AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 2: Selection
Boolean variable,if and if-then statement,
logical operation, conditional expression,
random number, swapping

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)
"Confucius is an Korean philosopher." (FALSE)
"Mr Leung is a billionaire." (could be TRUE or FALSE, 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 such inequality or equality.

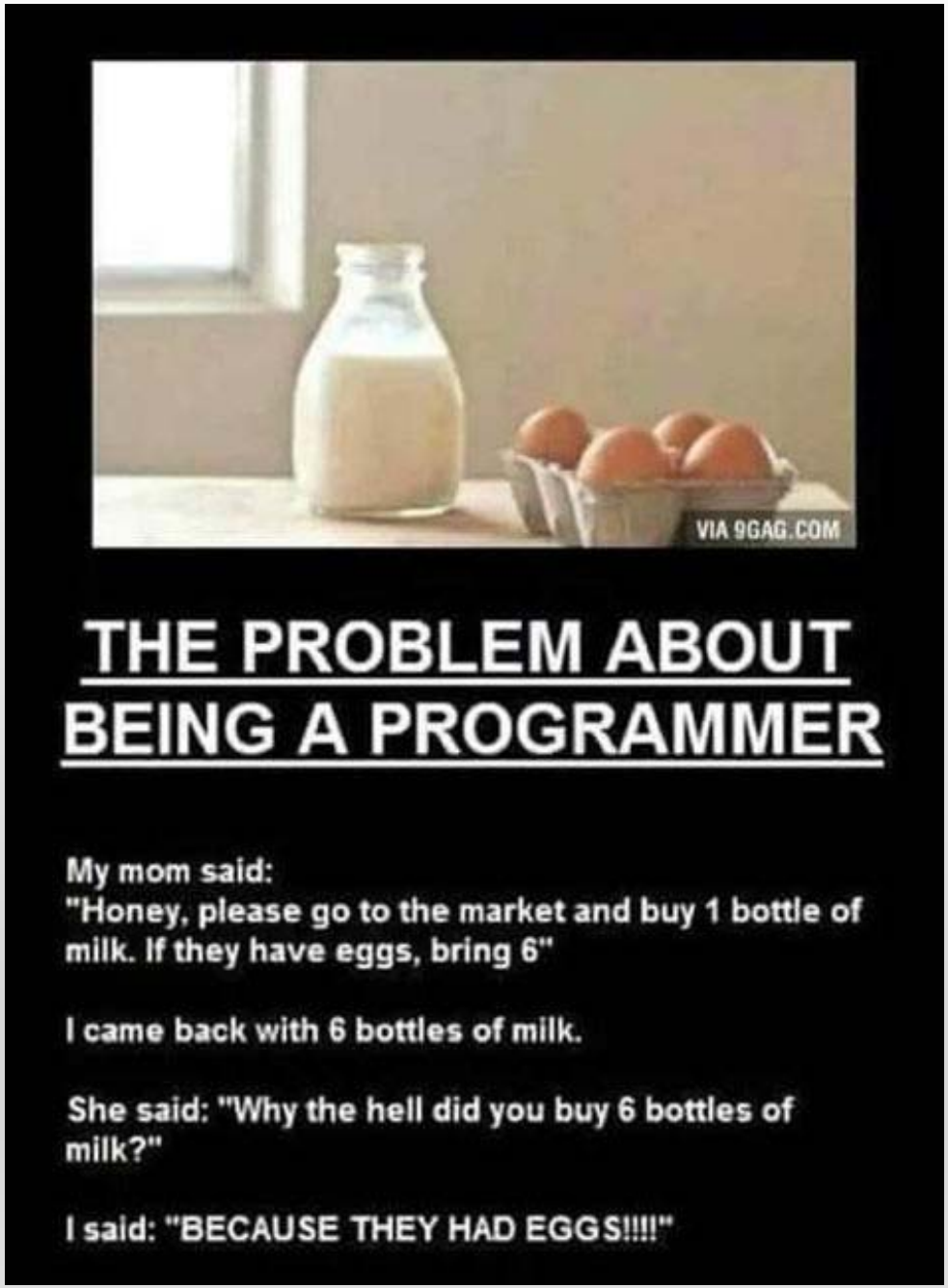
Name	Mathematical symbol	C++ Expression
equal to	$=$	==
not equal to	$\neq$	!=
greater than	$>$	>
greater than or equal to	$\geq$	>=
less than	$<$	<
less than or equal to	$\leq$	<=

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).

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.

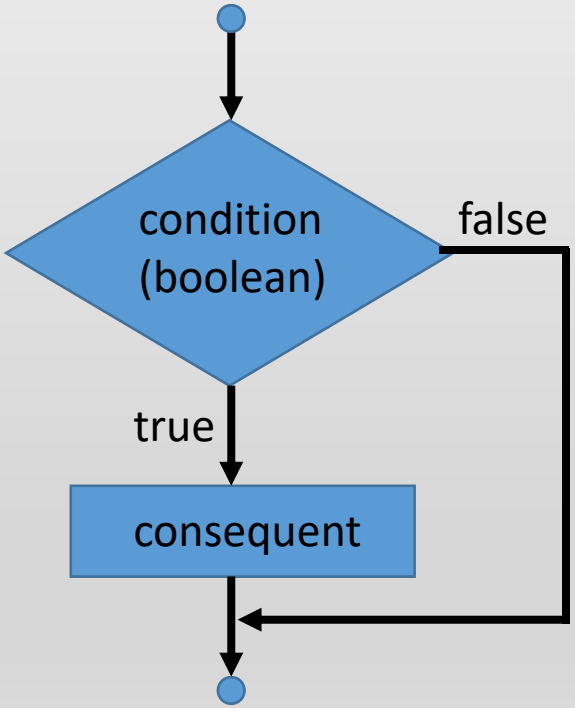
For an `if-else` statement, if the condition returns a false value, the alternative will be executed.



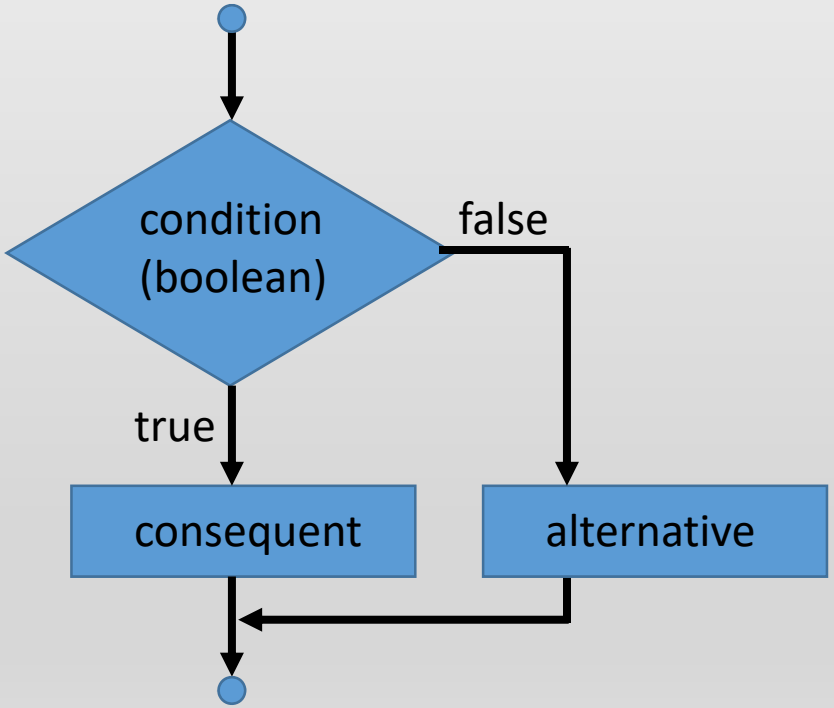

```
int milkToBuy = 1;
bool haveEggs;

cin >> haveEggs;

if (haveEggs)
        milkToBuy = 6;

cout << milkToBuy;
```

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 number;
 8         cout << "Enter an integer: ";
 9         cin >> number;
10
11         if (number % 5 == 0)
12                 cout << "HiFive" << endl;
13
14         if (number % 2 == 0)
15                 cout << "HiEven" << endl;
16
17         return 0;
18 }
19
```

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 number;
8          cout << "Enter an integer: ";
9          cin >> number;
10
11         if (number % 5 == 0)
12                 cout << "HiFive" << endl;
13
14         if (number % 2 == 0)
15                 cout << "HiEven" << endl;
16
17         return 0;
18 }
19
```

prompt user to input an integer

check if the integer is divisible by 5

check if the integer is divisible by 2

Example program 2.2

```
1 // 2.2 if-else Statement
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6         cout << "Enter weight in kilograms: ";
7         double weight;
8         cin >> weight;
9         cout << "Enter height in meters: ";
10        double height;
11        cin >> height;
12        double bmi = weight / (height * height);
13        cout << "BMI is " << bmi << endl;
14        if (bmi < 18.5)
15                cout << "Underweight" << endl;
16        else    if (bmi < 25)
17                cout << "Normal" << endl;
18                else    if (bmi < 30)
19                cout << "Overweight" << endl;
20                        else cout << "Obese" << endl;
21        return 0;
22 }
```

Example program 2.2

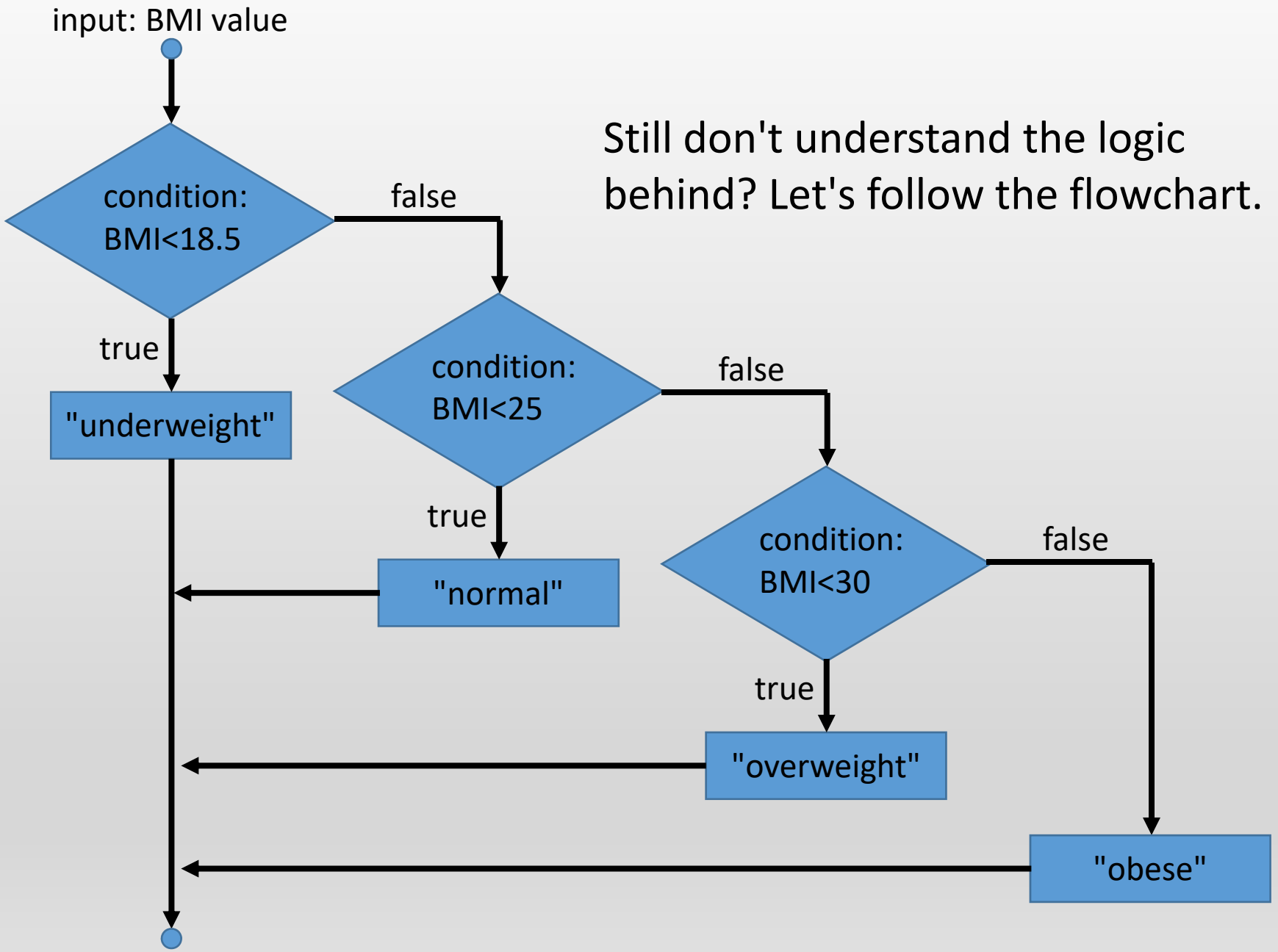
```
 1 // 2.2 if-else Statement
 2 #include <iostream>
 3 using namespace std;
 4 int main()
 5 {
 6         cout << "Enter weight in kilograms: ";
 7         double weight;
 8         cin >> weight;
 9         cout << "Enter height in meters: ";
10         double height;
11         cin >> height;
12         double bmi = weight / (height * height);
13         cout << "BMI is " << bmi << endl;
14         if (bmi < 18.5)
15                 cout << "Underweight" << endl;
16         else    if (bmi < 25)
17                 cout << "Normal" << endl;
18                 else    if (bmi < 30)
19                 cout << "Overweight" << endl;
20                         else cout << "Obese" << endl;
21         return 0;
22 }
```

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.

Becareful!

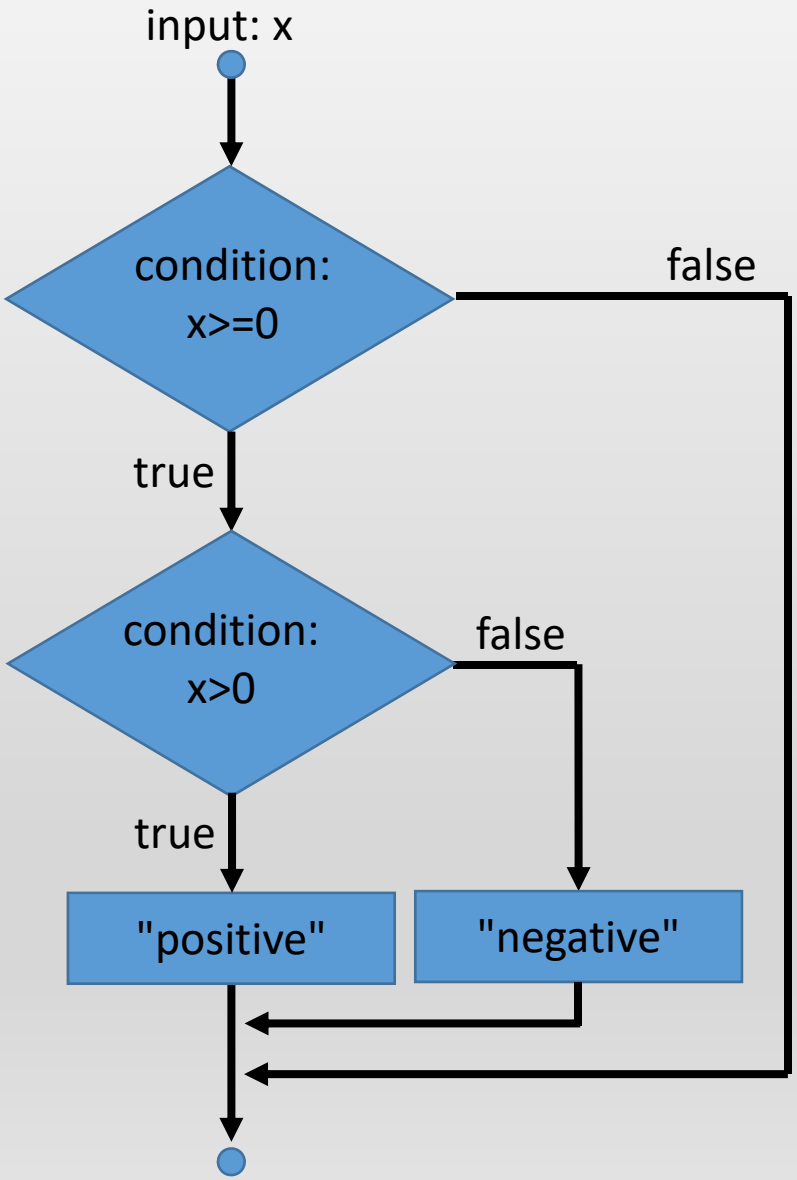
In an `if-else` statement, the `else` corresponds to the closest previous `if` in the same block. Compare the following programs, which one is correct? Test them by using different input values.

```
#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;
}
```

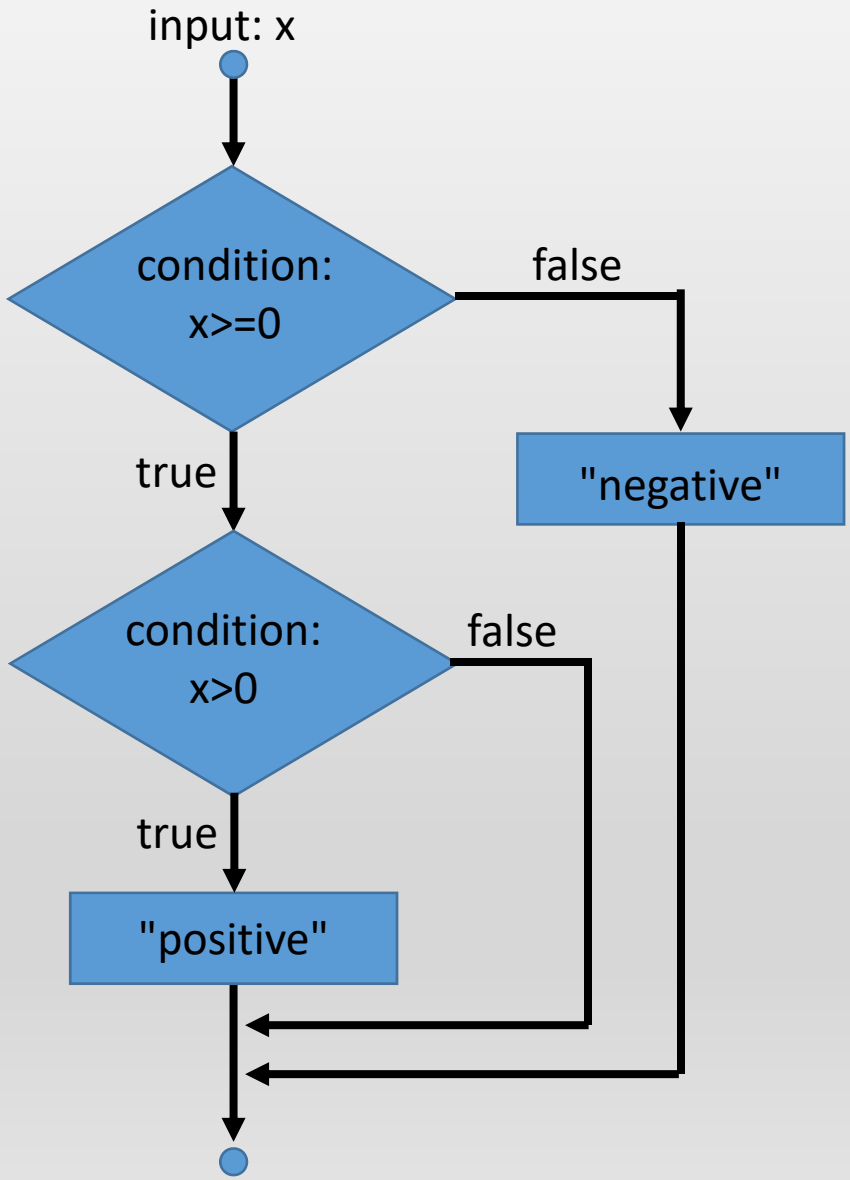
```
#include <iostream>
using namespace std;
int main()
{
int x;
cin >> x;
if (x>=0)
{
if (x>0) cout<<"positive"<<endl;
}
else cout<<"negative"<<endl;
return 0;
}
```

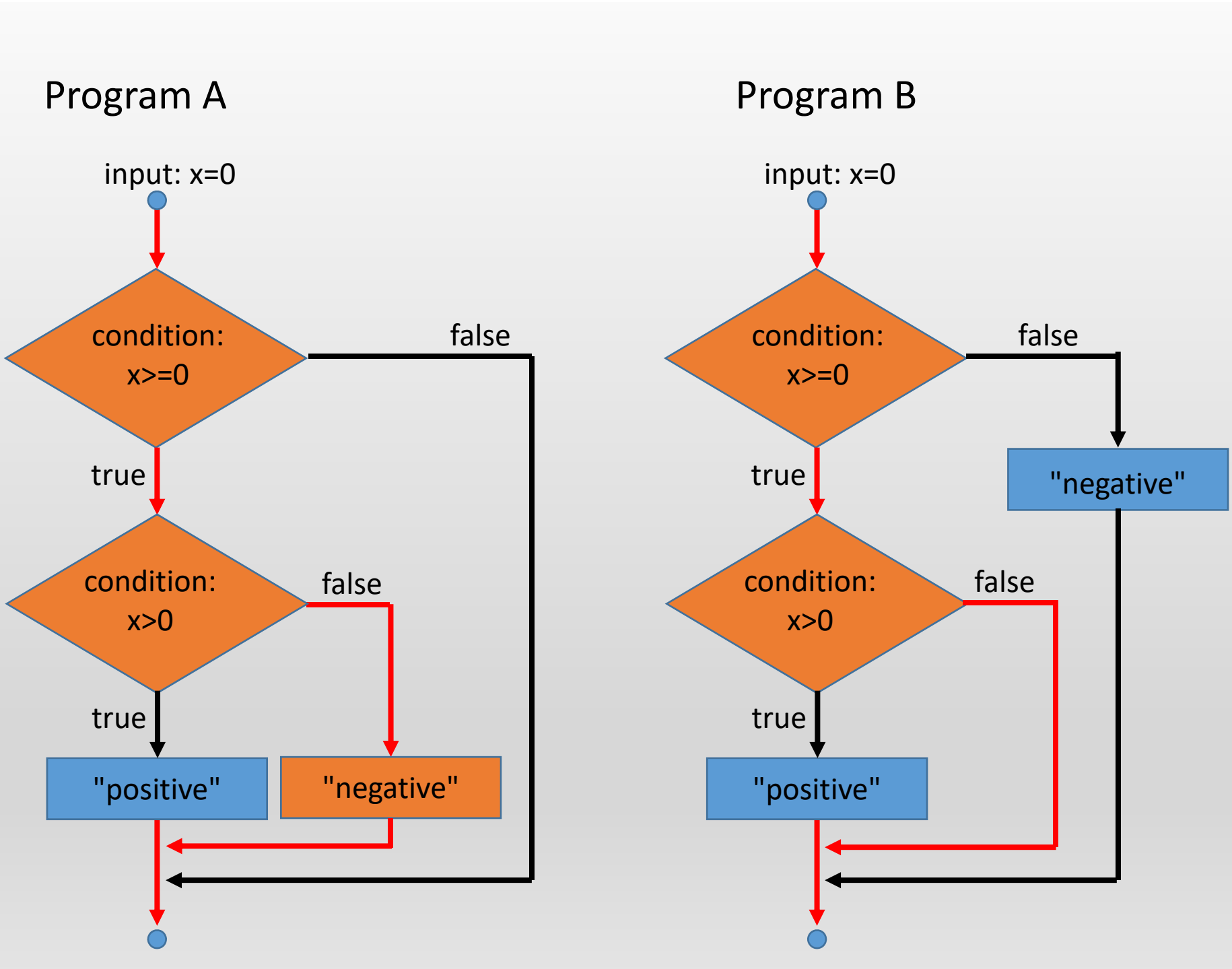
As a good habit, use intent correctly to identify the corresponding `if-else`. Use `{ }` to enclose several statements as a block.

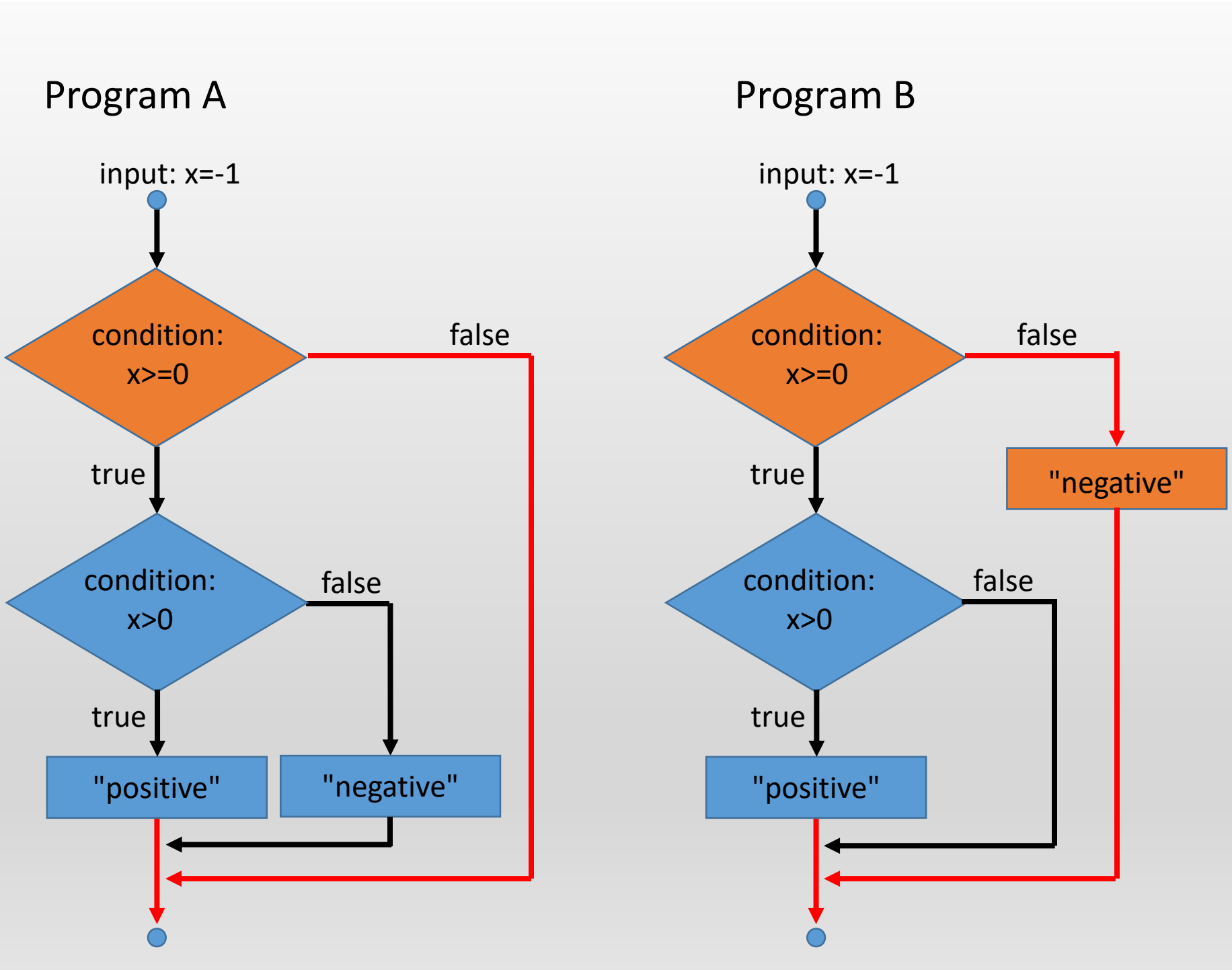
Program A



Program B




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


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"

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.

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

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

Example program 2.3

```
// 2.3 logical operation
#include <iostream>
using namespace std;
int main()
{
int a;
int b;
int 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

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)

Random number

Programs like generating a verification code or a gambling game usually involve randomness.

To generate a random number, use the `rand()` function in the `<cstdlib>` header file. This function returns a random integer between 0 and RAND_MAX which is platform-dependent constant, it is at least 32767.

Random integer from 0 to N: `rand() % (N+1)`

Random integer from 1 to N: `rand() % N + 1`

Random float point value from 0 to 1: `rand()/(RAND_MAX+1)`

However, when you run a simple program with `rand()` for several times, you will observe that every time it gives you the same number. The `rand()` function's algorithm uses a value called the seed to control how to generate the numbers. By default, the seed value is 1.

To change the seed, use the `srand()` function in the `<cstdlib>` header file. To ensure that the seed value is different each time you run the program, use `time(0)` which returns the current time in seconds elapsed since the time 00:00:00 on January 1, 1970 GMT. This function is included in the `<ctime>`

Example program 2.4

```
1  // 2.4 Generating Random Numbers
2  #include <iostream>
3  #include <ctime> // for time function
4  #include <cstdlib> // for rand and srand functions
5  using namespace std;
6  int main()
7  {
8           srand(time(0));
9           int number1 = rand() % 10;
10          int number2 = rand() % 10;
11          if (number1 < number2)
12          {
13                    int temp = number1;
14                    number1 = number2;
15                    number2 = temp;
16          }
17          cout << "What is " << number1 << " - " << number2 << "? ";
18          int answer;
19          cin >> answer;
20          if (number1 - number2 == answer)
21                    cout << "You are correct!";
22          else
23                    cout << "Your answer is wrong." << endl << number1
24                    << " - " << number2 << " should be "
25                    << (number1 - number2) << endl;
26          return 0;
27 }
```

Example program 2.4

```
1  // 2.4 Generating Random Numbers
2  #include <iostream>
3  #include <ctime> // for time function
4  #include <cstdlib> // for rand and srand functions
5  using namespace std;
6  int main()
7  {
8          srand(time(0));
9          int number1 = rand() % 10;
10         int number2 = rand() % 10;
11         if (number1 < number2)
12         {
13                 int temp = number1;
14                 number1 = number2;
15                 number2 = temp;
16         }
17         cout << "What is " << number1 << " - " << number2 << "? ";
18         int answer;
19         cin >> answer;
20         if (number1 - number2 == answer)
21                 cout << "You are correct!";
22         else
23                 cout << "Your answer is wrong." << endl << number1
24                 << " - " << number2 << " should be "
25                 << (number1 - number2) << endl;
26         return 0;
27 }
```

Generate two random single-digit integers

Make sure no1 >= no2. If not, swap them.

Prompt student to input the answer.

Check the answer and display the result.

Classwork Exercise

Write a program that simulates a rolling die, then ask users to guess the number of dots.

Samples:

```
Guess the number of dots show up:5
You win!
```

```
Guess the number of dots show up:5
You lose!
It should be 2
```

```cpp
// 2.5 Classwork
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;
int main()
{
	srand(time(0));
	int die = rand() % 6 + 1;
	cout << "Guess the number of dots shown up: ";
	int guess;
	cin >> guess;
	if (guess == die)
		cout << "You win!";
	else
		cout << "You lose!" << endl << "It should be "
		<< die << endl;
	return 0;
}
```

Swapping

In the previous example, we have seen a script that swap the values of two variables. Some of you might wonder why a temporary variable was declared. Suppose x=1 and y=2 are integers. Compare the following and deduce the updated value of x and y. Which of these algorithms can swap values of x, y?

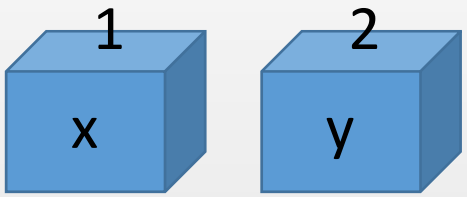
```
x = y;
y = x;
```

```
temp = x;
x = y;
y = temp;
```

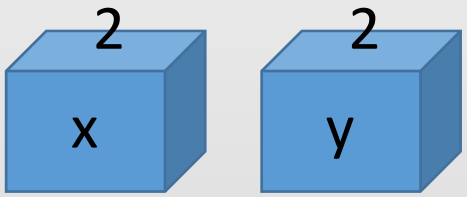
```
x = x + y;
y = x - y;
x = x – y;
```

```
x = y;
y = x;
```

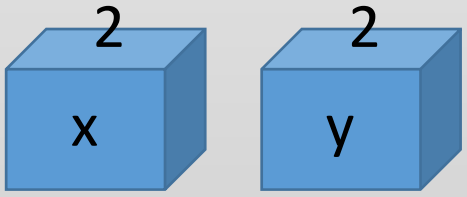
```
x = 1;
y = 2;
```




```
x = y;
```

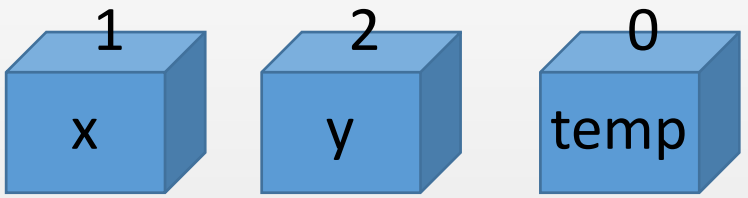



```
y = x;
```

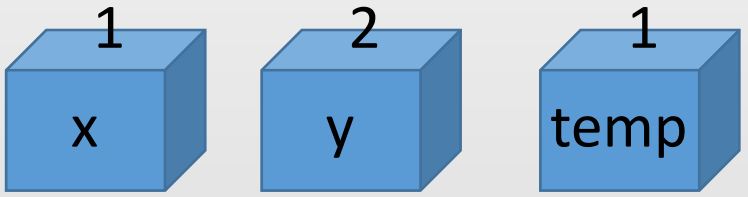


```
temp = x;
x = y;
y = temp;
```

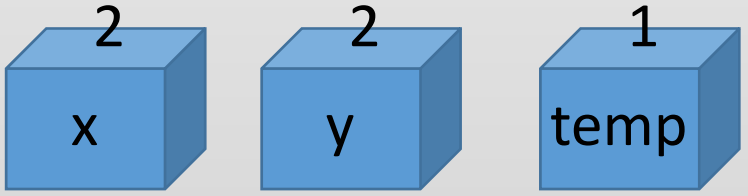
```
x = 1;
y = 2;
```



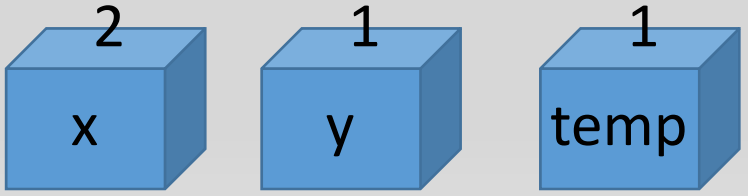

```
temp = x;
```




```
x = y;
```

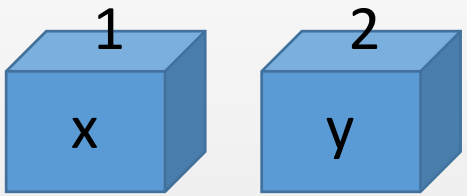



```
y = temp;
```

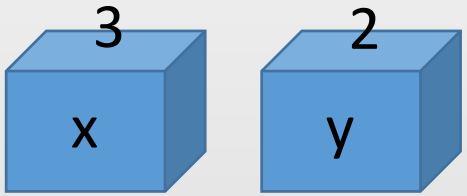


```
x = x + y;
y = x - y;
x = x - y;
```

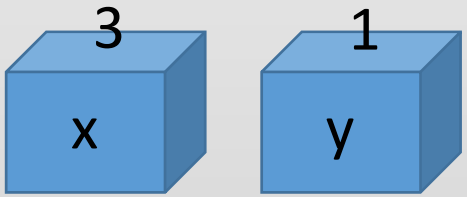
```
x = 1;
y = 2;
```




```
x = x + y;
```




```
y = x - y;
```




```
x = x - y;
```