2.14 Characters

Basics

A variable of ***char*** type, as in `char myChar;`, can store a single character like the letter m. A ***character literal*** is surrounded with single quotes, as in `myChar = 'm';`.

Figure 2.14.1: Simple char example: Arrow.

```
#include <iostream>
using namespace std;

int main() {
   char arrowBody;
   char arrowHead;

   arrowBody = '-';
   arrowHead = '>';

   cout << arrowBody << arrowBody << arrowBody << arrowHead << endl;

   arrowBody = 'o';

   cout << arrowBody << arrowBody << arrowBody << arrowHead << endl;

   return 0;
}
```

```
--->
ooo>
```

Feedback?

PARTICIPATION ACTIVITY 2.14.1: char data type.

1) Declare a character variable middleInitial.

char middleInitial;

Check Show answer

Correct

char middleInitial;

char is a built-in type like int.

2) Assume char variable userKey is already declared. Write a statement that assigns userKey with the letter a.

Correct

userKey = 'a';

userKey = 'a';

Check Show answer

A character literal is written surrounded by single quotes as in 'a'.

Feedback?

Getting a character from input

cin can be used to get a one character from input. Ex: `cin >> myChar;`.

Figure 2.14.2: Getting a character from input.

```
#include <iostream>
using namespace std;

int main() {
   char bodyChar;
   char headChar;

   cout << "Type two characters: ";
   cin  >> bodyChar;
   cin  >> headChar;

   // Output arrow body then head
   cout << bodyChar << bodyChar << bodyChar;
   cout << headChar << endl;

   return 0;
}
```

```
Type two characters: ->
--->
...
Type two characters:    *     /
***/
```

Feedback?

zyDE 2.14.1: char variables.

This program gets a character from input. Press Run to see how that character is changing the input character and pressing Run again.

Load default template...

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5     char   userChar;
6
7     cin  >> userChar;
8
```

```
*
```

Run

```
* *
 *
```

```
   cout << userChar << " " << userChar << end
   cout << " " << userChar << endl;
   cout << userChar << userChar << userChar <

   return 0;
}
```

```
***
```

Feedback?

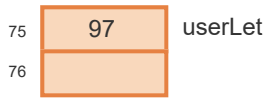
A character is internally stored as a number

Under the hood, a char variable stores a number. Ex: 'a' is stored as 97. In an output statement, the compiler outputs the number's corresponding character.

PARTICIPATION ACTIVITY 2.14.2: A char variable stores a number.

1 2 2x speed

```
char userLet;
userLet = 'a';
(output userLet)
```



a

When outputting a char variable, the compiler converts the number to the appropriate letter.

Feedback?

PARTICIPATION ACTIVITY 2.14.3: Character encodings.

Type a character: A ASCII number: **65**

Feedback?

ASCII is an early standard for encoding characters as numbers. The following table shows the ASCII encoding as a decimal number (Dec) for common printable characters (for readers who have studied binary numbers, the table shows the binary encoding also). Other characters such as control characters (e.g., a "line feed" character) or extended characters (e.g., the letter "n" with a tilde above it as used in Spanish) are not shown. Source: http://www.asciitable.com/.

Table 2.14.1: Character encodings as numbers in the ASCII standard.

Binary	Dec	Char	Binary	Dec	Char	Binary	Dec	Char
010 0000	32	space	100 0000	64	@	110 0000	96	`
010 0001	33	!	100 0001	65	A	110 0001	97	a
010 0010	34	"	100 0010	66	B	110 0010	98	b
010 0011	35	#	100 0011	67	C	110 0011	99	c
010 0100	36	$	100 0100	68	D	110 0100	100	d
010 0101	37	%	100 0101	69	E	110 0101	101	e
010 0110	38	&	100 0110	70	F	110 0110	102	f
010 0111	39	'	100 0111	71	G	110 0111	103	g
010 1000	40	(	100 1000	72	H	110 1000	104	h
010 1001	41	)	100 1001	73	I	110 1001	105	i
010 1010	42	*	100 1010	74	J	110 1010	106	j
010 1011	43	+	100 1011	75	K	110 1011	107	k
010 1100	44	,	100 1100	76	L	110 1100	108	l
010 1101	45	-	100 1101	77	M	110 1101	109	m
010 1110	46	.	100 1110	78	N	110 1110	110	n
010 1111	47	/	100 1111	79	O	110 1111	111	o
011 0000	48	0	101 0000	80	P	111 0000	112	p
011 0001	49	1	101 0001	81	Q	111 0001	113	q
011 0010	50	2	101 0010	82	R	111 0010	114	r

011 0011	51	3
011 0100	52	4
011 0101	53	5
011 0110	54	6
011 0111	55	7
011 1000	56	8
011 1001	57	9
011 1010	58	:
011 1011	59	;
011 1100	60	<
011 1101	61	=
011 1110	62	>
011 1111	63	?

101 0011	83	S
101 0100	84	T
101 0101	85	U
101 0110	86	V
101 0111	87	W
101 1000	88	X
101 1001	89	Y
101 1010	90	Z
101 1011	91	[
101 1100	92	\
101 1101	93	]
101 1110	94	^
101 1111	95	_

111 0011	115	s
111 0100	116	t
111 0101	117	u
111 0110	118	v
111 0111	119	w
111 1000	120	x
111 1001	121	y
111 1010	122	z
111 1011	123	{
111 1100	124	|
111 1101	125	}
111 1110	126	~

Feedback?

PARTICIPATION ACTIVITY 2.14.4: Character encodings.

1) 'A' is stored as _____.

- (●) 65
- ( ) 97

Correct

The ASCII table shows that uppercase 'A' is encoded as 65. In contrast, lowercase 'a' is encoded as 97.

2) '&' is stored as _____.

- (●) 38
- ( ) (no such encoding)

Correct

Encodings exist for various symbols, not just letters.

3) 7 is stored as _____.

- (●) 7
- ( ) 55

Correct

7 without quotes is an integer literal, so is stored as 7. In contrast, '7' with quotes is a character literal, and would

be stored as 55 (per the ASCII table).

4) A variable's memory location stores 88. Outputting that value as a character yields _____.

- (•) X
- ( ) x

Correct

88 is the encoding for uppercase 'X'. By itself, the 88 gives no information as to whether the number represents the integer 88 or the character 'X'. However, the variable's type and/or an output statement may specifically indicate that the value should be output as a character, in which case the 88 is output as X.

Feedback?

Escape sequences

In addition to regular characters like Z, $, or 5, character encoding includes numbers for several special characters. Ex: A newline character is encoded as 10. Because no visible character exists for a newline, the language uses an ***escape sequence***: A two-character sequence starting with \ that represents a special character. Ex: '\n' represents a newline character. Escape sequences also enable representing characters like ', ", or \. Ex: myChar = '\'' assigns myChar with a single-quote character. myChar = '\\' assigns myChar with \ (just '\' would yield a compiler error, since \' is the escape sequence for ', and then a closing ' is missing).

Table 2.14.2: Common escape sequences.

Escape sequence	Char
\n	newline
\t	tab
\'	single quote
\"	double quote
\\	backslash

Feedback?

PARTICIPATION ACTIVITY 2.14.5: Escape sequences.

1) Goal output: Say "Hello"

```
cout << _____ ;
```

- ○ "Say "Hello""
- ◉ "Say \"Hello\""
- ○ "Say \\"Hello\\""

Correct

Each \" represents a single " character. The \ is needed so the compiler doesn't treat the " before the H as the end of the string "Say ".

2) Goal output: OK bye (Assume a tab exists between OK and bye).

```
cout << _____ ;
```

- ◉ "OK\tbye"
- ○ "OK \tbye"
- ○ "OK\t bye"

Correct

Upon seeing the \, the compiler looks at the next character. Seeing the t, the compiler inserts a single tab character.

3) Given string "a\"b", the first character is stored in memory as 97 (the numeric value for 'a'). What is stored for the second character?

- ◉ 34
- ○ 92

Correct

The compiler treats \" as a single character, namely ", whose ASCII value is 34.

Feedback?

Common errors

A common error is to use double quotes rather than single quotes around a character literal, as in `myChar = "x"`, yielding a compiler error.

Similarly, a common error is to forget the quotes around a character literal, as in `myChar = x`, usually yielding a compiler error (unless x is also a declared variable, then perhaps yielding a logic error).

CHALLENGE ACTIVITY 2.14.1: Printing a message with ints and chars.

Print a message telling a user to press the letterToQuit key numPresses times to quit. End with newline. Ex: If letterToQuit = 'q' and numPresses = 2, print:

```
Press the q key 2 times to quit.
```

```
#include <iostream>
using namespace std;

int main() {
   char letterToQuit;
   int  numPresses;

   cin >> letterToQuit;
   cin >> numPresses;

   /* Your solution goes here  */
   cout << "Press the "<< letterToQuit <<" key "<< numPresses << " times to quit."<< endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with 'q' and 2.

Your output `Press the q key 2 times to quit.`

✓ Testing with 'x' and 4.

Your output `Press the x key 4 times to quit.`

Feedback?

CHALLENGE ACTIVITY 2.14.2: Outputting all combinations.

Output all combinations of character variables a, b, and c. If a = 'x', b = 'y', and c = 'z', then the output is:

`xyz xzy yxz yzx zxy zyx`

Your code will be tested in three different programs, with a, b, c assigned with 'x', 'y', 'z', then with '#', '$', '%', then with '1', '2', '3'.

```
1 include <iostream>
2 sing namespace std;
3
4 nt main() {
5   char a;
6   char b;
7   char c;
8
9   cin >> a;
10  cin >> b;
11  cin >> c;
12
13  /* Your solution goes here  */
14  cout << a << b << c << " " << a << c << b << " " << b << a << c << " " << b << c << a << "
15
16  cout << endl;
17
18  return 0;
19
```

Run ✓ All tests passed

✓ Testing with 'x', 'y', 'z'

Your output `xyz xzy yxz yzx zxy zyx`

✓ Testing with '#', '$', '%'

Your output `#$% #%$ $#% $%# %#$ %$#`

✓ Testing with '1', '2', '3'

Your output `123 132 213 231 312 321`

Feedback?