

SELECTION STRUCTURES

Flow of Control

- Statements execute in sequential (**linear**) order
 - Top-to-bottom**

```
int main()
```

```
{
```

```
    int x;
```

```
    x = 46;
```

```
    x++;
```

```
    cout << x << endl;
```

```
    return 0;
```

```
}
```

1

2

3

4

5

int x;

x = 46;

x++;

cout

return

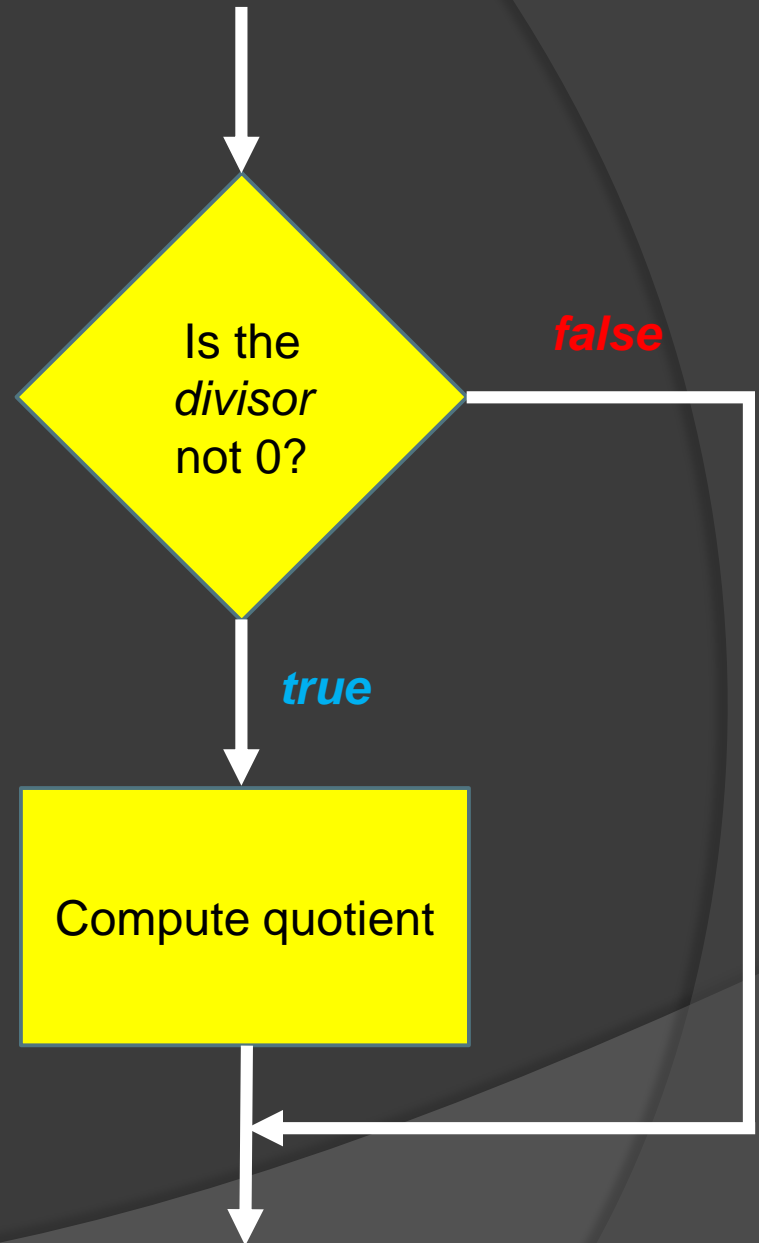
Flow of Control

- ④ What if we want to execute code only **some times**?
- ④ Write a C++ program that asks for the *dividend* and *divisor* and then computes the *quotient*
 - $quotient = \frac{dividend}{divisor}$
 - **But only if the divisor is not zero!**

Flow of Control

● Flow chart:

- Ask *yes/no* decision
- Execute code if *yes*
- Don't execute if *no*



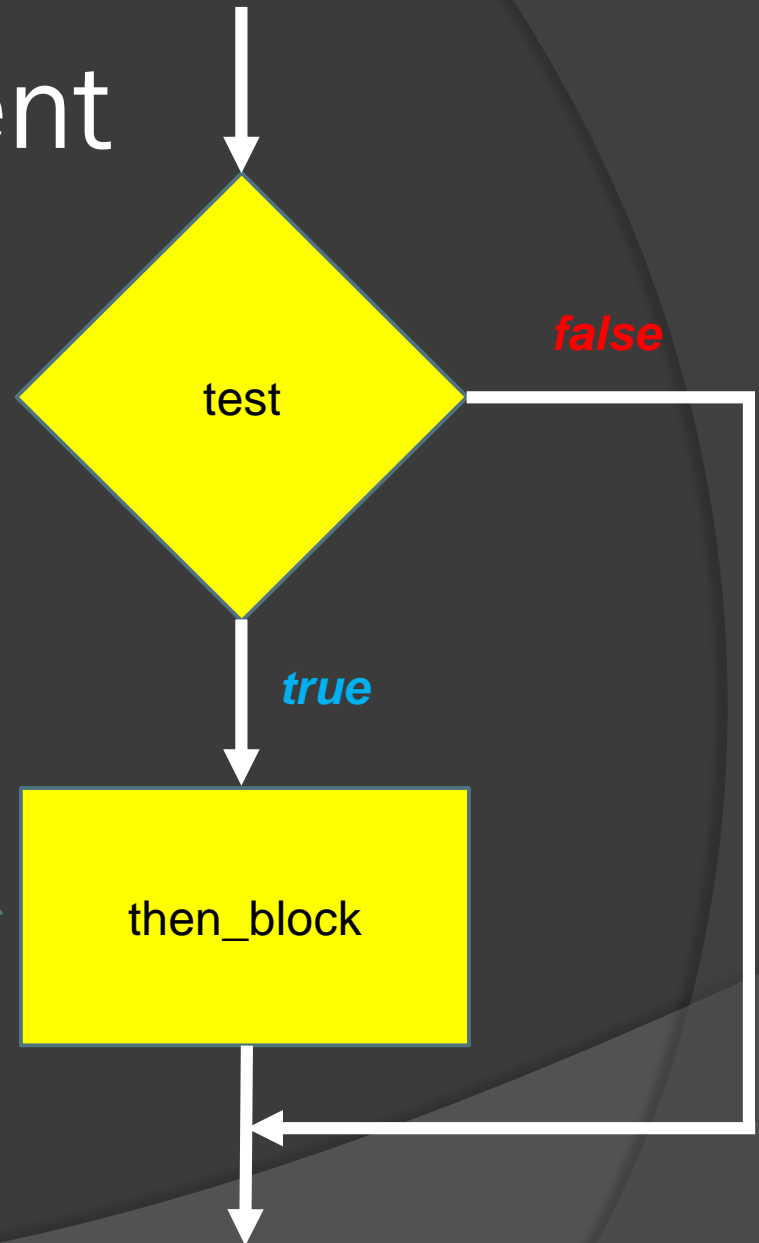
if-then statement

```
if (test)
```

```
{
```

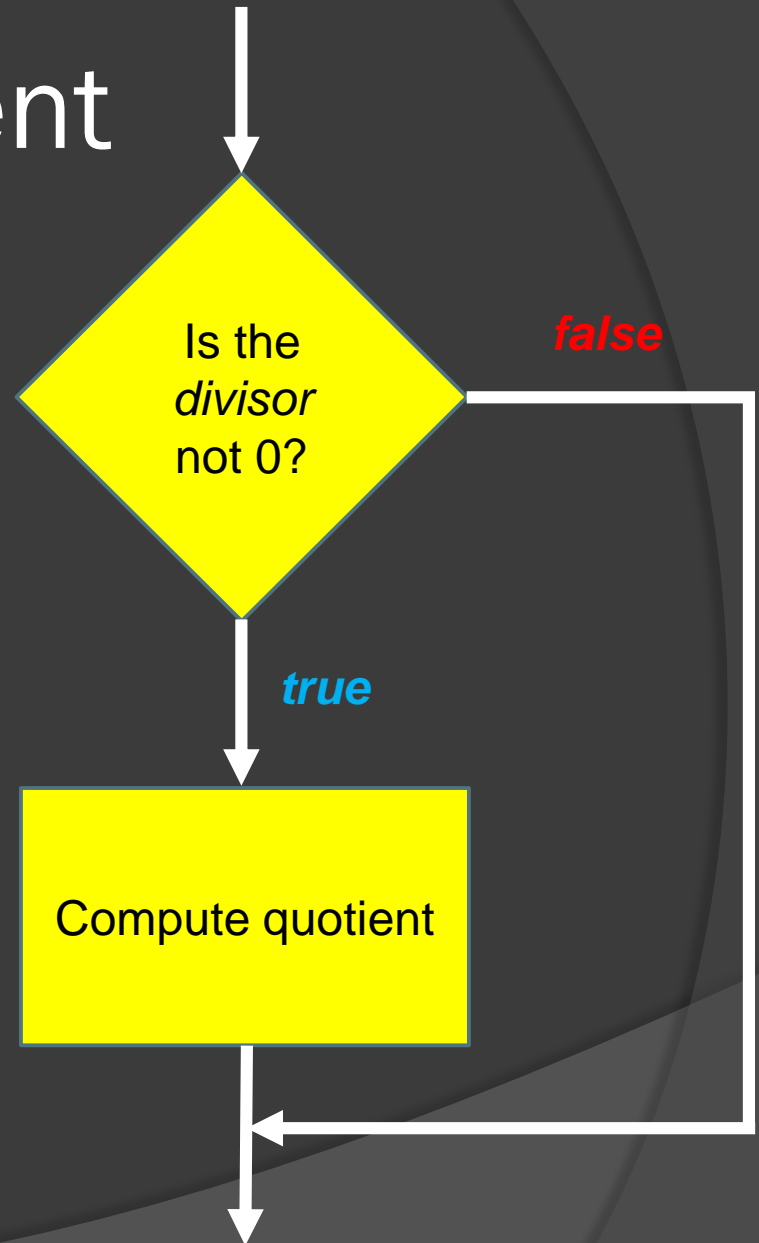
```
  then_block
```

```
}
```



if-then statement

```
if (divisor != 0)
{
    quotient = dividend/divisor;
}
```



if-then statement

```
if (conditional expression)
{
    Statements executed
    only if condition is true
}
```

- ⦿ The conditional expression is a **Boolean expression** (formula)
 - Evaluates to **true** or **false**

Boolean Expression

- Remember: An expression contains constants, variables, operators (relational & logical), and function calls, i.e. a formula
- The simplest Boolean expressions are **true** and **false**
- For example

```
if (true)  
{  
    cout << "This is always executed" << endl;  
}
```

- The **then_block** is ALWAYS executed

Relational Operators

Operator	Meaning	Example
<	less than	<code>x < 0</code>
>	greater than	<code>speed > 65</code>
<=	less than or equal to	<code>age <= 17</code>
>=	greater than or equal to	<code>gpa >= 3.5</code>
==	equal to	<code>initial == 'a'</code>
!=	not equal to	<code>divisor != 0</code>

- Relational operators return **false** (0) and **true** (1)

Relational Operators are *Binary Operators*

- Make sure operands have *similar* types

ASCII Table

- Compare numbers

$2 < 0$ evaluates to *false*

- Compare characters

$\text{'a'} < \text{'b'}$ evaluates to *true* (why???)

$\text{'G'} > \text{'M'}$ evaluates to *false*

$\text{'E'} \neq \text{'e'}$ evaluates to *true*



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																

Boolean Expression

- ⦿ A **boolean expression** asks a yes/no question, i.e. true/false
- ⦿ Is the value in variable `dogs` **positive**?
`dogs > 0`
- ⦿ Is the value in variable `age` **at least 18**?
`age >= 18`
- ⦿ Is the value in variable `cats` **at most 10**?
`cats <= 10`

Your Turn – Write the Conditional Expression for:

- Is the value in variable i **even**?

```
if ( i % 2 == 0 ) {
```

- Is the value in variable j **odd**?

```
if ( j % 2 == 1 ) {
```

- Is the sum of i and j **positive**?

```
if ( i + j > 0 ) {
```

booleanExpr.cpp

```
// Examples of Boolean expressions
#include <iostream>
using namespace std;

int main()
{
    double x(10);

    cout << "(4 != 4) evaluates to " << (4 != 4) << endl;
    cout << "('a' <= 'b') evaluates to " << ('a' <= 'b') << endl;
    cout << "('B' < 'b') evaluates to " << ('B' < 'b') << endl;
    cout << "((2.0 + 5.0) == 7) evaluates to " << ((2.0 + 5.0) ==
        7) << endl;
    cout << "((5/3) > 1.0) evaluates to " << ((5/3) > 1.0) << endl;

    cout << "(x == 10) evaluates to " << (x == 10) << endl;
    cout << "(x = 8) evaluates to " << (x = 8) << endl;

    return 0;
}
```

Note the single =

booleanExpr.cpp executed

`g++ booleanExpr.cpp`

`a.out`

`(4 != 4) evaluates to 0`

`('a' <= 'b') evaluates to 1`

`('B' < 'b') evaluates to 1`

`((2.0 + 5.0) == 7) evaluates to 1`

`((5/3) > 1.0) evaluates to 0`

`(x == 10) evaluates to 1`

`(x = 8) evaluates to 8`

Boolean vs Arithmetic Expressions

- ⦿ A **Boolean expression** evaluates to `true` or `false`
`age >= 16`
- ⦿ An **arithmetic expression** evaluates to a **numeric value**
`age + 1`
- ⦿ A Boolean expression may contain arithmetic expressions
 - What are the arithmetic expressions in this Boolean expression?
`age - 1 >= 15 + 1`
- ⦿ `age - 1`
- ⦿ `15 + 1`
- ⦿ `+` and `-` are arithmetic operators
- ⦿ `>=` is a Boolean operator

Logical Operators

- ◉ Write a Boolean expression that asks:
 - ❖ Is the temperature in variable `temp` between 10 and 32 degrees inclusive?

- ◉ How about?

`10 <= temp <= 32`

- Problem: `<=` is a binary operator and variable `temp` cannot be used as an operand in both operators
- ◉ We want to ask two questions!
 - Is `temp` at least 10 and is `temp` at most 32?
 - This is a **compound** statement

Logical Operators

- Logical operators

- And (**&&**)
- Or (**||**)

allow us to create **compound** statements

- Is variable** `temp` **between 10 and 32 inclusive?**

- **Is** `temp` **at least 10?**
- **Is** `temp` **at most 32?**

```
temp >= 10 && temp <= 32
```

Compound Logical Operators: `&&` and `||`

- Binary operators
- Operands are Boolean expressions
- The AND (`&&`) operator is true when **BOTH** operands evaluate to **true**
- The OR (`||`) operator is true if **EITHER** operand evaluates to **true**
- AND (`&&`) higher precedence than OR (`||`)

Your turn

- ◉ Write a Boolean expression that asks:
 - ❖ Is a person a *teenager* or a *senior*?
 - Variable *age* contains the person's age
- ◉ A teenager's age between 13 and 19
- ◉ A senior has age more than 80
- ◉ The question is yes, i.e. `true`, if **EITHER** of these two statements is true

Logical Operators

- ⦿ The NOT (!) operator asks if a question is false, i.e. not true
 - It negates a Boolean expression evaluation
 - **Unary** operator
 - The single operand is placed to the right of the !
- ⦿ Is a person's age at least 18?
`age >= 18`
- ⦿ Is a person's age NOT at least 18?
`!(age >= 18)`
 - Though, you probably want to use `age < 18`

Your Turn

- ⦿ Write at least two different Boolean expressions that ask:
 - ❖ Is the temperature NOT between 10 and 32? Use variable `temp`.

Operator Precedence

Operator	Associativity
! unary - ++ --	
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
&&	left to right
	left to right
= += -= *= /=	right to left

Note on truth values

- ⦿ Any integer that is not 0 is considered to be a Boolean **true** in C++
- ⦿ Relational and logical operators only evaluate to **false** (0) and **true** (1)

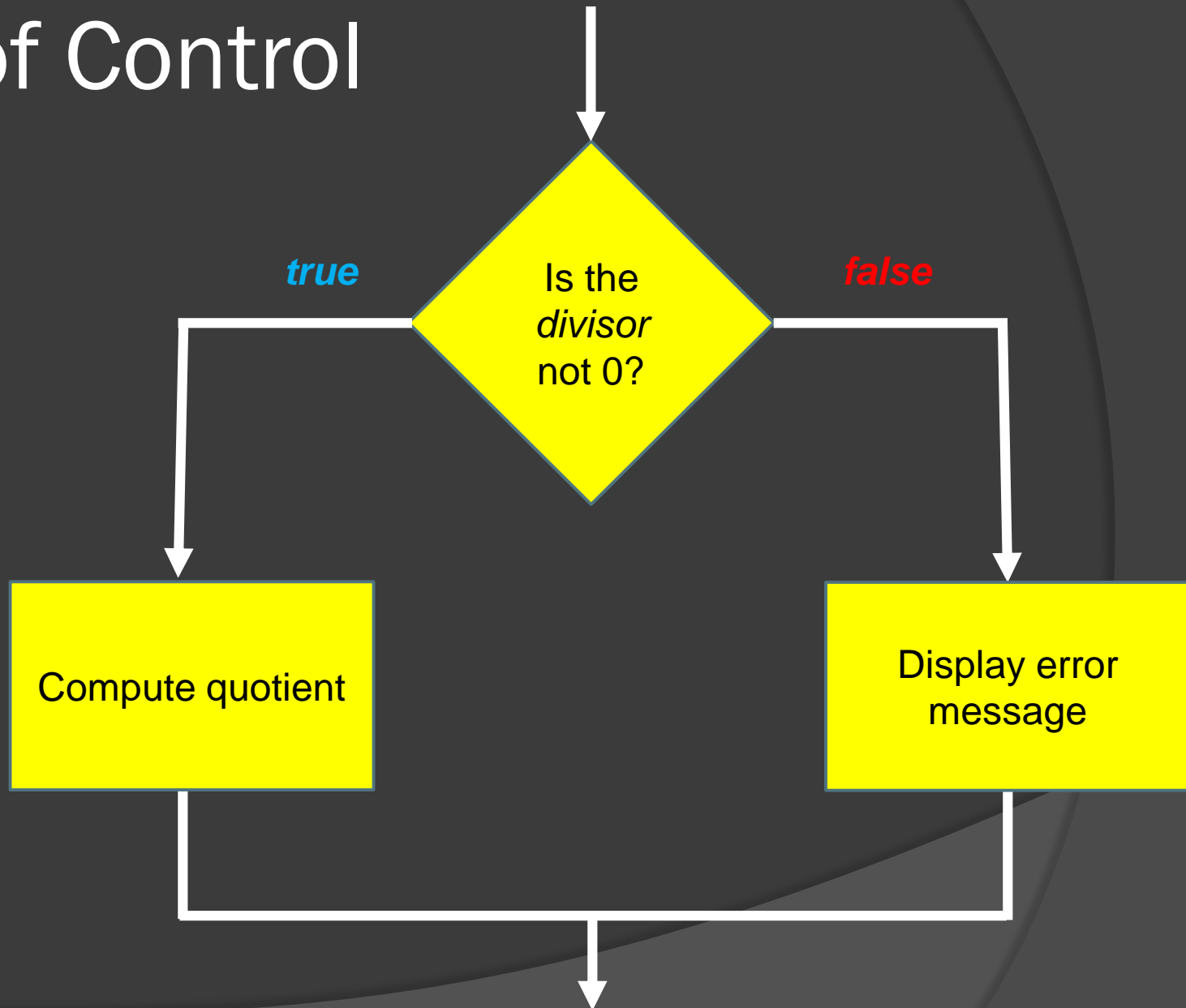
Your Turn

- What do each of the following Boolean expressions evaluate to?

```
int x(5), y(12);
```

- $x > 0 \ \&\& \ x < 10$ *True*
- $x \leq 0 \ || \ x \geq 10$ *False*
- $x - 1 == y / 5 + y \% 5$ *True*
- $x != y \ || \ !(x == y)$ *True*

Flow of Control



if-else statement

```
if (test)
```

```
{
```

```
    then_block
```

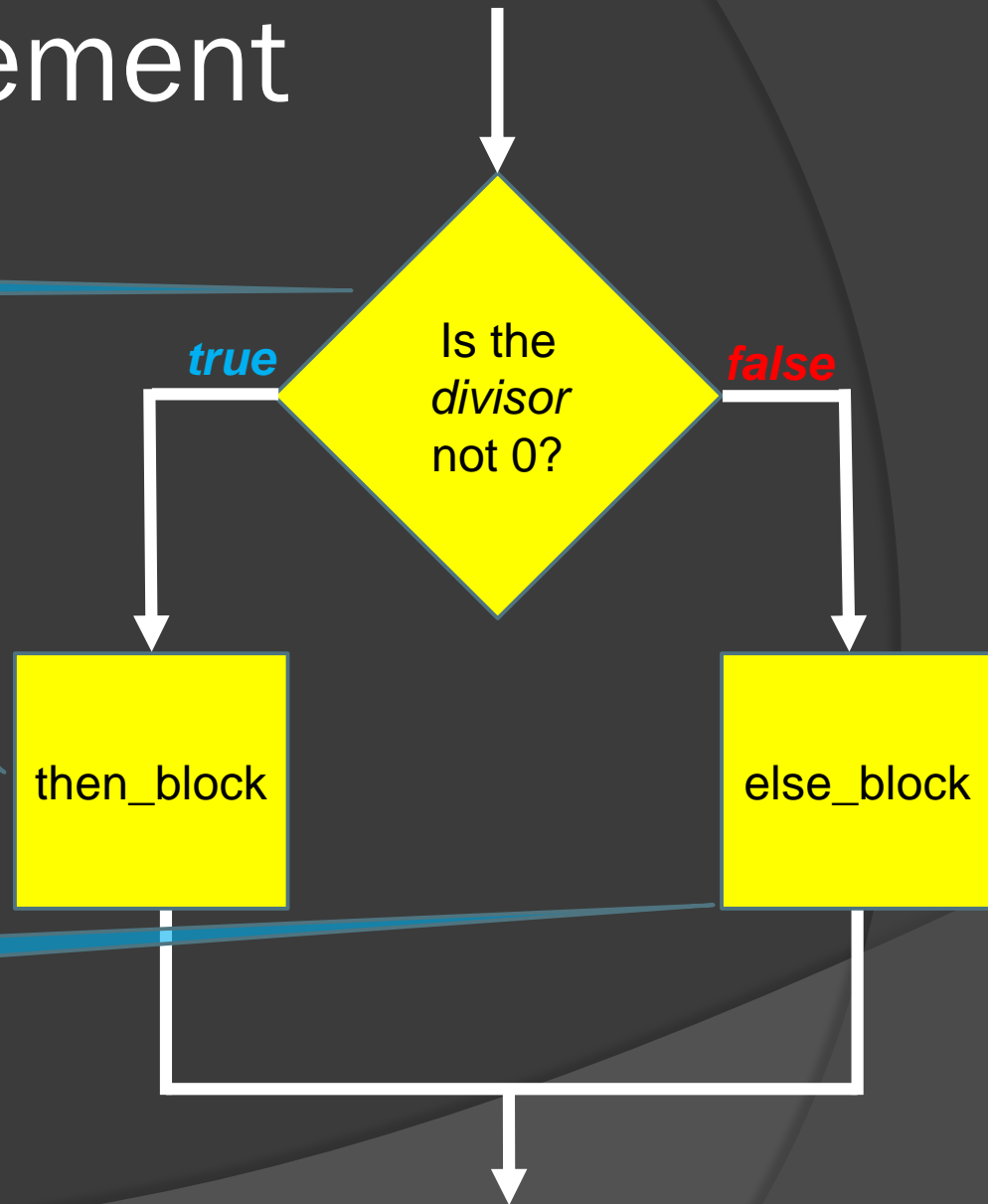
```
}
```

```
else
```

```
{
```

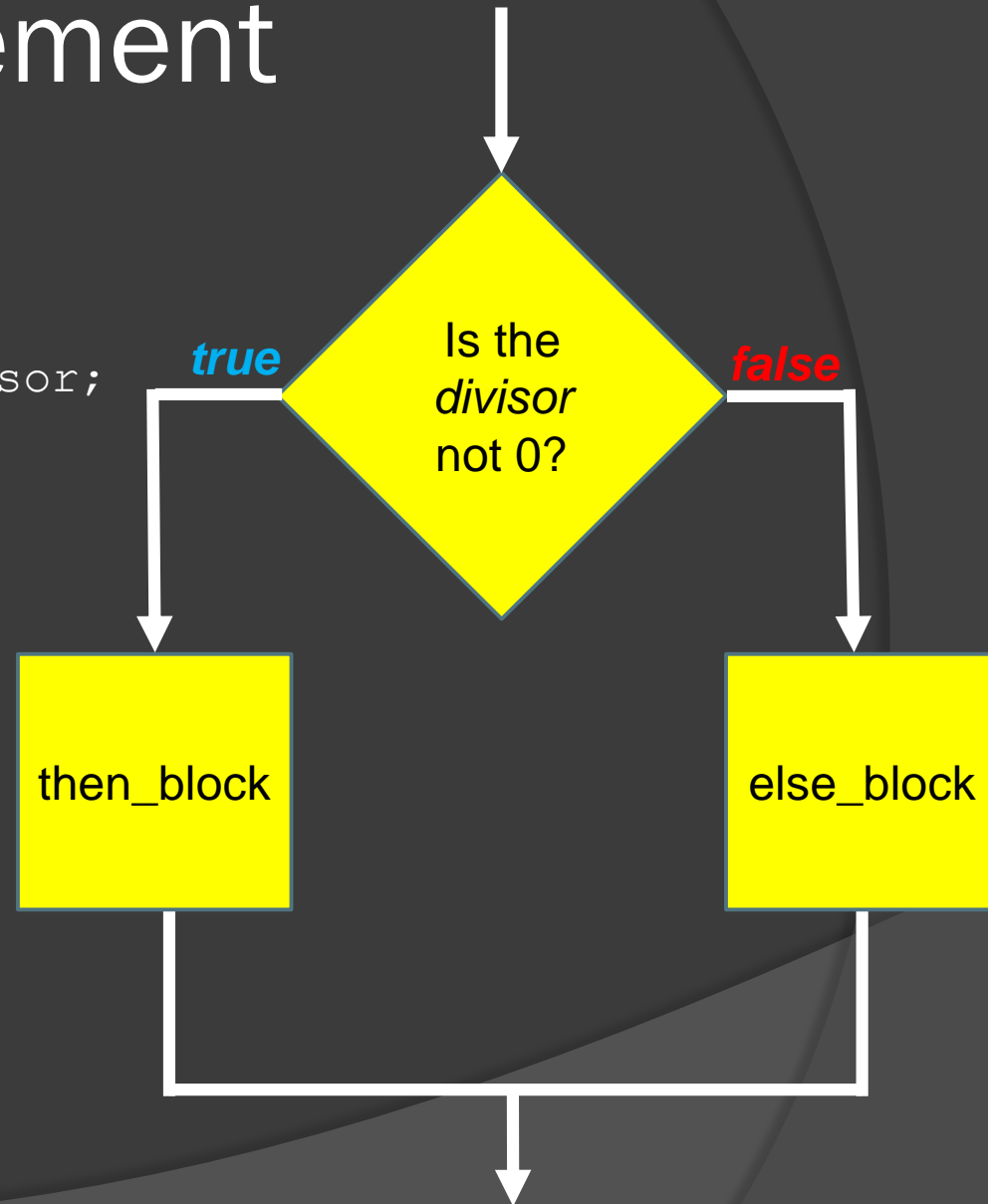
```
    else_block
```

```
}
```



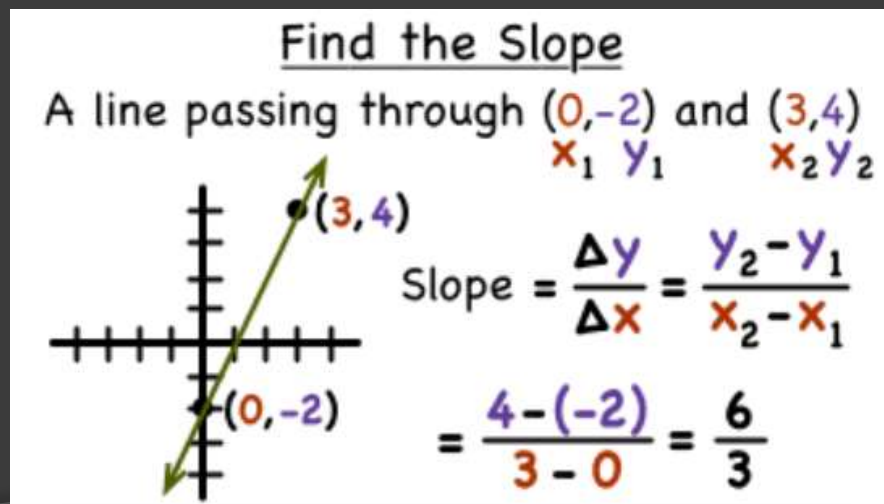
if-then statement

```
if (divisor != 0)
{
    quotient = dividend/divisor;
}
else
{
    cout << "Divide by 0";
}
```



Your Turn: Line Slope Revisited

- Write a C++ program to compute the slope of a line in Cartesian coordinates, i.e. the x-y plane given two points
- If Δx is 0 the slope is *infinite*
- If (x_1, y_1) is the same point as (x_2, y_2) then the slope is *undefined*



slope.cpp

```
#include <iostream>
using namespace std;

int main()
{
    double x1, y1, x2, y2, xdiff, ydiff, slope;

    cout << "Enter x and y coordinates of first point : ";
    cin >> x1 >> y1;
    cout << "Enter x and y coordinates of second point : ";
    cin >> x2 >> y2;
    ...
}
```

slope.cpp (cont.)

```
...  
    xdiff = x1 - x2;  
    ydiff = y1 - y2;  
  
    if (xdiff != 0.0)  
    {  
        slope = ydiff / xdiff;  
        cout << "The slope is: " << slope << endl;  
    }  
    else  
    {  
        cout << "The slope is infinite." << endl;  
    }  
  
    return 0;  
}
```

Δx is 0

```
...  
    xdiff = x1 - x2;  
    ydiff = y1 - y2;  
  
    if (xdiff != 0.0)  
    {  
        slope = ydiff / xdiff;  
        cout << "The slope is: " << slope << endl;  
    }  
    else  
    {  
        cout << "The slope is infinite." << endl;  
    }  
}
```

> slope.exe

Enter x and y coordinates of first point : 0 3

Enter x and y coordinates of second point : 4 1

The slope is: -0.5

```
...  
    xdiff = x1 - x2;    // Execute program with a new set of points  
    ydiff = y1 - y2;  
  
    if (xdiff != 0.0)  
    {  
        slope = ydiff / xdiff;  
        cout << "The slope is: " << slope << endl;  
    }  
    else  
    {  
        cout << "The slope is infinite." << endl;  
    }  
}
```

> slope.exe

Enter x and y coordinates of first point : 4 3

Enter x and y coordinates of second point : 4 1

The slope is infinite.

Δx is 0


```
...  
    xdiff = x1 - x2;    // Execute program with a new set of points  
    ydiff = y1 - y2;  
  
    if (xdiff != 0.0)  
    {  
        slope = ydiff / xdiff;  
        cout << "The slope is: " << slope << endl;  
    }  
    else  
    {  
        cout << "The slope is infinite." << endl;  
    }  
}
```

> slope.exe

Enter x and y coordinates of first point : 4 3

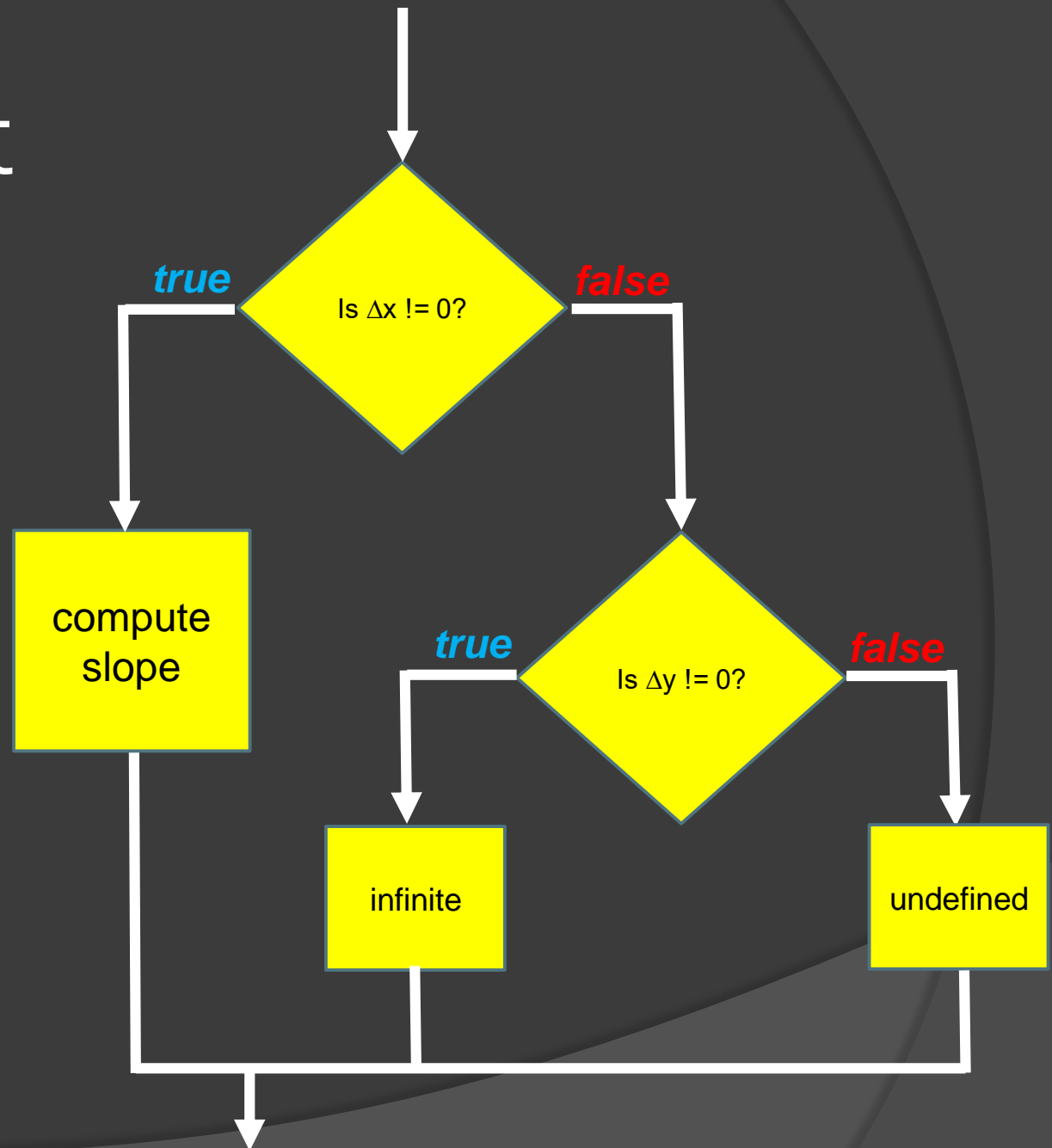
Same points!

Enter x and y coordinates of second point : 4 3

The slope is infinite.

WRONG! The slope is *undefined*

Flow Chart



slope2.cpp: Nested ifs

```
...
    xdiff = x1 - x2;
    ydiff = y1 - y2;

    if (xdiff != 0.0)
    {
        slope = ydiff / xdiff;
        cout << "The slope is: " << slope << endl;
    }
    else // we now know xdiff is zero!
    {
        if (ydiff != 0.0) {
            cout << "The slope is infinite." << endl;
        }
        else {
            cout << "Input Error: First point equals second point."
                << endl;
            cout << "Slope undefined." << endl;
        }
    }
...

```

```

...
if (xdiff != 0.0)
{
    slope = ydiff / xdiff;
    cout << "The slope is: " << slope << endl;
}
else
{
    if (ydiff != 0.0) {
        cout << "The slope is infinite." << endl;
    }
    else {
        cout << "Input Error: First point equals second point."
            << endl;
        cout << "Slope undefined." << endl;
    }
}
...

```

> slope.exe

Enter x and y coordinates of first point : 4 3

Enter x and y coordinates of second point : 4 3

Input Error: First point equals second point.

Slope undefined.

Nested Conditional Statements

```
...  
int main()  
{  
    int age(0);  
  
    cout << "Enter your age: ";  
    cin >> age;  
  
    if (age >= 15)  
    {  
        if (age == 15)  
        { cout << "You can get a learners permit." << endl; }  
        else  
        { cout << "You can get a license." << endl; }  
    }  
    else  
    { cout << "You are too young to drive." << endl; }  
  
    return 0;  
}
```

**What is the output on
input:**
10
15
20

Nested Conditional Statements

```
...  
int main()  
{  
    int age(0);  
  
    cout << "Enter your age: ";  
    cin >> age;  
  
    if (age >= 18)  
    {  
        cout << "You can vote." << endl;  
        if (age >= 35)  
        { cout << "You can become President." << endl; }  
    }  
    else  
    { cout << "You are too young to vote." << endl; }  
  
    return 0;  
}
```

What is the output on input:

10

20

40

logicalOperators1.cpp

```
// Examples of logical operators
#include <iostream>
using namespace std;

int main()
{
    int x(5);
    int y(25);

    if (x < 7 && y < 12) { cout << "true" << endl; }
    else { cout << "false" << endl; };

    if (x < 7 || y < 12) { cout << "true" << endl; }
    else { cout << "false" << endl; };

    if (x < 7 && !(y < 12)) { cout << "true" << endl; }
    else { cout << "false" << endl; };

    return 0;
}
```

What is output?



logicalOperators2.cpp

```
// Examples of logical operators
#include <iostream>
using namespace std;

int main()
{
    int x(5);
    int y(25);

    if ((x < 7 && y < 12) || (x > 7 && y > 12))
    { cout << "true" << endl; }
    else { cout << "false" << endl; };

    if ((x < 7 || y < 12) && (x > 7 || y > 12))
    { cout << "true" << endl; }
    else { cout << "false" << endl; };

    return 0;
}
```

What is output?



Boolean Variables

- ⦿ The Boolean data type is called *bool*

```
bool red(true);  
if (red)  
{  
    cout << "It is red" << endl;  
}  
else  
{  
    cout << "It is not red" << endl;  
}
```

Boolean Variables

```
bool red(true);  
if (red == true) // Avoid this!  
{  
    cout << "It is red" << endl;  
}  
else  
{  
    cout << "It is not red" << endl;  
}
```

Rewrite

- ⦿ Ask “Is it not red?”

```
bool red(true);  
if (!red)  
{  
    cout << "It is not red" << endl;  
}  
else  
{  
    cout << "It is red" << endl;  
}
```

Rewrite

```
bool red(true);  
if (red == false) // Avoid this!  
{  
    cout << "It is not red" << endl;  
}  
else  
{  
    cout << "It is red" << endl;  
}
```

Example of Boolean variables

```
...
int main()
{
    int age(0);
    bool flag_discount(false);

    cout << "Enter your age: ";
    cin >> age;

    if (age < 18)
    { flag_discount = true; }

    if (age >= 65)
    { flag_discount = true; }

    if (flag_discount)
    { cout << "You receive a discount." << endl; }
    else
    { cout << "No discount." << endl; }
    ...
}
```

comparisonExample.cpp

```
// comparison example
#include <iostream>
using namespace std;

int main()
{
    int year(0);

    cout << "Enter year: ";
    cin >> year;

    int k = year%4;
    if (k == 0)
    {
        cout << year << " is a US presidential election year." << endl;
    }
    else
    {
        cout << year << " is NOT a US presidential election year." << endl;
    }

    return 0;
}
```

```
...  
    int k = year%4;  
    if (k == 0)          // if k is equal to 0, ...  
    {  
        cout << year << " is a US presidential election year." << endl;  
    }  
    else  
    {  
        cout << year << " is NOT a US presidential election year." << endl;  
    }  
...  

```

> comparisonExample

Enter year: 2023

2021 is NOT a US presidential election year.

> comparisonExample

Enter year: 2024

2020 is a US presidential election year.

comparisonError.cpp

```
// comparison error
#include <iostream>
using namespace std;

int main()
{
    int year(0);

    cout << "Enter year: ";
    cin >> year;

    int k = year%4;
    if (k = 0) ← Note the single =
    {
        cout << year << " is a US presidential election year." << endl;
    }
    else
    {
        cout << year << " is NOT a US presidential election year." << endl;
    }

    return 0;
}
```



```
...
int k = year%4;
if (k = 0)
{
    cout << year << " is a US presidential election year." << endl;
}
else
{
    cout << year << " is NOT a US presidential election year." << endl;
}
...
```

```
> comparisonError
Enter year: 2023
2021 is NOT a US presidential election year.
```

```
> comparisonError
Enter year: 2024
2020 is NOT a US presidential election year.
```

WRONG!!!



Common Mistakes with Conditions

```
if (age = 18)
```

- The equality operator `==` is different from the assignment operator `=`
- We are actually assigning `age` to the value 18!!!
- The expression `age = 18` evaluates to `18`
- So the condition (boolean expression) **becomes** this when executed:

```
if (18)
{
    cout << "You are 18."
}
```
- 0 evaluates to **false** and any other number evaluates to **true**. Thus, the boolean expression in the condition, i.e. 18, evaluates to **true**

comparisonError2.cpp

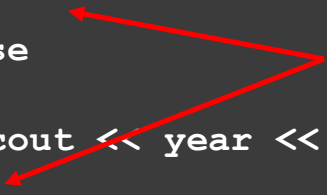
```
// comparison example
#include <iostream>
using namespace std;

int main()
{
    int year(0);

    cout << "Enter year: ";
    cin >> year;

    int k = year%4;
    if (k == 0)          // if k is equal to 0, ...
    {
        cout << year << " is a US presidential election year." << endl;
    };
    else
    {
        cout << year << " is NOT a US presidential election year." << endl;
    };

    return 0;
}
```



Note the ;

comparisonError2.cpp

```
...
int k = year%4;
if (k == 0)           // if k is equal to 0, ...
{
    cout << year << " is a US presidential election year." <<
    endl;
};
else
{
    cout << year << " is NOT a US presidential election
    year." << endl;
};
...
```

- **Syntax Error:**

main.cpp:17:3: error: 'else' without a previous 'if'
17 | else | ^~~~

comparisonError2.cpp

```
// comparison example
#include <iostream>
using namespace std;

int main()
{
    int year(0);

    cout << "Enter year: ";
    cin >> year;

    int k = year%4;
    if (k == 0);
    {
        cout << year << " is a US presidential election year." << endl;
    }
    else
    {
        cout << year << " is NOT a US presidential election year." << endl;
    };

    return 0;
}
```

Find the error.

Hint: there are actually 2 errors.

comparisonError2.cpp

Warning! Common Mistakes

- ⦿ Use '==' (equal to), not '=' (assignment)
- ⦿ Use '&&' not '&'
- ⦿ Use '||', not '|'
- ⦿ Do not put a semi-colon before an “else” clause
- ⦿ Do not put a semi-colon after the condition