NUMERO : 353 DATA : 24/09/2012

A P P U N T I

STUDENTE : Rinaldi

MATERIA : Informatica, esercizi + temi
Prof. Rivoira

ESERCITAZIONE (1)

INFORMATICA – a.a. 2010/11
Esercitazione di Laboratorio 1

Obiettivi dell'esercitazione

- Disegnare flow-chart e realizzare i primi semplici programmi C
- Prendere confidenza con il compilatore
- Utilizzare le funzioni messe a disposizione dal linguaggio C per acquisire da tastiera e visualizzare a video valori numerici interi e reali
- Realizzare semplici calcoli basati sugli operatori aritmetici di base
- Sperimentare l'uso dei costrutti condizionali per prendere decisioni

Contenuti tecnici

- Definizione e implementazione della funzione *main* in un progamma C
- Definizione di variabili intere (*int*) e reali (*float*), e loro utilizzo
- Uso di espressioni aritmetiche
- Acquisizione e stampa di valori numerici tramite le funzioni *scanf* e *printf*
- Uso preliminare dei costrutti condizionali *if* e *if-then-else*

Da risolvere preferibilmente in laboratorio

Esercizio 1. Utilizzando il compilatore, scrivere/compilare/eseguire un programma in linguaggio C che visualizzi su schermo la scritta "Hello world!".

Esercizio 2. Scrivere un programma che definisca 3 variabili intere chiamate *operand1*, *operand2* e *result*, e:
a) Acquisisca da tastiera il valore di *operand1* e *operand2* tramite la funzione *scanf*
b) Ne calcoli la somma e la salvi nella variabile *result*
c) Visualizzi a video il valore della variabile *result* utilizzando la funzione *printf*

Esercizio 3. Disegnare il flow-chart e, successivamente, scrivere un programma C per il calcolo del modulo di un numero; in particolare il programma dovrà:
a) Acquisire da tastiera un valore intero, positivo o negativo, e memorizzarlo in una variabile opportunamente definita
b) Stabilire utilizzando il costrutto condizionale *if* se tale variabile contiene un valore negativo e, in questo caso, trasformarlo nel corrispondente valore positivo
c) Stampare a video il valore finale, ovvero il modulo del valore acquisito

Da risolvere a casa

Esercizio 4. Scrivere un programma C in grado di risolvere un'equazione di primo grado espressa nella forma $ax+b=0$; in particolare il programma dovrà:
a) Definire due variabili reali (*float*), a e b per memorizzare i coefficienti dell'equazione
b) Definire una variabile reale chiamata x in cui memorizzare il risultato dell'equazione
c) Acquisire da tastiera il valore dei coefficienti a e b
d) Calcolare il valore di x e visualizzarlo a video
Approfondimento: considerare, tra gli altri, il caso in cui il valore di a sia uguale a 0.

Esercizio 5. Disegnare il flow-chart e, successivamente, scrivere un programma in grado di stabilire se un numero è pari o dispari; in particolare, tale programma dovrà:
a) Acquisire un valore intero da tastiera, memorizzandolo in una variabile
b) Utilizzare un operatore aritmetico opportuno per stabilire se il valore acquisito è divisibile per 2 o meno
c) A seconda dei casi, visualizzare a video un messaggio che indichi se il numero è pari o dispari.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 98

```
   if (variabileA == 0 && variabileB != 0) {
      printf("Impossibile dividere per zero!\n x = non esiste") ;
   }
   if (variabileA != 0 && variabileB == 0) {
   variabileX = 0;
   printf("Il valore di x e': %f", variabileX);
   }
  if (variabileA != 0 && variabileB != 0) {
   variabileX = -variabileB / variabileA;
   printf("Il valore di x e': %f", variabileX);
   }
      return 0;
}
```

ESERCIZIO 5

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

int main()
{
   int n, resto;
   printf("Programma in grado di stabilire se un numero e' pari o dispari
\n");
   printf("\n");
   printf("Inserisci numero da analizzare: ");
   scanf( "%d", &n);

   if ((n%2)== 0)
      printf("Il numero e' PARI");

else
  /* oppure al posto di "else" "if ((n%2)!= 0) "*/
      printf("Il numero e' DISPARI");

   return 0;
}
```

ESERCIZIO EXTRA

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

int main()
{
   int operand1, operand2;

   operand1 = 5;
   operand2 = operand1 *5;
   operand2 = operand2 +3;
   printf("operand2 = %d", operand2);
   return 0;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 98

```
}
```

ESERCIZIO 2

```
/*  Disegnare il flow-chart e, successivamente, scrivere un programma C
che
classifichi un triangolo date le lunghezze dei suoi lati. Il programma deve
implementare le seguenti funzionalità:
a. Ricevere da tastiere 3 numeri interi corrispondenti alle lunghezze dei
lati
b. Stabilire se il triangolo è valido, degenere o non valido
c. In caso sia valido, stabilire se si tratta di un triangolo
i. equilatero, isoscele o scaleno
ii. rettangolo
Suggerimento: un triangolo è valido se ogni lato è strettamente minore
della somma degli altri due, è degenere se un lato è uguale alla somma
degli altri due; un triangolo è rettangolo se rispetta il teorema di Pitagora.*/

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

int main()
{
   int a, b, c;

   printf("CLASSIFICARE UN TRIANGOLO IN BASE ALLE LUNGHEZZE DEI SUOI
LATI");
   printf("\n");
   printf("latoA: ");
   scanf("%d", &a);
   printf("latoB: ");
   scanf("%d", &b);
   printf("latoC: ");
   scanf("%d", &c);
   printf("\n");


   if ( a < b + c || b < c + a || c < a + b)
   {
      printf("il triangolo e' valido!");
      printf("\n");
      printf("\n");
      if (a==b && c==b && a==c)
      printf("il trinagolo e' equilatero!");
      else if ((a==b && a!=c )||(a==c && a!=b)|| (b==c && b!=a))
      printf("il tringolo e' isoscele!");
      else
      printf ("il trinagolo e' scaleno!");
       printf("\n");
   printf("\n");
      if ((a*a==(b*b) + (c*c) )|| (b*b==(a*a)+(c*c))||
(c*c==(a*a)+(b*b)) )

      printf(" il triangolo e' rettangolo!");
   }

   else if (a == b + c || b == c + a || c == a + b)
      printf("il triangolo e' degenere!");

   else
      printf("il triangolo non e' valido!");

       printf("\n");
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 98

}

ESERCIZIO 4

/* Si voglia valutare il valore massimo memorizzabile in variabili di tipo int, long e unsigned int.
Suggerimento: eseguire l'istruzione con il passo-passo del debug e osservare il risultato dell'assegnazione.
a) Verificare che non è una via percorribile provare ad assegnare valori via via più grandi: se ad es. si scrive l'istruzione dato = 3000000000, il compilatore non segnala errore, e al più segnala warning. Cosa si osserva in dato con il watch, dopo l'esecuzione dell'istruzione?
b) Si potrebbe provare a ottenere questi valori in modo "empirico", ovvero acquisendoli tramite la funzione scanf e ristampandoli opportunamente usando la printf. Verificare tramite un programma che anche questa via non è percorribile: il comportamento della scanf in caso di errore nei dati non è dominabile da chi scrive il programma.
c) Realizzare un algoritmo che, tenendo conto delle rappresentazioni binarie dei numeri senza segno e in complemento a 2, permetta di rilevare il valore max: per i numeri con segno, si può attribuire a dato il valore iniziale di 0, poi si incrementa ripetutamente dato. È noto che se si incrementa di 1 il valore più positivo, si ottiene overflow e il valore diventa negativo. Il valore cercato è dunque il precedente al primo valore negativo trovato. Tradurre l'algoritmo in programma e collaudarlo. Come si può modificare l'algoritmo (e il programma) perché operi con i numeri senza segno?

L'idea dell'esercizio 4 e' prendere dimestichezza con i tipi di dato e le loro dimensioni. Come forse avete visto a lezione, il tipo di dato "int" ha una dimensione in bit che e' diversa dal tipo "long". Nel caso del compilatore che usiamo, il tipo "int" ha una dimensione di 32 bit. Esiste anche il tipo "unsigned int" che ha la stessa dimensione del "int" ma non ha segno.

Uno degli obbiettivi dell'esercizio e' vedere come il compilatore gestisce gli overflow, cioe' vedere che succede quando il valore assegnato ad un tipo di dato e' maggiore della sua capacita'. Prova, ad esempio, ad eseguire questo pezzo di codice che manda in overflow una variabile di tipo intero:

```
  i = 2147483647; /* (2^32)/2 - 1
  printf("%d\n", i);
  i = i +1;
  printf("%d\n", i);
```

L'esercizio 4 richiede di fare piu' o meno lo stesso con gli altri tipi di dati, ma l'idea e' avere chiari i concetti delle capacita' dei tipi di dato.
*/

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

int main()
{
  int i;
  i = 2147483647; /* (2^32)/2 - 1*/
  printf("%d\n", i);
  i = i +1;
  printf("%d\n", i);
}
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 98

```
    printf("Inserisci un numero minore uguale di 40!");
  return 0;
}
```

ESERCIZIO 2B

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

int main()
{int i,n,l;
    printf ("inserisci un numero:");
    scanf("%d",&n);
    printf("\n");

if (n<=80){
    for (i=1;i<=n;i++){
    for(l=1; l<=n;l++){
    printf ("*");}
    printf("\n");
    }
    }
    else
    printf("Inserisci un numero minore uguale di 80!");
  return 0;
}
```

ESERCIZIO 3

/*Esercizio 3. Si scriva un programma C che acquisisca numeri interi positivi e negativi
in input da tastiera finché non viene inserito il valore 0 e ne calcoli la media. Il programma dovrà
a. Accumulare i valori acquisiti in una variabile somma opportunamente definita ed inizializzata
b. Contare il numero i di valori acquisiti
c. Effettuare la divisione tra somma e i
Approfondimento: modificare il programma in modo che durante l'acquisizione siano scartati i valori esterni ad un intervallo definito da due costanti MIN e MAX stabilite a piacimento dal programmatore.
*/

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

int main()
{
   float somma=0;
   int contatore =0;
   float numero=1;
   float media=0;

printf("acquisisco numeri interi  da tastiera finche' non viene inserito il valore zero! \n");
printf("alla fine esibisco il risultato della media dei numeri precedenti!\n\n\n");

   do {
   printf("\ninserisci un numero : ");
   scanf("%f", &numero);
   somma +=numero;
   ++contatore;
   printf("Hai inserito %d valori!\n", contatore);
   }
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 98

```
do {
   printf("\ninserisci un numero intero: ");
   scanf("%d", &numerointero);
   somma +=numerointero;
   ++contatore;
   media= somma/(contatore);
   printf("Hai inserito %d valori!\n", contatore);
   }

   while(contatore<10 && media>decimaleN);
printf ("\nLa media dei valori inseriti e': %f!\n\n\n\n", media);
printf("fine aquisizione dati!");

   return 0;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 98

```
  return 0;
  }
```

ESERCIZIO 2

```
/*Esercizio 2. Scrivere un programma C che definisca due vettori v1 e v2 di
N elementi
di tipo intero e li completi con valori acquisiti da tastiera secondo le
seguenti regole:
a. In v1 siano memorizzati tutti i valori positivi ed i valori negativi
multipli di 3
b. In v2 siano memorizzati i valori negativi non multipli di 3 e dispari
c. Tutti gli altri valori acquisiti siano ignorati
d. L'inserimento si conclude quando uno dei due vettori è pieno; a
questo punto si stampi a video il contenuto dei vettori acquisiti.*/
#include <stdio.h>
#include <stdlib.h>
#define N 10

int main()
{
  int i,j,k,w;
  int v1[N];
  int v2[N];
  int x=0;

  for (i=0,j=0; i<N && j<N; )
  {
    printf("Inserisci un valore da caricare nel rispettivo vettore: \n");
    scanf("%d",&x);

    if (x%3==0){
    v1[i]=x;
    k=i;
    i=i+1;
    }
    else if ((x<0 && x%3!=0)||(x%2!=0)){
    v2[j]=x;
    w=j;
    j=j+1;
    }
    else
    printf("Il valore immesso non èstato utilizzato!");

  }

  printf("i due vettori hanno componenti :\n");
  printf("\nv1:");
  for (i=0;i<=k;i++){
  printf("%d ",v1[i]);}

  printf("\nv2:");
  for (j=0;j<=w;j++){
  printf("%d ",v2[j]);
  }

  return 0;
}
```

ESERCIZIO 3

```
/*Esercizio 3. Scrivere un programma C che acquisisca un massimo di N
valori interi,
con N costante definita a piacimento. L'acquisizione deve procedere
finché la serie di numeri è monotona, ovvero costituita da numeri in ordine
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 98

ESERCITAZIONE 5

ESERCIZIO 1

/*Esercizio 1. Si scriva un programma C che:
a. legga un vettore di N elementi interi (con N costante predefinita)
b. determini se gli elementi di tale vettore costituiscono una successione palindroma.
Suggerimento: una successione si dice palindroma se e' identica letta da sinistra verso destra o da destra verso sinistra.
Esempio: le seguenti successioni di valori sono palindrome:
12 3 12
1 4 5 4 1
10 10 10
mentre la seguente non è palindroma:
1 3 4 3 2*/

```
#include <stdio.h>
#include <stdlib.h>
#define N 9

int main()
{
   int i;
   int pal[N];
   int V1[N];
   int x;
   int somma=0;

x=N-1;

for(i=0;i<N;i++)
{
   printf("inserisci un valore:");
   scanf("%d", & V1[i]);
}
for(i=0, x=N-1; i<(N-1)/2 && x>(N-1)/2; i++,x--)
  {
     if(V1[i]==V1[x])
     pal[i]=0;
     else
     pal[i]=1;
     }
for (i=0; i<(N-1)/2; i++)
{
   somma= somma+pal[i];
}

if (somma == 0)
printf("il vettore e' palindromo");
else
printf("il vettore non e' palindromo");

   return 0;
}
```

ESERCIZIO 2

/*Esercizio 2. Si scriva un programma C che:
a. legga 2 vettori di N elementi interi (con N costante predefinita)
b. stabilisca se i due vettori contengono gli stessi elementi, anche in ordine differente
Esempio: siano dati i due vettori seguenti:
v1 12 3 12 13 29
v2 12 29 13 3 12

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 98

```
printf("\nil vettore 2 ordinato ha componenti:");
for(j=0;j<N;j++){
printf("%d ",v2[j]);}
//confronto vettori
for(x=0;x<N;x++)
{if (v1[x]==v2[x])
flag=flag +0;
   else flag=flag+1;
}
if(flag==0)
printf("i due vettori contengono gli stessi elementi");
else
printf("i due vettori non contegono gli stessi elementi");
   return 0;
}
```

ESERCIZIO 3

/* Esercizio 3. Si scriva un programma C che legga da tastiera due numeri interi
corrispondenti a base ed esponente, ed esegua il calcolo della potenza baseesponente. 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 baseexponent sarà 9. In un altro caso con
base=2
exponent=3
Il risultato di baseesponente sarà 8.

Suggerimento: all'interno della funzione, calcolare la potenza moltiplicando iterativamente la base per se stessa un numero di volte pari all'esponente.*/

```
#include <stdio.h>
#include <stdlib.h>
int power (int,int);/*Prototipo della funzione*/
int main()
{
int base;
int esponente;
int risultato;
printf("inserisci base ed esponente:");
scanf("%d %d", &base, &esponente);

risultato=power(base,esponente);
printf("il risultato e':%d",risultato);
   return 0;
}
int power( int b, int e)
{
   int tot=1;
  int i;

  if (e!=0){
   for(i=1;i<=e;i++)
   {
      tot=tot*b;
   }
  }
  else
  tot=1;
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 98

```
  }
  }
  else
  tot=1;
   return tot;
}
```

ESERCIZIO 5

```
/* Esercizio 5. Si scriva un programma C che analizzi il contenuto di un
vettore alla
ricerca di valori duplicati. 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.*/
#include <stdio.h>
#include <stdlib.h>
#define N 10
int main()
{
   int vett[N],x[N],y[N];
   int i;
   int j;
   int flag=0;

printf("inserisci componenti vettori:\n");
   for(i=0;i<N;i++)
   { printf("componente%d: ",i);
   scanf("%d",&vett[i]);
   }

    for(i=0,flag=0;i<N;i++){
    x[i]=vett[i];
    for(j=1,flag=0;j<N;j++)
    {
      if(vett[i]==vett[j])
       flag=flag+1;
       y[i]=flag;
    }

    }

    for(i=0 ,flag=0;i<N;i++){
    if (y[i]!=1 && y[i]!=0){
    printf("\nil valore %d e' stato trovato %d volte!\n", x[i], y[i]);
    flag=1;

    }
    }
    if(flag==0)
    printf("non ci sono componenti duplicate una o piu' volte!");

  return 0;
}
```

ESERCIZIO 6

```
/* Esercizio 6. Scrivere un programma C che acquisisca da tastiera un
numero intero

positivo o negativo e stampi a video il valore binario in complemento a 2

su 8 bit. In caso il numero non sia rappresentabile nel numero di bit a

disposizione, stampare a video un messaggio di errore.
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 98

```
return 0;
}
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 98

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

int main()
{
 int ncaratteri=0;
 int nalfabetici=0;
 int nmaiuscoli=0;
 int ndecimali=0;
 int nspaziature=0;
 int nparole=1;
 int i=0;
 char x;

  printf("inserisci i caratteri:");
  x=getchar();

for(i=0;x!=10;i++)
/*(x=getchar())!= 10*/
{

if( isdigit(x))
ndecimali= ndecimali+1;

else if(isalpha(x))
nalfabetici=nalfabetici+1;

else if(isspace(x))
nspaziature=nspaziature+1;

if(isupper(x))
nmaiuscoli=nmaiuscoli+1;

x=getchar();
}
nparole=nparole+nspaziature;
ncaratteri=i+1;

 printf("\na. Numero di caratteri introdotti %d\nb. Numero di caratteri
alfabetici %d\nc. Numero di caratteri maiuscoli %d\nd. Numero di cifre
decimali %d\ne. Numero di caratteri di spaziatura %d\nf. Numero di parole
digitate
%d",ncaratteri,nalfabetici,nmaiuscoli,ndecimali,nspaziature,nparole );

   return 0;
}
```

ESERCIZIO 3

```
/*
```

Esercizio 3. Si scriva un programma C che:

a. Acquisisca una stringa di massimo N caratteri (con N valore costante)

b. Ne manipoli il contenuto

i. Trasformando tutte le lettere minuscole in maiuscole

ii. Rimpiazzando tutti i caratteri non alfanumerici con il carattere '_'

iii. Sostituendo i caratteri numerici con il carattere '*'

c. Scandisca la stringa manipolata per contare quante parole sono presenti al suo interno, considerando una o più occorrenze del carattere '_' come separatore tra parole.

Approfondimento: l'ordine in cui vengono eseguite le manipolazioni influenza il risultato? Verificare la risposta scrivendo due versioni del

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 98

```
   stringa[i]=getchar();
   j=i;
}

printf("la stringa acquisita e': ");
for (i=0;i<j;i++)
{
   printf("%c ",stringa[i]);
}

   return 0;
}
```

ESERCIZIO 5

```
/*
Esercizio 5. Realizzare un programma che legga da tastiera una sequenza di caratteri
costituita di sole lettere e spazi, e la compatti sostituendo 2 o più
occorrenze consecutive di una lettera con il numero di occorrenze seguito
dalla lettera e sostituendo 2 o più spazi con un solo spazio.
Esempio: La prima riga e' l'input, la seconda e' l'output.
AADDDDKPP PrRRRR
2A4DK2P Pr4R
wwwwwwwwwwYY SSSSS SSSSGGGGGGGGGGGGGG
10wYY 5S 4S14G
ZZZZZFZZQZZZZZZZZZZZZ
5ZF2ZQ13Z*/
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define N 20
int main()
{
//DICHIARAZIONI
char stringa[N];
char stringa2[N];
int i,x;
int n=1;
int j=0;
int w=0;
//inserimento caratteri
printf("inserire caratteri alfabetici e di spaziatura in caso contrario i
caratteri diversi vengono ignorati:");
x=getchar();
for (i=0;i<N;)
{if (isalpha(x)||isspace(x))
{
   stringa[i]=x;
   i++;
}
x=getchar();
}

//stampa caratteri corretti immessi
   printf("i caratteri che verranno analizzati sono:");
   printf("%s",stringa);
 //trovare lettere uguali consecutive
 for (i=0;i<N;i++)
 {
  if (stringa[i]==stringa[i+1]){
  n=n+1;

  stringa2[j]=n;
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 98

ESERCITAZIONE (7)

ESERCIZIO 1

```
/*Esercizio 1. Si scriva un programma che:
a. Definisca un vettore di caratteri e acquisisca una stringa al suo interno
b. Analizzi tale stringa rispondendo alle seguenti domande
i. Quanto è lunga la stringa?
ii. Quanti caratteri sono alfabetici e quanti numerici?
iii. Acquisita una seconda stringa, quest'ultima è inclusa nella prima?
(ad esempio: "importante" include "porta")*/
#include <stdio.h>
#include <stdlib.h>
#define N 100
#include <string.h>
#include <ctype.h>

int main()
{
   char stringa1[N];
   char stringa2[N];
   int contatore_caratteri=0;
   int lettere=0;
   int numeri=0;
   int i;

   printf("inserisci una stringa di caratteri:");
 gets(stringa1);
//lunghezza stringa
for(i=0;i<N;i++)
{
   if (stringa1[i]!='\0')
contatore_caratteri++;
   else
   i=N;
}
printf("\nla stringa e' lunga %d caratteri", contatore_caratteri);
//caratteri alfabetici?
//caratteri numerici?
for(i=0;i<contatore_caratteri;i++)
{
   if (isalpha(stringa1[i]))
      lettere++;
      else if (isdigit(stringa1[i]))
      numeri++;
}
printf("\nla stringa contiene %d lettere e %d numeri",lettere,numeri);
//seconda scritta
printf("\ninserisci un'altra stringa di caratteri:");
 gets(stringa2);
// verificare che la seconda scritta è contenuta nella prima

if (strstr(stringa1,stringa2))
{
 printf("la striga 2 e' contenuta in quella precedente!");
}
else
printf("la striga 2 non e' contenuta in quella precedente!");
   return 0;
}
```

ESERCIZIO 2

```
/*Esercizio 2. Si scriva un programma che acquisisca 2 stringhe
corrispondenti a 2 orari
nel formato hh:mm. Il programma deve:
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 98

```
        else if (orario1[1]==orario2[1])
         { if(orario1[3]< orario2[3])
         printf("il primo orario e' precedente al secondo\n");


            else if  (orario1[3] == orario2[3])
              {if (orario1[4]< orario2[4])
              printf("il primo orario e' precedente al secondo\n");

                 else
                   printf("il secondo orario e' precedente al primo\n");}}}



   return 0;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 98

```
  {
    sommatoria= sommatoria +v[i];
  }
  media=sommatoria/n;

  return media;
}
//FUNZIONE 2
int threshold_vect(int v[], int n, float media)
{int i;
int contatore=0;

  for(i=0;i<n;i++)
  {
    if (v[i]>media)
    contatore++;
  }

  return contatore;
}
```

ESERCIZIO 2

```
/*Esercizio 2. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
# define N 12
# define M 10

int insert_product(char products[][M], float price[],int n, char
new_product[])
{
  int i;
  int trovato;
  int flag=0;

  for(i=0; i<n && trovato!=0; i++)
  {
    if(strcmp(products[i], new_product)==0)
      trovato=0;
    else
      trovato=1;
  }

  if(trovato==1)//ho trovato un nuovo prodotto
  {
    for(i=0; i<n && flag!=1; i++)
    {
      if (price[i]==-2)
      {
        price[i]=-1;
        strcpy(products[i],new_product);
        flag=1;
      }
          }
  }

  return trovato;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 98

```
int main()
{
   char products[N][M];
   float price[N];
   int comando=10;
   int trovato;
   int exit=0;
   int i,j;
   char new_product[M];
   char old_product[M];
   char cmd_str[5];
   float prezzo_medio;
   float prezzo_max;
   char product[M];
   float new_price;
   int ok,rimosso;


   for(i=0; i<N; i++)
      price[i]=-2;

   for(i=0; exit==0; i++)
   {
      printf("\n\t1- inserire nuovo prodotto");
      printf("\n\t2- stampa elenco prodotti");
      printf("\n\t3- modificare prezzo di un prodotto");
      printf("\n\t4- rimuovere un prodotto");
      printf("\n\t0- esci dal programma");
      printf("\npremi il tasto corrispondente all'operazione da effettuare:");
      gets(cmd_str);
      sscanf(cmd_str, "%d",&comando);

      switch(comando)
      {
      case 1://iserire prodotto
      {
         printf("\n\ninserisci nome prodotto:");
         gets(new_product);
         //scanf("%s", & new_product);

         trovato=insert_product(products, price, N, new_product);
         if (trovato ==0)
            printf("\nil prodotto e' gia' presente nel catalogo e non e' stato
inserito");
         else if(trovato==1)
            printf("\nil prodotto e' stato inserito correttamante!");

      }
      break;
      case 2://stampare tutto prezzo medio prezzo max
         print_all( products, price, N, &prezzo_medio , &prezzo_max);

            printf("\n\n\nil prezzo medio dei prodotti e':  %f",prezzo_medio);
   printf("\nil prezzo del prodotto piu' costoso e' : %f",prezzo_max);
         break;
      case 3://modificare prezzo
      {
         printf("\ninserisci il nome del prodotto da aggiornare: ");
         scanf("%s", product);
         printf("\ninserisci il nuovo prezzo: ");
         scanf("%f", &new_price);
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 98

```
contatore=threshold_vect(Vettore1,N,Media);
printf("\n ci sono %d valori maggiori della media", contatore);

return 0;

}

float avg_vect(int V[],int n)
{
   int i;
   float somma=0;
   float media=0;

   for(i=0;i<n;i++)
   {somma=somma+ V[i];}

   media= somma/n;
   return media;


}
int threshold_vect(int v[], int n, float avg)
{
   int i;
   int k=0;

   for(i=0;i<n;i++)
   {
      if(v[i]>avg)
      k++;
   }
   return k;
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 98

```
};

int main(int argc, char* argv[])
{
 int i;
 float lunghezza_percorso=0;
float lunghezza_minima=0;
float segmento[N-1];
struct coordinate punto[N];

 if (argc!=2)
 {
    printf("\nNumero argomenti errato!");
    return 1;
 }


for (i=0;i<N;i++)
{
   printf("\ninserisci coordinata dell'ascissa del punto%d:",i+1);
   scanf ("%d", &punto[i].x);
   printf("\ninserisci coordinata dell'ordinata del punto%d:",i+1);
      scanf ("%d", &punto[i].y);
}

if (strcmp(argv[1],"-m")==0)    //calcola e stampa a video la lunghezza del
percorso composto dai 4 segmenti.
 {
    for (i=0;i<N-1;i++)
    { segmento[i]= sqrt(pow((punto[i+1].x -punto[i].x),2)+pow((punto[i+1].y -
punto[i].y),2));
      lunghezza_percorso+=segmento[i];
    }

printf("\nLa lunghezza del percorso  e': %f",lunghezza_percorso);
 }

 if (strcmp (argv[1],"-a")==0)  //  calcola e stampa a video la distanza
minima tra le coordinate inserite.
 {
    for(i=0;i<N-1;i++)
   {segmento[i]= sqrt(pow((punto[i+1].x -punto[i].x),2)+pow((punto[i+1].y -
punto[i].y),2));
   }
   lunghezza_minima=segmento[0];
   for(i=0;i<N-1;i++){
if(segmento[i]<lunghezza_minima)
      lunghezza_minima=segmento[i];
   }
  printf("\nLa distanza minima  e': %f",lunghezza_minima);
 }

 if (strcmp (argv[1],"-a")==1 &&strcmp (argv[1],"-m")==1)
    {
 printf("\ncomando inesistente");
 return 1;}

return 0;
}
```

ESERCIZIO 3

Esercizio 3. Si scriva un programma per la gestione di una rubrica di massimo 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 98

```
    {
    case 0:
    {
      printf("il programma è stato utilizzato %d vole!\nARRIVEDERCI!",i);
      exit=1;
    }
    break;
    case 1:
    {

      printf("Nome:");
      gets(nuovo_nome);
      printf("Cognome:");
      gets(nuovo_cognome);
      printf("Telefono fisso:");
      gets(nuovo_fisso);
      printf("Telefono mobile:");
      gets(nuovo_mobile);
      da_inserire=0;

      for (a=0; a<nomi_inseriti; a++);
      {
        if ((strcmp(nuovo_nome,n[a].nome)==0) &&
(strcmp(nuovo_cognome, n[a].cognome)==0))
        {
          printf("il nome inserito è già presente in rubrica!\n premere 0
per inserire comunque \n premere 1 per non inserire il numero in
rubrica!");
          scanf("%d",&da_inserire);
        }
      }
      if (nomi_inseriti==N)
      {
        printf("memoria piena!");
        da_inserire=1;
      }
      if (da_inserire==0)
      {
        strcpy(n[nomi_inseriti].nome, nuovo_nome);
        strcpy(n[nomi_inseriti].cognome, nuovo_cognome);
        strcpy(n[nomi_inseriti].fisso, nuovo_fisso);
        strcpy(n[nomi_inseriti].mobile, nuovo_mobile);
        nomi_inseriti++;
      }
    }
    break;
    case 2:
    {
      printf("\n NOME \t COGNOME \tFISSO \tMOBILE");
      for (a=0; a<nomi_inseriti; a++)
      {
        printf("\n%s %s %s %s", n[a].nome,n[a].cognome, n[a].fisso,
n[a].mobile);

      }

    }
    break;
    default:
      printf("comando inesistente!");
      break;
    }
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 98

ESERCITAZIONE (10)

ESERCIZIO 1

Esercizio 1. Si scriva un programma in C in grado di
a. Aprire un file di testo chiamato *input.txt* e contenente una parola per
riga (il cui numero non è noto a priori)
b. Contare il numero di parole incluse nel file
c. Chiudere il file e stampare a video il valore calcolato
Suggerimento:
prima di tutto creare il file *input.txt* con un editor a piacere (ad esempio notepad.exe) nella cartella in cui è presente il programma.
Approfondimento:
si consideri la possibilità che il file contenga più parole sulla stessa riga

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


int main()
{ int contatore=0;
 char riga[N];

   FILE *f;

f=fopen("input.txt", "r");

if(f==NULL)
{printf("errore apertura file");
exit(1);}

while(fgets(riga,N,f)!=NULL)
{ printf("la parola inserita nella riga e': %s", riga);
   contatore += 1;}
printf("\nil numero di parole inserite e':%d", contatore);


if(fclose(f)!=0)
{printf("errore in chiusura");
exit(1);}



   return 0;
}
```

ESERCIZIO 2

Esercizio 2. Si scriva un programma in C in grado di aprire un file di testo contenente

numeri interi (il cui numero non è noto a priori ed è al massimo 1000). Il
programma dovrà:
a. Aprire il file dopo averne acquisito il nome da linea di comando
b. Leggere da file i numeri e memorizzarli in un vettore
c. Chiudere il file
d. Tramite l'invocazione di una funzione, calcolare il valore minimo, massimo e medio dei numeri letti e restituirli alla funzione main.
e. Memorizzare i valori calcolati in un secondo file, il cui nome sia inserito da linea di comando.

```
#include <stdio.h>
#include <stdlib.h>
#define N 1000
float media(int V[], int lunghezza, int* min, int*max )
{
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 98

```
    if(fclose(f)!=0)
        printf("errore in chiusura");
  }

  else
    printf("Errore argomenti inseriti");



  return 0;
}
```

ESERCIZIO 3

Esercizio 3. Partendo dall'esercizio 2 dell'esercitazione 9, si scriva un programma che
permetta di supportare creazione e aggiornamento su file della rubrica telefonica descritta. In particolare il programma dovrà
a. Aprire un file chiamato *rubrica.txt*, inizialmente vuoto, e contenente
per ciascuna riga le seguenti informazioni
Nome Cognome Numero_di_telefono_fisso
Numero_di_telefono_mobile
b. Acquisirne il contenuto nella struttura *Nomi* utilizzando le funzioni *fgets* e *sscanf*
c. Eventualmente aggiornare l'elenco di nominativi e numeri telefonici
acquisendone altri da tastiera
d. Aggiornare il file includendo i nuovi nomi e chiudere il file.

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

typedef struct nomi
 { char nome[N];
 char cognome[N];
 char fisso [N];
 char mobile[N];
}Nomi;


int main()
{
   FILE *f;
 int nomi_inseriti=0;
 Nomi n[100];
 int i,j,exit=0, a;
char nuovo_nome[N];
char nuovo_cognome[N];
char nuovo_fisso[N];
char nuovo_mobile[N];
char istruzione[N];
int trovato=0;
char comando[N];
char riga[100], S[N], C[N], D[N], E[N];

for(i=0; exit!=1; i++)
{
   printf("\n\t\t MENU' PRINCIPALE \nscegliere l'operazione da
effettuare:\n");
   printf("premere 1 per inserire un nuovo nominativo \n");
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 98

ESERCITAZIONE (11)

ESERCIZIO 1

```
/*
Esercizio 1. Si scriva un programma in linguaggio C che legga il contenuto di un file
dopo averne ricevuto il nome da linea di comando. Il numero di righe del file sia al massimo 50 e ciascuna riga del file contenga i seguenti campi, ciascuno composto al massimo da 20 caratteri
<materia> <nome prof> <cognome prof> <periodo> <crediti> <% superamento esame>
Il programma deve stampare a video:
a. la materia che assegna più crediti in assoluto
b. per ciascun periodo didattico (considerandone al massimo 4), la materia più difficile da superare
Il programma dovrà infine richiedere l'inserimento da tastiera di un cognome di professore (massimo 20 caratteri) e stampare a video:
c. la somma dei crediti assegnati dalle materie che insegna
d. la media di superamento degli esami da lui tenuti
Approfondimento: ai punti a) e b) si consideri il caso in cui siano presenti due o più materie che assegnano il numero massimo di crediti o abbiano nello stesso periodo il minor tasso di superamento; il programma stampi l'elenco completo delle materie identificate
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RIGHE 50
#define CARI 20


struct DATI{
char materia [CAR];
char nome[CAR];
char cognome [CAR];
int periodo[CAR];
int crediti[CAR];
int perc_sup[CAR];

}

int main(int argc, char *argv[])
{

  FILE *f;


  if (argc==2)
  {
  f=fopen(argv[1], "r");




  }
  else printf("ERRORE : numero argomenti \n");


  return 0;
}
```

ESERCIZIO 2

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 98

Esercizio 2.Si scriva un programma che legga da file (nome è ricevuto da linea di
comando) informazioni ferroviarie. Per ciascuna linea, il file contiene le
seguenti informazioni (ciascuno dei campi non superi i 20 caratteri di
lunghezza)

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

Ricevuto come ulteriore parametro da linea di comando il nome di una
città, il programma calcoli il numero di treni in arrivo ed in partenza da
tale città se inclusa nell'elenco.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


int main(int argc, char* argv[])

{

   FILE *f;

   char riga[100];

   char partenza[100][20], arrivo [100][20];

   float ora1[100], ora2[100];

int i, j, contatore1=0, contatore2=0;


if (argc!=3)

{

   printf("errore di comando\n");

   exit(1);

}


f=fopen(argv[1],"r");

if(f==NULL)

{

   printf("errore di apertura\n");

   exit(1);

}

i=0;

while(fgets(riga,100,f)!=NULL)

{

   sscanf(riga,"%s %f %s %f", partenza[i], &ora1[i], arrivo[i], &ora2[i] );

   i++;
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 98

```
/*
Ricette di cucina

Suor Germana vuole realizzare una versione elettronica delle sue famose ricette di cucina,
sotto forma di un programma scritto in C. In particolare, si vuole che il programma
identifichi quali sono le ricette cucinabili, dato il contenuto attuale del frigorifero
di una
massaia.
Il programma accede a due file:
1. un file di testo (denominato Germana.txt) contenente gli ingredienti necessari per
   tutte le ricette di Suor Germana secondo il seguente formato:
     • ogni riga è nella forma ricetta ingrediente quantità
     • ricetta è una stringa (max 20 caratteri, senza spazi) che indica il nome della
       ricetta
     • ingrediente è una stringa (max 20 caratteri, senza spazi) che indica il nome
       di un ingrediente
     • quantità è un numero reale che indica la quantità di tale ingrediente nella
       ricetta corrispondente
     • sia ricetta, sia ingrediente sono ripetuti più volte nel file, ma sempre in
       associazione a ingredienti o ricette diversi
     • non è noto a priori il numero di righe del file, né è specificato alcun ordinamento
       noto per il file.
2. un file di testo (il cui nome è passato come primo parametro sulla linea di comando)
   rappresentante il contenuto attuale del frigorifero secondo il seguente formato:
     • ogni riga è nella forma ingrediente quantità
     • ingrediente corrisponde ad uno degli ingredienti presenti nel file delle ricette
     • quantità è un numero reale che identifica la quantità presente di tale ingrediente
       nel frigorifero
     • ogni ingrediente è presente una sola volta in questo file
     • non è noto a priori il numero di righe del file, né è specificato alcun ordinamento
       noto per il file.
Il programma riceve come argomenti sulla linea di comando il nome del file contenente
le disponibilità del frigorifero ed il nome di una ricetta, e deve fornire in output
l'elenco
degli ingredienti della ricetta (con l'indicazione se ciascuno di essi è disponibile o
meno) e
la conclusione finale se la ricetta scelta può essere preparata.

Ad esempio se i file Germana.txt e frigo.txt contenessero i seguenti dati:
     (Germana.txt)              (frigo.txt)
     padellino uovo 1           uovo 1
     frittata olio 0.3          olio 0.5
     padellino olio 0.2         parmigiano 0.1
     frittata uovo 1
     coque uovo 1
     frittata parmigiano 0.2
ed il programma (denominato cerca) venisse attivato con la riga di comando;
     cerca frigo.txt frittata
allora dovrebbe produrre il seguente risultato:
     Ingredienti:
     - olio: OK
     - uovo: OK
     - parmigiano: richiesto 0.2, disponibile 0.1
     Ricetta 'frittata' impossibile
*/

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

int main( int argc, char *argv[] )
{
     const int MAXRIGA = 300 ;
     const int MAXINGR = 100 ;
     const int LUN = 20 ;
     const char filericette[] =
"D:\\Silvano\\Didattica\\INFORMATICA\\SecondoSem\\Esercizi\\38-RicetteCucina\\Germana.txt"
 ;
          /* COMPOSIZIONE DELLA RICETTA RICHIESTA */
     char ingredienti[MAXINGR][LUN+1] ;
     double quantita[MAXINGR] ;
     int Ningr ; /* numero ingredienti totale della ricetta */
     FILE * f ;
     int ok[MAXINGR] ;
     int i, r ;
     char riga[MAXRIGA+1] ;
     char ricetta[LUN+1] ;
```

```
150         fclose(f) ;
151
152             /* 2B: sulla base del vettore ok[] decido se la ricetta
153                e' fattibile */
154         possibile = 1 ;
155         for ( i = 0 ; i<Ningr ; i++ )
156             if ( ok[i]==0 )
157                 possibile = 0 ;
158         if ( possibile==1 )
159             printf("\nRicetta POSSIBILE!!!\n") ;
160         else
161             printf("\nRicetta IMPOSSIBILE\n") ;
162         exit(0) ;
163     }
164
```

```
78          r = sscanf (riga, "%f %s", &temperatura, citta);
79          if (r==2)
80          {
81              if (strcmp(argv[2],citta) == 0)
82              {
83                  somma = somma+temperatura;
84                  cont ++;
85                  if (temperatura > soglia)
86                     supera++;
87              }
88          }
89          else
90              printf("Riga in formato errato - ignorata\n") ;
91      }
92      fclose (f);
93      printf ("La media delle temperature di %s e' %.1f\n", argv[2], somma/cont);
94      if (argc==4)
95          printf("La temperatura %.1f e' stata superata %d volte\n", soglia, supera);
96      exit(0) ;
97  }
98
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 98

```
75          {
76              printf("ERRORE: il file e' finito troppo presto\n") ;
77              exit(1) ;
78          }
79          if ( sscanf( riga, "%d %d", &nStanze, &hPiano ) != 2 )
80          {
81              printf("ERRORE: riga in formato errato\n") ;
82              exit(1) ;
83          }
84              /* opzionale: controllare che nStanze>=1 e 1<=hPiano<=h_max */
85          areaPiano = 0.0 ;
86              /* per ogni stanza del piano, da 0 a nStanze-1 */
87          for ( s = 0 ; s < nStanze; s++ )
88          {       /* leggi le misure */
89              if ( fgets( riga, MAX, f ) == NULL )
90              {
91                  printf("ERRORE: il file e' finito troppo presto\n") ;
92                  exit(1) ;
93              }
94              if ( sscanf( riga, "%d %d", &x, &y ) != 2 )
95              {
96                  printf("ERRORE: riga in formato errato\n") ;
97                  exit(1) ;
98              }
99                  /* aggiorna areaPiano */
100             areaPiano = areaPiano + (x * y)/10000.0 ;
101         }
102         areaTot = areaTot + areaPiano ;
103         volTot = volTot + areaPiano * (hPiano/100.0) ;
104     }
105     fclose(f) ;
106     printf("Superficie totale dell'edificio: %.2f metri quadri\n", areaTot) ;
107     printf("Volume totale dell'edificio: %.2f metri cubi\n", volTot ) ;
108     exit(0) ;
109 }
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 98

```
72      {
73          printf("ERRORE: formato errato nella prima riga\n") ;
74          exit(1) ;
75      }
76      c = 0 ;
77      while ( fgets( riga, LUN, f ) != NULL )
78      {
79          r = sscanf( riga, "%d %d", &mat,&vot ) ;
80          if ( r != 2 || mat<1 || mat>999999 || !( vot==0 || ( vot>=18 && vot<=30) ) )
81              printf("ATTENZIONE: riga in formato errato - ignorata\n") ;
82          else
83          {       /* aggiungi ai vettori */
84              matricola[c] = mat ;
85              voto[c] = vot ;
86              c++ ;
87          }
88      }
89      fclose(f) ;
90      if ( c != N )
91      {
92          printf("ERRORE di coerenza nel file\n") ;
93          exit(1) ;
94      }
95          /* 2) Acquisisci il comando dell'utente */
96      if ( !(
97              (argc==2 && strcmp(argv[1], "stat")==0) ||
98              (argc==4 && strcmp(argv[1], "voto")==0)
99          ) )
100     {
101         printf("ERRORE: numero argomenti errato\n");
102         exit(1) ;
103     }
104     strcpy( comando, argv[1] ) ;
105     if ( strcmp( comando, "voto" )==0 )
106     {
107         r1 = sscanf(argv[2], "%d", &mat ) ;
108         r2 = sscanf(argv[3], "%d", &vot ) ;
109         if ( r1 != 1 || r2 != 1 )
110         {
111             printf("ERRORE: matricola e voto devono essere numerici\n") ;
112             exit(1) ;
113         }
114     }
115         /* 2a) "stat" */
116     if ( strcmp(comando, "stat")==0 )
117     {   /* 2a1) calcola e stampa le statistiche */
118         c = 0 ;
119         somma = 0 ;
120         for ( i = 0 ; i < N ; i++ )
121             if ( voto[i]!=0 )
122             {
123                 c++ ;
124                 somma = somma + voto[i] ;
125             }
126         printf("promossi = %d (%f %%)\n", c, (double)c/(double)N*100.0 ) ;
127         printf("voto medio = %f\n", (double)somma/(double)c ) ;
128     }
129     else if ( strcmp(comando, "voto")==0 )
130     {       /* 2b) "voto nmatricola valorevoto" */
131             /* ricerca 'mat' all'interno del vettore matricola[] */
132             /* output: trovato=1/0, pos */
133         trovato = 0 ;
134         for (i=0; i<N && trovato==0; i++)
135             if ( matricola[i] == mat )
136             {
137                 trovato = 1 ;
138                 pos = i ;
139             }
140             /* controlla se e' valido */
141         if ( trovato == 1 && voto[pos]==0 )
142         {       /* modifica il voto all'interno del vettore */
143             voto[pos] = vot ;
144                 /* salva i vettori su file */
145             f = fopen( nomefile, "w" ) ;
146             if ( f==NULL )
147             {
148                 printf("ERRORE: impossibile scrivere il file modificato\n") ;
```

5

```
/*
Presenze ai corsi

Un professore vuole realizzare un programma che gli permetta di effettuare delle statistiche
sulle presenze ai corsi universitari da lui tenuti.
Ogni corso universitario è caratterizzato da un codice (es. 06AZNDI). Ogni volta che
il docente effettua una lezione, deve richiamare il programma per inserire le informazioni
relative a tale lezione, ed in particolare: data e numero di studenti presenti alla lezione.
Le informazioni sono memorizzate in un file di lavoro denominato lezioni.txt. Tale
file è composto da un numero variabile, non noto a priori, di righe, ciascuna delle quali contiene
le informazioni relative ad una singola lezione. Il file può contenere le informazioni
relative a molti corsi diversi, liberamente inframmezzati. Il formato di ciascuna riga del
file è il seguente:
codice data numstudenti
dove:
* codice è il codice del corso (max 10 caratteri, senza spazi);
* data è la data della lezione, rappresentata come numero intero tra 1 e 365;
* numstudenti è il numero di studenti presenti, rappresentato come numero intero positivo.
Il programma viene richiamato con due argomenti sulla linea di comando: il primo
argomento indica il codice del corso interessato, mentre il secondo indica l'operazione da
eseguire. L'operazione può essere I per "inserimento" oppure S per "statistiche." In particolare:
* nel caso di inserimento di una nuova lezione (relativa al corso indicato sulla linea di
comando), il programma chiederà all'utente le informazioni necessarie (data e numero
di studenti) ed aggiornerà il file di lavoro aggiungendovi una riga. Compiuta tale
elaborazione, il programma termina.
* stampa delle statistiche di un corso. In tal caso il programma calcola e stampa, per
il corso indicato sulla linea di comando, le seguenti quantità: data della lezione con
il maggior numero di studenti, data della lezione con il minor numero di studenti,
numero medio di studenti presenti alle lezioni. In seguito il programma termina.

Ad esempio, supponendo che il programma sia denominato registro, e che il file
lezioni.txt sia inizialmente vuoto, una possibile interazione con il programma è la
seguente (si noti che c:> è il prompt del sistema operativo):
    c:> registro 06AZNDI I
    Data: 101
    Studenti: 40
    c:> registro 04KK2WE I
    Data: 104
    Studenti: 99
    c:> registro 06AZNDI I
    Data: 98
    Studenti: 45
    c:> registro 06AZNDI S
    Il minimo di studenti si e' raggiunto in data 101
    Il massimo di studenti si e' raggiunto in data 98
    La media di studenti vale 42.5
*/

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

int main( int argc, char *argv[] )
{
    const char nomefile[] =
"D:\\Silvano\\Didattica\\INFORMATICA\\SecondoSem\\Esercizi\\36-PresenzeCorsi\\lezioni.txt"
 ;
    const int MAX = 100 ;
    char riga[MAX+1] ;
    char codice[MAX+1] ;
    int data, stud, r ;
    FILE * f ;
    int totStud ; /* somma tutte presenze */
    int nLezioni ; /* numero di lezioni del corso */
    int minStud, maxStud ;
    int dataMinStud, dataMaxStud ;

        /* Controlla i parametri ricevuti */
        /* argv[1] -> codice del corso */
        /* argv[2] -> comando "I" oppure "S" */
    if ( argc!=3 )
    {
```

```
149             }
150             fclose(f) ;
151                 /* stampa statistiche */
152             if ( nLezioni>=1 )
153             {
154                 printf("Il minimo di studenti si e' raggiunto in data %d\n", dataMinStud) ;
155                 printf("Il massimo di studenti si e' raggiunto in data %d\n", dataMaxStud) ;
156                 printf("La media del numero di studenti vale %.1f\n",(double)totStud / (
    double)nLezioni ) ;
157             }
158             else
159                 printf("Non ci sono lezioni del corso %s\n", argv[1]) ;
160         }
161         exit(0) ;
162     }
```

```
76          {
77              r = sscanf( riga, "%*d %*d %s", matricola ) ;
78                  /* NOTA: gli asterischi nella stringa di formato della sscanf
79                  (come %*d) servono per far leggere il dato corrispondente
80                  ma non memorizzarlo in alcuna variabile.
81                  In effetti qui i primi due campi numerici non ci servono */
82
83              if ( r != 1 )
84                  printf("Riga in formato errato - ignorata\n") ;
85              else
86              {
87                      /* Cerca se 'matricola' è già presente */
88                  presente = 0 ;
89                  for (i=0; i<N && presente==0; i++)
90                      if (strcmp(matricola, nomi[i])==0)
91                          presente=1;
92                  /* Se è nuovo, aggiungilo */
93                  if ( presente==0 )
94                  {
95                      strcpy( nomi[N], matricola ) ;
96                      N++ ;
97                  }
98              }
99          }
100         fclose(f) ;
101         printf("Ci sono %d dipendenti diversi\n", N) ;
102     }
103     else
104     {
105             /* CALCOLO DEL TEMPO LAVORATO DAL DIPENDENTE LA CUI    MATRICOLA È argv[2] */
106         max = 0 ;
107         min = 24*60 ;
108         passaggi = 0 ;
109         while ( fgets( riga, MAX, f ) != NULL )
110         {
111             r = sscanf( riga, "%d %d %s", &ore, &minuti, matricola ) ;
112             if ( r != 3 )
113                 printf("Riga in formato errato - ignorata\n") ;
114             else
115             {
116                 tempo = ore * 60 + minuti ;
117                 if ( strcmp( matricola, argv[2] ) == 0 )
118                 {
119                     if ( tempo<min )
120                         min = tempo ;
121                     if ( tempo>max )
122                         max = tempo ;
123                     passaggi ++ ;
124                 }
125             }
126         }
127         fclose(f) ;
128         if ( passaggi>=2 )
129             printf("Il dipendente di matricola %s ha lavorato per %d minuti\n",
130                     argv[2], max-min ) ;
131         else
132             printf("ERRORE: Il dipendente %s ha fatto solo %d passaggi\n",
133                     argv[2], passaggi) ;
134     }
135     exit(0);
136 }
```

```
73          considerata (in funzione di comando) */
74      f = fopen(argv[1],"r") ;
75      if ( f==NULL )
76      {
77          printf("ERRORE: impossibile aprire file %s\n", argv[1]) ;
78          exit(1) ;
79      }
80      totVoti = 0;
81      nVoti = 0 ;
82      while ( fgets(riga, MAX, f) != NULL )
83      {
84          r = sscanf( riga, "%d %d %*s %d", &matricola, &codice, &voto ) ;
85              /* Nota: %*s fa sì che la stringa NON venga memorizzata */
86          if ( r == 3 )
87              if (((comando == 's' && matricola == valore) ||
88                  (comando == 'e' && codice == valore ) ||
89                  (comando == 'a' && (codice/1000) == valore ))
90                  && voto>=18 )
91              {
92                  totVoti = totVoti + voto ;
93                  nVoti++ ;
94              }
95      }
96      fclose(f) ;
97      if ( nVoti>0 )
98          printf("Valore medio: %.1f\n", (double)totVoti / (double)nVoti ) ;
99      else
100         printf("Non ci sono esami che soddisfino i criteri di ricerca\n") ;
101     exit(0) ;
102 }
```

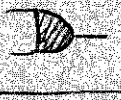
```
54                  /* comando 'find' */
55              /* cerca all'interno del file se esiste un amico 'destinatario'
56                 uguale ad argv[3]
57              */
58          f = fopen( argv[1], "r" ) ;
59          if ( f==NULL )
60          {
61              printf("ERRORE: impossibile aprire file %s\n", argv[1]) ;
62              exit(1) ;
63          }
64          printf("Cartoline ricevute da %s:\n", argv[3]) ;
65          while ( fgets( riga, MAX, f ) != NULL )
66          {
67              r = sscanf( riga, "%s %s %s", mitt, dest, luogo ) ;
68              if ( r==3 )
69              {   /* controlla se l'amico è quello giusto */
70                  if ( strcmp(dest, argv[3])==0 )
71                      printf(" %s da %s\n", mitt, luogo) ;
72              }
73              else
74                  printf("Riga in formato errato - ignorata\n") ;
75          }
76          fclose(f) ;
77      }
78      else if ( argc==6 && strcmp(argv[2], "new")==0 )
79      {
80                  /* comando 'new' */
81              /* controlla se esiste già una cartolina con
82                  mittente == argv[3]
83                  destinatario == argv[4]
84                  luogo == argv[5]
85              */
86          esiste = 0 ;
87          f = fopen( argv[1], "r" ) ;
88          if ( f==NULL )
89          {
90              printf("ERRORE: impossibile aprire file %s\n", argv[1]) ;
91              exit(1) ;
92          }
93          while ( fgets( riga, MAX, f ) != NULL )
94          {
95              r = sscanf( riga, "%s %s %s", mitt, dest, luogo ) ;
96              if ( r==3 )
97              {       /* controlla se la cartolina è duplicata */
98                  if ( strcmp(mitt, argv[3])==0 && strcmp(dest, argv[4])==0
    && strcmp(luogo, argv[5])==0 )
99                  {
100                     esiste = 1;
101                     printf("Attenzione: cartolina già esistente\n") ;
102                 }
103             }
104             else
105                 printf("Riga in formato errato - ignorata\n") ;
106         }
107         fclose(f) ;
108             /* se non esiste ancora, aggiunge una nuova riga al file */
109         if ( esiste==0 )
110         {       /* aggiungi una riga */
111             f = fopen( argv[1], "a" ) ;
112             if ( f==NULL )
113             {
114                 printf("ERRORE: impossibile modificare il file %s\n", argv[
    1]) ;
115                 exit(1) ;
116             }
117             fprintf( f, "%s %s %s\n", argv[3], argv[4], argv[5] ) ;
118             fclose(f) ;
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 98

X
(9)

```
1   /*
2   Gestione magazzino
3
4   Un'azienda deve tenere traccia dei beni
    presenti in un magazzino. L'utente inserisce da
5   tastiera dei "comandi" nel seguente formato:
6       bene EU quantità
7   dove:
8   • bene è il nome di un bene;
9   • EU è la lettera 'E' per entrata, 'U' per
    uscita;
10  • quantità è la quantità di bene entrata o
    uscita.
11
12  L'utente termina il caricamento inserendo un
    comando pari a FINE. In tal caso il programma
13  deve stampare le quantità di beni presenti a
    magazzino.
14  Esempio:
15      viti E 10
16      dadi E 50
17      viti U 5
18      viti E 3
19      FINE
20      Beni presenti nel magazzino:
21      viti 8
22      dati 50
23  */
24
25  #include <stdio.h>
26  #include <stdlib.h>
27  #include <string.h>
28  #define MAX 100
29  #define LUN 30
30
31  int main(void)
32  {
33      char prodotti[MAX][LUN] ;
34      char prod[LUN] ;
35      int quantita[MAX] ;
36      int qta ;
37      char dir ;
38      int i ;
39      int trovato ;
40      int N ; /* dimensione dei vettori
    prodotti[] e quantita[] */
41
42      N = 0 ;
```

```
81                  if ( trovato == -1 )
82                      printf ("Prodotto %s non
    trovato in magazzino\n", prod);
83                  else
84                      /* decrementa la posizione
    corrispondente del vettore quantita[] */
85                      quantita [trovato] =
    quantita [trovato] - qta ;
86              }
87          }
88      }
89      while ( strcmp(prod, "FINE") != 0 ) ;
90      printf ("Beni presenti in magazzino:\n");
91      for (i=0; i<N; i++)
92          printf ("%s %d\n" , prodotti [i],
    quantita [i]) ;
93      exit(0) ;
94  }
```

AND &&	•		VERE ENTRAMBE → VERA
OR ||	+		BASTA CHE UNA DELLE 2 SIA VERA → VERA
NOT !=	− / '		NEGA
XOR VER. FALSO FALSO VERO	⊕		A e B discordi → VERO

RAPPORTO O FATTORE di COMPRESSIONE $C = \frac{\text{dim. dati}}{\text{dim dati compressi}}$ $N:1$ $N\,x$

RISPARMIO DI SPAZIO $S = 1 - \frac{\text{dim dati compressi}}{\text{dim dati iniziali}}$ %

$\rightarrow$ dim dati iniziali $= \frac{\text{dim dati compressi}}{(1-S)}$

MP3 STEREO

- $1\ Hz = 1$ prelievo al secondo $\rightarrow$ FREQ. di CAMPIONAMENTO $= f$
- t = tempo audio in secondi
- Poiché è STEREO ha 2 CANALI STEREO $= n°$ $\rightarrow$ NUMERO CANALI nello STEREO $n=2$
- Dimensione BIT PER CAMPIONE $= d$

Dimensione file in bit $= f \cdot d \cdot n° \cdot t$

(f: in Hz; d: in BIT; $n° = 2$ nello STEREO; t: in secondi)

NUMERO COLORI $= 2^n \rightarrow$ (NUMERO DI BIT PER PIXEL)

es. 3 BIT PER PIXEL $2^3 = 8$ colori

NUMERO BIT PER K colori $= \log_2 K = \frac{\log_{10} K}{\log_{10} 2}$

TEMPO MEDIO ACCESSO in MEMORIA

$T_M = H \cdot T_{CACHE} + (1-H) \cdot T_{RAM}$

H = hit ratio (% delle celle trovate nella cache rispetto al totale accessi in memoria)

$H = \frac{n°\text{celle trovate nella cache}}{\text{TOT celle cercate}} \cdot 100$

(se aumenta la CACHE aumenta H)

FILE

APRIRE FILE

DICHIARARE →

```
File *f;
```
(nome PROTOTIPO FILE)

```
f = fopen (<nome file> , <TIPO ACCESSO> );
```
stringa o nome.txt

"r" "w" (scrive ma cancella) "a" (aggiunge)

SEMPRE FARE CONTROLLI →

```
if (f == NULL)
{ printf ("ERRORE APERTURA FILE");
  exit(-1); }
```

CHIUDERE FILE

CHIUDE E CONTROLLA →

```
if( fclose (f) != 0)
{ printf (" ERRORE CHIUSURA file");
  exit (-1);
  }
```

PUNTATORI pag 264

corrisponde all'indirizzo di memoria del DATO, con in più l'informazione del tipo di dato individuato (cioè "puntato")

l'operatore unitario di indirizzo è [redacted]

&<nomevariabile>

- **Definizione PUNTARE**

<tipo> * <nome-ptr>;

(N.B. Se punto un vettore punto l'elemento [0] di esso)

- **CONFRONTO TRA PUNTATORI**

Se p1 e p2 sono 2 puntatori

p1 > p2 { è vera se l'elemento individuato da p1 segue in memoria l'elemento individuato da p2

p1 == p2 { vera se l'elemento individuato da p1 ha uguale indirizzo di p2, cioè è lo stesso

!

FUNZIONI

• Definizione di una funzione

<tipo> <nome funzione> (<parametri_formali>)

{ <definizioni_locali>

<corpo della funzione>

return (<expr>) (parentesi facoltative) }

("<tipo> <nome>", separati da ",")

definizione delle variabili note nella funzione, sono utilizzate solo da questa funzione e da altre funzioni da essa richiamate

• <tipo> = tipo di valore ritornato dalla funzione.

→ 2 casi → la funzione non ritorna nessun valore <tipo> = void

→ la f. ritorna un valore → char, int, float, * puntatore, struttura

• <parametri_formali> è la lista dei valori in ingresso e/o uscita della funzione; essi possono mancare ma in questo caso la lista è sostituita dalla parola void

• (expr) espressione che produce il valore ritornato al programma chiamante

→ se la funzione è di tipo void l'istruzione [return (<expr>);] può essere omessa!

oppure appare senza alcun valore di ritorno

FUNZIONI

1) DOMANDA: QUANTI PARAMETRI MI DEVE RESTITUIRE?

RISP

→ 1 PARAMETRO → FACILE ogni f RESTITUISCE 1 PARAMETRO

→ + di 1 PARAMETRO → HO BISOGNO DEI PUNTATORI → QUANTI? TANTI QUANTI I PARAMETRI CHE VOGLIO 1 CHE MI DARÀ LA FUNZIONE

<TIPO DELLA FUNZIONE> <nome funzione> (<TIPO1> <nome1>, <TIPO2> <nome2>, ...);

<TIPO> *nome PUNTATORE

```
int main ()
{
  ...
  ...
  x = nome funzione ( a , b , c ..... , &n )
}
```

deve essere dello stesso TIPO1

x diventa così il valore di funzione che HO SCRITTO in RETURN deve essere dichiarata con = tipo x0 !

A QSTA VARIABILE VIENE ASSEGNATO IL VALORE DEL PUNTATORE ALL'INDIRIZZO (DEVONO AVERE = TIPO) È COME SE

scanf("%i", &n)

e da tastiera scrivessi il numero che nella funzione ho salvato nel puntatore.

VETTORI

<TIPO> <nome vettore> [<dimensione vettore>];

(#Define dimensione M)

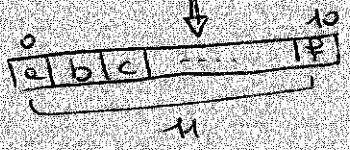
(Elemento 0 → vettore[0])

INIZIALIZZAZIONE di un vettore

Esempio

int vettore [5] = {-1, 2, 3, 4, 5}

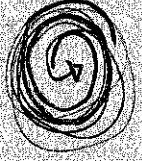
char vettore [11] = {'a', 'b', 'c' ..... 'f'}



Ciclo FOR per riempire in ordine un vettore Esempio

```
for (i=0; i<n; i++)
{
    vettore[i] = 0;
}
```

FOR pag 48



for (inizializzazioni ; condizione ; incrementi)
{ istruzioni ; }

NB

IF pag 25

NB nelle condizioni (== && != || > < >= <=)

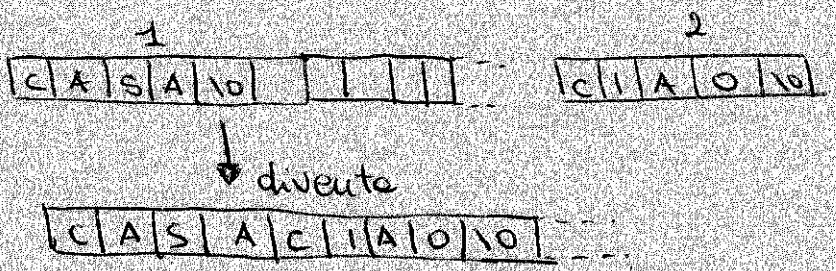
if (condizione)
{ istruzione ; }
else if (condizione) { istruzione ; }
else { istruzione ; }

SIZEOF

sizeof (<TIPO>) ;
↑
ci dice quanti byte occupa il "TIPO"

STRCAT copia una stringa al termine di un'altra stringa
(concatenazione di stringhe)

STRCAT (nome stringa^{1} destinazione, nome stringa^{2} da copiare);



STRNCAT

STRNCAT (string-destinazione, string-da copiare, n)

n: intero che dice i primi n caratteri che copia da a

STRCMP (confronta 2 stringe)

STRcmp (nome string 1, nome string 2) = X;

X è un intero

X = 0 → stringhe uguali

X > 0 → str 1 segue alfabeticamente la str 2

X < 0 → se 1 precede 2

STRNCMP

STRNCMP (Stringa 1, Stringa 2, n) = x

analizza i soli primi n caratteri delle 2 stringhe

STRCAT copia una stringa di termine da un'altra stringa (concatenazione di stringhe)

STRCAT (nome stringa 1 destinazione, nome stringa 2 da copiare);

STRNCAT

STRNCAT (string_destinazione, String_da copiare, n)

intero che dice i primi n caratteri che copia da a

STRCMP (confronta 2 stringe)

strcmp (nome string 1, nome string 2) == X;

X è un intero

X = 0 → stringhe uguali

X > 0 → str 1 segue alfabeticamente la str 2

X < 0 → se 1 precede 2

STRNCMP

STRNCMP (Stringa 1, Stringa 2, n) = x

n ↑ analizza solo i primi n caratteri delle 2 Stringhe

STRLEN

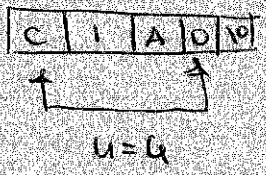
int STRLEN (nome stringa);

u = STRLEN (nome stringa);

↓

numero dei caratteri della stringa ESCLUSO "\0"

ES



STRCPY

char * STRCPY (stringa destinazione, stringa copiare);

la copia qui

NB

copia "\0" in automatico!

STRNCPY

char * STRNCPY (stringa destinazione, stringa da copiare, u)

intero che ci dice quti caratteri di devo copiare in u

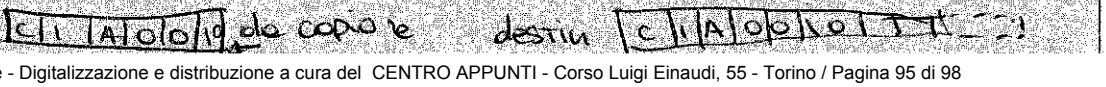
NB

"\0" non viene copiato se

(~~stringa dest, stringa~~ corta di ~~stringa~~)

stringa da copiare è + corta di u

ES u = 10



STRINGHE → #include <string.h>

- Sono VETTORI di CARATTERI ASCII
  ↓
  numeri
  lettere...

- ULTIMO CARATTERE è SEMPRE "\0"

PRINTF e SCANF si usano con %s

OCCHIO PERÒ:

▷ ignoreti caratteri iniziali di spaziatura
' ' '\n' '\t'

▷ termina la lettura quando incontra caratteri di spaziatura!

INIZIALIZZARE STRINGA:

CHAR nomestringa[lunghezza + 1] = {'s', 't', 'r', 'i', 'n', 'g', 'a', '\0'};
                                                                  ↑ \0

oppure

CHAR nomestringa[M] = "STRINGA"; (è lunga quanto M)

CHAR nomestringa[ ] = "STRINGA"; (è lunga quanto serve)