Prove Intermedie Corso FIODL

Domenico Capano

Indice

Introduzione

In questa dispensa, allegabile al libro di Domenico Capano "*E-Learning: un esperimento via web su corsi di Fondamenti di Informatica. Progettare in modalità elearning con focus sul discente*" e reperibile al link: http://www.comunedasa.it/elearning tratteremo delle Prove Intermedie, assegnate nel corso online FIODL[1], illustrandone tracce e loro soluzioni.

In tale libro avevamo diviso i due corsi di Fondamenti di Informatica (F1 ed F2), trattati nel web (al link: http://digilander.libero.it/capano/index.html), in Moduli.

Ogni Modulo dei due corsi aveva una Prova finale, di valutazione ed autovalutazione degli studenti e generatrice di crediti formativi di corso.

La Prova finale o Prova Intermedia, come sapranno i lettori di quel testo, è stata definita come un *SottoModulo[2] erogato alla fine dei SottoModuli compresi nel Modulo che la contiene*.

Le Prove Intermedie hanno almeno tre principali motivi d'essere presenti, in un corso a distanza:

a) Verifica, da parte dei tutors, mentors e docenti, dell'apprendimento acquisito dai learners studiando il modulo in cui essa è contenuta.

b) Diminuzione della sensazione d'isolamento, insita in un corso a distanza, dei learners, (in quanto il cercare di affrontare positivamente le prove intermedie, spinge il discente a relazionarsi maggiormente con gli altri protagonisti dell'aula virtuale, colleghi di corso, docenti e tutors;

c) Strumento per attribuire i crediti formativi, di corso, intermedi.

Le Prove intermedie di corso rappresentano l'interazione studente-(tutor e/o docente), *comunicazione uno a uno*.

Una Prova Intermedia in questo contesto è un esercizio in linguaggio C, precisamente *traccia dell'esercizio e sua possibile soluzione da noi proposta*, che rappresentano un SottoModulo particolare costituito da un unico learning object[3].

Abbiamo proposto sette Prove Intermedie per gli studenti di Fondamenti di Informatica 1 e sette Prove Intermedie per gli studenti di Fondamenti di Informatica 2 che riportiamo di seguito nella loro interezza.

---

[1] **F**ondamenti di **I**nformatica in **O**pen and **D**istance **L**earning

[2] Un SottoModulo (che può essere o una lezione del corso o un test di autovalutazione od una Prova Intermedia) è un insieme di Learning Objects

[3] "atomo di conoscenza" autoconsistente

1 Prove Intermedie FIODL Fondamenti 1

1.1 Traccia Prova Intermedia 1 di F1

Scrivete un programma in linguaggio macchina simbolico che calcola e stampi la seguente espressione: N1+(N2-1), supponendo di memorizzare i numeri N1 ed N2 nelle celle di memoria di indirizzo 150 e 151 rispettivamente; usate le istruzioni della seguente tabella (LMS). facoltativamente: eseguite la traduzione del programma in linguaggio macchina.*/.

Tabella Linguaggio Macchina Simbolico

Tabella del nostro linguaggio macchina simbolico LMS					
Sintassi			Semantica		
codice operativo		operando	Operazionale	Discorsiva	Simbolo
1	0001	IND	(ACC) ←(ACC)+(IND)	Somma	SUM
2	0010	IND	(ACC) ←(ACC)-(IND)	Sottrazione	SOT
3	0011	IND	(ACC) ←(IND)	Loading	LOAD
4	0100	IND	(IND) ←(ACC)	Memorizzazione	STORE
5	0101	IND	(IND) ←(UL)	Lettura	READ
6	0110	IND	(US) ←(IND)	Scrittura	PRINT
7	0111	IND	(PC) ←IND	Salto incondizionato	JUMP
8	1000	IND	SE (ACC)>=0 (PC) ←IND	Salto condizionato	JUMPC
9	1001		STOP	Arresto del programma	STOP

1.1.1 Soluzione Prova Intermedia 1 di F1

/*Scrivete un programma in linguaggio macchina simbolico che calcola e stampi la seguente espressione: N1+(N2-1), supponendo di memorizzare i numeri N1 ed N2 nelle celle di memoria di indirizzo 150 e 151 rispettivamente; usate le istruzioni della tabella LMS;

facoltativamente: eseguite la traduzione del programma in linguaggio macchina.*/

ANALISI: dobbiamo leggere i numeri N1 , N2 ed 1 dall'unità di input, poi calcolare l'espressione N1 + (N2 - 1) e poi stampare il risultato.

DATI: N1, N2, 1
Risultato=(N1 + (N2-1))

SINTESI ALGORITMO CON RAFFINAMENTI:

1 leggi da input N1 e memorizzalo nella cella di indirizzo 150
2 leggi da input N2 e memorizzalo nella cella di indirizzo 151
3 carica N2 in ACC
4 leggi da input 1 e memorizzalo nella cella di indirizzo 151
5 esegui la differenza tra (ACC) ed (151) e metti il risultato in ACC
6 esegui la somma tra (ACC) e N1 e metti il risultato in ACC
7 memorizza (ACC) nella cella di indirizzo 152
8 stampa il contenuto della cella di indirizzo 152
9 stop

PROGRAMMAZIONE IN LINGUAGGIO MACCHINA SIMBOLICO LMS:

1 (150) ← (UL)
2 (151) ← (UL)
3 (ACC) ← (151)
4 (151) ← (UL)
5 (ACC) ← (ACC) - (151)
6 (ACC) ← (ACC) + (150)
7 (152) ← (ACC)
8 (US) ← (152)
9 STOP

Supponendo che il programma abbia inizio all'indirizzo 200 la sua traduzione in decimale e':

INDIRIZZI | ISTRUZIONI/DATI
... | ...
150 | N1
151 | N2
152 | RISULTATO
... | .....
200 | 5 | 150
201 | 5 | 151
202 | 3 | 151
203 | 5 | 151
204 | 2 | 151
205 | 1 | 150
206 | 4 | 152
207 | 6 | 152
208 | 9 |

SCRITTURA FINALE IN BINARIO:

INDIRIZZI | ISTR/DATI
... | ...
10010110 | N1
10010111 | N2
10011000 | RIS.
... | ...
11001000 | 0101 | 10010110
11001001 | 0101 | 10010111
11001010 | 0011 | 10010111
11001011 | 0101 | 10010111
11001100 | 0010 | 10010111
11001101 | 0001 | 10010110
11001110 | 0100 | 10011000
11001111 | 0110 | 10011000
11010000 | 1001 |

1.2 Traccia Prova Intermedia 2 di F1

Scrivete un programma in linguaggio C, che letto da input un intero n>0, calcoli il massimo ed il minimo, fra gli n numeri reali anche essi letti da input e li visualizzi a video.

1.2.1 Soluzione Prova Intermedia 2 di F1

```
#include <stdio.h>
/*programma che calcola il massimo ed il minimo di n numeri reali letti
da input con n anche esso letto da input*/
int main()
{
float max,min, num;
int i, n;
char car;
printf("Inserisci il totale dei numeri di cui si vuol calcolare max e min? ");
scanf("%d", &n);
printf("Inserisci il 1) numero: ");
scanf("%f", &max);
min=max;
 for (i=1; i<n; i++)
 {
  printf("Inserisci il %d) numero: ",i+1);
  scanf("%f", &num);
  if (num<min)
   min=num;
  else
 if (num>max)
  max=num;
 }
printf("Il max dei %d interi digitati e' %.3f, mentre il minimo e' %.3f.\n",
n, max, min);
scanf("\n%c",&car);
return(1);
}
```

1.3 Traccia della prova 3 di FI-ODL

Progettate un programma in linguaggio C, che calcoli il numero delle combinazioni semplici Cnk, di n oggetti di classe k, con n e k interi positivi

letti da input, con il vincolo n>=k.
Le Cnk sono date dalla formula:
Cnk=[n*(n-1)*(n-2)*...*(n-k+1)]/[k*(k-1)*(k-2)*...*1]

1.3.1 Soluzione Prova Intermedia 3 per gli studenti di fondamenti 1

/*Progettate un programma in linguaggio C che calcoli il numero delle

combinazioni semplici Cnk di n oggetti di classe k con n e k interi positivi

letti da input, con il vincolo che n>=k. Le Cnk sono date dalla formula:
Cnk=[n*(n-1)*(n-2)*...*(n-k+1)]/[k*(k-1)*(k-2)*...*1]*/

```
#include<stdio.h>
int main()
{
int n,k,i,c,cont,num_cs=1;
int den_cs=1;
char car;
printf("\nInserite il numero di oggetti n\n");
scanf("%d",&n);
printf("\nInserite il numero di classi k\n");
scanf("%d",&k);
cont=n;
 if(n>=k)
 {
   while(cont>=(n-k+1))
   {
   num_cs=cont*num_cs;
   cont--;
   }
  for(i=0;i<k;i++)
  den_cs=den_cs*(i+1);
 c=num_cs/den_cs;
```

```
 printf("\nIl numero delle combinazioni semplici e' = %d\n",c);

 }
else
 printf("\nInserite un valore di n maggiore od uguale a k\n");
scanf("\n%c",&car);
return 0;
}
```

1.4 Traccia prova 4 di FI-ODL

Progettate un programma in linguaggio C, che esegua la fusione tra due vettori di interi A e B ordinati in senso crescente, producendo un terzo vettore C ordinato.

Il vettore risultante dalla fusione deve contenere tutti gli elementi di A e di B anche se tali elementi si ripetono.

1.4.1 Soluzione prova intermedia 4 facoltativa per gli studenti di fondamenti 1

```
/*Progettate un programma in linguaggio C, che esegua la fusione tra due vettori
di interi A e B ordinati in senso crescente, producendo un terzo vettore C ordinato.
Il vettore risultante dalla fusione deve contenere tutti gli elementi di A e di B anche se tali elementi si ripetono*/

#include <stdio.h>
#define DIM_A 100
#define DIM_B 100
#define DIM_C 200

/*prototipi delle funzioni usate*/
int leggi_A(int A[]);
int leggi_B(int B[]);
void fusione_vettori(int numa,int numb,int A[],int B[]);

int main()
{
```

```
 int A[DIM_A];
 int B[DIM_B];
 char car;
 int na,nb;
 na=leggi_A(A);
 nb=leggi_B(B);
 fusione_vettori(na,nb,A,B);
 printf("\nInserite un carattere\n");
 scanf("\n%c",&car);
 return(0);
}

int leggi_A(int A[])
{
int i, num_a;
printf("\nInserite il numero di elementi del vettore A ");
scanf("%d",&num_a);
for(i=0;i<num_a;i++)
 {
 printf("inserite il valore %d del vettore A ",i+1);
 scanf("%d",&A[i]);
 }
for(i=0;i<num_a;i++)
 printf("%d\n",A[i]);

return(num_a);
}

int leggi_B(int B[])
{
int i, num_b;
printf("\nInserite il numero di elementi del vettore B ");
scanf("%d",&num_b);
 for(i=0;i<num_b;i++)
 {
 printf("inserite il valore %d del vettore B ",i+1);
 scanf("%d",&B[i]);
 }
 for(i=0;i<num_b;i++)
 printf("%d\n",B[i]);

return(num_b);
```

```
}

void fusione_vettori(int numa,int numb,int A[DIM_A],int B[DIM_B])
{
int C[DIM_C];
int i=0,j=0,num_c=0;
while((i<numa)&&(j<numb))
{
 if(A[i]<=B[j])
 {
 C[num_c]=A[i];
 i++;
 num_c++;
 }
 else
  {
  C[num_c]=B[j];
  j++;
  num_c++;
  }
}
while(i<numa)
{
 C[num_c]=A[i];
 i++;
 num_c++;
}
while(j<numb)
{
 C[num_c]=B[j];
 j++;
 num_c++;
}
printf("\necco il vettore fusione ordinato C se gli elementi di A e B ");
printf("sono stati inseriti ordinati\n");
for(i=0;i<num_c;i++)
 printf("%d\n",C[i]);
}
```

1.5 Traccia della Prova 5 di FI-ODL

Progettate un programma in linguaggio C, che legga un array di n interi da input; verifichi l'eventuale presenza, di un intero digitato da input, all'interno dell'array e stampi l'esito di tale ricerca: (esempio: elemento non trovato, elemento cercato e' in posizione x all'interno dell'array); l'elemento da cercare deve essere passato come parametro alla funzione che si utilizza, nel programma.

1.5.1 Soluzione prova 5 per fondamenti 1

/*Progettate un programma in linguaggio C, che legga un array di n interi da input; verifichi l'eventuale presenza di un elemento intero all'interno dell'array e stampi l'esito di tale ricerca: (esempio: elemento non trovato, elemento cercato e' in posizione x all'interno dell'array); l'elemento da cercare deve essere passato come parametro alla funzione che si utilizza, nel programma.*/

```
#include <stdio.h>

#define N 10
typedef int Tipo_array[N];
Tipo_array nome_array;

/*prototipi delle funzioni usate*/
void leggi_array();
void trova_elemento(int elem );


int main()
{
char car;
int elemento;
leggi_array();
printf("\ninserite l'elemento intero da cercare\n");
scanf("\n%d",&elemento);
trova_elemento(elemento);
scanf("\n%d",&car);
return(0);
}
```

```
void leggi_array()
{
int i;
printf("\ninserite i %d elementi interi dell'array\n",N);
 for(i=0;i<N;i++)
  {printf("%d ) ",i+1);
   scanf("%d",&nome_array[i]);
  }
}

void trova_elemento(int elem)
{
int k,trovato=0;
for(k=0;k<N;k++)
 if(nome_array[k]==elem)
 {
  trovato++;
  printf("\nl'elemento cercato %d si trova in posizione %d\n",elem,k+1);
 }

if(trovato==0)
  printf("\nl'elemento %d non esiste nell'array\n",elem);

}
```

1.6 Traccia della prova 6 di FI-ODL

Progettate un programma in linguaggio C, il quale legge un vettore che rappresenta le eta' di un gruppo di persone, e stampa tutte le eta' comprese tra la minima e la massima non presenti nel vettore. Se ad esempio il vettore fosse composto da 5,8,2 si dovrebbe stampare 3,4,6,7 .

1.6.1 Soluzione prova 6 per fondamenti 1

```
/*Progettate un programma in linguaggio C, il quale legge un vettore che
rappresenta le eta' di un gruppo di persone, e stampa tutte le eta' comprese tra
la minima e la massima non presenti nel vettore. Se ad esempio il vettore
fosse composto da 5,8,2 si dovrebbe stampare 3,4,6,7 .*/
#include<stdio.h>
```

```
#define NUM_ETA 10
#define ETA_MAX 121
int main(void) {
int cont;
int minimo; /* minima eta' nel vettore */
int massimo; /* massima eta' nel vettore */
int i; /* indice di ciclo */
int eta[NUM_ETA]; /* vettore delle eta' */
int presenza_eta[ETA_MAX]; /* vettore in cui l'eta' e' un indice: il
generico elemento e' uno se la corrispondente eta' e' nel vettore */
char car;

/*lettura vettore*/
printf("inserite le %d eta' delle persone\n",NUM_ETA);

for(i=0; i<NUM_ETA; i++)
 scanf("%d", &eta[i]);


/* trova il massimo e il minimo nel vettore */
minimo=eta[0];
massimo=eta[0];

 for(i=1; i<NUM_ETA; i++) {
  if( eta[i]>massimo )
  massimo=eta[i];
   if( eta[i]<minimo )
    minimo=eta[i];
 }


/* inizializza il vettore delle presenze */
for(i=0; i<ETA_MAX; i++)
presenza_eta[i]=0;

/* marca ogni eta' nel vettore come presente */
for(i=0; i<NUM_ETA; i++)
 presenza_eta[eta[i]]=1;

/* stampa gli indici che sono compresi fra massimo e minimo
e il cui elemento del vettore presenza vale 0 */
cont=0;
```

```
for(i=minimo+1; i<massimo-1; i++)
{ cont++;
 if( presenza_eta[i]==0 )
  printf("%d eta' non presente\n", i);
  if((cont%10)==0) /* visualizzo 10 elementi alla volta*/
   {
   printf("digitare un carattere \n");
   scanf("%c",&car);
   }
 }
return 0;
}
```

1.7 Traccia della prova 7 di FI-ODL

Progettate un programma in linguaggio C, il quale concateni due stringhe lette da input, di al piu' 35 caratteri ciascuna; e stampi le tre stringhe: prima stringa, seconda stringa, stringa concatenata.

1.7.1 Soluzione Prova 7 per fondamenti 1

```
/*Progettate un programma in linguaggio C, il quale concateni due stringhe
lette da input, di al piu' 35 caratteri ciascuna e stampi le tre stringhe:
prima stringa, seconda stringa e terza stringa.*/

#include<stdio.h>
#define L_MAX 35

int main()
{
char car;
char prima_stringa[L_MAX+1], seconda_stringa[L_MAX+1];
char terza_stringa[2*L_MAX+1];
int i=0,j=0;
printf("\nScrivi la prima stringa ");
 gets(prima_stringa);
printf("\nScrivi la seconda stringa ");
 gets(seconda_stringa);
printf("\nho letto: %s e %s",prima_stringa,seconda_stringa);
printf("\nla stringa concatenata e'\n");
```

```
      for(;prima_stringa[i];i++)
       terza_stringa[i]=prima_stringa[i];
      for(;seconda_stringa[j];j++)
        terza_stringa[i+j]=seconda_stringa[j];/*copia seconda in terza a partire
da dove ero arrivato con l'indice i*/
    terza_stringa[i+j]='\0'; /*chiusura esplicita della terza stringa*/
    printf("\n%s\n",terza_stringa);
    printf("\ninserite un carattere per terminare\n");
     scanf("%c",&car);
    return(0);
    }
```

2 Prove Intermedie Per Fondamenti 2

2.1 Traccia Prova 1 di FI-ODL

Progettate un programma in C, che calcoli i numeri di Fibonacci, di n numeri interi, con n letto da input e li visualizzi a video; usate una funzione per calcolare tali numeri , sia nella versione iterativa che ricorsiva; indicate il numero di attivazioni necessarie nella funzione ricorsiva per calcolare un numero di Fibonacci.

suggerimento:

i numeri della serie di Fibonacci si ottengono cosi: Fib(x)= 1 se x<=1 e Fib(x)= Fib(x-1)+Fib(x-2) se x>1; dove x e' il generico intero >=0.

2.1.1 Soluzione prova 1 per studenti di fondamenti 2

2.1.2 Versione iterativa

```
#include<stdio.h>

/*prototipo della funzione fib*/
long int fib(int);
int main()
{
char car;
int k;
long int risultato;
```

```
int numero,num_elementi;
printf("\nInserite quanti sono gli elementi della serie di fibonacci\n");
scanf("\n%d",&num_elementi);
for(k=0;k<num_elementi;k++)
 {
 printf("\nInserite un numero intero:\n");
 scanf("%d",&numero);
 risultato=fib(numero);
 printf("Fibonacci(%d) = %ld\n",numero,risultato);
 }
printf("\ndigitate un carattere\n");
scanf("\n%c",&car);
return (0);
}

/*definizione della funzione iterativa di fibonacci*/
long int fib(int x)
{
long Fib1=1;
long Fib2;
long Fib=1;
int i;
for(i=0;i<x-1;i++)
 {
 Fib2=Fib1;
 Fib1=Fib;
 Fib=Fib1+Fib2;
 }
return(Fib);
}
```

2.1.3 Versione ricorsiva

```
#include<stdio.h>

/*prototipo della funzione fib*/
long int fib(int);
/*variabile globale*/

long int numero_attivazioni=1; /*inizializzazione*/
```

```
int main()
{
char car;
int k;
long int risultato;
int numero, num_elementi;
printf("\nInserite quanti sono gli elementi della serie di fibonacci\n");
scanf("\n%d",&num_elementi);
for(k=0;k<num_elementi;k++)
 {
 printf("\nInserite un numero intero:\n");
 scanf("%d",&numero);
 risultato=fib(numero);
 printf("Fibonacci(%d) = %ld\n",numero,risultato);
 printf("\nIl numero di attivazioni e': %ld uguale a      Fibonacci(%d)
",numero_attivazioni,numero);
 numero_attivazioni=1; /*reinizializzazione*/
 }
scanf("\n%c",&car);
return (0);
}

/*definizione della funzione ricorsiva di fibonacci*/
long int fib(int x)
{
 if((x==0)||(x==1))
  return(1);
  else
   {
   numero_attivazioni++;
   return(fib(x-1)+fib(x-2));
   }
}
```

2.2 Traccia prova 2 di FI-ODL

Esercitazione facoltativa

Scrivete un programma in linguaggio macchina simbolico che calcola la seguente espressione: N1+(N2-1), supponendo di memorizzare i numeri N1 ed N2 nelle celle di memoria di indirizzo 150 e 151 rispettivamente; usate le istruzioni della LMS ; eseguite la traduzione del programma in linguaggio macchina.

2.2.1 Soluzione Prova 2 di FI-ODL

Vedere la soluzione Prova 1 per Fondamenti di Informatica 1

2.3 Traccia prova 3 di FI-ODL

Progettate un programma in linguaggio C, che esegua la fusione tra due vettori di interi A e B ordinati in senso crescente, producendo un terzo vettore C ordinato.

Il vettore risultante dalla fusione deve contenere tutti gli elementi di A e di B anche se tali elementi si ripetono.

2.3.1 Soluzione prova intermedia 3 per gli studenti di fondamenti 2

```
/*Progettate un programma in linguaggio C, che esegua la fusione tra due vettori
di interi A e B ordinati in senso crescente, producendo un terzo vettore C ordinato.
Il vettore risultante dalla fusione deve contenere tutti gli elementi di A e di B anche se tali elementi si ripetono*/

#include <stdio.h>
#define DIM_A 100
#define DIM_B 100
#define DIM_C 200

/*prototipi delle funzioni usate*/
int leggi_A(int A[]);
int leggi_B(int B[]);
```

```
void fusione_vettori(int numa,int numb,int A[],int B[]);

int main()
{
 int A[DIM_A];
 int B[DIM_B];
 char car;
 int na,nb;
 na=leggi_A(A);
 nb=leggi_B(B);
 fusione_vettori(na,nb,A,B);
 printf("\nInserite un carattere\n");
 scanf("\n%c",&car);
 return(0);
}

int leggi_A(int A[])
{
int i, num_a;
printf("\nInserite il numero di elementi del vettore A ");
scanf("%d",&num_a);
for(i=0;i<num_a;i++)
 {
 printf("inserite il valore %d del vettore A ",i+1);
 scanf("%d",&A[i]);
 }
for(i=0;i<num_a;i++)
 printf("%d\n",A[i]);

return(num_a);
}

int leggi_B(int B[])
{
int i, num_b;
printf("\nInserite il numero di elementi del vettore B ");
scanf("%d",&num_b);
 for(i=0;i<num_b;i++)
 {
 printf("inserite il valore %d del vettore B ",i+1);
 scanf("%d",&B[i]);
 }
```

```
 for(i=0;i<num_b;i++)
 printf("%d\n",B[i]);

return(num_b);
}

void fusione_vettori(int numa,int numb,int A[DIM_A],int B[DIM_B])
{
int C[DIM_C];
int i=0,j=0,num_c=0;
while((i<numa)&&(j<numb))
{
 if(A[i]<=B[j])
 {
 C[num_c]=A[i];
 i++;
 num_c++;
 }
 else
  {
  C[num_c]=B[j];
  j++;
  num_c++;
  }
}
while(i<numa)
{
 C[num_c]=A[i];
 i++;
 num_c++;
}
while(j<numb)
{
 C[num_c]=B[j];
 j++;
 num_c++;
}
printf("\necco il vettore fusione ordinato C se gli elementi di A e B ");
printf("sono stati inseriti ordinati\n");
for(i=0;i<num_c;i++)
 printf("%d\n",C[i]);
}
```

2.4 Traccia della prova 4 di FI-ODL

Progettate un programma in linguaggio C, che calcoli il numero delle combinazioni semplici Cnk, di n oggetti di classe k, con n e k interi positivi o nulli, letti da input, con il vincolo n>=k.

Le Cnk sono date dalla formula:

Cnk=n!/[k! *(n-k)!]

2.4.1 Soluzione prova facoltativa 4 per fondamenti 2

```
/* Progettate un programma in linguaggio C, che calcoli il numero delle
combinazioni semplici Cnk, di n oggetti di classe k, con n e k interi
positivi o nulli, letti da input, con il vincolo n>=k.
Le Cnk sono date dalla formula:
Cnk=n!/[k! *(n-k)!]
*/

#include <stdio.h>
long int fatt(int n); /* prototipo della funzione fattoriale*/
int main()
{
char car;
int n ,k;
long int comb;
printf("\nInserite il numero di oggetti ");
scanf("%d",&n);
printf("\nInserite il numero di classi ");
scanf("%d",&k);
if(n>=k)

{
comb= fatt(n)/(fatt(k)*fatt(n-k));
printf("\nLe combinazioni di %d oggetti presi a %d a %d sono %ld: \n",
n,k,k,comb);
}
else
printf("\nCalcolo non effettuabile n deve essere >= a k \n");
printf("\ninserire un carattere per terminare\n");
scanf("\n%c",&car);
```

```
return(0);
}


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

2.5 Traccia della prova 5 di FI-ODL

Progettate un programma in linguaggio C, che legga un intero n, dopo legga n reali, memorizzi gli n reali in un array dinamico e li stampi in ordine inverso.

2.5.1 Soluzione prova 5 per fondamenti 2

```
/*Progettate un programma in linguaggio C, che legga un intero n, dopo
legga n reali, memorizzi gli n reali in un array dinamico e li stampi in ordine
inverso.*/
#include<stdio.h>
#include<stdlib.h>
/*programma che accetta n reali,con n e ed i reali letti da input,
li registra su un array dinamico e li stampa in ordine inverso*/
int leggi_n();
float *leggi_array(int n);
void stampa_array(float *, int n);

int main()
{
float *punt_a;
int num_el;
num_el=leggi_n();
punt_a=leggi_array(num_el);
```

```
    stampa_array(punt_a,num_el);
    free(punt_a);return 0;
    }


    int leggi_n()
    {
    int n;
    printf("\ninserite il numero di elementi dell'array\n");
    scanf("%d",&n);
    return n;
    }

    float *leggi_array(int n)
    {
    int i=0;
    float *pArray,*aux;
    float elemento;
    printf("\ninserite gli elementi reali dell'array\n");
    pArray=malloc(n*sizeof(float));
    while(i<n)
    {
    scanf("%f",&elemento);
    *(pArray+i)=elemento;
    i++;
    }
    return pArray;
    }

    void stampa_array(float *punt, int n)
    {
    int i=0;
    printf("\nEcco gli/i %d elementi reali che abbiamo inserito nell'array
dinamico\n",n);
    for(i=n-1;i>=0;i--)
    printf("%d) %.2f ", i+1, *(punt+i));
    }
```

2.6 Traccia della prova 6 di FI-ODL

Avendo le seguenti definizioni: typedef int TipoElemLista;
struct StructLista { TipoElemLista info; struct StructLista *next; };
typedef struct StructLista TipoNodoLista;
typedef TipoNodoLista *TipoLista;
progettate due funzioni di tipo void a) che restituisce una copia della lista

puntata da lis "di tipo TipoLista" b) che inverta la lista puntata da lis;

fornite le due funzioni nella versione iterativa.

suggerimento:
in a) conviene utilizzare un nodo generatore;
in b) utilizzare due puntatori, succ e prec, successivo e precedente a quello corrente.

2.6.1 Soluzione prova intermedia 6 per fondamenti 2

```
/*Avendo le seguenti definizioni: typedef int TipoElemLista;
 struct StructLista { TipoElemLista info;
                 struct StructLista *next; };
typedef struct StructLista TipoNodoLista;
typedef TipoNodoLista *TipoLista;
progettate due funzioni di tipo void a) che restituisce una copia della lista puntata da lis "di tipo TipoLista" b) che inverta la lista puntata da lis; fornite le due funzioni nella versione iterativa.
suggerimento: in a) conviene utilizzare un nodo generatore; in b) utilizzare due puntatori, succ e prec, successivo e precedente a quello corrente.*/

void CopiaLista(TipoLista lis, TipoLista *copia)
/* Restituisce una copia della lista lis.
Versione iterativa. Utilizza nodo generatore. */
{
TipoLista prec; /* puntatore all'elemento precedente */
```

```
    prec = malloc(sizeof(TipoNodoLista)); /* creazione del nodo generatore
*/
    /* scansione e copia della lista */
    *copia = prec;
    while (lis != NULL) { /* copia dell'elemento corrente di lis */
    prec->next = malloc(sizeof(TipoNodoLista));
    prec = prec->next;
    prec->info = lis->info;
    lis = lis->next;
    }
    prec->next = NULL; /* chiusura della lista */
    /* eliminazione del nodo generatore */
    prec = *copia;
    *copia = (*copia)->next;
    free(prec);
    } /* CopiaLista */


    void InvertiLista(TipoLista *lis)
    /* Inverte la lista lis. Versione iterativa. */
    {
    TipoLista prec = NULL; /* puntatore all'elemento che precede quello
corrente */
    TipoLista suc; /* puntatore all'elemento successivo a quello corrente */

    while (*lis != NULL) {
    suc = *lis;
    *lis = (*lis)->next;
    suc->next = prec;
    prec = suc;
    }
    *lis = prec;
    } /* InvertiLista */
```

2.7 Traccia della prova 7 di FI-ODL

1) Scrivete un programma in linguaggio C, il quale concateni due stringhe di al piu' 35 caratteri ciascuna; le funzioni utilizzate devono ricevere, come parametri, gli array considerandoli puntatori; la scansione degli array deve avvenire mediante espressioni puntatore.

2) Facoltativamente:

Scrivete un programma in linguaggio C che lette da input due matrici di interi A e B calcoli, attraverso una funzione, il prodotto tra le due matrici A (n,p) e B(p,m), le quali non abbiano una dimensione prefissata, ottenendo una matrice C(n,m) che deve essere stampata a video.

2.7.1 Soluzione prova intermedia 7 per fondamenti 2

```
/* 1) Scrivete un programma in linguaggio C, il quale concateni due stringhe
   lette da input, di al piu' 35 caratteri ciascuna;le funzioni utilizzate
   devono ricevere, come parametri, gli array considerandoli puntatori;
   la scansione degli array deve avvenire mediante espressioni puntatore.*/

   #include<stdio.h>
   #define L_MAX 35
   void leggi_stringa(char *);
   void concatena(char *,char *, char *);

   int main()
   {
   char car;
   char prima_stringa[L_MAX+1], seconda_stringa[L_MAX+1];
   char terza_stringa[2*L_MAX+1];
   int i=0,j=0;
   leggi_stringa(prima_stringa);
   leggi_stringa(seconda_stringa);
   printf("\nho letto: %s e %s",prima_stringa,seconda_stringa);
   concatena(prima_stringa,seconda_stringa,terza_stringa);
   printf("\nla stringa concatenata e'\n");
   printf("\n%s\n",terza_stringa);
```

```
printf("\ninserite un carattere per terminare\n");
scanf("%c",&car);
return 0;
}

void leggi_stringa(char *string)
{
printf("\nScrivi una stringa stringa ");
gets(string);

}

void concatena(char *pr,char *sec,char *te)
{
int i,j;
for(i=0;*(pr+i);i++) /*copia prima in terza*/
  *(te+i)=*(pr+i);
for(j=0;*(sec+j);j++) /*copia seconda in terza*/
  *(te+i+j)=*(sec+j);
*(te+i+j)='\0'; /*chiusura esplicita di terza*/

}
```

2.7.2 *Esercizio prova 7 Facoltativo*

/* 2) Scrivete un programma in linguaggio C, che lette da input due matrici di interi A e B calcoli, attraverso una funzione, il prodotto tra le due matrici A(n,p)
e B(p,m), le quali non abbiano una dimensione prefissata, ottenendo una matrice C(n,m) che deve essere stampata a video*/

```
#include<stdio.h>
#include<stdlib.h>
int *leggi_matrice_a(int n_rig_a,int p_col_a);
int *leggi_matrice_b(int p_rig_b, int m_col_b);
void calcola_prodotto(int *A_mat,int *B_mat, int n, int p, int m);
void stampa_matrice(int *A_B_C, int n, int m,char *string,char tipo);

int main()
{
```

```
    int n,p,m;
    int *Am,*Bm,*C;
    char car;
    printf("\nProgramma per il calcolo del prodotto fra due matrici A e B di
interi\n");

    printf("\ninserire il numero di righe di A \n");
    scanf("%d",&n);
    printf("\ninserire il numero di colonne di A \n");
    scanf("%d",&p);
    printf("\ninserire il numero di colonne di B \n");
    scanf("%d",&m);

    Am=leggi_matrice_a(n,p);
    Bm=leggi_matrice_b(p,m);
    stampa_matrice(Am,n,p,"RISTAMPATA",'A');
    calcola_prodotto(Am,Bm,n,p,m);
    free(Am);free(Bm);
    printf("\ndigita un carattere per uscire\n");
    scanf("\n%c",&car);

    return(0);
    }

    int *leggi_matrice_a(int n_rig_a,int p_col_a)
    {
    int i,j;
    int *A;
    A=malloc(n_rig_a*p_col_a*sizeof(int));
    printf("\ninserite %d numeri interi della matrice A letta per
righe\n",n_rig_a*p_col_a);
     for(i=0;i<n_rig_a;++i)
      for(j=i*p_col_a;j<p_col_a*i+p_col_a;j++)
       scanf("%d",(A+j));
    stampa_matrice(A,n_rig_a,p_col_a,"",'A');
    return(A);
    }

    int *leggi_matrice_b(int p_rig_b, int m_col_b)
    {
    int i,j;
    int *B;
```

```
    B=malloc(p_rig_b*m_col_b*sizeof(int));
    printf("\ninserite %d numeri interi della matrice B letta per
righe\n",p_rig_b*m_col_b);
     for(i=0;i<p_rig_b;i++)
      for(j=i*m_col_b;j<(i+1)*m_col_b;j++)
       scanf("%d",(B+j));

    stampa_matrice(B,p_rig_b,m_col_b,"",'B');
    return(B);
    }

    void calcola_prodotto(int *A,int *B, int n, int p, int m)
    {
    int i,j,s,k;
    int *C;
    C=malloc(n*m*sizeof(int));
     for(i=0;i<n;i++)
     {
      for(j=i*m;j<i*m+m;j++)
      {
         *(C+j)=0;  /*inizializzazione del j-esimo elemento della matrice
prodotto*/
       for(s=0;s<p;s++)
        *(C+j)=*(C+j) + *(A+i*p+s)* *(B+s*m+j-m*i);
      }
     }
    stampa_matrice(C,n,m,"PRODOTTO ",'C');
    }

    void stampa_matrice(int *A_B_C,int n, int m,char *string,char tipo )
    {
    int i,j;
    printf("\nECCO LA MATRICE %s %c \n",string,tipo);
     for(i=0;i<n;i++)
     {
      for(j=i*m;j<i*m+m;j++)
       printf("%5d ",*(A_B_C+j));
      printf("\n");/*va alla prossima riga della matrice */
     }
    }
```

2.7.3 Altra soluzione esercizio 7 facoltativo

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

void prodotto(int[],int[],int[],int,int,int);
void stampa(int[],int,int);

int main(){

int i,m,n,p;
int *tab_a,*tab_b,*tab_c;

printf("introdurre dimensioni tabella a:\nnumero righe: ");
scanf("%d",&n);
printf("numero colonne: ");
scanf("%d",&p);
printf("introdurre dimensioni tabella b:\nnumero colonne: ");
scanf("%d",&m);

tab_a=malloc(n*p*sizeof(int));
tab_b=malloc(p*m*sizeof(int));
tab_c=malloc(n*m*sizeof(int));

printf("\nInserire dati matrice a:\n");
for(i=0;i<=n*p-1;i++)
scanf("%d",(tab_a+i));

printf("\nInserire dati matrice b:\n");
for(i=0;i<=p*m-1;i++)
scanf("%d",(tab_b+i));

prodotto(tab_a,tab_b,tab_c,n,m,p);

printf("\nMatrice A\n");
stampa(tab_a,n,p);
printf("\n\nMatrice B\n");
stampa(tab_b,p,m);
printf("\n\nMatrice C\n");
stampa(tab_c,n,m);

return 0;
```

```
}


void prodotto(int a[],int b[],int c[],int n,int m,int p){
int i,j,k;
printf("%d",p);
for(j=0;j<=n-1;j++)
for(i=0;i<=m-1;i++){

c[j*m+i]=0;
for(k=0;k<=p-1;k++)
c[j*m+i]+=a[j*p+k]*b[k*m+i];
}
}


void stampa(int tab[],int riga,int colonna){
int i,j;

for(j=0;j<=riga-1;j++)
for(i=0;i<=colonna-1;i++){
printf("%4d",tab[j*colonna+i]);
if(i==colonna-1)
printf("\n");
}

}
```