More on Files

Esercizi sulla lettura di file di testo

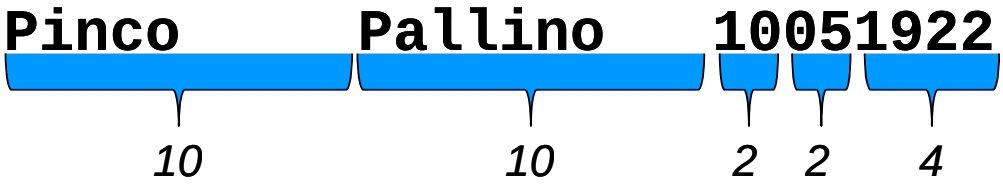
Strutturazione dei file di testo

- Le strutture possono essere le più disparate
- …in fondo è “solo” un problema algoritmico

- Abbiamo già tutti i mattoni, occorre metterli insieme in modo sensato!

Lettura di campi a dim. fissa

- Si usa **`fscanf()`** per “mangiare” il numero di caratteri corretto
- Si usa **`sscanf()`** per convertire (eventualmente) in tipi diversi dalla stringa



Lettura di campi a dim. fissa

```
#define NOME_LEN 21
#define PERSONA_ARRAY_DIM 20
typedef struct
{
   int giorno;
   int mese;
   int anno;
} Data;

typedef struct
{
   char nome[NOME_LEN];
   char cognome[NOME_LEN];
   Data nascita;
} Persona;
```

Lettura di campi a dim. fissa

```
void readPersona(FILE *f, Persona *pers)
{
   char giorno[3], mese[3], anno[5];
   fscanf(f, "%10c%10c", pers->cognome, pers->nome);
   pers->cognome[10] = '\0';
   pers->nome[10] = '\0';

   fscanf(f, "%2c%2c%4c", giorno, mese, anno);
   giorno[2] = '\0';
   mese[2] = '\0';
   anno[4] = '\0';

   sscanf(giorno, "%d", &pers->nascita.giorno);
   sscanf(mese, "%d", &pers->nascita.mese);
   sscanf(anno, "%d", &pers->nascita.anno);
}
```

Servono i terminatori

Servono i terminatori

Lettura di campi a dim. fissa

E il controllo d'errore!?

- Verificare che il valore di ritorno di **`fscanf`** sia quello che ci si attende (numero di parametri correttamente convertiti ed assegnati)
- Restituire un'indicazione del successo della lettura
- Il prototipo diventa:

```
Boolean readLine(FILE *f, Persona *pers);
```

Linea con num. fisso di campi

- Separazione tramite opportuno carattere di separazione
- È il caso più ricorrente
- Si tratta solo di capire come effettuare la lettura di ogni campo

Esempio

Un'agenzia di navigazione desidera informatizzare la gestione dei viaggi delle proprie navi merci. In un file di testo (Ships.txt) sono programmati i viaggi e le relative navi. Ogni riga del file comprende, nell'ordine:

- ***nome della nave*** (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- ***capacità della nave*** in tonnellate (un intero) + un carattere ';' di separazione
- ***nome del porto di partenza*** (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- ***data di partenza*** nel formato gg/mm/aaaa + un carattere 'T' di separazione
- ***orario di partenza*** (ora locale del porto di partenza con offset rispetto al GMT – vedere file Ships.txt per esempi) + un carattere ';' di separazione
- ***nome del porto di arrivo*** (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- ***data di arrivo***, come sopra
- ***orario di arrivo***, come sopra

Esempio – file Ships.txt

```
Nina;100;Ravenna;28/02/2008T02:30+1;Taranto;01/03/2008T10:15+1
Pinta;80;Ravenna;07/10/2008T14:30+1;New York;23/10/2008T20:15-6
…
```

Consigli?

- Usare **`readField()`** (in una delle sue varianti) dove necessario
- Usare **`fscanf()`** dove possibile

readField (v.2)

Versione che restituisce il carattere di separazione: più flessibile

```
char readField(char buffer[], char sep, FILE *f)
{
    int i = 0;
    char ch = fgetc(f);
    while (ch != sep && ch != 10 && ch != EOF)
    {
        buffer[i++] = ch;
        ch = fgetc(f);
    }
    buffer[i] = '\0';
    return ch;
}
```

Esempio – DateTime

```
#ifndef TIME
#define TIME

typedef struct DateTimeStruct
{
   int day, month, year;
   int hour, minute, gmtOffset;
} DateTime;

#endif
```


Esempio – Ship

```
#ifndef SHIP
#define SHIP

typedef char ShipName[17];
typedef char PortName[17];
typedef struct ShipStruct
{
   ShipName name;
   int capacity;
   PortName departurePort;
   DateTime departureTime;
   PortName arrivalPort;
   DateTime arrivalTime;
} Ship;

#endif
```

Esempio – readShipFromTxt (1)

```
Boolean readShipFromTxt(FILE *f, Ship *ship)
{
   Boolean ok;
   int capacity;
   int day, month, year, hour, minute, gmtOffset;
   ok = readField(ship->name, ';', f) == ';';
   ok = ok && fscanf(f, "%d;", &capacity) == 1;
   if (ok)
   {
       ship->capacity = capacity;
   }

…
```


Esempio – readShipFromTxt (2)

```
…

   ok = ok && readField(ship->departurePort, ';', f) == ';';
   ok = ok && fscanf(f, "%d/%d/%dT%d:%d%d;", &day, &month,
   &year, &hour, &minute, &gmtOffset) == 6;
   ok = ok && fillDateTime(day, month, year, hour, minute,
       gmtOffset, &ship->departureTime);

   ok = ok && readField(ship->arrivalPort, ';', f) == ';';
   ok = ok && fscanf(f, "%d/%d/%dT%d:%d%d", &day, &month,
   &year, &hour, &minute, &gmtOffset) == 6;
   fgetc(f);  //Mangia il terminatore di linea (qualunque
              //esso sia: \n o EOF)
   ok = ok && fillDateTime(day, month, year, hour, minute,
       gmtOffset, &ship->arrivalTime);
   return ok;
}
```

Esempio – fillDateTime

- Che cosa fa

```
Boolean fillDateTime(int day, int month, int year,
          int hour, int minute, int gmtOffset,
          DateTime *time) ?
```

- Verifica che i dati in ingresso siano sensati
  - Se lo sono li assegna alla struttura e restituisce true
  - Altrimenti restituisce false
- …un ottimo esercizio per CASA!
- …attenzione agli anni **bisestili**

Lettura di una entità

- In generale, tutte le funzioni di lettura di una singola entità possono essere ricondotte alla forma:

```
Boolean readEntity…(FILE *f, Entity *e);
```

- In questo modo la lettura di un insieme di entità dello stesso tipo diventa piuttosto standard… e banale

Lettura di un insieme di entità

- Si suppongano definiti:
  - tipo **`Entity`**
  - funzione di lettura per **`Entity`** (vedi slide precedente)
- A prescindere dal tipo di contenitore:
  - Leggere un'entità tramite l'apposita funzione di lettura
  - Se la lettura ha avuto successo, inserire l'entità nel contenitore, altrimenti terminare
  - Eventualmente (se significativo) restituire il numero di entità lette o il contenitore o entrambi
- Per come sono state costruite le cose, la funzione di lettura di un insieme di entità varia più al variare del contenitore che al variare dell'entità
- La variazione dell'entità è stata incapsulata nella funzione di lettura dell'entità stessa

Lettura di un insieme di entità - array

```
int readEntityArray(char fileName[], EntityArray ea)
{
  int i = 0;
  FILE *entityFile = fopen(fileName, "r");
  if (entityFile != NULL)
  {
      for (i = 0; readEntity(entityFile, ea[i]); i++);
      fclose(entityFile);
      return i;
  }
  else
  {
      return 0;
  }
}
```

Lettura di un insieme di entità - array

La funzione di lettura di cui alla slide precedente può notevolmente variare a seconda delle “condizioni al contorno”:

- Allocazione dinamica dell’array – quando deve essere grande l’array allocato?
- Array di puntatori ad entità – come va allocata/creata l’entità?
- …e se il contenitore non è un array?

Altri esempi?

- Più avanti – esercizi su compiti d’esame!