AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Chapter 5: Data Structure I - string
characters and strings, string functions,
string processing

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

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.

If you want to interchange between the character and its integer value according to ASCII, you may use the following:

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

Classwork exercise 5.1

Determine the output of the following code:

```
char letter = 'A';

cout << letter << endl;

cout << (int)letter << endl;

cout << letter + 1 << endl;

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

Classwork exercise 5.1

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 5.1

```
 1 // 5.1 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 an integer " << endl;
11           cin >> number;
12           if (letter >= 65 && letter <= 90)
13                     cout << "The letter in next " << number << " position 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 5.1

```
 1 // 5.1 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 << " position 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 want to assign them into 0 to 25 (mod 26). We need to subtract by 65 which is the ASCII of the starting letter 'A', and then add 65 back to the value and output the corresponding character.

Character functions

C++ provides several functions for testing and converting a character in the `<cctype>` library.

Suppose ch is a character variable:

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.
`isalpha(ch)`	Returns `true` if the specified character is an alphabet.
`isspace(ch)`	Returns `true` if the specified character is a whitespace character.
`isdigit(ch)`	Returns `true` if the specified character is a digit.
`tolower(ch)`	Returns the lowercase (ASCII value) of the specified character.
`toupper(ch)`	Returns the uppercase (ASCII value) of the specified character.

Notice that the conversion functions `tolower` and `toupper` will not change the character value stored in `ch`. They simply return the corresponding ASCII value which is an integer.

Example program 5.2

```
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 }
```

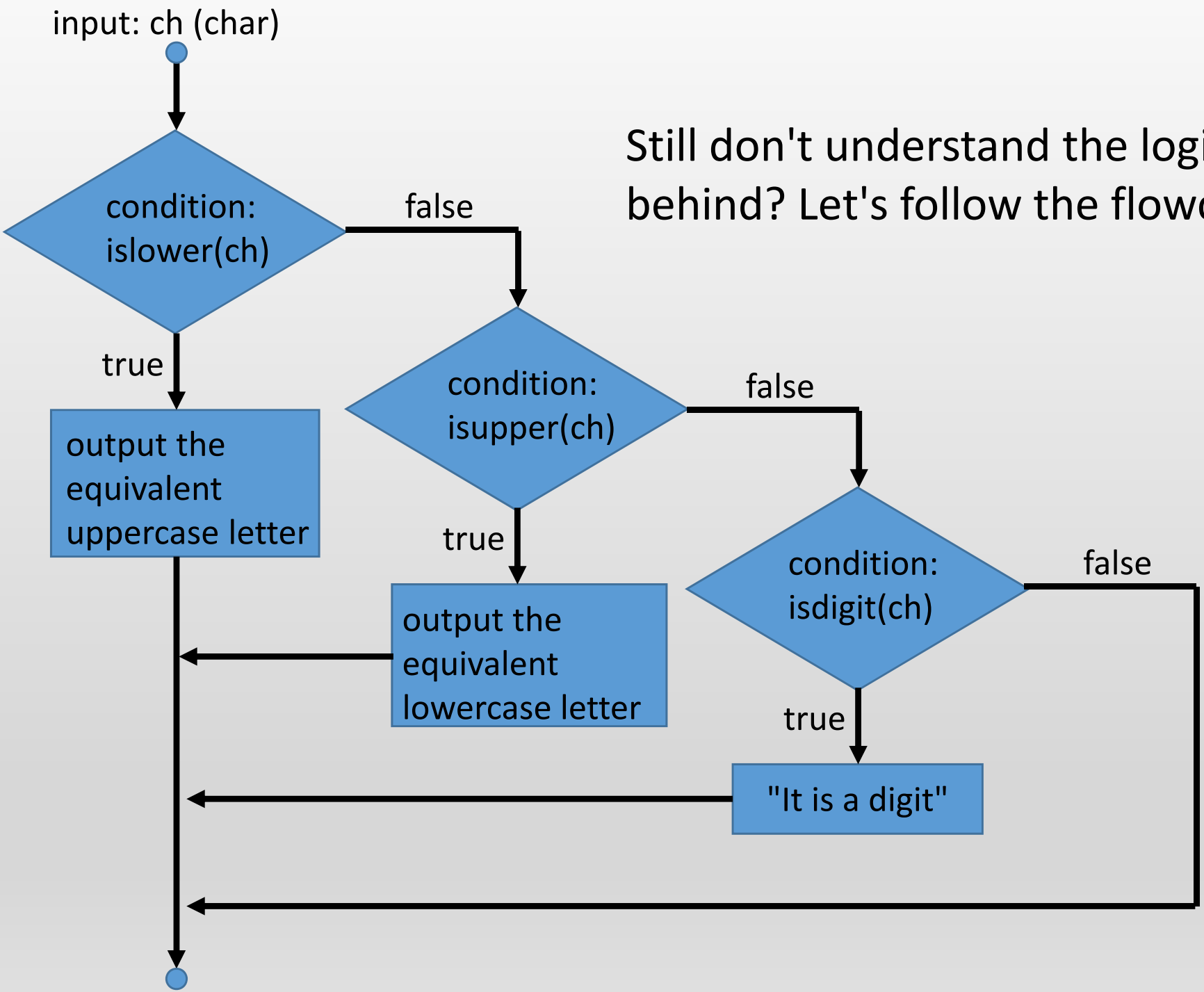
Example program 5.2

```
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>` library.

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)`** by considering it as an array.

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 5.2
Evaluate the output of the following:

```
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.

Example 5.3 Compare the order of two words in dictionary.

```
 1 #include <iostream>
 2 #include <string>
 3 using namespace std;
 4 int main()
 5 {
 6          string s1, s2;
 7          cout << "Enter the first word: ";
 8          cin >> s1;
 9          cout << "Enter the second word: ";
10          cin >> s2;
11
12          if (s1==s2)
13                    cout << "They are the same word.";
14          else if (s1<s2)
15                    cout << s1 << " is in front of " << s2;
16          else      cout << s1 << " is behind " << s2;
17          return 0;
18 }
```

A string can be read from the keyboard using cin. However, cin only allows reading the value before a space or new line.

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 the delimit character.

Example program 5.4a

```
 1 #include <iostream>
 2 #include <string>
 3 using namespace std;
 4 int main()
 5 {
 6 string name;
 7 cout << "Please enter your name: ";
 8 cin >> name;
 9 cout << "Hello " << name;
10 return 0
11 }
```

Read up to spacebar

Example program 5.4b

```
 1 #include <iostream>
 2 #include <string>
 3 using namespace std;
 4 int main()
 5 {
 6 string name;
 7 cout << "Please enter your name: ";
 8 getline(cin, name);
 9 cout << "Hello " << name;
10 return 0
11 }
```

Read up to newline

If you type "Chan Tai Man" as the input. What would be the output of the two programs?

```
Please enter your name: Chan Tai Man
Hello Chan
--------------------------------
Process exited after 3.736 seconds with return value 0
Press any key to continue . . .
```

```
Please enter your name: Chan Tai Man
Hello Chan Tai Man
--------------------------------
Process exited after 3.346 seconds with return value 0
Press any key to continue . . .
```

More string functions

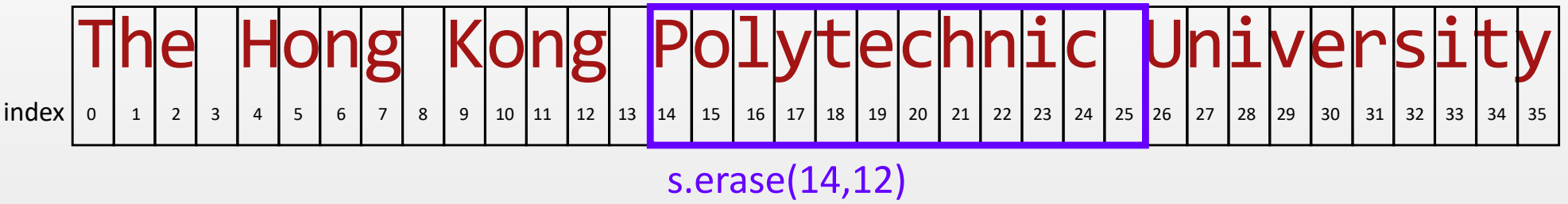
Two functions below for string objects are very common to use. However, the syntax is different from Mathematical functions that we have learnt before.

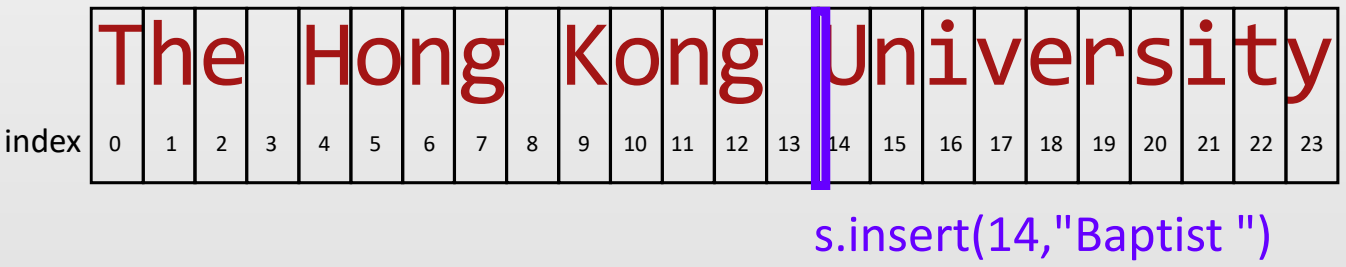
Function	Description
`erase(index, length)`	Delete part of the string at the index for a certain length.
`insert(index,string)`	Insert another string to an existing string at the index.
`substr(index,length)`	Read part of the string at the index for a certain length.

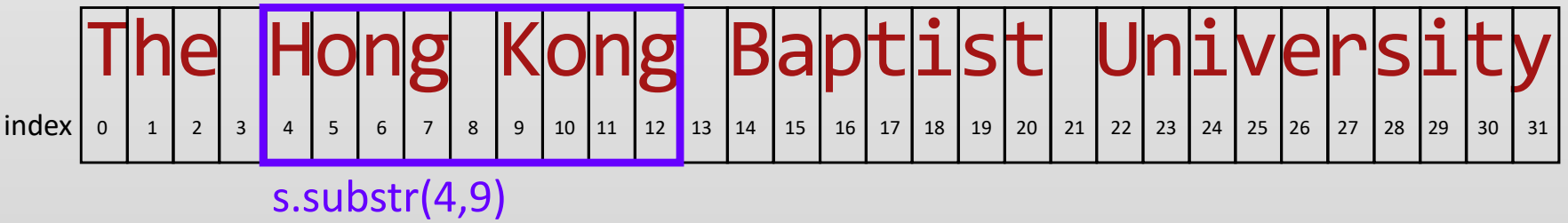
Notice that `s.erase()` and `s.insert()` will affect the string stored in `s`. However, `s.substr()` will just return another string without changing the string stored in s.

Example program 5.5

```
 1 #include <iostream>
 2 #include <string> // for using strings
 3 using namespace std;
 4 int main()
 5 {
 6           string s = "The Hong Kong Polytechnic University";
 7           s.erase(14,12);
 8           cout << s << endl;
 9           s.insert(14, "Baptist ");
10           cout << s << endl;
11           string t = s.substr(4,9);
12           cout << t << endl;
13           return 0;
14 }
```


The Hong Kong Polytechnic University
index
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
s.erase(14,12)


The Hong Kong University
index
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
s.insert(14,"Baptist ")


The Hong Kong Baptist University
index
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
s.substr(4,9)


Hong Kong
index
0 1 2 3 4 5 6 7 8

Using for loop in string processing

Recall two string functions:

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

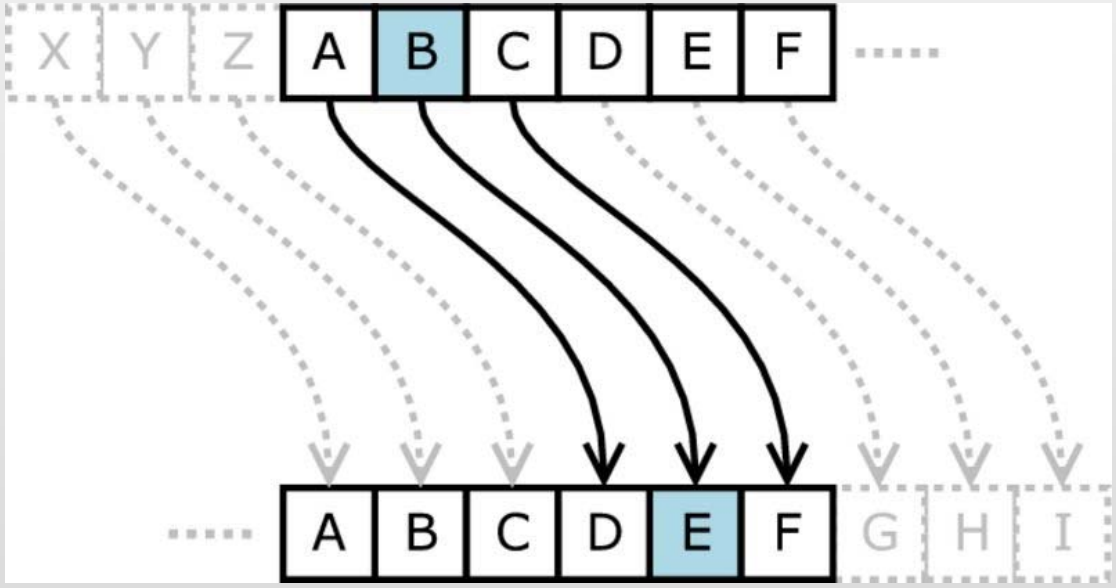
For simplicity, `s.at(i)` can be written as `s[i]`

For a string s, we can use the syntax:

```
for (int i = 0; i < s.length(); i++)
   {
      …… s[i] ……
   }
```

To process the string by each of its character.

One famous example of string processing is **Caesar Cipher**.
It is named after Julian Caesar. It is believed that Caesar used this encoding method to send military messages to his generals.



Example program 5.6 Caesar Cipher

Write a program that reads a message as string, input a key, and output the encoded message using Caesar Cipher with the key.

Assume all characters are upper-case.

Sample output:

```
Enter a message: PROGRAMMING IS FUN
Enter a key: 5
The encoded message: UWTLWFRRNSL NX KZS
```

```cpp
// 5.6 Caesar Cipher
#include <iostream>
#include <string>
using namespace std;
int main()
{
	string msg;
	cout << "Enter a message: ";
	getline (cin,msg);
	int key;
	cout << "Enter a key: ";
	cin >> key;
	for (int i=0; i<msg.length(); i++)
		{
			if (isupper(msg[i]))
				msg[i] = (msg[i] - 65 + key) % 26 + 65;
		}
	cout << "The encoded message is: " << msg;
	return 0;
}
```

Another typical string processing problem is searching, which means finding a particular word as a substring within a string. This can be achieved either by using a for-loop, or directly use the function `find` in the `<string>` library.

Within a string s1, we can find the starting position where string s2 first appear by

```
s1.find(s2)
```

Oppositely, we can find the starting position where string s2 last appear within s1 by

```
s1.rfind(s2)
```

Notice that both `find` are `rfind` functions under a string. Therefore they are called using the dot operator.

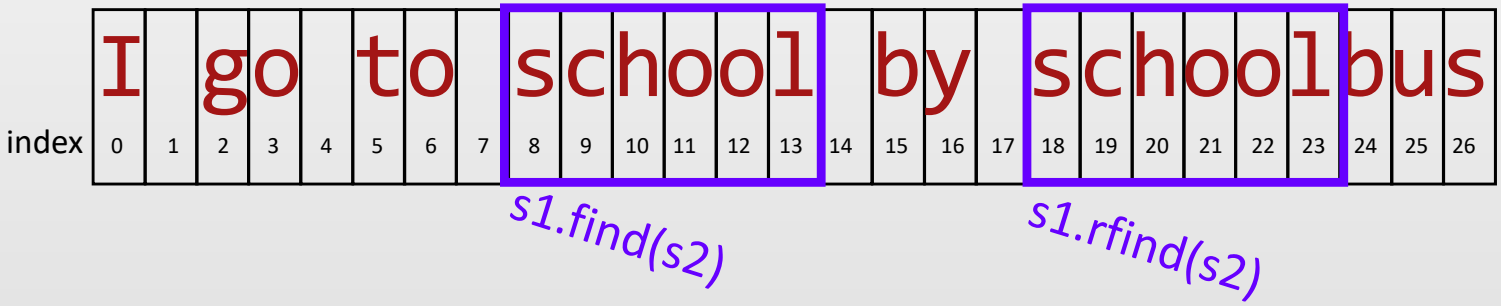
In addition, we can use `find` to search for s2 after a specified index pos within s1.

```
s1.find(s2, pos)
```

Similarly, we can use `rfind` to search for s2 before a specified index pos within s1.

```
s1.rfind(s2, pos)
```

Example 5.7 Write a program that prompts the user to enter a sentence and a word. Check if the word appears in the sentence. If yes, give the index where it first and last appears.




```
Enter a sentence: I go to school by schoolbus
Enter a word: taxi
The word does not appear
```

```
Enter a sentence: I go to school by schoolbus
Enter a word: school
The word first appears at index 8
The word last appears at index 18
```

```
 1 // 5.7a wording searching
 2 #include <iostream>
 3 #include <string>
 4 using namespace std;
 5 int main()
 6 {
 7           string s1, s2;
 8           cout << "Enter a sentence: ";
 9           getline (cin,s1);
10           cout << "Enter a word: ";
11           cin >> s2;
12           int p1 = s1.find(s2);
13           int p2 = s1.rfind(s2);
14           if (p1<0 || p1>=s1.length())
15                     cout << "The word does not appear" << endl;
16           else {
17                     cout << "The word first appears at index " << p1 << endl;
18                     cout << "The word last appears at index " << p2 << endl;}
19           return 0;
20 }
```

Alternately, we may also use a for-loop to do the searching.

```cpp
// 5.7b wording searching
#include <iostream>
#include <string>
using namespace std;
int main()
{
        string s1, s2;
        cout << "Enter a sentence: ";
        getline (cin,s1);
        cout << "Enter a word: ";
        cin >> s2;
        int p1=-1, p2=-1;
        for (int i=0; i<s1.length(); i++)
                if (s1.substr(i,s2.length())==s2){
                        p1=i;
                        break;}
        for (int i=s1.length()-s2.length(); i>=0; i--)
                if (s1.substr(i,s2.length())==s2){
                        p2=i;
                        break;}
        if (p1==-1)
                cout << "The word does not appear" << endl;
        else {
                cout << "The word first appears at index " << p1 << endl;
                cout << "The word last appears at index " << p2 << endl;}
        return 0;
}
```