AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 2: Control statement I - selection
boolean variable, if and if-else statement,
logical operations, conditional expression,
switch statement, mathematics expressions

What is true/false?

A logical statement is a descriptive sentence that could be either **true** or **false** but not both, based on a known set of facts.

Example of logical statement:
"Stephen Hawking passed away in 2018." (true)
"100 is less than 99." (false)
"N is an odd number"
(dependent on N, could be true or false but not both)

Example of non logical statements:
"Hi, how are you doing?" (not a descriptive sentence)
"This statement is false." (a paradox)

Boolean variable

A Boolean variable is a variable that takes only two possible values, 0 (false) and 1 (true). In C++ we use `bool` .

Equalities and inequalities in mathematics can be regarded as logical statements, which true/false value can be assigned to a Boolean variable. Brackets should be used around an inequality or equality. The six relational operators are as follows.

Name	Mathematical symbol	C++ Expression
equal to	$=$	==
not equal to	$\neq$	!=
greater than	$>$	>
greater than or equal to	$\geq$	>=
less than	$<$	<
less than or equal to	$\leq$	<=

if statement

An `if` statement is a construct that enables a program to specify alternative path of execution. It consists of two parts: condition (a Boolean expression) and consequent (statements). Refer to the syntax below:

```
if (boolean_condition)
   consequent;
```

If the condition returns a true value, then the consequent will be executed. If the condition returns a false value, the consequent will be ignored.

In case the consequent consists of several statements, group them together as a block using the braces {}.

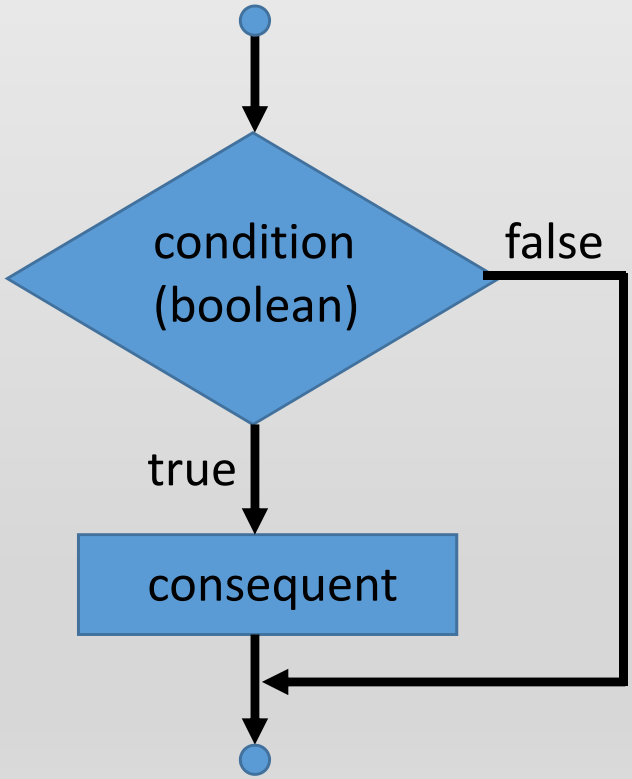
```
if (boolean_condition) {
   consequent1;
   consequent2;
   consequent3;
}
```

For an `if-else` statement, if the condition returns a false value, the alternative will be executed.

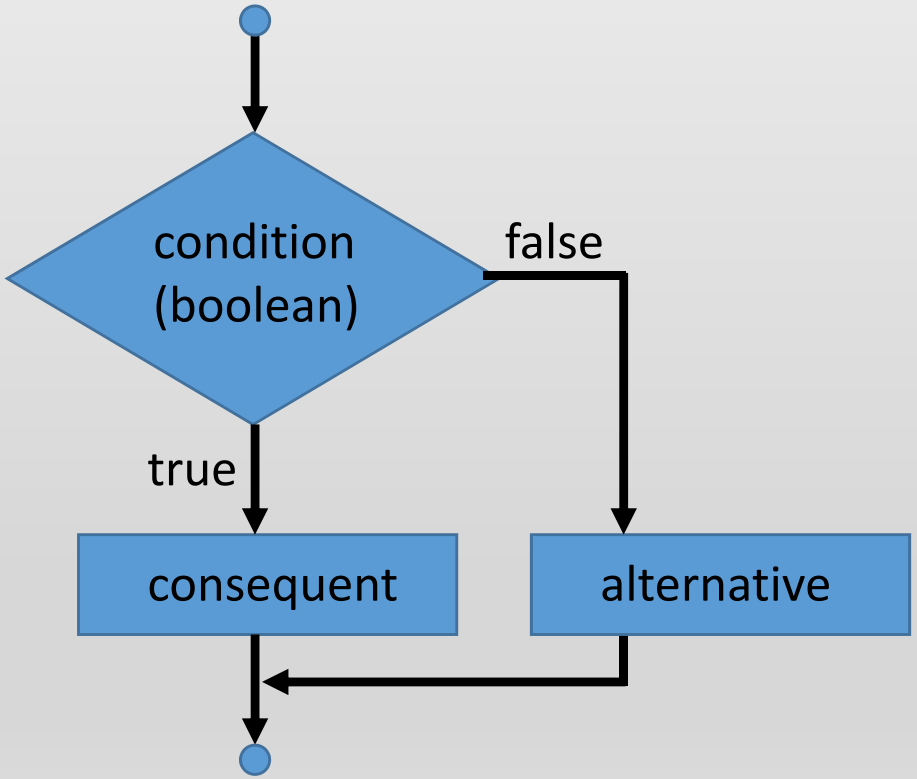
```
if (boolean_condition)
   consequent;
else
   alternative;
```

We can express how a programming language executes the syntax of statements by showing a flowchart. A flowchart is a diagram that describes an algorithm or process.

`if` statement



`if-else` statement



Don't forget to use () for the boolean condition!

Example program 2.1

```
1  // 2.1 if Statement and bool Data Type
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7          int m,n;
8          cout << "Enter the first integer: ";
9          cin >> m;
10         cout << "Enter the second integer: ";
11         cin >> n;
12         if (m%n == 0)
13                 cout << m << " is divisible by " << n << endl;
14         else {
15                 cout << m << " is not divisible by " << n << endl;
16                 cout << "The remainder is " << m%n << endl;
17         }
18         return 0;
19 }
```

prompt user to input two integers

check if m is divisible by n

consequent

alternative

Be careful!

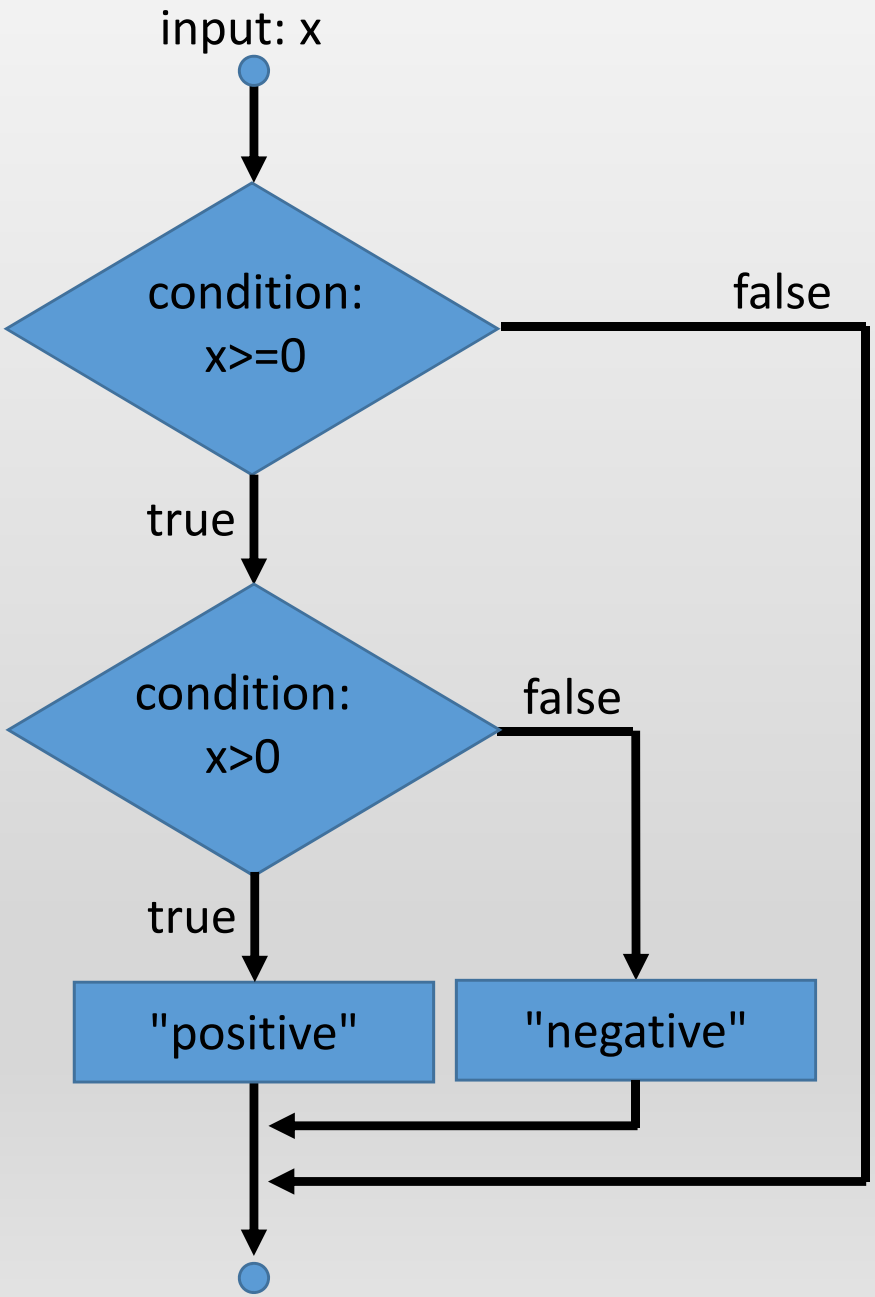
In an `if-else` statement, the `else` corresponds to the closest previous `if` in the same block. Compare the following programs, which one is correct? Test them by using different input values.

```
#include <iostream>
using namespace std;
int main()
{
int x;
cin >> x;
if (x>=0)

if (x>0) cout<<"positive"<<endl;

else cout<<"negative"<<endl;
return 0;
}
```

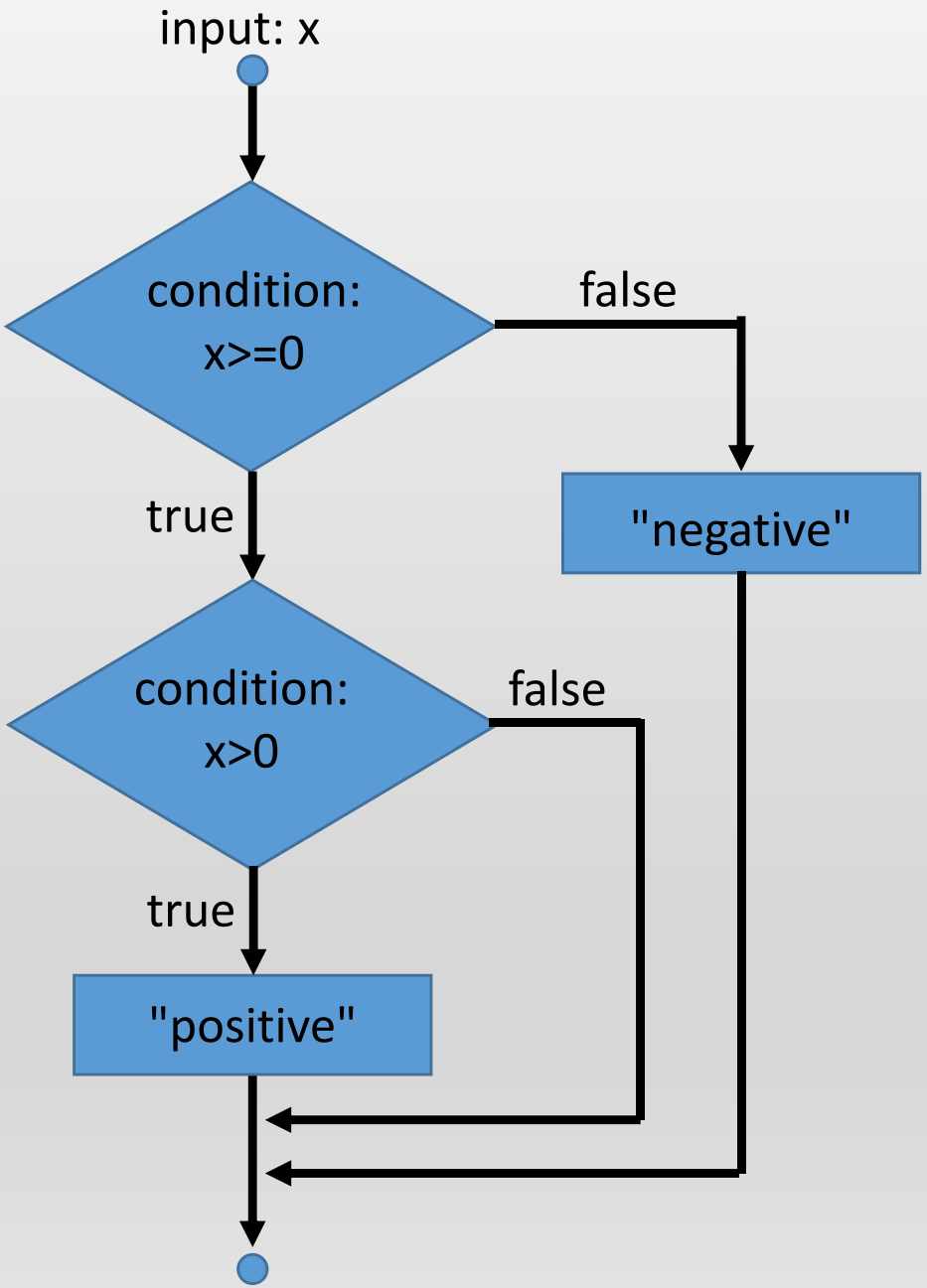
```
#include <iostream>
using namespace std;
int main()
{
int x;
cin >> x;
if (x>=0)
{
if (x>0) cout<<"positive"<<endl;
}
else cout<<"negative"<<endl;
return 0;
}
```

As a good habit, use intent correctly to identify the corresponding `if-else`. Use `{ }` to enclose several statements as a block.

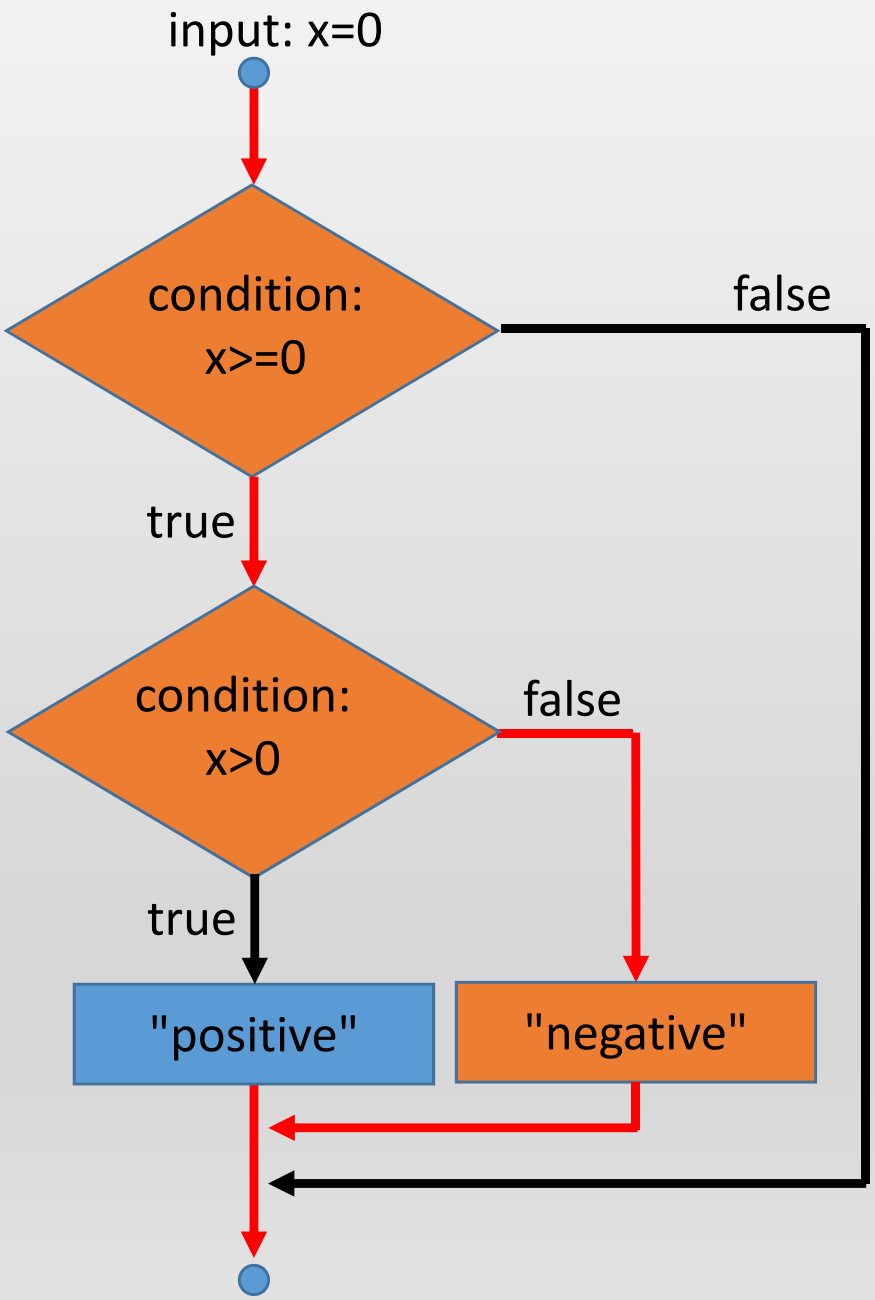
Program A



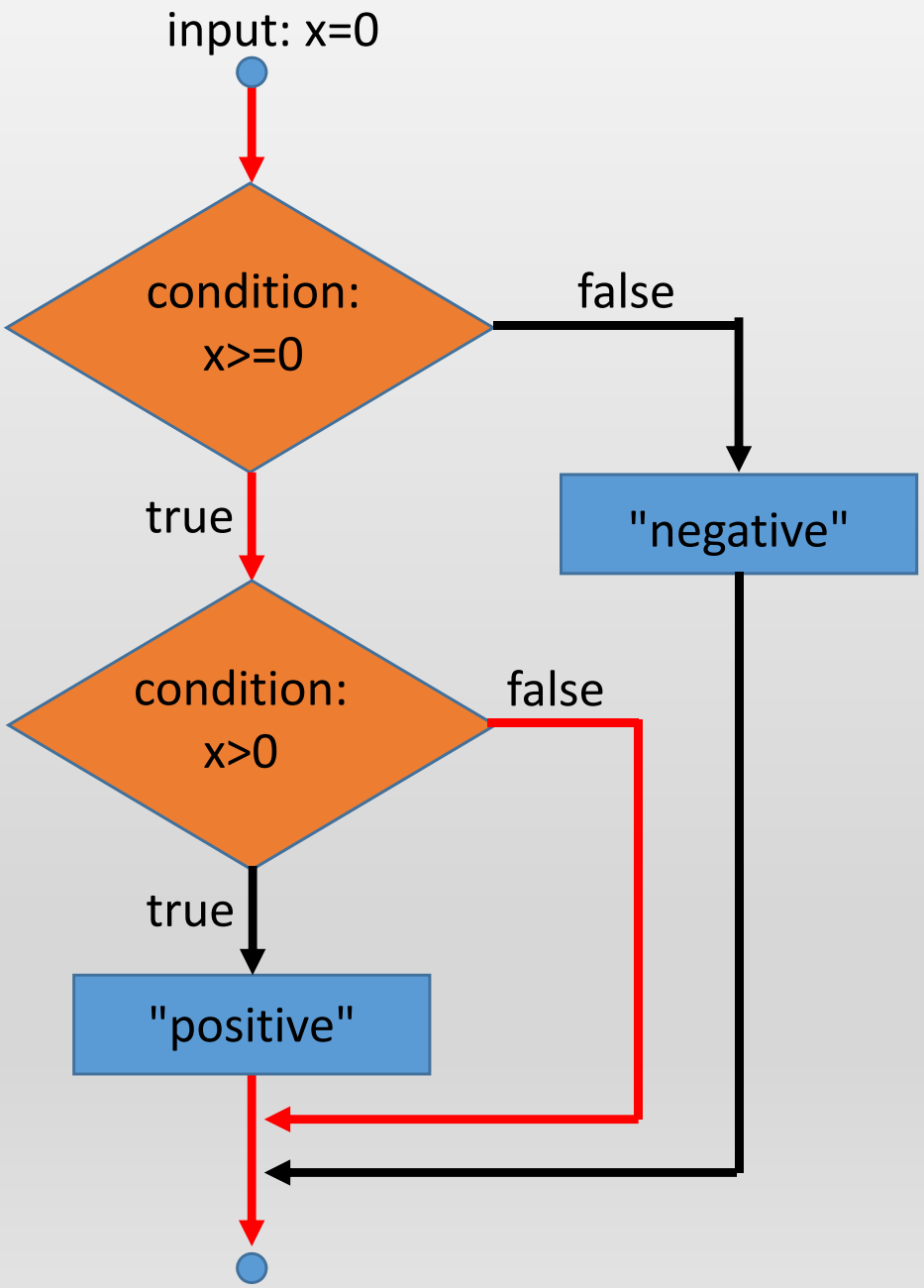
Program B

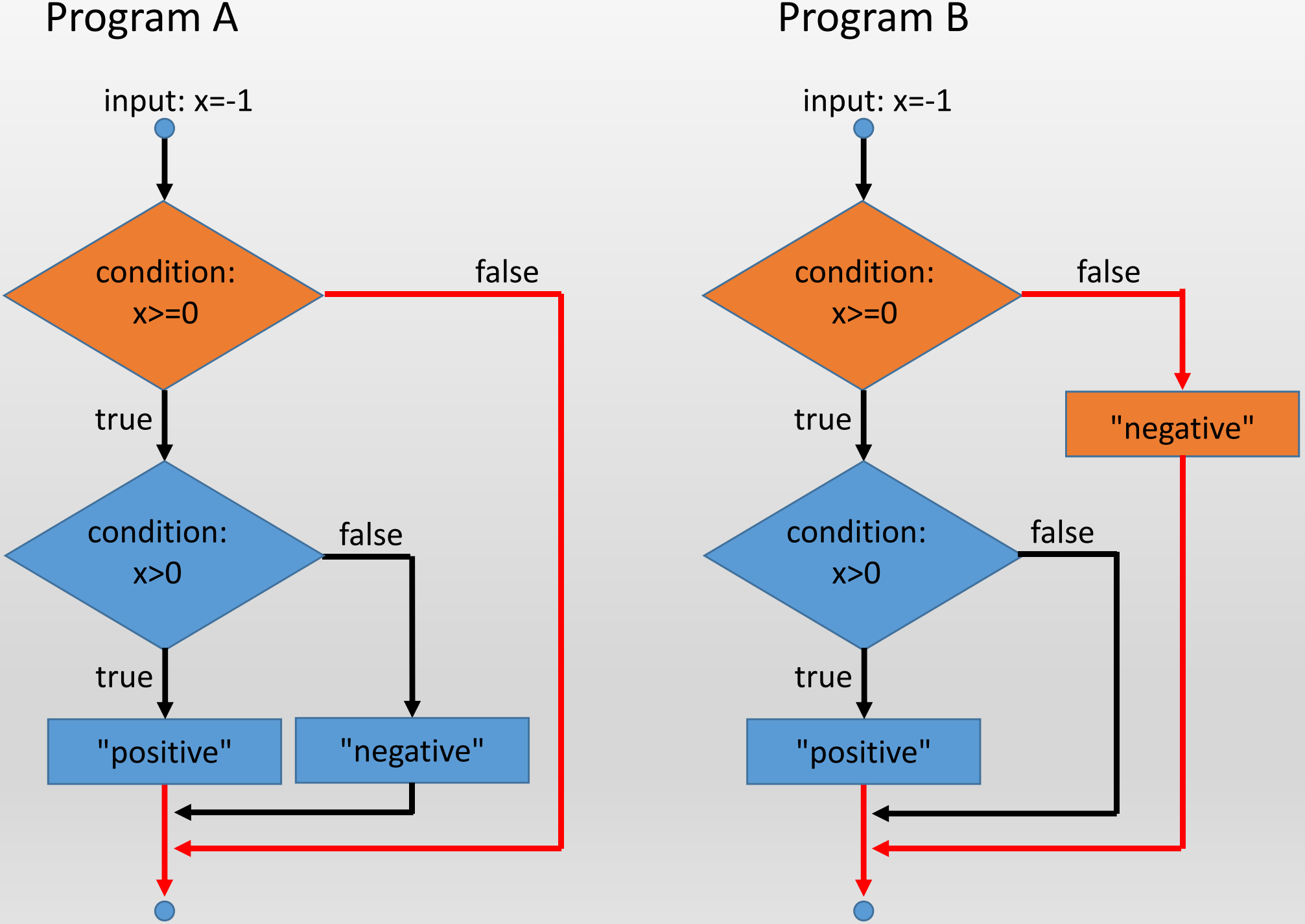


Program A



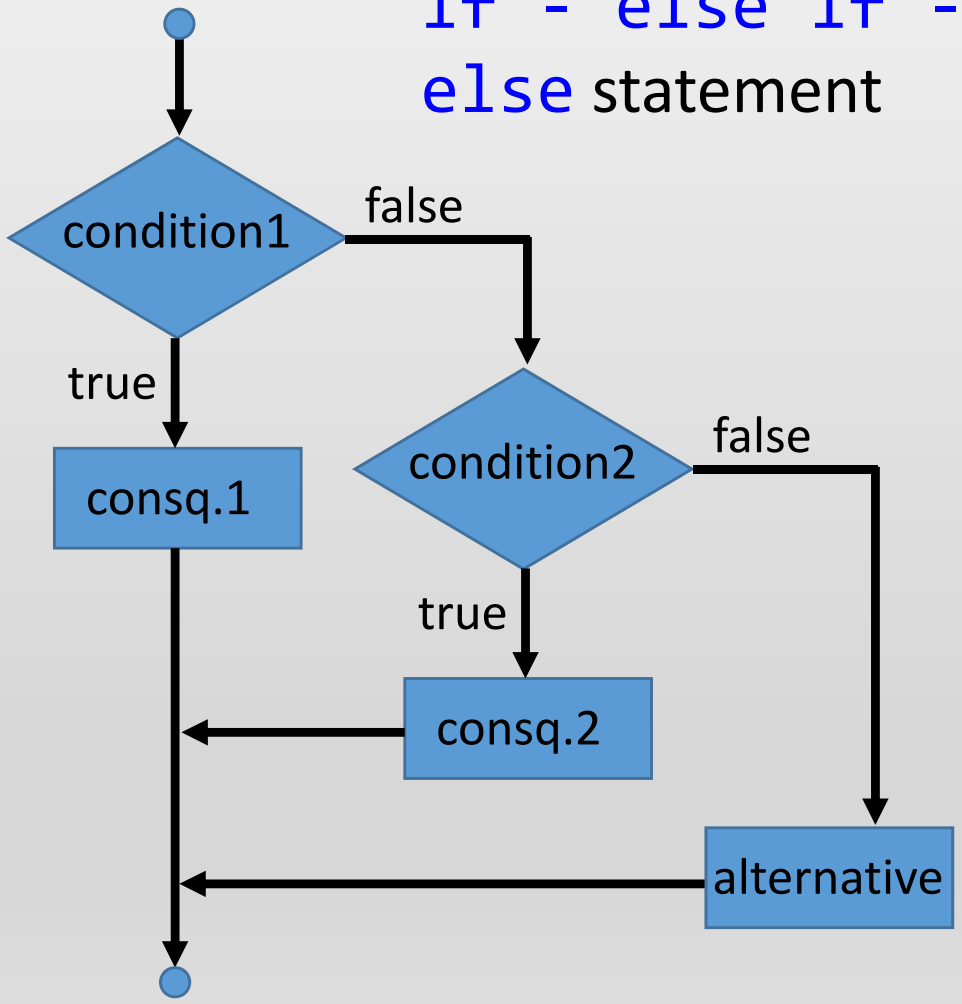
Program B




Program A
input: x=-1
condition:
x>=0
false
true
condition:
x>0
false
true
"positive"
"negative"
Program B
input: x=-1
condition:
x>=0
false
"negative"
true
condition:
x>0
false
true
"positive"

In an if-else statement, after the first condition fails, we might want to check the second condition before going to the next one. The **if - else if -else** structure can be used for such situation.

```
if (condition1)
      consequent1
else if (condition2)
      consequent2
else
      alternative
```



Example program 2.2 write a program that prompts user to enter height (m) and weight (kg), then evaluate BMI.

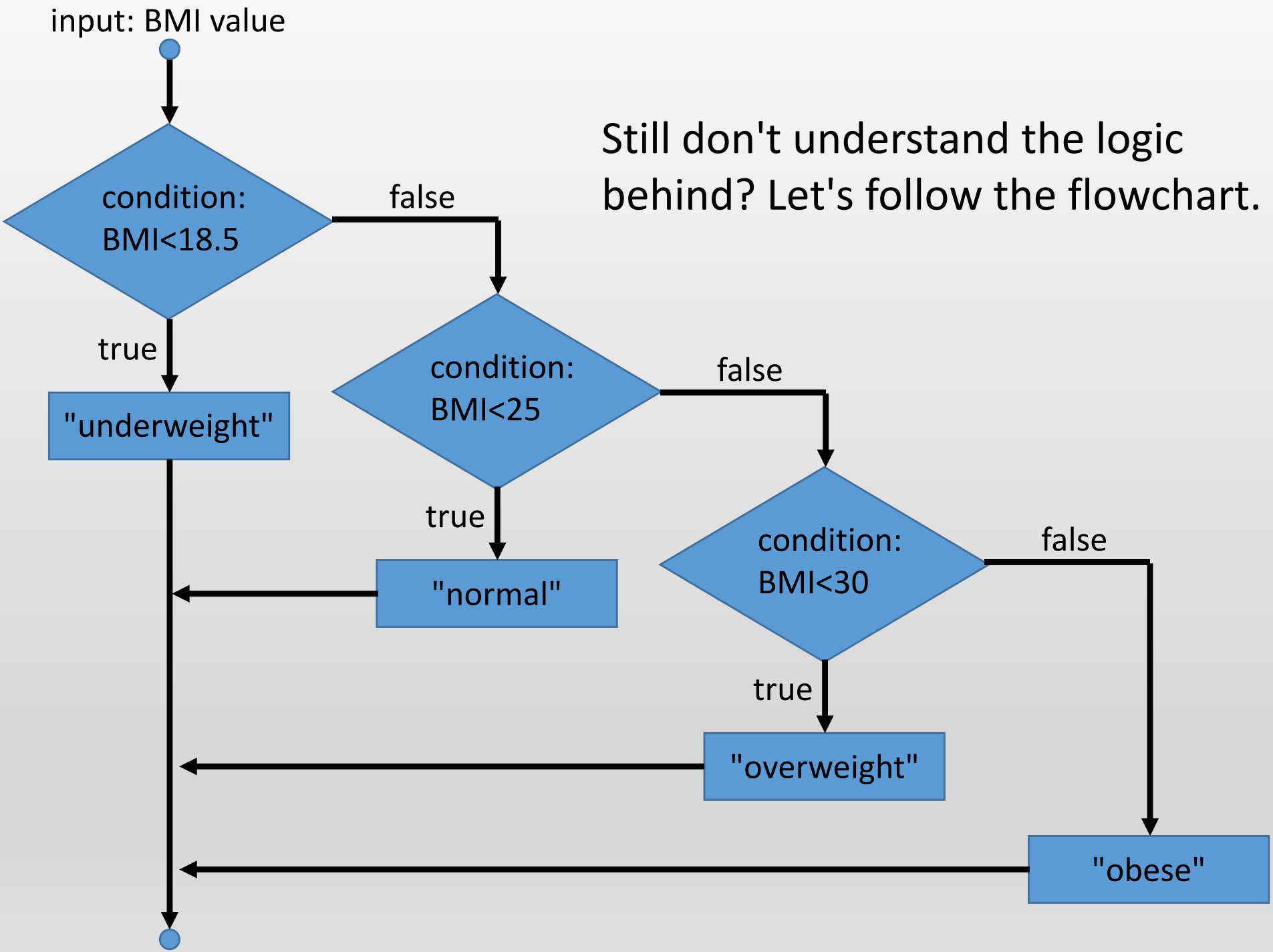
```
1  // 2.2 if-else Statement
2  #include <iostream>
3  using namespace std;
4  int main() {
5          cout << "Enter weight in kilograms: ";
6          double weight;
7          cin >> weight;
8          cout << "Enter height in meters: ";
9          double height;
10         cin >> height;
11         double bmi = weight / (height * height);
12         cout << "BMI is " << bmi << endl;
13         if (bmi < 18.5)
14                 cout << "Underweight" << endl;
15         else    if (bmi < 25)
16                         cout << "Normal" << endl;
17                 else    if (bmi < 30)
18                                 cout << "Overweight" << endl;
19                         else cout << "Obese" << endl;
20         return 0;
21 }
```

prompt user to input weight

prompt user to input height

calculate and output BMI

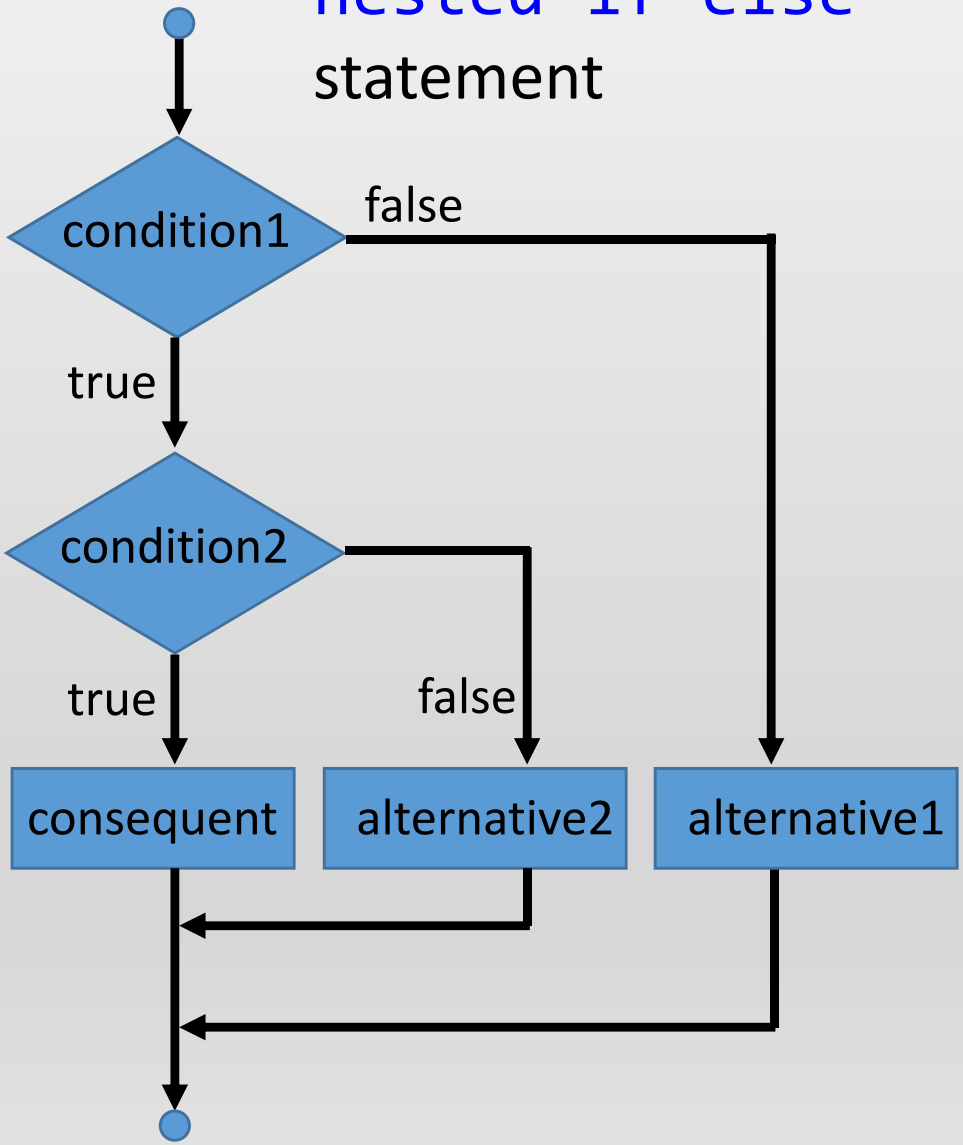
match the BMI to one of the four ranges



Still don't understand the logic behind? Let's follow the flowchart.

Another common structure in programming is the **nested-if** and **nested-if-else** structure. If the condition is satisfied, we will check the second condition.

```
if (condition1)
      if (condition2)
            consequent;
      else
            alternative2;
else
      alternative1;
```



Example 2.3 A scholarship is open for year 2 or above students with GPA 3.0 or above to apply. Write a program using nested-if-else structure to evaluate the eligibility of the potential applicant.

```
1  // 2.3 nested if-else structure
2  #include <iostream>
3  using namespace std;
4  int main() {
5          int yr;
6          double gpa;
7          cout << "Enter your year of study: ";
8          cin >> yr;
9          if (yr >= 2) {
10                 cout << "Enter your GPA: ";
11                 cin >> gpa;
12                 if (gpa >= 3.0)
13                         cout << "Your application is received.";
14                 else
15                         cout << "Sorry, your GPA is too low.";
16         else
17                 cout << "Sorry, you are under year 2.";
18         return 0;
19 }
```

check for year condition

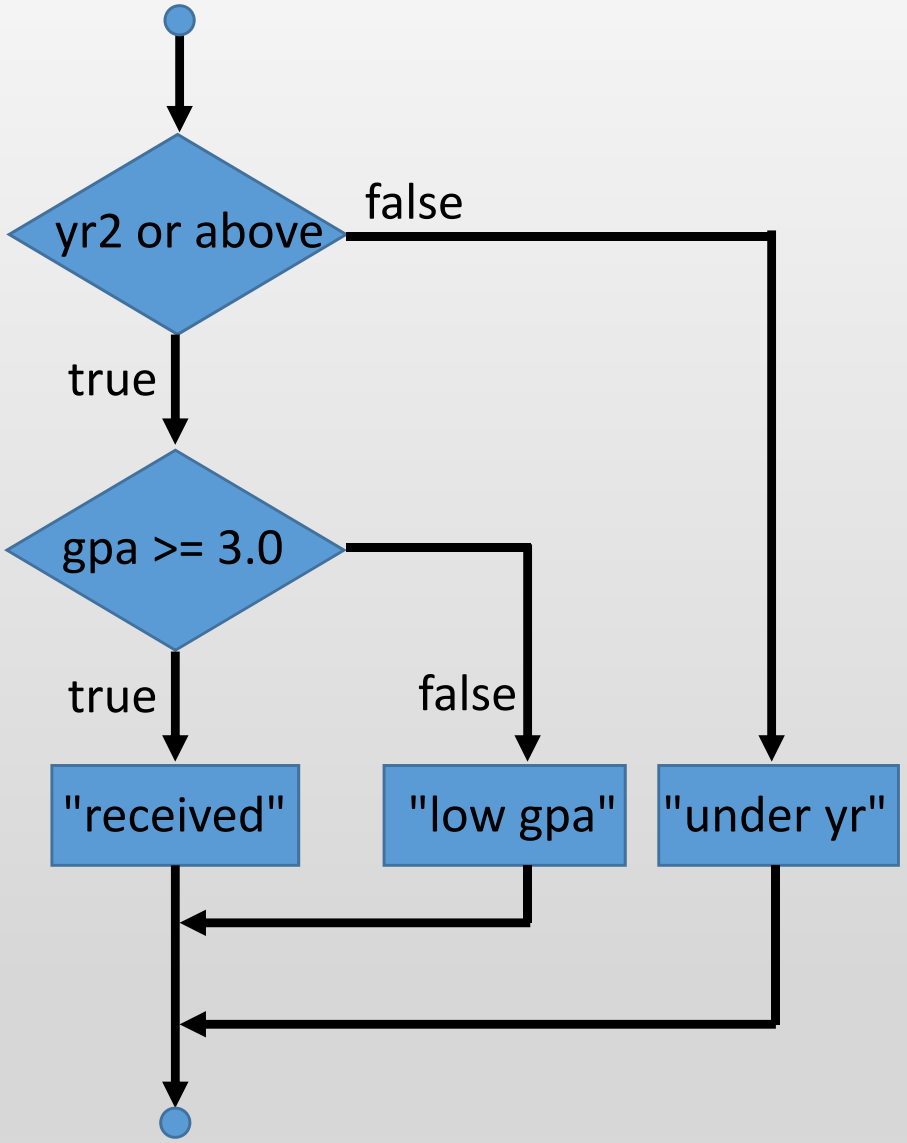
check for GPA condition

Some sample inputs and outputs:

```
Enter your year of study: 1
Sorry, you are under year 2
```

```
Enter your year of study: 3
Enter your GPA: 2.8
Sorry, your GPA is too low
```

```
Enter your year of study: 4
Enter your GPA: 3.7
Your application is received.
```



logical operations

We can form a new logical statement based on one, two or even more logical statements together by using logical operation.

Name	Operation	Mathematical symbol	C++ Expression
logical negation	`not(P)`	¬	!
Logical conjunction	`P and Q`	∧	&&
Logical disjunction	`P or Q`	∨	||

Logical AND (&&) is a binary operator that returns true if and only if both of the conditions are true; otherwise, it returns false.

Logical OR (||) is a binary operator that returns true if and only if either or both of the conditions are true; otherwise, it returns false.

Negation (!) is a unary operator which takes one condition only. The operator returns opposite value as the condition.

The results of **logical operators** is explained by a **truth-table**.

Unary

P	not(P)
True	False
False	True

Binary

P	Q	P and Q	P or Q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Precedence of different operations

It tells which operator should be done first and then next on a statement of the program.

Operator	Operation	Precedence
()	Parentheses	First. If the parentheses are nested, the innermost pair is evaluated first. The order of evaluation of two sets of parentheses that are not nested, however, is not specified in the C++ standard.
!	Negation	Second. If there are several, they're evaluated from right to left.
* / %	Multiplication, Division, Modulus	Second. If there are several, they're evaluated from left to right.
+ -	Addition, Subtraction	Third. If there are several, they're evaluated from left to right.
< <= > >=	Relational	Third. If there are several, they're evaluated from left to right.
== !=	Equality	Forth. If there are several, they're evaluated from left to right.
&&	Logical AND	Fifth. If there are several, they're evaluated from left to right.
||	Logical OR	Sixth. If there are several, they're evaluated from left to right.

Classwork exercise 2.1

Assuming that x is 1, show the result of the following Boolean expressions:

a) (true) && (3 > 4)

b) !(x > 0) && (x > 0)

c) (x > 1) || (x < 1)

d) (x != 0) || (x == 0)

e) (x >= 0) || (x < 0)

f) (x != 1) == !(x == 1)

Solution to classwork exercise

Assuming that x is 1, show the result of the following Boolean expressions:

a) (true) && (3 > 4) 0

b) !(x > 0) && (x > 0) 0

c) (x > 1) || (x < 1) 0

d) (x != 0) || (x == 0) 1

e) (x >= 0) || (x < 0) 1

f) (x != 1) == !(x == 1) 1

We may combine several logical expressions (Boolean values) together to evaluate if the condition is true or false.

Using logical conjunction (and), the consequence will be executed only if all the conditions are True.

```
if (cond1 && cond2 && cond3)
      consequent;
```

Using logical disjunction (or), the consequence will be executed if either one of the conditions is True.

```
if (cond1 || cond2 || cond3)
      consequent;
```

In the next example, we will write a program that prompts users to enter three angles in integer degree, then determine what kind of triangle is formed by these angles.

Example program 2.4

```
// 2.4 logical operation
#include <iostream>
using namespace std;
int main()
{
int a,b,c;
cout << "Enter three angles in degree: ";
cin >> a >> b >> c;
if (a>0 && b>0 && c>0 && (a+b+c==180))
        if (a==90 || b==90 || c==90) cout << "right angled triangle" << endl;
        else if (a>90 || b>90 || c>90) cout << "obtuse triangle" << endl;
                else cout << "acute triangle" << endl;
else cout << "not a triangle" << endl;
return 0;
}
```

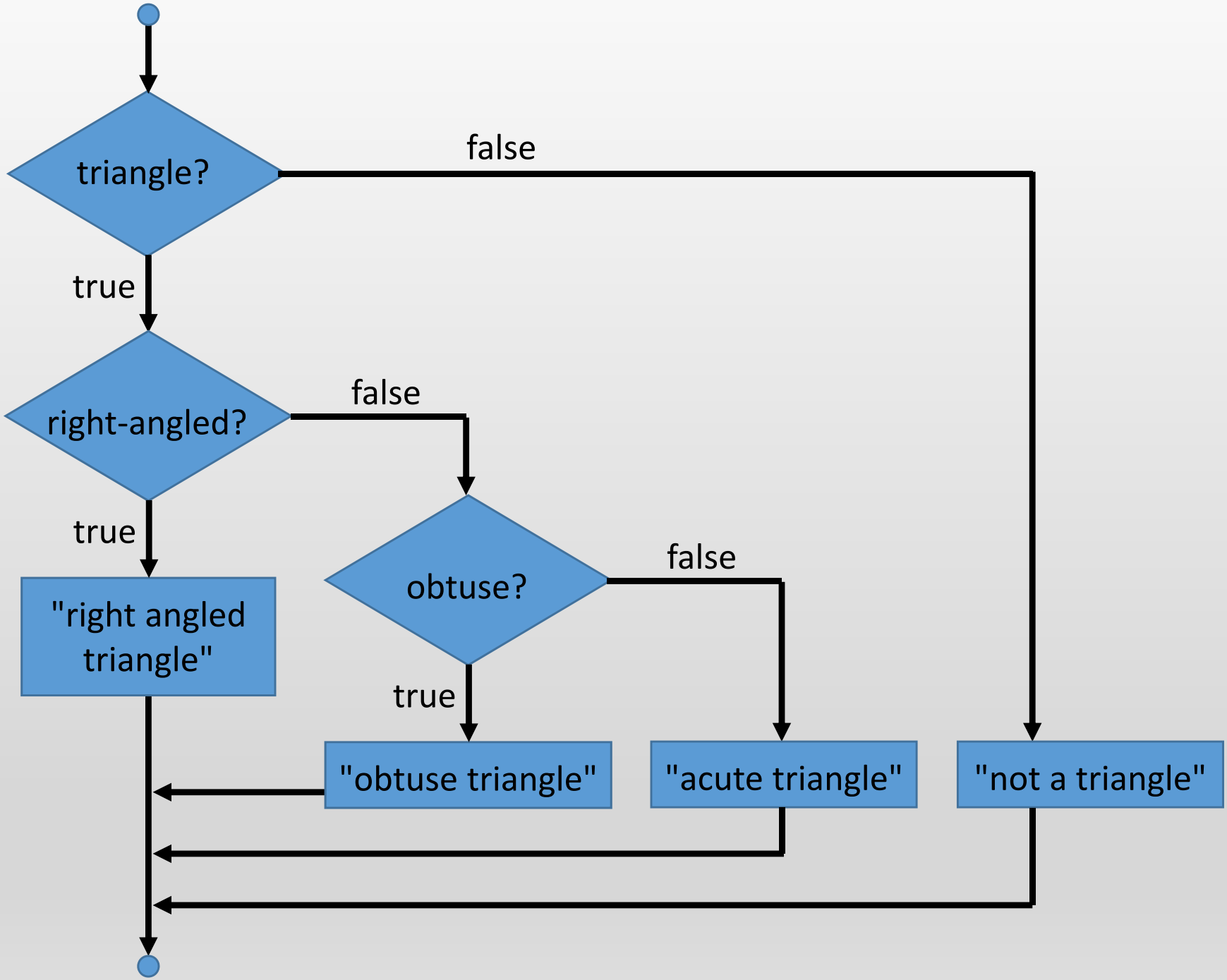
```
if (a>0 && b>0 && c>0 && (a+b+c==180))
        if (a==90 || b==90 || c==90) cout << "right angled triangle" << endl;
        else    if (a>90 || b>90 || c>90) cout << "obtuse triangle" << endl;
                else cout << "acute triangle" << endl;
else cout << "not a triangle" << endl;
```

check if all angles are positive also the angle sum is 180 degree, if yes triangle, if no not triangle

check if any one of the three angles is 90 degree, if yes right angled triangle

check if any one of the three angles is greater than 90 degree, if yes obtuse triangle

if it is a triangle but fail to be right angled or obtuse, then it is an acute triangle


triangle?
false
true
right-angled?
false
true
"right angled triangle"
obtuse?
false
true
"obtuse triangle"
"acute triangle"
"not a triangle"

Conditional expression

You might want to simplify your code especially when there are too many if-else statements. Consider the following:

```
if (x > 0)
      y = 1;
else
      y = -1;
```

```
y = x > 0 ? 1 : -1;
```

The conditional expression on the right is equivalent to the if-else statement. The syntax is:
(condition)?(consequent value):(alternative value)

Switch statement

Using if-else statements for several cases could be quite redious. We can use **switch statement** to execute operations based on the value of a variable or an expression. If none of the cases are fulfilled, the default statements will be executed. The syntax is as follows:

```
switch (switch_expression)
{
      case value1: consequent1; break;
      case value2: consequent2; break;
      ……
      case valueN: consequentN; break;
      default: default_consequent;
}
```

Example 2.5 Suppose `grade` is a `double` variable. We would like to convert the letter grade into an integer score.

```
 1 // 2.5 switch statement
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6           char grade;
 7           double score;
 8           cout << "Enter grade: ";
 9           cin >> grade;
10           switch (grade)
11           {
12                     case 'A': score = 4; break;
13                     case 'B': score = 3; break;
14                     case 'C': score = 2; break;
15                     case 'D': score = 1; break;
16                     case 'F': score = 0; break;
17                     default: score = 0;
18           }
19           cout << "Your score is: " << score;
20           return 0;
21 }
```

Mathematical constants in C++

You need to add the header `#include <cmath>` in order to use the mathematical functions and constants in the cmath library. There are a few common mathematical constants that have been defined in the <cmath> library.

Algebraic Expression	Rounded value	C++ Expression
π	3.14159	`M_PI`
e	2.71828	`M_E`
$\pi/2$	1.5708	`M_PI_2`
$\sqrt{2}$	1.41421	`M_SQRT2`

Warning: case sensitive!

Mathematical functions in C++

There are already some basic arithmetic operations in C++. In the <cmath> library, you can find more mathematical functions.

Comparison functions (already in C++):

Operation	Function	Algebraic Expression	C++ Expression
maximum	`max(,)`	$\max(a, b)$	`max(a,b)`
minimum	`min(,)`	$\min(a, b)$	`min(a,b)`
absolute value	`abs()`	$\lvert a \rvert$	`abs(a)`

Trigonometric functions (in radian):

sine	`sin()`	$\sin a$	`sin(a)`
cosine	`cos()`	$\cos a$	`cos(a)`
tangent	`tan()`	$\tan a$	`tan(a)`
arc sine	`asin()`	$\sin^{-1} a$	`asin(a)`
arc cosine	`acos()`	$\cos^{-1} a$	`acos(a)`
arc tangent	`atan()`	$\tan^{-1} a$	`atan(a)`

Power, exponential and logarithmic functions:

Operation	Function	Algebraic Expression	C++ Expression
power	`pow(,)`	a^b	`pow(a,b)`
square root	`sqrt()`	$\sqrt{a}$	`sqrt(a)`
exponential	`exp()`	e^a	`exp(a)`
natural log	`log()`	$\ln a$	`log(a)`
binary log	`log2()`	$\log_2 a$	`log2(a)`
decimal log	`log10()`	$\log_{10} a$	`log10(a)`

Rounding functions:

Operation	Function	Algebraic Expression	C++ Expression
Round up	`ceil()`	$\lceil a \rceil$	`ceil(a)`
Round down	`floor()`	$\lfloor a \rfloor$	`floor(a)`
Truncation	`trunc()`	$[a] = \begin{cases} \lfloor a \rfloor, a \geq 0 \\ \lceil a \rceil, a < 0 \end{cases}$	`trunc(a)`

*Warning: The expression [ ] (integer part) is ambiguous. It can be defined as floor or truncation on different textbooks.

Classwork exercise 2.2

Compute the values in column (i) by your mind and those in column (ii) by using C++. Compare and see if they are equivalent.

	(i) Mathematical expression	(ii) C++ Expression
a)	$\frac{\sqrt{4}}{5}$	`sqrt(4)/5`
b)	$\sin(2\pi)$	`sin(2 * M_PI)`
c)	$\ln e^{999}$	`log(pow(M_E,999))`
d)	$e^{2 \ln 3}$	`pow(M_E,2*log(3.0))`
e)	$[-9.999]$	`trunc(-9.999)`

Notes to the solution:

(b) M_PI is a constant in the form of float point value, with only a finite number of digits. It has a tiny difference from the actual π.

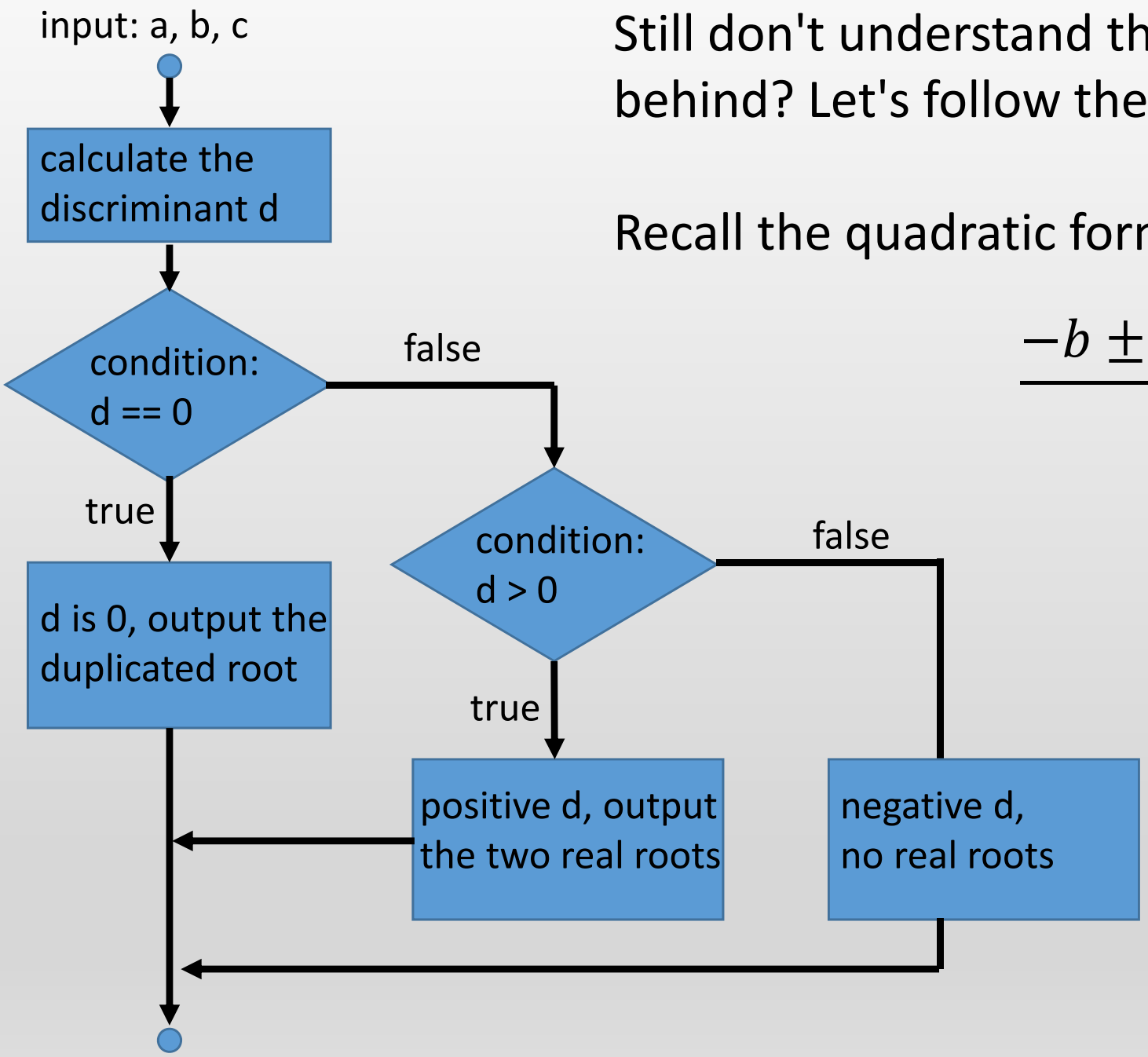
(c) pow(M_E,999) is evaluated first, which cause value overflow.

	(i) Mathematical expression		(ii) C++ Expression	
a)	$\frac{\sqrt{4}}{5}$	0.4	`sqrt(4)/5`	0.4
b)	$\sin(2\pi)$	0	`sin(2 * M_PI)`	-2.44929e-016
c)	$\ln e^{999}$	999	`log(pow(M_E,999))`	inf
d)	$e^{2\ln 3}$	9	`pow(M_E,2*log(3.0))`	9
e)	$[-9.999]$	-9	`trunc(-9.999)`	-9

Example program 2.6

Write a program that solves a quadratic equation ($a \neq 0$).

```
 1 // 2.6 Quadratic equation
 2 #include <iostream>
 3 #include <cmath>
 4 using namespace std;
 5 int main()
 6 {
 7          double a, b, c, d;
 8          cout << "a x^2 + b x + c = 0" << endl;
 9          cout << "Enter the values of coefficients a, b, c: ";
10          cin >> a >> b >> c;
11          d = pow(b,2) - 4 * a * c;
12          if (d==0)
13                    cout << "The duplicated root is: " << -b/2/a << endl;
14          else if (d>0)
15                    cout << "The two roots are: " << (-b+sqrt(d))/2/a
16                    <<  " and " << (-b-sqrt(d))/2/a << endl;
17                    else cout << "There are no real roots" << endl;
18          return 0;
19 }
```



Still don't understand the logic behind? Let's follow the flowchart.

Recall the quadratic formula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$