Strings and Ciphers

Topics: strings, char, ciphers, loops
Lab 6: https://maryash.github.io/135/labs/lab_06.html

- Sadab Hafiz

More about `char` type

For historical reasons, type `char` is a numeric type, just like `int`. Each character in C++ is represented by its ascii code. Therefore:

‘A’ = 65 | ‘B’ = 66 | ‘C’ = 67 | ‘D’ = 68 | ‘E’ = 69 and so on...

In order to see the full ascii table, use this terminal command: `man ascii`

Since characters are numeric types, we can do mathematical operations:

```
cout << (‘A’ + 25 == ‘Z’) << endl;          // Prints 1 (TRUE)

cout << ‘A’ + 25 << endl;                  // Prints 90

char var = ‘a’;                            // var = ‘a’

var++;                                     // ‘a’ += 1

cout << var << endl;                       // Prints ‘b’
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Useful character functions and operations

If you want to convert a character to an ascii, you can do it using (int):

```
cout << (int)‘a’ << endl;                    // Prints 97
```

If you want to convert an ascii to a `char`, you can do it using (char):

```
cout << (char)97 << endl;                    // Prints ‘a’
```

These useful functions are from the <cctype> library:

```
isalpha(c)      // Checks if char c is alphabetic, returns true if it is and false otherwise

isdigit(c)      // Checks if char c is decimal digit, returns true if it is and false otherwise

isspace(c)      // Checks if char c is a white-space, return true if it is and false otherwise
```

White-space characters: Space(‘ ’), Horizontal-tab(‘\t’), Newline(‘\n’), Vertical-tab(‘\v’), Feed(‘\f’), Carriage return(‘\r’)

Helpful `string` functions

A `string` is a C++ type for representing sequences of characters. In order to get the size of the string, use `length()` or `size()` function. Each index of the string can be accessed using `[ ]`. String indexes start at 0 similar to array indexes. Thus, similar to arrays, iterate a string like this:

```
string var = “RYAN”;

for (int i = 0; i < var.size(); i++) {

     cout << “Index: ” << i << “   Char: ” << var[i] << endl;

}

/*Output:       Index: 0   Char: R

               Index: 1   Char: Y

               Index: 2   Char: A

               Index: 3   Char: N              */
```

`cin` vs `getline( )

As stated in earlier classes, the input stream operator(>>) only gets the input until it runs into a white-space character. In contrast, getline is able to get the entire line.

```
string var = “”;

cin >> var;

cout << var;
```

vs

```
string var = “”;

getline(cin, var);

cout << var;
```

Input: Who is Mark?

```
Output:

Who
```

```
Output:

Who is Mark?
```

Note: Since the input operator (>>) reads a word at a time, you can combine it with a while loop to read an input word by word. The white-space characters will be lost.

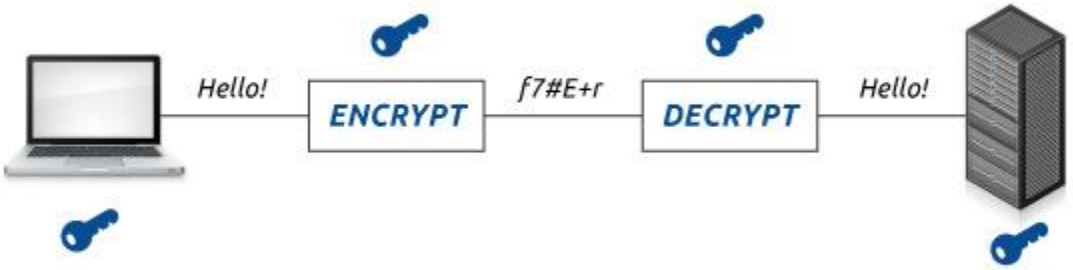
String concatenation and equality

The following example shows how to add strings together:

```
int main(){

     string a = “Genady”;                    // initialize string a

     string b = “Maryash”;                   // initialize string b

     a += “ ”;                               // add a space to a (a = “Genady” + “ ” = “Genady ”)

     string full_name = a + b;               // full_name = “Genady ” + “Maryash”

     if (a != b){                            // if string a is not equal to string b

          cout << full_name << endl;         // Prints “Genady Maryash”

     }

}
```

What is Cryptography?

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.



Ciphers

Until modern times, cryptography referred almost exclusively to encryption and decryption. Find out more: Modern Cryptography

Encryption is the process of converting ordinary information (called plaintext) into unintelligible text (called ciphertext).

Decryption is the reverse process (converting ciphertext back to plaintext).

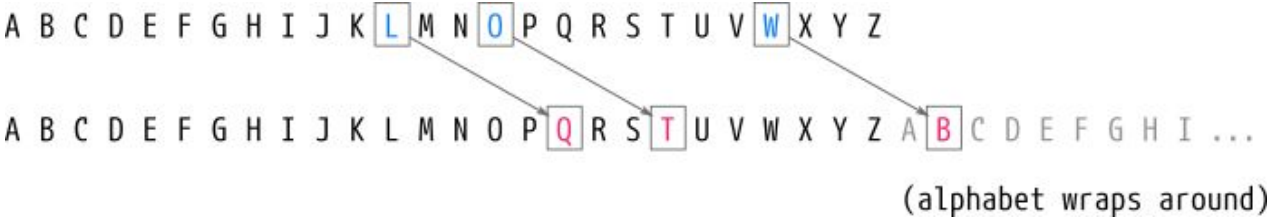
A cipher is a pair of algorithms for performing encryption and decryption. In this lab, you will implement Caesar and Vigenere ciphers.

Caesar Cipher

The plaintext is shifted to the right by a number. For example:

Enter plaintext : Hello, World!
Enter shift : 10
Ciphertext : Rovvy, Gybvn!

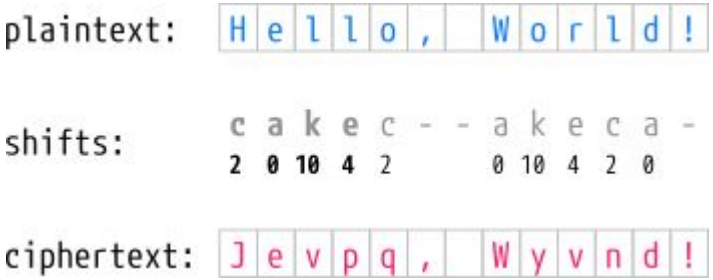
Only uppercase and lowercase letters are shifted. The letters wrap around when it reaches ‘z’ (use modulo operator!)



Decrypting would require you to shift left which would reverse the right shift. Consider writing a separate function to shift a single character to the left.

Vigenere Cipher

In Caesar Cipher, each letter is shifted by a fixed number. What if the number changed every time? Vigenere cipher takes a `keyword` and the shift number depends on the letters in that keyword. For example, the keyword here is `cake`:



Similar to Caesar cipher, the letters wrap around. The keyword is going to be lowercase. Consider using while loop to loop through the indexes of the keyword.

Decrypting would require you to shift left. Why not write the function I referred to in the last slide? You can use that for both ciphers.