

## ESERCIZIO

Si scriva un programma C che:

- a) legga da terminale una sequenza di interi terminati dal valore 0 (uno su ogni linea) e li inserisca in un vettore DIGIT di 20 componenti (si suppone che la sequenza sia di lunghezza minore o uguale a 20);
- b) Stampi la media degli interi positivi pari.

## Soluzione:

```
#include <stdio.h>

main()
{
    int DIGIT[20];
    int i,j,n_pos_pari;
    float media;

    i=0;
    do
    {
        printf("Inserisci il numero %d: ",i);
        scanf("%d", &DIGIT[i]);
        i=i+1;
    }
    while (DIGIT[i-1]!=0);
    /* i-2 e' la posizione dell'ultimo numero
    inserito */

    media=0;
    n_pos_pari=0;
    for(j=0;j<i-1;j++)
    {
        if (DIGIT[j]>0)
            if (DIGIT[j]%2==0)
            {
                media=media+DIGIT[j];
                n_pos_pari++;
            }
    }
    media=media/n_pos_pari;
    printf("La media vale: %f\n",media);
}
```

### ESERCIZIO: Split (difficoltà: \*\*\*\*\*)

Scrivere una procedura split che, dato un array A di interi, lo divide in due array A1 ed A2 secondo la seguente regola:

1. Si inserisca la più lunga sequenza ordinata che inizia dal primo elemento nel vettore A1.
2. Partendo dall'elemento successivo, si inserisca la sequenza ordinata più lunga nel vettore A2.
3. Si continui così (eseguendo i passi 1 e 2) fino alla fine dell'array.

Per tutti gli array, si supponga che contengano solo elementi positivi e che l'ultimo elemento sia lo zero.

Per esempio, se A={1,2,4,2,5,3,5,6,4,3,2,1,3,5,6,3,0,5,3}

si avrà

A1={1,2,4,3,5,6,3,1,3,5,6}

A2={2,5,4,2,3}

(gli elementi 5 e 3 non sono importanti in quanto sono dopo il terminatore)

### Soluzione:

*Primo livello di specifica:*

```
while A non è finito
{
  toglì il primo elemento da A,
  Se è maggiore dell'ultimo elemento,
  allora mettilo nello stesso vettore
  altrimenti cambia vettore
}
```

*Secondo livello di specifica:*

```
Sia N la dimensione massima dei 3 vettori.
Na la posizione corrente nel vettore A.
Na1 la posizione corrente nel vettore A1.
Na2 la posizione corrente nel vettore A2.
Ultimo: Ultimo elemento considerato.
UltVet: valore Boolean che indica:
      TRUE: l'ultimo vettore usato è Na1
      FALSE: l'ultimo vettore usato è Na2

Inizializza le variabili;
while (A[Na]!=0)
{
  X = A[Na];
  incrementa Na;
  Se X<Ultimo
  allora { UltVet= !UltVet; }
  Se UltVet
    allora { A1[Na1]=X;
              incrementa Na1; }
    altrim { A2[Na2]=X;
             incrementa Na2; }
  Ultimo=X;
}
```

```

#define TRUE 1
#define FALSE 0

void split(int A[], int A1[], int A2[])
{
    int X, Ultimo, UltVet, Na, Na1, Na2;

    Ultimo=A[0];
    UltVet=TRUE;
    Na=0;
    Na1=0;
    Na2=0;
    while (A[Na]!=0)
    {
        X=A[Na];
        Na++;
        if (X<Ultimo)
            { UltVet = !UltVet;}
        if (UltVet)
            { A1[Na1] = X;
              Na1++;}
        else
            { A2[Na2] = X;
              Na2++;}
        Ultimo=X;
    }
    A1[Na1]=0; /* Mette a zero l'ultimo*/
    A2[Na2]=0;
}

```

### Esercizio: Merge (difficoltà \*\*\*\*)

Scrivere una procedura Merge che, dati due vettori ordinati di elementi positivi e terminanti con zero, produce il vettore unione dei due vettori (ordinato).

### Soluzione:

*Primo livello di specifica:*

```

merge(A,B,M)
{
    Se A è vuoto, allora copia B in M
    Se B è vuoto, allora copia A in M
    sia Xa= primo elemento di A
    sia Xb= primo elemento di B
    Se Xa<Xb
        allora togli il primo elemento da A
        aggiungi Xa in fondo a M
        merge(A,B,M)
    altrimenti
        togli il primo elemento da B
        aggiungi Xb in fondo a M
        merge(A,B,M)
}

```

*Secondo livello di specifica:*

```
Sia Na l'indice dell'elemento a cui  
siamo arrivati in A  
Sia Nb l'indice dell'elemento a cui  
siamo arrivati in B  
Sia Nm l'indice dell'elemento a cui  
siamo arrivati in M  
merge(A,B,M,Na,Nb,Nm)  
{  
    Se A[Na]==0  
        copia(M,B,Nm,Nb)  
    altrimenti  
        Se B[Nb]==0  
            copia(M,A,Nm,Na)  
        altrimenti  
            Se A[Na]<B[Nb]  
                {M[Nm]=Na;  
                 merge(A,B,M,Na+1,Nb,Nm+1);  
            }  
        altrimenti  
            {M[Nm]=Nb;  
             merge(A,B,M,Na,Nb+1,Nm+1);  
        }  
}
```

*Codifica:*

```
void copia(int A[], int B[], int Na, int Nb);  
  
void merge(int A[],int B[],int M[],int Na, int  
Nb,int Nm)  
{  
    if (A[Na]==0)  
        { copia(M,A,Nm,Na);}  
    else if (B[Nb]==0)  
        {copia(M,B,Nm,Nb);}  
    else  
        if (A[Na]<B[Nb])  
            { M[Nm]=A[Na];  
              merge(A,B,M,Na+1,Nb,Nm+1); }  
        else  
            { M[Nm]=B[Nb];  
              merge(A,B,M,Na,Nb+1,Nm+1); }  
}  
  
void copia(int A[], int B[], int Na, int Nb)  
{  
    if (B[Nb]!=0)  
        { A[Na]=B[Nb];  
          copia(A,B,Na+1,Nb+1); }  
}
```

### Esercizio: Split-Merge

Si costruisca una procedura che ordina un vettore di interi, alternando una fase di Split con una di Merge.

### Soluzione:

*Primo livello di specifica:*

```
Finché il vettore A non è ordinato
{split(A,A1,A2);
 merge(A1,A2,A);
}
```

*Secondo livello di specifica:*

Il vettore è ordinato se l'ultima fase di split ha prodotto uno dei vettori vuoto (in particolare, il secondo)

```
{split(A,A1,A2);
 merge(A1,A2,A);
} Finché il vettore A2 non è vuoto
```

*Codifica:*

```
#define N 50
void split-merge(int A[])
{  int A1[N];
   int A2[N];

   do
   {   split(A,A1,A2);
       merge(A1,A2,A,0,0,0);
   } while (A2[0] !=0);
}
```

### ESERCIZIO: Vettore di numeri primi (difficoltà \*\*\*)

Si consideri un vettore di elementi interi chiamato "Primi". L'elemento  $i$ -esimo del vettore deve essere a 1 se l'intero  $i$  è primo, 0 altrimenti. Si costruisca una procedura che, dato un numero  $N$ , produce il vettore "Primi" da 2 fino ad  $N$  (gli elementi più piccoli di 2 non sono considerati).

**Osservazione:** Un numero  $j$  è primo se non è divisibile per alcun numero primo più piccolo di lui.

### Soluzione:

*Primo livello di specifica:*

Se  $N=2$

Scrivi 1 in posizione  $N$ -esima

Altrimenti

Calcola il vettore fino alla posizione  $N-1$

Scorri il vettore; verifica se  $N$  è divisibile per gli elementi a 1

Se  $N$  non è divisibile, metti a 1 l'elemento  $N$ -esimo.

*Secondo livello di specifica:*

sia  $N_{primo}$  una variabile che è vera se crediamo che l'elemento sia primo (non abbiamo ancora avuto prova che non è primo)

Se  $N=2$

Scrivi 1 in posizione  $N$ -esima

Altrimenti

Calcola il vettore fino alla posizione  $N-1$

Scorri il vettore con un indice  $i$ .

Se l'elemento  $i$ -esimo è a 1, prova a dividere  $N$  per  $i$ .

Se la divisione dà zero, allora  $N$  non è primo  
altrimenti continua.

```

#define TRUE 1
#define FALSE 0
void cal_primi(int Primi[], int N)
{
    int Nprimo=TRUE;
    int i;
    if (N==2)
    { Primi[N]=TRUE; }
    else
    {cal_primi(Primi[], N-1);
     while (Nprimo==TRUE && i<N)
     { if (Primi[i]==TRUE)
       { Nprimo = ((N%i==0)?TRUE:FALSE); }
       i++;
     }
     Primi[N]=Nprimo;
    }
}

```

### Esercizio: Crivello di Eratostene (difficoltà \*\*\*)

Si consideri un vettore di elementi interi chiamato "Primi". L'elemento  $i$ -esimo del vettore deve essere a 1 se l'intero  $i$  è primo, 0 altrimenti (come nell'esercizio precedente). Si costruisca una procedura che, dato un numero  $N$ , produce il vettore "Primi" da 2 fino ad  $N$  senza utilizzare la divisione (né l'operatore % né l'operatore /).

**Osservazione:** Si possono eliminare tutti i multipli di 2, poi tutti i multipli di 3, poi tutti i multipli di 5, ... di tutti i numeri primi calcolati finora. Rimarranno solo i numeri primi.

### Soluzione:

*Primo livello di specifica:*

Metto a TRUE tutto il vettore, da 2 a  $N$

Considero un indice  $i$  che va da 2 ad  $N$

```

{   Se Primi[i] è vero, allora elimino tutti i multipli di i;
    incremento i
}

```

*Codifica:*

```

#define TRUE 1
#define FALSE 0
void crivello(int Primi[], int N)
{
    int i,j;
    for (i=2; i<N; i++)
        Primi[i]=TRUE;
    for (i=2; i<N; i++)
        if (Primi[i]==TRUE)
            { for (j=2*i; j<N; j += i)
              Primi[j] = FALSE;
            }
}

```

### Esercizio: (difficoltà \*)

Si costruisca una procedura C che, dato il vettore Primi definito nell'esercizio precedente, stampa i numeri primi da 2 ad N.

### Soluzione:

```
#define TRUE 1
#define FALSE 0
procedure stampa(int Primi[], int N)
{
    for (i=2; i<N; i++)
        if (Primi[i]==TRUE)
            {printf("%d ",i);}
    printf("\n");
}
```

### ESERCIZIO

Si realizzi un programma C che legga da standard input i dati relativi ai corsi tenuti presso una università. In particolare, per ogni corso vengono dati:

- **denominazione** del corso: una stringa che riporta il nome del corso;
- **nome** del docente: una stringa che rappresenta il nome del docente del corso;
- **cognome** del docente: una stringa che rappresenta il cognome del docente del corso;
- **iscritti**: un intero che indica il numero di studenti che frequentano il corso.

Il programma deve stampare la denominazione ed il cognome relativi a tutti i corsi che hanno il numero di iscritti maggiore o uguale alla media aritmetica degli iscritti (calcolata su tutti i corsi).

```

#include <stdio.h>
#define N 300
struct stud {
    char denominazione[20];
    char nome_docente[10];
    char cognome_docente[15];
    int studenti;
} corsi[N];

main()
{int i, somma, nc;
 float media;

printf("Inserisci il numero dei corsi ");
scanf("%d", &nc);

/* inserimento dati */
for (i=0; i<nc; i++)
{
    printf("Inserisci il nome del corso ");
    scanf("%s",corsi[i].denominazione);
    printf("Inserisci il nome del docente ");
    scanf("%s",corsi[i].nome_docente);
    printf("Inserisci il cognome del docente ");
    scanf("%s",corsi[i].cognome_docente);
    printf("Inserisci il numero degli iscritti");
    scanf("%d",&corsi[i].studenti);
}
somma=0;
for (i=0; i< nc; i++)
    somma=somma + corsi[i].studenti;
media=((float)somma)/nc;
for (i=0; i< nc; i++)
    if (corsi[i].studenti>=media)
        printf("%s %s\n",corsi[i].denominazione,
            corsi[i].cognome_docente);
}

```

## ESERCIZIO

Leggere da input alcuni caratteri alfabetici maiuscoli (per ipotesi al massimo 10) e riscriverli in uscita evitando di ripetere caratteri già stampati.

### Soluzione:

```

while <ci sono caratteri da leggere>
{
    <leggi carattere>;
    if <non già memorizzato>
        <memorizzalo in una struttura dati>;
};
while <ci sono elementi della struttura
dati>
    <stampa elemento>;

```

Occorre una struttura dati in cui memorizzare (senza ripetizioni) gli elementi letti in ingresso.

```
char A[10];
```



### Codifica:

```
#include <stdio.h>

main()
{ char A[10], c;
  int i, j, inseriti, trovato;

  inseriti=0;
  printf("\n Dammi 10 caratteri: ");
  for (i=0; (i<10); i++)
  {
    leggi_carattere(&c);
    printf("letto %c\n", c);
    /* verifica unicita' */
    trovato=0;
    for(j=0;(j<inseriti)&&!trovato; j++)
      { if (c==A[j])
        trovato=1;
      }
    if (!trovato)
      { A[j]=c;
        inseriti++;
        printf("inserito %c\n", c);
      }
    trovato=0;
  }
  printf("Inseriti %d caratteri \n",
        inseriti);
}

void leggi_carattere(char *c)
{ char s[1];
  scanf("%1s",s);
  *c=s[0];}
```

### ESERCIZIO

Programma che calcola la lunghezza di una stringa.

```
#include <stdio.h>

/* lunghezza di una stringa */

main()
{
  char   str[81]; /* stringa di al max.
                  80 caratteri */
  int    i;

  printf("\nImmettere una stringa: \t");
  scanf("%s",&str); /* %s formato
                    stringa */

  for (i=0; str[i]!='\0'; i++);

  printf("\nLunghezza: \t %d\n",i);
}

• Vengono acquisiti i caratteri in ingresso fino al primo carattere
  di spaziatura (bianco, newline)
```

## ESERCIZIO: Analisi di un programma

Dato il seguente programma C:

```
#include <stdio.h>
#define D 4
float A(float *V, int k)
{int i;
 float s=0.0;
 for(i=0;i<D; i+=k)
 {
     V[i]=i;
     s=s+V[i];
 }
 return s;
}

main()
{ float V[]={1.5, 2.5, 3.5, 4.5};
 int i;
 for (i=1; i<D; i=i+1)
     V[i]=V[i-1];
 printf("%f\n", A(V,2));
 for(i=0; i<D; i=i+1)
     printf("%f", V[i]);
}
```

Cosa stampa il programma? La risposta deve essere opportunamente motivata.

## Soluzione

Dopo la prima iterazione il vettore **V** vale

**V[D]={1.5,1.5,1.5,1.5}**

perché il ciclo copia ciascun elemento nella posizione successiva, quindi il primo elemento viene replicato per tutto il vettore.

La chiamata della funzione **A** restituisce 2 che viene stampato dalla **printf**.

Dopo la chiamata il vettore **V** vale

**V[D]={0,1.5,2,1.5}**

Il secondo ciclo stampa quindi

**0,1.5,2,1.5**

## ESERCIZIO: Analisi di un programma

Indicare che cosa stampa il seguente programma. La risposta deve essere opportunamente motivata.

```
#include <stdio.h>

main()
{
    int A[6]={0,0,0,0,0,0};
    int i;

    for (i=0; i<6; i+= 3)
        A[i]= A[i] + 2*i;

    for(i=0; i<6; i++)
        if (A[i])
            printf("%d\n",A[i]);
        else
            printf("%d\n",i-A[i]);
}
```

### Soluzione

Dopo il primo ciclo for il vettore A vale:

A=

0	0	0	6	0	0
---	---	---	---	---	---

Il secondo ciclo for stampa quindi:

0	1	2	6	4	5
---	---	---	---	---	---

## ESERCIZIO

Scrivere una funzione con la seguente interfaccia:

**void concat(char s1[], char s2[], char s3[])**,  
che ponga nella stringa **s3** la concatenazione delle stringhe **s1** e **s2**. Per esempio: se **s1**="eta", **s2**="beta", al ritorno dalla chiamata a **concat** si deve avere **s3**="etabeta".

Ovviamente, non si deve fare uso di alcuna funzione di libreria relativa alla gestione delle stringhe.

```
#include <stdio.h>

void concat(char s1[],char s2[], char s3[])
{
    int i=0,j=0;

    while (s1[i] != '\0') s3[j++] = s1[i++];
    i = 0;
    while (s2[i] != '\0') s3[j++] = s2[i++];
    s3[j] = '\0';
}
```

*Oppure:*

```
void concat(char s1[],char s2[], char s3[])
{
    int i,j;

    for (i=0; s1[i] != '\0'; i++)
        s3[i] = s1[i];
    for (j=i,i=0; s2[i] != '\0'; i++,j++)
        s3[j] = s2[i];
    s3[j] = '\0';
}
```

## ESERCIZIO

Scrivere una funzione con la seguente interfaccia

**void Scambia(char s[])**, che scambi il primo carattere della stringa *s* con l'ultimo. E' possibile, ma non necessario, fare uso di funzioni di libreria per la gestione delle stringhe.

```
void Scambia(char s[])
{
    int i;
    char c;

    for (i=0; s[i] != '\0'; i++);
    /* all'uscita dal ciclo i vale strlen(s) */

    c = s[0];
    s[0] = s[i-1];
    s[i-1] = c;
}
```

## ESERCIZIO

Scrivere una funzione che copi in una stringa destinazione una parte di una stringa sorgente, dal carattere **n1** al carattere **n2**. Scrivere un main cliente di prova.

```
#include <stdio.h>
#define N 81

void sottostr(char sorg[],int n1,int n2,char
               dest[]);

main() {
    char s[N],d[N];
    int n1,n2;
    printf("Inserire una stringa:");
    scanf("%s",s);
    printf("\ninizio sottostringa: ");
    scanf("%d",&n1);
    printf("fine sottostringa: ");
    scanf("%d",&n2);
    sottostr(s,n1,n2,d);
    printf("\nLa sottostringa e': ");
    printf("%s",d);
}

void sottostr(char sorg[],int n1, int n2,
               char dest[])
{
    int i=0;
    while ((sorg[n1] != '\0') && (n1<=n2))
        dest[i++] = sorg[n1++];
    dest[i] = '\0';
}
```

### Osservazione

Non viene fatto alcun controllo sugli estremi della sottostringa, né sulla dimensione della stringa destinazione. Esercizio: modificare il programma inserendo tali controlli.

## ESERCIZIO

Progettare un programma che legga una stringa da terminale e riconosca se la stringa è palindroma (per esempio: “acca”, “abcXcba” sono stringhe palindrome).

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

main() {
    char parola[81];
    int i=0,n;

    printf("\nInserire parola: ");
    scanf("%s",parola);
    n = strlen(parola);

    while (i<n/2 && parola[i]==parola[n-i-1])
        i++;

    if (i==n/2) printf("\npalindroma");
    else printf("\nnon palindroma");
}
```

## ESERCIZIO

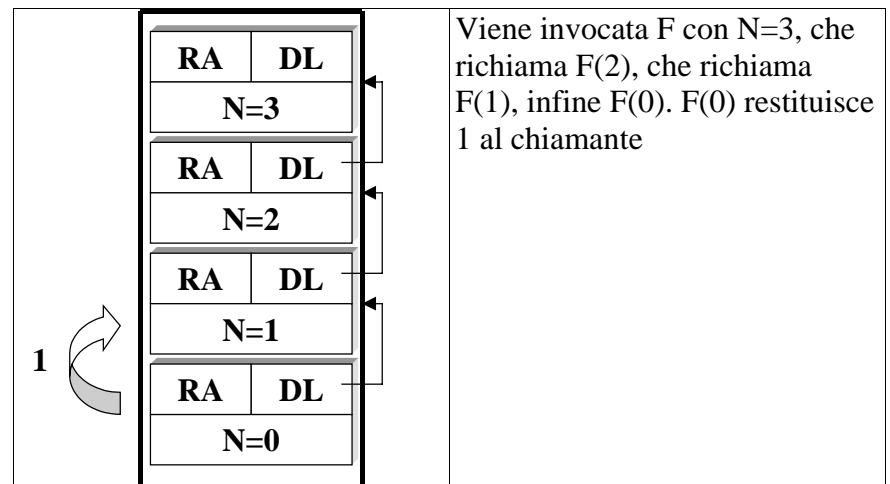
Si consideri la seguente funzione F la cui specifica è data in modo ricorsivo (si supponga N intero):

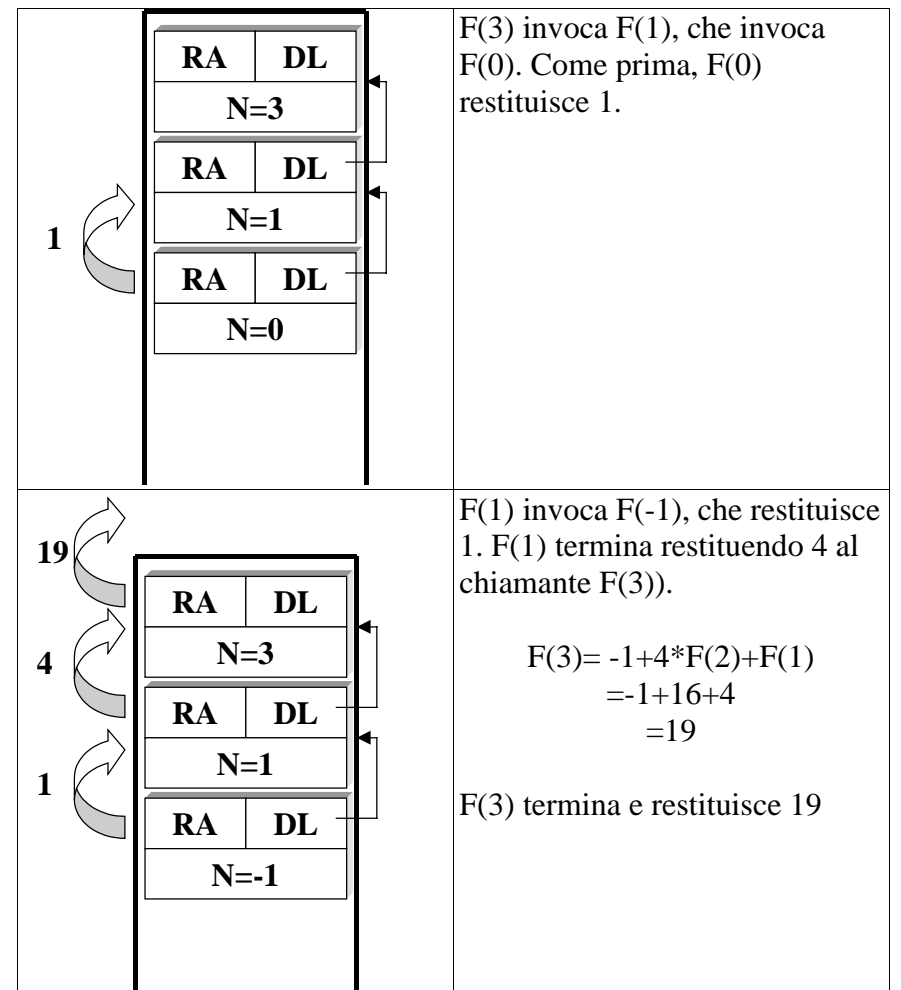
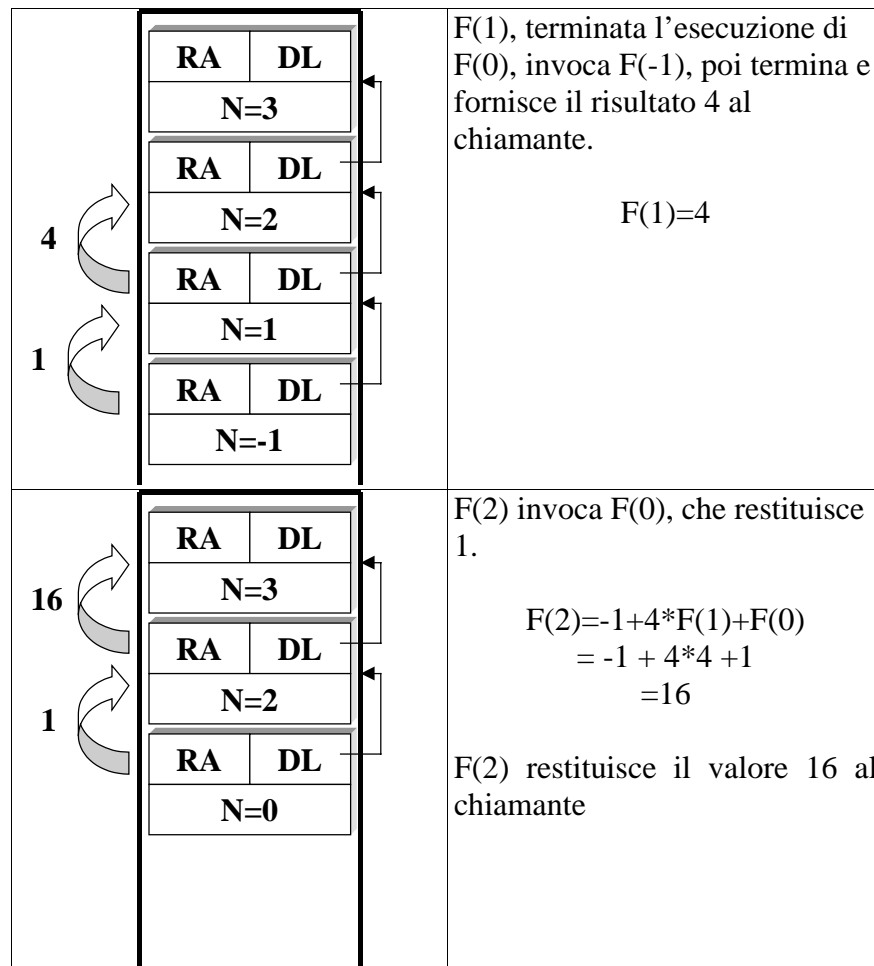
$F(N)$  = restituisce 1 se  $N \leq 0$ ,  
-1+4\*F(N-1) + F(N-2), altrimenti

1. Si scriva la funzione C che realizzerebbe tale specifica
2. Si scriva il risultato della funzione quando chiamata con  $N=3$  e si mostri la sequenza dei record di attivazione;

## SOLUZIONE:

```
int F(int N)
{
    if (N<=0) return 1
    else return -1+4*F(N-1) + F(N-2);
}
```





## ESERCIZIO

Si consideri la seguente funzione F la cui specifica è data in modo ricorsivo (si supponga N intero):

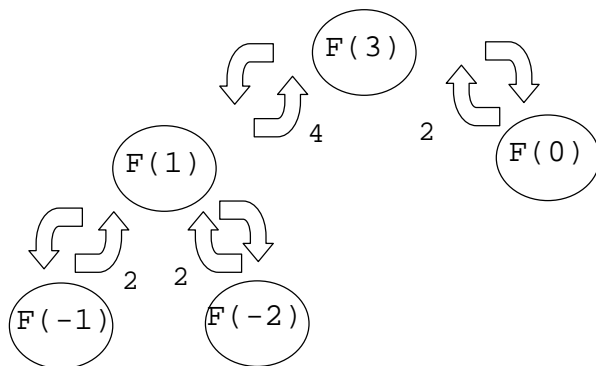
$F(N)$  = restituisce 2 se  $N \leq 0$ ,

$F(N-2) * F(N-3)$ , altrimenti

- Si scriva il risultato della funzione quando chiamata con  $N=3$  e si mostrino i valori intermedi assunti da N;
- Si scriva la funzione C che realizzerebbe tale specifica
- Si mostrino i record di attivazione nello stack

## Soluzione:

Sequenza di attivazioni:

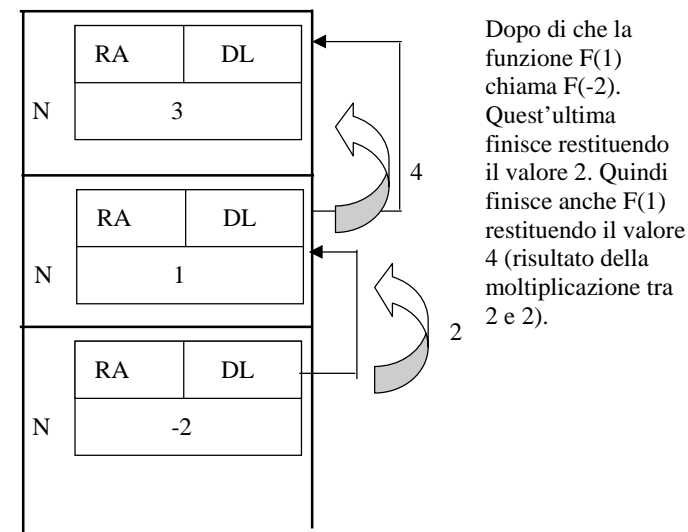
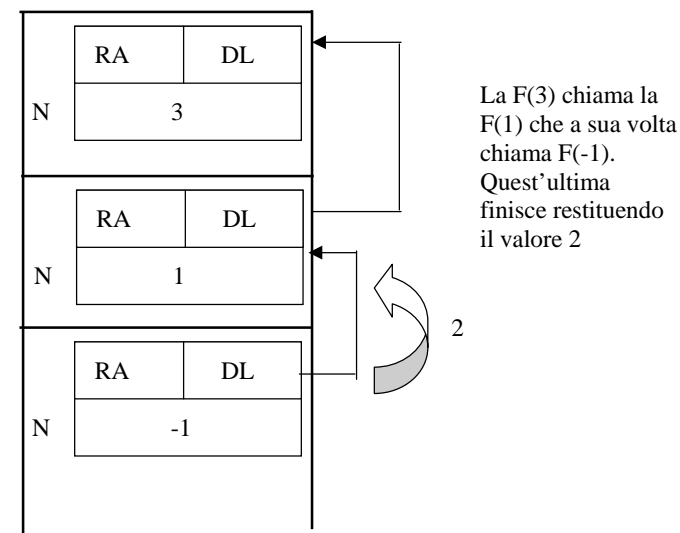


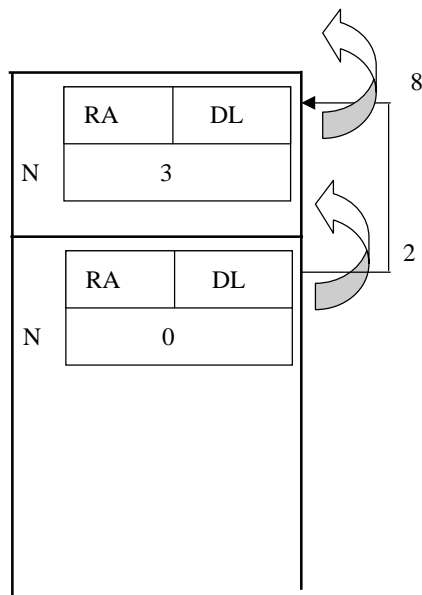
**F(3)** restituisce il valore 8

Valori assunti da N: 3 1 -1 -2 0

```
int F(int N)
{if (N<=0) return 2;
 else return F(N-2)*F(N-3);
}
```

Record di attivazione della chiamata con  $N = 3$





Dopo di che la funzione  $F(3)$  chiama  $F(0)$ . Quest'ultima finisce restituendo il valore 2. Quindi finisce anche  $F(3)$  restituendo il valore 8 (risultato della moltiplicazione tra 4 e 2).

## ESERCIZIO: Numeri Primi

Si scriva una funzione  $C$  che calcola se un numero è primo, utilizzando la seguente specifica definita ricorsivamente:

- 2 è primo.
- un numero  $N$  è primo se non è divisibile per alcun numero primo strettamente compreso fra 2 ed  $N/2$ .

## SOLUZIONE:

```
int div(N,M)
{ return (N % M) == 0;
}

int primo(int N)
{ int i, T = 1;
  if (N == 2)
    return 1
  else
    {for (i=2; i<N/2; i++)
      if (primo(i))
        Trovato = Trovato && !div(N,i);
      return Trovato;
    }
}
```



## ESERCIZIO

Si consideri la seguente procedura P la cui specifica è data in modo ricorsivo (si supponga N intero):

$P(N)$  = stampa  $N$  se  $N \leq 10$ ,  
          stampa  $N$  ed invoca  $P(N-10)$ , altrimenti

- Scrivere il codice C di tale procedura.
- La procedura è tail ricorsiva?
- Si scriva la sequenza di valori stampati quando la procedura è chiamata con  $N=30$ . Si mostri anche la sequenza dei record di attivazione, sia nel caso il compilatore effettui l'ottimizzazione tail, sia se non la effettua.

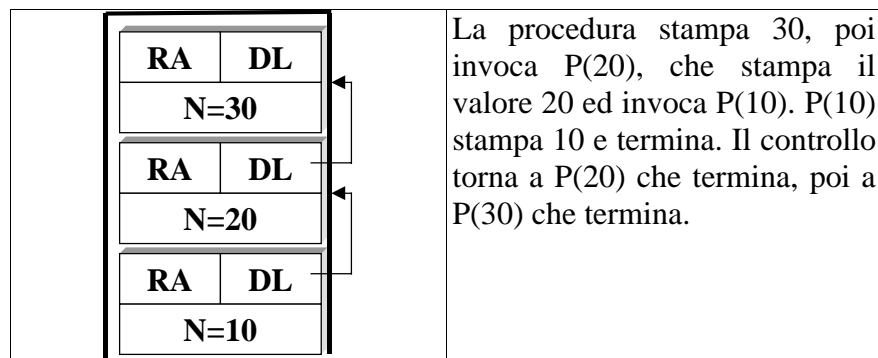
## Soluzione:

a) Codice della procedura

```
void P(int N)
{ printf("%d ",N);
  if (N>10)
    { P(N-10); }
}
```

b) La procedura è tail ricorsiva, in quanto la chiamata ricorsiva è l'ultima operazione effettuata.

c) La procedura stampa la sequenza  
30 20 10



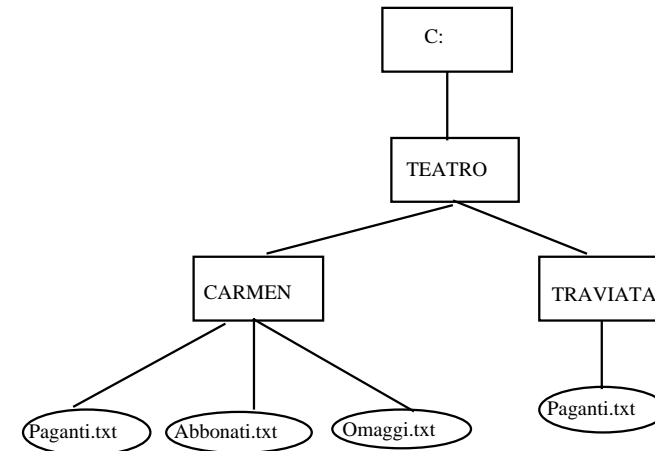
Se il compilatore effettua l'ottimizzazione tail, viene riutilizzato sempre lo stesso record di attivazione:

RA	DL
N = <del>30</del> 20 10	

La procedura stampa 30, poi invoca P(20), senza allocare un nuovo record di attivazione. Stampa il valore 20 ed invoca P(10). P(10) stampa 10 e termina. Il controllo torna direttamente al programma chiamante.

## ESERCIZIO

Si consideri il file system in figura:



Supponendo di trovarsi come direttorio corrente in CARMEN come ci si può riferire al file Paganti.txt sotto il direttorio TRAVIATA tramite percorso relativo. Come si può riferire al medesimo file tramite percorso assoluto?

Supponendo di trovarsi come direttorio corrente in TEATRO come ci si può riferire al file Abbonati.txt sotto il direttorio CARMEN tramite percorso relativo. Come si può riferire al medesimo file tramite percorso assoluto?

## SOLUZIONE

Percorso relativo

../TEATRO/TRAVIATA/Paganti.txt

Percorso assoluto

C:/TEATRO/TRAVIATA/Paganti.txt

Percorso relativo

CARMEN/Abbonati.txt

Percorso assoluto

C:/TEATRO/ CARMEN/Abbonati.txt