Προγραμματισμός Ι

Χαρακτήρες

Πανεπιστήμιο Πελοποννήσου
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Νικόλαος Δ. Τσελίκας

Χαρακτήρες - Εισαγωγή

- Έως τώρα έχουμε κατά κύριο λόγο χρησιμοποιήσει τους αριθμητικούς τύπους δεδομένων `int`, `float` και `double`
- Ο τύπος δεδομένων `char` είναι κι αυτός **αριθμητικός**
- Για τη διαχείριση των χαρακτήρων (και των αλφαριθμητικών στο επόμενο κεφάλαιο), θα θεωρήσουμε ότι το σύνολο των χαρακτήρων που υποστηρίζει ο υπολογιστής είναι κωδικοποιημένο σύμφωνα με το πιο διαδεδομένο πρότυπο, τον **κώδικα ASCII**, που **αντιστοιχίζει** κάθε χαρακτήρα σε μία **αριθμητική τιμή**
- Ο **ASCII κώδικας** αντιστοιχίζει (κωδικοποιεί) ένα σύνολο χαρακτήρων που αποτελείται από γράμματα, αριθμούς, σημεία στίξης, κτλ... με ακέραιες τιμές ανάμεσα στο `0` και το `255`
- Π.χ. η ASCII τιμή του χαρακτήρα `'C'` είναι το `67`, ενώ η ASCII τιμή του χαρακτήρα `'c'` είναι το `99`

Πίνακας ASCII - Βασικοί χαρακτήρες (0-127)

Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
(nul)	0	0x00	(sp)	32	0x20	@	64	0x40	`	96	0x60
(soh)	1	0x01	!	33	0x21	A	65	0x41	a	97	0x61
(stx)	2	0x02	"	34	0x22	B	66	0x42	b	98	0x62
(etx)	3	0x03	#	35	0x23	C	67	0x43	c	99	0x63
(eot)	4	0x04	$	36	0x24	D	68	0x44	d	100	0x64
(enq)	5	0x05	%	37	0x25	E	69	0x45	e	101	0x65
(ack)	6	0x06	&	38	0x26	F	70	0x46	f	102	0x66
(bel)	7	0x07	'	39	0x27	G	71	0x47	g	103	0x67
(bs)	8	0x08	(	40	0x28	H	72	0x48	h	104	0x68
(ht)	9	0x09	)	41	0x29	I	73	0x49	i	105	0x69
(nl)	10	0x0a	*	42	0x2a	J	74	0x4a	j	106	0x6a
(vt)	11	0x0b	+	43	0x2b	K	75	0x4b	k	107	0x6b
(np)	12	0x0c	,	44	0x2c	L	76	0x4c	l	108	0x6c
(cr)	13	0x0d	-	45	0x2d	M	77	0x4d	m	109	0x6d
(so)	14	0x0e	.	46	0x2e	N	78	0x4e	n	110	0x6e
(si)	15	0x0f	/	47	0x2f	O	79	0x4f	o	111	0x6f
(dle)	16	0x10	0	48	0x30	P	80	0x50	p	112	0x70
(dc1)	17	0x11	1	49	0x31	Q	81	0x51	q	113	0x71
(dc2)	18	0x12	2	50	0x32	R	82	0x52	r	114	0x72
(dc3)	19	0x13	3	51	0x33	S	83	0x53	s	115	0x73
(dc4)	20	0x14	4	52	0x34	T	84	0x54	t	116	0x74
(nak)	21	0x15	5	53	0x35	U	85	0x55	u	117	0x75
(syn)	22	0x16	6	54	0x36	V	86	0x56	v	118	0x76
(etb)	23	0x17	7	55	0x37	W	87	0x57	w	119	0x77
(can)	24	0x18	8	56	0x38	X	88	0x58	x	120	0x78
(em)	25	0x19	9	57	0x39	Y	89	0x59	y	121	0x79
(sub)	26	0x1a	:	58	0x3a	Z	90	0x5a	z	122	0x7a
(esc)	27	0x1b	;	59	0x3b	[	91	0x5b	{	123	0x7b
(fs)	28	0x1c	<	60	0x3c	\	92	0x5c	|	124	0x7c
(gs)	29	0x1d	=	61	0x3d	]	93	0x5d	}	125	0x7d
(rs)	30	0x1e	>	62	0x3e	^	94	0x5e	~	126	0x7e
(us)	31	0x1f	?	63	0x3f	_	95	0x5f	(del)	127	0x7f

Πίνακας ASCII - Επιπλέον χαρακτήρες (128-255)

Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
Ç	128	0x80	á	160	0xa0	└	192	0xc0	α	224	0xe0
ü	129	0x81	í	161	0xa1	┴	193	0xc1	ß	225	0xe1
é	130	0x82	ó	162	0xa2	┬	194	0xc2	Γ	226	0xe2
â	131	0x83	ú	163	0xa3	├	195	0xc3	π	227	0xe3
ä	132	0x84	ñ	164	0xa4	─	196	0xc4	Σ	228	0xe4
à	133	0x85	Ñ	165	0xa5	┼	197	0xc5	σ	229	0xe5
å	134	0x86	ª	166	0xa6	╞	198	0xc6	µ	230	0xe6
ç	135	0x87	º	167	0xa7	╟	199	0xc7	τ	231	0xe7
ê	136	0x88	¿	168	0xa8	╚	200	0xc8	Φ	232	0xe8
ë	137	0x89	⌐	169	0xa9	╔	201	0xc9	Θ	233	0xe9
è	138	0x8a	¬	170	0xaa	╩	202	0xca	Ω	234	0xea
ï	139	0x8b	½	171	0xab	╦	203	0xcb	δ	235	0xeb
î	140	0x8c	¼	172	0xac	╠	204	0xcc	∞	236	0xec
ì	141	0x8d	¡	173	0xad	═	205	0xcd	φ	237	0xed
Ä	142	0x8e	«	174	0xae	╬	206	0xce	ε	238	0xee
Å	143	0x8f	»	175	0xaf	╧	207	0xcf	∩	239	0xef
É	144	0x90	░	176	0xb0	╨	208	0xd0	≡	240	0xf0
æ	145	0x91	▒	177	0xb1	╤	209	0xd1	±	241	0xf1
Æ	146	0x92	▓	178	0xb2	╥	210	0xd2	≥	242	0xf2
ô	147	0x93	│	179	0xb3	╙	211	0xd3	≤	243	0xf3
ö	148	0x94	┤	180	0xb4	╘	212	0xd4	⌠	244	0xf4
ò	149	0x95	╡	181	0xb5	╒	213	0xd5	⌡	245	0xf5
û	150	0x96	╢	182	0xb6	╓	214	0xd6	÷	246	0xf6
ù	151	0x97	╖	183	0xb7	╫	215	0xd7	≈	247	0xf7
ÿ	152	0x98	╕	184	0xb8	╪	216	0xd8	°	248	0xf8
Ö	153	0x99	╣	185	0xb9	┘	217	0xd9	∙	249	0xf9
Ü	154	0x9a	║	186	0xba	┌	218	0xda	·	250	0xfa
¢	155	0x9b	╗	187	0xbb	█	219	0xdb	√	251	0xfb
£	156	0x9c	╝	188	0xbc	▄	220	0xdc	ⁿ	252	0xfc
¥	157	0x9d	╜	189	0xbd	▌	221	0xdd	²	253	0xfd
Pts	158	0x9e	╛	190	0xbe	▐	222	0xde	■	254	0xfe
ƒ	159	0x9f	┐	191	0xbf	▀	223	0xdf		255	0xff

Ο τύπος char (Ι)

- Για να αποθηκεύσουμε ένα χαρακτήρα σε μία μεταβλητή χρησιμοποιούμε τον τύπο `char`
- Βέβαια, επειδή ο χαρακτήρας κωδικοποιείται σαν ακέραιος, μπορούμε να χρησιμοποιήσουμε και κάποιον άλλο ακέραιο τύπο (π.χ. `int`)
- Στο επόμενο παράδειγμα δηλώνεται μία μεταβλητή τύπου `char` με το όνομα `ch` και αποθηκεύεται ο χαρακτήρας `'c'` σε αυτήν

```
char ch;
ch = 'c';
```

Όταν χρησιμοποιείται κάποιος σταθερός χαρακτήρας πρέπει να περικλείεται σε **μονές αποστρόφους** (`' '`) και **όχι** σε διπλά εισαγωγικά

Ο τύπος char (II)

- Όταν αποθηκεύεται ένας χαρακτήρας σε μία μεταβλητή τύπου **char**, στην πραγματικότητα αποθηκεύεται η ASCII τιμή του χαρακτήρα
- Δηλαδή, στο προηγούμενο παράδειγμα στη μεταβλητή **ch** αποθηκεύτηκε η τιμή **99**
- Επομένως, οι εντολές:

```
ch = 'c';
```

και

```
ch = 99;
```

είναι ισοδύναμες

- Παρατηρήστε ότι οι πιο συνηθισμένοι χαρακτήρες, όπως **γράμματα**, **ψηφία** και **σημεία στίξης** αντιστοιχίζονται σε αριθμητικές τιμές ανάμεσα στο **0** και το **127**
- Οι χαρακτήρες με τιμές από **128** έως **255** αποτελούν το εκτεταμένο ASCII σύνολο και αντιστοιχίζονται σε εξεζητημένα γράμματα και ειδικά σύμβολα
- Επειδή η μέγιστη τιμή που μπορεί να πάρει μία μεταβλητή **char** είναι το **127** (θυμηθείτε ότι το εύρος τιμών του **char** είναι **-128…127**) σε περίπτωση που θέλουμε να αποθηκεύσουμε σε μία μεταβλητή **char** ένα χαρακτήρα με ASCII τιμή μεγαλύτερη από **127**, πρέπει να χρησιμοποιήσουμε τον τύπο **unsigned char** ή **int**

Εμφάνιση Χαρακτήρα

- Για την εμφάνιση ενός χαρακτήρα στην οθόνη (μέσω της `printf()`) χρησιμοποιείται το `%c`, ενώ για την εμφάνιση της ASCII τιμής του χρησιμοποιείται αντίστοιχα το `%d`
- Στο παράδειγμα δηλώνεται μία μεταβλητή τύπου `char` με το όνομα `ch` και αποθηκεύεται ο χαρακτήρας `'a'` σε αυτήν
- Στη συνέχεια εμφανίζεται στην οθόνη ο χαρακτήρας αυτός καθώς και η αντίστοιχη ASCII τιμή του, χρησιμοποιώντας τα προσδιοριστικά μετατροπής `%c` και `%d`, αντίστοιχα

```
#include <stdio.h>
int main(void)
{
        char ch;

        ch = 'a';
        printf("Char = %c and its ASCII code is %d\n", ch, ch);
        return 0;
}
```

Ουσιαστικά, όταν ένας χαρακτήρας περιέχεται σε μία έκφραση – είτε είναι σταθερά είτε μεταβλητή – η C τον χειρίζεται σαν ακέραιο και χρησιμοποιεί την ASCII τιμή του

Παρατηρήσεις

- Ουσιαστικά, κάθε χαρακτήρας δεν είναι τίποτα άλλο παρά ένας μικρός ακέραιος αριθμός από 0 έως και 255
- Ανάλογα με το προσδιοριστικό μετατροπής που θα χρησιμοποιήσουμε (%c ή %d) εμφανίζεται ο ίδιος ο χαρακτήρας ή η ASCII τιμή του, αντίστοιχα
- Αφού η C χειρίζεται τους χαρακτήρες σαν ακέραιους, μπορούμε να χρησιμοποιήσουμε χαρακτήρες και σε αριθμητικές εκφράσεις
- Π.χ.:

```
char ch = 'c';
int i;
ch++; /* Η μεταβλητή ch γίνεται 'd'. */
ch = 68; /* Η μεταβλητή ch γίνεται 'D'. */
i = ch-3; /* Η μεταβλητή i γίνεται 'A' δηλ. 65. */
```

Παραδείγματα (Ι)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main(void)
{
        printf("Char = %c and its ASCII code = %d\n",'a'+3,'a'+3);
        return 0;
}
```

Έξοδος: `Char = d and its ASCII code = 100`

Παραδείγματα (ΙΙ)

- Γράψτε ένα πρόγραμμα το οποίο να εμφανίζει την ASCII τιμή του χαρακτήρα που αντιστοιχεί στην αλλαγή νέας γραμμής ή ισοδύναμα στο πάτημα του πλήκτρου Enter

```
#include <stdio.h>
int main(void)
{
        printf("ASCII code = %d\n",'\n');
        return 0;
}
```

Παραδείγματα (ΙΙΙ)

- Γράψτε ένα πρόγραμμα το οποίο να εμφανίζει στην οθόνη όλους τους χαρακτήρες και τις αντίστοιχες ASCII τιμές αυτών

```
#include <stdio.h>
int main(void)
{
	int i;
	for(i = 0; i < 256; i++)
		printf("Char = %c and its ASCII code = %d\n",i,i);
	return 0;
}
```

Παραδείγματα (IV)

- Παρατηρώντας προσεκτικά τον πίνακα με τις ASCII τιμές των χαρακτήρων, γράψτε ένα πρόγραμμα το οποίο να διαβάζει έναν κεφαλαίο χαρακτήρα και να εμφανίζει τον αντίστοιχο πεζό

```
#include <stdio.h>
int main(void)
{
        char ch;

        printf("Enter character: ");
        scanf("%c",&ch);

        printf("Char = %c\n",ch+32);
        return 0;
}
```

Παραδείγματα (V)

- Γράψτε ένα πρόγραμμα που να διαβάζει τρεις χαρακτήρες και να ελέγχει αν είναι συνεχόμενοι στον πίνακα ASCII ή όχι.

```
#include <stdio.h>
int main(void)
{
	char ch1, ch2, ch3;

	printf("Enter characters: ");
	scanf("%c%c%c", &ch1, &ch2, &ch3);

	if((ch1+1 == ch2) && (ch2+1 == ch3))
		printf("Consecutive\n");
	else
		printf("Not Consecutive\n");
	return 0;
}
```

Παραδείγματα (VI)

- Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει δύο χαρακτήρες και να εμφανίζει τους χαρακτήρες που βρίσκονται μεταξύ τους στο ASCII σύνολο. Για παράδειγμα, αν ο χρήστης εισάγει **af** ή **fa** το πρόγραμμα να εμφανίζει **bcde**

```
#include <stdio.h>
int main(void)
{
	char ch1, ch2;

	printf("Enter characters: ");
	scanf("%c%c", &ch1, &ch2);

	if(ch1 < ch2)
	{
		ch1++;
		while(ch1 != ch2)
		{
			printf("%c", ch1);
			ch1++;
		}
	}
	else
	{
		ch2++;
		while(ch2 != ch1)
		{
			printf("%c", ch2);
			ch2++;
		}
	}
	return 0;
}
```

Η συνάρτηση getchar() (1/3)

- Η συνάρτηση getchar() διαβάζει έναν χαρακτήρα από το stdin (το οποίο εξ' ορισμού συνδέεται με το πληκτρολόγιο)
- Το πρωτότυπό της δηλώνεται στο stdio.h, ως εξής:

```
int getchar();
```

- Η getchar() αρχίζει να διαβάζει χαρακτήρες όταν ο χρήστης πατήσει *Enter*
- Αν η getchar() εκτελεστεί επιτυχημένα, επιστρέφει τον χαρακτήρα που διαβάστηκε
- Αν δεν υπάρχει άλλος χαρακτήρας στο stdin για να διαβαστεί ή αν συμβεί κάποιο λάθος, η getchar() επιστρέφει μία ειδική σταθερά που ονομάζεται EOF (*end of file*) και έχει τιμή -1 (που είναι διαφορετική από την τιμή οποιουδήποτε char)
- Η getchar() εκτελείται πιο γρήγορα από τη scanf(), γιατί η scanf() είναι μία σύνθετη συνάρτηση που έχει σχεδιαστεί για να διαβάζει διάφορους τύπους δεδομένων και όχι μόνο χαρακτήρες

Η συνάρτηση getchar() (2/3)

- Για παράδειγμα, αν ο χρήστης πληκτρολογήσει abc και πατήσει *Enter*, η πρώτη κλήση της getchar() επιστρέφει 'a', η δεύτερη επιστρέφει 'b', η τρίτη 'c' και η τέταρτη '\n'
- Αν η getchar() κληθεί ξανά, τότε το πρόγραμμα «κολλάει» έως ότου ο χρήστης πληκτρολογήσει νέο χαρακτήρα (ή νέους χαρακτήρες) και πατήσει ξανά *Enter*
- Π.χ. το παρακάτω πρόγραμμα χρησιμοποιεί την getchar() για να εμφανίσει τους χαρακτήρες που πληκτρολόγησε ο χρήστης (και το πλήθος τους), έως ότου πατήσει *Enter*

```
#include <stdio.h>
int main(void)
{
	int ch, sum;

	printf("Enter characters: ");
	sum = 0;
	while((ch = getchar()) != '\n' && ch != EOF) /* The inner parentheses are necessary, because the != operator has greater precedence than =. */
	{
		sum++;
		printf("%c", ch);
	}
	printf("\nTotal number is = %d\n", sum);
	return 0;
}
```

Η συνάρτηση getchar() (3/3)

- Στο προηγούμενο παράδειγμα η getchar() επιστρέφει έναν προς έναν όλους τους χαρακτήρες που πληκτρολόγησε ο χρήστης μέχρι να συναντήσει τον χαρακτήρα '\n'
- Κάθε χαρακτήρας αποθηκεύεται στη μεταβλητή ch και η τιμή του συγκρίνεται με την τιμή του EOF (που είναι -1) για να εξετάσουμε αν συνέβη κάποιο λάθος στην ανάγνωση χαρακτήρα
- Παρατηρήστε ότι η επιστρεφόμενη τιμή της getchar() πρέπει να αποθηκεύεται σε μία μεταβλητή τύπου int και όχι τύπου char
- Ας δούμε το γιατί:
  - Αν θεωρήσουμε ότι (π.χ. στο προηγούμενο παράδειγμα η μεταβλητή ch ήταν δηλωμένη ως char, τότε:
  - Αν ήταν δηλωμένη ως προσημασμένη char μεταβλητή και ο χρήστης είχε εισαγάγει τον χαρακτήρα με ASCII τιμή 255, θα αποθηκευόταν ως -1 στη μεταβλητή ch (θυμηθείτε το εύρος char είναι -128…127), οπότε, η συνθήκη που ελέγχει την τιμή του χαρακτήρα έναντι της τιμής του EOF θα γινόταν ψευδής και ο βρόχος θα σταματούσε να διαβάζει άλλους χαρακτήρες
  - Αν η ch ήταν δηλωμένη ως unsigned char και η getchar() επέστρεφε EOF, το EOF δεν θα αποθηκευόταν ως -1,(αφού η μεταβλητή είναι unsigned) οπότε ο βρόχος δεν θα τερμάτιζε ποτέ (ατέρμων βρόχος)

Η συνάρτηση putchar()

- Η συνάρτηση `putchar()` γράφει έναν χαρακτήρα στο `stdout`
- Π.χ.: `putchar('a');`
- Η συνάρτηση `putchar()` επιστρέφει τον χαρακτήρα που γράφτηκε ή `EOF` για να υποδείξει την εμφάνιση κάποιου λάθους
- Δεδομένου ότι η `putchar()` είναι σχεδιασμένη μόνο για να εκτυπώνει χαρακτήρες, παρουσιάζει καλύτερη απόδοση έναντι της `printf()`
- Όπως και η `getchar()` έτσι και η `putchar()` συνήθως υλοποιείται ως μακροεντολή για αυξημένη ταχύτητα εκτέλεσης