Laboratorio 1

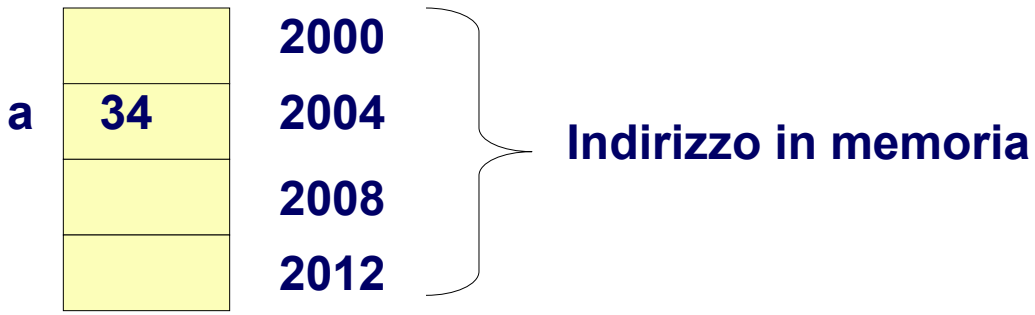
Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2025-2026

Università degli Studi di Trento
pierluigi.roberti@unitn.it

Introduzione al C

- In informatica, una **variabile** identifica una porzione di memoria destinata a **contenere dei dati**, che **possono essere modificati** nel corso dell'esecuzione di un programma. Una variabile è caratterizzata da un nome
- Una **costante** è invece una posizione di memoria che **mantiene lo stesso valore per tutta la durata dell'esecuzione del programma**.
- In C, linguaggio tipizzato, ciascuna **variabile** è caratterizzata da un **tipo di dato**, che specifica come devono essere considerati i dati rappresentati
  - **int** => numero intero
  - **float** => numero reale
  - **char** => carattere
- **Da sapere/ricordare:**
  - Le variabili vanno sempre "dichiarate"
    <tipo> <variabile>; esempio **int a;**
  - Le variabili vanno sempre "inizializzate"
    <variabile> = <espressione>; esempio **a = 34;**



Esempi

- `int a;`
  - Dichiarazione
- `a = 100;`
  - Inizializzazione

- `int a=100;`
  - Dichiarazione e inizializzazione

- `int n = 100;`
- `char c = 'z';`
- `float f = 1.56;`

Float
Separatore parte intera e parte decimale: punto [.]
e NON virgola

Nomi variabili

- Identificativi univoci
  - All'interno dello stesso campo di esistenza (si vedrà il concetto più avanti nel corso)
- Caratteri ammessi
  - Caratteri maiuscoli [A-Z]
  - Caratteri minuscoli [a-z]
  - Numeri [0-9]
  - Underscore [_]
- NON può cominciare con un numero
- Nomi significativi
  - In base al valore che la variabile contiene

- Primo carattere minuscolo
  - Per prassi

Esempi

- ~~`int a;`~~
  `int n_lanci;`
  - Variabile contiene il numero di lanci di un dado
- ~~`char x;`~~
  `char m_p;`
  - Variabile identifica mattino (m) o pomeriggio (p)
- ~~`float z;`~~
  `float temperatura;`
  `float temp;`
  - Variabile contiene valore di temperatura

Stampa – printf

```
printf( format [, param1, param2, …] );
```

- format
  - Testo che deve essere scritto a video
  - Può contenere valori (descrittori di formato) che devono essere specificati nei successivi parametri opzionali
- paramX
  - Opzionale
  - Sostituisce il descrittore di formato corrispondente
  - Valore esplicito, operazione, variabile
- Uso
  - Fornire istruzioni
  - Fornire risultati

Esempi

Descrittore di formato e valore DEVONO essere congruenti!!

- `printf("ciao mondo");`
  - A video: `Ciao mondo`
- `printf("valore: %d", 12);`
  - A video: `valore: 12`
- `printf("risultato: %d", 12+8);`
  - A video: `valore: 20`
- ```
  int x = 100;
  printf("valore: %d", x);
  ```
  - A video: `valore: 100`
- ```
  char a = 'z'; int b = 100; float h = 12.14;
  printf("valori: %c %d %f", a, b, h);
  ```
  - A video: `valori: Z 100 12.14`

Lettura – scanf

```
scanf( format [, param1, param2, …] );
```

- format
  - Tipi di dato che ci si attende di leggere
  - Contiene valori (descrittori di formato) che devono essere specificati nei successivi parametri opzionali
- paramX
  - Opzionale
  - Contenitori (variabili) dove i valori letti devono essere memorizzati
- Uso
  - Lettura di informazioni da tastiera (da utente)

Esempi

- ```
  int x;
  scanf("%d", &x);
  ```

- ```
  char a; int b; float g;
  scanf("%c", &a);
  scanf("%d", &b);
  scanf("%f", &g);
  ```

Descrittore di formato e valore DEVONO essere congruenti!!

Importante **&nome_variabile**

Nota – scanf di carattere

- Descrittore di formato: %c
- Consiglio: sintassi campo format
  - Lettura di un carattere
  - Un solo descrittore di formato
- Possibile avere più di un descrittore di formato per ogni scanf
  - Utilizzarlo quando confidenti con sintassi e funzionamento della funzione
- Se problemi in lettura
  - Far precedere la lettura del carattere dalla funzione
  - Windows

```
fflush(stdin);
```

  - Linux/MacOS

```
scanf("%*c");
```

Descrittori di formato

%d %i
valori interi relativi in base decimale

%u
valori interi senza segno (unsigned int)

%f
valori con virgola (float) in notazione “normale”

%e
valori con virgola in notazione esponenziale

%g
valori con virgola
seglie in modo automatico la notazione migliore

%c
carattere

%s
stringa (si vedra poi!!)

%o
interi assoluti in base ottale

%x
interi assoluti in base esadecimale

Sequenze particolari

- Sequenze di caratteri che provocano particolari comportamenti

\n
: New line (return, invio, andata a capo)

\t
: Tabulazione orizzontale

\"
: Carattere virgolette

\\
: Carattere barra

Commenti

- I commenti sono porzioni di testo del programma che vengono ignorate dal compilatore
  - Non contribuiscono all'esecuzione
- Sono estremamente utili per documentare il funzionamento del programma!
  - Ne semplificano la comprensione e quindi, ad esempio, la modifica

Commenti

- Su una sola linea:
  - **//** …

Es:

```
printf("a=%d", a); // stampa il valore di a
```

- Su più linee:
  - **/*** …
    … ***/**

Esempio:

```
/* la prossima istruzione stampa un
messaggio seguito dal valore di a */
printf("a=%d", a);
```

Esercizio 1

- Scrivere un programma che esegue il prodotto di 2 numeri interi x e y e ne stampa il valore z a video
  - Inizialmente, scrivere il programma in modo che i valori x e y siano assegnati all'interno del programma stesso
  - Provate a non assegnare un valore a x e y. Cosa succede?
  - Modificare il programma in modo da leggere i valori di x e y da tastiera

- **Da sapere/ricordare:**
  - La prima linea del programma deve essere:

    #include <stdio.h>
  - Non dimenticarsi l'intestazione main(){ … }
  - Per leggere un intero da tastiera:

    scanf("%d", &<variabile>);
  - Per scrivere un intero:

    printf("<messaggio> %d", <variabile>);
  - Ricordarsi di inserire **system("PAUSE")** alla fine del programma
    - Altrimenti la finestra di esecuzione si chiude automaticamente e non riuscite a vedere il risultato…

Esercizio 1 - Soluzione

```
#include <stdio.h>

int main() {
  int x, y, z;
  printf("valore di x?\n");
  scanf("%d", &x);
  printf("valore di y?\n");
  scanf("%d", &y);
  z = x * y;
  printf("Il prodotto di %d e %d e' %d", x,y,z);
  system("PAUSE");
}
```

Esercizio 2

- Scrivere un programma che esegue la divisione di 2 numeri x e y
- **Da sapere/ricordare:**
  - La prima linea del programma deve essere:
    #include <stdio.h>
  - Non dimenticarsi l'intestazione main(){ … }
  - In generale, il risultato di una operazione aritmetica è un valore reale e non intero. Quindi:
    - le variabili vanno dichiarate float e non int
    - Nelle istruzioni di I/O bisogna usare %f invece di %d
  - Per leggere un numero decimale da tastiera:
    scanf("%f", &<variabile>)
  - Per scrivere un valore reale (float):
    printf("<messaggio> %f", <variabile>)
  - Le variabili vanno "dichiarate"
    - Per dichiarare un numero reale (con la virgola): float <variabile>
  - Provare a vedere cosa succede se il risultato è dichiarato int quando ci si dovrebbe invece attendere un float

Esercizio 2 - Soluzione

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  float d1, d2;
  float ris;
  scanf("%f", &d1);
  scanf("%f", &d2);
  ris=d1/d2;
  printf("la divisione tra %f e %f vale %f\n",
                  d1, d2, ris);

  system("PAUSE");
  return 0;
}
```

Esercizio 3

- Scrivere un programma che, dato un valore da tastiera corrispondente a una temperatura in gradi Fahrenheit, ne stampa a video il valore in gradi Celsius

- **Da sapere/ricordare:**
  - °C = 5/9 × (°F - 32)
  - In generale, il risultato è un valore reale e non intero. Il C richiede che:
    - le variabili siano dichiarate float e non int
    - I valori abbiano tutti virgola decimale (es., 5.0, 32.0)
    - Nelle istruzioni di I/O bisogna usare %f invece di %d
    - Provare per credere: cosa succede se sono dichiarate interi e se usate 5/9 anziché 5.0/9.0 ?

Esercizio 3 - Soluzione

```
#include <stdio.h>

int main() {
  float fahr, cels;
  printf("Inserire la temperatura
         in Fahrenheit da convertire\n");
  scanf("%f",&fahr);
  cels = (float)5/9 * (fahr - 32.0);
  printf("La temperatura di %f gradi Fahrenheit
         corrisponde a %f gradi Celsius\n",fahr,cels);
  system("PAUSE");
}
```

Esercizio 3 - Soluzione

```
#include <stdio.h>

int main() {
  float fahr, cels;
  printf("Inserire la temperatura
         in Fahrenheit da convertire\n");
  scanf("%f",&fahr);
  cels = 5/(float)9 * (fahr - 32.0);
  printf("La temperatura di %f gradi Fahrenheit
         corrisponde a %f gradi Celsius\n",fahr,cels);
  system("PAUSE");
}
```

Esercizio 3 - Soluzione

```
#include <stdio.h>

int main() {
  float fahr, cels;
  printf("Inserire la temperatura
         in Fahrenheit da convertire\n");
  scanf("%f",&fahr);
  cels = (5.0/9.0) * (fahr - 32.0);
  printf("La temperatura di %f gradi Fahrenheit
         corrisponde a %f gradi Celsius\n",fahr,cels);
  system("PAUSE");
}
```

Esercizio 4.1

- Cosa stampa il seguente programma? Perché?

```
int main()
{
    char car;
    int n1, n2, totint;
    float f1,f2, totfloat;
    printf("Inserisci un carattere: ");
    scanf("%c", &car);
    printf("\nInserisci un numero intero n1: ");
    scanf("%i", &n1);
    printf("\nInserisci un secondo numero intero n2: ");
    scanf("%i", &n2);
    printf("\nInserisci un numero con virgola f1: ");
    scanf("%f", &f1);
    printf("\nInserisci un secondo numero con virgola f2: ");
    scanf("%f", &f2);
```

Esercizio 4.2

```
    printf("\n\n%c%c %c\n%c\n", car, car, car, car);

    totint=n1+n2;
    printf("n1+n2= %d\n\n", totint);
    totint=n1*n2;
    printf("n1*n2= %d\n\n", totint);
    totfloat=n1/n2;
    printf("n1/n2= %f\n\n", totfloat);
    printf("n1/n2= %f\n\n", n1/n2);
    totfloat=f1/f2;
    printf("f1/f2= %f\n\n", totfloat);
    printf("f1/f2= %f\n\n", f1/f2);
    totfloat=n1+f2;
    printf("n1+f2= %f\n\n", totfloat);
    return 0;
}
```

Esercizio 5

- Perché questo funzionamento?

```
#include <stdio.h>
int main()
{
   printf("\n%f\n", .1+.1+.1+.1+.1+.1+.1+.1+.1+.1);
   printf("\n%e\n", .1+.1+.1+.1+.1+.1+.1+.1+.1+.1);
   printf("\n%f\n", .1+.1+.1+.1+.1+.1+.1+.1+.1+.1-1);
   printf("\n%e\n", .1+.1+.1+.1+.1+.1+.1+.1+.1+.1-1);
   system("PAUSE");
   return 0;
}
```

Esercizio 6

- Scrivere un programma in grado di
  - Richiedere all'utente 4 caratteri
  - Memorizzare i 4 caratteri in altrettante variabili
  - Stampare i 4 caratteri nel seguente modo
    - 1 per riga
    - Tutti su una sola riga senza separazione tra i caratteri
    - Tutti su una sola riga separati da spazi

Esercizio 6 soluzione

```
#include <stdio.h>
int main()
{
 char c1,c2,c3,c4;
 fflush(stdin); printf("c1="); scanf("%c",&c1);
  fflush(stdin); printf("c2="); scanf("%c",&c2);
  fflush(stdin); printf("c3="); scanf("%c",&c3);
  fflush(stdin); printf("c4="); scanf("%c",&c4);

  printf("Output1:\n%c\n%c\n%c\n%c\n",c1,c2,c3,c4);
  printf("Output2:\n%c%c%c%c\n",c1,c2,c3,c4);
  printf("Output3:\n%c %c %c %c\n",c1,c2,c3,c4);
   system("PAUSE");
   return 0;
}
```

Esercizio 6 – input

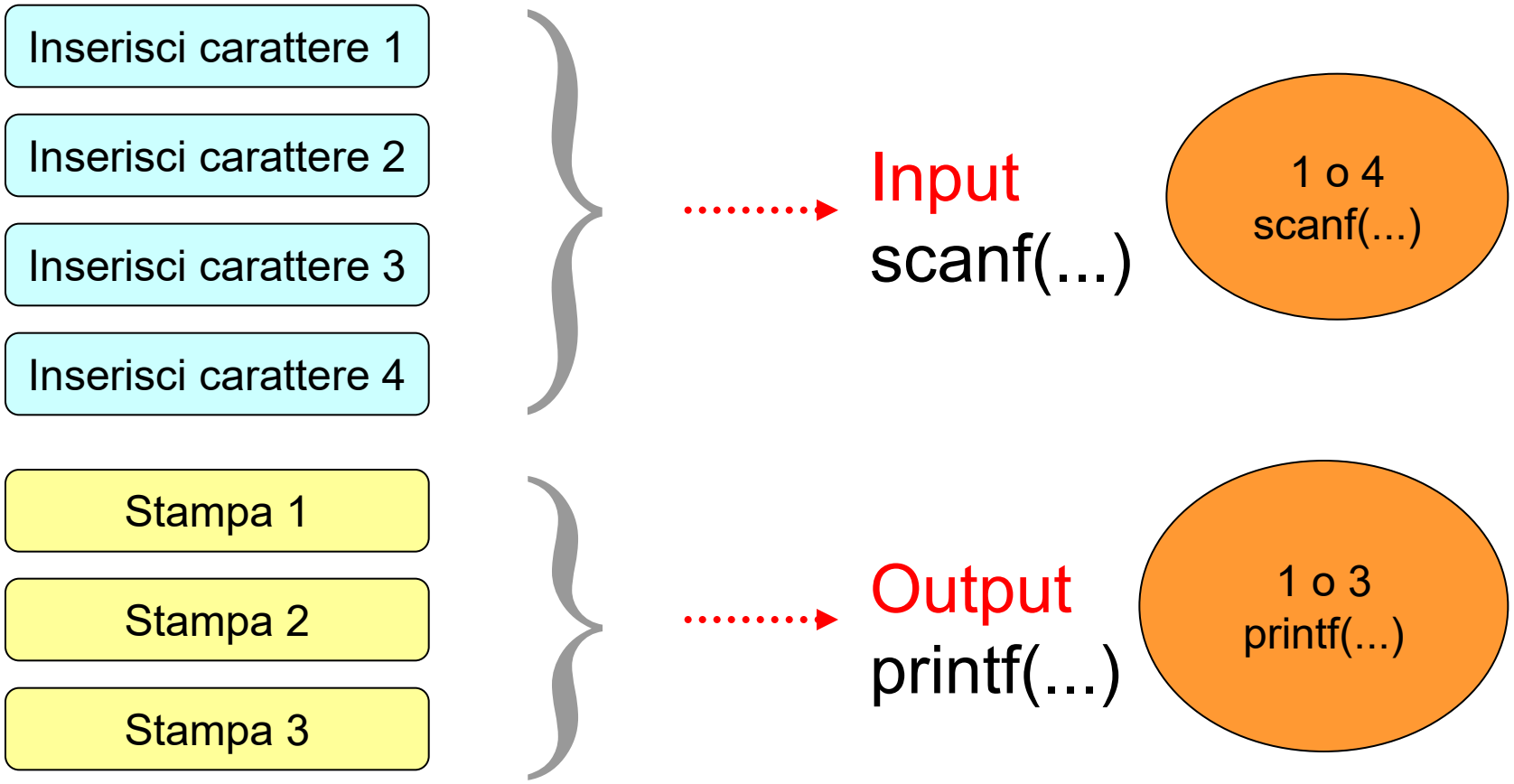
- Esempio di esecuzione del programma

```
c1= c
c2= 1
c3= x
c4= 5
```

Esercizio 6 – output

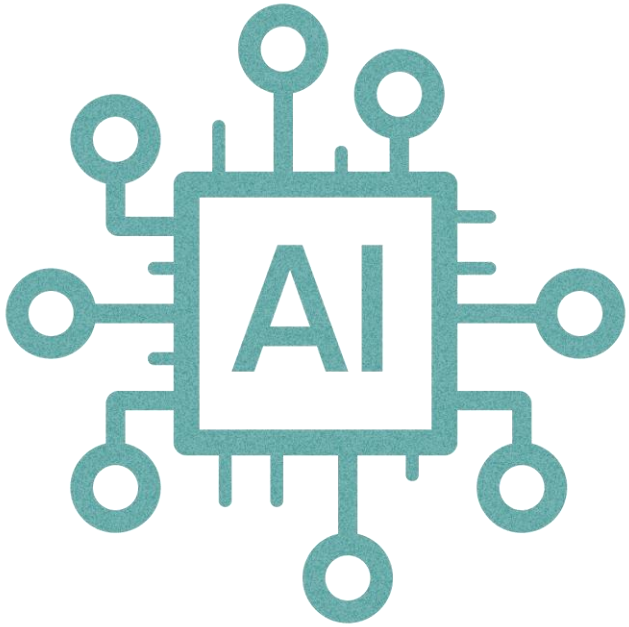
```
Output 1:
c
1
x
5
Output 2:
c1x5
Output 3:
c  1  x  5
```

Esercizio 6 note



Quiz per verifica competenze

Quiz generati automaticamente usando l'applicazione:
Generative AI 4 Education
e supervisionati dal docente.



Il quiz è anonimo.

Bisogna essere autenticati nel dominio UNITN!

Non sono esempi di domande di esame!

https://forms.gle/VKwyDM32h3NBNx3a8