



Università degli Studi di Cagliari
Corso di Laurea in Ingegneria Biomedica

ELEMENTI DI INFORMATICA

https://www.unica.it/unica/page/it/gianluca_marcialis

A.A. 2021/2022

Docente: **Gian Luca Marcialis**

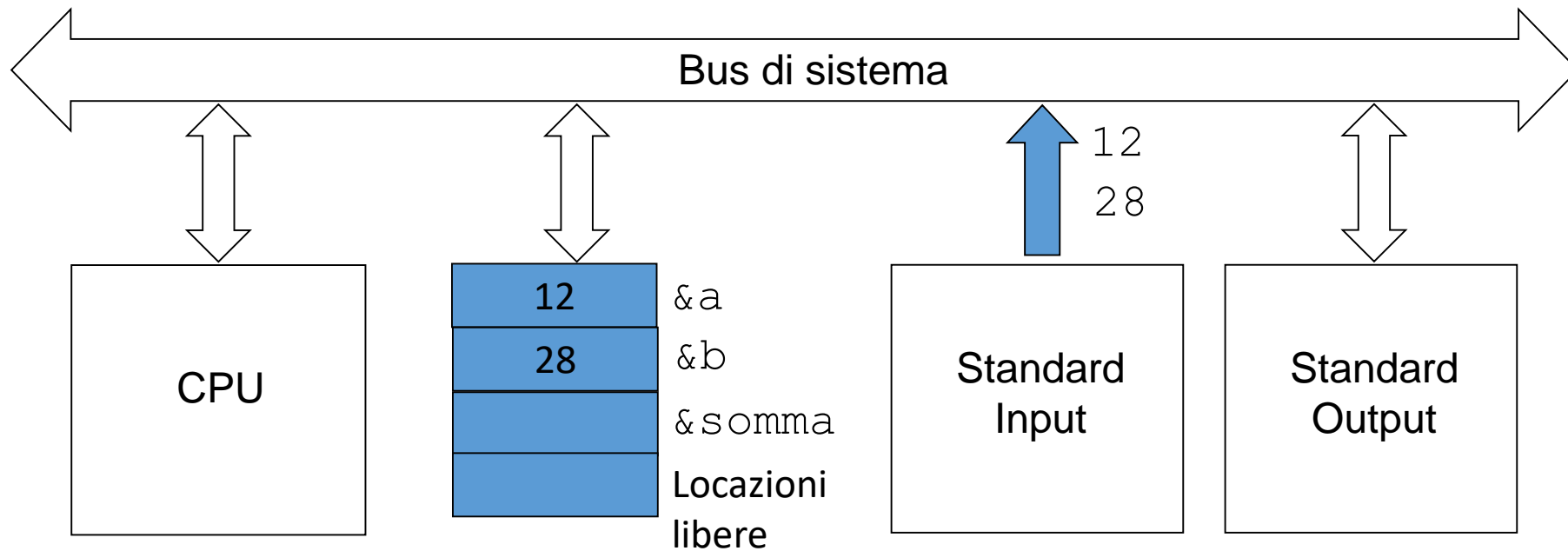
LINGUAGGIO C
Gestione dei file

Sommario

- Scrivere e leggere da qualunque I/O
- Il tipo `FILE`
 - Funzioni `fscanf` e `fprintf`
 - EOF
- Funzioni `sscanf` e `sprintf`

Passaggio da standard I/O

- Lo stato della “macchina virtuale C” dopo la lettura di due valori numerici dalla tastiera



Il tipo `FILE`

- E' un particolare tipo, che serve ad identificare appunto I/O diverso da "standard I/O"
- Una variabile di tipo `FILE` identifica univocamente una "lavagna" ("stream") nella quale si possono scrivere informazioni o cercarne ("leggerne"), se presenti
- Le funzioni per gestire i file sono presenti nella libreria `<stdio.h>`
- Per definire una variabile di tipo `FILE`:

```
FILE *fp;
```

Funzioni per gestire i file

- `fp = fopen(char* NomeFile, char* Modalità)`
 - “Apri” lo “stream” identificato da `fp` (assegnandogli un valore)
 - `NomeFile` è una stringa con il nome del file. Es. `NomeFile="pippo.txt"`
 - `Modalità` è una stringa di uno o due caratteri:
 - “w” → se il file è aperto in scrittura
 - “r” → se è aperto in lettura
 - “a” → se è aperto in modalità “append”, ovvero si vogliono aggiungere ai contenuti presenti nuovi contenuti scritti “in fondo” al file
 - L’uscita di `fopen()` è una costante di valore `NULL` se il file aperto in lettura/append non esiste oppure c’è un errore nel nome del file.
- `int fclose(FILE *fp)`
 - E’ la funzione che chiude il file dopo che su esso sono state compiute le operazioni desiderate, restituendo un valore diverso da 0 se l’operazione è andata a buon fine
 - Un file va **sempre** chiuso al termine del suo utilizzo

Scrittura/lettura su file

- `fprintf(FILE *fp, char *StringaDaScrivere, parametri)`
 - E' l'analogo di `printf` per i file, si usa esattamente nella stessa maniera
 - **Es.** `fprintf(fp, "L'area del cerchio è pari a %f.\n", area);`
 - Scrive la stringa tra virgolette sul file aperto identificato da `fp`
- `fscanf(FILE *fp, char* EspressioneDaLeggere, parametri)`
 - Analogo di `scanf` per i file
 - **Es.** `fscanf(fp, "%d %d", &a, &b);`
 - Si aspetta di leggere due valori interi da file e li memorizza nelle variabili `a` e `b`

Esercizio

- Scrivere una funzione C `scriviVettore`, che, ricevendo in ingresso una stringa `NomeFile` e un vettore di N interi v , scriva il contenuto su un file di nome indicato da `NomeFile`, andando a capo per ogni valore scritto
- La funzione deve restituire il valore 1 se l'operazione è andata a buon fine, 0 altrimenti

Soluzione

```
int scriviVettore(char* NomeFile, int* vettore, int N)
{
    FILE *fp;
    int i;

    fp=fopen(NomeFile,"w");
    if (fp==NULL) return 0;

    for (i=0; i<N; i++)
        fprintf(fp,"%d\n",v[i]);
    fclose(fp);

    return 1;
}
```


Gestione degli errori nei file

- **int** `fEOF(FILE *fp)`
 - Controlla nella struttura di descrizione dello stato del file cui fa riferimento `fp` se è stata raggiunta la fine del file nella precedente operazione di lettura o scrittura
 - Restituisce 0 se la condizione di fine file non è stata raggiunta, un valore diverso da 0 in caso contrario

Esercizio

- Scrivere una funzione che legga una sequenza di interi da file «input.txt» e lo memorizzi in un vettore di N elementi utilizzato come parametro. Restituisca il numero di elementi effettivamente letti.
 - Leggere da file una sequenza di interi, separati da spazi o “a capo”, e memorizzarla in un vettore.
 - Si stampi a video eventualmente un avviso se la lunghezza massima del vettore è stata raggiunta interrompendo l’acquisizione di valori
- Scrivere una funzione che stampi a video il vettore acquisito
- Scrivere la funzione «main» in cui si acquisisca da file il cui nome è fornito da tastiera un certo numero di interi e lo stampi a video

Soluzione senza funzioni

```
/*Programma che legge max 50 interi da file e li memorizza in un vettore*/
#include <stdio.h>
int main()
{
    int i, j, vettore[50];
    char nomefile[100];
    FILE *fp;

    printf("Inserire il nome del file:\n");
    scanf("%s", nomefile);

    fp=fopen(nomefile, "r");
    i=0;
    while ((!feof(fp)) && (i<50)) /*finché il file non è finito e i<50*/
    {
        fscanf(fp, "%d", &vettore[i]);
        i=i+1; /* oppure i++; */
    }
    fclose(fp);

    for(j=0; j<i; j++)
        printf("\nv[%d] = %d", vettore[j]);

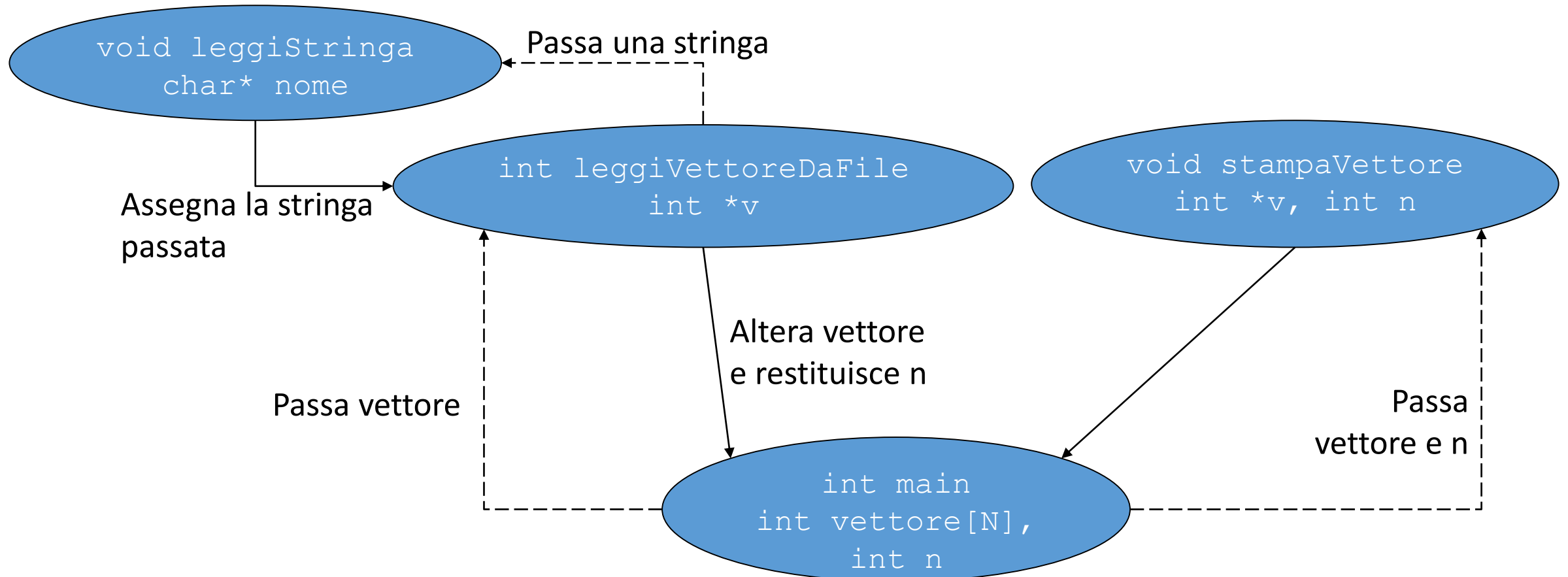
    return 0;
}
```

Lettura del nome del file

Ciclo di lettura dei dati
dal file e
memorizzazione dei
dati nel vettore

Ciclo di stampa dei dati
memorizzati nel vettore

Soluzione con funzioni: visione bottom-up



Funzioni di lettura

```
int leggiVettoreDaFile(int* v)
{
    int i;
    FILE *f;
    char nomefile[50];

    leggiStringa(nomefile);

    f=fopen(nomefile, "r");
    for(i=0; (i<N) && (!feof(f)); i++)
        fscanf(f, "%d", &v[i]);
    fclose(f);
    return i;
}
```

```
void leggiStringa(char* s)
{
    scanf("%s", &s[0]);
    return;
}
```

Funzione di stampa

```
void stampaVettore(int* v, int n)
{
    int i;

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

Funzione main

```
int main()  
{  
    int vettore[N], n;  
  
    n=leggiVettoreDaFile(vettore);  
    stampaVettore(vettore, n);  
  
    return 0;  
}
```

Programma completo

```
/*Programma che legge max 50 interi da file e li memorizza in un vettore*/
#include <stdio.h>
#define N 100

void leggiStringa(char* s);
int leggiVettoreDaFile(int* v);          /* Prototipi di funzione */
void stampaVettore(int *v, int n);

int main
{
    .... /*corpo della main*/
}

void leggiStringa(char* s)
{ ... /* corpo di leggiStringa*/}
int leggiVettoreDaFile(int *v)
{ ... /* corpo di leggiVettoreDaFile*/}
void stampaVettore(int* v, int n)
{ ... /* corpo di stampaVettore*/}
```


Esercizio

- Scrivere un programma C che legga da file «dati.txt» un numero imprecisato di triple di valori reali e verifichi quante di esse possono rappresentare un triangolo generico, e quante cateti ed ipotenusa di un triangolo rettangolo. Stampi su file «esito.txt» il numero di triangoli in generale sono stati rilevati ed il numero di triangoli rettangoli in particolare.
- Si realizzi la soluzione in forma modulare. Scrivere in particolare:
 - Una funzione `assegna_massimo` che, dati tre valori reali, faccia sì che all'uscita della funzione il primo contenga il massimo dei tre, e gli altri due siano riassegnati di conseguenza.
 - Una funzione `verifica_triangolo` che, ricevendo tre valori reali, essendo il primo il massimo dei tre, restituisca 1 se essi rispettano la disuguaglianza triangolare, 0 altrimenti.
 - Una funzione `verifica_rettangolo` che, ricevendo tre valori reali, essendo il primo il massimo dei tre, restituisca 1 se essi possono rappresentare ipotenusa e cateti di un triangolo rettangolo, 0 altrimenti.
 - Una funzione `salva` che, ricevendo in ingresso due interi, li memorizzi su file «esito.txt».

Appendice: scrittura e lettura su stringhe – analogia coi file

- Funzioni per manipolare stringhe della libreria `<string.h>`
- `sprintf(char*StringaDiScrittura, char*StringaFormattata, parametri);`
 - Analoga a `printf` e `fprintf`, solo che scrive nella stringa indicata da `StringaDiScrittura`, finché c'è spazio
 - Per esempio provare la sequenza di istruzioni:

```
sprintf(frase,"Il perimetro del rettangolo è %f.\n",perimetro);
printf(frase);
```
- `sscanf(char*StringaDiLettura, char*StringaFormattata, parametri);`

Per saperne di più

- Ceri, Mandriola, Sbattella, *Informatica – arte e mestiere*, Capp.3-4, McGraw-Hill
- Kernighan, Ritchie, *Il linguaggio C*, Cap. 1, Pearson-Prentice Hall