Προγραμματισμός

Τύποι Μεταβλητών
Τελεστές
Βασική Είσοδος/Έξοδος

Μέρος Α

Τύποι δεδομένων

Τύποι Δεδομένων

- Ένας ***τύπος δεδομένων*** είναι ένα ***σύνολο τιμών*** και ένα ***σύνολο λειτουργιών*** (πράξεων) που μπορούν να εφαρμοστούν σε αυτές τις τιμές
- **Βασικοί τύποι δεδομένων**:
  - int, float, double, char
- **Σύνθετοι τύποι δεδομένων**:
  - arrays, structs, unions

Τύποι δεδομένων - int

- **Τύπος: int** (**ακέραιος** – integer)
- **Μέγεθος: 4 bytes:**
  - πεδίο τιμών: $-2^{31}-1$ … $+(2^{31}-1)$
- **Πράξεις:**
  - πρόσθεση (*+*), αφαίρεση (*-*), πολλαπλασιασμός (***), διαίρεση (*/*), υπόλοιπο (*%*)
- Παραδείγματα κυριολεκτικών τιμών:
  - 2189456 0 50 +24562 –3245 13576313

Ακέραιοι τύποι

- Μια ακέραια μεταβλητή (π.χ. τύπου **int**) δηλώνεται ως εξής:

```
int i;
```

- Η τιμή της είναι απροσδιόριστη. Αρχική τιμή (π.χ. 10) δίνεται με την εντολή:

```
int i = 10;
```

Εύρος τιμών

- Οι τιμές που μπορεί να λάβει μια ακέραια μεταβλητή καθορίζονται από την υλοποίηση. Το μέγεθος του τύπου **int** απαιτείται να είναι τουλάχιστο 16 bits, επομένως μπορεί να αναπαραστήσει αριθμούς στο διάστημα -2^{15}, 2^{15} = [-32768, 32768) τουλάχιστο.
- Στη C υπάρχουν τρία είδη ακεραίων, **short int** , **int** και **long int** , με ελάχιστα μεγέθη 8, 16, 32 bits αντίστοιχα.
- Το ακριβές μέγεθος, *σε πολλαπλάσια του μεγέθους του **char***, δίνεται από τον τελεστή **sizeof**() με όρισμα τον κάθε τύπο. Καθένας από τους τύπους ακεραίου μπορεί να ορίζεται ως **signed** (προεπιλεγμένο) ή **unsigned**.

Πραγματικοί τύποι

- Στη C ορίζονται τρεις τύποι πραγματικών αριθμών:
  - απλής ακρίβειας (**float**),
  - διπλής ακρίβειας (**double**) και
  - εκτεταμένης ακρίβειας (**long double**).

- Μία σειρά αριθμητικών ψηφίων χωρίς κενά, που περιλαμβάνει τελεία (στη θέση της υποδιαστολής) συμβολίζει πραγματική σταθερά τύπου **double** .
  - Πριν ή μετά την υποδιαστολή μπορεί να μην υπάρχουν ψηφία.
  - Ο χαρακτήρας e ή E, αν υπάρχει, ακολουθείται από τον ακέραιο εκθέτη του 10 με τη δύναμη του οποίου πολλαπλασιάζεται ο αμέσως προηγούμενος του e/E αριθμός:
  - Αν ο αριθμός τελειώνει σε F ή f είναι τύπου **float** · αν τελειώνει σε L ή l είναι τύπου **long double** .

Τύποι δεδομένων - float

- **Τύπος: float** (κινητής υποδιαστολής **απλής** ακρίβειας – floating point)
- **Μέγεθος: 4 bytes**
  - Πράξεις: πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), διαίρεση (/)
- Τιμές μιας μεταβλητής τύπου *float* μπορεί να είναι:
  - π.χ. 3.01, 110.8, -0.01, κλπ.

Τύποι δεδομένων - double

- **Τύπος: double** (κινητής υποδιαστολής διπλής ακρίβειας – double precision)
  - Ίδιος τύπος με float αλλά με μεγαλύτερη ακρίβεια
  - περιέχει δηλ. διπλό αριθμό δεκαδικών ψηφίων απ' ότι ο τύπος *float*)
- **Μέγεθος: 8 bytes**
  - Πράξεις: πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), διαίρεση (/)
- Τιμή μιας μεταβλητής τύπου *double* μπορεί να είναι:
  - π.χ 1.045623

Τύποι δεδομένων – char (1/2)

- **Τύπος: char** (χαρακτήρας – character)
  - Αναπαριστά ατομικούς χαρακτήρες
  - (A-Z, a-z, 0-9, !@$%&#, ειδικά σύμβολα \n, κλπ.)
- **Μέγεθος: 1 byte**
- Κυριολεκτικές τιμές εσωκλείονται σε μονούς αποστρόφους,π.χ.
  - ‘A’ , ‘a’ , ‘9’ , ‘”’ , ‘ ’ , ‘*’ , ‘\n’ , ‘\” , κτλ
  - *char xaraktiras;*
  - *xaraktiras = ‘A’;*

Τύποι δεδομένων – char (2/2)

- Κάθε χαρακτήρας αντιστοιχεί σ' ένα μοναδικό κωδικό ASCII
- Οι χαρακτήρες αποθηκεύονται, κυριολεκτικά, στη μνήμη ως αριθμοί.
  - Συνήθως ερμηνεύονται ως ASCII χαρακτήρες.
  - Θα δούμε στη συνέχεια όμως περιπτώσεις που μας είναι χρήσιμο να τους μεταχειριζόμαστε σαν αριθμούς.
- Χαρακτήρες αλφαβήτου, ψηφίων, ειδικοί ('\n', '\t'...)
  - – '0' ascii:48, '1' ascii:49,.., '9' ascii:57
  - – 'A' ascii:65, ..., 'Z' ascii:90
  - – 'a' ascii:97, ..., 'z' ascii:122

Τύπος χαρακτήρα

- Μια μεταβλητή τύπου χαρακτήρα (**char**) με όνομα π.χ. c, δηλώνεται ως εξής:

```
char c;
```

- Οι τιμές που μπορεί να πάρει είναι ένας χαρακτήρας από το σύνολο χαρακτήρων της υλοποίησης· αυτό είναι σχεδόν πάντοτε, αλλά όχι υποχρεωτικά, το σύνολο ASCII.
- Η δήλωση

```
char c = 'a';
```

- ορίζει μεταβλητή τύπου χαρακτήρα με όνομα c και με συγκεκριμένη αρχική τιμή, το σταθερό χαρακτήρα `a'. Παρατηρήστε ότι ο τελευταίος περικλείεται σε αποστρόφους (').

Τύπος χαρακτήρα

- Κάποιοι από τους χαρακτήρες του συστήματος χρειάζονται ειδικό συμβολισμό για να αναπαρασταθούν. Παρουσιάζονται στον Πίνακα 2.2, μαζί με τους γενικούς τρόπους προσδιορισμού (σε δεκαεξαδικό και οκταδικό σύστημα) οποιουδήποτε χαρακτήρα. Π.χ.

```
char newline = '\n';
char bell = '\a';
char alpha = '\141'; // alpha = 'a' in ASCII
char Alpha = '\x61'; // Alpha = 'a' in ASCII
```

- Χρησιμοποιείται ως **int** . Στη C προβλέπεται η μετατροπή σε **int** όλων των ακέραιων ποσοτήτων που εμφανίζονται σε μία έκφραση με τύπο "μικρότερο" από **int** προτού εκτελεστούν οι πράξεις και υπολογιστεί η τιμή της έκφρασης.

Τύπος χαρακτήρα

\'	Απόστροφος
\"	Εισαγωγικά
\\	Ανάποδη κάθετος
\a	Κουδούνι
\b	Διαγραφή χαρακτήρα
\f	Αλλαγή σελίδας
\n	Αλλαγή γραμμής
\r	Carriage return
\t	Οριζόντιο tab
\v	Κατακόρυφο tab

ASCII table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

ASCII table

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	⌂

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Παράδειγμα 1

```
/*program example2.c. This program adds three integer variables
and displays the results*/
#include <stdio.h> /*use of pre-processor*/
int main()
{
        /*declaration of variables*/
        int val1, val2, val3, sum;
        /*assign values*/
        val1=1;
        val2=2;
        val3=3;
        /*compute sum*/
        sum = val1 + val2 + val3;
        /*display sum*/
        printf("The sum of %d and %d and %d is %d\n",
                val1, val2, val3, sum);
        return 0;
}
```

Παράδειγμα 2

```
/*program example3.c. This program adds three integer
variables and displays the results*/
#include <stdio.h> /*use of pre-processor*/
main ( )
{
      int val1=1, val2=2, val3=3, sum;
      /*compute sum*/
      sum = val1 + val2 + val3;
      /*display sum*/
      printf ("The sum of %d and %d and %d is %d\n",
            val1, val2, val3, sum);
}
```

Αποτέλεσμα εκτέλεσης:

The sum of 1 and 2 and 3 is 6

Επιλογή Τύπου Δεδομένων

- Αριθμός μαθητών — int
- Βάρος, Μάζα — float, double
- Εμβαδό, Όγκος — float, double
- Όνομα — char (string)

void

- Ο τύπος **void** χρησιμοποιείται κυρίως ως τύπος του αποτελέσματος μιας συνάρτησης για να δηλώσει ότι η συγκεκριμένη συνάρτηση δεν επιστρέφει αποτέλεσμα.
- Μπορεί επίσης να χρησιμοποιηθεί ως μοναδικό όρισμα μιας συνάρτησης, υποδηλώνοντας με αυτόν τον ενναλακτικό τρόπο την κενή λίστα ορισμάτων.
- Η μόνη άλλη χρήση του είναι στον τύπο **void*** (δείκτης σε **void**) ως δείκτης σε αντικείμενο άγνωστου τύπου. Με αυτή τη μορφή χρησιμοποιείται ως τύπος ορίσματος ή επιστρεφόμενης τιμής γενικευμένης συνάρτησης.

Μετατροπές Τύπων – Αριθμητικοί Τελεστές

- Οι εκφράσεις λοιπόν έχουν: **μια τιμή και ένα τύπο**
- Τι γίνεται αν στην εφαρμογή του κανόνα
  - έκφραση1 ← έκφραση2 τελεστής έκφραση3
- οι τύποι της έκφραση2 και έκφρασης3 είναι διαφορετικοί;
- Π.χ,. x = y + z, όπου το y είναι int και το z είναι float
- **Η «μικρότερη» έκφραση μετατρέπεται στην «μεγαλύτερη»**
- Η τελική εκφραση1 έχει τον τύπο της «μεγαλύτερης» έκφρασης

Μετατροπές Τύπων

- Εκχωρήσεις τιμής σε μεταβλητή διαφορετικού τύπου
  - Το δεξί μέρος μετατρέπεται στον τύπο του αριστερού μέρους

- ***ΠΡΟΒΛΗΜΑ: η μετατροπή μπορεί να χάνει δεδομένα/ακρίβεια***

  - **int x = 15.0/2.0;**
  - char = unsigned char ή signed char, εξαρτάται από τον μεταφραστή

Παράδειγμα

- **int** a = 3.14; // a is 3
- **short int** b = 12121212121.3; // b = ??

Ρητή Μετατροπή Τύπων

- Type casting
- Ρητή μετατροπή από ένα τύπο μεταβλητής σε κάποιον άλλο.
- x = (int) y;
- γενική μορφή:
  - έκφραση1 <- (τύπος) έκφραση2
  - η έκφραση1 παίρνει την τιμή της μετατρεπόμενης τιμής της έκφρασης2

```
int sum = 2 + 3 + 5;
int N = 3;
// Wrong value
double mean1 = sum / N;
// Correct value
double mean2 = ((double) sum) / N;
```

Μέρος Β

Μεταβλητές

Μεταβλητές

- «Επώνυμες» θέσεις μνήμης
- Στη C όλες οι μεταβλητές πρέπει να δηλώνονται
- Δηλώσεις:
  - τύπος όνομα;
  - int myfirstvariable;
- Initialization: int myFirstVariable = 6;
- Δηλώσεις πολλών μεταβλητών με κόμμα
  - int x,y = 0,z = -1;

Καθολικές (global) και τοπικές μεταβλητές (local)

```
double sum;   // global

int nextOfX(int x)
{
    x = x + 1;
    sum = sum+1;
    return x;
}

main()
 {
    int x = 0;    // local
    sum = 0;
    x = nextOfX(0);
    x = nextOfX(x);
}
```

Ονόματα Μεταβλητών

- Αποτελούνται από λατινικά γράμματα (a-z,A-Z), αριθμητικά ψηφία (0-9), και underscore (_). Ονόματα που αρχίζουν με underscore είναι πιθανό να χρησιμοποιούνται από τον compiler.
  - Ξεκινούν από γράμμα ή _ και μπορούν στη συνέχεια να περιέχουν και αριθμούς (λ.χ. myVar_1_int)
- *Κεφαλαία και πεζά γράμματα είναι διαφορετικά*.
  - case sensitive (λ.χ. var1, Var1)
- Δεν επιτρέπεται η χρήση των προκαθορισμένων λέξεων της C

Μη αποδεκτά ονόματα:

ena lathos onoma, pali_latho$, 1234qwer, delete, .onoma+

Αποδεκτά ονόματα:

timi, value12, ena_onoma_me_poly_megalo_mikos, sqrt, Delete

Μεταβλητές

- Οι μεταβλητές μπορεί να είναι:
  - Τοπικές: έχει πρόσβαση σε αυτές μόνο η συνάρτηση στην οποία έχουν δηλωθεί.
    - Μόλις ολοκληρωθεί η εκτέλεση της συνάρτησης η μεταβλητή «χάνεται»
  - Καθολικές: έχουν πρόσβαση σε αυτές όλες οι συναρτήσεις που βρίσκονται στο ίδιο αρχείο
    - Διατηρούνται κατά την εκτέλεση του προγράμματος
    - Προσοχή στη χρήση τους.

Τύποι Μεταβλητών και Εκφράσεων

- Και οι μεταβλητές αλλά **και** οι εκφράσεις έχουν ένα τύπο δεδομένων
- Τύπος μεταβλητών σε τρία μέρη:
  - Μέρος πρώτο: signed, unsigned, τίποτα
  - Μέρος δεύτερο: long ή short ή τίποτα
  - Μέρος τρίτο: char, int, float, double
- Το τελευταίο είναι το μόνο απαραίτητο για να δηλώσει τύπο
- Δεν δίνουν όλοι οι συνδυασμοί συντακτικά σωστές δηλώσεις
  - int x;
  - long int x;
  - long float x;
  - unsigned char x;

Αναπαράσταση και Εύρος Τιμών

- Χαρακτήρες <-> char , 1 byte, 8 bits, αναπαριστούν $2^8 = 256$ τιμές
- Ακέραιοι <-> int $\{-2^{\#bits-1}, -2^{\#bits-1} + 1, \ldots, 2^{\#bits-1} - 1\}$
- Πραγματικοί <-> float ή double

  Οι double έχουν μεγαλύτερο εύρος τιμών και μεγαλύτερη ακρίβεια αναπαράστασης από τους float

C type	Size (bytes)	Lower bound	Upper bound
char	1	—	—
unsigned char	1	0	255
short int	2	−32768	+32767
unsigned short int	2	0	65536
(long) int	4	-2^{31}	$+2^{31} - 1$
float	4	$-3.2 \times 10^{\pm 38}$	$+3.2 \times 10^{\pm 38}$
double	8	$-1.7 \times 10^{\pm 308}$	$+1.7 \times 10^{\pm 308}$

Τύπος Μεταβλητών και Σταθερών

- Ο τύπος μιας μεταβλητής σε μια έκφραση είναι ο τύπος με την οποία την δηλώνουμε
- Ο τύπος μιας σταθεράς είναι
  - Σταθερά χαρακτήρα
    - 'x' -> μετατροπή στον αντίστοιχο ακέραιο ASCII
  - Ακέραια (π.χ., 1234)
    - int, long int, unsigned long int, όποια τον χωράει
    - 1234U -> unsigned int ή unsigned long int
    - 1234L -> long int ή unsigned long int
  - Σταθερά κινητής υποδιαστολής
    - 0.1234 -> double
    - 0.1234f -> float

Σταθερές ποσότητες

- Ποσότητες που έχουν γνωστή αρχική τιμή και δεν αλλάζουν σε όλη την εκτέλεση του προγράμματος, δηλώνονται ως σταθερές ώστε ο compiler να μπορεί να προβεί σε βελτιστοποίηση του κώδικα και, ταυτόχρονα, να μπορεί να μας ειδοποιήσει αν κατά λάθος προσπαθήσουμε να μεταβάλουμε στο πρόγραμμα την τιμή ποσότητας που λογικά είναι σταθερή.
- Η δήλωση τέτοιας ποσότητας γίνεται χρησιμοποιώντας την προκαθορισμένη λέξη **const** και συνοδεύεται υποχρεωτικά με απόδοση της αρχικής (και μόνιμης) τιμής:
- **double const** pi = 3.141592653589793;
- **int const** maximum = 100;
- Σε ποσότητες που έχουν δηλωθεί ως **const** δεν μπορεί να γίνει ανάθεση τιμής.
- Καλό είναι να χρησιμοποιούνται συμβολικές σταθερές για να αποφεύγεται η χρήση "μαγικών αριθμών" στον κώδικα. Αν μία ποσότητα που είναι σταθερή (π.χ. πλήθος στοιχείων σε πίνακα, φυσικές ή μαθηματικές σταθερές) χρησιμοποιείται με την αριθμητική της τιμή και όχι με συμβολικό όνομα καθίσταται ιδιαίτερα δύσκολη η αλλαγή της καθώς πρέπει να αναγνωριστεί και να τροποποιηθεί σε όλα τα σημεία του κώδικα που εμφανίζεται.

Στατικές Μεταβλητές

- Οι μεταβλητές μπορεί να είναι:
  - **Τοπικές:** μια μόνο συνάρτηση χρησιμοποιεί την μεταβλητή που πρέπει να την θυμάται μεταξύ διαφορετικών κλήσεων. Μοιάζουν με τις καθολικές μεταβλητές αλλά με τη διαφορά πως δε «χάνονται» μετά το τέλος της συνάρτησης.
  - **Καθολικές:** ένα μικρό σύνολο από συναρτήσεις χρησιμοποιεί την μεταβλητή, οπότε η μεταβλητή δηλώνεται σαν στατική και όλες αυτές οι συναρτήσεις μπαίνουν στο ίδιο αρχείο
- Αρχικοποίηση στην αρχή της εκτέλεσης του **προγράμματος**
  - static int x=8;

Static variables

```
void a()
{
    static int staticVariable = 500; /* mono thn prwth fora*/
    staticVariable++;
    printf( "%d\n", staticVariable);
}
```

Output

501

502

503

....

Define – Const – Include

#define <macro> <replacement name>

#define FALSE 0

#define TRUE !FALSE

#define N 1000 /*ΠΡΟΣΟΧΗ!! – ΔΕΝ ΜΠΟΡΩ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΣΩ ΤΟ Ν ΩΣ ΜΕΤΑΒΛΗΤΗ*/

#define Pi 3.14159

- Ορίζεται στην αρχή κάθε προγράμματος και μπορεί να χρησιμοποιηθεί σε όλο το πρόγραμμα.
- Το Pi θα αντικατασταθεί από τον 3.14159 στο πρώτο βήμα του compilation (preprocessing)

int const a = 1;

- ορίζει μια σταθερά – δεν μπορεί να αλλάξει τιμή

#include <file>

#include <stdio.h>

#include "file"

Μέρος Γ

Εκφράσεις

Αριθμητικές Εκφράσεις

- Σύνταξη: a τ b ή τ a
  - τ είναι ο **τελεστής (operator)**
  - a, b είναι **τελεσταίοι (operands)**
- Τελεσταίοι μπορεί να είναι
  - Σταθερές (π.χ. **KMS_PER_MILE * miles**)
  - Μεταβλητές (π.χ. c = **a + b**)
  - Κλήση συνάρτησης που επιστρέφει αριθμό
    - (π.χ c = **sum(a,b) + sum(b,a)**)
  - Έκφραση (χρήση παρενθέσεων)

Αριθμητικοί Τελεστές

Όνομα	**Τελεστής**	**Παράδειγμα**
Πρόσθεση	+	num1 + num2
Αφαίρεση	-	initial - spent
Πολ/σμός	*	age * 6
Διαίρεση	/	sum / count
Υπόλοιπο	%	m % n

Υπόλοιπο (modulo) %

- Η έκφραση **m % n** επιστρέφει το υπόλοιπο της διαίρεσης του **m** με το **n.**
- Το modulo είναι ακέραιος τελεστής - και οι δύο τελεσταίοι **πρέπει να είναι ακέραιοι (ή/και τύπου *char*)**.
- π.χ :
  - 17 % 5 = 2
  - 6 % 3 = 0
  - 9 % 2 = 1
  - 5 % 8 = 5

Ακέραια διαίρεση

- Εάν και οι δύο τελεσταίοι είναι ακέραιοι τότε θα πάρετε ακέραιο ως απάντηση. Το δεκαδικό τμήμα απορρίπτεται. π.χ. :
  - 17 / 5 = 3
  - 4 / 3 = 1
  - 35 / 9 = 3
- Εάν ένας εκ των δυο τελεσταίων είναι float (ή double) θα πάρετε float (και αντίστοιχα double) ως αποτέλεσμα. π.χ.
  - 4.0 / 3 · 1.333333

Διαίρεση με το 0

- Δεν ορίζεται διαίρεση με 0 στα μαθηματικά.
- Αν επιτρέψετε διαίρεση με το 0 σε ένα πρόγραμμα θα προκληθεί σφάλμα ή θα λάβετε ως αποτέλεσμα το NaN (Not a Number).
  - Ανάλογα με το προγραμματιστικό περιβάλλον, Η εκτέλεση του προγράμματος θα τερματιστεί απότομα – **runtime error**

- *Θα μάθουμε αργότερα πώς να αποφεύγουμε τη διαίρεση με το 0*

Τελεστής Ανάθεσης (=)

- Σύνταξη: **μεταβλητή = έκφραση;**
- Παραδείγματα:
  - area = PI * radius * radius;
  - count = count + 1;
  - new_number = old_number;
  - average = total / count;
- Η τιμή του αποτελέσματος της έκφρασης στα δεξιά του τελεστή ανάθεσης αποθηκεύεται στη διεύθυνση της μεταβλητής

Τύπος Έκφρασης

- Ορίζεται από τους τύπους των τελεστέων
  - char, int, float, double

• int τ int	int	5/2	2
• double τ double	double	5.0/2.0	2.5
• int τ double	double	5/2.0	2.5
• double τ int	double	5.0/2	2.5
• int τ char	int	5+’a’	102

Ascii:97

Μετατροπή Τύπων

- **Αυτόματη μετατροπή**
  - Σε ανάθεση (αυτόματα) η τιμή στα δεξιά του = μετατρέπεται στον τύπο της μεταβλητής στα αριστερά του **=**
  - int x = 3.14; /* 3 */
  - float x = (2/3); /* 0.0 */
- **Ρητή Μετατροπή (Casting)**
  - float x = (float) 2/3; /* 2.0/3, 2.0/3.0, 0.66666 */
  - float x = (float) (2/3); /* 0.0 */

Τελεστές – Προτεραιότητα Πράξεων

- == (ισότητα), != (ανισότητα), < , > , <=, >=
- ! (όχι), && για λογικό ΚΑΙ (AND), || για λογικό Η (OR)
- Πρόσθεση – Αφαίρεση : **+ -**
- Πολλ/σμος – Διαίρεση :* /
  - double x = 5/2; //x = 2.0
  - double y = 5.0/2; //x = 2.5
- Υπόλοιπο διαίρεσης : **%**
  - int x = 5 % 2; //x = 1
- Σειρά των πράξεων:
  1. ( )
  2. !
  3. * / % (από αριστερά προς τα δεξιά)
  4. + - (από αριστερά προς τα δεξιά)
  5. < <= >= >
  6. == !=
- Επιστρέφουν 1 (ΑΛΗΘΕΣ) ή 0 (ΨΕΥΔΕΣ)

```
int s = (x == 1) || (x == 2);
int s = ! (x == 1) && (x == 2);
int s = 1+((x == 1) || (x != 2))*2;
```

0 || X -> X
!0 || X -> 1
0 && X -> 0
!0 && X -> X

Λογικοί Τελεστές

==	ίσο	!=	Άνισο
>	μεγαλύτερο	<	Μικρότερο
>=	μεγαλύτερο ή ίσο	<=	μικρότερο ή ίσο

Το αποτέλεσμα της σύγκρισης είναι λογική ποσότητα και επομένως έχει τιμή **true** ή **false** . Π.χ. το 3.0>2.0 είναι **true** ενώ το 2!=1+1 είναι **false** . Οι αριθμητικοί τελεστές έχουν μεγαλύτερη προτεραιότητα από τους σχεσιακούς.

Λογικοί Τελεστές

- Για τη σύνδεση λογικών εκφράσεων η C παρέχει τους λογικούς τελεστές

  ! (NOT), && (AND), || (OR)

- Οι τελεστές && και || δρουν μεταξύ δύο λογικών ποσοτήτων ή εκφράσεων και σχηματίζουν μια νέα λογική ποσότητα ενώ ο τελεστής ! δρα σε μία:
- Ο τελεστής ! δρα στη λογική έκφραση που τον ακολουθεί και της αλλάζει την τιμή:

  Το ! (4 > 3) είναι **false**

  Το ! (4 < 3) είναι **true**

- Η λογική έκφραση που σχηματίζεται συνδέοντας δύο εκφράσεις με τον τελεστή && έχει τιμή **true** μόνο αν και οι δύο ποσότητες είναι **true** . Σε άλλη περίπτωση είναι **false** :

  Το (4 > 3) && (3.0 > 2.0) είναι **true**

  Το (4 < 3) && (3.0 > 2.0) είναι **false**

- Ο τελεστής || μεταξύ δύο λογικών εκφράσεων σχηματίζει μια νέα ποσότητα με τιμή **true** αν έστω και μία από τις δύο ποσότητες είναι **true** , αλλιώς είναι **false** :

  Το (4 > 3) || (3.0 < 2.0) είναι **true**

  Το (4 < 3) || (3.0 < 2.0) είναι **false**

Λογικοί Τελεστές

- Μια έκφραση με σχεσιακούς ή λογικούς τελεστές της C, μπορεί να ανατεθεί σε μεταβλητές τύπου **int** :

int a = 3==2;

int b = ((i > 0) && (i < max));

- Ας αναφέρουμε εδώ ένα σημαντικό χαρακτηριστικό της C: ο υπολογισμός των λογικών εκφράσεων με θεμελιώδεις τύπους εκτελείται από αριστερά προς τα δεξιά και σταματά όταν έχει προσδιοριστεί η τελική τιμή (short-circuit evaluation). Π.χ. στην έκφραση
- (i < 0) || (i > max) αν ισχύει i < 0 τότε η συνολική έκφραση είναι **true** ανεξάρτητα από τη δεύτερη συνθήκη, η οποία δεν υπολογίζεται. Ανάλογα, στην έκφραση
- (i < 0) && (i > max) αν i >= 0 τότε η συνολική έκφραση είναι **false** και δεν υπολογίζεται το i > max .

Το χαρακτηριστικό αυτό είναι σημαντικό καθώς το τμήμα της λογικής έκφρασης που παραλείπεται μπορεί να περιλαμβάνει κλήση συνάρτησης με μεγάλες απαιτήσεις σε χρόνο εκτέλεσης ή μνήμη.

Αριθμητικοί Τελεστές

- Μοναδιαίοι +,- :
  - Δρώντας σε ένα αριθμό a μας δίνουν τον ίδιο αριθμό ή τον αντίθετό του αντίστοιχα.
- Δυαδικοί +,-,* :
  - Δρώντας μεταξύ δύο αριθμών a,b μας δίνουν το άθροισμα, τη διαφορά και το γινόμενό τους αντίστοιχα.
- Δυαδικός / μεταξύ πραγματικών a,b:
  - δίνει το λόγο a/b.
- Δυαδικοί /,% μεταξύ ακεραίων
  - δίνουν το πηλίκο και το υπόλοιπο της διαίρεσης αντίστοιχα.

Προσέξτε ότι δεν υπάρχει τελεστής για ύψωση σε δύναμη.

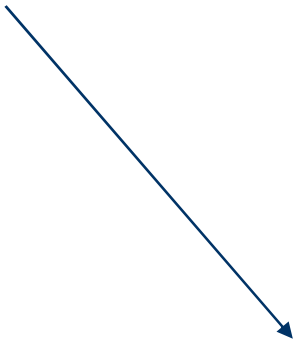
Τελεστές μοναδιαίας αύξησης / μείωσης

- *i++; (i--;)* : μεταθεματικοί (postfix)
- *++i; (--i;)* : προθεματικοί (prefix)
- Η αύξηση/μείωση συμβαίνει πριν (για προθεματικούς)/μετά (για μεταθεματικούς) την χρησιμοποίηση της τιμής
- Αν μια έκφραση έχει πολλαπλά ++ εξαρτάται από τον μεταγλωττιστή
- x = i++;
- x = ++i;

Τελεστές αντικατάστασης

- *έκφραση1 τελεστής= έκφραση2*
- *i += 2;* : i = i + 2;

- x *= y + 2;



δεξιά προς αριστερά!!

x = x * (y + 2);

Παράδειγμα

- b = --a + c;
  - ισοδυναμεί με
    - a = a-1;
    - b = a + c;
- ενώ το b = a-- + c;
  - ισοδυναμεί με
    - b = a + c;
    - a = a-1;

Bit operations

- bitwise AND &
- bitwise XOR ^
- bitwise OR |

- Ποια η διαφορά των (A|B) και (A||B) όπου, έστω A και B δύο ακέραιοι;

Μέρος Δ

Είσοδος / Έξοδος

printf / scanf

- Εισαγωγή
- Μορφή εισόδου / εξόδου
- Εκτυπώσεις με διαφορετική μορφοποίηση
- Παράμετροι
  - %c -- character
  - %d -- integer
  - %f -- float
  - %lf -- double
- Παράδειγμα

```
int x;
char c;
scanf("%d",&x);
c = 'a';
printf("x = %d c(char) = %c c(int) = %d\n",x,c,c);
```

```
#include <stdio.h>

void main ()
{
        double x;
        int n;

        /* Read in an integer. */
        printf("Please enter an integer: ");
        scanf("%d", &n);
        printf("The integer was %d\n\n", n);

        /* Read in a double. */
        printf("Please enter a double: ");
        scanf("%lf", &x);
        printf("The double was %g\n\n", x);

        /* Read in an integer and a double. */
        printf("Please enter an integer and a floating-point number: ");
        scanf("%d%lf", &n, &x);
        printf("The numbers were %d %g\n", n, x);
}
```

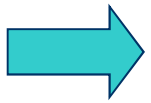
printf / scanf

- Μορφή εισόδου / εξόδου
- Παράμετροι
  - %c -- character
  - %d -- integer
  - %f -- float
  - %lf -- double
- Παράδειγμα

```
float f;

scanf("%f",&f);

printf("f = %f\nf ~= %.2f \n",f,f);
```

Παράδειγμα

```
#include <stdio.h>

int main()
{
    int a = 1, b = 1, x = 0, y = 0;
       double w;


     x = 1 + a++;
     printf( "x = %d\n", x);
     printf( "a = %d\n", a);
     y = ++b;
     printf( "y = %d\n", y);
     printf( "b = %d\n", b);
}
```



```
/* results
2
2
2
2
*/
```

Format identifiers

%d %i	Decimal signed integer.
%o	Octal integer.
%x %X	Hex integer.
%u	Unsigned integer.
%c	Character.
%s	String.
%f	double
%e %E	double.
%g %G	double.
%p	pointer.

Example I

```
#include <stdio.h>
int main ()
{
  /* We will use a floating-point and an integer variable. */
  double x;
  int n;

  /* Read in an integer. */
  printf("Please enter an integer: ");
  scanf("%d", &n);
  printf("The integer was %d\n\n", n);
}
```

Example I cont

```
  /* Read in a double. */
  printf("Please enter a double: ");
  scanf("%lf", &x);
  printf("The double was %g\n\n", x);

  /* Read in an integer and a double. */
  printf("Please enter an integer and a floating-point number: ");
  scanf("%d%lf", &n, &x);
  printf("The numbers were %d %g\n", n, x);
}
```

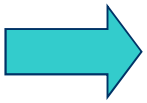
```
int day, year;
scanf("%d %d", &day, &year);
```

```
(1) char c, s[10]; int i; float f;
(2) scanf("%c", &c);  // reads next character and puts its value in c
(3) scanf("%s",  s);  // reads next word and converts it to a string
(4) scanf("%i", &i);  // reads next word and converts it to an integer
(5) scanf("%f", &f);  // reads next word and converts it to a float
(6) scanf("%c %s %i %f", &c, s, &i, &f);  // multiple reads
(7) printf("c=%c s=%s i=%i r=%3.1f¥n", c, s, i, f);
```

Παράδειγμα

```
#include <stdio.h>
int a;

void f(int a)
{
   int b=0;
   static int c=0;
   b++;
   c++;
   printf("%d %d %d\n",a,b,c);
}

void g(int b)
{
   b = a+b++;
   a = (b*b)/3;

   if (a > b)
      f(a);
   else
      f(b);
}
```

```
int main()
{
   for(a=1; a<=20; a++)
   {
      g(a);
   }
}
```



```
/* results*/

3 1 1
27 1 2
```

Τυπικά Λάθη

- printf
  - λιγότερα ορίσματα από όσα περιμένει
  - διαφορετικά ορίσματα από αυτά που περιμένει
- scanf
  - τα προηγούμενα
  - Ξεχάσατε το &
- Ακούσιες μετατροπές τύπων
- Ξεχάσατε το ;