NETB141 Computer programming labs project

Sonia Ivanova Mileva, F70487

Semester: autumn, 2013–2014

1 Problem text

Variant 3: Morse code encoder

Write a program that reads from the standard input a text in English and outputs in the standard output the same text encoded in Morse code. The program must have the following features:

1. It reads text in English from the standard input. The text is composed by Latin letters, decimal numbers and punctuation (full stop'.', question mark '?', exclamation mark '!').

2. The text is transformed character by character to Morse code, and the corresponding code is displayed in the standard output.

3. The program also represents the Morse code as sound using the PC speakers.

Technical requirements: The program must be composed by more than one function.

2 Solution overview

Our program reads the whole input, assigns it to a string variable and then for each character of it, using the ASCII representation, calls the function *morse_code*, *which* returns the Morse code of the character. Finally it calls the function *sound*, with the return value of *morse_code* as parameter, which turns the dashes and fullstops into sound.

3 Algorithm description

3.1 string morse_code(int a)

This function takes the ASCII decimal number of a character (letter, digit, full stop, question mark or exclamation mark) and prints its Morse code.

The ASCII table:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

The capital letters have numbers between 64 and 91 so if the parameter is in this range we subtract from it the number of the first in the interval (A), add 1 and we get a number from 1 to 26. We use a switch statement for these numbers and return the correspondent Morse code. Analogically we do the same for the small letters (between 96 and 123) and the digits (between 47 and 58, but here we do not add 1). There are three more cases for the exclamation mark(33), full stop(46), and question mark(66). If the parameter does not fit anywhere above it must be a wrong symbol so we return “Incorrect input”.

We use the table below for the Morse code representation:

Character	Code	Character	Code	Character	Code	Character	Code	Character	Code	Character	Code
A (info)	·–	J (info)	·–––	S (info)	···	1 (info)	·––––	Period [.]	·–·–·–	Colon [:]	––––···
B (info)	–···	K (info)	–·–	T (info)	–	2 (info)	··–––	Comma [,]	––··––	Semicolon [;]	–·–·–·
C (info)	–·–·	L (info)	·–··	U (info)	··–	3 (info)	···–––	Question mark [?]	··––··	Double dash [=]	–···–
D (info)	–··	M (info)	––	V (info)	···–	4 (info)	····–	Apostrophe [']	·––––·	Plus [+]	·–·–·
E (info)	·	N (info)	–·	W (info)	·––	5 (info)	·····	Exclamation mark [!]	–·–·––	Hyphen, Minus [-]	–····–
F (info)	··–·	O (info)	–––	X (info)	–··–	6 (info)	–····	Slash [/], Fraction bar	–··–·	Underscore [_]	··––·–
G (info)	––·	P (info)	·––·	Y (info)	–·––	7 (info)	––···	Parenthesis open [(]	–·––·	Quotation mark ["]	·–··–·
H (info)	····	Q (info)	––·–	Z (info)	––··	8 (info)	–––··	Parenthesis close [)]	–·––·–	Dollar sign [$]	···–··–
I (info)	··	R (info)	·–·	0 (info)	–––––	9 (info)	––––·	Ampersand [&], Wait	·–···	At sign [@]	·––·–· (=A+C, see below)

3.2 *void sound(string s)*

This function takes a string in Morse code and makes the sound representation.

For each character of the string we play sound: shorter for “.” and longer for “-”. That is done with the two global constant variables for sound representation *S_DOT* and *S_BAR*. If there has been an incorrect input we break the loop and exit the function with no sound played.

3.3 Functions for pauses: we use three such functions, they are all inline in order to save time because they will execute frequently and are also short ones.

- ***inline void timeSleep(clock_t sec)*** – it pauses the program for *sec* seconds with a while loop. We use *clock()* from the *ctime* library
- ***inline void shortPause()*** - it pauses the program for 1 second
- ***inline void longPause()*** - it pauses the program for 2 seconds