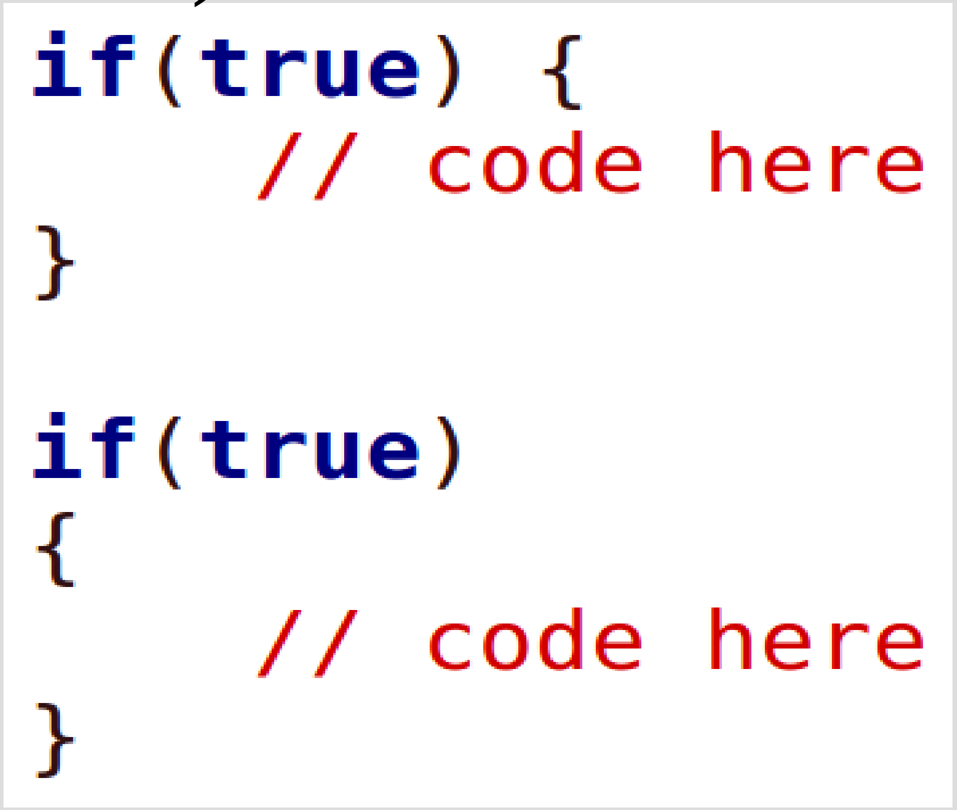
(See: scope.cpp)

If... if... else!

When in doubt, use parenthesis and blocks!
(Some people like to put the first brace after the if, others on a new line)

What happens if you have an if if else?

(See: ifIfElse.cpp)



```
if(true) {
        // code here
}

if(true)
{
        // code here
}
```

Multiway if/else

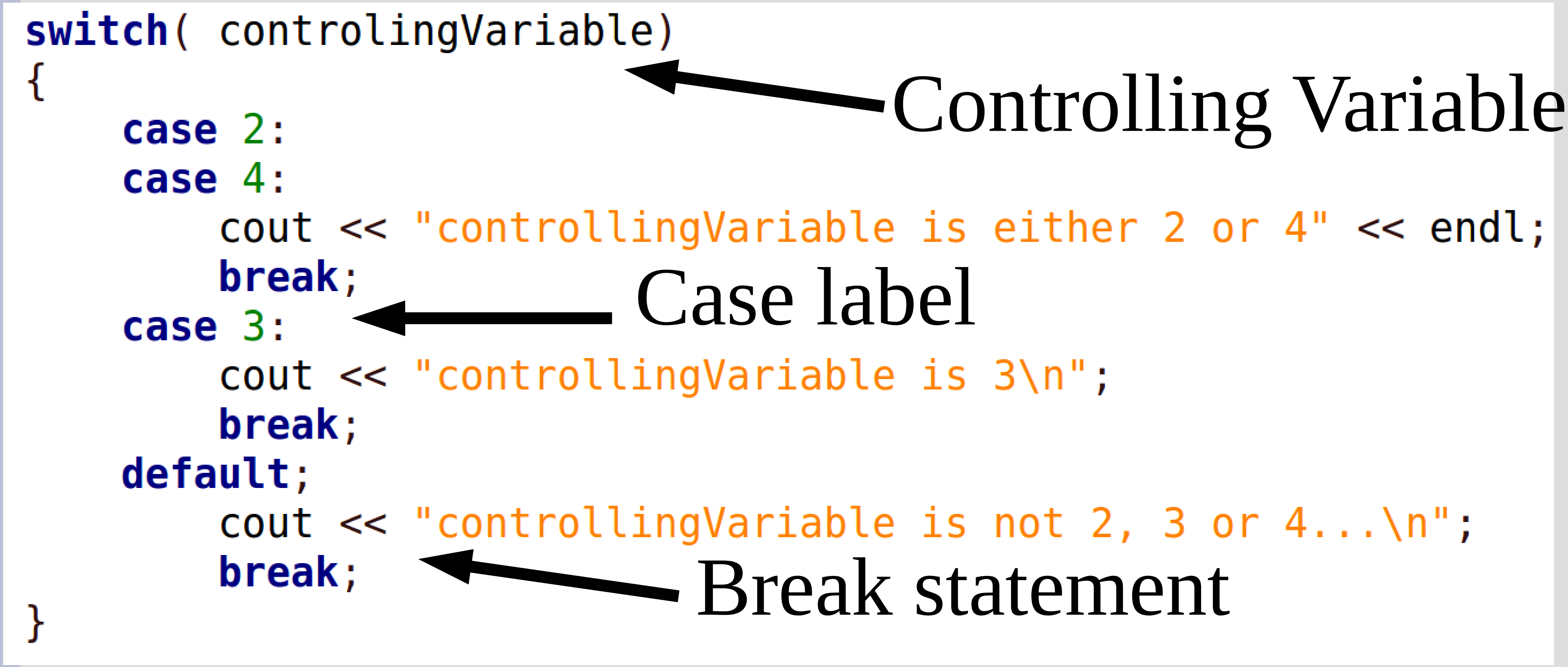
This is a special format if you put an if statement after an else.

This second “if statement” only is tested when the first “if statement” is not true

(See: grades.cpp)

Switch

A switch statement checks to see if a variable has a specific value.



```
switch( controlingVariable)
{
    case 2:
    case 4:
        cout << "controllingVariable is either 2 or 4" << endl;
        break;
    case 3:
        cout << "controllingVariable is 3\n";
        break;
    default;
        cout << "controllingVariable is not 2, 3 or 4...\n";
        break;
}
```

Switch

If the value of the controlling variable
is found in a case label, all code until
a break statement is ran (or the switch ends)

Switch statements only test equality
with case labels (not greater or less than)

(See: switch.cpp)

Switch

Switch statements can be written as multiway if/else statements.

Could use just “if statements” but “else if” shows only one of these will run

(See: switchToIf.cpp)

Conditional operator

We will not use in this class, but if you use other people's code you will encounter

Shorthand for an if-else statement

(boolean) ? [if true] : [if false]

Example:
max = (x>y) ? x : y;
(See: max.cpp)