AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 3:
Mathematical constants, Mathematical functions, characters, strings
formatting output

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

Mathematical functions in C++

In the <cmath> library, there are also a number of very useful mathematical functions. Notice that for trigonometric functions radian is used.

Trigonometric functions:

Operation	Function	Algebraic Expression	C++ Expression
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)`
decimal log	log10()	$\log_{10} a$	`log10(a)`

Rounding and selection functions:

Operation	Function	Algebraic Expression	C++ Expression
Round up	`ceil()`	$\lceil a \rceil$	`ceil(a)`
Round down	`floor()`	$\lfloor a \rfloor$	`floor(a)`
maximum	`max(,)`	$\max(a, b)$	`max(a,b)`
minimum	`min(,)`	$\min(a, b)$	`min(a,b)`
absolute value	`abs()`	$\lvert a \rvert$	`abs(a)`

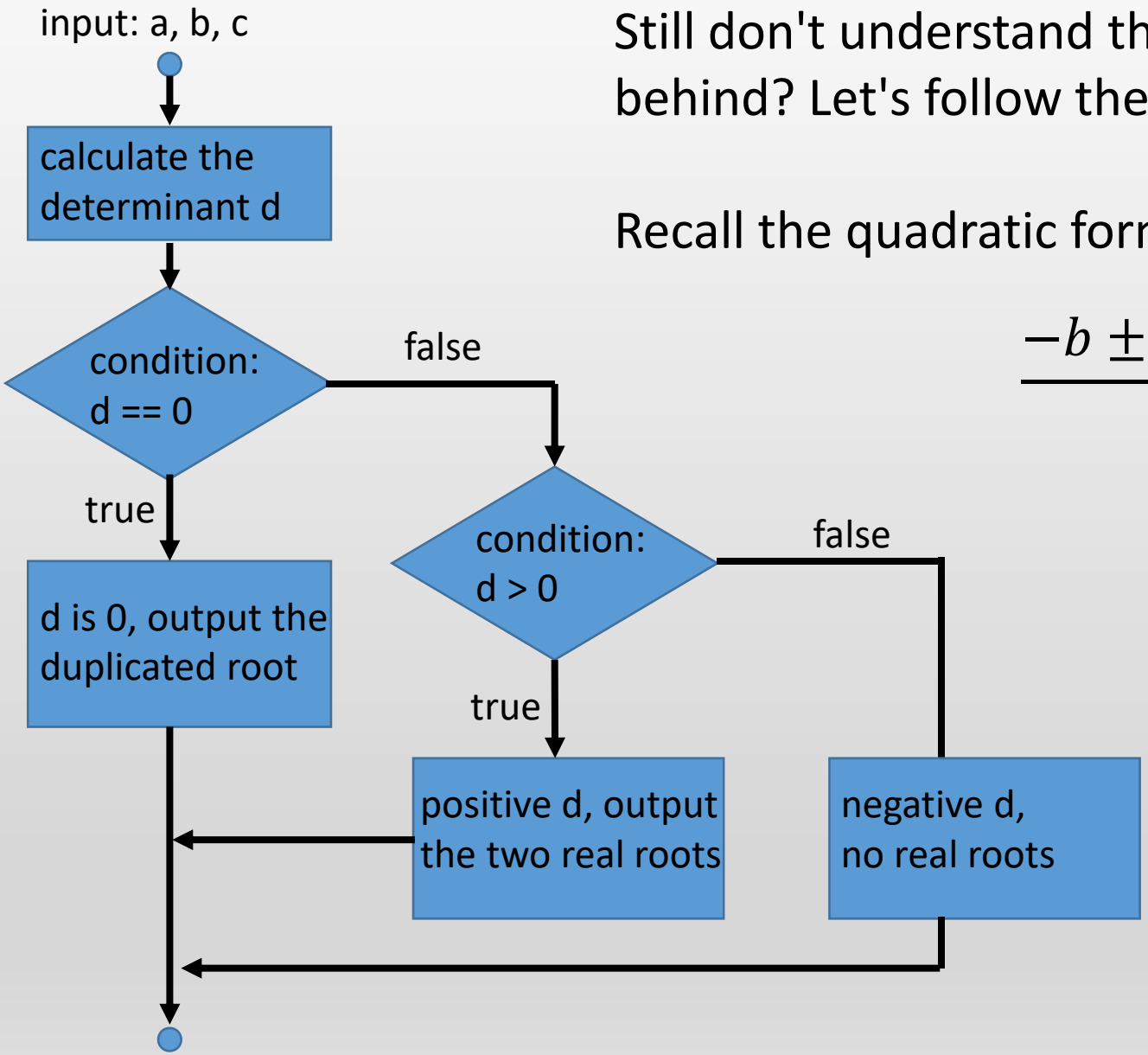
Classwork exercise

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)	$\sqrt{4}$	**2**	`sqrt(4)`	**2**
b)	$\sin(2\pi)$	**0**	`sin(2 * M_PI)`	**-2.44929e-016**
c)	$\ln e$	**1**	`log(M_E)`	**1**
d)	$e^{2\ln 3}$	**9**	`pow(M_E,2*log(3.0))`	**9**
e)	$\lfloor -5.6 \rfloor$	**-6**	`floor(-5.6)`	**-6**
f)	$\max(2, \min(3,4))$	**3**	`max(2,min(3,4))`	**3**

Example program 3.1

```
 1 // 3.1 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:

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

Formatting console output

C++ provides additional functions for formatting how a value is displayed. These functions are called stream manipulators and are included in the `<iomanip>` header.

Function	Description	Effect
`setprecision(n)`	sets the precision of a `float` or `double` number	Permanent
`fixed`	displays a `float` or `double` numbers in nonscientific (fixed) notation. By default, the fixed number of digits after the decimal point is 6.	Permanent
`scientific`	displays floating-point numbers in scientific notation	Permanent
`showpoint`	causes a `float` or `double` numbers to be displayed with a decimal point with trailing zeros even if it has no fractional part.	Permanent
`setw(width)`	specifies the width of a print field	Temporary
`left`	justifies the output to the left	Permanent
`right`	justifies the output to the right (default)	Permanent

Example program 3.6a

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           double amount = 12618.98;
 7           double interestRate = 0.0013;
 8           double interest = amount * interestRate;
 9           cout << "Interest is " << interest << endl;
10           cout << "Interest is " << fixed << setprecision(2)
11                     << interest << endl;
12           return 0;
13 }
```

Result:

```
Interest is 16.4047
Interest is 16.40
```

The actual answer is 16.404674. By default, 6 significant figures will be shown. By using `fixed` and then `setprecision` to be 2, only 2 decimal places will be shown.

Example program 3.6b

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           double number = 12.34567;
 7           cout << setprecision(3) << number << " "
 8                     << setprecision(4) << number << " "
 9                     << setprecision(5) << number << " "
10                     << setprecision(6) << number << endl;
11           return 0;
12 }
```

Result:

```
12.3 12.35 12.346 12.3457
```

By just `setprecision` to be n,
n significant figures will be shown.

Example program 3.6c

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           cout << setprecision(6);
 7           cout << 1.23 << endl;
 8           cout << showpoint << 1.23 << endl;
 9           cout << fixed << showpoint << 1.23 << endl;
10           return 0;
11 }
```

Result:

```
1.23
1.23000
1.230000
```

We have first `setprecision` to be 6. Since the value to show is exactly 1.23, the zeros at back will not be displayed.

Using `showpoint` , 6 significant digits including zeros will be displayed.

Using `fixed` and `showpoint` , 6 decimal places including zeros will be displayed.

Example program 3.6d

```
 1 #include <iostream>
 2 #include <iomanip>
 3 using namespace std;
 4 int main()
 5 {
 6           cout << right;
 7           cout << setw(8) << 1.23 << endl;
 8           cout << setw(8) << "Hi!" << endl;
 9           cout << left;
10           cout << setw(8) << 1.23 << endl;
11           cout << setw(8) << "Hi!" << endl;
12           return 0;
13 }
```

Result:

```
    1.23
     Hi!
1.23
Hi!
```

Using `setw` we can set the display width to be 8 in case the number or string to display is less than 8 characters. Additional spacing will be displayed.

We can use either `right` or `left` to set the alignment.

Classwork exercise

Show the output of the following, use ☐ for a space.

```
double monthlyPayment = 1345.4567;
double totalPayment = 866.887234;
cout << setprecision(7);
cout << monthlyPayment << endl;
cout << totalPayment << endl;
cout << fixed << setprecision(2);
cout << setw(8) << monthlyPayment << endl;
cout << setw(8) << totalPayment << endl;
```

Answer to classwork exercise

Show the output of the following, use ☐ for a space.

`1345.457`	(7 sig. fig.)
`866.8872`	(7 sig. fig.)
☐`1345.46`	(2d.p., width = 8)
☐☐`866.89`	(2d.p., width = 8)

Character

Character is a data type denoted by char. A character is a single letter enclosed in single quotation marks. For example:

```
char firstletter = 'A';
char secondletter = '4';
```

Please be careful that the character 'A' is different from some variable A, also the character '4' is different from the number 4. Notice that character is case sensitive.

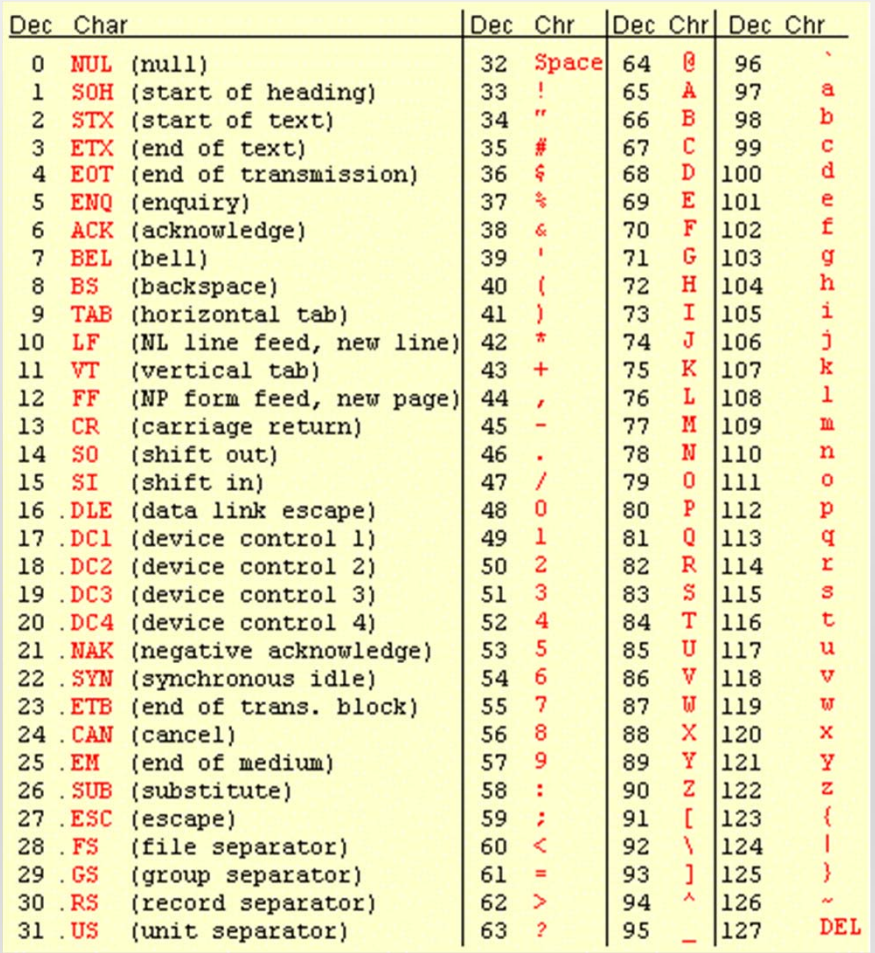
Besides digits, upper and lower letters, special characters, there are also some escape sequences,

Escape sequence	Name	Escape sequence	Name
\b	backspace	\r	carriage return
\t	tab	\\	backslash
\n	linefeed	\"	double quote
\f	formfeed		

ASCII Code

Each character has been assigned to a number called ASCII code (American Standard Code for Information Interchange)

Characters	ASCII Code
'0' to '9'	48 to 57
'A' to 'Z'	65 to 90
'a' to 'z'	97 to 122
\t	9
\"	34
\\	92



Dec	Char	Dec	Chr	Dec	Chr	Dec	Chr
0	NUL (null)	32	Space	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Some special symbols are listed on extended ASCII code table.

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Input and Output

We can use **`cout`** and **`cin`** as usual.

Character operation

With ASCII code, characters can be computed as numbers.
Once a character variable is put into a mathematical operation, its ASCII code will be accounted. The result is a number.

`(int)`: return the ASCII code of a character
`(char)`: return the corresponding character of an ASCII code

Classwork exercise

Determine the output of the following code:

```
char letter = 'A';

cout << letter << endl;                 A

cout << (int)letter << endl;            65

cout << letter + 1 << endl;             66

cout << (char)(letter + 1) << endl;     B
```

Example program 3.2

```
1  // 3.2 Character
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6           char letter;
7           int number;
8           cout << "Enter a letter from A to Z " << endl;
9           cin >> letter;
10          cout << "Enter n integer " << endl;
11          cin >> number;
12          if (letter >= 65 && letter <= 90)
13                    cout << "The letter in next " << number << " space is "
14                    << (char)((letter + number - 65) % 26 + 65);
15          else
16                    cout << "Your input is not from A to Z." << endl;
17          return 0;
18 }
```

Example program 3.2

```
1  // 3.2 Character
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          char letter;
7          int number;
8          cout << "Enter a letter from A to Z " << endl;
9          cin >> letter;
10         cout << "Enter n integer " << endl;
11         cin >> number;
12         if (letter >= 65 && letter <= 90)
13                 cout << "The letter in next " << number << " space is "
14                 << (char)((letter + number - 65) % 26 + 65);
15         else
16                 cout << "Your input is not from A to Z." << endl;
17         return 0;
18 }
```

Input a letter and a number from keyboard

Check the letter is within A to Z

Suppose A to Z is a cycle, since there are 26 letters, we assign them to 0 to 25 (mod 26). We need to subtract 65 which is the ASCII of the starting letter A, and then add it back to output the corresponding character.

Character function

C++ provides several functions for testing a character and for converting a character in the `<cctype>` header file:

Function	Description
`islower(ch)`	Returns `true` if the specified character is a lowercase letter.
`isupper(ch)`	Returns `true` if the specified character is an uppercase letter.
`isspace(ch)`	Returns `true` if the specified character is a whitespace character.
`tolower(ch)`	Returns the lowercase of the specified character.
`toupper(ch)`	Returns the uppercase of the specified character.

Example program 3.3

```
 1 #include <iostream>
 2 #include <cctype>
 3 using namespace std;
 4 int main()
 5 {
 6           cout << "Enter a character: ";
 7           char ch;
 8           cin >> ch;
 9           cout << "You entered " << ch << endl;
10           if (islower(ch))
11           {
12                     cout << "It is a lowercase letter " << endl;
13                     cout << "Its equivalent uppercase letter is " <<
14                               static_cast<char>(toupper(ch)) << endl;
15           }
16           else if (isupper(ch))
17           {
18                     cout << "It is an uppercase letter " << endl;
19                     cout << "Its equivalent lowercase letter is " <<
20                               static_cast<char>(tolower(ch)) << endl;
21           }
22           else if (isdigit(ch))
23           {
24                     cout << "It is a digit character " << endl;
25           }
26           return 0;
27 }
```

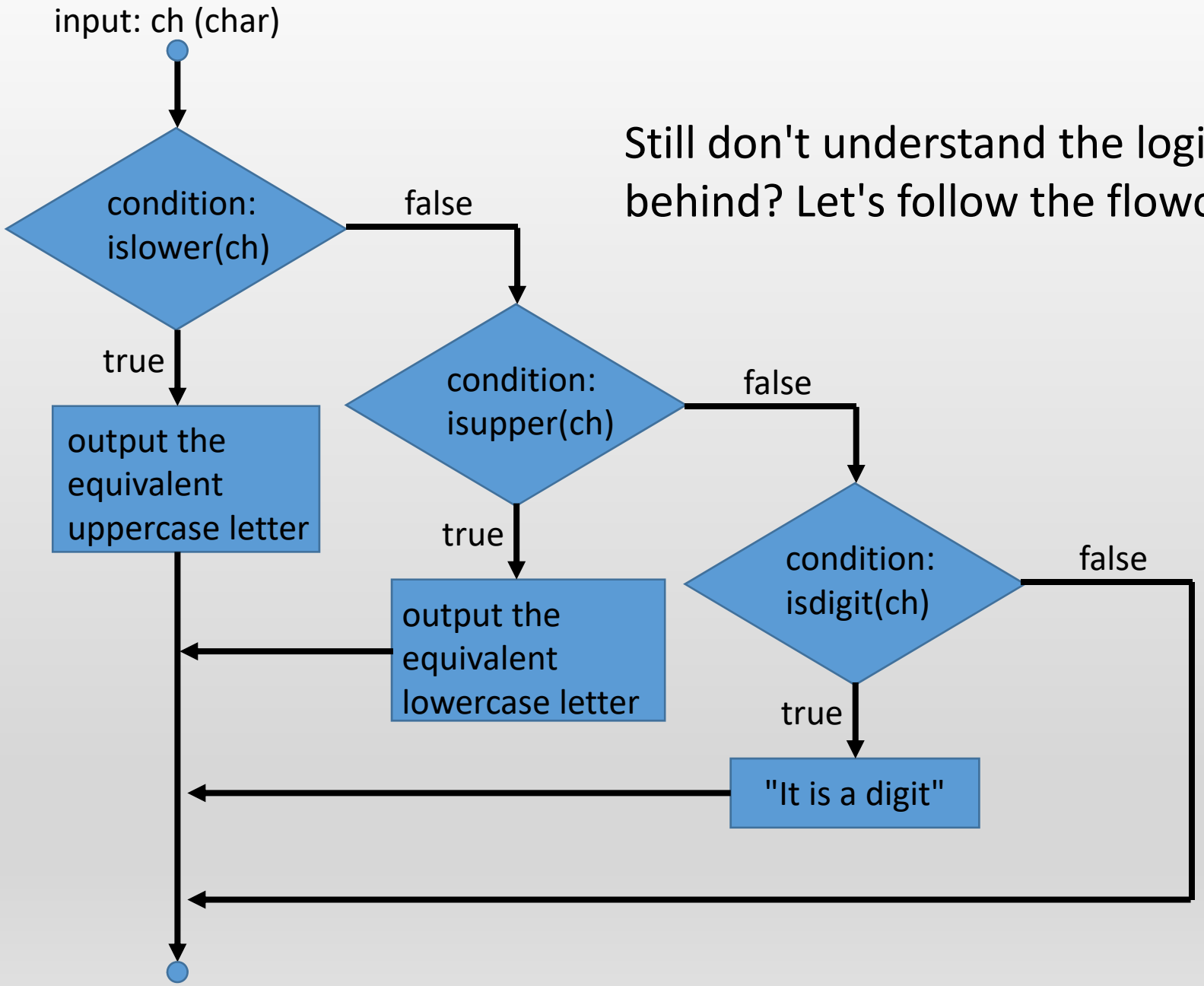
Example program 3.3

```
1  #include <iostream>
2  #include <cctype>
3  using namespace std;
4  int main()
5  {
6           cout << "Enter a character: ";
7           char ch;
8           cin >> ch;
9           cout << "You entered " << ch << endl;
10          if (islower(ch))
11          {
12                    cout << "It is a lowercase letter " << endl;
13                    cout << "Its equivalent uppercase letter is " <<
14                              (char)(toupper(ch)) << endl;
15          }
16          else if (isupper(ch))
17          {
18                    cout << "It is an uppercase letter " << endl;
19                    cout << "Its equivalent lowercase letter is " <<
20                              (char)(tolower(ch)) << endl;
21          }
22          else if (isdigit(ch))
23          {
24                    cout << "It is a digit character " << endl;
25          }
26          return 0;
27 }
```

Check if ch is lowercase, then return corresponding uppercase letter.

Check if ch is uppercase, then return corresponding lowercase letter.

Check if ch is a digit.



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

String

The `char` type represents only one character. To represent a string of characters, we can use the data type called `string`. However, `string` is not a primitive type, it is a predefined class in the `<string>` header file.

Double quotation mark is used to enclose the content of a string. For example:

```
string message = "AMA2222 is fun!";

string name;
cout << "What is your name?" << endl;
cin >> name;
```

String function

Two functions below for string objects are very common to use. However, the syntax is different from Mathematical functions that we have learnt before.

Function	Description
`length()`	Returns the number of characters in this string.
`at(index)`	Returns the character at the specified index from this string.

For example:

```
string message = "AMA2222 is fun!";
cout << "The length is " << message.length() << endl;
cout << "The third character is " << message.at(2);
```

15

A

Notice that the index of a string counts from 0. In the previous example, a string with length 15 have its characters indexed from 0 to 14. Also notice that a spacing is also counted as a character.

Using the `at()` function, `message.at(2)` gives the third letter "A" of the string in our common sense. But a computer scientist might address that as the "2nd character".

There is a simpler way of doing so. By putting the index in a square bracket after the string name, `message[2]` gives the same result as `message.at(2)` for convenience.

Two or more strings can be concatenated together in an order simply by using + the plus sign. The result is also a string.

To update an existing string variable by concatenating another string, use the += operator.

Classwork exercise: evaluate the output

```
string s1 = "pine";
string s2 = "apple";
s1 += s2;
string s3 = s1 + "pen";
cout << s1 << endl;
cout << s2 << endl;
cout << s3 << endl;
```

pineapple
apple
pineapplepen

You can use the relational operators ==, !=, <, <=, >, >= to compare two strings. This is done by comparing their corresponding characters one by one from left to right.

A string can be read from the keyboard using the cin object. C++ provides the getline function in the string header file, which reads a string from the keyboard using the following syntax:

```
getline(cin, s, delimitCharacter);
```

The function stops reading characters when the delimit character is encountered. It is read but not stored into the string. By default, '\n' (new line) is used as delimit character. To clear the input, you can use the function

```
fflush(stdin);
```

Example program 3.4

```
 1 #include <iostream>
 2 #include <string>
 3 using namespace std;
 4 int main()
 5 {
 6           string surname;
 7           cout << "Enter the Chinese name of the man: ";
 8           getline(cin, surname, ' ');
 9           fflush(stdin);
10
11           string wifename;
12           cout << "Enter the Chinese name of the woman: ";
13           getline(cin, wifename, '\n');
14
15           string newname = surname + " " + wifename;
16           cout << "After marriage, the woman will be called " << newname;
17           return 0;
18 }
```

Example program 3.4

```
 1 #include <iostream>
 2 #include <string>
 3 using namespace std;
 4 int main()
 5 {
 6         string surname;
 7         cout << "Enter the Chinese name of the man: ";
 8         getline(cin, surname, ' ');
 9         fflush(stdin);
10
11         string wifename;
12         cout << "Enter the Chinese name of the woman: ";
13         getline(cin, wifename, '\n');
14
15         string newname = surname + " " + wifename;
16         cout << "After marriage, the woman will be called " << newname;
17         return 0;
18 }
```

Ask user to enter the man's name but only read before the spacebar, keeping only his surname

Read the full name of the woman

Form a new name by concatenating the man's surname and the woman's full name

Example program 3.5

A lottery game generates a random two-digit number (each digit can be 0 to 9), prompts the user to enter a two-digit number, and determines whether the user wins according to the following:

- Match the lottery number in exact order, win $10,000

- Match both digits but in wrong order, win $3,000

- Match only one digit, win $1,000

```cpp
#include <iostream>
#include <string> // for using strings
#include <ctime> // for time function
#include <cstdlib> // for rand and srand functions
using namespace std;
int main()
{
        string lottery;
        srand(time(0));
        int digit = rand() % 10;
        lottery += static_cast<char>(digit + '0');
        digit = rand() % 10;
        lottery += static_cast<char>(digit + '0');
        cout << "Enter your lottery pick (two digits): ";
        string guess;
        cin >> guess;
        cout << "The lottery number is " << lottery << endl;
        if (guess == lottery)
                cout << "Exact match: you win $10,000" << endl;
        else if (guess[1] == lottery[0] && guess[0] == lottery[1])
                cout << "Match all digits: you win $3,000" << endl;
        else if (guess[0] == lottery[0] || guess[0] == lottery[1]
                || guess[1] == lottery[0] || guess[1] == lottery[1])
                cout << "Match one digit: you win $1,000" << endl;
        else
                cout << "Sorry, no match" << endl;
        return 0;
}
```

```
 8          string lottery;
 9          srand(time(0));
10          int digit = rand() % 10;
11          lottery += static_cast<char>(digit + '0');
12          digit = rand() % 10;
13          lottery += static_cast<char>(digit + '0');
```

Generates two random digits and concatenate them to a string named lottery

```
14          cout << "Enter your lottery pick (two digits): ";
15          string guess;
16          cin >> guess;
17          cout << "The lottery number is " << lottery << endl;
```

Prompt the user to enter a guess, then output the lottery number.

```
18          if (guess == lottery)
19                  cout << "Exact match: you win $10,000" << endl;
```

Case 1: the two strings are exactly the same

```
20          else if (guess[1] == lottery[0] && guess[0] == lottery[1])
21                  cout << "Match all digits: you win $3,000" << endl;
```

Case 2: the two strings are different, but they have the same characters with opposite orders

```
22          else if (guess[0] == lottery[0] || guess[0] == lottery[1]
23                  || guess[1] == lottery[0] || guess[1] == lottery[1])
24                  cout << "Match one digit: you win $1,000" << endl;
```

Case 3: both Case 1 and Case 2 fails, but either one character matches

```
25          else
26                  cout << "Sorry, no match" << endl;
```

Case 4: all of Case 1, Case 2 or Case 3 fail