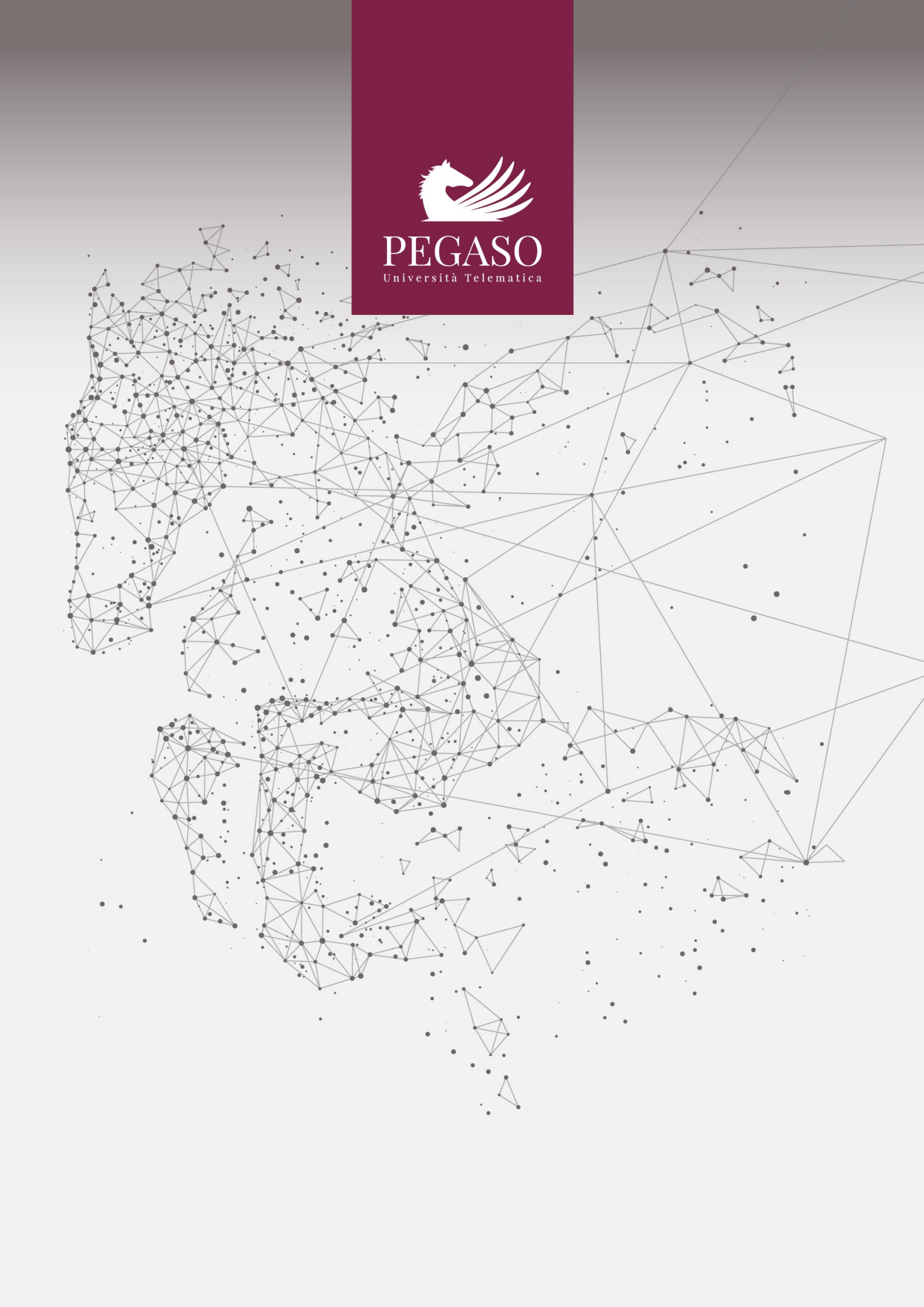

PEGASO
Università Telematica

Indice

1. Funzione getchar()

Consideriamo il seguente esempio:

```
01  // Programma C
02  // Acquisizione di caratteri di input.
03  #include <stdio.h>
04
05  int main(void)
06  {
07     printf(  "Enter a char, EOF to end:\n" );
08     int ch;
09     while ( (ch = getchar()) != EOF ) {
10        if ( ch != '\n' ) {
11           printf(  "Char + 1 is:%c\n", ch );
12           printf(  "Enter a char, EOF to end:\n" );
13        }
14     }
15  }
```

```
Enter a char, EOF to end:
a
Char + 1 is: b
Enter a char, EOF to end:
B
Char + 1 is: C
^D _____Non tutti i sistemi mostrano una rappresentazione del carattere EOF
```

Nell'intestazione:

```
while ((ch = getchar()) != EOF)
```

viene eseguita per prima l'assegnazione fra parentesi

```
(ch = getchar())
```

La funzione

`getchar (da <stdio.h>)`

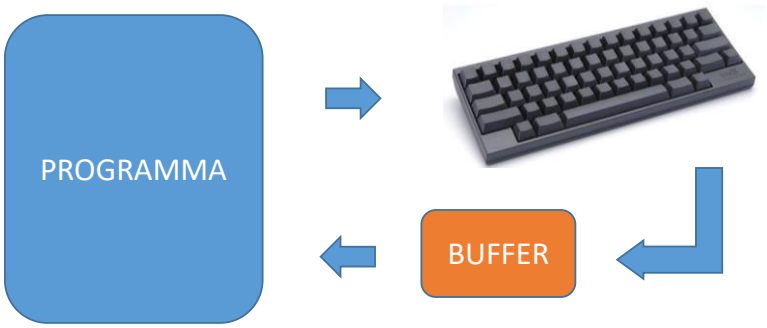
legge un carattere dalla tastiera e lo memorizza nella variabile intera `ch`.

Quando viene chiamata `getchar()`, il controllo del programma passa all'utente.

Il programma rimane in attesa che l'utente inserisca un carattere sulla tastiera.

La funzione `getchar()` appartiene al gruppo di funzioni che gestiscono l'input da tastiera mediante un **buffer di memoria**.

Il controllo dell'esecuzione non viene restituito al programma fino a quando l'utente non preme il tasto *Invio*.



NOTA: il contenuto del buffer viene passato al programma solo quando l'utente preme *Invio*.

Quindi l'utente può inserire quanti caratteri vuole, non c'è un controllo.

Questo è potenzialmente pericoloso.

Dal buffer, ogni volta che viene chiamata `getchar()` viene prelevato un solo carattere.

I caratteri sono normalmente memorizzati in variabili di tipo `char`.

Tuttavia, una caratteristica importante del C è che i caratteri possono essere memorizzati in qualsiasi tipo di dato intero, perché di solito sono rappresentati nel computer come interi di un solo byte.

La funzione `getchar` restituisce come `int` il carattere immesso dall'utente.

Possiamo trattare un carattere o come un intero o come un carattere, a seconda del suo uso.

2. Caratteri ed EOF

L'istruzione

```
printf("The character (%c) has the value %d.\n", 'a', 'a');
```

usa gli specificatori di conversione `%c e %d` per stampare, rispettivamente, il carattere 'a' e il suo valore intero.

Il risultato è

```
The character (a) has the value 97.
```

ASCII

Il numero intero 97 è la rappresentazione numerica del carattere nel computer.

Molti computer oggi usano il set di caratteri ASCII (American Standard Code for Information Interchange) nel quale il numero 97 rappresenta la lettera minuscola 'a'.

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Symbol	Decimal	Binary
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110
g	103	01100111
h	104	01101000
i	105	01101001
j	106	01101010
k	107	01101011
l	108	01101100
m	109	01101101
n	110	01101110
o	111	01101111
p	112	01110000
q	113	01110001
r	114	01110010
s	115	01110011
t	116	01110100
u	117	01110101
v	118	01110110
w	119	01110111
x	120	01111000
y	121	01111001
z	122	01111010

I caratteri si possono leggere con `scanf` usando lo specificatore di conversione `%c`.

L'intera operazione di assegnazione ha in realtà un valore.

Questo valore è assegnato alla variabile sul lato sinistro dell'operatore =.

Il valore dell'espressione di assegnazione

```
ch = getchar()
```

è il carattere restituito da `getchar` e assegnato alla variabile `ch`.

Il fatto che le assegnazioni abbiano valori può essere utile per assegnare a diverse variabili lo stesso valore.

Ad esempio,

```
a = b = c = 0;
```

esegue dapprima l'assegnazione `c = 0` (poiché l'operatore = è associativo da destra a sinistra).

Alla variabile b è allora assegnato il valore dell'assegnazione `c = 0` (che è 0).

Quindi, alla variabile a è assegnato il valore dell'assegnazione `b = (c = 0)` (che è pure 0).

Nel programma il valore dell'assegnazione

```
ch = getchar()
```

viene confrontato con il valore di `EOF` (un simbolo che indica la fine di un file e il cui acronimo sta per "end of file").

Usiamo `EOF` (che normalmente ha il valore –1) come valore sentinella.

L'utente digita una combinazione di tasti dipendente dal sistema per indicare "end of file", ossia "non ho più dati da inserire".

`EOF` è una costante intera simbolica definita nell'intestazione `<stdio.h>`

Se il valore assegnato a `ch` è uguale a `EOF`, il programma termina.

Abbiamo scelto di rappresentare in questo programma i caratteri con il tipo `int`, perché `EOF` ha un valore intero (come si è detto, normalmente –1).

Note per la portabilità

- La combinazione di tasti per inserire un `EOF` (end of file) dipende dal sistema.
- Effettuare il test sulla costante simbolica `EOF` (anziché –1) rende i programmi più portabili.

  Il C standard stabilisce che `EOF` è un valore intero negativo (ma non necessariamente –1).

Pertanto, `EOF` potrebbe avere valori differenti su sistemi differenti.

3. Inserimento dell'indicatore EOF

Sui sistemi Linux/UNIX/Mac OS X, l'indicatore EOF è inserito scrivendo

<Ctrl> d

su un riga separata.

Questa notazione *<Ctrl> d* significa premere simultaneamente il tasto *Ctrl*

e il tasto *d*.

Su altri sistemi, come Windows di Microsoft, l'indicatore EOF può essere inserito

scrivendo

<Ctrl> z

Anche su Windows dovete premere Invio.

L'utente inserisce i voti attraverso la tastiera; quando preme il tasto Invio, i caratteri sono letti dalla funzione `getchar` un carattere alla volta.

La lettura dei caratteri uno alla volta può causare alcuni problemi.

Per far sì che un programma legga i caratteri dovete inviarli al computer premendo Invio. Questo fa sì che venga inserito nell'input il carattere *newline* dopo il carattere che desideriamo processare.

Spesso, questo carattere *newline* deve essere opportunamente ignorato per far funzionare il programma in modo corretto.

☺ *Prevenzione di errori*

Ricordatevi di prevedere un'elaborazione specifica per il carattere newline (o altri caratteri di spaziatura) nell'input quando processate i caratteri uno alla volta.

4. Note sui tipi interi

Una costante di tipo carattere può essere rappresentata come un carattere specifico tra virgolette singole, come ad esempio 'A'.

I caratteri devono essere racchiusi entro virgolette singole per essere riconosciuti come costanti di tipo carattere (i caratteri tra virgolette doppie sono riconosciuti come stringhe). Le costanti intere sono semplicemente valori interi.

I linguaggi portabili come il C devono prevedere dimensioni flessibili per i tipi di dati. Applicazioni differenti possono necessitare di interi di dimensioni differenti.

Il C fornisce diversi tipi di dati per rappresentare interi.

Oltre a `int` e a char, il C fornisce i tipi `short int` (che si può abbreviare in short) e `long int` (che si può abbreviare in `long`), così come variazioni `unsigned` di tutti i tipi interi.

Il C standard specifica l'intervallo minimo di valori per ogni tipo di intero, ma l'intervallo reale può essere maggiore e dipende dall'implementazione.

Per gli `short int` l'intervallo minimo va da –32767 a +32767.

Per la maggior parte dei calcoli con interi, i `long int` sono sufficienti.

L'intervallo minimo di valori per i `long int` va da –2147483647 a +2147483647. L'intervallo di valori per un `int` è maggiore o uguale a quello di uno `short int` e minore o uguale a quello di un `long int`.

Su molte delle piattaforme odierne gli `int` e i `long int` rappresentano lo stesso intervallo di valori.

Il tipo di dati `signed char` può essere utilizzato per rappresentare interi nell'intervallo da –127 a +127 o uno qualunque dei caratteri nell'insieme dei caratteri del computer.

Riferimenti bibliografici

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.