Introduzione al linguaggio C

Reti di Elaboratori - 18/19
Corso di Laurea in Informatica
Università degli Studi di Roma “La Sapienza”

Andrea Coletta
coletta@di.uniroma1.it

Scorsa lezione:

- Struttura, compilazione ed esecuzione di programma C
- Regole di sintassi
- Includere librerie
- Tipi semplici
- Operazioni matematiche
- Alcune funzioni di base (printf, scanf, pow, …)
- Costrutti condizionali (if, else, …)
- Cicli ed iterazioni (for, while, …)

Cosa vedremo oggi:

- Strutture dati complesse
- Funzioni
- Puntatori
- Gestione della memoria

+

- Esercizi

Tipi

Finora abbiamo visto: **double**, **float, int, char**

Cosa manca?

Struct - Stringhe - Vettori - .....

Struct

Una struct è un raggruppamento di variabili anche di tipo diverso.

- (simile ad un oggetto ma senza metodi)

Le variabili sono denominate **membri,** in genere di tipo diverso, e sono identificate da un nome comune **(tag).**

Definizione: La definizione crea un nuovo tipo di dato.

```
struct nome_tag
{
    tipo1 nome_membro1;
    tipo2 nome_membro2;
    //...
};
```

```
struct point
{
    int x;
    int y;
    int z;
};
```

Struct

Dichiarazione:

```
struct tag variable_name;
```

```
//definizione
struct point
{
    int x;
    int y;
    int z;
};

//inizializzazione
struct point my_point = {0, 1, 2};
```

E' possibile inizializzare la variabile struct:

- con valori costanti utilizzando le parentesi graffe
- Mediante l'assegnazione di un'altra struct dello stesso tipo
- Chiamando una funzione che restituisca la struct

```
struct point my_point = {10,12,14};
struct point my_point = my_other_point;
struct point my_point = get_point(...);
```

Struct

Si può accedere ai singoli membri di una struct:

```
my_point.x;
my_point.y = 23;
```

Va utilizzato **il nome della variabile non** del tag della struct.

Possiamo avere struct di struct →

```
struct linea
{
    struct point p1;
    struct point p2;
}

struct linea my_linea = {{2, 3, 4}, {12, 9, 10}};

my_linea.p1.x;
```

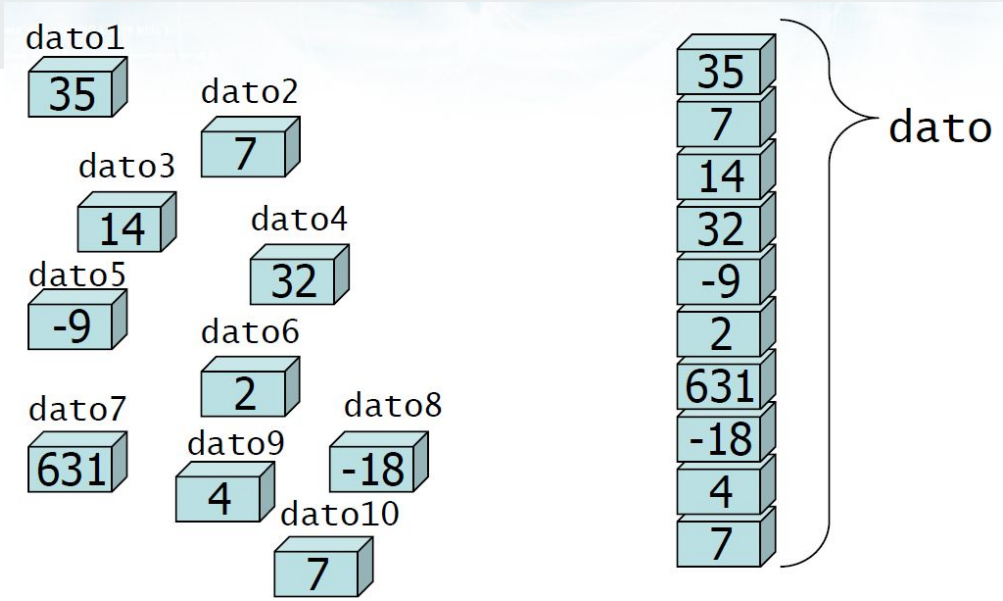
Struct - Esercizio

- **Provare a costruire una struct che rappresenta un triangolo.**
- **Prendete in input i vertici.**
- **Stampate a video le informazioni sul triangolo.**

```
//definizione
struct point
{
    int x;
    int y;
    int z;
};

//inizializzazione
struct point my_point = {0, 1, 2};
```

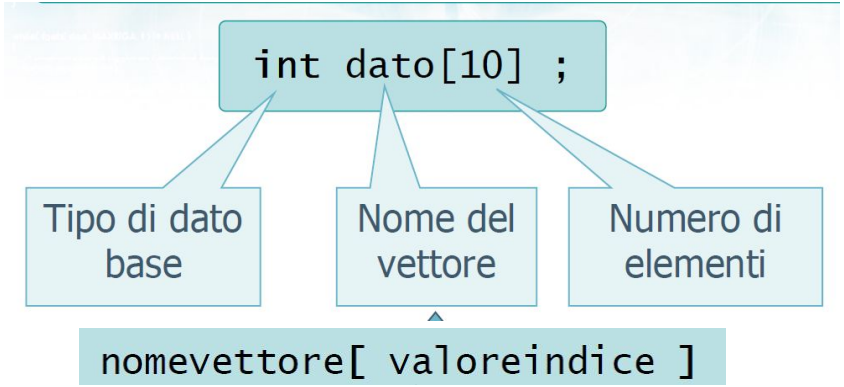
Vettori

- Permettono di aggregare serie di dati
- Elementi tutti dello stesso tipo
- Numero di elementi fisso (N)
- Elementi: O, 1, …, N -2, N-1



- Example

```
int dato[10];

for (i = 0; i < 10; i++)
    dato[i] = 0;
```



Caratteristiche di un vettore

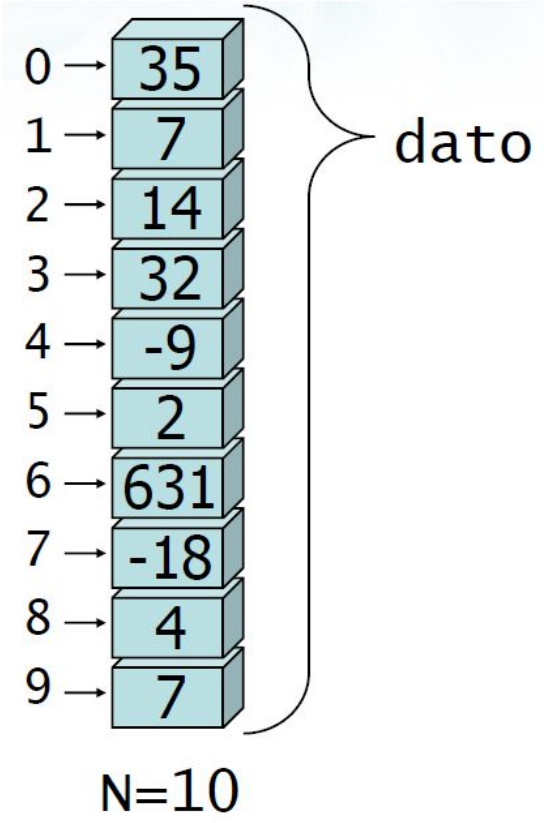
- **Caratteristiche statiche**
- Nome
  - dato
- Tipo di dato base
  - Int
- Dimensione totale
  - 10

Caratteristiche dinamiche

- Valori assunti dalle singole celle
  - 35, 7, 14, 32, ...

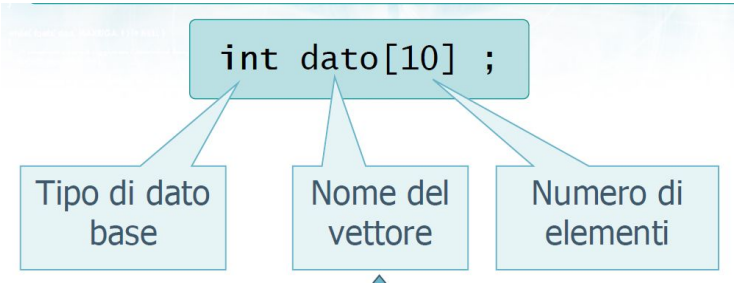
Ogni cella ha sempre un valore

- Impossibile avere celle “vuote”
- Le celle non inizializzate contengono valori ignoti



Vettore - esercizio

- Definire un vettore
- Leggere da tastiera 10 numeri
- Salvarli all'interno del vettore
- Stamparli in ordine inverso



Soluzione:

```
int dato[10];

for (i = 0; i < 10; i++)
    scanf("%d", &dato[i]);

for (i = 9; i >= 0; i--)
    printf("%d\n", dato[i]);
```

Cosa succede se stampate direttamente il vettore senza prima averlo scritto?

Se provate ad accedere all'11-esimo elemento?

Vettore - esercizio

Chi ci garantisce che per errore non utilizziamo dimensioni errate (differenti)?

Soluzione:

```
int dato[10];

for (i = 0; i < 10; i++)
    scanf("%d", &dato[i]);

for (i = 9; i >= 0; i--)
    printf("%d\n", dato[i]);
```

Cosa succede se stampate direttamente il vettore senza prima averlo scritto?

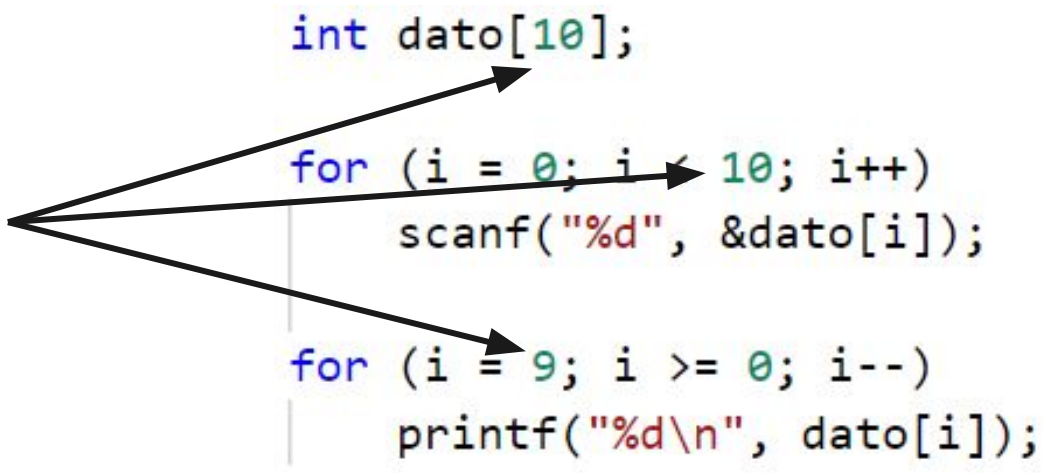
Se provate ad accedere all'11-esimo elemento?

Vettore - esercizio

Chi ci garantisce che per errore non utilizziamo dimensioni errate (differenti)?

Se volessi lavorare con 20 dati dovrei modificare in tutti questi punti?

Soluzione:



Cosa succede se stampate direttamente il vettore senza prima averlo scritto?

Se provate ad accedere all'11-esimo elemento?

Vettore - esercizio

Soluzione:

```
int dato[10];

for (i = 0; i < 10; i++)
    scanf("%d", &dato[i]);

for (i = 9; i >= 0; i--)
    printf("%d\n", dato[i]);
```

Chi ci garantisce che per errore non utilizziamo dimensioni errate (differenti)?

Se volessi lavorare con 20 dati dovrei modificare in tutti questi punti?

Soluzione: ***costanti simboliche***

Cosa succede se stampate direttamente il vettore senza prima averlo scritto?

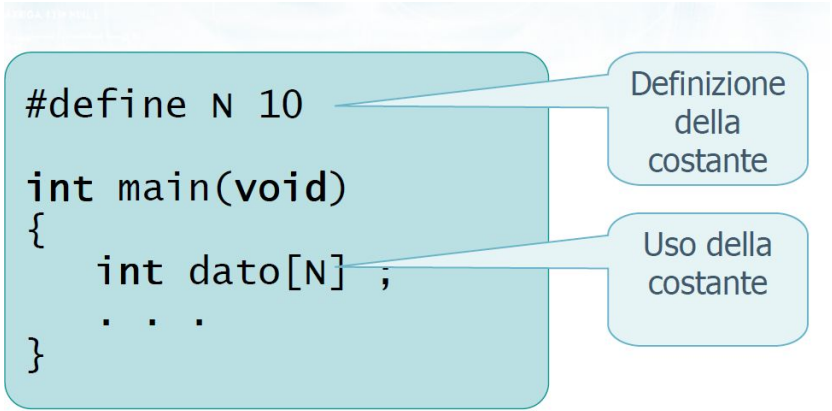
Se provate ad accedere all'11-esimo elemento?

Costanti simboliche

- Associamo un nome simbolico ad una costante e usiamo direttamente il nome simbolico
- Il compilatore si occuperà di sostituire, ad ogni occorrenza del nome, il valore numerico della costante.

Abbiamo due modi:

1- **#define**

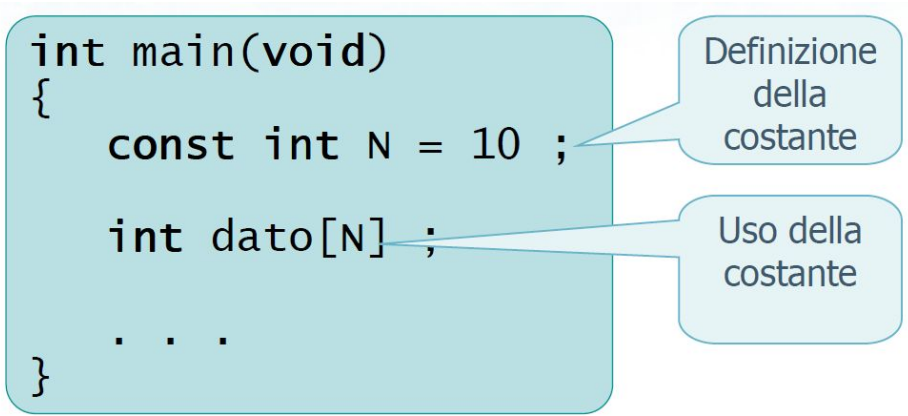


Costanti simboliche

- Associamo un nome simbolico ad una costante e usiamo direttamente il nome simbolico
- Il compilatore si occuperà di sostituire, ad ogni occorrenza del nome, il valore numerico della costante.

Abbiamo due modi:

2- Modificatore **const**



#define - const

```c
#define MAX 10

int main(void)
{
    int i;
    int dato[MAX];

    for (i = 0; i < MAX; i++)
        scanf("%d", &dato[i]);

    for (i = 0; i < MAX; i++)
        printf("%d\n", dato[i]);
}
```

```c
int main(void)
{
    const int MAX = 10;
    int i;
    int dato[MAX];

    for (i = 0; i < MAX; i++)
        scanf("%d", &dato[i]);
    for (i = 0; i < MAX; i++)
        printf("%d\n", dato[i]);
}
```

Operazioni su dati del vettore

- utilizzabile all'interno di un'espressione
  - `tot = tot + dato[i] ;`
- utilizzabile in istruzioni di assegnazione
  - `dato[0] = 0 ;`
- utilizzabile per stampare il valore
  - `printf("%d\n", dato[k]) ;`
- utilizzabile per leggere un valore
  - `scanf("%d\n", &dato[k]) ;`

- `if (dato[i]==0)`
  - se l'elemento contiene zero
- `if (dato[i]==dato[i+1])`
  - due elementi consecutivi uguali
- `dato[i] = dato[i] + 1 ;`
  - incrementa l'elemento `i`-esimo
- `dato[i] = dato[i+1] ;`
  - copia un dato dalla cella successiva

Vettori - Esercizio 3

- Scrivere un programma che:
  - Utilizzi **const** oppure **define**
  - Prende in input **10** valori e li inserisce in un vettore
  - Crea un nuovo vettore con gli elementi del primo invertiti [9, 8, …0]
  - Cerca il massimo valore all'interno del vettore invertito
  - Stampa l'indice del massimo valore

Vettore 4 - Istogramma

Dato un vettore con all'interno i seguenti valori:

3, 5, 7, 2, 5, 3, 1, 6

Stampare l'istogramma relativo:

```
###
#####
#######
...
```

Le Stringhe

Come sono rappresentati in C?

- **Vettori di caratteri**

Ricordiamo che:

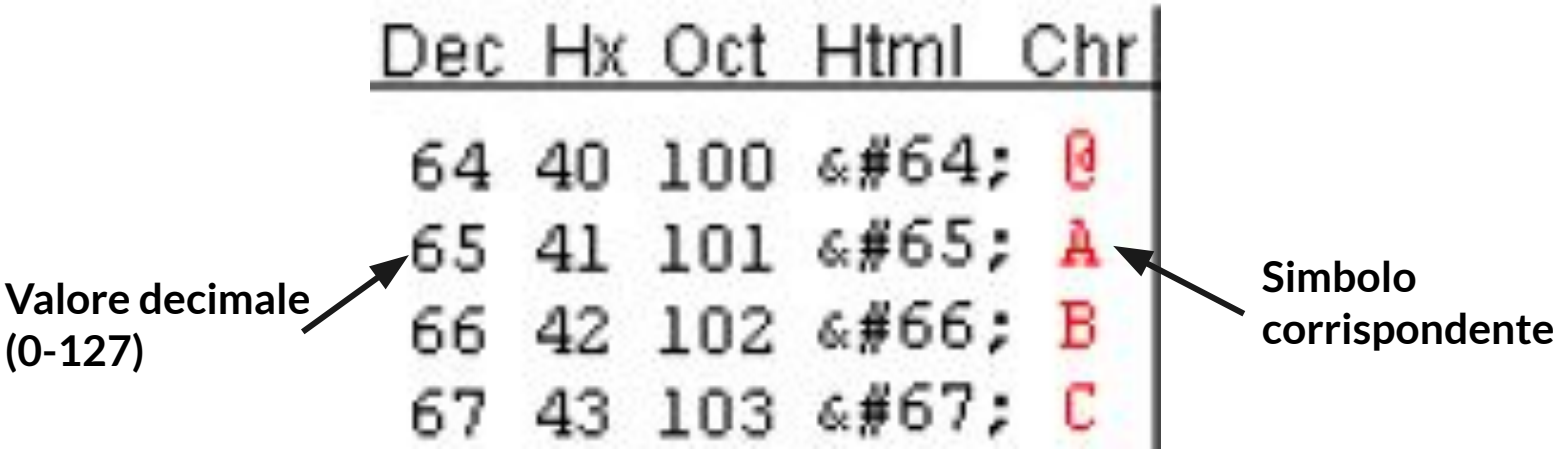
- Ogni carattere viene rappresentato dal proprio codice ASCII
- Sono sufficienti 7 bit per rappresentare ciascun carattere
- Il C usa variabili di 8 bit (1 byte)
- Non sono previste le lettere accentate né altri simboli diacritici
    - Richiedono estensioni speciali e librerie specifiche

char

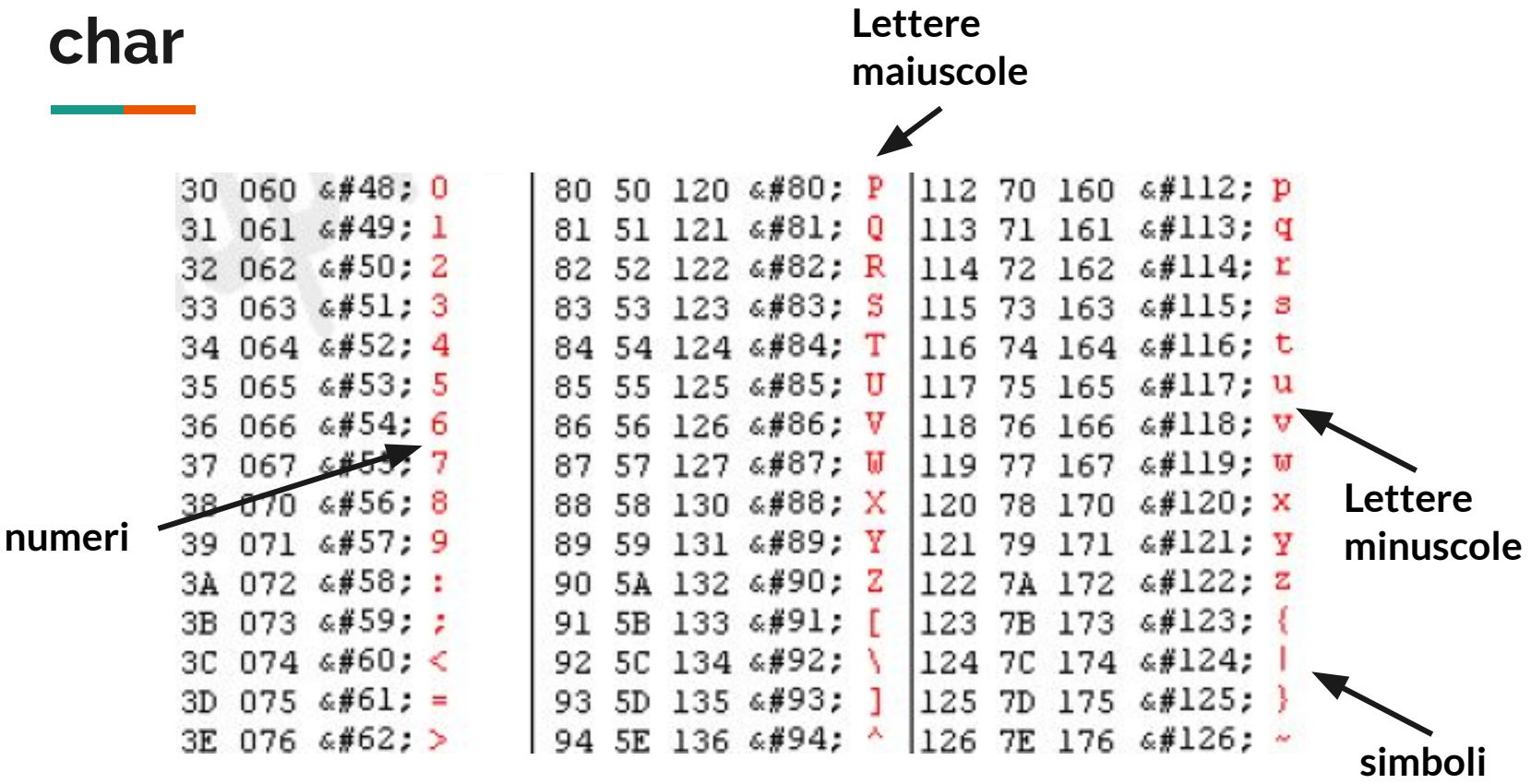
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of 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

Source: www.lookuptables.com

char


Dec Hx Oct Html Chr
64 40 100 @ @
65 41 101 A A
66 42 102 B B
67 43 103 C C
Valore decimale (0-127)
Simbolo corrispondente

char



30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~

char

```c
int i ;
char c ;

c = 'A' ;
c = 65 ; /* equivalente! */
i = c ; /* i sarà 65 */
c = c + 1 ; /* c sarà 66 = 'B' */
c = c * 2 ; /* non ha senso... */
if (c == 'Z') ...
for( c='A'; c<='Z'; c++) ...
```

```c
if( ch>='A' && ch<='Z' )
    printf("%c lettera maiuscola\n", ch) ;

if( ch>='a' && ch<='z' )
    printf("%c lettera minuscola\n", ch) ;

if( (ch>='A' && ch<='Z') ||
    (ch>='a' && ch<='z') )
    printf("%c lettera\n", ch) ;
```

char

Definite in `<ctype.h>`

Analizzano un singolo carattere, identificandone la tipologia

- Lettera
  - Maiuscola
  - Minuscola
- Cifra
- Punteggiatura

- isalpha
- isupper
- islower
- isdigit
- isalnum
- isxdigit
- ispunct
- isgraph
- isprint
- isspace
- iscntrl

Definite in `<ctype.h>`

Convertono tra lettere maiuscole e lettere minuscole

- toupper
- tolower

char - Esercizio

Scrivere un programma che:

- Definisca una nuova stringa di testo qualsiasi
- Se è tutta maiuscola
    - Allora stampare “VERO”
    - Altrimenti
        - “FALSO”

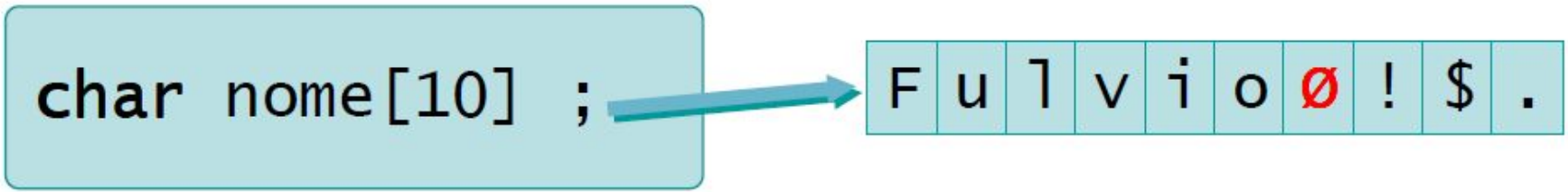
```
const char name[] = {'c','i','a','o','\0'};

 char stringa[] = "Hello world";
```

Stringhe

Per ottimizzare l'utilizzo delle stringhe il linguaggio C richiede che ogni stringa sia terminata con un carattere speciale, **un terminatore**.

Il terminatore corrisponde al carattere di codice **ASCII pari a zero:**

- nome[6] = **0 ;**
- nome[6] = **'\0'**



Tutte le funzioni della libreria standard C rispettano questa convenzione:

- Si aspettano che la stringa sia terminata (es. printf("%s", stringa);)
- Restituiscono sempre stringhe terminate

getchar e putchar

Esistono due insiemi di funzioni che permettono di leggere e stampare variabili di tipo char:

1. Le funzioni printf/scanf, usando lo specificatore di formato "%c"
2. Le funzioni **putchar** e **getchar**

```
char ch ;

printf("%c", ch) ;
```

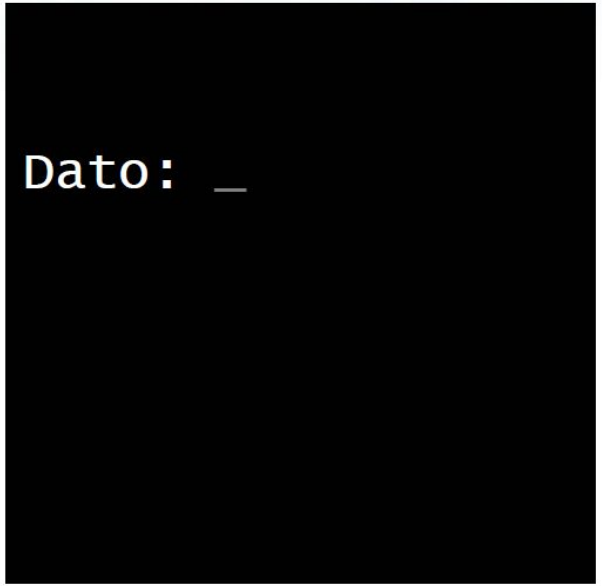
```
char ch ;

scanf("%c", &ch) ;
```

```
char ch ;

putchar(ch) ;
```

```
char ch ;

ch = getchar() ;
```

getchar

```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```



Il programma stampa l'invito ad inserire un dato

getchar

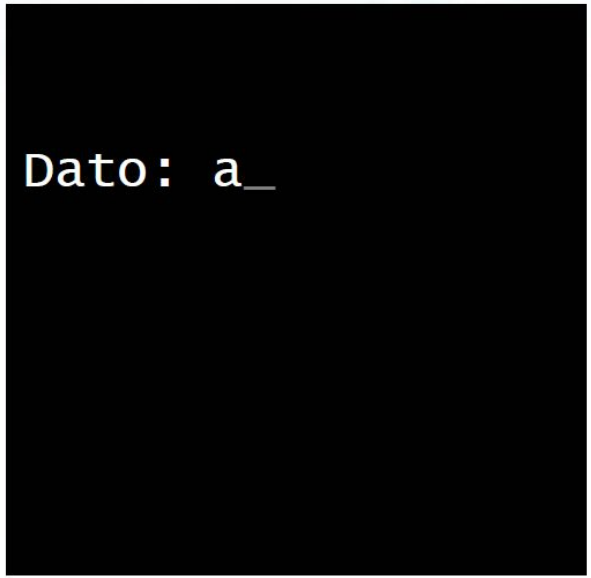
```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```

```
Dato: _
```

getchar blocca il programma in attesa del dato

getchar

```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```



L'utente immette 'a', il programma non lo riceve

getchar

```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```

```
Dato: a
_
```

L'utente immette Invio, il programma prosegue

getchar

```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```

```
Dato: a
_
```

Ora ch='a', il programma fa un'altra getchar()

getchar

```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```

```
Dato: a
_
```

Il programma **non** si blocca in attesa dell'utente

getchar

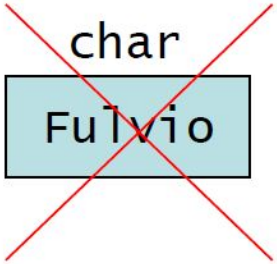
```
char ch,ch2 ;

printf("Dato: ");

ch = getchar() ;

ch2 = getchar() ;
```

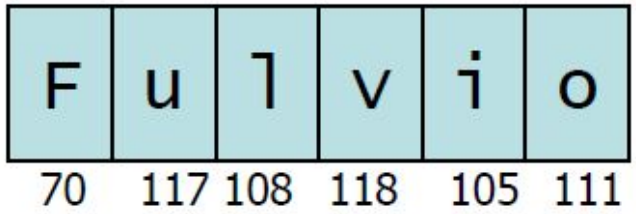
```
Dato: a
_
```

C'era già un carattere pronto: Invio! ch2='\n'

Stringhe

1. Una stringa è una struttura dati capace di memorizzare sequenze di caratteri
2. In C non esiste un tipo di dato specifico (salvo librerie esterne)
3. Si usano vettori di caratteri
4. La lunghezza di una stringa è tipicamente variabile durante l'esecuzione del programma





Stringhe (es. 6)

Leggere una stringa da tastiera:

- Decidere dimensione massima
- Definire il vettore per la stringa
- Leggere da tastiera i caratteri immessi finchè non c'è un carattere di end-line oppure superiamo la dimensione max.
- Stampare la stringa

```
char ch ;

scanf("%c", &ch) ;
```

```
char ch ;

ch = getchar() ;
```

Credits:

Parte dei contenuti/slides/immagini sono presi da:

http://www.cs.unibo.it/~dilena/Homepage//PROG1617_files/Slide3.pdf

http://elite.polito.it/files/courses/12BHD/progr/DVD-slides.pdf

https://www.devapp.it/wordpress/7-operazioni-matematiche-e-logiche/

http://disi.unitn.it/~agiordani/mat/puntatori.pdf

https://www.html.it/pag/15403/controlli-condizionali-if-else/

https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/