NUMERO: 2434A ANNO: 2019

# A P P U N T I

STUDENTE: Ferrera Alessandra

MATERIA: Temi di Informatica - Esami Svolti + Schemi - Prof. Mezzalama

```
while (scanf("%d", &a))
{ if (a==0)
  continue;
```

# PUNTI CHIAVE DELLA PROGRAMMAZIONE IN C

QUANDO LA SOMMA DI DUE ELEM DI UN VETTORE è UGUALE AD A

```
/*k è l'elemento che considero e al quale voglio sommare i valori nelle altre posizioni. La prima volta k=3, ci sommo 4, poi 5, 6, 7. k<N-1 perchè non ha senso sommare 7 con gli altri, sarebbe una ripetizione*/
cont=0;
for(k=0;k<N-1;k++)
 for(i=k+1;i<N;i++){
  if(v[k]+v[i]==a){
    cont++;
    printf("\n v[k]=%d, v[i]=%d",v[k],v[i]);
  }
}
printf("\ncont= %d",cont);
```

SMETTERE DI LEGGERE VETTORE QUANDO INSERISCO ZERO O CHIEDERE REINSERIMENTO DI UN NUOVO VALORE != 0

```
while(k<M){
  printf("v2[]= ");
  scanf("%d",&v2[k]);
if (v2[k]!=0)
  k++;
/* se voglio smettere di leggere il vettore quando trovo 0
if (v2[k]!=0)
  k++;
else
  k=M;*/
//quindi se inserisco nel v2 lo zero, devo reinserire un nuovo valore per quel k
```

INIZIALIZZAZIONI PARTICOLARI

```
max=INT_MIN;
min=INT_MAX;
```

ITERAZIONI FIBONACCI

```
{int a0,a1,a2,i;
a0=0;
a1=1;
i=0;
for (i=0; i<N; i++)
{  printf("%d\n",a0);
   a2=a1+a0;
   a0=a1;
   a1=a2;
}
```

0, 1, 2, 3, 5, 8

TRIANGOLO

```
for(i=0; i<N; i++){
   for (j=0; j<i; j++){
      printf(" ");
   }for (k=0; k<N-i; k++){
 printf("*");
   }printf("\n");
}
```

```
***
 **
  *
```

CONTORNO QUADRATO CON DIAGONALE

```
//qui stampo prima riga di asterishi
for (i=0;i<N-2;i++){
   printf("*"); //asterisco del bordo sx
   for(k=0;k<i;k++)
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 76

```
for(j=2;(j<v[i])&&(flag==1);j++){ // divisori 2-v[i]
        if (v[i] % j==0)
          flag=0; //quindi non è primo
    } } if (flag==0){
          for( j=i ; j<N-(cont)-1 ; j++ ){ COME CON BUBBLE → J<N-i-1
            v[j]=v[j+1]; //traslo tutto il vettore
di una posizione. Cont inizializzato a 0
}  v[N-(cont)-1]=a; //a= -1
    cont++;
    i--; //resto alla posizione di prima
  }}
```

CONTROLLO SE NON E' PALINDROMA

```
If (v[i] != v[N-1-i] )  //come due freccie stessa
direzione verso opposto →←
```

V1 CONTIENE V2?

```
{int v1[N]={2,3,4,5,2}, v2[M]={3,4};
int i, j, flag;
for( i=0; i < N-M+1; i++ ){ //IMPORTANTE N-M+1
  flag=0;
  for(j=0; (j<M) && (flag==0) ;j++ ){
    if  ( v2[j] != v1[i+j] )
      flag=1;
  }
  if(flag==0)
    printf("trovato in %d", i);
}
```

ORDINAMENTO A BOLLE

```
for (i=0;i<N-1;i++) //su quanti vettori devo  fare
l'operazione
  for(k=0;k<N-i-1;k++) {
    if(v[k]>=v[k+1]){
      temporanea=v[k];
      v[k]=v[k+1];
      v[k+1]=temporanea;  //scambio
    //per ordine decrescente, v[k]<v[k+1] }  }
```

ORDINAMENTO DECRESCENTE

```
for(i=0; i<DIM-1; i++){
  max= v3[ i ];  //E' LA CHIAVE PER ORDINAMENTI
  for ( j=i;  j<DIM;  j++){
    if(v3[j]>=max){
      max=v3[j];
      pos=j;
    } }
  temp=v3[i];
  v3[i]=v3[pos];
  v3[pos]=temp;
}
```

CONSIDERO IL SINGOLO ELEMENTO COME SE FOSSE IL MAX, SCORRO TUTTO IL VETTORE PER CAPIRE DOVE METTERLO

ORDINAMENTO CRESCENTE

```
i=0;
while(i<N){
  printf("v= ");
  scanf("%d",&v[i]);
  printf("\n");
  if ( i != 0 ){//dal secondo valore in poi
    k=i;
    do {
      if ( v[k] <= v[k-1] ) {
        temp=v[k];
        v[k]=v[k-1];
        v[k-1]=temp;
        k--;
```

```
   parole++;}
if((capo==1)&&(c==' ')){
   printf("\n");
   capo=0;
   spazi++;}//se hai due spazi consecutivi vai a
capo
if(c==' '){
  capo=1;//ricorda che hai messo uno spazio
}
if((capo==1)&&(c!=' '))
   capo=0;//ma se dopo uno spazio non ne trovo
un altro, azzero capo
printf("%c",c);
}while(c!='.');

printf("\nparole= %d\nrighe= %d\nlunghezza
media parola= %f", parole, spazi,
(float)lettere/parole);
```

PUNTATORI

```
float statistica(int v[],int dim,int *pmin,int *pmax){
*pmax=v[0]; //non posso scrivere max=0 perchè
max è nel main
*pmin=v[0];
//calcolo max e min
for(i=0;i<dim;i++){
  if(v[i]>=*pmax)
     *pmax=v[i];
  else{
     if(v[i]<=*pmin)
        *pmin=v[i];
  } }
```

MATRICI

DICHIARAZIONE:

```
m[R][C]={{1,2,3,4},{2,2,2,3},{1,1,1,1},{2,2,2,2}};
```

LETTURA.

```
for(i=0;i<R;i++){
     for(k=0;k<C;k++){
        printf("m[%d][%d]=",i,k);
        scanf("%d",&m[i][k]);
     }}
```

SOTTOMATRICI

```
printf("nelle 3x3,avro' piu' pari o dispari?\n\n");
for(j=0;j<R-2;j++){
     for(k=0;k<C-2;k++){
//trovo punto inizio di ciascuna matrice 3x3
contpari=0; contdisp=0;
for(a=j;a<j+3;a++){//mi muovo nella 3x3
  for(b=k;b<k+3;b++){
    { n=m[a][b];
       if(n%2==0)
          contpari++;
       else
          contdisp++;
    } }
//in j e k ho coordinate [j][k] del punto d'inizio 3x3
```

STRINGA CARATTERI. SE MIN->MAIU, SE NUM->*, SE CARATTERE SPECIALE->' '

```
{char s[N+1];
int parole,i,lung;
parole=0;
printf("inserisci stringa\n");
```

CONTROLLI

File open fp==NULL

Argc numero parametri

FILE *fp;

| **DOMANDA 4 (PROGRAMMAZIONE** | |
|---|---|

Laputa è l'isola volante descritta da Jonathan Swift ne *I Viaggi di Gulliver* del 1726. Il re vuole costruire la nuova capitale in un luogo in cui guardando nelle direzioni dei punti cardinali (Nord, Sud, Est e Ovest), si può scorgere il confine dell'isola.

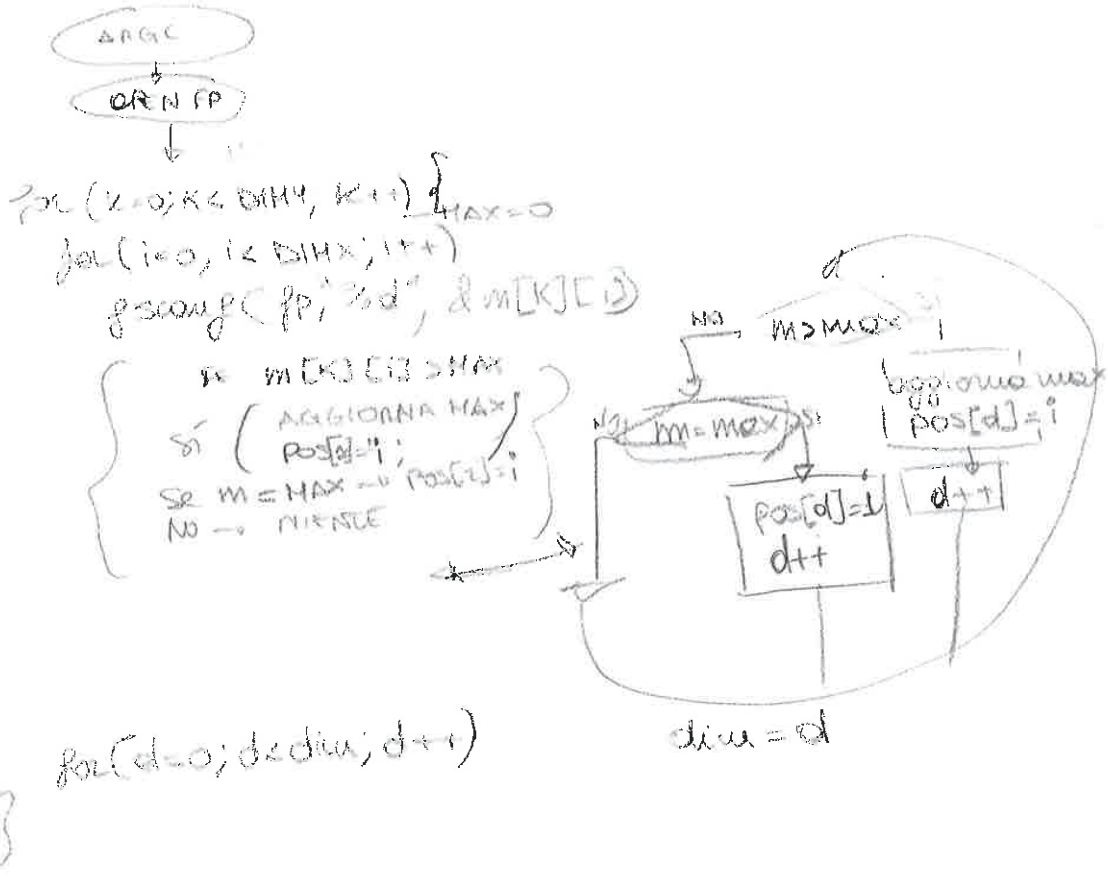
La mappa dell'isola è contenuta in un file. La mappa è una matrice di numeri interi di dimensione **DIMX** per **DIMY** che specifica l'altitudine del territorio nel settore corrispondente. Zero (0) rappresenta la mancanza di territorio. **DIMX** e **DIMY** sono costanti definite attraverso due **#define**.

Il programma riceve il nome del file che descrive la mappa dell'isola come unico argomento sulla linea di comando, e deve identificare le coordinate di tutti i punti in cui è possibile costruire la capitale. Ovvero le coordinate (x, y) di tutti i punti che sono contemporaneamente massimo della riga e massimo della colonna a cui appartengono. Il punto di coordinate (0, 0) è quello in alto a sinistra.

Ad esempio (DIMX=8, DIMY=5), se il file **mappa.dat** contiene:

```
0 0 0 1 1 0 0 0
0 1 4 3 2 2 0 0
1 2 ■ 2 2 4 1 1
1 1 2 2 ■ 2 1 0
0 0 0 2 1 1 0 0
```

```
C:\> esame mappa.dat
La capitale potrebbe essere costruita in (2, 2) (4,3)
```



### #include <stdio.h>

**FILE *fopen(char *filename, char * mode)** – Apertura di un file (mode: "r" lettura – "w" scrittura – "a" append)
**FILE *freopen(char *filename, char * mode, FILE *file_pointer)** - Riassegna un file puntatore ad un file diverso.
**int fclose(FILE *file_pointer)** - Chiude un file
**int feof(FILE *file_pointer)** - Controlla se e' stato incontrato un end-of-file in un file.
**int fflush(FILE *file_pointer)** - Svuota il buffer di un file.
**int getchar(void)** - Legge un carattere da "stdin" (tastiera)
**int fgetc(FILE *file_pointer)** - Prende un carattere da un file
**char *gets(char *buffer)** - Legge una riga da "stdin" (tastiera)
**char *fgets(char *string, int maxchar, FILE *file_pointer)** - Legge una riga da un file.
**int printf(char *format _string, ...)** - Scrive output formattato su "stdout" (schermo)
**int fprintf(FILE *file_pointer, char *format_string, ...)** - Scrive output formattato in un file.
**int sprintf(char *string, char *format_string, ...)** - Scrive output formattato su una stringa
**int fputc(int c, FILE *file_pointer)** - Scrive un carattere in un file
**int putchar(int c)** - Scrive un carattere su "stdout" (schermo)
**int puts(char *string)** - Scrive una stringa su "stdout" (schermo)
**int fputs(char *string, FILE *file_pointer)** - Scrive una stringa in un file.
**int scanf(char *format_string, args)** - Legge input formattato da "stdin" (tastiera)
**int fscanf(FILE *file_pointer, char *format string, args)** - Legge input formattato da file
**int sscanf(char *buffer, char *format_string, args)** - Legge input formattato da una stringa
**EOF** – end of file (costante a valore negativo)
**NULL** - puntatore nullo (valore 0)

### #include <stdlib.h>

**double atof(char *string)** - Converte una stringa in un valore in floating point.
**int atoi(char *string)** - Converte una stringa in un valore integer.
**lnt atol(char *string)** - Converte una stringa in un valore long integer.
**void exit(int val)** – Termina il programma, restituendo il valore 'val'.
**EXIT_FAILURE** - costante per segnalare terminazione senza successo del programma con exit(); valore diverso da zero
**EXIT_SUCCESS** - segnala terminazione con successo del programma con exit(); vale 0

### #include <string.h>

**char *stpcpy (char *dest, char *src)** - Copia una stringa in un'altra. Restituisce dest
**char *strncpy(char *s1, char *s2, size_t n)** - Copia i primi "n" caratteri di s2 in s1. Restituisce s1
**int strcmp(char *s1, char *s2)** - Confronta s1 e s2 per determinare l'ordine alfabetico (<0, s1 prima di s2, 0 uguali, >0 s1 dopo s2)
**int strncmp(char *s1, char *s2, size_t n)** - Confronta i primi "n" caratteri di due stringhe.
**char *strcpy(char *s1, char *s2)** - Copia s2 in s1. Restituisce s1
**int strlen(char *string)** - Determina la lunghezza di una stringa.
**char *strcat(char *s1, char *s2, size_t n)** - Aggiunge s2 a s1. Ritorna s1
**char *strncat(char *s1, char *s2, size_t n)** - Aggiunge "n" caratteri di s2 a s1. Ritorna s1
**char *strchr(char *string, int c)** - Cerca la prima occorrenza del carattere 'c' in string; restituisce un puntatore alla prima occorrenza di c in s, NULL se non presente
**char *strrchr(char *string, int c)** - Cerca l'ultima occorrenza del carattere 'c' in string
**char* strstr(char* s, char* t)** - Restituisce un puntatore alla prima occorrenza di t all'interno di s. Restituisce NULL se t non è presente in s.
**char* strtok(char* s, const char* t)** - scompone s in token, i caratteri che delimitano i token sono contenuti in t. Restituisce il puntatore al token (NULL se non ne trova nessuno). Alla prima chiamata in s va inserita la stringa da scomporre e in t i caratteri che delimitano i vari token. Per operare sulla stessa stringa, alle successive chiamate al posto di s si deve passare NULL

### #include <ctype.h>

**int isalnum(int c)** - Vero se 'c' e' alfanumerico.
**int isalpha(int c)** - Vero se 'c' e' una lettera dell'alfabeto.
**int iscntrl(int c)** - Vero se 'c' e' un carattere di controllo.
**int isdigit(int c)** - Vero se 'c' e' un numero decimale.
**int islower(int c)** - Vero se 'c' e' una lettera minuscola.
**int isprint(int c)** - Vero se 'c' e' un carattere stampabile.
**int ispunct (int c)** - Vero se 'c' e' un carattere di punteggiatura.
**int isspace(int c)** - Vero se 'c' e' un carattere spazio.
**int isupper(int c)** - Vero se 'c' e' una lettera maiuscola.
**tolower(int c)** - Converte 'c' in minuscolo.
**int toupper(int c)** - Converte 'c' in maiuscolo.

### #include <math.h>

**int abs (int n)** – valore assoluto intero
**long labs(long n)** – valore assoluto long
**double fabs (double x )** – valore assoluto di x
**double acos(double x)** - arcocoseno
**double asin(double x)** - arcoseno
**double atan(double x)** - arcotangente
**double atan2(double y, double x)** – arcotangente di y/x.
**double ceil(double x)** – intero superiore a x

Informatica – 04/02/2016 – **durata complessiva: 2h**

| NOME | | COGNOME | |
|---|---|---|---|
| MATRICOLA | | | D1 |

☐AAA-BARC ☐BARD-BOUH ☐BOUI-CART ☐CARU-CONS ☐CARU-CONS CONT-DEMAR ☐DEMAS-FERRD ☐FERRE-GIAQ ☐GIAR-LAEZ ☐LAFA-MANC ☐MAND-MIQZ MIRA-PAHZ ☐PAIA-PODD ☐PODE-ROSSE ☐ROSSF-SIQZ ☐SIRA-TUCB ☐TUCC-ZZZ ☐Poli@Home ☐5 Crediti ☐AAA-LIB/English ☐LIC-ZZZ/English ☐Altro:.....

| **Domanda 1** | *Risultato* |
|---|---|
| Scrivere il valore decimale del seguente numero binario su 6 bit: 110011, interpretato come:<br>• Complemento a 2<br>• Modulo e Segno | CA2:<br><br>MS: |

*Passaggi più significativi per arrivare al risultato*

$CA_2 \rightarrow 110011 \quad 001100 + 000001 = 13 \Rightarrow -13$

MS [redacted] $n = 10011 = 19$

SEGNATO $MS = -19$

Determinare se la seguente uguaglianza Booleana è verificata
(not a) and (not b) and (not c) = not (a or b or c)

$\bar{A} \cdot \bar{B} \cdot \bar{C} = \overline{A+B+C}$ TEOREMA DI DE MORGAN

**Domanda 3**

Spiegare brevemente i diversi tipi di memoria presenti in un calcolatore

MEMORIA PRINCIPALE (RAM)
MEMORIA SECONDARIA (HARD DISK / USB)
REGISTRI IN CPU

**int sscanf(char *buffer, char *format_string, args) -** Legge input formattato da una stringa

**EOF** – end of file (costante a valore negativo)

**NULL** - puntatore nullo (valore 0)

**#include <stdlib.h>**

**double atof(char *string)** - Converte una stringa in un valore in floatingpoint.

**int atoi(char *string)** - Converte una stringa in un valore integer.

**int atol(char *string)** - Converte una stringa in un valore long integer.

**#include <string.h>**

**char *stpcpy (char *dest, char *src) -** Copia una stringa in un'altra. Restituisce dest

**char *strncpy(char *s1, char *s2, size_t n) -** Copia i primi "n" caratteri di s2 in s1. Restituisce s1

**intstrcmp(char *s1, char *s2) -** Confronta s1 e s2 per determinare l'ordine alfabetico (<0, s1 prima di s2, 0 uguali, >0 s1 dopo s2)

**int strncmp(char *s1, char *s2, size_t n) -** Confronta i primi "n" caratteri di due stringhe.

**char *strcpy(char *s1, char *s2) -** Copia s2 in s1. Restituisce s1

**int strlen(char *string) -** Determina la lunghezza di una stringa.

**char *strcat(char *s1, char *s2, size_t n) -** Aggiunge s2 a s1. Ritorna s1

**char *strncat(char *s1, char *s2, size_t n) -** Aggiunge "n" caratteri di s2 a s1. Ritorna s1

**double atan(double x) -** arcotangente

**double atan2(double y, double x) –** arcotangente di y/x.

**double ceil(double x) –** intero sup. a x

**double floor(double x) –** intero inf. a x.

**double cos(double x) –** x in radianti

**double sin(double x) –** x in radianti

**double tan(double x) –** x in radianti

**double cosh(double x) –** cos. perbolico

**double sinh(double x) –** seno iperbolico

**double tanh(double x) –** tang. iperbolica

**double exp(double x) -** $e^x$

**double log(double x)** - log naturale di x

**double log10 (double x ) –** log base 10

**double pow (double x, double y)** - $x^y$

**int rand (void) –** intero casuale tra 0 e RND_MAX.

**int random(int max_num) –** valore casuale tra 0 e max_num.

**double sqrt(double x) –** radice quadrata

**#include <limits.h>**

**INT_MAX** - Indica il più grande valore che è possibile rappresentare con un int.

**INT_MIN** - Indica il più piccolo valore che è possibile rappresentare con un int.

**LONG_MAX** - Indica il più grande valore che è possibile rappresentare con un long.

**LONG_MIN** - Indica il più piccolo valore che è possibile rappresentare con un long.

SIMULAZIONE ESAME DEL 2/09/2013

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

#define MAXU 40
#define MAXD 500

int main(int argc, char *argv[])
{ int i,k,flag,dim;
 char nome[MAXD][MAXU+1], cognome[MAXD][MAXU+1], localita'[MAXD][MAXU+1];
 int distanza[MAXD];
 int velocita'[MAXD], spazio[MAXD];
 char mezzo1[MAXD][MAXU+1], mezzo2[MAXD][MAXU+1];


  FILE *f1;
  FILE *f2;
  FILE *f3;

  if(argc!=3) {
    printf("errore numero parametri linea di comando");
    return -1;
   }
  f1 = fopen(argv[1], "r");
  f2 = fopen("trasporti.txt", "r");
  f3 = fopen(argv[2], "w");

    if((f1==NULL)||(f2==NULL)||(f3==NULL)) {
      printf("errore nell'apertura di un file");
   return -2;
   }
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 76

T PARTE PRATICA: 1h

TEMPO PERMANENTE TEORIA: 1h

[K],

ES 1 siano $\begin{cases} n_1 : 2B \\ n_2 : A2 \end{cases}$ due numeri in esadecimale C2

→ $2B = \underbrace{0010}_{2}\ \underbrace{1011}_{11}$ → è un numero positivo $= 11 + 2^5 = 11 + 32 = 43_{(10)}$

$A2 = \underbrace{1010}_{10}\ \underbrace{0010}_{2}$ → è un numero negativo $= -128 + 32 + 2 = -94_{(10)}$

$$\begin{array}{r} 00101011 + \\ 10100010 = \\ \hline 11001101 \end{array}$$ → non ho overflow, ho segni discordi!

questo è un numero negativo $= -128 + 64 + 8 + 4 + 1 = -51_{(10)}$

in esadecimale, $1100 = 2^3 + 2^2 = 8 + 4 = 12 = C$

$1101 = 13 = D$ → CD

ES 2

Una memoria ha un parallelismo di 8 bit

dimensione Abus per avere 64 KByte memoria?

$$\text{Memoria} = Dbus \cdot 2^{|[Abus]|}$$

$Dbus = 8$ bit

$Mem = 64$ KByte

$1\text{Byte} = 8\text{bit} \rightarrow 1\text{KByte} = 8 \cdot 10^3\text{bit} = 8 \cdot 2^{10}\text{bit}$

$= 64 \cdot 8 \cdot 10^3\text{bit} = 2^6 \cdot 2^3 \cdot 2^{10}\text{bit}$

→ $\text{Memoria} = 2^6 \cdot 2^3 \cdot 2^{10} = 2^3 \cdot 2^{|Abus|}$ →

$6 + 3 + 10 = 3 + X$ → $X = 16 = Abus$

differenze tra memoria centrale e memoria di massa:

La memoria centrale può essere di due tipi: RAM (volatile ma più veloce) e ROM (di sola lettura ma non volatile).

La memoria di massa è costituita da dischi, apparecchiature esterne varie come per esempio quelle costituite da schede magnetiche. È permanente, più lenta ma più economica. La sua dimensione dipende dai bus di I/O, mentre quella della memoria interna dipende dalla capacità dei chip bus interni.

3

PROGRAMMAZIONE

ho una platea di un cinema. E' una matrice con NF (=numero file) e NP (=numero poltrone) → [NF] [NP] con #define

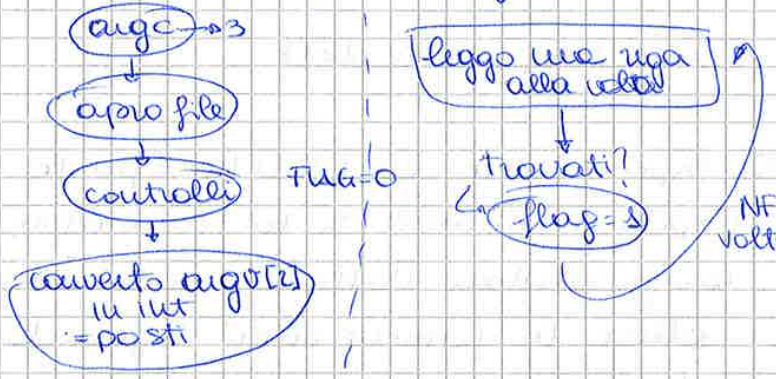
Si scrive il carattere 'o' se il posto è occupato e 'L' se è libero.

leggo da linea di comando il nome del file e il numero di posti adiacenti richiesti. Il programma deve trovare tutti i posti adiacenti nel numero richiesto.

Se scrivo 3 → stampa tutti i possibili posti, dove ho 3 poltrone libere

Se non ho più posti → segnalalo

Mio 21:54




```
#include <stdio.h>
#include <stdlib.h>
#define NF 4
#define NP 6

int main(int argc, char *argv[])
{ FILE *fp;
 int flag, i, k, libero, j;
 char m[NP+1];
 int plibero;


  if(argc!=3){
   printf("errore parametri linea di comando!\n");
      return -1;}
   }

  sscanf(argv[2], "%d", &posti); // è la stessa cosa di posti=atoi(argv[2])
```

SIMULAZIONE DEL 4/02/2013

- Rappresentare in C2 e M&S su 6 bit il numero $(-15)_{10}$

~~CA2 → $-2^6 + 2^5 + 2^4 + 2^0 = 110001$~~

CA2 → su 6 bit quindi $-2^5 = -32+16+1 = -2^5+2^4+2^0 =$ 110001

M&S → su 5 bit → $2^0+2^1+2^2+2^3 =$ 101111

- domanda 2:

int = 16 bit = 2 byte

se ho typedef struct a {

char nome[10]; → 10

char cognome[13]; 13

char ISBN[17]; 17

int copie; → 2

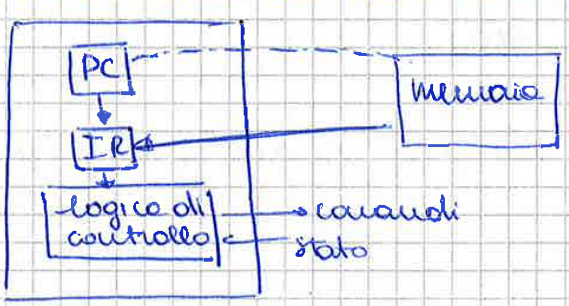
float prezzo; → 4

}

in tutto in memoria avrò uno spazio occupato di 46 byte

- domanda 3:

L'unità di controllo CU è una componente della CPU. Dialoga con la memoria e controlla/coordina le varie "azioni" da eseguire




PC = contiene l'operazione da compiere subito dopo, l'indirizzo in memoria.

IR = registro che contiene temporaneamente l'istruzione da eseguire e la manda alla logica di controllo che riceve informazioni dai flag ed emette comandi



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 76

SIMULAZIONE 13/07/2012 TURNO A ~~h: 16÷15~~ h: 21:50

Studenti preimmatricolati. Nome file da linea di comando

<nome> <cognome> <regione> <data immat> <area> → ingegn I, archit A

- dimensione max stringhe S=30
- non so n° righe file — gg/mm/aaaa
- mese → tra 04 ÷ 09 → 6
- dimensione massima regioni → 20

1) Quanti studenti in I e quanti in A per ogni regione
2) Se non ho immatricolati in quel mese, niente. Altrimenti quanti ne ho
3) da dove ho n° massimo studenti immatricolati

apri file → argc == 2?

```
struct stud {
    char regione[S];
    int ing;
    int arch, anch;
    int tot;
};
```

FLAG 0 → c'è già? Sì → [ing++ / arch++] — FLAG=1 — (-1)
No → inserisci lo; dim++

```
I → studenti[20];
struct date {
    int mm;
    int nuovi;
};  → mesi[6]
```

i=0; K=0; dimmax=0; dimdue=0

fscanf (regione[i] = regione?)

scandisco la data (g,m,a) CON FLAG != 2
se m è presente nello 2° vol
→ nuovi++; FLAG=2
se m non c'è → metto lo nuovi=1 dimdue++

Chiudi
max=0
printf 1 (intanto int tot (1))
se tot > max, aggiorno e conservo in best = i;
printf, (i=0; i<dimdue)

- rappresento in complemento a due e modulo e segno su 5 bit il numero $(-12)_{10}$
  - → M&S = ~~12=1100=2²·3¹~~ = 12 = 8+4 = $2^3+2^2$ = 11100
  - → CA2 = -12 => so che $2^4=16$, quindi $-16+4=-12$ → $-2^4+2^2$ → 10100

- Abus con 32bit. Dbus con 64bit
  memoria fisica = $2^{|Abus|}$ · Dbus = $2^{32}\cdot 64 = 2^{30}\cdot 2^2 \cdot 8$ byte = 32 GB

- Le memorie ROM sono memorie interne ~~caratteristiche~~ non volatili, di tipo lettura (read only memory) e di veloce accesso. Possono essere PROM, non programmabili col esclusivamente leggibili, oppure EEPROM, leggibili ma anche scrivibili a velocità molto basse. La ROM contiene tipicam. progr. d'avvio [BIOS].

11

```
struct first primo[N];
struct secnd secondo[M];
char nome[S+1], cognome[S+1], regione[S+1], data[S+1];
char area;
int flag, dimuno, dimdue, i, best, max;
int g, m, a;

FILE *fp;
if(argc != 2) {
  printf("errore parametri linea di comando!\n");
    return -1;
}
fp = fopen(argv[1], "r");
if (fp == NULL) {
  printf("errore file!\n");
    return -1;
}
  i=0;
  dimuno=0;
  dimdue=0;

while(fscanf(fp, "%s %s %s %s %c", nome, cognome, regione, data, &area)
      != EOF) {

    flag=0;
    for(i=0; (i<dimuno) && (flag==0); i++) {

        if(strcmp(regione, primo[i].reg)==0) {
            if((area=='A')||(area=='a')) {
                primo[i].anch++;
            } else
                primo[i].img++;
            flag=1;
        }
    } if(flag==0) {
        strcpy(primo[dimuno].reg, regione);
        da
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 76

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

#define S 17
#define MAX_CODICI 100
```

SIMULAZIONE DEL 13/07/2012

```
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <limits.h>

#define MAXR 14
#define C 30
# define N 3

int main (int argc, char *argv[])
{ int i, flag, dim, bomber;
  char w[N];
 char nome[MAXR][C], cognome[MAXR][C], ruolo[MAXR][C];
 float Km[MAXR];
 int gol[MAXR], falli[MAXR], subiti[MAXR], passaggi[MAXR], tiri[MAXR];


 FILE *fp;
 strcpy(w, argv[2]);
 if((strcmp(w, "-a")==0)||(strcmp(w, "-b")==0)) {
    if (argc != 3) {
      printf("errore parametri linea comando!\n");
   return -1;}
 }
  else {
  if(argc!=2){
   printf("errore parametri linea comando!\n");
   return -2;
  }
 }
 fp = fopen(argv[1], "r");
   if (fp == NULL) {
     printf("errore apertura file!\n");
  return -3;
  }
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 76

```
        }
            break;

    case 'c':
        boumber = MIN_INT;   → INT_MIN
    for(i=0; (i<MAXR) && (fscanf(fp,"%s %s %s %f %d %d %d %d %d", nome[i],
        cognome[i], ruolo[i], &km[i], &passaggi[i], &tiri[i], &gol[i], &falli[i],
        &subiti[i])!=EOF); i++) {

            if (boumber < gol[i])
                boumber = gol[i]

        }
    dim = i+1;

    for (i=0; i<dim; i++) {
            if (gol[i] == boumber) {
        printf("%s %s %s %d goal\n", nome[i], ~~robot~~ cognome[i], ruolo[i],
            boumber);
        }
    }

            break;
}
    fclose(fp);

    return 0;
}
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 76

ESERCITAZIONE 10

ho un file. Ogni riga [LC]

<partenza> <arrivo> <h.mm> <h:mm>

- leggi da tastiera: stazione partenza e stazione di arrivo e tempo (h). Trovare i primi 4 aerei che partono e arrivano dove ho deciso però hanno un tempo di percorrenza inferiore di quello scelto. Ordinare tali aerei secondo l'ora di partenza e scriverli su un file in uscita [LC]

apro file

controllo su argc != 3

leggo da tastiera

fscanf file != EOF

strcmp partenza, tpartenza

strcmp arrivo, tarrivo

orario < torario

altrimenti niente

chiudi

metto in una matrice con 4 righe e 3 colonne

se struttura vuota → non ho soluzioni

altrimenti riordino

scrivi e chiudi

ATTENZIONE! h:mm → separo le componenti, h*60+mm

```
#define MAX 4
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 20

/* shp=soluzione orario partenza
   sa=soluzione arrivo
   t=tastiera */

int main(int argc, char *argv[])
{ char partenza[N+1], arrivo[N+1], sp[MAX][N+1], sa[MAX][N+1];
  int ahh, amm, ha, hp, phh, pmm;
int torario, i, orario, sha[MAX], shp[MAX], diff;
char tpartenza[N+1], tarrivo[N+1], pTemp[N+1];
int hTemp;

FILE *fp;
if (argc!=3){
 printf("errore parametri linea di comando!\n");
   return -1;
}
fp= fopen(argv[1], "r");
if(fp==NULL){
 printf("errore apertura file!\n");
 return -1;}
```

```
for(i=0; i<dim-1; i++){
    for(k=0; k<dim-i-1; k++){
        if (shp[k] > shp[k+1])
        h.temp = shp[k];
        shp[k] = shp[k+1];
        shp[k+1] = temp;

        htemp = sha[k];
        sha[k] = shp[k+1];
        shp[k+1] = htemp;
```

```
i,famm)!=EOF){ && (i<MAX)){
```

```
ma scanf
```

```
        strcpy(ptemp, partenza[k]);
        strcpy(partenza[k], partenza[k+1]);
        strcpy(partenza[k+1], ptemp);

        strcpy(ptemp, arrivo[k]);
        strcpy(arrivo[k], arrivo[k+1]);
        strcpy(arrivo[k+1], ptemp);
```

avevo messo tutto in una tabella, le stazioni di partenza sono in sp, quelle di arrivo in sa;

```
// ho ricordinato tutta la mia tabella con ordinam. a bolle
    }
  }
  fp = fopen(argv[2], "w");
  if(fp==NULL){
    printf("errore apertura file");
      return -1;
  }
for(i=0; i<dim; i++){
  fprintf(fp," %s %s %d %d\n", sp[i], sa[i], &shp[i], &sha[i]);
}
fclose(fp);
  return 0;
}
```

```
    dim = d;

for(d=0; d<dim; d++) {
    flag=0;

    for(w=0; (w<DIM4) && (flag==0); w++) {
        if (m[w][pos[d]] >= max)   → solo maggiore
            flag=1;
    }
    if (flag==0) {
        printf("la capitale potrebbe essere costruita in (%d;%d)\n", k, pos[d]);
    }
}
}
    fclose(fp);
return 0;
}
```

## GESTIONE DI UN MAGAZZINO

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 1000
#define M 20
int insert_product(char warehouse[][M], float price[],int n, char new_product[],float
price_new_product);
void print_all(char warehouse[][M],float price[],int n,float *avg,float *max);
int update_product(char warehouse[][M],float price[],int n,char product[],float new_price);
int remove_product(char warehouse[][M],float price[],int n,char old_product[]);

int main()
{int scelta,i;
char nome[M],warehouse[N][M];
float prezzo,price[N],media;
float massimo;
//inizializzo il vettore price a -2
for(i=0;i<N;i++)
   price[i]=-2;
for(i=0;i<N;i++){
   strcpy(warehouse[i],"0");

}

do{//iterativo, continua fino a che non scelgo l'opzione 3 per uscire dal programma
do{
   printf("inserisci 1=nuovo prodotto, 2= Stampa listino attuale 3= exit\n");
   printf("4= aggiornamento prezzo 5= rimozione prodotto\n");
scanf("%d",&scelta);
}while((scelta!=1)&&(scelta!=2)&&(scelta!=3)&&(scelta!=4)&&(scelta!=5));
printf("\n");
switch(scelta){
case 1:
   printf("nome del nuovo prodotto: ");
   scanf("%s",nome);//la stringa del nuovo prodotto
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 76

```
    if((price[k]==-2)&&(k!=0)){
      flag=-2;
      dim=k;
    }if(price[0]==-2){
      flag=-2;
      dim=0;
    }
}
if(flag==2){
      printf("magazzino pieno!\n");
      return 2;
    }

for(k=0;k<dim;k++){
  if(strcmp(warehouse[k],new_product)==0){
   printf("prodotto gia' presente in magazzino\n");
   return 0;
  }
}
strcpy(warehouse[dim],new_product);
price[dim]=price_new_product;
if(price[dim]!=-2)
return 1;
else{
   printf("errore nell'inserimento prodotto");
   return -3;
}
}

void print_all(char warehouse[][M],float price[],int n,float *avg,float *max){
int k;
float media=0;
int flag=2,dim=n;
*max=price[0];
*avg=media;
```

```
        flag=-2;
        dim=0;
    }
    }

if (flag==2){
    dim=n;
    flag=-2;
}

for(k=0;(k<dim)&&(flag==-2);k++){
   if(strcmp(product,warehouse[k])==0){
      flag=1;
      price[k]=new_price;
   }
}
if(price[k]==new_price)
   return 0;
else
   return 1;

}

int remove_product(char warehouse[][M],float price[],int n,char old_product[]){
int k,dim,a;
int flag=2;

for(k=0;(k<n)&&(flag==2);k++){
   if((price[k]==-2)&&(k!=0)){
      flag=-2;
      dim=k;
   }if(price[0]==-2){
      flag=-2;
      dim=0;
   }
   }
```

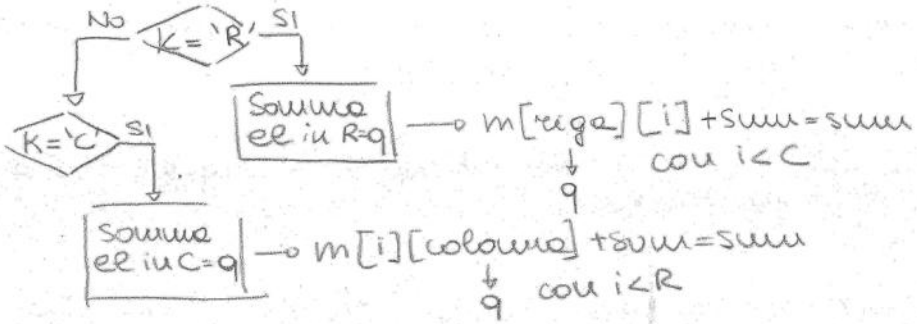
LEZIONE 30

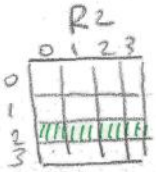
- DA UN NUMERO N SEGUITO DA 'R' O 'C' TRAMITE UNA FUNZIONE SOMMARE EL IN QUELLA RIGA O COLONNA.
( PASSAGGIO DI UNA MATRICE AD UNA FUNZIONE)

'R'
'C'

int funzione (int m[][C], int R, char k, int q)

passaggio matrice  passaggio parametri



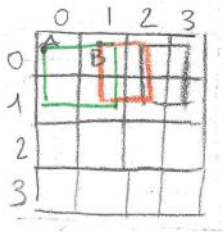




```
# include <stdio.h>
# include <stdlib.h>
# define R 3
# define C 4
int ff (int m [][C], int R, char k, int q);

int main ()
{int m[R][C] = {{1,2,3,4},{3,5,7,9},{1,2,3,4}};
 char rigacolonna;
 int posizione, s;
 printf("inserisci r o c");
 putchar(rigacolonna);
 printf("inserisci posizione);
 scanf("%d", &posizione);
 s= ff (m, R, rigacolonna, posizione);
 printf("somma= %d", s);

  return 0;
  }

 // mi occupo ora della funzione
int ff (int m [][C], int R, char k, int q)
{int i, sum;
 sum=0;
  if (k=='r'){
    for (i=0; i<C; i++){
       sum= sum+m[q][i]; } }
  else for(i=0; i<R; i++)
       sum= sum+m[i][q];
 return sum;
}
```

// oppure
```
posizione...
printf("riga o c?");
scanf("%*c%c", &rigacolonna);
```
↓
Tolgo il carattere prima perché dopo pos prendo invio.

LEZIONE 30

• MATRICE m[R][C]. CONTIENE DEGLI INT. DELLE SOTTOMATRICI 2×2, QUALI HANNO Σ ELEM=0

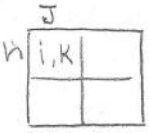



parto dall'elemento nell'angolo delle 2×2.

i=0; i<= R-1

k=0; K< C-1 → PUNTO(i;k) = PUNTO DI PARTENZA SOTTOMATRICE

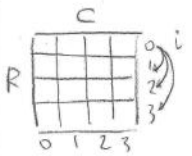
```
#include <stdio.h>
#include <stdlib.h>
#define R 3
#define C 3
int main ()
{ int m[R][C];
  int i, K;
  int Sum;
  int J, h;

for(i=0; i<R; i++){
     for(k=0; k<C; k++)
       {printf("elem[%d][%d]:", i, k);
        scanf("%d",&m[i][j]);

for(i=0; i<R-1; i++){          // punto partenza 2x2
     for(K=0; K<C-1; k++) {
       Sum=0;
       for(J=i; J<i+2; J++){   // posso avere solo 2x2, quindi J<i+2
          for(h=K; h<K+2; h++)
            Sum=Sum+m[J][h];
          }
       if(sum==0)
         printf("somma=0 in m[%d][%d]\n", i, K);
  //mi dice in che punto inizia la matrice 2x2 con somma=0;
   }
   }
  return 0;
   }
```




R=5
C=6
sottom. 3×3
che hanno
n° el pari > n° el dispari

LEZIONE 29

• HO UNA MATRICE. Ci sono almeno 2 righe uguali tra loro? Lo scandisco nell'ordine

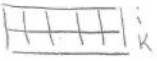



la 4ª riga (i=3) non devo considerarla, è l'ho già confrontata con tutto.

i = riga di riferimento

k = righe con cui confronto i

for(i=0; i<R-1; i++) → 1ª riga, 2ª...




for(K=i+1; K<R; K++)

// Se la 1ª riga = 2ª riga...

j = per scorrere nelle righe lungo le colonne

for(J=0; J<C; J++)

if(m[i][J] != m[K][J])

flag=0; // ho un el. di verso

```
#include <stdio.h>
#include <stdlib.h>
#define R 4
#define C 3

int main()
{int m[R][C]={{2,1,2},{1,2,3},{2,2,2},{3,4,5}};
 int i,K,J;
  int flag,cont=0;
    for(i=0; i<K-1; i++){
        for(K=i+1; K<R; K++){ flag=0;
          for(J=0; J<C; J++){
            if(m[i][J]!=m[K][J])        // scandisce le colonne
                flag=1;

            };
             if(flag==0)
               cont++;
             if((flag==0)&&(cont==2))
             { printf(" ho almeno 2 righe uguali, %d e %d", i,k);
                i=R; }
             }
             }

    return 0;
  }
```

R voglio avere ALMENO due righe uguali. Mi fermo appena ne trovo 2

Test.exe [-a] [-b] <nome file>

ma posso dare

Test.exe <nome>

test.exe [-a] [-b] <nome>  quindi (argc>=2) && (argc<5)

test.exe [-b] [-a] <nome>

test.exe [-a] <nome>

Test.exe [-b] <nome>

per copiare il <nome> in una stringa → strcpy(filename, argv(argc-1))

parto da argv[0]

CONCORSO INTELL.

N giudici, K candidati, 0÷5 voto=capra, 5=genio.

Posso avere $k_{MAX}=100$, $N_{MAX}=10$.

Il n° di giudici e K sono introdotti da linea di comando

→ determinare il k + intelligente e qual è il giudice più severo

```
#include <stdio.h>
#include <stdlib.h>
#define MAXK 100
#define MAXN 10


int main(int argc, char* argv[])
{int N, K; voti [MAXK][MAXN]; totcandidato[MAXK], Totgiudici[MAXN];
  if (argc !=3) {
    printf("n° parametri errato\n utilizzo giusto: nome, N, K");
    exit(1); //o return 1; }
  K = atoi(argv[1]);
  if ((K<=0)||(K>MAXK))
  { printf("n° candidati sbagliato");
    exit(2); }
```

# 12BHD INFORMATICA, A.A. 2017/2018
## Esercitazione di Laboratorio 5

Obiettivi dell'esercitazione
- Scrivere programmi in grado di memorizzare e elaborare molti valori

Contenuti tecnici
- Consolidamento uso dei cicli
- Introduzione all'uso dei vettori

*Da risolvere preferibilmente in laboratorio*

Esercizio 1. Scrivere un programma C che, acquisiti 2 numeri interi positivi ne calcoli il massimo comune divisore utilizzando al formula di Eulero.

Formula di Eulero o metodo dei resti: si procede per divisioni successive del numero maggiore per quello minore, sostituendo ad ogni passo il valore maggiore con il minore ed il minore col resto della divisione. Il processo termina quando il resto è 0.

Esempio: A = 34 , B = 18

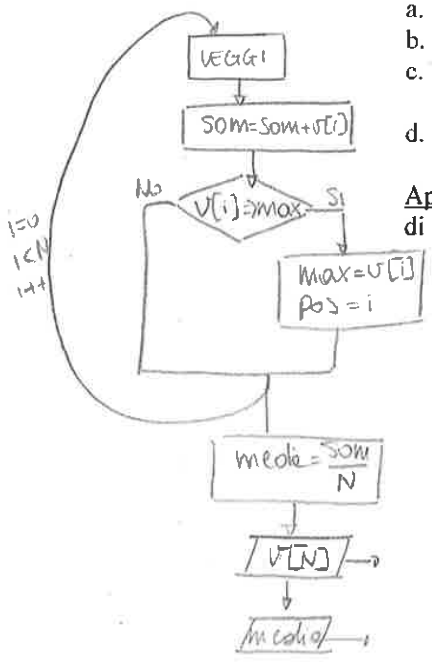
passo 1: 34 % 18 = 16
passo 2: 18 % 16 = 2
passo 3: 16 % 2 = 0 ← stop!

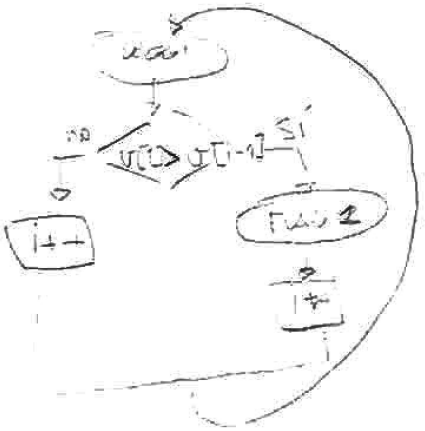
Risultato: MCD = 2

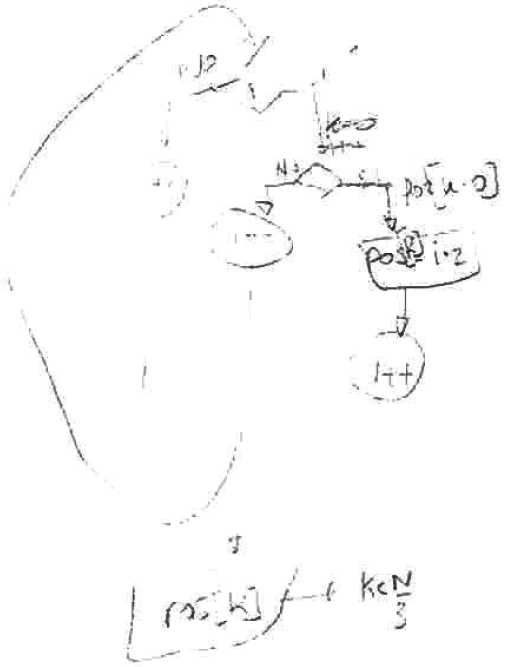
Suggerimento: disegnare innanzitutto il flowchart del metodo tenendo conto che ogni passo corrisponde ad una iterazione ed in secondo luogo procedere alla stesura del codice.

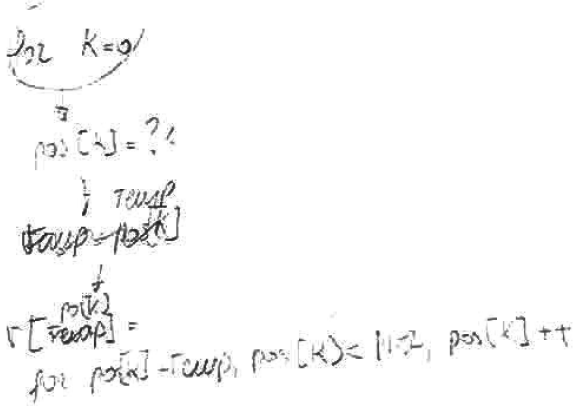
Esercizio 2. Si scriva un programma C che definisca e manipoli un vettore composto di 10 elementi interi; il programma deve
a. Acquisire valori da tastiera e memorizzarli all'interno del vettore
b. Stampare il contenuto del vettore al termine dell'acquisizione
c. Calcolare e stampare la media dei valori nel vettore utilizzando una variabile di tipo float
d. Individuare e stampare a video il valore massimo e la sua posizione ordinale nel vettore.

Approfondimento: considerare il caso in cui il valore massimo occorre più di una volta, e stampare tutte le relative posizioni.

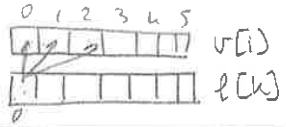








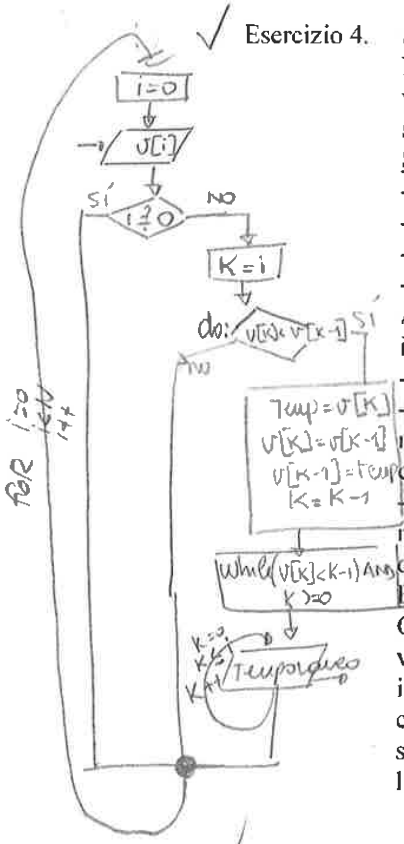
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 76




*Da risolvere a casa*

Esercizio 3. Si scriva un programma C che analizzi il contenuto di un vettore alla ricerca di valori replicati. Il programma dovrà in particolare:

a. Acquisire i valori del vettore da tastiera
b. Scandire il vettore stabilendo se al suo interno esistono valori ripetuti 2 o più volte.
c. Stampi l'elenco dei numeri ripetuti e il numero di occorrenze relative, verificando che ciascun numero compaia una volta sola in tale elenco.




Esercizio 4. Si scriva un programma C che legga da tastiera in un vettore di lunghezza N una sequenza di N numeri, li ordini in senso crescente man mano che vengono introdotti (ordinamento per inserimento, insertion sort) e alla fine stampi il contenuto del vettore.

Suggerimento: a prima vista, sembra che l'algoritmo da seguire sia:
- leggo un valore
- cerco la sua posizione nel vettore, tenendo conto di quanti ho inserito
- sposto tutti i successivi in avanti di una posizione (partendo dal basso)
- inserisco il valore nella sua posizione ordinata

Ad una più attenta analisi, ci si accorge che la parte che cerca la posizione in cui inserire il nuovo valore è ridondante. Si può procedere così;
- leggo un valore
- analizzo il vettore a partire dal basso: se il valore che ho introdotto è minore del dato del vettore che sto considerando, sposto quest'ultimo nella cella successiva
- itero il procedimento tornando all'indietro, finché non trovo un dato minore del valore introdotto (o sono arrivato in cima al vettore): il valore è da inserire nella cella successiva.

Esempio: ho già inserito nel vettore i dati 2, 5, 7, 9. Il valore letto sia 3. Confronto 3 con 9. 3 è minore, sposto in avanti di una posizione 9. Il vettore diventa 2, 5, 7, , 9. Confronto 3 con 7, è ancora minore, sposto il 7 in avanti di un posto. Il vettore diventa 2, 5, , 7, 9. Quando arrivo a confrontare 3 con 2, verifico che è maggiore, lo inserisco nella cella successiva (che era stata liberata al passo precedente). Formalizzare l'algoritmo e tradurlo in C.

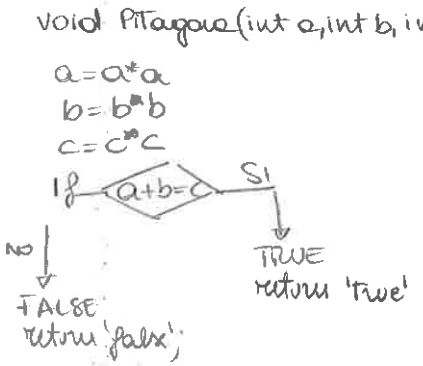
Esercizio 5. Si scriva un programma C che legga da tastiera due numeri interi corrispondenti a base ed esponente, ed esegua il calcolo della potenza $base^{esponente}$. Il programma deve invocare una funzione chiamata *power* dal programma main, con il seguente prototipo:

*int power(int base, int exponent);*

Esempio: siano dati i seguenti valori
*base*=3
*exponent*=2
Il risultato di $base^{exponent}$ sarà 9. In un altro caso con
*base*=2

utilizzando il linguaggio di programmazione C. L'algoritmo realizzato deve essere testato sul calcolatore in modo da verificarne la correttezza sintattica e semantica.

Esercizio 8. [1]Realizzare un programma che generi e stampi tutte le terne pitagoriche nell'intervallo degli interi (A, B e C formano una terna pitagorica se $A^2 + B^2 = C^2$). E' richiesto che il test venga effettuato da una funzione che restituisca il valore TRUE se la terna passata come parametro e' pitagorica, FALSE altrimenti. Suggerimento: attenzione all'overflow della somma!



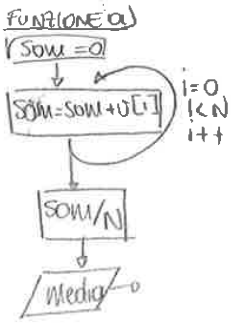

[1] Questo esercizio sarà svolto in modo multimediale e inserito sul Portale, tra il materiale comune, nelle settimane successive.

e. il numero di caratteri di spaziatura;
f. Il numero di parole digitate, dove per parola si intende una sequenza di caratteri alfabetici contigui ("ciao 123 mondo !" dà 2 parole).

Suggerimento: Si utilizzino le funzioni della libreria standard dichiarate nell'header file `<ctype.h>` e si utilizzi una singola variabile di tipo carattere per l'acquisizione.

---

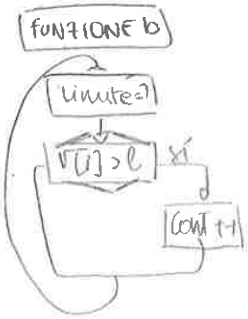
*Da risolvere a casa*




Esercizio 4. Si scriva un programma in grado di manipolare gli elementi di un vettore di interi. Tale programma, dopo aver acquisito il contenuto del vettore, invoca due funzioni:

a. `avgVect`: calcola la media degli elementi del vettore, restituendo tale valore alla funzione chiamante;
b. `upperLimit`: conta il numero di elementi che hanno valore superiore ad un certo limite, restituendolo alla funzione chiamante.

Il programma deve infine visualizzare la media dei valori del vettore e il numero di elementi che superano la media.




Suggerimento: per la funzione `mediaVett` il prototipo sarà:

```
float avgVect (int v[], int n);
```

mentre per la funzione `superanoLimite` il prototipo sarà:

```
int upperLimit (int v[], int n, float limit);
```

Approfondimento: si condensino le due funzioni descritte in un'unica funzione che restituisca il valore medio e che memorizzi nella variabile corrispondente al parametro `superiori` del prototipo il numero di elementi di valore superiore alla media:

```
float over_Avg (int v[], int n, int *superiori);
```

Esercizio 5. Si scriva un programma C che:

a. definisca due variabili di tipo carattere;
b. ne acquisisca il contenuto da tastiera;
c. stabilisca se i caratteri sono entrambi alfabetici:
   i. in caso positivo, controlli se sono uguali e, se non lo sono, stampi i due caratteri in ordine alfabetico;
   ii. in caso negativo, specifichi tramite messaggio se almeno uno dei caratteri è una cifra.

STRINGHE

Esercizio 6. Si realizzi un programma che permetta di inserire da tastiera un testo e che lo stampi su video, cambiando in maiuscolo ogni carattere di inizio parola.
Ad esempio se in ingresso viene fornito il seguente testo:

```
fatti non foste
per viver come bruti
ma per seguir virtute e canoscenza
```

su video deve apparire così:

```
Fatti Non  Foste
Per  Viver Come Bruti
Ma Per Seguir Virtute E Canoscenza
```

# 12BHD INFORMATICA, A.A. 2017/2018

## Esercitazione di Laboratorio 9

Obiettivi dell'esercitazione

- Realizzare programmi con dati complessi, uso dei file

Contenuti tecnici

- Uso di matrici di interi e caratteri
- Uso di insiemi di vettori "paralleli"
- Lettura di file di tipo testo

*Da risolvere preferibilmente in laboratorio*

Esercizio 1. Una matrice di caratteri rappresenta in forma schematizzata una palude. La palude è costituita da zone di fango, rappresentate dal carattere '.', e da zone pietrose, indicate dal carattere '*'. Le dimensioni della matrice possono essere fissate a piacere, mediante dei *#define*, comunque non superiori a 25 righe e 80 colonne.

Esempio di palude:

```
**.*.*....*
..*.*...**.
*...*.*....
.*.*.*.*.*.
..*.*...*.*
```

Realizzare un programma che cerchi nella palude un percorso da sinistra a destra, senza salti, costituito tutto da zone pietrose adiacenti. Si ipotizzi che, adiacente a destra di un punto pietroso, ci possa essere al più un altro punto pietroso (non ci sono diramazioni), sulla stessa riga, sulla riga in alto o sulla riga in basso. Il programma deve visualizzare la sequenza righe in cui ci sono le pietre del percorso trovato (le colonne sono implicite, ci deve essere una pietra per ogni colonna), oppure avvertire che non esiste un percorso. Per la prima versione del programma si utilizzi una matrice di stringhe predefinita. Come approfondimento, la palude viene introdotta da tastiera, e nella versione finale, si legge la palude da file.

Esercizio 2. Si scriva un programma in grado di gestire un listino prezzi, ovvero un programma con cui sia possibile gestire un elenco di prodotti e i loro relativi prezzi in €. Il programma utilizza una matrice di caratteri chiamata *warehouse* di dimensione NxM per memorizzare i nomi dei prodotti (massimo N prodotti) e un vettore di numeri decimali chiamato *price* di dimensione N usato per memorizzare i prezzi dei prodotti (il prezzo presente nell'*i*-esima cella di *price* corrisponde al prezzo del prodotto il cui

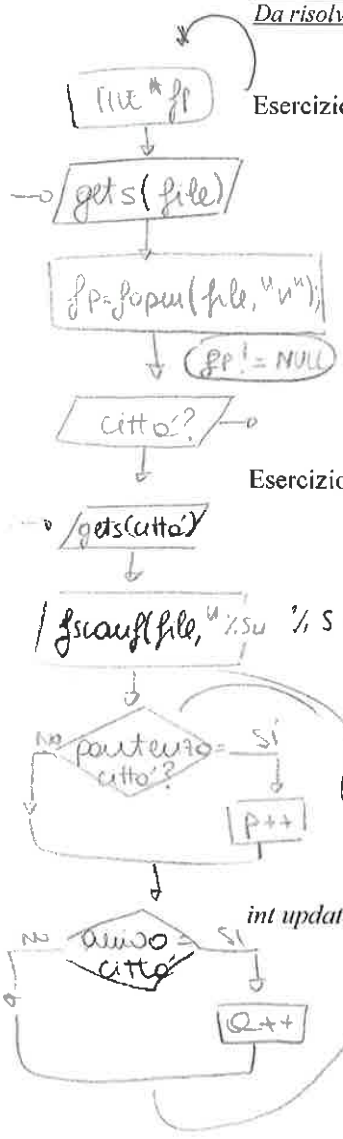
- *print_all*: è una funzione che permette di visualizzare a video il contenuto del listino (elenco prodotti e relativi prezzi). Inoltre, la funzione restituisce due valori (tramite due parametri passati per indirizzo): il prezzo medio ed il prezzo massimo dei prodotti presenti nel listino. Visualizzare a video i due valori restituiti.

*void print_all(char warehouse[][M], float price[], int n, float *avg, float *max);*

Avvertenza: l'esercizio suggerisce di utilizzare il vettore *price* contemporaneamente in due modi: per contenere il prezzo della merce, come flag di posizione vuota. Tenerne conto quando si cerca il nome di un prodotto, quando si effettua la stampa, etc.

---

*Da risolvere a casa*




Esercizio 3. Si scriva un programma che legga da un file, il cui nome è introdotto da tastiera, alcune informazioni ferroviarie. Per ciascuna linea, il file contiene le seguenti informazioni (ciascuno dei campi non superi i 20 caratteri di lunghezza e sia privo di spazi)

*<stazione_partenza> <ora_partenza> <stazione_arrivo> <ora_arrivo>*

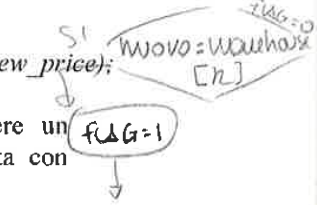
Il programma riceve poi da tastiera il nome di una città: il programma calcoli e stampi il numero di treni in arrivo ed il numero di treni in partenza da tale città (se inclusa nell'elenco).

Esercizio 4. Estendere il programma realizzato come Esercizio 2 aggiungendo due ulteriori funzioni:

- 4) Aggiornamento prezzo prodotto (*update_product*)
- 5) Rimozione prodotto (*remove_product*)

Le due operazioni sono realizzate tramite l'invocazione delle seguenti funzioni:

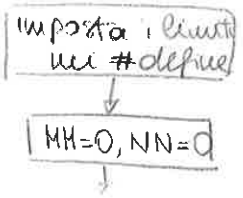
a) nel main: nome prodotto, nuovo prezzo nella funz.




a. *update_product*: è una funzione che permette di aggiornare il prezzo di uno specifico prodotto. La funzione riceve il nome del prodotto da aggiornare e il suo nuovo prezzo. La funzione restituisce 1 se l'aggiornamento è avvenuto con successo, 0 se il prodotto non esiste nel listino.

*int update_product(char warehouse[][M], float price[], int n, char product[], int new_price);*

b. *remove_product*: è una funzione che permette di rimuovere un prodotto dal listino; restituisce 1 se la rimozione è avvenuta con





punto r, c = punto partenza:
es r = 2, c = 1
m = 2
n = 3
ATTENTA

Esercizio 6. Si realizzi un programma che permetta di introdurre da tastiera una matrice di interi. Le dimensioni della matrice, *M x N*, possono essere fissate a priori con dei *#define*, ma sarebbe preferibile che fosse il programma a determinare automaticamente il numero di righe e il numero di colonne della matrice introdotta, entro i limiti massimi fissati dal programma (vedi nota). Il programma deve azzerare la sottomatrice di dimensioni m x n, con m<*M*, n<*N*, a partire dall'elemento r, c (riga, colonna). I valori di m, n, c ed r siano richiesti da tastiera. Il programma visualizzi la matrice prima e dopo l'azzeramento. Il programma deve controllare di non andare oltre i limiti della matrice.


Approfondimento: generalizzare il programma realizzando l'operazione mediante una funzione che effettui la sostituzione di valore, il cui prototipo sia:

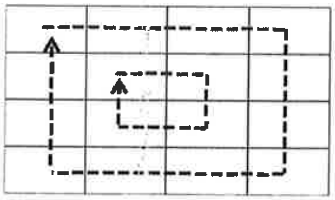
```
void Set (int m[][N], int m, int n, int i, int j, int val)
```

dove *val* è il valore da sostituire.

Nota: un modo semplice per rilevare le dimensioni di una matrice introdotta da tastiera è il seguente: si pone riga e colonna a zero. Si legge fino ad EOF (<CONTROL>+Z) un intero e il carattere successivo. Si memorizza il valore letto nella posizione (riga, colonna) e si incrementa colonna. Si testa poi il carattere letto: se è new-line (è finita la riga), si azzera colonna e si incrementa riga. Alla fine del ciclo riga dice quante righe si sono lette, invece colonna deve essere memorizzata prima dell'azzeramento. Servono ovviamente i controlli per non superare il numero di righe e colonne effettive della matrice.

Esercizio 7. Si scriva un programma in grado di riempire una matrice quadrata di dimensioni NxN di interi (con N pari e maggiore o uguale a 4 definito come costante tramite la direttiva *define*) secondo lo schema delle cornici concentriche; per ogni cornice si parta dalla cella in alto a sinistra e si riempia la cornice progressivamente con numeri crescenti a partire da 1.

m[R][C]
m[i][K]

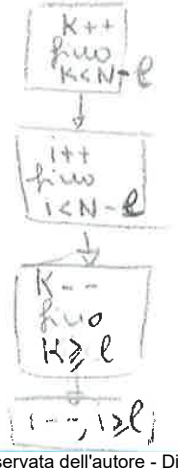


| 0 | 1 | 2 | 3 | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 0 |
| 12 | 1 | 2 | 5 | 1 |
| 11 | 4 | 3 | 6 | 2 |
| 10 | 9 | 8 | 7 | 3 |

K = ℓ - 1
i = ℓ - 2

Si proceda infine con la stampa della matrice assicurandosi che colonne e righe siano correttamente allineate.

1°, ℓ = 1
2°, ℓ = 2




© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 76

```
program.exe -a  x1,y1 x2,y2 x3,y3 x4,y4
```
oppure:
```
program.exe -m  x1,y1 x2,y2 x3,y3 x4,y4
```

Attenzione: ogni coppia di coordinate $x_i, y_i$ deve essere scritta senza spazi in mezzo, mentre le coppie di coordinate devono essere separate da almeno uno spazio.

Esercizio 5. Si consideri il seguente esercizio già proposto e riportato qui per comodità. Si chiede di realizzarlo leggendo la matrice da un file (invece che dalla tastiera) il cui nome viene passato come primo parametro nella riga di comando, sviluppando anche la seconda parte facoltativa, in cui la lunghezza della sequenza e il valore da cercare sia passati rispettivamente come secondo e terzo parametro.
Testo dell'esercizio:
Si scriva un programma C che:

- legga da tastiera una matrice quadrata di dimensione uguale a 5 righe e 5 colonne
- rintracci se tale matrice contiene delle sequenze di elementi adiacenti uguali a zero di lunghezza uguale o maggiore di 3
- visualizzi l'indice di riga in cui tali sequenze si presentano.

Esempio.
Sia la matrice la seguente:

0 0 0 4 5
1 2 0 4 5
1 0 0 4 0
1 2 3 4 5
1 0 0 0 0

La sequenza di valori "0 0 0" compare nella prima e nell'ultima riga e quindi occorre riportare una indicazione del tipo:

La sequenza compare nella riga 1
La sequenza compare nella riga 5

Si osservi che la riga 3 non contiene la sequenza indicata in quanto i tre zeri non si trovano in posizioni contigue.

FACOLTATIVO
Si effettui lo stesso controllo anche lungo le colonne.
Nell'esempio precedente occorre visualizzare, oltre ai messaggi già indicati, anche il seguente:
La sequenza compare nella colonna 3