CHARACTERS AND STRINGS

CSSE 120—Rose Hulman Institute of Technology

Characters and Strings

Characters in Python

- Just a one-character ***string***

```
>>> myChar = 'C'
>>> print myChar
C
>>> print ord(myChar) # converts character to int
67
>>> print chr(67)          # converts int to character
C
```

Characters in C

- C's **char** type is really a kind of number!
- A **char** takes 1 byte of storage space
- Predict the output:

```
char myChar;
myChar = 'C';
printf("%c\n",  myChar);   /* %c is format spec. for char */
printf("%d\n", myChar);
printf("%c\n", 67);
myChar++;
printf("%c\n",  myChar);
```

Seven Ways to Say 'A'

```
int i = 'A';
printf("A");
printf("%c", 'A');
printf("%c", 'B'-1);
printf("%c", i);
putchar('A');   /* can "push" single characters to console*/
putchar(toupper('a'));  /* Need to #include <ctype.h> */
putchar(i);
```

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	Space	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	DEL

Summary: Math with Characters

```
'C' + 1 == 'D'
char b = 'b';
b--;
putchar(b);  /* outputs a */
```

- Combine these ideas to write a **for** loop that prints the characters from 'a' to 'z' on a single line
  - Try this in Eclipse; you may work with a neighbor
  - Write your answer on your quiz

Character Input

- To read a single character from the console use:
  - **getchar()**
  - Caveat: **getchar()** returns an **int**, either a **char** value or **EOF** (end of file)

Note: most operating systems only pass characters to your program after the user presses the **enter** key

```
int inChar;
int count = 0;
printf("\n\nType some text, then press 'Enter': ");
fflush(stdout);
inChar = getchar();
while (inChar != '\n') {
        count++;
        inChar = getchar();
}
printf("\nYou entered %d characters.", count);
```

Q4

Character Functions: *ctype.h*

- Conversion Functions:
  - **int tolower(int c);**
  - **int toupper(int c);**
- Test functions:
  - **isdigit(c)**
  - **isalpha(c)**
  - **islower(c)**
  - **isupper(c)**
  - **isspace(c)**

See the C *Library Reference* link on the Course Resources for more functions.

Just Stringing You Along

- A string in C is just
  - An array of characters,
  - with a '\0' at the end
- Examples:
  - **char firstName[] = "Lou"; char lastName[10];**

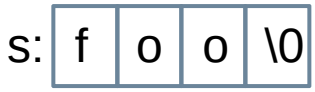
…note difference in box-and-pointer diagrams

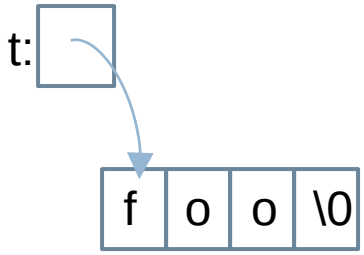
- How would we assign "Gehrig" to lastName?
  1. char lastName[] = "Gehrig"
  2. character-by-character assignment
  3. strcpy(coming soon)

String variables vs. constants

- String Variable
- char s[] = “foo”;

- String Constant
- char *t = “foo”;
- Strings declared in this way **cannot** be mutated!





String Functions: *string.h*

Function	Purpose
char *strcpy(char *dest, char* src)	copy string src to string dest, including '\0'; return dest **Note: strings are *mutable* in C, unlike Python!** **Must reserve space for dest before calling strcpy**
char *strcat(char *dest, char* src)	concatenate string src to end of dest; return dest. **Must reserve space for dest before calling strcat**
int strcmp(char *str1, char *str2)	compare string str1 to string str2, return a negative number if str1<str2, zero if str1==str2, or positive otherwise
size_t strlen(char *str)	return length of str (size_t is a typedef for int on most systems) **Doesn't include the null character**

Note: we usually ignore the return values from strcpy and strcat, since they mutate dest.

See Kochan or the C *Library Reference* link on Course Resources page for more info.

String Concatenation Using *strcat()*

- Consider:

```
char s1[] = "Go, Red! Go, White! ";
char s2[] = "Go Rose, Fight!";
/* ??? */
printf("%s\n", s3);
```

- What goes in the space? We want:
  - the output to be

    Go, Red! Go, White! Go Rose, Fight!

  - and no additional string literals

Summary: Strings in C

- Strings are arrays of characters:
  - **char firstName[] = "Lou";**

  or
  - **char lastName[10];**
    strcpy(lastName, "Gehrig");

- "Null terminated", that is, a '\0' at the end
- Strings are mutable (since arrays are pointers)
- Don't forget to reserve enough space to hold the string

When C Gives You Lemons…

- Problem:
  - Python includes high level functions for strings
  - C (and some other languages) do not
  - What if you need to use C, but also need strings?
- Solution: Make your own string functions!
- Homework:
  - Check out ***Session26CharactersAndStrings*** from SVN
  - Let's start it together.