POLITECNICO DI BARI
DIPARTIMENTO DI ELETTROTECNICA ED ELETTRONICA

RACCOLTA DELLE ESERCITAZIONI DI FONDAMENTI DI INFORMATICA A.A. 2001/2002	STATO DELLE REVISIONI
ED. 04 REV. 00 DEL 19/04/2004	SOST. ED. 03 REV. 00 DEL 09/05/2003

RACCOLTA DELLE ESERCITAZIONI
per i corsi di

FONDAMENTI DI INFORMATICA[1]
INFORMATICA A[1]
FONDAMENTI DI INFORMNATICA II [2]

DIAGRAMMI DI FLUSSO E CODIFICA IN C DEGLI ALGORITMI

[1] INGEGNERIA MECCANICA , CIVILE*, ELETTRICA*
[2] INGEGNERIA INFORMATICA, DELL' AUTOMAZIONE
*SEDE DI FOGGIA

COPIA CONTROLLATA N. 7

STATO DELLE REVISIONI

DATA	REVISIONE	DESCRIZIONE	EMESSO	VERIFICATO	APPROVATO
21/03/2001	ED. 01 - REV. 01	EMISSIONE	PT	PT	PT
02/11/01	ED. 02 - REV. 00	EMISSIONE	PT	PT	PT
13/11/01	ED. 02 - REV. 01	EMISSIONE	PT	PT	PT
02/01/02	ED. 02 - REV. 02	EMISSIONE	PT	PT	PT
15/01/02	ED. 02 - REV. 03	EMISSIONE	PT	PT	PT
09/05/03	ED. 03 - REV. 00	EMISSIONE	PT	PT	PT
19/04/04	ED. 04 - REV. 00	EMISSIONE	PT	PT	PT

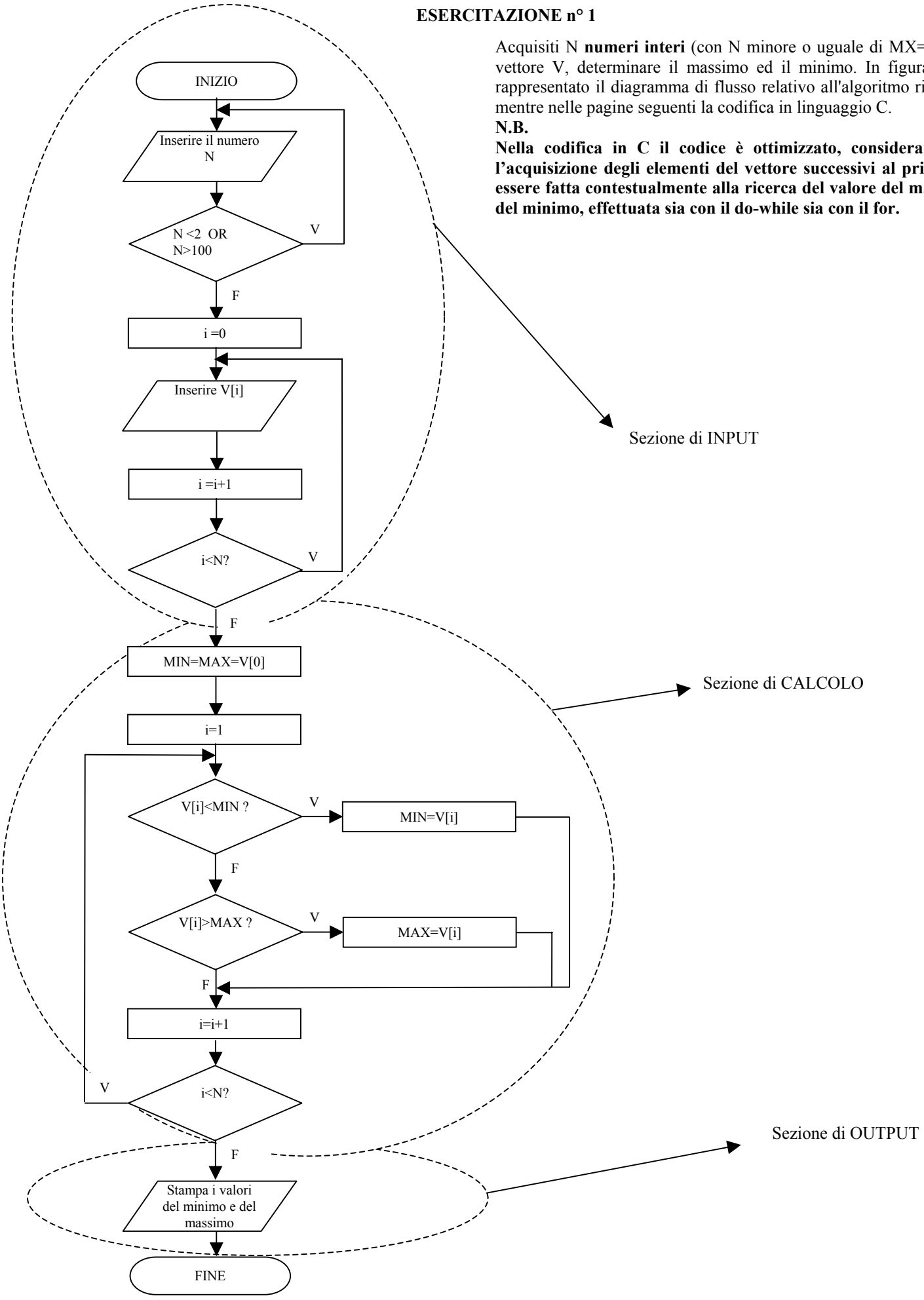
ELABORATO DA: PH.D. ING. V. BEVILACQUA TITOLARE (PT) DEL CORSO
VERIFICATO DA: PH.D. ING. V. BEVILACQUA TITOLARE (PT) DEL CORSO
EMESSO DA: PH.D. ING. V. BEVILACQUA TITOLARE (PT) DEL CORSO
APPROVATO DA: PH.D. ING. V. BEVILACQUA TITOLARE (PT) DEL CORSO

ESERCITAZIONE n° 1

Acquisiti N **numeri interi** (con N minore o uguale di MX=100) nel vettore V, determinare il massimo ed il minimo. In figura è stato rappresentato il diagramma di flusso relativo all'algoritmo risolutivo, mentre nelle pagine seguenti la codifica in linguaggio C.

N.B.

Nella codifica in C il codice è ottimizzato, considerando che l'acquisizione degli elementi del vettore successivi al primo, può essere fatta contestualmente alla ricerca del valore del massimo e del minimo, effettuata sia con il do-while sia con il for.



/* RICERCA DEL MINIMO E DEL MASSIMO UTILIZZANDO IL COSTRUTTO DO-WHILE */

```
#include <stdio.h>
#include <conio.h> /* solo Windows */
#define DIM_MAX 100
void main ()
{
      int i,N,MIN,MAX,V[DIM_MAX];
      char c;
do
      {
            do
            /* occorrono almeno due elementi perché abbia senso la ricerca
            del minimo e del massimo */
            {
                  printf("\n");
                  printf(" N = ");
                  scanf("%d",&N);
            }
            while (N<2 || N>DIM_MAX);
            printf("V[0]=");
            scanf("%d", &V[0]); /*inserisco solo il primo elem. del vettore*/
            MIN=MAX=V[0];
            i=1;
            do
            {
                  printf("V[%d]=",i);
                  scanf("%d", &V[i]);/*inserisco gli altri elem. del vettore*/
                  if (V[i] < MIN)
                        MIN = V[i];
                  else
                        if (V[i] > MAX)
                              MAX = V[i];
                  i = i+1;
            }
            while (i<N);
            /* output su monitor */
            printf (" Il massimo vale %d\n", MAX);
            printf (" Il minimo vale %d\n", MIN);
            /* opzione */
            printf(" Si vuole rieseguire l'algoritmo ? s/n ");}
            while (c= getch() != 'n');
}
```

/* RICERCA DEL MINIMO E DEL MASSIMO UTILIZZANDO IL CICLO FOR */

```
#include <stdio.h>
#include <conio.h>
#define DIM_MAX 100
void main ()
{
      int i,N,MIN,MAX,V[DIM_MAX];
      char c;
do
      {
            do
            {
                  printf("\n");
                  printf(" N = ");
                  scanf("%d",&N);
            }
            while (N<2 || N>DIM_MAX);
            printf("V[0]=");
            scanf("%d", &V[0]); /*inserisco solo il primo elem. del vettore*/
            MIN=MAX=V[0];
            for (i=1;i<N;i++)
            {
                  printf("V[%d]=",i);
                  scanf("%d", &V[i]);/*inserisco gli altri elem. del vettore*/
                  if (V[i] < MIN)
                        MIN = V[i];
                  else
                        if (V[i] > MAX)
                              MAX = V[i];
            }
            printf (" Il massimo vale %d\n", MAX);
            printf (" Il minimo vale %d\n", MIN);
            printf(" Si vuole rieseguire l'algoritmo ? s/n ");}
            while (c= getch() != 'n');
}
```

ESERCITAZIONE n° 2

```
/* CALCOLO DELLA MEDIA DI UNA SEQUENZA DI VALORI INTRODOTTI IN UN VETTORE */

#include <stdio.h>
#include <conio.h>
#define DIM_MAX 100
void main ()
{
      int i,N,V[DIM_MAX],SOMMA;
      float MEDIA;
      char c;
do
{
do
            {
         printf("\n");
                  printf(" N = ");
                  scanf("%d",&N);
            }
while (N<2 || N>DIM_MAX);
SOMMA = 0;
for (i=0;i<N;i++)
{
printf("V[%d]=",i);
scanf("%d", &V[i]);/*inserisco gli altri elem. del vettore*/
SOMMA = SOMMA + V[i];
}
MEDIA = (float)SOMMA/N;/* cast della variabile somma */
printf (" La media vale %f\n", MEDIA);
/* le precedenti due righe di codice possono essere sostituite dalla seguente :
printf (" La media vale %f\n", (float)SOMMA/N); */

printf("\n");
printf(" Si vuole rieseguire l'algoritmo ? s/n ");}
            while (c= getch() != 'n' );

/* output su stampante standard < stdprn >
se si aggiunge la seguente riga di codice la stampa avviene su stampante
fprintf(stdprn, "media = %f\n", MEDIA); */
}
```

ESERCITAZIONE n° 3

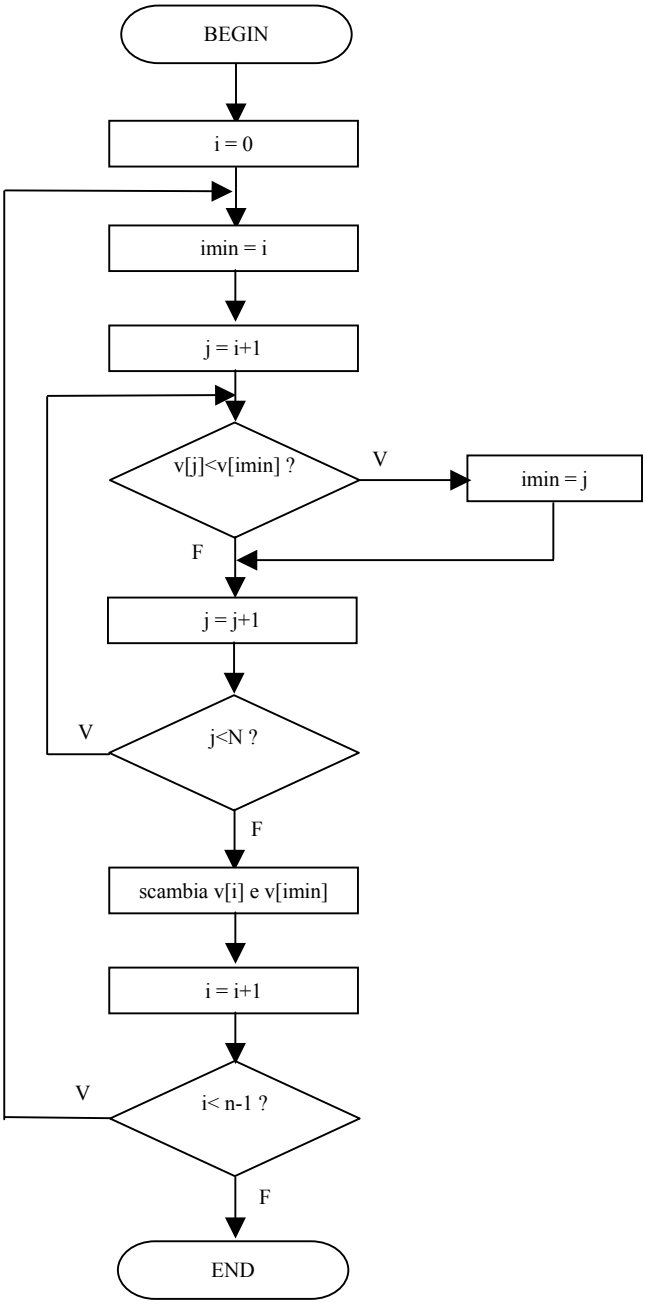
```
/* ALGORITMO DEL FATTORIALE – iterativo */

#include <stdio.h>
#include <conio.h>

void main()
{
   int N,AUX;
   float FAT;
   do
   {
    do
  {
  printf("Inserire il numero N non negativo di cui si vuole il fattoriale : ");
  scanf("%d",&N);
  }
    while(N < 0);
    FAT=1;
    for (AUX=1;AUX<=N;AUX++)
    FAT=FAT*AUX;
    printf(" Il fattoriale di %d vale %.0f \n", N, FAT);
printf(" Vuoi rieseguire il programma? s/n \n");
}
while(getch()!='n');
}
```

/* ALGORITMO DEL FATTORIALE – ricorsivo (dopo aver studiato le funzioni)*/

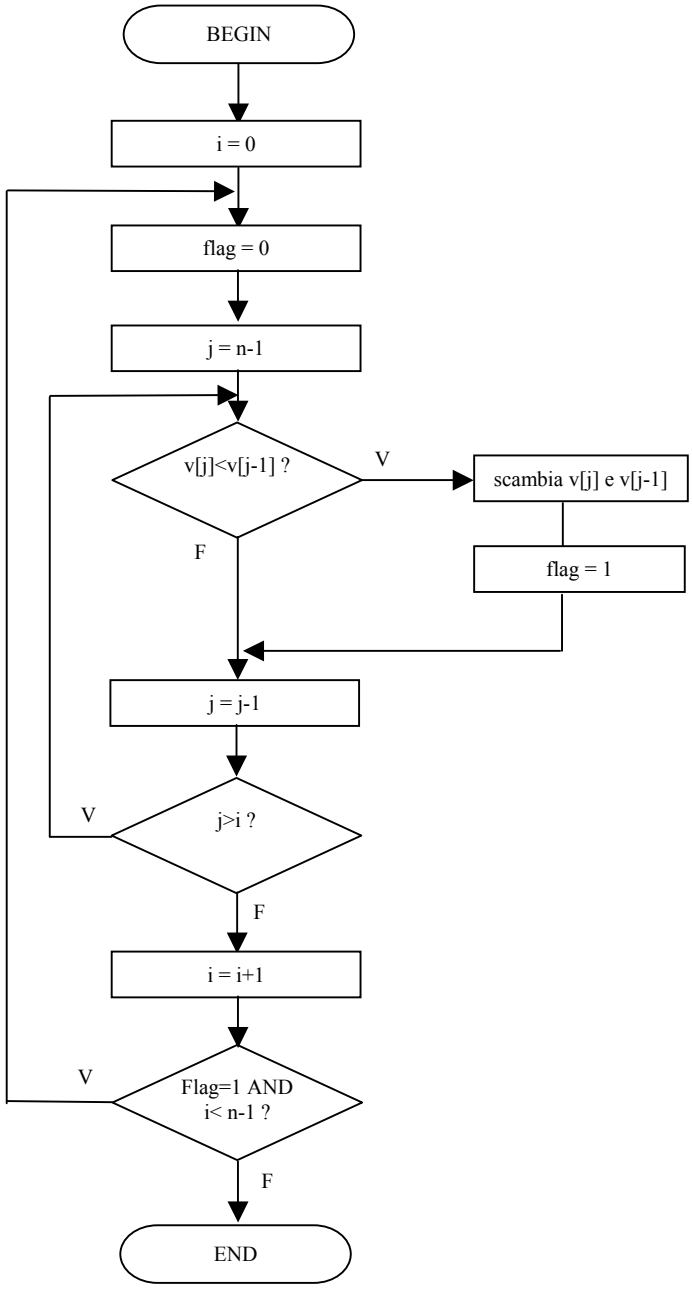
```
#include <stdio.h>
#include <conio.h>

int fatt(int);
void main()
{
   int N;
   do
   {
    do
      {
      printf("Immettere il numero di cui si vuole il fattoriale \n");
      scanf("%d",&N);
      }
    while(N <0);
         printf("il fattoriale di %d vale %d\n", N, fatt(N));
         printf("rieseguire?\n");
}
while(getch()!='n');
}

int fatt (int X)
{
if (X==0)
return (1);
else
return(X*fatt(X-1));
}
```

ESERCITAZIONE n° 4

Ordinamento con il metodo di selezione



Ordinamento con il metodo Bubble Sort



/* ORDINAMENTO DI UN VETTORE */

```
#include <stdio.h>
#include <conio.h>
#define DIM_MAX 100
void main()
{
   int N,i,imin,j,temp,V[DIM_MAX];
   do
   {
    do
      {
      printf("Immettere dimensione del vettore N= : ");
      scanf("%d",&N);
      }
    while(!(N >= 2 && N <= 100));
    printf(" Bravo, ci devono essere almeno due elementi \n");
for (i=0;i<N;i++)
  {
  printf(" Introduci V[%d] = ", i);
  scanf("%d",&V[i]);
  }
for (i=0;i<N-1;i++)
{
      imin=i;
      for (j=i+1;j<N;j++)
       if (V[j]<V[imin])
            imin=j;
       temp=V[i];
       V[i]=V[imin];
       V[imin] = temp;
}
printf("il vettore ordinato: \n");
for (i=0;i<N;i++)
      printf(" V[%d] = %d \n", i, V[i]);
printf(" Vuoi rieseguire il programma? s/n \n");
}
while(getch()!='n');
}
```

ESERCITAZIONE n° 5

/* ORDINAMENTO PER FUSIONE DI DUE VETTORI ORDINATI */

```
#include <stdio.h>
#include <conio.h>
#define DIM_MAX_V 100
#define DIM_MAX_W 50
#define DIM_MAX_Z DIM_MAX_V+DIM_MAX_W
void main()
{
   int N,M,i,j,k,V[DIM_MAX_V], W[DIM_MAX_W], Z[DIM_MAX_Z];
   do
   {
    do
      {
      printf("Immettere dimensione del vettore V : ");
      scanf("%d",&N);
      }
    while(!(N >= 2 && N <= DIM_MAX_V));
    for (i=0;i<N;i++)
  {
  printf(" Introduci V[%d] = ", i);
  scanf("%d",&V[i]);
  }
   do
      {
      printf("Immettere dimensione del vettore W : ");
      scanf("%d",&M);
      }
    while(!(M >= 2 && M <= DIM_MAX_W));
    for (i=0;i<M;i++)
  {
  printf(" Introduci W[%d] = ", i);
  scanf("%d",&W[i]);
  }
  /* fusione dei due vettori ordinati */
  i=0; j=0; k=0;
  do
  {
  if (V[i]<W[j])
  Z[k++]=V[i++];
  else
```

```
if (V[i]==W[j])
  {
  Z[k++]=V[i++];
  j++;
  }
  else
  Z[k++]=W[j++];
  }
  while(i<N && j<M);
   if(i<N)
      for(;i<N;Z[k++]=V[i++])
      ;
      else
      for(;j<M;Z[k++]=W[j++])
      ;
  for (i=0;i<k;i++)
      printf(" Z[%d] = %d \n", i, Z[i]);
printf(" Vuoi rieseguire il programma? s/n \n");
}
while(getch()!='n');
                                          }
```

ESERCITAZIONE n° 6

```
/* PRODOTTO DI DUE MATRICI */

#include <stdio.h>
#include <conio.h>
#define MAXRIGHE_A  10
#define MAXCOLONNE_A  10
#define MAXRIGHE_B MAXCOLONNE_A
#define MAXCOLONNE_B  10
#define MAXRIGHE_P  MAXRIGHE_A
#define MAXCOLONNE_P MAXCOLONNE_B
void main (void)
{
int i,j,k,N,M,Q,P;
int MA[MAXRIGHE_A][MAXCOLONNE_A],MB[MAXRIGHE_B][MAXCOLONNE_B];
int MP[MAXRIGHE_P][MAXCOLONNE_P];
do
{
      do    {
      printf("Inserisci il numero di righe della matrice A, N=");
      scanf("%d",&N);
            } while(N<2||N>MAXRIGHE_A);
      do    {
      printf("Inserisci il numero di colonne della matrice A, M=");
      scanf("%d",&M);
            } while(M<2||M>MAXCOLONNE_A);
      do    {
            printf("Inserisci il numero di righe della matrice B, Q=");
            scanf("%d",&Q);
            } while(Q!=M);
      do    {
            printf("Inserisci il numero di colonne della matrice B, P=");
            scanf("%d",&P);
            } while(P<2||P>MAXCOLONNE_B);
printf("elementi della prima matrice\n");
      for(i=0;i<N;i++)
         for(j=0;j<M;j++)
            {
                  printf("elemento MA[%d][%d] =",i+1,j+1);
                  scanf("%d",&MA[i][j]);
            }
printf("elementi della seconda matrice\n");
      for(i=0;i<M;i++)
      for(j=0;j<P;j++)
```

```
            {
            printf("elemento MB[%d][%d]=",i+1,j+1);
            scanf("%d",&MB[i][j]);
            }
/* calcolo della matrice prodotto */     for(i=0;i<N;i++)

      for(j=0;j<P;j++)
                                                                  {

      MP[i][j] = 0;

      for(k=0;k<M;k++)

      MP[i][j]= MP[i][j]+MA[i][k]*MB[k][j];
                                                                  }
printf("\n \n PRIMA MATRICE \n");         for(i=0;i<N;i++)
                                                      {
                                                      printf(" \n");
                                                      for(j=0;j<M;j++)
                                                      printf(" %5d ",MA[i][j]);
                                                      }
printf("\n \n SECONDA MATRICE \n");       for(i=0;i<M;i++)
                                                      {
                                                      printf(" \n");
                                                      for(j=0;j<P;j++)
                                                      printf(" %5d ",MB[i][j]);
                                                      }
printf (" \n \n MATRICE PRODOTTO \n");    for(i=0;i<N;i++)
                                                      {
                                                      printf(" \n");
                                                      for(j=0;j<P;j++)
                                                      printf(" %5d ",MP[i][j]);
                                                      }
printf(" \n \n Vuoi rieseguire? s/n \n");
}
while(getch()!= 'n');
}
```

ESERCITAZIONE n° 7

/* RICERCA DICOTOMICA IN UN VETTORE ORDINATO */

```
#include <stdio.h>
#include <conio.h>
#define DIM_MAX_V 2000

void main()
{
   int N,ELE,POS,SINISTRO,DESTRO,i,V[DIM_MAX_V], passi;
   do
   {
    do
      {
      printf("Immettere dimensione del vettore V : ");
      scanf("%d",&N);
      }
    while(!(N >= 2 && N <= DIM_MAX_V));
    for (i=0;i<N;i++)
      {
      V[i]=i+1;
      printf("V[%d]=%d \n",i,V[i]);
      }
      printf("Immettere elemento da ricercare ele=  ");
      scanf("%d",&ELE);
/* ricerca dicotomica */
SINISTRO=0; DESTRO=N-1; POS=-1, passi=0;
      do
      {
      passi++;
      i=(SINISTRO+DESTRO)/2;
      if (V[i]==ELE)
         POS=i;
      else
          if(V[i]<ELE)
          SINISTRO=i+1;
          else
          DESTRO=i-1;
      }
      while(SINISTRO<=DESTRO && POS == -1);
```

```
      if (POS != -1)
      {
      printf(" %d si trova in posizione %d \n", ELE, POS+1);
      printf(" risposta fornita in passi %d \n", passi);
      }
      else
      {
      printf(" %d non presente nel vettore \n", ELE);
      printf(" risposta fornita in passi %d \n", passi);
      }
     printf(" Vuoi rieseguire il programma? s/n \n");
}
while(getch()!='n');
}
```

ESERCITAZIONE n° 8

```
/* ALGORITMO DELLA POTENZA DI UN NUMERO */

#include <stdio.h>
#include <conio.h>
main( )
{int i, E, Base, Esponente;
float Potenza;
do
{
Potenza=1;
printf ("Inserisci Base: ");
scanf ("%d", &Base);
printf ("Inserisci Esponente: ");
scanf ("%d", &Esponente);
            if (Esponente>0)
            E= Esponente;
       else
             E= -1*Esponente;
                   for (i=1; i<=E; i++)
             Potenza = Potenza * Base;
        if (Esponente<0)
             Potenza = 1/Potenza;
printf ("La Potenza vale %f \n", Potenza);
printf("vuoi rieseguire ? \n");
}
while(getch()!='n');
}
```

/* VISUALIZZAZIONE DEI CARATTERI DI UNA STRINGA */

```
#include <stdio.h>
#include <conio.h>
void main()
{
   char frase[]= " oggi studiamo le stringhe ";
   int i=0;
   while(frase[i] != '\0')
   {
   printf(" %c = %d %x \n", frase[i], frase[i], frase[i]);
   i++;
   }
   printf("%s", frase);
   getch();
}
```

/* COPIA DI UNA STRINGA SU UN'ALTRA CON O SENZA FUNZIONE STRCPY */

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main()
{
   char originale[]= " oggi studiamo le stringhe ";
   /*int i=0;*/
   char copia[80];
   strcpy(copia,originale);
   /*
   for(i=0; (copia[i]=originale[i]) != '\0'; i++)
   ;
   */
   printf("%s %s", originale, copia);
   getch();
}
```

/* CONCATENAZIONE DI DUE STRINGHE */

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main()
{
   char cognome[80],nome[160];
   int i,j;
   printf("Inserisci il tuo cognome: ");
   scanf("%s",cognome);
```

```
   printf("Inserisci il tuo nome: ");
   scanf("%s",nome);
   for(i=0; (nome[i]) != '\0'; i++)
   ;
   printf("Il tuo nome %s ha %d letterere \n", nome,i);
   nome[i]= ' ';   /* inseriamo uno spazio fra nome e cognome */
   i++;
   for (j=0; (nome[i]=cognome[j]) != '\0'; i++,j++)
   ;
   printf("Il tuo cognome %s ha %d lettere \n", cognome,j);
   printf("nome e cognome: %s \n",nome);
   printf("Il tuo nome e cognome hanno bisogno di %d caratteri",i);
   if(i>20)
   printf(", accidenti");
   getch();
}
```

/* CONCATENAZIONE DI DUE STRINGHE CON FUNZIONE STRCAT */

```
#include <stdio.h>
#include <string.h>
void main()
{
   char cognome[80],nome[160];
   int i,j;
   printf("Inserisci il tuo cognome: ");
   scanf("%s",cognome);
   printf("Inserisci il tuo nome: ");
   scanf("%s",nome);
   strcat (nome,cognome); /* nessuno spazio fra nome e cognome */
   printf("%s", nome);
}
```

/* CONFRONTO ALFABETICO FRA STRINGHE */

```
#include <stdio.h>
#include <conio.h>
void main()
{
   char parola1[80],parola2[160];
   int i;
   printf("Inserisci la prima parola: ");
   scanf("%s",parola1);
   printf("Inserisci la seconda parola: ");
   scanf("%s",parola2);
for(i=0;(parola1[i]==parola2[i])&&(parola1[i]!='\0')&&(parola2[i]!='\0');i++)
```

```
   ;
   if (parola1[i]==parola2[i])
   printf(" %s e %s sono uguali \n",parola1, parola2);
   else
   {
   if (parola1[i]>parola2[i])
   printf(" %s precede in ordine alfabetico %s \n",parola2, parola1);
   else
   printf(" %s precede in ordine alfabetico %s \n",parola1, parola2);
   }
   getch();
   }
```

/* CONFRONTO ALFABETICO CON LA FUNZIONE STRCMP DEL FILE HEADER STRING.H */

```
#include <stdio.h>
#include <string.h>
void main()
{
   char parola1[80],parola2[160];
   int i,test;
   printf("Inserisci la prima parola: ");
   scanf("%s",parola1);
   printf("Inserisci la seconda parola: ");
   scanf("%s",parola2);
   if( (test = (strcmp(parola1, parola2))) == 0)
   printf(" %s e %s sono uguali \n",parola1, parola2);
   else
   {
   if (test>0)
   printf(" %s precede in ordine alfabetico %s \n",parola2, parola1);
   else
   printf(" %s precede in ordine alfabetico %s \n",parola1, parola2);
   }
   }
```

/* CONVERSIONE DA BASE 16 A BASE 10*/

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
   char codice[80],cifra[80];
   int i,j,k;
   do
   {
   float risultato = 0;
   printf(" \n Inserisci il codice esadecimale: ");
   scanf("%s",codice);
   for(i=0; ((codice[i]) != '\0'&& (codice[i]) != '.'); i++)
   {
   if(codice[i]>=48&&codice[i]<=57)
   cifra[i]= codice[i]-48;
   else
   if (codice[i]>=65&&codice[i]<=70)
   cifra[i]=codice[i]-55;
   }
   j=i;
   i=i+1;
   for(; (codice[i]) != '\0'; i++)
   {
   if(codice[i]>=48&&codice[i]<=57)
   cifra[i-1]= codice[i]-48;
   else
   if (codice[i]>=65&&codice[i]<=70)
   cifra[i-1]=codice[i]-55;
   }
   for(k=0;k<i-1;k++)
   risultato= risultato+(cifra[k]*pow(16,(j-k-1)));
   printf(" valore decimale = %f",risultato);
   printf(" \n vuoi rieseguire ?");
   }
   while(getch()!= 'n');
}
```

ESERCITAZIONE n° 9

```
/* INTRODUZIONE ALLE FUNZIONI */

/* esempio di passaggio di risultati usando variabili globali */
/* funzione di scambio /

#include <stdio.h>
#include <conio.h>

int primoglobale, secondoglobale; /* variabili globali */

void scambia (int,  int);
/* dichiarazione della funzione, fra parentesi ci sono i tipi degli argomenti /

void main ()
{
    int primolocale, secondolocale; /* variabili locali al main */
  do
      {
printf ( "immetti due numeri:\n" );
scanf ( "%d%d", &primolocale, &secondolocale );
printf ("nell'ordine iniziale prima %d poi %d \n", primolocale, secondolocale );
scambia (primolocale, secondolocale);
/* la funzione scambia viene chiamata ed utilizza come parametri attuali, variabili
locali */
printf ( "nell'ordine finale prima %d poi %d \n", primoglobale, secondoglobale );
printf("vuoi rieseguire? s/n");
}
 while (getch() != 'n');
    }

void scambia(int a, int b) /* definizione della funzione */
/* tra parentesi ora le variabili formali con i loro tipi */
/* funzione che restituisce in "primoglobale" (globale) il secondolocale ,
 * ed in "secondoglobale" (globale) il primolocale */
{
   primoglobale=b;
   secondoglobale=a;
}
```

/* Esempio di funzioni definite dall'utente : parametri passati per valore*/

```
#include <stdio.h>
float a,b,c,d; /* variabili globali utilizzate dal main ma distinte dalle
*omonime locali;*/
float delta(float, float, float);

void main()
      {
            printf("\na=");
            scanf("%f",&a);
            printf("\nb=");
            scanf("%f",&b);
            printf("\nc=");
            scanf("%f",&c);
            d=delta(a,b,c);
            printf("delta=%f",d);
      }

float delta(float a, float b, float c)
/* a,b e c sono gli argomenti della funzione: valgono localmente
*la funzione che si è definita deve restituire un float */
      {
            float d; /* variabile locale */
            d=b*b-4*a*c;
            return d; /* il valore viene restituito alla funzione */
      }
```

/* esempio di funzione con risultato void:
prende come argomento un numero intero, e lo stampa in base due */

```
#include <stdio.h>

void stampaBaseDue ( int );

void main ()
{
  int num;
  printf ("inserisci intero positivo: ");
  scanf ("%d", &num );

  stampaBaseDue (num);
}
```

```
void stampaBaseDue (int n)
{
  int i=0, cifreBin[80];
  while (n>1) {
      cifreBin[i] = n%2;
      n = n/2;
      i++;
  }
  /* usciti dal ciclo, n e' 0 oppure 1 */
  cifreBin[i] = n;

  /* stampa l'array alla rovescia */
  for ( ; i>=0; i--)
      printf("%d",cifreBin[i]);
  printf ("\n");
}
```

ESERCITAZIONE n° 10

/* INTRODUZIONE AI PUNTATORI */

```
int a=5; /* dichiarazione di una variabile intera e contemporanea inizializzazione*/
int *p; /* dichiarazione di una variabile di tipo puntatore ad intero */
/* in questo caso il "simbolo" * ha solo valore sintattico in quanto specifica che
tipo di variabile si sta dichiarando: il puntatore è p */
p=&a; /* il puntatore p è inizializzato con l'indirizzo della variabile a  */
/* a questo punto a e *p costituiscono il cosiddetto alias entrambi valgono 5 */

Es: printf(" la varibile intera a che si trova all'indirizzo %p vale %d ",p,*p);
```

/* Esempio di funzioni definite dall'utente : parametri passati per indirizzo */

```
#include <stdio.h>
void scambia(int *, int *);
/* dichiarazione: gli argomenti sono di tipo puntatore */

void main()
      {
            int a,b;
            printf("a e b valgono rispettivamente\n");
            scanf("%d %d",&a,&b);
            scambia(&a,&b);
/* la funzione scambia è chiamata passandole gli indirizzi */
            printf("a=%d e b=%d\n",a,b);
      }

void scambia(int *px, int *py)
```

/* visto che px e py sono variabili formali di tipo puntatore deputate a contenere indirizzi possono contenere &a &b e quindi si può effettuare lo scambio lavorando sui valori reali di a e di b e non sulle loro copie per mezzo dell'operatore unario di dereferenziazione. In questo caso, infatti, px e py contengono una copia degli indirizzi di a e b ma *px e *py sono gli originali e non le copie di a e b */

```
      {
            int temp;
            temp = *px;
            *px = *py;
            *py = temp;
      }
```

/* la funzione calcolaMedia non restituisce risultati, essi sono memorizzati nelle variabili di cui è passato l'indirizzo */

```
#include <stdio.h>
#include <math.h>
#include <conio.h>

void calcolaMedie (int, int, float *, float *);

void main ()
{
    int x, y;
    float ma, mg; /* media aritmetica e media geometrica */
    do
    {
    printf ("immetti due interi: \n");
    scanf ("%d%d", &x, &y);
    calcolaMedie (x, y, &ma, &mg);
    printf ("media aritmetica = %.2f\nmedia geometrica = %.2f\n", ma, mg);
    printf("rieseguire?");
    }
    while(getch()!='n');
}

void calcolaMedie (int a, int b, float *puntMarit, float *puntMgeom)
{
    *puntMarit = (a + b)/2.0;  /* 2.0 per avere divisione frazionaria
                                * e non divisione intera */
    if (a*b < 0)
     *puntMgeom = -1;
    else
     *puntMgeom = sqrt (a * b);
}
```

```
/* gestione di una sequenza */
#include <stdio.h>
#define MAX_ELE 100 /* numero massimo di elementi */

int leggiScelta (void);
int leggiVett (int *); /* legge da ingresso i valori,
                        * memorizza i valori in un vettore, e restituisce il numero
di elementi */
void stampaVett (int, int *);

void main ()
{
    int scelta, dimensione, vettore [ MAX_ELE ];
    do {
     scelta = leggiScelta (); /* utilizzo di switch case, con break e default */
     switch (scelta) {
         case 1: dimensione = leggiVett (vettore);
/* vettore è il puntatore a vettore[ MAX-ELE] vettore coincide con &vettore[0]; */
              break;
         case 2: stampaVett (dimensione, vettore);
                 break;
         case 0: printf ("\nfine operazioni\n");
                 break;
           default: printf("avevo detto 0 1 oppure 2 \n");
     }
    } while (scelta != 0);

}

int leggiScelta (void)
{
    int scelta;
    printf ("menu principale\n");
    printf ("\t 0: fine\n\t 1: immissione\n\t 2: stampa\n\t scelta: ");
    scanf ("%d", &scelta);
    return scelta;
}

int leggiVett (int *vet)
{
    int i, n;
    printf ("quanti elementi? ");
    scanf ("%d", &n);
    printf ("immetti %d valori:\n", n);
```

```
    for (i=0; i<n; i++)
     scanf ("%d", &vet[i]);
    return n;
}

void stampaVett (int n, int *vet)
{
    int i;
    printf ("sequenza in memoria:\n");
    for (i=0; i<n; i++)
     printf ("%d ", vet[i]);
    printf ("\n");
}
```

ESERCITAZIONE n° 11

```
/* UTILIZZO DI FILES e modalità "r", "w", "a" */
/* lettura e scrittura utilizzando files e funzioni */

    #include <stdio.h>
    #include <conio.h>
    #define N_MAX_VETT 9
    void leggiVet (FILE *, int, int *); /* dichiarazione */
    void azzeraSottoMedia (int, int *); /* dichiarazione */
    void stampaVet (int, int *); /* dichiarazione */
    void scriviVet (FILE *, int,int *); /* dichiarazione */
    void main ()
    { FILE  *fileinput, *fileoutput;
      int  vet[N_MAX_VETT],N;
      char  fileingresso[12], fileuscita[12];
      do
       {
       do
       {  printf("\n");
       printf ("nome file di dati di ingresso: ");
       scanf ("%s", &fileingresso);
       fileinput = fopen(fileingresso,"r");/*creazione file di ingresso in lettura*/
       if  (fileinput == NULL)
       printf("Il file %s non esiste \n", fileingresso); }
       while (fileinput == NULL);
       printf ("nome file di dati di uscita: ");
       scanf ("%s", &fileuscita);
       fileoutput =fopen (fileuscita, "a"); /*creazione file di uscita in
    scrittura*/
       do  {
       printf("Inserisci dimensione del vettore N=");
       scanf("%d",&N);}
       while(N<2||N>N_MAX_VETT);
      leggiVet (fileinput, N, vet);  /* legge il vettore dal file */
      printf ("Vettore iniziale:\n");
      stampaVet (N, vet);
      /* QUI E' CHIAMATA LA FUNZIONE, USANDO N E vet COME PARAMETRI ATTUALI */
      azzeraSottoMedia (N, vet);
      printf ("Vettore con componenti sotto la media azzerate:\n");
      stampaVet (N, vet);
      scriviVet (fileoutput, N, vet);
      fclose (fileinput);
      fclose (fileoutput);
```

```
  printf("Vuoi rieseguire? s/n ");
  }
  while(getch()!= 'n');
  }

void azzeraSottoMedia (int n, int *v) /* lettura e stampa di vettore */
{   int i;
    float media = 0;
    for (i=0; i<n; i++)
       media += v[i];
    media = media/n;
    for (i=0; i<n; i++)
       if (v[i] < media)
          v[i] = 0; }
void leggiVet (FILE *f, int n, int *v)/* n il numero di interi da leggere da f*/
{   int  i;
    for (i=0; i<n; i++)
      fscanf (f, "%d", &v[i]); }
void scriviVet (FILE *f, int n, int *v)/* n il num. di interi da scrivere in f*/
{   int  i;
fprintf (f, "se leggi %d elementi del vettore \n",n);
   for (i=0; i<n; i++)
      fprintf (f, "%d", v[i]);
   fprintf (f, "\n");
     }
void stampaVet (int n, int *v)
{   int i;
    for (i=0; i<n; i++)
      printf ("%d ", v[i]);
    printf ("\n"); }
```

/* esempio sul campo d'azione, visibilità e mascheramento - cosa stampa? */

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

void p1 (void);
void p2 (void);
int x = 1; /* x globale */
int z; /* z globale */

void main ()
{
    do
    {
    int y=10;
    printf(" \n");
    printf ( "prima stampa di x di main: %d\n", x );  /* stampa 1 */
    printf ( "prima stampa di y di main: %d\n", y );  /* stampa 10 */
    printf ( "prima stampa di z di main: %d\n", z );  /* stampa 0 */
      p2 ();
    printf ( "seconda stampa di x di main: %d\n", x ); /* stampa 4 */
    printf ( "seconda stampa di y di main: %d\n", y ); /* stampa 10 */
      printf ( "seconda stampa di z di main: %d\n", z ); /* stampa 30 */
    printf(" Vuoi rieseguire?");
    }
    while(getch()!= 'n');
}


void p1 (void)
{
    int y;
    x = 4; /* ? quale x cambia p1 ? */
    y=  40;   /* p1 non cambia nessuna y */
      z=15;
    printf ( "stampa x in p1: %d\n", x );    /* stampa 4 */
    printf ( "stampa y in p1: %d\n", y );    /* stampa 40 */
      printf ( "stampa z in p1: %d\n", z );    /* stampa 15 */
}

void p2 (void)
{

      int y=  20; /* y di p2 non maschera la y locale al main */
```

```
    p1 ();
      z=30;
    printf ( "stampa x in p2: %d\n", x );     /* stampa 4 */
    printf ( "stampa y in p2: %d\n", y );     /* stampa 20 */
      printf ( "stampa z in p2: %d\n", z );     /* stampa 30 */
}
```

ESERCITAZIONE n° 12

/* esempio di matrice passata per indirizzo ad una funzione: quando una matrice array a due dimensioni), viene passata ad una funzione ciò che viene passato è il puntatore al suo primo elemento [0][0]. La funzione che riceve la matrice deve specificare obbligatoriamente la seconda dimensione (quella più a destra), affinchè il compilatore conosca la lunghezza delle righe della matrice stessa */

```
#include <stdio.h>
#include <conio.h>
#define MAXRIGHE  10
#define MAXC  10

void insmatrice (int,int,int (*)[MAXC]);
void visualizzamatrice  (int,int,int (*)[MAXC]);
void xmat(int,int,int,int (*)[MAXC],int (*)[MAXC], int (*)[MAXC]);

void main (void)
{
int N,M,P,Q;
int MA[MAXRIGHE][MAXC],MB[MAXRIGHE][MAXC],MP[MAXRIGHE][MAXC];
do
{
      do
   {
      printf("Inserisci il numero di righe della matrice A, N=");
      scanf("%d",&N);
      }
    while(N<2||N>MAXRIGHE);
      do
   {
      printf("Inserisci il numero di colonne della matrice A, M=");
      scanf("%d",&M);
      }
   while(M<2||M>MAXC);
      do
   {
            printf("Inserisci il numero di righe della matrice B, Q=");
            scanf("%d",&Q);
      }
   while(Q!=M);
      do
      {
            printf("Inserisci il numero di colonne della matrice B, P=");
```

```
            scanf("%d",&P);
            }
    while(P<2||P>MAXC);
      printf("\n");
      printf("Inserisci la prima matrice \n");
      insmatrice(N,M,MA);
      printf("Inserisci la seconda matrice \n");
      insmatrice(M,P,MB);
      printf("prima matrice \n");
      visualizzamatrice  (N,M,MA);
      printf("seconda matrice \n");
      visualizzamatrice  (M,P,MB);
      xmat(N,M,P,MA,MB,MP);
      printf("matrice prodotto \n");
      visualizzamatrice  (N,P,MP);
      printf(" \n \n Vuoi rieseguire? s/n \n");
}
while(getch()!= 'n');
}

void insmatrice (int a, int b, int (*matrice)[MAXC])
{
      printf("\n");
   int i,j;
      for(i=0;i<a;i++)
         for(j=0;j<b;j++)
            {
                  printf("elemento [%d][%d] =",i+1,j+1);
                  scanf("%d",&matrice[i][j]);
            }
            printf(" \n ");
      }

void xmat(int a,int b,int c,int (*mat1)[MAXC],int (*mat2)[MAXC],int
(*mat3)[MAXC])
{
int i,j,k;
/* calcolo della matrice prodotto */
                              for(i=0;i<a;i++)
                                    for(j=0;j<c;j++)
                                    {
                                    mat3[i][j] = 0;
                                    for(k=0;k<b;k++)
```

```
                                    mat3[i][j]=
mat3[i][j]+mat1[i][k]*mat2[k][j];
                                    }
}

void visualizzamatrice  (int a, int b,int (*matrice)[MAXC])
{
  int i,j;
                        for(i=0;i<a;i++)
      {
                              printf("\n");
            for(j=0;j<b;j++)
            printf(" %5d ",matrice[i][j]);
      }
                           printf( " \n \n");
}
```

ESERCITAZIONE n° 13

```
/* TIPI DEFINITI DALL'UTENTE: LE STRUTTURE */

#include <stdio.h>
#include <conio.h>
#define frequentanti 165

struct Studente {
char nome[20], cognome[20],matricola[9];
int voto;
}; /* dichiarazione globale della struttura */

void main (void)
{

int i;

struct Studente meccanica[frequentanti]; /* Studente può tipizzare l'array
meccanica[] */
do
{
      for (i=0;i<frequentanti;i++)
      {
      printf("Inserisci il nome ");
      scanf("%s",meccanica[i].nome);
/* l'operatore punto per accedere al campo */
      printf("Inserisci il cognome ");
      scanf("%s",meccanica[i].cognome);
      printf("Inserisci la matricola ");
      scanf("%s",meccanica[i].matricola);
      printf("Inserisci il voto ");
      scanf("%d",&meccanica[i].voto);
      }
printf("Elenco degli ammessi \n");
for (i=0;i<frequentanti;i++)
if(meccanica[i].voto>15)
printf("%s %s %s %d \n", meccanica[i].nome, meccanica[i].cognome,
meccanica[i].matricola, meccanica[i].voto);
printf("rieseguire?");
}
while (getch()!='n');
}
```

```
/* ESEMPIO DI STUTTURE COME ARGOMENTI DI FUNZIONI */

#include <stdio.h>
#include <conio.h>

struct tm
{
 int ore;
 int minuti;
 int secondi;
};

void mostra( struct tm *);
void aggiorna( struct tm *);

void main(void)
{
 struct tm tempo;
int i;
 tempo.ore=0;
 tempo.minuti=0;
 tempo.secondi=0;

 for(i=0;i<86400;i++)
 {
 aggiorna(&tempo);  /* &tempo.secondi è l'indirizzo del campo secondi della
struttura tempo */
 mostra(&tempo);
 }
 getch();
 }

void aggiorna( struct tm *t)
/* struct tm *t dichiara formalmente un puntatore t ad una struttura di tipo tm */
{
t->secondi++;

/* l'operatore freccia (segno di meno seguito da segno di maggiore) viene usato per fare
riferimento ad una struttura tramite puntatore infatti t->secondi++; equivale a
(*t).secondi + 1; */

if(t->secondi==60)
{
t->secondi=0;
```

```
t->minuti++;
}
if(t->minuti==60)
{
t->minuti=0;
t->ore++;
}
if(t->ore==24)
{
t->ore=0;
}
}
void mostra(struct tm *t)
{
printf("%02d:",t->ore);
printf("%02d:",t->minuti);
printf("%02d:",t->secondi);
printf("\n");
}
```

/* Operatorio ternario ? */

```
y=(x>3) ? 0:1; ⇔ if (x>3) y=0; else y=1;

#include <stdio.h>

void main (void)
{
int i,j;
printf("inserisci i e j\n");
scanf("%d%d",&i,&j);
printf(" i vale %d\n j vale %d\n",i,j);
(i>j) ? printf(" i maggiore di j\n") : printf(" i minore di j\n");
}
```

ESERCITAZIONE n° 14

/* Le strutture informative: la pila, la coda, lista, i grafi */

```
/* pila realizzata con vettore */

#include<stdio.h>
#include <conio.h>
#include<stdlib.h>
#define lung 5

void inserisci(int *, int *, int);
void estrai(int *, int *, int *);
void visualizza(int *, int);

void main(void)
{
int elem,punt_testa=0;
int scelta =-1;
int pila[lung];
while(scelta != 0)
{
printf("scelta = 1 per inserire, 2 per estrarre, 3 per visualizzare, 0 per
terminare \n");
scanf("%d", &scelta);
printf("\n");
switch (scelta)
{
case 1:
     {
     if (punt_testa>=lung)
     printf("pila piena \n");
     else
     {
     printf("elemento da inserire =");
     scanf("%d",&elem);
     inserisci(pila,&punt_testa, elem);
     printf("\n");
     }
     }
     break;
case 2 :
     {if (punt_testa==0)
```

```
     printf("pila vuota \n");
     else
     {
     estrai(pila,&punt_testa,&elem);
     printf("%d eliminato \n", elem);
     }
     }
     break;
case 3: visualizza(pila, punt_testa);
     break;
case 0 : printf("fine\n");
     break;
default: printf("non ci sono altre operazioni definite su una pila\n");
     break;
}
}
}

void inserisci(int *vettore, int *p, int elemento)
{
vettore[*p]=elemento;
(++*p);
}
/*
N.B. se inserisco nell'ordine nella pila prima 1 poi 2 poi 3 poi 4 e poi 5
si ha: vettore[0]=1; vettore[1]=2; vettore[2]=3; vettore[3]=4; vettore[4]=5;
*/

void estrai(int *vettore, int *p, int *elemento)
{
*elemento=vettore[--*p];
}

void visualizza (int *vettore, int p)
{
while(p>=1)
printf("%d \n", vettore[--p]);printf ("\n");
}
```

```
/* coda realizzata con un vettore gestito ciclicamente: esempio svolgimento di
una traccia d'esame */

#include<stdio.h>
#define lunghezza 5

void inserisci(int *, int *, int, int *);
void estrai(int *, int *, int *, int *);
void visualizza(int *, int, int );

void main(void)
{
int elem,punt_ins=0,punt_el=0,presenti=0;
int scelta = -1;
int coda[lunghezza];
while(scelta != 0)
{
printf("? 1 per inserire, 2 per estrarre, 3 per visualizzare, 0 per terminare
\n");
scanf("%d", &scelta);
printf("\n");
switch (scelta)
{
case 1:
{
if (presenti==lunghezza)
printf("coda piena");
else
{
printf("elemento da inserire =");
scanf("%d",&elem);
inserisci(coda,&punt_ins, elem, &presenti);
printf("\n");
}
}
break;
case 2 :
{if (presenti==0 )
{
punt_ins=punt_el=0; /*condizione di robustezza dell'algoritmo*/
printf("coda vuota");
}
else
{
```

```
estrai(coda,&punt_el, &elem, &presenti);
printf("%d eliminato", elem);
}
}
break;
case 3: visualizza(coda, punt_el, presenti);
break;
case 0 : printf("fine\n");
}
}
}

void inserisci(int *vettore, int *puntoins, int elemento, int *presenti)
{
vettore[*puntoins]=elemento;
*puntoins=((*puntoins+1)%lunghezza);
*presenti=*presenti+1;
}

void estrai(int *vettore, int *puntoel, int *elemento, int *presenti)
{
*elemento=vettore[*puntoel];
*puntoel=(*puntoel+1)%lunghezza;
*presenti=*presenti-1;
}

void visualizza (int *coda, int p, int presenti)
{
int i;
printf("ci sono %d elementi nella coda \n", presenti);
for(i=0;i<presenti;i++)
printf("%d \n", coda[(p+i)%lunghezza]);
printf ("\n");
}
```

```
/* la lista: esempio di doppio puntatore per passare lista che è un puntatore
per indirizzo */

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

struct el{int info;
          struct el *prox;};

void inserisci(struct el **,int );
int ricerca(struct el *,int, int );

void main(void)
{
int num,elem,i;
struct el *lista;
lista=NULL;
printf("%d bytes per gli interi \n",(sizeof(int)));
printf("%d bytes per il puntatore ad el \n",(sizeof(struct el *)));
printf("%d bytes per el \n",(sizeof(struct el )));
printf("quanti sono gli elementi della lista ? \n");
scanf("%d",&elem);
for(i=0;i<elem;i++)
{
printf("inserisci un numero ");
scanf("%d",&num);
inserisci(&lista,num);
}
do
{
printf("inserisci il numero cercato \n");
scanf("%d",&num);
if((ricerca(lista,num,elem))== 0 )
printf("il numero inserito non si trova nella lista\n");
printf("rieseguire?\n");
}
while(getch()!='n');
}

int ricerca(struct el * lista,int elemento, int posizione)
{
int flag; int cont=0;
struct el *p;
```

```
p=lista;flag=0;
while((p!=NULL&&(flag==0)))
{
cont++;
if(p->info==elemento)
{
flag=1;
printf("%d si trova in posizione %d \n" , elemento, posizione - cont +1);
}
p=p->prox;
}
return flag;
}
void inserisci(struct el ** listaformale,int elemento)
{
struct el *p;
p=(struct el *)malloc(sizeof(struct el));
/* allocazione dinamica della memoria nella zona heap */
p->info=elemento;
p->prox=*listaformale;
*listaformale=p;
}
```

```
/* GESTIONE DI LISTA ORDINATA
   Operazioni di inserimento, eliminazione e visualizzazione
*/

#include<stdio.h>
#include <malloc.h>

struct elemento{
  int inf;
  struct elemento *pun;
};

void gestione_lista(void);
struct elemento *inserimento(struct elemento *);
struct elemento *eliminazione(struct elemento *);
void visualizzazione(struct elemento *);

main()
{
  gestione_lista();
}

void gestione_lista(void)
{
struct elemento *punt_lista = NULL;
int scelta = -1;
char pausa;

while(scelta!=0) {
  printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
  printf("\t\tGESTIONE DI UNA SEQUENZA DI VALORI ORDINATI\n");
  printf("\t\t        MEDIANTE UNA STRUTTURA A LISTA");
  printf("\n\n\n\t\t\t 1. Per inserire un elemento");
  printf("\n\n\t\t\t 2. Per eliminare un elemento");
  printf("\n\n\t\t\t 3. Per visualizzare la lista");
  printf("\n\n\t\t\t 0. Per finire");
  printf("\n\n\n\t\t\t\t Scegliere una opzione : ");
  scanf("\%d", &scelta);
  printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

switch(scelta) {
   case 1:
      punt_lista = inserimento(punt_lista);
      break;
```

```
   case 2:
      punt_lista = eliminazione(punt_lista);
      break;
   case 3:
      visualizzazione(punt_lista);
                printf("\n\nQualsiasi tasto per continuare...");
      scanf("%c%c", &pausa, &pausa);
      break;
  }
}
}

/* Visualizzazione della lista */
void visualizzazione(struct elemento *p)
{
struct elemento *paus = p;

printf("\npunt_lista ");
if (paus == NULL) printf(" --> NULL");
else
  do {
    printf("--> %d ",paus ->inf);
    paus = paus->pun;
     } while (paus!=NULL);
}



/* Inserimento del valore passato dall'utente nella lista
   mantenendo l'ordinamento                                  */
struct elemento *inserimento(struct elemento *p)
{
struct elemento *p0, *p1;
int posizione;

/* Creazione elemento */
p0 = (struct elemento *)malloc(sizeof(struct elemento));

printf("\nInserisci l'informazione (un numero intero): ");
scanf("%d", &p0->inf);

if(p==NULL) {   /* Se la lista è vuota, l'elemento      */
   p=p0;        /* diventa il primo e unico della lista */
   p->pun = NULL;
}
```

```
else {
  if(p->inf > p0->inf) {  /* Se il valore dell'elemento e' */
  p0->pun = p;            /* inferiore al primo, l'elemento */
  p=p0;                   /* diventa il primo della lista */
  }
  else { /* Ricerca della posizione di inserimento */
     p1 = p;
     posizione = 0;
     while(p1->pun!=NULL && posizione!=1) {
       if(p1->pun->inf < p0->inf)
         p1 = p1->pun;      /* Scorre in avanti p1 */
       else
          posizione = 1;    /* Interrompe lo scorrimento */
     }
     p0->pun = p1->pun;     /* Connessione all'elemento successivo */
     p1->pun = p0;          /* Connessione all'elemento precedente */
  }
}
return (p); /* Ritorno del puntatore all'inizio della lista*/
}


/* Eliminazione dell'elemento richiesto dalla lista ordinata */
struct elemento *eliminazione(struct elemento *p)
{
struct elemento *p1 = p, *p2;
struct elemento e;
int posizione = 0;
char pausa;

printf("\nInserisci l'informazione da eliminare: ");
scanf("%d", &e.inf);

if(p1!=NULL) {           /* Se la lista è vuota fine */
  if(p1->inf == e.inf) { /* Se è il primo da eliminare */
    p2 = p1;
    p = p->pun;          /* si modifica il puntatore alla testa */
    free(p2);
    return(p);
  }
  else {                 /* Ricerca dell'elemento da eliminare */
    while(p1->pun!=NULL && posizione!=1) {
      if(p1->pun->inf!=e.inf)
         p1 = p1->pun;  /* Scorre in avanti p1 */
```

```
      else {
        posizione = 1;   /* Interrompe lo scorrimento */
        p2 = p1->pun;
        p1->pun = p1->pun->pun; /* Eliminazione elemento */
        free(p2);               /* Libera la memoria */
      return(p);
      }
    }
  }
}
if(!posizione) {
  printf("\nElemento non incontrato nella lista ");
  scanf("%c%c", &pausa, &pausa);
}
return (p);
}
```

```
/* gestione di una pila con lista lineare */

#include<stdio.h>
#include<malloc.h>

struct elemento
{
int inf;
struct elemento *pun;
};

void gestione_pila(void);
struct elemento *inserimento(struct elemento *, int );
struct elemento *eliminazione(struct elemento *, int *);
int pila_vuota(struct elemento *);
void visualizzazione_pila(struct elemento *);

void main()
{
gestione_pila();
}


void gestione_pila(void)
{
struct elemento *punt_testa=NULL;
int scelta=-1, ele;
char pausa;
while(scelta != 0)
{
printf("scelta = 1 per inserire, 2 per estrarre, 3 per visualizzare, 0 per
terminare \n");
scanf("%d", &scelta);
printf("\n");
switch (scelta)
{
case 1:
     printf("elemento da inserire =");
     scanf("%d",&ele);
     punt_testa=inserimento(punt_testa, ele);
     if(punt_testa==NULL)
     {
     printf("inserimento impossibile");
     printf("memoria disponibile terminata");
```

```
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     }
     break;
case 2 :
     if (pila_vuota(punt_testa))
     {
     printf("eliminazione impossibile");
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     }
     else
     {
     punt_testa=eliminazione(punt_testa,&ele);
     printf("%d eliminato \n", ele);
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     }
     break;
case 3: visualizzazione_pila(punt_testa);
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     break;
}
}
}

void visualizzazione_pila(struct elemento *p)
{
struct elemento *paus=p;
printf("\ntesta della pila");
while(paus!=NULL)
{
printf("\n%d", paus->inf);
paus=paus->pun;
}
}

struct elemento *inserimento(struct elemento *p, int ele)
{
struct elemento *paus;
paus= (struct elemento *)malloc(sizeof(struct elemento));
if (paus==NULL) return (NULL);
paus->pun=p;
```

```
p=paus;
p->inf=ele;
return(p);
}

struct elemento *eliminazione(struct elemento *p, int *ele)
{
struct elemento *paus;
*ele=p->inf;
paus= p;
p=p->pun;
free(paus);
return(p);
}

int pila_vuota(struct elemento *p)
{
if (p==NULL)
return(1);
else
return(0);
}
```

```
/* gestione di una pila con lista lineare e doppio puntatore */

#include<stdio.h>
#include<malloc.h>

struct elemento
{
int inf;
struct elemento *pun;
};

void gestione_pila(void);
void inserimento(struct elemento **, int );
void eliminazione(struct elemento **, int *);
int pila_vuota(struct elemento *);
void visualizzazione_pila(struct elemento *);

void main()
{
gestione_pila();
}

void gestione_pila(void)
{
struct elemento *punt_testa=NULL;
int scelta=-1, ele;
char pausa;
while(scelta != 0)
{
printf("scelta = 1 per inserire, 2 per estrarre, 3 per visualizzare, 0 per
terminare \n");
scanf("%d", &scelta);
printf("\n");
switch (scelta)
{
case 1:
     printf("elemento da inserire =");
     scanf("%d",&ele);
     inserimento(&punt_testa, ele);
     if(punt_testa==NULL)
     {
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     }
```

```
     break;
case 2 :
     if (pila_vuota(punt_testa))
     {
     printf("eliminazione impossibile");
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     }
     else
     {
     eliminazione(&punt_testa,&ele);
     printf("%d eliminato \n", ele);
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     }
     break;
case 3: visualizzazione_pila(punt_testa);
     printf("\n\nqualsiasi tasto per continuare");
     scanf("%c%c", &pausa, &pausa);
     break;
}
}
}


void visualizzazione_pila(struct elemento *p)
{
struct elemento *paus=p;
printf("\ntesta della pila");
while(paus!=NULL)
{
printf("\n%d", paus->inf);
paus=paus->pun;
}
}


void inserimento(struct elemento **p, int ele)
{
struct elemento *paus;
paus= (struct elemento *)malloc(sizeof(struct elemento));

if (paus==NULL)
     printf("eliminazione impossibile");
```

```
paus->pun=*p;
*p=paus;
(*p)->inf=ele;
}



void eliminazione(struct elemento **p, int *ele)
{
struct elemento *paus;
*ele=(*p)->inf;
paus= *p;
*p=(*p)->pun;
free(paus);
}

int pila_vuota(struct elemento *p)
{
if (p==NULL)
return(1);
else
return(0);
}
```

```
/* Trasformazione della rappresentazione di un grafo da una matrice di
adiacenze a una lista di successori */

#include<stdio.h>
#include <malloc.h>

struct nodo {     /* Struttura di un nodo */
   char inf;
   struct successore *pun;
};

struct successore {     /* Elemento della lista di successori */
   int inf;
   struct successore *pun;
};

int a[10][10];        /* Matrice di adiacenze */
struct nodo s[10];    /* Array di nodi */
int n;                /* Numero di nodi */
char esci;

void mat_adiacenze(void);
void vis_mat_adiacenze(void);
void successori(void);
void crea_succ(int, int);
void visita(void);

main()
{
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
mat_adiacenze();       /* Creazione della matrice di adiacenze */
vis_mat_adiacenze();   /* Visualizzazione della matrice */
successori();          /* Creazione delle liste di successori */
visita();              /* Visual. dei successori di ogni nodo */
printf("\n\nQualsiasi tasto per continuare...");
scanf("%c%c", &esci, &esci);
}

/* Crea la matrice di adiacenze */
void mat_adiacenze(void)
{
int i,j;
char invio;
```

```
printf("\nNumero di nodi: ");
scanf("%d", &n);
scanf("%c", &invio);

for(i=0;i<n;i++) {     /* Richiesta etichette dei nodi */
  printf("\nEtichetta del nodo: ");
  scanf("%c", &s[i].inf);
  scanf("%c", &invio);
  s[i].pun = NULL;
  }

for(i=0; i<n; i++)     /* Richiesta archi orientati */
  for(j=0; j<n; j++) {
    printf("\nArco orientato da [%c] a [%c] (0 no, 1 si) ? ",
                                                    s[i].inf, s[j].inf);
    scanf("%d", &a[i][j]);
  }
}

/* Visualizza la matrice di adiacenze */

void vis_mat_adiacenze(void)
{
int i, j;

printf("\nMATRICE DI ADIACENZE\n");
for(i=0; i<n; i++)   /* Visualizza i nodi (colonne) */
  printf("    %c", s[i].inf);

for(i=0; i<n; i++) {
  printf("\n%c   ", s[i].inf); /* Visualizza i nodi (righe) */
for(j=0; j<n; j++)
  printf("%d    ", a[i][j]); /* Visualizza gli archi */
}
}

/* Crea le liste di successori. Per ogni arco rappresentato
   nella matrice di adiacenze chiama crea_succ()            */

void successori(void)
{
int i,j;

for(i=0; i<n; i++)
```

```
  for(j=0; j<n; j++) {
    if(a[i][j]==1)
     crea_succ(i,j);
  }
}

/* Dato un certo arco nella matrice di adiacenze crea
   il rispettivo elemento di lista                    */

   void crea_succ(int i, int j)
   {
   struct successore *p;
   if(s[i].pun==NULL) { /* Non esiste la lista dei successori */
     s[i].pun = (struct successore *)(malloc(sizeof(struct successore)));
     s[i].pun->inf = j;
     s[i].pun->pun = NULL;
   }
   else {               /* Esiste la lista dei successori */
     p = s[i].pun;
     while(p->pun!=NULL)
                                  p = p->pun;
     p->pun = (struct successore *)(malloc(sizeof(struct successore)));
     p = p->pun;
     p->inf = j;
     p->pun = NULL;
   }
   }
/* Per ogni nodo del grafo restituisce i suoi successori.
   Lavora sulle liste di successori                       */

void visita(void)
{
int i;
struct successore*p;
printf("\n");
for(i=0; i<n; i++) {
  printf("\n[%c] ha come successori: ", s[i].inf);
  p = s[i].pun;
  while(p!=NULL) {
    printf(" %c", s[p->inf].inf);
    p = p->pun;
  }
}
}
```

Alcuni Testi di Riferimento adottati per questa dispensa sul linguaggio C:

Bellini – Guidi: Linguaggio C - McGraw-Hill
Schildt: C – The Complete Reference – Second Edition - McGraw-Hill