UNIVERSITY OF TRENTO - Italy

Laboratorio 14

Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2024/2025
Università degli Studi di Trento

Funzioni per gestione file

- fopen
  FILE * fopen (const char *filename, const char *mode);
- fclose
  int fclose (FILE* stream);
- feof
  int feof (FILE* stream);

- fprintf
  int fprintf (FILE* stream, const char *format, ...);
- fscanf
  int fscanf (FILE* stream, const char *format, ...);

- fwrite
  size_t fwrite (const void *ptr, size_t size, size_t count, FILE* stream);
- fread
  size_t fread (void *ptr, size_t size, size_t count, FILE* stream);

fopen: parametro "mode" – 1/2

- Testo | Binario
  - r | rb: read
  - w | wb: write
    - Se il file non esiste: lo crea
    - Se il file esiste: cancella eventuale contenuto
  - a | ab: append
    - Se il file non esiste: lo crea
    - Operazioni di riposizione indice (esempio: fseek) non funzionano

fopen: parametro "mode" – 2/2

- Testo | Binario
  - r+ | rb+: read e write
    - Il file deve esistere
  - w+ | wb+: write e read
    - Se il file non esiste: lo crea
    - Se il file esiste: cancella eventuale contenuto
  - a+ | ab+: append e read
    - Se il file non esiste: lo crea
    - Operazioni di riposizione indice (esempio: fseek) agiscono sono su streaming di input
- **NON** usare operatori con **+**
  - Problemi in riposizionamento indice
  - Necessarie operazioni di fflush tra operazioni di scrittura e lettura

Gestione file

- **Formato testo**
- Lettura
  - Apertura stream
    - fopen(), mode: **r**
  - Controllo apertura corretta
  - Lettura
    - **fscanf**()
  - Chiusura stream
    - fclose()
- Scrittura
  - Apertura stream
    - fopen(), mode: **w** | **a**
  - Controllo apertura corretta
  - Scrittura
    - **fprintf**()
  - Chiusura stream
    - fclose()

- **Formato binario**
- Lettura
  - Apertura stream
    - fopen(), mode: **rb**
  - Controllo apertura corretta
  - Lettura
    - **fread**()
  - Chiusura stream
    - fclose()
- Scrittura
  - Apertura stream
    - fopen(), mode: **wb** | **ab**
  - Controllo apertura corretta
  - Scrittura
    - **fwrite**()
  - Chiusura stream
    - fclose()

Esercizio 1 – variabile vs file

Scrittura su file e lettura da file di una singola variabile di tipo int
Il file è chiamato «`esempio1.txt`»
Definire una variabile: `var`
Inizializzarla con il valore: `12`

Scrittura su file – testo

```
int var = 12;
FILE *f_txt;
// formato TESTO
// scrittura valore singolo
f_txt = fopen("esempio1.txt", "w");
if(f_txt==NULL){
  printf("Errore apertura file. Variabile w");
  return 1;
}
fprintf(f_txt, "%d", var);
fclose(f_txt);
```

Un FILE è una struttura che contiene:

- Campo per Modalità di utilizzo:
  - Lettura, scrittura o lettura e scrittura;
- Un campo per la Posizione corrente:
  - Punta al prossimo byte da leggere o scrivere sul file;
- Un campo indicatore di errore (per lettura e/o scrittura)
- Un campo contenente un indicatore di end-of-file (eof).

Lettura da file – testo

```
int var_letta;
FILE *f_txt;
// formato TESTO
// lettura valore singolo
f_txt = fopen("esempio1.txt", "r");
if(f_txt==NULL){
  printf("Errore apertura file. Variabile r");
  return 1;
}
fscanf(f_txt, "%d", &var_letta);
printf("valore letto: %d\n", var_letta);
fclose(f_txt);
```

Scrittura su file – binario

```
int var = 25;
FILE *f_bin;
// formato BINARIO
// scrittura valore singolo
f_bin = fopen("esempio1bin.txt", "wb");
if(f_bin==NULL){
  printf("Errore apertura file. Variabile wb");
  return 1;
}
fwrite(&var, sizeof(var), 1, f_bin);
//fwrite(&var, sizeof(int), 1, f_bin);
fclose(f_bin);
```

Lettura da file – binario

```
int var2_letta;
FILE *f_bin;
// formato BINARIO
// lettura valore singolo
f_bin = fopen("esempio1bin.txt", "rb");
if(f_bin==NULL){
  printf("Errore apertura file. Variabile rb");
  return 1;
}
fread(&var2_letta, sizeof(var2_letta), 1, f_bin);
//fread(&var2_letta, sizeof(int), 1, f_bin);
printf("valore letto binario: %d\n", var2_letta);
fclose(f_bin);
```

Esercizio 2 – array vs file

Scrittura su file e lettura da file di un
array di tipo int: `vet`
Il file è chiamato «`esempio2array.txt`»

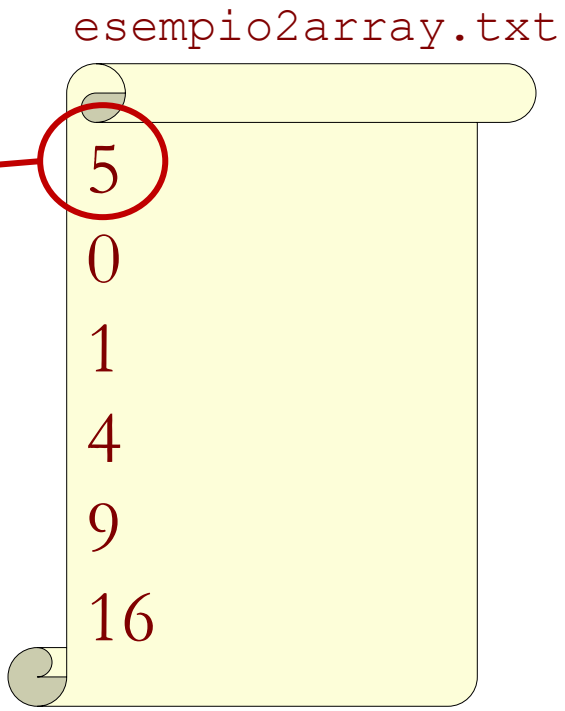
Definire una variabile: `vet`
Inizializzarlo con i valori: `{0,1,4,9,16}`

Scrittura su file – testo – V1a

```
#define N 5
int vet[N] = {0,1,4,9,16};
FILE *f_txt;
int i;
// formato TESTO
// scrittura array
// prima riga: numero elementi
// righe successive: elementi
f_txt = fopen("esempio2array.txt", "w");
if(f_txt==NULL){
  printf("Errore apertura file. Array w");
  return 1;
}
fprintf(f_txt, "%d\n", N);
for(i=0 ; i< N ; i++){
  fprintf(f_txt, "%d\n", vet[i]);
}
fclose(f_txt);
```

Lettura da file – testo – V1b

```
#define N 5
int vet_letto[N]; int i, j; int dim;
FILE *f_txt;
f_txt = fopen("esempio2array.txt", "r");
if(f_txt==NULL){
  printf("Errore apertura file. Array r");
  return 1;
}
fscanf(f_txt, "%d", &dim);
j=0;
// opzione 1 - conteggio
while( j<dim ){
  fscanf(f_txt, "%d", &vet_letto[j]);
  j++;
}
printf("valori letti: ");
for(i=0 ; i<dim ; i++){
  printf("%d ", vet_letto[i]);
}
printf("\n");
fclose(f_txt);
```



Lettura da file – testo – V1c

```
#define N 5
int vet_letto[N]; int i, j; int dim;
FILE *f_txt;
f_txt = fopen("esempio2array.txt", "r");
if(f_txt==NULL){
  printf("Errore apertura file. Array r");
  return 1;
}
fscanf(f_txt, "%d", &dim);
j=0;
// opzione 2 - utilizzo feof
while( !feof(f_txt) ){
  fscanf(f_txt, "%d", &vet_letto[j]);
  j++;
}
printf("valori letti: ");
for(i=0 ; i<dim ; i++){ // for(i=0 ; i<j ; i++)
  printf("%d ", vet_letto[i]);
}
printf("\n");
fclose(f_txt);
```

Non necessario avere nel file tale info!!!

```
int feof( FILE* stream );
```
return 0 se si raggiunge end-of-file

Lettura da file – testo – V1d

```
#define N 5
int vet_letto[N]; int i, j; int dim;
FILE *f_txt;
// lettura array - opzione 2
f_txt = fopen("esempio2array.txt", "r");
if(f_txt==NULL){
  printf("Errore apertura file. Array r");
  return 1;
}
fscanf(f_txt, "%d", &dim);
j=0;
// opzione 3 -- utilizzo valore return scanf
while( fscanf(f_txt, "%d", &vet_letto2[j])==1 ){
  j++;
}
printf("valori letti (2): ");
for(i=0 ; i<dim ; i++){ // for(i=0 ; i<j ; i++)
  printf("%d ", vet_letto2[i]);
}
printf("\n");
fclose(f_txt);
```

```
int scanf( ... );
      return numero di elementi letti
```

Scrittura su file – testo – V2a

```
#define N 5
int vet[N] = {0,1,4,9,16};
FILE *f_txt;
int i;
// formato TESTO
// scrittura array
// NO numero elementi
// un elemento per ogni riga
f_txt = fopen("esempio2array.txt", "w");
if(f_txt==NULL){
  printf("Errore apertura file. Array w");
  return 1;
}
for(i=0 ; i< N ; i++){
  fprintf(f_txt, "%d\n", vet[i]);
}
fclose(f_txt);
```

Lettura da file – testo – V2b

```
#define N 5
int vet_letto[5]; int i; int dim;
FILE *f_txt;
// lettura array - opzione 1
f_txt = fopen("esempio2array.txt", "r");
if(f_txt==NULL){
  printf("Errore apertura file. Array r");
  return 1;
}
dim=0;
// opzione 1 - utilizzo feof
while( !feof(f_txt) ){
  fscanf(f_txt, "%d", &vet_letto[dim]);
  if( !feof(f_txt) ){ dim++; }
}
printf("valori letti: ");
for(i=0 ; i<dim ; i++){
  printf("%d ", vet_letto[i]);
}
printf("\n");
fclose(f_txt);
```

controllo necessario per eventuali righe vuote a fine file

Lettura da file – testo – V2c

```
int vet_letto[5]; int i; int dim;
FILE *f_txt;
// lettura array - opzione 2
f_txt = fopen("esempio2array.txt", "r");
if(f_txt==NULL){
  printf("Errore apertura file. Array r");
  return 1;
}
dim=0;
// opzione 2 -- utilizzo valore return scanf
while( fscanf(f_txt, "%d", &vet_letto2[dim])==1 ){
  dim++;
}
printf("valori letti (2): ");
for(i=0 ; i<dim ; i++){
  printf("%d ", vet_letto2[i]);
}
printf("\n");
fclose(f_txt);
```

Nessun controllo aggiuntivo necessario
Eventuali righe vuote: non si legge la quantità prevista di dati

Scrittura su file – binario

```
int vet2[5] = {10,20,30,40,50};
FILE *f_bin;
// formato BINARIO
// scrittura array
f_bin = fopen("esempio2arraybin.txt", "wb");
if(f_bin==NULL){
  printf("Errore apertura file. Array wb");
  return 1;
}
//vet2 contiene indirizzo array
fwrite(vet2, sizeof(int), 5, f_bin);
fclose(f_bin);
```

Lettura da file – binario

```
int vet2_letto[5];
FILE *f_bin;
// lettura arrray
f_bin = fopen("esempio2arraybin.txt", "rb");
if(f_bin==NULL){
  printf("Errore apertura file. Array rb 1");
  return 1;
}
// numero elementi noto a priori
fread(vet2_letto, sizeof(int), 5, f_bin);
printf("valori letti binari (1): ");
for(i=0 ; i<5 ; i++){
  printf("%d ", vet2_letto[i]);
}
printf("\n");
fclose(f_bin);
```

Lettura da file – binario

```
int vet2_letto[5];
FILE *f_bin;
// lettura arrray
f_bin = fopen("esempio2arraybin.txt", "rb");
if(f_bin==NULL){
  printf("Errore apertura file. Array rb 2");
  return 1;
}
// numero elementi non noto
dim=0;
while( !feof(f_bin) ){
  fread(vet3_letto[dim], sizeof(int), 1, f_bin);
  if( !feof(f_bin) )
    dim++;
}
printf("valori letti binari (2): ");
for(i=0 ; i<dim ; i++){
  printf("%d ", vet3_letto[i]);
}
printf("\n");
fclose(f_bin);
```

Esercizio 3

- Scrittura su file e lettura da file di Array di tipo int: `vet`
- Utilizzo di funzioni
  - Esempi con file di tipo **testo**
- Opzione1
  - Apertura e chiusura file IN funzione
  - `int scriviFile(int v[], int dim, char* nomefile);`
  - `int leggiFile(int v[], int* dim, char* nomefile);`
- Opzione 2
  - Apertura e chiusura file FUORI dalla funzione
  - `int scriviFile_v2(int v[], int dim, FILE* f);`
  - `int leggiFile_v2(int v[], int* dim, FILE* f);`

Opzione 1 – param=nomeFile

```
#define N 5
int main(int argc, char *argv[]){
   int vet[N] = {0,1,4,9,16};
   // Opzione 1 - apertura e chiusura file in funzione
   char nome_file_1[30] = "esempio3.txt";
   int vet_letto[N];
   int dim;
   int i, ret;
   ret = scriviFile(vet, N, nome_file_1);
   if(ret==0){ printf("errore scrittura file (1)\n");}
   ret = leggiFile(vet_letto, &dim, nome_file_1);
   if (ret==0){
     printf("errore lettura file (1)\n");
   }
   else{
     printf("Lettura 1\n");
     for(i=0 ;i<dim ; i++) printf("%d ", vet_letto[i]);
     printf("\n");
   }
   return 0;
}
```

Opzione 1 – scriviFile

```
int scriviFile(int v[], int dim, char* nomefile){
   FILE *f;
   int i;
   f = fopen(nomefile, "w");
   if(f==NULL){
     printf("Errore apertura file. Array w");
     return 0;
   }
   for(i=0 ; i<dim ; i++){
     fprintf(f, "%d\n", v[i]);
   }
   fclose(f);
   return 1;
}
```

Opzione 1 – leggiFile

```
int leggiFile(int v[], int *dim, char* nomefile){
   FILE* f;
   f = fopen(nomefile, "r");
   if(f==NULL){
     printf("Errore apertura file. Array r");
     return 0;
   }
   (*dim)=0;  //dim viene modificato dalla funzione
   // opzione 1 -- utilizzo feof
   // numero dati non noto
   while( !feof(f) ){
     fscanf(f, "%d", &v[(*dim)]);
     if( !feof(f) ) (*dim)++;
   }
   fclose(f);
   return 1;
}
```

Opzione 2 – param FILE*

```
#define N 5
int main(int argc, char *argv[]){
   // Opzione 2 - apertura e chiusura file FUORI funzione
   char nome_file_2[30] = "esempio3_2.txt";
   int vet2[N] = {10,20,30,40,50}; int vet2_letto[N], dim, i;
   FILE *ff;
   if( (ff=fopen(nome_file_2, "w"))==NULL){
     printf("errore scrittura file (2)\n");
   }else{
     scriviFile_v2(vet2, N, ff);
     fclose(ff);
   }
   if( (ff=fopen(nome_file_2, "r"))==NULL){
     printf("errore lettura file (2)\n");
   }else{
     leggiFile_v2(vet2_letto, &dim, ff);
     fclose(ff);
     for(i=0 ;i<dim ; i++) { printf("%d ", vet_letto[i]);}
   }
   return 0;
}
```

Opzione 2 – scriviFile leggiFile

```
int scriviFile_v2(int v[], int dim, FILE* f){
   int i;
   for(i=0 ; i<dim ; i++){
     fprintf(f, "%d\n", v[i]);
   }
   return 1;
}

int leggiFile_v2(int v[], int* dim, FILE* f){
   (*dim)=0;
   // opzione 1 -- utilizzo feof
   while( !feof(f) ){
     fscanf(f, "%d", &v[(*dim)]);
     if( !feof(f) ) (*dim)++;
   }
   return 1;
}
```

Esercizio 4

```
typedef struct Forza{
  float m, a;
} Forza;
```

- Creare variabile **f** array 3 elementi di tipo **Forza**
- Tramite la funzione InitArrayForze inizializzare f

  (m input da tastiera e a random 0..359)
- Scrittura del contenuto di **f** sul file **dati.txt** nella forma

```
23.000 110
12.600 1
4.600 164
```

- Lettura dal file **dati.txt** e stampa a video del contenuto

Esercizio 5

```
typedef struct Forza{
  char label[10]; //max 9 caratteri
  float x, y, modulo;
} Forza;
```

- Creare una variabile **f** di tipo **Forza**
- Leggere, salvando i dati in **f**, il contenuto del file **dati.txt** che contiene i seguenti valori :

```
Forza1 (1.1,1.8)=12.4
Forza2 (2.5,2.9)=5.2
Forza3 (1.9,3.2)=8.9
```

- Ogni valore letto dovrà essere salvato nel file **risultati.txt** con il modulo moltiplicato per 2 con l'istruzione **fwrite**
- Lettura del contenuto del file **risultati.txt** tramite l'istruzione **fread** con stampa a video nella forma:

```
Forza1 (1.0;1.0) 24.80
Forza2 (2.0;2.0) 10.40
Forza3 (1.0;3.0) 17.80
```

Esercizio 6

```
typedef struct Tforza{
  char label[10];
  float x, y, modulo;
} Tforza;
```

```
typedef struct Tcollezione{
  Tforza v[30];
  int n;
} Tcollezione;
```

- Creare una variabile **col** di tipo **Tcollezione**
- Leggere, salvando i dati in **col**, il contenuto del file **dati.txt** che contiene i seguenti valori :

```
Forza1 (1.1,1.8)=12.4
Forza2 (2.5,2.9)=5.2
Forza3 (1.9,3.2)=8.9
```

- OPZIONE1: Scrittura di **col** in file **datibin.txt** in formato binario
- Lettura del contenuto del file **datibin.txt** con Stampa a video (controllo)
- OPZIONE2: Scrittura dell'array **col.v** in file **datibin2.txt** in formato binario
- Lettura del contenuto del file **datibin2.txt** con Stampa a video (controllo)

Esercizio 6 – scrittura fwrite

Scrittura modalità binaria

- Opzione 1
  - scrivere il contenuto di coll

```
fwrite (&col, sizeof(Tcollezione), 1, pf);
```

- Opzione 2
  - scrivere l'array coll.elementi

```
fwrite(col.elementi, sizeof(Tforza), col.n, pf);
```

Esercizio 6 – lettura fread

Lettura modalità binaria

- Opzione scrittura 1
  - Lettura dell'intera collezione

```
fread (&col, sizeof(Tcollezione), 1, pf);
```

- Opzione scrittura 2
  - Numero elementi noti a priori

```
fread (col.elementi, sizeof(Tforza), col.n, pf);
```

  - Numero elementi **non** noti a priori
    - Iterazioni fino a che non si raggiunge fine del file

```
fread (col.elementi[i], sizeof(Tforza), 1, pf);
```

Esercizio 7

- Scrivere un programma che
  - Legge da tastiera un numero x (inserito dall'utente)
  - Calcola il fattoriale del numero x e stampa a video il risultato. Utilizzare il seguente algoritmo per il calcolo del fattoriale

```
ris=1;
for(i=1 ; i<=x, i++){
    ris *= i;
}
```

  - Per ogni iterazione dell'algoritmo precedente, salvare su file fattoriale.txt i valori intermedi del contenuto della variabile ris
    - Formato file: testo

- In aggiunta
  - Riscrivere il programma precedente utilizzando la funzione `int fatt(int n, FILE* pf);`
    - Scrive sul file i valori intermedi
    - Restituisce il fattoriale di n

Esercizio 8

- Usare progetto sorgente cartella «E14.9 Esercizio8_Agenda_iniziale»
- Lettura e scrittura su file
  - scrittura su file (al termine del programma)

```
int scriviFile(Tagenda pa, char *nomefile);
```

  - lettura file (per inizializzazione agenda)

```
int leggiFile(Tagenda *pa, char *nomefile);
```

  - scrittura su file - **binario**

```
int scriviFileBin(Tagenda pa, char *nomefile);
```

  - lettura file - **binario**

```
int leggiFileBin(Tagenda *pa, char *nomefile);
```

Formato file testo

- Opzione formato 1

```
3
10 6 2010 10 30 10 6 2010 11 30 Piscina
11 6 2010 14 30 11 6 2010 16 0 Studio
11 7 2010 16 0 11 6 2010 16 30 Appuntamento
```

Formato file testo

- Opzione formato 2

```
3
10 6 2010
10 30
10 6 2010
11 30
Piscina
11 6 2010
14 30
11 6 2010
16 0
Studio
11 7 2010
16 0
11 6 2010
16 30
Appuntamento
```

Struttura file utilizzata

Funzione scriviFile

```
int scriviFile(Tagenda pa, char *nomefile);
```

- Algoritmo
  - Scrivere prima riga
    - numero di elementi
  - Per le righe successive
    - scrivere uno alla volta tutti gli eventi presenti nell'agenda
- Return
  - 1 ok
  - 0 errore

Funzione leggiFile

```
int leggiFile(Tagenda *pa, char *nomefile);
```

- Algoritmo
  - Leggere prima riga
    - numero di elementi
  - Per le righe successive
    - leggere uno alla volta tutti gli eventi presenti nell'agenda
- Return
  - 1 ok
  - 0 errore
- Nota
  - Congruenza con la struttura utilizzata in fase di scrittura

Funzione scriviFileBin

```
int scriviFileBin(Tagenda pa, char* nomefile_agenda)
```

- File contiene l'agenda
  - Scrittura del contenuto della variabile di tipo Tagenda
- Return
  - 1 ok
  - 0 errore

Funzione leggiFileBin

```
int leggiFileBin(Tagenda *pa, char *nomefile);
```

- Procedura
  - Leggere una struttura di tipo Tagenda
  - Salvare i dati in variabile pa

Scrittura file binario - alternativa

```
int scriviFileBinV2(Tagenda pa, char* nomefile_eventi)
```

- File contiene gli eventi
  - Scrittura del contenuto del campo eventi (array di tipo Tevento)
- Return
  - 1 ok oppure 0 errore

```
int leggiFileBinV2(Tagenda *pa, char *nomefile);
```

- Procedura
  - Fino a che non si raggiunge la fine del file
    - leggere elemento di tipo Tevento
  - Salvare i dati in varibile pa
    - E aggiornare opportunamente tutti il dato del numero di elementi
- Return
  - 1 ok oppure 0 errore