Characters

Outline

1. Character Representation in C Language

2. Reading/Writing Characters

3. Character Operations

Appendix: Built-in Character-Related Functions

1. Introduction

```
char ch;
ch = 'A';
```

- A character (a `char`-type value) is represented as an 8-bit integer that corresponds to the ASCII code of the character.

- A character constant (in a C source file) is enclosed by a single quote.
  - 'A' – This is a character (a value of type `char`).
  - "A" – This is a string with one character. (A string is a sequence of characters.)

ASCII Chart

	0	1	2	3	4	5	6	7	8	9
0	NUL							BEL	BS	HT
10	LF	VT	FF	CR						
20								ESC		
30			SP	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL		

Special characters

For example, ASCII code of **'A'** is 65

Special characters shown in the previous slide

Symbol	ASCII	Description	Escaped characters in C
NUL	**0**	**null character**	`'\0'`
BEL	7	Bell (Cause a beep)	`'\a'`
BS	8	Backspace	`'\b'`
HT	**9**	**Horizontal Tab**	`'\t'`
LF	**10**	**Line feed (New line)**	`'\n'`
VT	11	Vertical Tab	`'\v'`
FF	12	Formfeed	`'\f'`
CR	13	Carriage Return	`'\r'`
ESC	27	Escape	
SP	**32**	**Space**	`' '`
DEL	127	Delete	

1.1. **char**-type value

```
1  char ch;
2  int num;
3
4  ch = 'A';    // Assign the ASCII code of 'A' to ch
5               // Same as writing:       ch = 65;
6  num = ch;
7
8  // All three lines output "A 65"
9  printf("%c %d\n", ch, ch);
10 printf("%c %d\n", num, num);
11 printf("%c %d\n", 65, 65);
```

"%c" tells **`printf()`** to treat an integer as the ASCII code of a character and output the character.

A character can be outputted as an integer or as a character depends on "how" we format its value.

2.1. Character Processing (Input)

- Input: reading one character at a time.

Function	Description
`char input = getchar();`	`getchar()` returns one character. Note that, later in the course, we will use `int` type to store the return value of `getchar()`.
`char input;` `scanf("%c", &input);`	`scanf("%c", ...)` when the pattern `"%c"` appears, `scanf()` expects an input of size one byte (i.e., char type).

2.2. Character Processing (Output)

- Output: printing one character at a time.

Function	Description
`char input = 'a';` `putchar(input);`	**`putchar()`** expects a number in [0, 255] and prints the corresponding character out.
`char input = 'a';` `printf("%c", input);`	**`printf("%c", ...)`** when the pattern **`"%c"`** appears, **`printf()`** expects a number in the range [0, 255].

3. Character Operations

- Since characters are integers, we can apply
  - arithmetic operations (+, -, *, /, %), and
  - relational operations (>, >=, ==, !=, etc.)

  on characters.

3.1. Checking if a character is a digit

Character	0	1	2	3	4	5	6	7	8	9
ASCII Code	48	49	50	51	52	53	54	55	56	57

- The ASCII codes for digits are in the range [48-57], and there is no non-digit characters inside that range.
  - Note: **`'0' != 0`**

```
1  char ch;
2  ch = getchar();
3  if (ch >= 48 && ch <= 57)
4      printf("A digit!\n");
5  else
6      printf("Not a digit!\n");
```

```
1  char ch;
2  ch = getchar();
3  if (ch >= '0' && ch <= '9')
4      printf("A digit!\n");
5  else
6      printf("Not a digit!\n");
```

This version is more readable.

3.2. Checking if a character is an alphabet

Character	A	B	C	D	E	...	W	X	Y	Z
ASCII Code	65	66	67	68	69	...	87	88	89	90

Character	a	b	c	d	e	...	w	x	y	z
ASCII Code	97	98	99	100	101	...	119	120	121	122

- Uppercase and lowercase alphabets reside in two different ranges but they are arranged alphabetically within their respective range.

```
1  char ch;
2  ch = getchar();
3  if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))
4      printf("An alphabet!\n");
5  else
6      printf("Not an alphabet!\n");
```

3.3. Converting uppercase to lowercase

Character	A	B	C	D	E	...	W	X	Y	Z
ASCII Code	65	66	67	68	69	...	87	88	89	90

Character	a	b	c	d	e	...	w	x	y	z
ASCII Code	97	98	99	100	101	...	119	120	121	122

A difference of 32

- The difference between the ASCII codes of a lowercase and the corresponding uppercase letter is 32 (i.e., **`'a'-'A'`**)

```
1  char ch;
2  ch = getchar();
3  if (ch >= 'A' && ch <= 'Z')
4      ch = ch - 'A' + 'a';
5
6  printf("Result = %c\n", ch);
```

```
1  char ch;
2  ch = getchar();
3  if (ch >= 'A' && ch <= 'Z')
4      ch = ch + 32;
5
6  printf("Result = %c\n", ch);
```

This version is more readable.

3.4. Additional Example

- This example expects an alphabet character input and replies with the order of the character in the alphabet. It's not perfect though. How can you improve it? (Hint: 1st, 2nd, 3rd, 4th...)

```
1  char ch;
2  ch = getchar();
3
4  if (ch >= 'a' && ch <= 'z') {
5      printf("%c is the %dth character\n",ch,ch-'a'+1);
6  } else if (ch >= 'A' && ch <= 'Z') {
7      printf("%c is the %dth character\n",ch,ch-'A'+1);
8  } else {
9      printf("Not an alphabet.\n");
10 }
11
```

Summary

- Characters representation in C Language
- Reading/writing Characters
- Character operations

Appendix: Built-in Character-Related Functions

- Remember to add `#include <ctype.h>`

Function name	Description (return 0 or 1)
`isascii()`	If an integer is in the range [0,127], return `1 (true)` Else, return `0 (false)`
`isdigit()`	If an integer is in the range [`'0'`, `'9'`], return `1 (true)` Else, return `0 (false)`
`islower()`	If an integer is in the range [`'a'`, `'z'`], return `1 (true)` Else, return `0 (false)`
`isupper()`	If an integer is in the range [`'A'`, `'Z'`], return `1 (true)` Else, return `0 (false)`
`isspace()`	If an integer represents a whitespace (space, tab, or newline) character, return `1 (true)` Else, return `0 (false)`

Appendix: Built-in Character-Related Functions

- Remember to add `#include <ctype.h>`

Function name	Description (return 0 or 1)
`isalpha()`	It is the same as: `( islower(input) || isupper(input) )`
`isalnum()`	It is the same as: `( islower(input) || isupper(input) || isdigit(input) )`
`tolower()`	If an integer is in the range [`'A'`, `'Z'`], return `lower-case representation of the input` Else, return the input value
`toupper()`	If an integer is in the range [`'a'`, `'z'`], return `upper-case representation of the input` Else, return the input value

Appendix: Built-in Character-Related Functions (Example)

```
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int main(void) {
5      char ch, input = getchar();
6
7      if ( isdigit(input) )
8          printf("A digit\n");
9      else if( isupper(input) ) {
10         ch = tolower(input);
11         printf("The lowercase of %c is %c.\n", input, ch);
12     }
13
14     return 0;
15 }
16
```

Reading Assignment

- C: How to Program, 8th ed, Deitel and Deitel
- Chapter 8 C Characters and Strings
  - Sections 8.1 – 8.3: Fundamentals of Characters and Related Library Functions