AA 2021-22

FONDAMENTI DI INFORMATICA

L A B O R A T O R I O

[prof.ssa RAFFAELA MIRANDOLA]

POLITECNICO
MILANO 1863

Informazioni Logistiche

LABS:

Online Giovedi 9:15-12:15

GIO 14/10/21	Lab1	9:15-12:15	AULA VIRTUALE
GIO 28/10/21	Lab2	" "	" "
GIO 18/11/21	Lab3	" "	" "
GIO 02/12/21	Lab4	" "	" "
GIO 16/12/21	Lab5	" "	" "

Tutors:

sofia.gautieri@mail.polimi.it
agnese.taluzzi@mail.polimi.it
pietro1.lodi@mail.polimi.it

AULA VIRTUALE:
https://politecnicomilano.webex.com/meet/gianenrico.conti

contatti | mail | link

Responsabili di Laboratorio

Ing. *Gian Enrico* Conti gianenrico.conti@polimi.it

Sito web del corso

https://webeep.polimi.it/course/view.php?id=1128

Sottoprogrammi e File

Esercizio 4.1 "spedizioni"

Si considerino le seguenti strutture dati che rappresentano delle scatole da spedire. Si noti che una scatola può contenere un numero variabile di oggetti e per ogni oggetto si specifica il peso per unità ed il numero di unità contenute nella scatola.

```
#define N_OGGETTI 100
#define N_SCATOLE 10
#define MAX_STR 100
#define PESO_CONTAINER 100

typedef struct{
  char descrizione[MAX_STR+1];
  float peso_per_unita;
  int quanti;
  int codice;
} t_oggetto;

typedef struct{
  t_oggetto oggetti[N_OGGETTI];
  int n_oggetti;
  char descrizione[MAX_STR+1];
  int codice;
} t_scatola;
```

Esercizio 4.1 b "spedizioni”

a) Realizzare un sottoprogramma C che riceve come parametri un array di scatole sc, la sua dimensione dim ed un valore in virgola mobile che rappresenta un peso. Analizzando le scatole nell'ordine in cui si trovano nell'array, il sottoprogramma valuta e restituisce il numero di scatole che può essere immagazzinato in un container avente una capienza pari al peso ricevuto come ultimo parametro.

b) Scrivere un programma C che chiede all'utente i dati contenuti in dieci scatole; per ciascuna scatola il programma chiede prima quanti oggetti sono contenuti e poi i dati di ciascun oggetto. In seguito il programma invoca la funzione sopra definita specificando il valore 100.0 come capienza del container e visualizza il risultato a video.

Esercizio 4.1 soluzione (tipi/proto)

```c
#include <stdio.h>
#include <string.h>

#define N_OGGETTI 100
#define N_SCATOLE 10
#define MAX_STR 100
#define PESO_CONTAINER 100

typedef struct{
  char descrizione[MAX_STR+1];
  float peso_per_unita;
  int quanti;
  int codice;
} t_oggetto;

typedef struct{
  t_oggetto oggetti[N_OGGETTI];
  int n_oggetti;
  char descrizione[MAX_STR+1];
  int codice;
} t_scatola;

int analizza(t_scatola sc[], int dim, float capienza);
```

Esercizio 4.1 soluzione

```c
int main(){
  t_scatola sc[N_SCATOLE];
  int i, ris, j;

  for(i=0; i<N_SCATOLE; i++){
    printf("SCATOLA %d - INS COD, DESCR, #OGGETTI: ", i);
    scanf("%d %s %d", &sc[i].codice, sc[i].descrizione, &sc[i].n_oggetti);
    for(j=0; j < sc[i].n_oggetti; j++){
      printf("  OGGETTO %d - INS COD, DESCR, PESO UNITA', QUANTI: ", j);
      scanf("%d %s %f %d", &sc[i].oggetti[j].codice, sc[i].oggetti[j].descrizione,
                           &sc[i].oggetti[j].peso_per_unita, &sc[i].oggetti[j].quanti);
    }
  }
  ris = analizza(sc, N_SCATOLE, PESO_CONTAINER);
  printf("%d\n", ris);
  return 0;
}


int analizza(t_scatola sc[], int dim, float capienza){
  int i, j;
  for(i=0; i<dim && capienza>0; i++){
    for(j=0; j < sc[i].n_oggetti; j++)
      capienza = capienza - sc[i].oggetti[j].quanti * sc[i].oggetti[j].peso_per_unita;
  }
  if(capienza<0)
    i--;
  return i;
}
```

Esercizio 4.2 “lettura file”

Scrivere un programma che apre un file di testo il cui nome è TEST.txt contenente un sequenza di lunghezza indefinita (0 o più) di numeri interi.

Il programma calcola e stampa a video il valore massimo e quante volte tale valore si è presentato nel file.

Esercizio 4.2 soluzione

```c
#include <stdio.h>
#define FN "TEST.txt"

int main( ){
  FILE *fp;
  int max, n, t;

  fp = fopen(FN, "r");
  if(fp!=NULL) {
    fscanf(fp, "%d ", &n);
    if(!feof(fp)){
      max = n;
      t = 1;
      fscanf(fp, "%d", &n);
      while (!feof(fp)) {
        if(n > max){
          max = n;
          t = 1;
        } else if(n == max)
          t++;
        fscanf(fp, "%d", &n);
      }
      printf("max %d compare %d volte\n", max, t);
    } else
      printf("Sequenza vuota\n");
    fclose(fp);
  }
  else
    printf("Errore apertura file\n");
  return 0;
}
```

Esercizio 4.3 “sottoprogrammi e file”

a) Scrivere una funzione **analizzastringa** (ed il relativo prototipo) che riceve come parametro una stringa. La funzione:
- restituisce 1 se la stringa è lunga più di 5 caratteri e contiene almeno 2 cifre numeriche ed un carattere che non sia una cifra;
- in caso negativo la funzione restituisce 0.

b) Scrivere un programma che apre in lettura un file di testo testo.txt contenente una serie di lunghezza indefinita di parole ciascuna lunga al massimo 15 caratteri ed in scrittura un secondo file risultato.txt.

Il programma legge ciascuna parola contenuta in testo.txt e la scrive in risultato.txt soltanto se l'invocazione della funzione **analizzastringa** su di essa porta ad un risultato positivo (la funzione restituisce 1).

(Segue...)

Esercizio 4.3 “sottoprogrammi e file”: esempio

Ad esempio, se il file contiene il seguente testo:

Ciao45 baubau 2345678 esame2020 trallallero1 bimbumbam345 a234

Il programma salverà in risultato.txt il seguente testo:

Ciao45
esame2020
bimbumbam345

Esercizio 4.3 soluzioni

```c
#include<stdio.h>

#define MAXS 15
#define FILENAMEIN "testo.txt"
#define FILENAMEOUT "risultato.txt"

int analizzastringa(char str[]);

int main(){
  FILE* fpi, *fpo;
  char str[MAXS+1];

  fpi = fopen(FILENAMEIN, "r");
  if(fpi){
    fpo = fopen(FILENAMEOUT, "w");
    if(fpo){
      fscanf(fpi, "%s", str);
      while(!feof(fpi)){
        if(analizzastringa(str))
          fprintf(fpo, "%s\n", str);
        fscanf(fpi, "%s", str);
      }
      fclose(fpo);
    } else
      printf("Errore out\n");
    fclose(fpi);
  } else
    printf("Errore in\n");
  return 0;
}

int analizzastringa(char str[]) {
  int numCifre, len;

  for(numCifre=0, len=0; str[len]!='\0'; len++)
    if(str[len] >= '0' && str[len] <= '9')
      numCifre++;
  return numCifre >= 2 && len > numCifre && len > 5;
}
```

Esercizio 4.4 “convertinumero”

a) Scrivere un sottoprogramma **convertinumero** che riceve come parametro una stringa str e due parametri interi num e valido passati per indirizzo in cui salvare i risultati. Se str è composta soltanto da cifre (char da ‘0’ a ‘9’), il sottoprogramma converte la stringa in un numero, la salva in num ed assegna 1 al parametro valido. Altrimenti il sottoprogramma assegna 0 a valido e non modifica num.
NOTA: è vietato l’utilizzo di funzioni di conversione quali atoi o sscanf.

b) Scrivere un programma che apre un file di testo testo.txt contenente una serie di lunghezza indefinita di parole ciascuna lunga al massimo 15 caratteri. Il programma legge ciascuna parola contenuta nel file e, mediante il sottoprogramma **convertinumero**, verifica se si tratta di una stringa composta solo da cifre e in caso affermativo la converte in un numero. Il programma somma tutti i numeri identificati e presenta a video il risultato finale.

Ad esempio, se il file contiene il seguente testo:
Ciao domani 120 10 alfa23 esame 5 informatica 34tre -34
Il programma stamperà a video il seguente messaggio:
135

Esercizio 4.4 soluzione

```c
#include <stdio.h>

#define MAXS 15
#define FILENAME "testo.txt"

void convertinumero(char str[], int *ris, int *valido);

int main(){
  FILE* fp;
  char str[MAXS+1];
  int somma, n, v;

  fp = fopen(FILENAME, "r");
  if(fp){
    somma = 0;
    fscanf(fp, "%s", str);
    while(!feof(fp)){
      convertinumero(str, &n, &v);
      if(v)
        somma = somma + n;
      fscanf(fp, "%s", str);
    }
    printf("%d\n", somma);
    fclose(fp);
  } else
    printf("Errore\n");
  return 0;
}

void convertinumero(char str[], int *ris, int *valido){
  int pot10, i;
  *valido=1;
  for(i=0; str[i]!='\0'; i++);

  for(i--, *valido=1, *ris=0, pot10=1; i>=0 && *valido; i--, pot10=pot10*10){
    if(str[i]>='0' && str[i]<='9'){
      *ris = *ris + pot10 * (str[i]-'0');
    } else
      *valido=0;
  }
}
```

Esercizio 4.5 "Liste"

Data le definizione di lista vista a lezione, e le funzioni definite,

a) scrivere un sottoprogramma **listaMedie**() che riceve in ingresso una lista **l1** di numeri naturali.
Il sottoprogramma costruisce una nuova lista **l2** in cui ogni elemento è calcolato come la **media** tra un valore di **l1** ed il suo successivo.
L'ultimo elemento della lista, per cui non esiste un elemento successivo, viene ricopiato nella nuova lista.
Infine il sottoprogramma restituisce la nuova lista.

Esempio: l1 = 8 5 16 7 -> 6 10 11 7

b) scrivere un sottoprogramma che salva una lista di interi in un file binario il cui nome è ricevuto come parametro
c) scrivere un sottoprogramma che carica una lista di interi di dimensione indefinita da un file binario il cui nome è ricevuto come parametro
d) scrivere un sottoprogramma ricorsivo per eseguire l'inserimento in ordine crescente

Esercizio 4.5 (parte 1 defs e prototipi)

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct nodo{
  int dato;
  struct nodo *next;
} nodo_t;

typedef nodo_t* Ptr_nodo;

Ptr_nodo inserisciInCoda(Ptr_nodo, int);
Ptr_nodo inserisciInTesta(Ptr_nodo, int);
Ptr_nodo rimuovi(Ptr_nodo, int);
Ptr_nodo rimuoviTutti(Ptr_nodo, int);
void visualizza(Ptr_nodo);
Ptr_nodo distruggiLista(Ptr_nodo);
int esisteElemento(Ptr_nodo, int);
```

Esercizio 4.5 (parte 2)

```
Ptr_nodo inserisciInCoda(Ptr_nodo lista, int num){
  Ptr_nodo prec;
  Ptr_nodo tmp;
  tmp = (Ptr_nodo) malloc(sizeof(nodo_t));
  if(tmp != NULL){
    tmp->next = NULL;
    tmp->dato = num;
    if(lista == NULL)
      lista = tmp;
    else{
      for(prec=lista;prec->next!=NULL;prec=prec->next);
      prec->next = tmp;
    }
  } else
      printf("Memoria esaurita!\n");
  return lista;
}

Ptr_nodo inserisciInTesta(Ptr_nodo lista, int num){
  Ptr_nodo tmp;
  tmp = (Ptr_nodo) malloc(sizeof(nodo_t));
  if(tmp != NULL){
    tmp->dato = num;
    tmp->next = lista;
    lista = tmp;
  } else
      printf("Memoria esaurita!\n");
  return lista;
}
```

Esercizio 4.5 (parte 3)

```c
void visualizza(Ptr_nodo lista){
  printf("Elementi nella lista: ");
  while(lista != NULL){
    printf("%d ", lista->dato);
    lista = lista->next;
  }
  printf("--|\n");
}

Ptr_nodo distruggiLista(Ptr_nodo lista){
  Ptr_nodo tmp;
  while(lista!= NULL){
    tmp = lista;
    lista = lista->next;
    free(tmp);
  }
  return NULL;
}

int esisteElemento(Ptr_nodo lista, int num){
  for(;lista && lista->dato!=num;lista=lista->next);
  return (lista != NULL);
}
```

Esercizio 4.5 (parte 4)

```
Ptr_nodo rimuovi(Ptr_nodo lista, int num){
  Ptr_nodo curr, prec, canc;
  int found;
  found=0;
  curr = lista;
  prec = NULL;
  while(curr && !found){
    if(curr->dato==num){
      found=1;
      canc = curr;
      curr = curr->next;
      if(prec!=NULL)
        prec->next = curr;
      else
        lista = curr;
      free(canc);
    }
    else{
      prec=curr;
      curr = curr->next;
    }
  }
  return lista;
}

Ptr_nodo rimuoviTutti(Ptr_nodo lista, int num){
  Ptr_nodo curr, prec, canc;
  curr = lista;
  prec = NULL;
  while(curr){
    if(curr->dato==num){
      canc = curr;
      curr = curr->next;
      if(prec!=NULL)
        prec->next = curr;
      else
        lista = curr;
      free(canc);
    }
    else{
      prec=curr;
      curr = curr->next;
    }
  }
  return lista;
}
```

Esercizio 4.5 (parte 5)

```
Ptr_nodo cancellaCoda(Ptr_nodo l){
 Ptr_nodo tmp, prec;
 if(l!= NULL){
     prec=l;
    if(l->next==NULL){
      free(prec);
      l = NULL;
    } else {
     for(prec=l; prec->next->next!=NULL; prec=prec->next);
      tmp = prec->next;
      prec->next = NULL;
      free(tmp);
    }
  }
  return l;
}
```

Esercizio 4.5 (parte 6: main)

```c
int main(){
  int num;
  Ptr_nodo lista1,lista2, lista3, curr;
  lista1 = NULL;
  lista2 = NULL;
  lista3 = NULL;

  scanf("%d", &num);
  while(num!=0) {
    lista1 = inserisciInCoda(lista1, num);
    scanf("%d", &num);
  }
  scanf("%d", &num);
  while(num!=0) {
    lista2 = inserisciInCoda(lista2, num);
    scanf("%d", &num);
  }

  for(curr=lista1; curr;curr=curr->next)
    if(esisteElemento(lista2,curr->dato) && !esisteElemento(lista3,curr->dato))
      lista3=inserisciInCoda(lista3,curr->dato);

  visualizza(lista3);

  lista1 = distruggiLista(lista1);
  lista2 = distruggiLista(lista2);
  lista3 = distruggiLista(lista3);
  return 0;
}
```