Esercizi di Informatica

A cura di Massimiliano Colombo ed Elisabetta Di Nitto

(Vi preghiamo di segnalarci ogni errore che vi capitasse di trovare.)

Indice

1. Strutture di controllo e tipi semplici

Esercizio 1.01

Dato il seguente programma, per quali valori della variabile a viene stampata la stringa: “la condizione e` vera”?

```
#include <stdio.h>
main()
{
  int a;
  scanf(“%d”, &a);
  if ((a > 0) && (a <= 0))
    printf(“La condizione e` vera\n”);
  else printf(“La condizione e` falsa\n”);
}
```

Soluzione

Non esiste nessun valore di a per cui la stringa indicata viene stampata. Infatti, la condizione dell’if è sempre falsa perchè non è mai possibile che a sia allo stesso tempo maggiore e minore o uguale a zero (si noti che le due parti della condizione sono legate da un AND). Di conseguenza, per ogni valore di a verrà sempre eseguita l’istruzione che si trova nel ramo else dell’if in questione, e questo causerà la stampa della stringa “la condizione è falsa”.

Esercizio 1.02

Si consideri il seguente frammento di programma:

```
if (a < 10)
{
  if (b < 20)
       printf(“prova1”);
  else printf(“prova2”);
}
else printf(“prova2”);
```

È possibile scriverlo in modo equivalente senza utilizzare il secondo if?

Soluzione

Il frammento di programma può essere riscritto nel modo seguente:

```
if (a < 10 && b < 20)
  printf(“prova1”);
else printf(“prova2”);
```

Per dimostrare che il nuovo frammento è equivalente a quello di partenza, possiamo esaminarne la struttura ed il modo con cui si comportano in base ai valori delle variabili a e b. In particolare, guardando alla struttura del primo frammento, notiamo che il ramo else più esterno, che dà luogo alla stampa della stringa “prova2” viene eseguito per i valori di a maggiori o uguali a 10, indipendentemente dal valore di b. Guardando alla struttura del secondo programma, ci accorgiamo che per questi stessi valori di a viene stampata la stessa stringa. In entrambi i frammenti, infine, per valori di a minori di dieci, la decisione se stampare la stringa “prova1” o “prova2” dipende dal valore di b. In particolare, per valori di b minori di 20 viene stampata “prova1”.

Esercizio 1.03

Quante volte viene eseguita l’operazione di stampa nel seguente programma?

```
main()
{
  int i;
  i = 5;
```

```
  while (i>=0)
    printf("Il valore di i e`: %d\n", i);
}
```

Soluzione

La stringa viene stampata infinite volte perchè la condizione del while rimane sempre vera dal momento che non ci sono istruzioni nel ciclo che consentano di far descrescere il valore di i.

Esercizio 1.04

Si considerino i seguenti due esempi:

….
int a;
…
a = 12;
while(a<10)
a++;
…
	….
	int a;
	…
	a = 12;
	while(a>10)
	a++;
	…

Quante volte vengono eseguite le istruzioni nel corpo di ciascuno dei due while?

Soluzione

Nel primo frammento di programma l'istruzione contenuta nel ciclo non viene eseguita mai perchè dato l'assegnamento del valore 12 ad a, la condizione del while non è mai verificata. Nel secondo frammento di programma, dal punto di vista logico il corpo del ciclo while viene eseguito infinite volte perchè la condizione del while rimane sempre vera. Dal punto di vista tecnico, il numero di interi rappresentabili nel calcolatore è finito. Quando a è pari al massimo valore rappresentabile, un ulteriore incremento ha come effetto il fatto di assegnare ad a il minimo numero negativo rappresentabile. Quando questo si verifica, la condizione del ciclo diventa falsa e l'esecuzione del corpo del ciclo termina.

Esercizio 1.05

Si considerino i seguenti due frammenti di programma, scritti in linguaggio C. I risultati che producono sono equivalenti? Si motivi la risposta.

```
/* primo frammento */
int i, a[10];
for (i = 0; i<10; i++)
    a[i] = i;


/* secondo frammento */
int i, a[10];
i = 9;
while (i>=0)
{
     a[i] = i;
     i--;
}
```

Soluzione

I due frammenti di programma producono risultati equivalenti. In particolare, entrambi assegnano agli elementi di un array di 10 interi un valore che dipende dalla loro posizione nell'array. Mentre il primo frammento effettua l'operazione di assegnamento di valore dall'elemento di posizione zero dell'array, il secondo frammento parte dall'elemento che si trova nell'ultima posizione (la posizione 9).

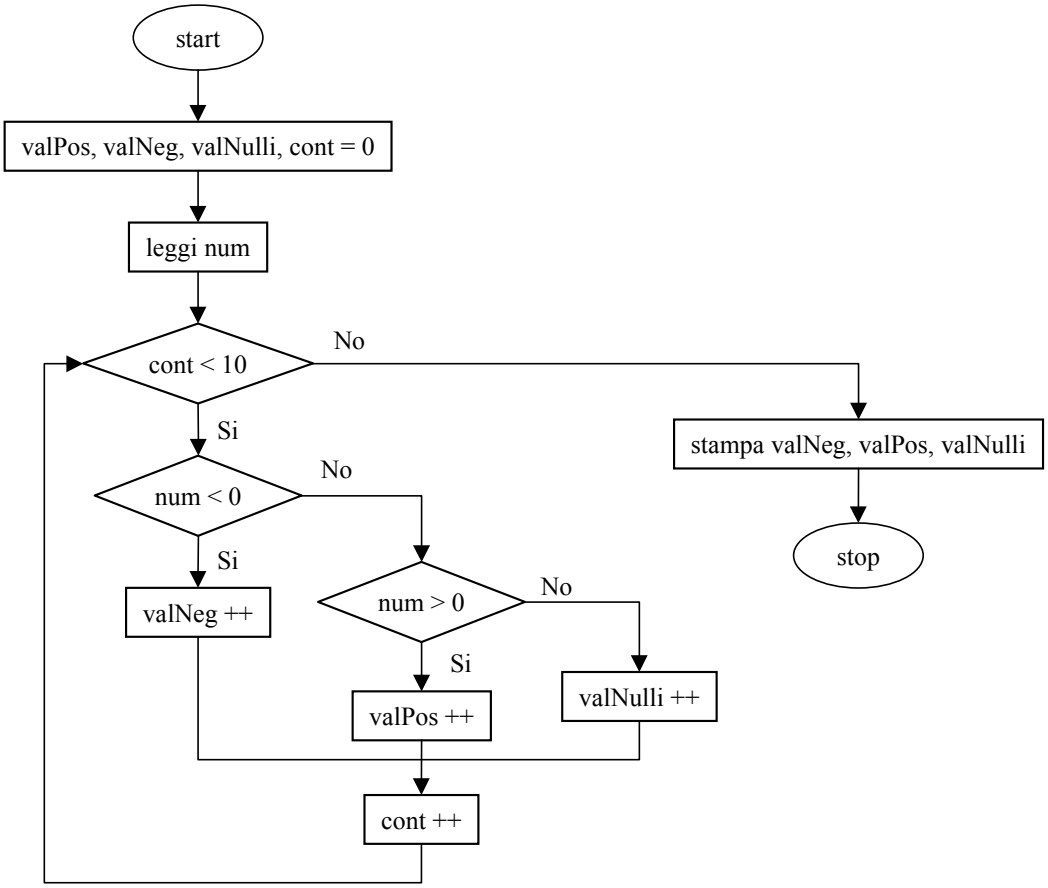
Esercizio 1.06

Si scriva un programma C (ed il relativo diagramma a blocchi) che legge da tastiera una sequenza di 10 numeri interi e, al termine, stampa a video il numero dei numeri letti che sono maggiori di zero, di quelli che sono minori di zero e di quelli nulli.

Soluzione

La soluzione si basa sull'idea di incrementare tre contatori man mano che i numeri vengono letti da tastiera. I contatori indicano il numero di elementi positivi, negativi e nulli letti. Un ulteriore contatore viene utilizzato per contare il numero di valori letti in modo tale da terminare il programma quando si arriva a leggere il decimo numero.

Diagramma a blocchi



Codice

```
#include <stdio.h>

void main()
{
  int num, cont, valPos, valNeg, valNulli;

  valPos = 0;
  valNeg = 0;
  valNulli = 0;
  cont = 0;
  printf("inserisci 10 numeri\n");
  while(cont < 10)
    {
      scanf("%d", &num);
      if(num < 0)
      valNeg++;
      else if(num > 0)
      valPos++;
      else valNulli++;
      cont++;
```

```
    }
  printf("Il numero di valori positivi e`: %d\n", valPos);
  printf("Il numero di valori negativi e`: %d\n", valNeg);
  printf("Il numero di valori pari a zero e`: %d\n", valNulli);

}
```

Si noti che uno dei contatori è inutile e può essere eliminato scrivendo opportunamente il programma. Per esempio, il numero di valori pari a zero può essere calcolato a partire dalla conoscenza di valPos, valNeg, e cont. Si riscriva il programma tenendo conto di questa osservazione.

Esercizio 1.07 Calcolo del fattoriale di un numero

Si sviluppi un programma che acquisisce da tastiera un numero e ne calcola il fattoriale. Si ricorda che le regola di calcolo è la seguente:
n! = n*(n-1)! se n > 1
n! = 1 se n è pari a 0 oppure 1

Soluzione

```
#include<stdio.h>
/* Programma per il calcolo del fattoriale.
   Regola di calcolo:
   n!=n(n-1)! se n>1
   n!=1 se n=0 o n=1
*/

void main()
{
  long numero, fatt;

  printf("Questo programma calcola il fattoriale di un numero.\n");
  printf("Inserisci il numero: ");
  scanf("%d", &numero);
  printf("\n");

  if (numero < 0)
    printf("Errore! Il numero deve essere positivo\n");
  else
    {
      if(numero == 1 || numero == 0)
         fatt = 1;
       else
                 {
                   fatt = 1;
                   while(numero > 1)
                     {
                       fatt = fatt*numero;
                       numero = numero - 1;
                     }
                 }
      printf("Il risultato e` %d\n", fatt);
    }
}
```

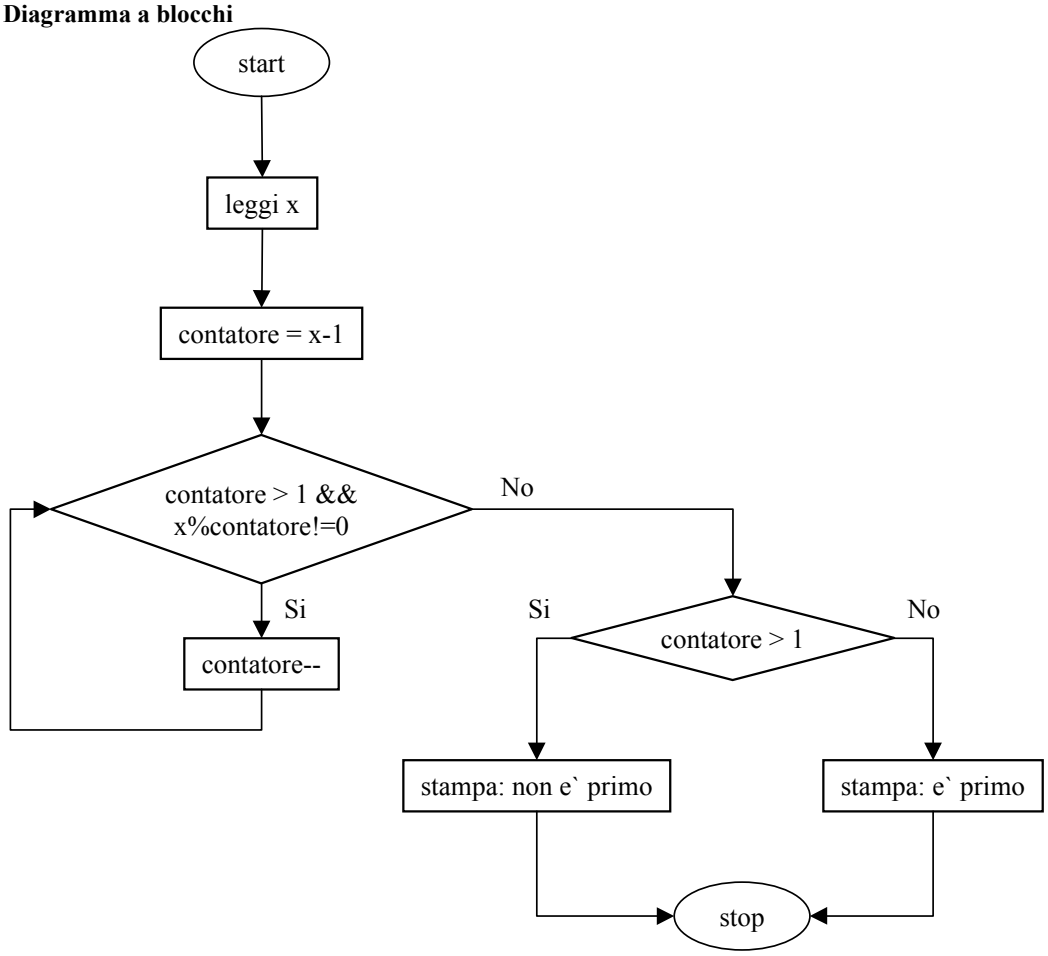
Esercizio 1.08

Si implementi un programma che decide se un numero dato dall'utente è primo oppure no. Si ricorda che un numero si dice primo se non è divisibile per nessuno dei valori compresi tra il numero stesso e 1. Si definisca il diagramma a blocchi prima di procedere alla scrittura del codice.

Soluzione

La soluzione proposta si basa sull'idea di calcolare il resto della divisione tra il numero letto da tastiera e tutti i numeri compresi tra questo e 1. Se questo resto è diverso da zero per tutti i valori considerati, allora il numero è certamente primo. Si provi a cercare una soluzione più ottimizzata.

Diagramma a blocchi



Codice

```
/* programma che determina se un numero e` primo o no */

#include <stdio.h>

void main()
{
  int x, contatore;

  printf("Inserisci un numero: ");
  scanf("%d", &x);
  printf("\n");
  contatore = x-1;
  while (contatore > 1 && x%contatore!=0)
    contatore--;
  if(contatore > 1)
    printf("il numero %d non e` primo. Il valore %d e` un
divisore\n",
         x, contatore);
  else
    printf("Il numero %d e` primo\n");
}
```

Esercizio 1.09

a) Si legga una sequenza di numeri interi e se ne stampi il valore massimo.

Ci si basi sulle ipotesi seguenti:

- il valore di ciascun elemento appartenente alla sequenza è > 0 e < 100
- non si conosce a priori il numero degli elementi appartenenti alla sequenza
- la fase di acquisizione termina quando viene letto il "terminatore" (valore 999).
- la sequenza potrebbe essere vuota.

b) Si consideri il problema del punto a) e si formuli una soluzione, stavolta ipotizzando che il numero di elementi della sequenza di ingresso sia sempre pari a 10.

Soluzione caso a)

```
#include<stdio.h> /* direttiva per il compilatore */
void main()
{
  /* parte dichiarativa del programma */
  int X, max;

  /* parte esecutiva */
  max = 0;
  /* per guidare l'utente */
  printf("Introduci un numero. 999 e` il terminatore: ");

  /*acquisisci il numero */
  scanf("%d",&X);

  while (X != 999)
    {
      /* i valori non compresi tra 0 e 100 vengono saltati */
      if (X>max && X>0 && X<100)
      max = X;
      printf("Introduci un altro numero. 999 e` il teminatore: ");
      scanf("%d",&X);
    }
  /* controllo del caso di sequenza vuota */
  if(max == 0)
    printf("La sequenza di numeri e` vuota!");
  else
    printf("\nil massimo vale %d\n",max);
}
```

Soluzione caso b)

```
#include<stdio.h> /* direttiva per il compilatore */
void main()
{
  /* parte dichiarativa del programma */
  int X, max, cont;

  /* parte esecutiva */
  max = 0;
  cont = 0;
  /* per guidare l'utente */
  printf("Introduci 10 numeri\n");
  while (cont<10)
    {
      scanf("%d",&X);

      /* i valori non compresi tra 0 e 100 vengono saltati,
      il contatore viene incrementato solo in caso contrario */
```

```
      if (X>0 && X<100)
      {
        cont = cont + 1;
        if (X > max)
          max = X;
      }
    }
    printf("\nil massimo vale %d\n",max);
}
```

Esercizio 1.10

Si legga una sequenza di numeri interi e se ne stampi il valore medio.
Le ipotesi sono:

- il valore di ciascun elemento appartenente alla sequenza è > 0 e < 100
- non si conosce a priori il numero degli elementi appartenenti alla sequenza
- la fase di acquisizione termina quando viene letto il "terminatore" (valore 999)
- la sequenza potrebbe essere vuota.

Soluzione

```
#include<stdio.h> /* direttiva per il compilatore */
void main()
{
  /* parte dichiarativa del programma */
  int X, somma, cont;
  float media;
  /* parte esecutiva */

  /* inizializzazione delle variabili */
  somma = 0;
  cont = 0;
  /* per guidare l'utente */
  printf("Introduci un numero. 999 e` il terminatore: ");
  /*acquisisci il numero */
  scanf("%d",&X);

  /* calcolo della somma dei numeri nella sequenza */
  while (X != 999)
    {
      /* i valori non compresi tra 0 e 100 vengono saltati */
      if (X>0 && X<100)
      {
        somma = somma + X;
        cont = cont + 1;
      }
      printf("Introduci un altro numero. 999 e` il teminatore: ");
      scanf("%d",&X);
    }
  /* controllo del caso di sequenza vuota */
  if(cont == 0)
    printf("La sequenza di numeri e` vuota!");
  else
    {
      /* calcolo della media */
      media = (float) somma/cont;
      printf("\nla media e` %f\n",media);
    }
}
```

Esercizio 1.11

Si legga una sequenza di numeri interi. Ogni volta che viene letto il valore 0, si stampi la frase “ho letto uno 0”. Se nella sequenza letta non e’ contenuto neppure uno 0, si stampi la frase “non ho trovato nessuno 0”.
Le ipotesi sono:

- il valore di ciascun elemento appartenente alla sequenza è maggiore o uguale a 0 e < 100
- non si conosce a priori il numero degli elementi appartenenti alla sequenza
- la fase di acquisizione termina quando viene letto il “terminatore” (valore 999)
- la sequenza potrebbe essere vuota.

Soluzione

```
#include<stdio.h> /* direttiva per il compilatore */
void main()
{
  /* parte dichiarativa del programma */
  int X;
  int indicaZero;
  /* parte esecutiva */

  /* inizializzazione delle variabili */
  indicaZero = 0; /* cambiera` valore quando si trova uno zero */
  /* per guidare l'utente */
  printf("Introduci un numero. 999 e` il terminatore: ");
  /*acquisisci il numero */
  scanf("%d",&X);

  while (X != 999)
    {
      if (X==0)
      {
        printf("ho letto uno 0\n");
        indicaZero = 1;
      }
      printf("Introduci un altro numero. 999 e` il teminatore: ");
      scanf("%d",&X);
    }
  if(indicaZero == 0)
    printf("Non ho trovato nessuno 0\n");
}
```

Esercizio 1.12

Si leggano due numeri interi di valore maggiore o uguale a 0 e si stampi il prodotto dei due numeri letti calcolato per somme successive. Si ottimizzi l’esecuzione del ciclo di calcolo.

Soluzione

```
/* attenzione! Questo programma funziona correttamente solo con i
   numeri positivi */
#include<stdio.h> /* direttiva per il compilatore */
void main()
{
  int X, Y, min, num, ris;

  ris = 0;
  printf("Inserisci il primo numero positivo\n");
  scanf("%d", &X);
  printf("Inserisci il secondo numero positivo\n");
  scanf("%d", &Y);

  /* stabilisco quale dei due numeri e` minore per
```

```
     minimizzare il numero di somme da effettuare */
  if(X < Y)
    {
      min = X;
      num = Y;
    }
  else
  {
    min = Y;
    num = X;
  }

  while(min > 0)
    {
      ris = ris + num;
      min = min - 1;
    }
  printf("Il risultato e` %d\n", ris);
}
```

Esercizio 1.13

Si scriva un programma che acquisisce una sequenza di caratteri dallo standard input e restituisce sullo standard output la sequenza codificata. La codifica viene effettuata sostituendo a ciascun carattere alfabetico il carattere successivo nella tabella ASCII.

Soluzione

```
#include <stdio.h>
void main()
{
  char ch;
  printf("Digita la sequenza di caratteri: ");

  /* acquisizione del primo carattere */
  scanf("%c", &ch);

  /* l'acquisizione di nuovi caratteri termina quando si incontra il
     carattere # */
  while(ch!='#')
    {
      /* controlla se il carattere letto e` alfabetico */
      if (((ch>='a') && (ch<='z')) || ((ch>='A') && (ch<='Z')))
      /* stampa il carattere successivo a quello letto */
      printf("%c", ch+1);
      else /* il carattere non e` alfabetico */
      printf("%c", ch);
      /* acquisizione di un nuovo carattere */
      scanf("%c", &ch);
    }
}
```

2. Array

Esercizio 2.01

Si consideri il seguente frammento di codice:

```
int a[100];
int i;
```

```
for(i = 0; i<500; i++)
   a[i] = 0;
```
Quale errore è stato introdotto? Motivare brevemente la risposta.

Soluzione

L'indice i nel ciclo for supera i limiti stabiliti per l'array. Di conseguenza, si cerca di assegnare il valore zero a celle di memoria che non fanno parte dell'array.

Esercizio 2.02

Si consideri la seguente dichiarazione di matrice 3x3:
```
int m[3][3];
```
Si scriva il frammento di programma C necessario per assegnare valore zero a tutti gli elementi della matrice.

Soluzione

```
for(i=0;i<3;i++)
   for(j=0;j<3;j++)
     m[i][j] = 0;
```
La presenza dei due cicli innestati consente ai due indici i e j di variare tra 0 e 3 in modo indipendente l'uno dall'altro.

Esercizio 2.03

Si scriva un programma C, che chiede all'utente di inserire 5 numeri interi inferiori al valore massimo di 80; quindi calcola la media dei valori letti; infine stampa sul video la media e, per ciascuno dei 5 numeri letti, stampa un numero di * pari al suo valore. Per esempio, se l'array contiene i valore 3, 4, 1, 6, 2 il programma stampa a video:

```
media = 3,2
***
****
*
******
**
```

Il programma verifica che i valori forniti dall'utente siano non negativi e inferiori al valore limite.

Soluzione

```
#include <stdio.h>
#define NUMERI_DA_LEGGERE 5 /* numero di numeri che devono essere
letti */

void main()
{
  int numeri[NUMERI_DA_LEGGERE]; /* array per contenere i numeri
                                    letti */
  int contatore; /* contatore per scorrere l'array numeri */
  float totale;  /* contiene la somma dei numeri letti da tastiera
                    Attenzione! E` float perche` nel calcolo della
                    media voglio ottenere un float come risultato
                   (vedi il codice relativo al calcolo) */
  int i;         /* un altro contatore per la stampa del diagramma a
                    barre */

  printf("Inserire %d numeri non negativi e minori o uguali ad 80\n",
        NUMERI_DA_LEGGERE);
  totale=0.0;
  contatore=0;
  /* ciclo per la lettura dei numeri, loro controllo e calcolo della
somma */
```

```
  while(contatore<NUMERI_DA_LEGGERE)
    {
      printf("%d: ", contatore+1);
      scanf("%d",&(numeri[contatore]));
      if(numeri[contatore]<0|| numeri[contatore]>80)
      {
        /* numero inserito non valido */
        printf("Il numero %d non e' valido\n", numeri[contatore]);
        printf("I numeri inseriti devono essere non negativi e ");
        printf("minori di 81\n");
      }
      else
      {
        /* numero inserito valido */
        totale=totale+numeri[contatore];
        contatore++;
      }
    }

  /* stampa media */
  printf("\n\nMedia: %f\n\n", totale/NUMERI_DA_LEGGERE);

  /* stampa diagramma a barre */
  for(contatore=0;contatore<NUMERI_DA_LEGGERE;contatore++)
    {
      for(i=0;i<numeri[contatore];i++)
        printf("*");
      printf("\n");
    }
}
```

Esercizio 2.04 Ordinamento per inserzione lineare

Scrivere un programma che acquisisce da tastiera una sequenza di 10 interi, li memorizza in un array e, infine, ordina gli elementi contenuti nell'array (si veda l'eserciziario Bellettini, Sbattella, ...).

Soluzione 1 (uso del construtto while)

```
#include <stdio.h>
#define MAX_DIM 10
void main()
{
  int a[MAX_DIM];
  int i, j, x;

  /* acquisizione dei valori */
  i = 0;
  while (i<MAX_DIM)
    {
      scanf("%d", &a[i]);
      i++;
    }

  /* ordinamento */
  i = 1;
  while (i<MAX_DIM)
    {
      x = a[i];
      j = i-1;
      while (j>=0 && a[j] > x)
      {
        a[j+1] = a[j];
        j--;
```

```
      }
      a[j+1] = x;
      i++;
    }
  i = 0;
  while (i<MAX_DIM)
    {
      printf("%d ", a[i]);
      i++;
    }
}
```

Soluzione 2 (uso del construtto for)

```
#include <stdio.h>
#define MAX_DIM 5
void main(int argc, char argv[]) {
      int a[MAX_DIM];
      int i, j, dep;

      /* acquisizione valori */
      printf("Inserisci %d valori\n", MAX_DIM);

      for (i=0; i<MAX_DIM; i++) {

            printf("%d) ", i+1);
            scanf("%d", &a[i]);
            printf("\n");
      }

      /* ordinamento */
      for (i=0; i<MAX_DIM-1; i++) {

            for (j=i+1; j<MAX_DIM; j++) {

                  /* scambio */
                  if (a[i]>a[j]) {

                        dep = a[i];
                        a[i] = a[j];
                        a[j] = dep;
                  }
            }
      }

      printf("\n\nValori ordinati:\n\n ");
      /* scrittura degli elementi ordinati */
      for (i=0; i<MAX_DIM; i++) printf("%d)= %d\n", i+1, a[i]);

}
```

Esercizio 2.05 Parole palindromi

Sviluppare un programma che acquisisce una sequenza di caratteri terminata dal carattere ‘#’ e stabilisce se la sequenza è palindrome oppure no (per esempio, "ada" è palindrome perche` si legge allo stesso modo sia da destra che da sinistra).

Soluzione

```
#include <stdio.h>
#define MAX_DIM 10
```

```
void main ()
{
  char stringa[MAX_DIM];
  int contatore;
  int i, j;

  /* acquisizione della stringa */
  contatore = 0;
  /* N.B. la lettura di una stringa carattere per carattere potrebbe
      dare esiti diversi in base al sistema operativo sottostante,
      Windows o Unix. Nel caso di Windows, fra l'inserimento di un
      carattere e l'altro della stringa non deve essere digitato
      alcun carattere di "Invio", altrimenti verrebbe letto anch'esso
      come un carattere utile della stringa e non come un separatore.
      L'alternativa è, ovviamente, leggere l'intera striga
      utilizzando il carattere di controllo %s nella scanf
  */
  scanf("%c", &stringa[contatore]);
  while(stringa[contatore]!='#')
    {
      contatore = contatore + 1;
      scanf("%c", &stringa[contatore]);
    }

  i=0; /* indica l'estremita` sinistra della stringa */
  j=contatore-1; /* indica l'estremita` destra della stringa */

  /* considero i caratteri contenuti in celle che si trovano agli
     estremi opposti della stringa. Se questi sono uguali mi sposto
     verso il centro. Termino quando trovo degli elementi diversi
     oppure quanto ho considerato tutti gli elementi (i e j sono
     entrambi posizionati al centro della stringa) */
  while(stringa[i]==stringa[j] && i<j)
    {
      i++;
      j--;
    }

  /* se la condizione e` vera, allora il ciclo precedente e`
terminato
     dopo aver considerato tutti gli elementi. Di conseguenza la
stringa
     e` palindrome */
  if(i>=j)
    printf("la stringa e` palindrome\n");
  else printf("la stringa non e` palindrome\n");
}
```

Esercizio 2.06 Ricerca di una sequenza in un array di caratteri

Si sviluppi un programma che rileva la presenza di particolari sequenze di caratteri in una stringa. Le sequenze da individuare sono:
ab
ae
ai
ll programma stampa la posizione della prima sequenza individuata

Soluzione

```
#include <stdio.h>

typedef enum {falso, vero} boolean;

void main()
{
  char a[20];
  int pos;
  boolean trovato;

  printf("inserisci una stringa terminata dal carattere #: ");
  pos=0;
  scanf("%c", &a[pos]);
  while(pos<20 && a[pos]!='#')
  {
      pos++;
      scanf("%c", &a[pos]);

  }
/* se l'ultimo elemento inserito non e` il # (si e` usciti dal ciclo
perche` si sono inseriti 20 caratteri, allora si opera una
sostituzione dell'ultimo carattere con # */
  if(a[pos]!='#')
     a[pos] = '#';
  trovato = falso;
  pos=0;
  while(a[pos]!='#' && trovato == falso)
      if (a[pos] == 'a' && (a[pos+1] == 'b' ||
                            a[pos+1] == 'e' ||
                            a[pos+1] == 'i'))
            trovato = vero;
      else pos = pos +1;
  if (trovato == vero)
    printf("Ho trovato una sequenza a partire da %d\n", pos);
  else printf("Non ho trovato nessuna delle sequenze cercate\n");
}
```

Esercizio 2.07 Ricerca binaria in un array di interi

Si sviluppi un programma che svolge le seguenti operazioni:
- Acquisisce una sequenza ORDINATA di interi da tastiera e li memorizza in un array. NB: si assuma che l'utente sia disciplinato e che inserisca i valori in ordine strettamente crescente.
- Richiede all'utente un valore da cercare.
- Svolge la ricerca tenendo conto del fatto che l'array è ordinato.

Oltre all'algoritmo banale, è possibile utilizzare il seguente algoritmo chiamato di **ricerca binaria**:
Si considera l'elemento centrale dell'array. Se questo è pari all'elemento cercato, l'algoritmo di ricerca termina.
Viceversa, se l'elemento cercato è minore dell'elemento centrale, si prosegue la ricerca nella porzione sinistra dell'array utilizzando lo stesso approccio indicato precedentemente. In caso contrario, si prosegue nella parte destra.
Per esempio:
l'array è di 7 posizioni e contiene i seguenti elementi: {1, 3, 5, 7, 8, 10, 13}
l'elemento cercato è 9
la posizione centrale è pari a lunghezza array/2 = 7/2 = 3
L'elemento di posizione 3 è il valore 7, che è minore dell'elemento cercato.
Si stabilisce che la nuova porzione di array da considerare è quella che va dalla posizione 4 (chiamiamola inizio) alla posizione 7 (chiamiamola fine).
La posizione centrale di questa porzione di array è pari a inizio + (fine-inizio)/2 = 4
Si reitera il procedimento nella porzione di array compresa tra il vecchio valore di inizio ed un valore di fine pari a 4 (la vecchia posizione centrale). Si termina quando si trova l'elemento cercato oppure

quando si arriva ad una situazione in cui la condizione inizio < fine non è più vera (ho considerato tutti gli elementi possibili).

Soluzione

```
#include <stdio.h>#define LUNGH 10

typedef enum {true, false} boolean;

void main()
{
  boolean trovato;
  int a[LUNGH];
  int i, elem, inizio, fine;

  printf("Inserisci una sequenza ORDINATA di numeri\n");
  for(i=0;i<LUNGH;i++)
    scanf("%d", &a[i]);

  printf("Inserisci l'elemento da cercare nella sequenza: ");
  scanf("%d", &elem);

  /* definisco le posizioni di inizio e fine della porzione di array
     che considero */
  inizio = 0;
  fine = LUNGH;
  trovato = false;
  while (trovato == false && inizio < fine)
    {
      /* mi posiziono al centro della porzione di array */
      i = inizio + (fine-inizio)/2;
      if(a[i] == elem)
      trovato = true;
      else if (a[i] < elem)
           inizio = i+1;
           else fine = i;
    }
  if(trovato == true)
    printf("L'elemento e` in posizione %d\n", i);
  else printf("L'elemento non e` nell'array\n");
}
```

Esercizio 2.08 Merge di due array ordinati

Si scriva un programma che costruisce un vettore di interi a partire dalle informazioni contenute in altri due vettori. Il criterio di costruzione del vettore deve essere tale per cui gli elementi in esso contenuti siano ordinati in modo crescente. Si supponga che anche i due vettori di partenza contengano elementi ordinati in modo crescente, e che le dimensioni dei tre vettori possano essere diverse tra loro.

Soluzione

```
#include <stdio.h>
#define MDIM1   7
#define MDIM2   10
#define MDIM3   15

void main(){/* a e b sono gli array usati come input per il
 * merge. Si suppone che essi contengano valori
 * ordinati in modo crescente.
 * Si noti che gli array sono stati inizializzati al solo
 * scopo di concentrarsi esclusivamente sul merge*/
```

```
int a[MDIM1] = {2,5,6,7,11,12,18};
int b[MDIM2] = {0,1,3,8,9,15,17,19,22,23};

/* c e' l'array usato per contenere
 * il risultato dell'operazione */
int c[MDIM3];

  int i, j, k;

  i = 0;
  j = 0;
  k = 0;

  while (k<MDIM3 && i<MDIM1 && j<MDIM2)  {
     if(a[i]<b[j])
     {
       c[k] = a[i];
       i++;
     }
     else
     {
       c[k] = b[j];
       j++;
     }
     k++;
  }
  while(i<MDIM1 && k<MDIM3)
  {
     c[k] = a[i];
     i++;
     k++;
  }
  while(j<MDIM2 && k<MDIM3)
  {
     c[k] = b[j];
     j++;
     k++;
  }

printf("Primo array\n");
for(i=0;i<MDIM1;i++)
   printf("%d, ", a[i]);

printf("\nSecondo array\n");
for(i=0;i<MDIM2;i++)
   printf("%d, ", b[i]);

printf("\nArray risultante\n");
for(i=0;i<MDIM3;i++)
   printf("%d, ", c[i]);
}
```

Esercizio 2.09 Somma di vettori

Si sviluppi un programma che acquisisce da tastiera i valori di due vettori di 10 posizioni e calcola un nuovo vettore come risultato della somma tra i primi due.

3. Struct

Esercizio 3.01 Inserimento ordinato

Si scriva un programma che:

- definisca un tipo di dato Studente. Ogni studente è caratterizzato da un nome, un cognome e una matricola.
- acquisisca i dati di 10 studenti e, per ogni nuovo studente inserito, proceda ad un inserimento ordinato (lo scopo dell'esercizio è ordinare gli studenti durante il processo di inserimento degli stessi e non dopo averli inseriti tutti con un algoritmo di ordinamento).
- stampi i dati degli studenti.

Soluzione

```
#include <stdio.h>

#define MAX_STR      20
#define MAX_NUM_EL   4

typedef char stringa[MAX_STR];
typedef struct {
        int matricola;
        stringa nome;
        stringa cognome;
} studente;

void main()
{
        studente a[MAX_NUM_EL];
        studente tmp;
        int i, j, numEl;

        for (numEl=0;numEl<MAX_NUM_EL;numEl++)
        {
                /*acquisizione dei dati di ogni studente */
                printf("Inserisci le informazioni di uno studente\n");
                printf("nome: ");
                scanf("%s", tmp.nome);
                printf("cognome: ");
                scanf("%s", tmp.cognome);
                printf("matricola: ");
                scanf("%d", &tmp.matricola);
                printf("\nDati acquisiti.\n");


                /* se l'array e` vuoto si effettua l'inserimento direttamente */
                if (numEl == 0)
                        a[0] = tmp;
                else
                {

                        /* cerco la posizione di inserimento */
                        i = 0;
                        while(i<numEl && a[i].matricola<tmp.matricola)
                                i++;

                        /* creo lo spazio per inserire l'elemento */
                        j = numEl;
```

```
                        while (j>i)
                        {
                                a[j] = a[j-1];
                                j--;
                        }

                        /* inserisco l'elemento nella posizione corretta */
                        a[i] = tmp;
                }
        }

        /* stampo i numeri di matricola a scopo di test */
        for(i=0;i<MAX_NUM_EL;i++)
                printf("%d\n", a[i].matricola);
}
```

Esercizio 3.02 Rilievi altimetrici

Sviluppare un programma che svolge le seguenti operazioni:

- Acquisisce informazioni relative ad una serie di rilievi altimetrici, fino ad un massimo di 10 rilievi. Ogni rilievo è caratterizzato da una latitudine, una longitudine ed una altitudine.
- Terminata la fase di acquisizione, stampa sullo schermo le informazioni relative a tutti i rilievi per i quali il valore della longitudine è pari.

Soluzione

La soluzione proposta utilizza una struct con due campi per memorizzare le informazioni sui rilievi. Il primo campo è l'array in cui le informazioni vengono memorizzate. Il secondo campo ci consente di tenere traccia del numero di rilievi effettivamente contenuti nell'array.

```
#include <stdio.h>
#define MAX_RIL 10

typedef struct {
  int longit;
  int latid;
  int altid;
} rilievo;

typedef struct {
  rilievo a[MAX_RIL];
  int numRilievi;
}archivio;

void main()
{
  archivio arch;
  int decisione, i;

  arch.numRilievi=0;

  /* acquisizione dati */
  printf("Vuoi inserire un dato (se si digita 1, altrimenti 0: ?");
  scanf("%d", &decisione);
  while (decisione==1 && arch.numRilievi<MAX_RIL)
    {
      scanf("%d", &(arch.a[arch.numRilievi].longit));
      scanf("%d", &(arch.a[arch.numRilievi].latid));
      scanf("%d", &(arch.a[arch.numRilievi].altid));
      arch.numRilievi++;
```

```
      printf("Vuoi inserire un nuovo dato (se si digita 1, altrimenti
0: ?");
      scanf("%d", &decisione);
    }

  /* Individuazione e stampa degli elementi con valore di longitudine
pari */
  for(i=0;i<arch.numRilievi;i++)
    if(arch.a[i].longit%2==0)
      {
      printf("\n%d, %d, %d\n", arch.a[i].longit, arch.a[i].latid,
             arch.a[i].altid);
      }
}
```

Esercizio 3.03

Si vuole implementare un programma che calcola le aree di tre tipi di figure geometriche: rettangoli, triangoli e quadrati. Si supponga che i dati relativi alle figure geometriche vengano memorizzate in struct del tipo:

```
typedef struct {
      int tipo;
      int a;
      int b;
} figura;
```

dove il campo tipo assume valore 0 nel caso in cui la figura sia un rettangolo (in questo caso i valori memorizzati nei campi a e b corrispondono ai due lati), valore 1 nel caso in cui la figura sia un triangolo (in questo caso i valori memorizzati nei campi a e b sono la base e l'altezza), valore 2 nel caso in cui la figura sia un quadrato (in questo caso il valore in a è il lato del quadrato ed il valore presente in b viene trascurato).

Soluzione che presuppone l'uso di sottoprogrammi

La soluzione si basa sull'idea di modularizzare il programma in tre parti: il programma principale e due sottoprogrammi che si occupano rispettivamente della fare si lettura dei dati relativi alla figura geometrica ed al calcolo dell'area.

```
#include <stdio.h>

typedef struct {
               int tipo;
               int a;
               int b;
} figura;

figura leggiFigura();
int calcolaArea(figura fig);

void main()
{
  figura f;
  int a;
  /* acquisizione dei dati sulla figura */
  f = leggiFigura();
  /* calcolo dell'area */
  a = calcolaArea(f);
  /* stampa del risultato */
  printf("L'area della figura e` %d\n", a);
}

figura leggiFigura()
```

```
{
  figura fig;

  printf("Inserisci il tipo della figura: ");
  scanf("%d", &fig.tipo);
  printf("\n");
  switch(fig.tipo) {
  case 0:         /* rettangolo */
    printf("inserisci i valori dei due lati: ");
    scanf("%d", &fig.a);
    scanf("%d", &fig.b);
    break;
  case 1:         /* triangolo */
    printf("inserisci i valori di base e altezza: ");
    scanf("%d", &fig.a);
    scanf("%d", &fig.b);
    break;
  case 2:         /* quadrato */
    printf("inserisci il valore del lato: ");
    scanf("%d", &fig.a);
    break;
  default:
    printf("Tipo di figura non legale per il programma\n");
  }
  return fig;
}

int calcolaArea(figura fig)
{
  switch(fig.tipo) {
  case 0:         /* rettangolo */
    return fig.a*fig.b;
    break;
  case 1:         /* triangolo */
    return (fig.a*fig.b)/2;
    break;
  case 2:         /* quadrato */
    return (fig.a*fig.a);
    break;
  default:
    printf("Tipo di figura non legale per il programma\n");
  }
}
```

Si noti che, grazie alla modularizzazione introdotta, è possibile sfruttare i due sottoprogrammi anche in un contesto più complesso. Di seguito viene mostrato una nuova versione del programma principale che sfrutta i due sottoprogrammi per acquisire e calcolare l'area di più figure geometriche.

```
void main()
{
  figura ff[10];
  int a, num, i;
/* richiesta del numero di figure da considerare */
  num = 11;
  while(num > 10)
  {
    printf("Quante figure vuoi considerare (il massimo e` 10)?");
    scanf("%d", &num);
  }
  /* acquisizione dei dati sulle figure */
  for (i = 0; i < num; i++)
```

```
    ff[i] = leggiFigura();
  /* calcolo delle aree */
  for (i = 0; i < num; i++)
  {
    a = calcolaArea(ff[i]);
    printf("L'area della figura e` %d\n", a);
  }
}
```

Esercizio 3.04 Ricerca in un array di struct di tipo studente

Si sviluppi un programma che svolge le seguenti operazioni:

- Acquisisce le informazioni relative a cinque studenti e le memorizza in un array.
- Richiede all'utente di inserire il numero di matricola di uno studente di cui cercare le informazioni nell'array.
- Effettua la ricerca e stampa i dati che eventualmente verranno trovati nell'array.

Soluzione

```
#include <stdio.h>
#define MAX_STR      20
#define MAX_NUM_EL   5

typedef char stringa[MAX_STR];
typedef struct {
  int matricola;
  stringa nome;
  stringa cognome;
} studente;

typedef enum {falso, vero} boolean;

void main()
{
  studente st[MAX_NUM_EL];
  int i, j;
  int matr;
  boolean trovato;
  for (i=0;i<MAX_NUM_EL;i++)
    {
      /*acquisizione dei dati di ogni studente */
      printf("Inserisci le informazioni di uno studente\n");
      printf("nome: ");
      scanf("%s", st[i].nome);
      printf("cognome: ");
      scanf("%s", st[i].cognome);
      printf("matricola: ");
      scanf("%d", &st[i].matricola);
      printf("\nDati acquisiti.\n");
    }

  printf("inserisci la matricola dello studente: ");
  scanf("%d", &matr);

  trovato = falso;
  for (i=0; i<MAX_NUM_EL && trovato == falso; i++)
    if (matr == st[i].matricola)
      trovato = vero;
  if (trovato == vero)
    {
      printf("informazioni sullo studente: \n");
      printf("matricola: %d\n", st[i-1].matricola);
```

```
      printf("nome: %s\n", st[i-1].nome);
      printf("cognome: %s\n", st[i-1].cognome);
    }
  else printf("informazioni non presenti in archivio\n");
}
```

4. Funzioni e passaggio di parametri

Esercizio 4.01

Si consideri il seguente programma:

```
void scambia(int a, int b)
{
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
}
void main()
{
    int num1, num2;

    scanf("%d", &num1);
    scanf("%d", &num2);
    scambia(num1, num2);
    printf("num1 = %d, num2 = %d", num1, num2);
}
```

Quali sono i valori stampati dalla printf per le variabili num1 e num2 nell'ipotesi che i valori acquisiti da tastiera ed assegnati all'inizio del programma a queste due variabili siano rispettivamente 10 e 20?

Soluzione

Il valore di num1 sarà 10, il valore di num2 sarà 20. num1 e num2, infatti, sono state passate per valore al sottoprogramma scambia. Di conseguenza, le operazioni effettuate nel contesto di scambia sui due parametri formali a e b non hanno nessun impatto sulle variabili in questione. Se avessimo voluto che i valori delle due variabili num1 e num2 venissero effettivamente invertiti, avremmo dovuto modificare il programma nel modo seguente:

```
void scambia(int *a, int *b)
{
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}
void main()
{
    int num1, num2;

    scanf("%d", &num1);
    scanf("%d", &num2);
    scambia(&num1, &num2);
    printf("num1 = %d, num2 = %d", num1, num2);
}
```

Esercizio 4.02

Si implementi un sottoprogramma che riceve come parametro le informazioni relative ad un rettangolo, ne calcola l'area e restituisce tale valore al chiamante utilizzando il meccanismo del valore di ritorno. Si supponga che le informazioni relative ai rettangoli nel programma vengano rappresentate utilizzando il seguente tipo struct:

```
typedef struct {
    float lato1;
    float lato2;
} Rettangolo;
```

Soluzione

```
float areaRett(Rettangolo r)
{
    return (r.lato1 * r.lato2);
}
```

Esercizio 4.03

Si consideri il seguente programma:

```
#include <stdio.h>

void incrCondizionato(int *a)
{
     if (*a>10)
         *a = *a+1;
}

void main()
{
     int b, c;
     b = 10;
     incrCondizionato(&b);
     printf("Il valore di b e`: %d\n", b);
     c = 11;
     incrCondizionato(&c);
     printf("Il valore di c e`: %d\n", c);
}
```

Quali sono i valori di b e c che vengono stampati dalle due printf? Motivare brevemente la risposta.

Soluzione

Il valore stampato dalla prima printf è 10, mentre il valore stampato dalla seconda printf è 12. Il sottoprogramma incrCondizionato riceve gli indirizzi di b e di c, ma incrementa solo il valore di c perchè il valore contenuto in b non rispetta la condizione dell'if che regola l'incremento.

Esercizio 4.04

Si consideri il seguente frammento di programma:

```
#include <stdio.h>

void sottopr1()
{
   int a;
   if (a!=b)
      printf("e` diverso");
   else printf("e` uguale");
}

void main()
{
   int b;

   scanf("%d", &a);
   scanf("%d", &b);
   sottopr1();
}
```

È corretto l'uso della variabile `a` nel main?

È corretto l'uso della stessa variabile `a` nel sottoprogramma?
È corretto l'uso della variabile `b` nel sottoprogramma?

Soluzione

- L'uso di a nel main non è corretto perchè nel main e nell'ambiente globale non esiste una dichiarazione di variabile che corrisponde ad a. a infatti è stata dichiarata nel sottoprogramma sottopr1. Tale dichiarazione non è visibile nel main.
- L'uso di a nel sottoprogramma è corretto dal momento che nel sottoprogramma stesso esiste una dichiarazione di a.
- L'uso di b nel sottoprogramma non è corretto perchè non esiste nel sottoprogramma nè nell'ambiente globale una dichiarazione per b. La dichiarazione di b nel main non è visibile nel sottoprogramma.

Esercizio 4.05

Si considerino i seguenti sottoprogrammi:

```
float sottopr1(char a, float b)
{
   if (a=='p')
      return b/2.0;
   else return b;
}

void sottopr2(char a, float b, float *c)
{
   if (a=='p')
      *c = b/2.0;
   else *c = b;
}
```

In che cosa si differenziano? Quale è la sintassi corretta per chiamarli entrambi?

Soluzione

I due sottoprogrammi svolgono lo stesso compito ma si differenziano per il modo con cui il risultato del calcolo viene passato al programma chiamante. Nel primo caso si utilizza il meccanismo del valore di ritorno, mentre nel secondo caso si utilizza il passaggio di parametri per indirizzo. La sintassi che deve essere utilizzata per chiamare i due sottoprogrammi è la seguente:

```
float x, y, z;
.....
z = sottopr1(x, y);
sottopr2(x, y, &z);
```

Esercizio 4.06

Si individuino gli errori presenti nel seguente programma. Si correggano gli errori del programma lasciando invariata l'intestazione del sottoprogramma funz. Si indichino i valori stampati dall'istruzione printf del programma principale per effetto dell'esecuzione del programma corretto, nell'ipotesi che il valore letto dalla scanf sia 20.

```
#include <stdio.h>

void funz(int a, int *b)
{
    int c;
    scanf("%d\n", c);
    *b = c;
    b = &a;
}
void main()
{
   int e, f;
   f = 10;
```

```
   funz(f, e);
   printf("%d %d\n", f, e);
}
```

Soluzione

Gli errori sono i seguenti:

1. La chiamata di funz non è compatibile con la testata del sottoprogramma. Infatti, il sottoprogramma si aspetta di ricevere un puntatore come secondo parametro.
2. Nella scanf presente nel sottoprogramma manca il simbolo &.

La versione corretta del sottoprogramma è la seguente:

```
#include <stdio.h>

void funz(int a, int *b)
{
    int c;
    scanf("%d\n", &c);
    *b = c;
    b = &a;
}
void main()
{
   int e, f;
   f = 10;
   funz(f, &e);
   printf("%d %d\n", f, e);
}
```

Supponendo che il valore letto da tastiera sia 20, i valori stampati nel programma principali per le variabili f ed e sono rispettivamente 10 e 20. Si noti che l'ultima istruzione sottoprogramma funz (`b = &a`) non ha nessun impatto sul valore della variabile e. Tale istruzione, infatti, non opera sul contenuto della variabile puntata da b, che nel caso specifico è e. Invece, essa modifica il valore dell'indirizzo contenuto in b. Dopo l'esecuzione di questa operazione, infatti, b non punta più ad e, ma al parametro a.

Esercizio 4.07

Si implementi un **sottoprogramma** che riceve i seguenti parametri:

- Una struct contenente un array di interi `a` e la sua dimensione
- un valore intero `b`

Il sottoprogramma calcola il numero di occorrenze del valore intero `b` nell'array `a` e ritorna al chiamante il valore trovato. Per esempio, se l'array contiene i seguenti valori: {0, 2, 3, 0, 3, 5} ed il valore intero `b` passato al sottoprogramma è 3, il risultato che il sottoprogramma dovrà produrre e passare al chiamante è 2.

Soluzione

```
#include <stdio.h>

typedef struct
{
  int a[10];
  int num;
} archivio;

int conta(archivio arch, int b)
{
  int i, cont;

  cont = 0;
  for(i=0;i<arch.num;i++)
    if(arch.a[i] == b)
      cont++;
```

```
  return cont;
}
```

Esercizio 4.08

Data la seguente definizione di tipo:

```
typedef struct {
   int N;
   int num[10];
} numeri;
```

Cosa fa la seguente funzione?

```
int XXX(numeri a)
{
   int i;
   int cont;

   cont = 0;
   for(i = 0; i < a.N; i++)
   {
     if (a.num[i]%2 != 0)
     cont++;
   }
   return(cont);
}
```

Data la seguente situazione iniziale:

```
a.N = 8;

a.num = {12, 4, 0, 3, 5, 0, 3, 2, 0 , 2}
```

Quale valore restituisce la funzione?

Soluzione

La funzione conta il numero di valori dispari (non divisibili per due) che sono presenti nell'array incapsulato nella struct numeri, e restituisce tale numero al chiamante.
Nell'esempio riportato sopra, il valore restituito dalla funzione è 3.

Esercizio 4.09

Si definisca un sottoprogramma che effettua una ricerca per numero di matricola in un array che contiene informazioni relative agli studenti di un corso. Se l'elemento cercato si trova nell'array, il sottoprogramma restituisce al chiamante l'**indirizzo** della cella corrispondente.
Si supponga che l'array su cui si effettua la ricerca sia **ordinato in modo crescente**. Si definiscano le strutture dati necessarie per lo sviluppo del sottoprogramma.

Soluzione

```
#include <stdio.h>

#define MAX_STR      20
#define MAX_NUM_EL   10

typedef char stringa[MAX_STR];
typedef struct {
  int matricola;
  stringa nome;
  stringa cognome;
} studente;
```

```
typedef struct {
  studente stud[MAX_NUM_EL];
  int numSt;
} archivioSt;
```

```
studente * ricerca(archivioSt *a, int matr)
{
  int i;
  for (i=0; i<a->numSt && matr>a->stud[i].matricola; i++);
  if (matr == a->stud[i].matricola)
    return &a->stud[i];
  else return NULL;
}
```

Si noti che l'archivio deve essere passato come parametro. Si rappresentino graficamente gli ambienti di esecuzione del chiamante e del chiamato e si rifletta sulle motivazioni di questa scelta.

Esercizio 4.10

Si sviluppi un sottoprogramma che a partire dagli elementi contenuti in due array di interi ordinati costruisce un terzo array ordinato. Per esempio, se i due array di input contengono i seguenti valori: a1 = {3, 4, 5, 10, 12}, a2 = {1, 2, 6, 11, 18}, l'array risultante conterrà i seguenti valori: ris = {1, 2, 3, 4, 5, 6, 10, 11, 12, 18}.
Si supponga che il sottoprogramma operi su array definiti come variabili globali.

Soluzione

```
#include <stdio.h>
#define MDIM1   10
#define MDIM2   10
#define MDIM3   15
/* a e b sono gli array usati come input dalla funzione
 * di merge. Si suppone che essi contengano valori
 * ordinati in modo crescente.
 * Si noti che gli array sono stati inizializzati al solo
 * scopo di effettuare un test del sottoprogramma merge */

int a[MDIM1] = {2,5,6,7,11,12,18,20,21,26};
int b[MDIM2] = {0,1,3,8,9,15,17,19,22,23};

/* c e' l'array usato dalla funzione di merge per contenere
 * il risultato dell'operazione */
int c[MDIM3];

void merge()
{
  int i, j, k;

  i = 0;
  j = 0;
  k = 0;

  while (k<MDIM3 && i<MDIM1 && i<MDIM2)
  {
     if(a[i]<b[j])
     {
       c[k] = a[i];
       i++;
```

```
      }
      else
      {
        c[k] = b[j];
        j++;
      }
      k++;
  }
  while(i<MDIM1 && k<MDIM3)
  {
      c[k] = a[i];
      i++;
      k++;
  }
  while(j<MDIM2 && k<MDIM3)
  {
      c[k] = b[j];
      j++;
      k++;
  }
}

/* programma di test */
void main()
{
  int i;

  merge();
  printf("Primo array\n");
  for(i=0;i<MDIM1;i++)
    printf("%d, ", a[i]);

  printf("\nSecondo array\n");
  for(i=0;i<MDIM2;i++)
    printf("%d, ", b[i]);

  printf("\nArray risultante\n");
  for(i=0;i<MDIM3;i++)
    printf("%d, ", c[i]);
}
```

Esercizio 4.11 Archivio di date

Si definisca:

- Un tipo TipoArchivio con campi NumDate di tipo intero e SeqDate array di 100 elementi di tipo Data.
- Un tipo VettoreDiPuntatori come array di 100 elementi di tipo puntatore a elementi di tipo Data.

Si scriva un main che utilizza una variabile A1 di tipo TipoArchivio, una variabile VP di tipo VettoreDiPuntatori ed una variabile DataOdierna di tipo Data.
Scrivere la porzione di un main che, per ogni elemento della Sequenza contenuta nell'archivio A1, confronta l'elemento del vettore con la data odierna. Se l'elemento del vettore precede la data odierna il main assegna un puntatore del vettore VP all'elemento della sequenza appena analizzato.
Utilizzare la funzione Precede qui di seguito, che permette di comparare due date.
Eseguire l'assegnamento del vettore prima internamente al main e poi tramite procedura.

```
boolean Precede(Data d1, Data d2)
{
  if(d1.anno<d2.anno)
    return vero;
  else if(d1.anno==d2.anno)
    if(d1.mese<d2.mese)
```

```
      return vero;
  else if(d1.mese==d2.mese)
    if(d1.giorno<=d2.giorno)
      return vero;
    else return falso;
}
```

Soluzione

```
#include <stdio.h>

typedef struct {
  int giorno;
  int mese;
  int anno;
} Data;

typedef struct {
  Data SeqDate[100];
  int NumDate;
} TipoArchivio;

typedef Data* VettoreDiPuntatori[100];

typedef enum {falso, vero} boolean;

Data LeggiData();
void StampaData(Data d);
boolean Precede(Data d1, Data d2);

Data LeggiData()
{
  Data d;
  printf("Inserisci una nuova data.\nGiorno: ");

  scanf("%d", &d.giorno);
  printf("Mese: ");
  scanf("%d", &d.mese);
  printf("Anno: ");
  scanf("%d", &d.anno);
  return d;
}

void StampaData(Data d)
{
  printf("Giorno %d ", d.giorno);
  printf("Mese %d ", d.mese);
  printf("Anno %d\n", d.anno);
}

void main()
{
  TipoArchivio A1;
  VettoreDiPuntatori VP;
  Data DataOdierna;
  int i, j;
  A1.NumDate=0;
  for(i=0;i<3;i++)
    {
```

```
      A1.SeqDate[i]=LeggiData();
      A1.NumDate++;
    }

  printf("Data Odierna\n");
  DataOdierna = LeggiData();
  j=0;
  for(i=0;i<A1.NumDate;i++)
    if(Precede(A1.SeqDate[i], DataOdierna) == vero)
      {
      VP[j] = &A1.SeqDate[i];
      j++;
      }
  for (i=0;i<j;i++)
    StampaData(*(VP[i]));
}
```

Esercizio 4.12 Gestione dei rilievi altimetrici

Un programma manipola dati relativi a rilievi altimetrici. Per ogni rilievo, il programma deve mantenere le informazioni relative all'altezza misurata ed alle coordinate del punto in cui la misurazione è avvenuta (per semplicità si considerino coordinate cartesiane di un piano). Il programma gestisce al più 100 rilievi. Le operazioni che il programma consente di effettuare sono le seguenti:

- Acquisizione dei dati altimetrici da tastiera e memorizzazione dei dati in una opportuna struttura dati.
- Modifica di un dato altimetrico già acquisito. In particolare, il programma richiede all'utente le coordinate del punto di cui vuole modificare le coordinate, e consente all'utente di inserire una nuova misurazione altimetrica per queste coordinate. Tale misurazione va a sostituire quella precedentemente acquisita.
- Cancellazione di un rilievo

Nel programma vengono definite le seguenti strutture dati:
Una struct di tipo **`rilievo`** che descrive i dati relativi a ciascun rilievo.
Una struct di tipo **`archivioRilievi`** che raggruppa tutti i rilievi effettuati. Questa struct è definita nel modo seguente:

```
typedef struct {
   rilievo arch[100];
   int numRilievi;
} archivioRilievi;
```

`numRilievi` indica il numero di dati significativi presenti nell'archivio in ogni momento dell'esecuzione del programma.
Il programma si appoggia a tre sottoprogrammi chiamati **`inserisciNuovoRilievo`** e **`modificaRilievo`** e **`cancellaRilievo`** per effettuare le operazioni elencate sopra.

Si definiscano in C:

1. La struct `rilievo`.
2. Le testate dei tre sottoprogrammi `inserisciNuovoRilievo`, `modificaRilievo`, e `cancellaRilievo`.
3. L'implementazione del sottoprogramma `cancellaRilievo`.
4. Un main che svolge le seguenti operazioni (appoggiandosi ai due sottoprogrammi menzionati):

- acquisisce tre dati altimetrici;
- consente all'utente di modificare uno dei tre dati inseriti;
- consente all'utente di cancellare uno dei dati inseriti.

stampa il contenuto del campo `arch` della struct `archivioRilievi`.

Soluzione

```
#include <stdio.h>

typedef struct {
```

```
  float x;
  float y;
} punto;

typedef struct {
  float altezza;
  punto coordinate;
} rilievo;

typedef struct {
   rilievo arch[100];
   int numRilievi;
} archivioRilievi;

typedef enum {FALSE, TRUE} bool;

archivioRilievi ar;

void inserisciNuovoRilievo();
void modificaRilievo(punto p);
void stampaRilievi();
void cancellaRilievo(punto p);

void cancellaRilievo(punto p)
{
  int i;
  i=0;
  while(i<ar.numRilievi && (ar.arch[i].coordinate.x!=p.x
      || ar.arch[i].coordinate.y!=p.y))
    i++;
  if(i<ar.numRilievi)
    {
      ar.numRilievi=ar.numRilievi-1;
      while(i<(ar.numRilievi))
      {
        ar.arch[i]=ar.arch[i+1];
        i++;
      }
    }
}

void main()
{
  punto pnt;

  ar.numRilievi = 0;

  inserisciNuovoRilievo();
  inserisciNuovoRilievo();
  inserisciNuovoRilievo();
  stampaRilievi();

  pnt.x = 10.0;
  pnt.y = 20.0;
  modificaRilievo(pnt);

  stampaRilievi();
  printf("Inserisci le coordinate del punto da eliminare\n");
  scanf("%f", &pnt.x);
  scanf("%f", &pnt.y);
  cancellaRilievo(pnt);
```

```
  stampaRilievi();
}
```

5. Gestione delle stringhe

Esercizio 5.01

Si sviluppi un sottoprogramma che cerca una stringa in un elenco di stringhe. Il sottoprogramma restituisce un intero che rappresenta la posizione della stringa nell'elenco se questa esiste, e –1 in caso contrario.

Soluzione

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

typedef enum{false, true} boolean;
typedef char stringa[20];

int ricerca(stringa vocabolario[], int dim, stringa parola)
{
  int i;
  boolean trovato;

  trovato = false;
  i=0;
  while(i<dim && trovato == false)
    {
      if(strcmp(vocabolario[i], parola) == 0)
                trovato = true;
      else i++;
    }
  if(trovato == true)
    return i;
  else return -1;
}

void main()
{
  stringa voc[10];
  stringa p;
  int i;
  printf("Inserisci le parole del vocabolario\n");
  for(i=0;i<10;i++)
    scanf("%s", voc[i]);
  printf("Inserisci la parola da cercare\n");
  scanf("%s", p);
  printf("%d\n", ricerca(voc, 10, p));
}
```

Esercizio 5.02

Si scriva un sottoprogramma che riceve come parametro un array di stringhe e le stampa a schermo in ordine alfabetico.

Soluzione

Vengono presentate due soluzioni per il problema dato. Nella prima soluzione l'array di stringhe viene prima riordinato alfabeticamente e poi stampato. Nella seconda soluzione, la stampa in ordine alfabetico viene effettuata senza riordinare preventivamente l'array.

Prima soluzione

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

/* La funzione stampaInOrdine stampa le stringhe nel vettore passato
come parametro in ordine alfabetico.
Riceve due parametri:
- il numero delle stringhe presenti nel vettore
- il vettore delle stringhe */
void stampaInOrdine(int N, char stringhe[10][20])
{
   /* definizione delle variabili */
   int i;
   int cont;

   /* Per ordinare il vettore si e` utilizzato l'algoritmo
dell'Ordinamento
      Semplice.
      Tale algoritmo procede in questo modo:
      - confronta il primo elemento del vettore con tutti gli altri,
      se ne trova uno piu` piccolo si scambiano le posizioni e questo
      diventa il nuovo primo elemento. Al termine di questa prima
      "passata" il primo elemento e` nella corretta posizione.
      - confronta il secondo elemento del vettore con tutti gli
altri,
      se ne trova uno piu` piccolo si scambiano le posizioni e questo
      diventa il nuovo secondo elemento. Al termine di questa seconda
      "passata" il secondo elemento e` nella corretta posizione.
      - continua cosi fino a che non ha posizionato tutti gli
elementi
      in modo corretto.
      Da quanto detto si capisce che si deve procedere con due cicli
di
      for uno dentro l'altro, come descritto nel codice seguente */
   /* E' il ciclo piu` esterno, ad ogni passo l'elemento di posizione
      "cont" sara` in posizione corretta */
   for (cont=0;cont<N;cont++)
   {
        /* E' il ciclo piu` interno. Vengono confrontati l'elemento
in
           posizione "cont" e quello in posizione i-esima. Se
quest'ultimo
           e` piu` piccolo viene effettuato lo scambio di posizione
*/
        for (i=cont; i<N; i++)
        {
                int confronto = strcmp(stringhe[i],stringhe[cont]);
                if (confronto<0)
                {
                   /* Scambio fra gli elementi.
                      Viene utilizzata la variabile temporanea temp
*/
                     char temp [20];
                     strcpy(temp,stringhe[i]);
```

```
                        strcpy(stringhe[i],stringhe[cont]);
                        strcpy(stringhe[cont],temp);
                }
        }
   }
   /* Ciclo di stampa delle stringhe ordinate */
   for (i=0; i<N; i++)
   {
        printf("%s\n",stringhe[i]);
   }

}

/* Esempio di main */
void main()
{
    int i;
    char stringhe[10][20] = {"buon\0",
                        "anno\0",
                        "a\0",
                        "tutti\0",
                        "gli\0",
                        "studenti\0",
                        "del corso\0",
                        "di Fondamenti\0",
                        "di\0",
                        "informatica!\0"};
    stampaInOrdine(10,stringhe);
}
```

Seconda soluzione

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

/* La funzione stampaInOrdine stampa le stringhe nel vettore passato
   come parametro in ordine alfabetico.
   Riceve due parametri:
   - il numero delle stringhe presenti nel vettore
   - il vettore delle stringhe */
void stampaInOrdine(int N, char stringhe[10][20])
{
   /* definizione delle variabili */
   int i;
   int cont;
   /* questa variabile conterra` l'indice dell'elemento
      piu` piccolo del vettore */
   int minore;


   /* Il sottoprogramma stampa le stringhe in ordine alfabetico.
      A tal fine procede in questo modo:
      - cerca l'elemento piu` piccolo del vettore e ne memorizza
      l'indice nella variabile "minore";
      - stampa l'elemento trovato e lo sostituisce con la stringa
piu`
      grande possibile (quella con tutte "z").
      In questo modo alla prossima ricerca questa stringa non verra`
      piu` considerata perche` e` la piu` grande possibile.
      - ripete i passi precedenti per un numero di volte pari al
      numero delle stringhe totali.
```

```
      Da quanto detto si capisce che si deve procedere con due cicli
      di for uno dentro l'altro, come descritto nel codice seguente
*/
   /* Il ciclo piu` esterno ripete per N volte la ricerca
      dell'elemento piu` piccolo */
   for (cont=0;cont<N;cont++)
   {
     minore = 0;
     /* questo ciclo scorre tutto il vettore per trovare l'elemento
        piu` piccolo */
     for (i=0; i<N; i++)
       {
         /* definizione della variabile che verra` usata per il
            confronto */
         int confronto;

         /* E' il ciclo piu` interno. Vengono confrontati l'elemento
            in posizione "minore" e quello in posizione i-esima.
            Se quest'ultimo e` piu` piccolo, viene aggiornato il
            valore di minore  */
         confronto = strcmp(stringhe[i],stringhe[minore]);
         if (confronto<0)
           {
             minore = i;
           }
       }
     printf("%s\n",stringhe[minore]);
     strcpy(stringhe[minore],"zzzzzzzzzzzzzzzzzzz\0");
   }
}

/* Esempio di main */
void main()
{
  int i;
  char stringhe[10][20] = {"buon\0",
                           "anno\0",
                           "a\0",
                           "tutti\0",
                           "gli\0",
                           "studenti\0",
                           "del corso\0",
                           "di Fondamenti\0",
                           "di\0",
                           "informatica!\0"};
  stampaInOrdine(10,stringhe);
}
```

Esercizio 5.03 Ricerca di una sottostringa

Si scriva un programma che riceve in input una stringa s (source) e una seconda stringa t (target). Il programma deve ricercare la stringa t all'interno della stringa s.

Soluzione

```
#include <stdio.h>

char* strstr(char s[], char t[])
{
```

```
 int i,j,k;
 for(i=0; s[i]!='\0'; i++) {
  j = i;
  k = 0;
  while(t[k]!='\0' && s[j] == t[k])
    {
        j = j+1;
        k = k+1;
    }
  if (k>0 && t[k] == '\0')
    return s+i;
 }
 return NULL;
}

/* implementazione alternativa che ritorna un indice invece che un puntatore */
int strindex(char s[], char t[])
{
 int i,j,k;
 for(i=0; s[i]!='\0'; i++) {
  j = i;
  k = 0;
  while(t[k]!='\0' && s[j]==t[k])
    {
        j = j+1;
        k = k+1;
    }
  if (k>0 && t[k] == '\0')
    return i;
 }
 return -1;
}

void main()
{
 char orig[40];
 char str[10];
 char *p;
 int l;

 printf("Inserisci una stringa: ");
 scanf("%s", orig);
 printf("\nInserisci la stringa da cercare: ");
 scanf("%s", str);
 p = strstr(orig, str);
 if (p!=NULL)
  printf("\n%s\n", p);
 else printf("\nLa stringa %s non e` contenuta in %s\n",
        str, orig);
 l = strindex(orig, str);
 if(l!=-1)
  printf("\n%s\n", orig+l);
 else printf("\nLa stringa %s non e` contenuta in %s\n",
        str, orig);
}
```

Esercizio 5.04 Concatenazione di due stringhe

Si scriva un programma che, ricevute in ingresso due stringhe str1 e str2, ne esegua la concatenazione (str1str2) e ne stampi a video il risultato.

Soluzione

```
#include <stdio.h>


#define MAX_STR1 40
#define MAX_STR2 10

int strlen(char *);

char *strcat(char s1[], char s2[])
{
 int i,j;

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

void main()
{
 char str1[MAX_STR1];
 char str2[MAX_STR2];
 char str3[MAX_STR1];
 char *p;
 int l;

 printf("Inserisci una stringa con un numero di caratteri inferiore a\n");
 printf("%d: ", MAX_STR1-MAX_STR2);
 scanf("%s", str1);
 while(strlen(str1)>MAX_STR1-MAX_STR2)
   {
    printf("Inserisci una stringa con un numero di caratteri inferiore a\n");
    printf("%d: ", MAX_STR1-MAX_STR2);
    scanf("%s", str1);
   }
 printf("\nInserisci un'altra stringa con un numero di caratteri");
 printf("inferiore a %d: ", MAX_STR2);
 scanf("%s", str2);
 p = strcat(str1, str2);
 printf("\n%s\n", p);

}
```

6. Gestione dei file

Esercizio 6.01 Lettura di una sequenza di stringhe da file

Si scriva un programma che legge da un file di testo una sequenza di stringhe e le stampa sullo schermo

Soluzione

Nota: la presente soluzione assume che il file contenente la sequenza di stringhe esiste già. È possibile crearlo utilizzando un qualsiasi editor di testo. Il nome del file da creare è in.txt.

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

```
void main()
{
   FILE *in; char str[30];
   int letti;


   in=fopen("in.txt","r");

   if(in==NULL)
      printf("Impossibile aprire il file");
   else
   {
      letti=fscanf(in,"%s",str);

      /*
       * NOTA: il seguente ciclo termina se si verifica una delle
       * seguenti situazioni:
       *   - viene raggiunta la fine di in.txt
       *   - si verifica un errore di lettura da in.txt
       */
      while(letti>0)
      {
         printf("%s",str);
         letti=fscanf(in,"%s",str);
      }
      /*
       * si verifica se l'uscita dal ciclo sia avvenuta per la
       * terminazione del file in.txt oppure per il verificarsi di un
errore
       */
      if(ferror(in)==1)
         printf("Errore di i/o");
      if(fclose(in)==EOF)
         printf("Errore di chiusura file");
   }
}
```

Esercizio 6.02 Scrittura di una sequenza di interi su file

Scrivere un programma che acquisisce una sequenza di interi da standard input e la salva su un file in modalità testo

Soluzione

```
#include <stdio.h>

void main()
{
   FILE *p; int val;
   int scritti;


   p=fopen("outInt.txt","w");

   if(p==NULL)
      printf("Impossibile aprire il file");
   else
   {
     printf("Inserisci una sequenza di interi terminata dal numero
999\n");
     scanf("%d", &val);
     while(val!=999)
       {
```

```
                fprintf(p,"%d\n",val);
                scanf("%d", &val);
        }
        if(fclose(p)==EOF)
           printf("Errore di chiusura file");
    }
}
```

Esercizio 6.03 Copia di un file di testo

Si scriva un programma che esegue la copia di un file di testo. Il programma legge carattere per carattere il contenuto del file "in.txt" e lo ricopia nel file "out.txt".

Soluzione

```
#include <stdio.h>

void main()
{
  FILE *in; char car;
  FILE *out;
  int letti,scritti;

  scritti=1;

  in=fopen("in.txt","r");
  out=fopen("out.txt","w");

  if(in==NULL || out==NULL)
    printf("Impossibile aprire un file");
  else
  {
    letti=fscanf(in,"%c",&car);

    /*
     * NOTA: il seguente ciclo termina se si verifica una delle
     * seguenti situazioni:
     *   - viene raggiunta la fine di in.txt
     *   - si verifica un errore di lettura da in.txt
     *   - si verifica un errore di scrittura in out.txt
     */
    while(letti>0 && scritti>0)
    {
      scritti=fprintf(out,"%c",car);
      letti=fscanf(in,"%c",&car);
    }
    /*
     * si verifica se l'uscita dal ciclo sia avvenuta per la
     * terminazione del file in.txt oppure per il verificarsi di un errore
     */
    if(ferror(in)==1 || ferror(out)==1)
      printf("Errore di i/o");
    if(fclose(in)==EOF || fclose(out)==EOF)
      printf("Errore di chiusura file");
  }
}
```

Esercizio 6.04 Ordinamento di un archivio di interi

Scrivete un programma che legge da un file di testo una sequenza di interi, la ordina in modo crescente e salva su file la sequenza risultate sovrascrivendo il vecchio contenuto.

Soluzione

Nota: la presente soluzione assume che il file contenente la sequenza di interi esiste già. È possibile crearlo utilizzando un qualsiasi editor di testo. Il nome del file da creare è arch.txt. Estendete la soluzione in modo da consentire all'utente di indicare il nome del file contenete la sequenza di interi.

```
#include <stdio.h>
#define MAX_NUM  10

void ordinamento(int a[], int dim);

void main()
{
   FILE *arch;
   int tmp[10];
   int letti, scritti;
   int i, numElem;

   arch=fopen("arch.txt","r");

   if(arch==NULL)
      printf("Impossibile aprire il file");
   else
   {
     letti = 1;
     i=0;
     while(letti>0 && i<MAX_NUM)
      {
                letti=fscanf(arch,"%d",&tmp[i]);
                i++;
      }
     /* numElem viene usata per tenere traccia del numero di elementi
                significativi contenuti in tmp */
     numElem = i;
     /* ordino l'array */
     ordinamento(tmp, numElem);
     /* adesso tutto e` pronto per riscrivere gli elementi nel file
                attenzione pero`! Il file deve essere riaperto nella
modalita`
                corretta */
     if(fclose(arch)==EOF)
         printf("Errore di chiusura file");
     arch=fopen("arch.txt","w");
     if(arch==NULL)
       printf("Impossibile aprire il file");
     else
       {
                scritti = 1;
                i=0;
                while(scritti>0 && i<numElem)
                  {
                    scritti=fprintf(arch,"%d ",tmp[i]);
                    i++;
                  }
                if(ferror(arch)==1)
                  printf("Errore di i/o");
                if(fclose(arch)==EOF)
                  printf("Errore di chiusura file");
```

```
        }
    }
}


void ordinamento(int a[], int dim)
{
  int cont, i, tmp;

  for (cont=0;cont<dim-1;cont++)
    {
      /* E' il ciclo piu` interno. Vengono confrontati l'elemento in
                posizione i-esima e quello in posizione i+1-esima.
Se quest'ultimo
                e` piu` piccolo viene effettuato lo scambio di
posizione */
      for (i=0; i<dim-1; i++)
               {
                 if (a[i]>a[i+1])
                   {
                     /* Scambio fra gli elementi.
                          Viene utilizzata la variabile temporanea
temp */
                     tmp = a[i];
                     a[i] = a[i+1];
                     a[i+1] = tmp;
                   }
               }
    }
}
```

Esercizio 6.05

Si implementino due programmi. Il primo si occupa di creare un file in modalita` testo e di inserire nel file valori interi acquisiti dallo standard input, fino ad un massimo di 20. Il secondo legge i dati dal file e ne calcola la media stampandola a video ed "appendendola" ad un altro file di testo aperto in modalita` "append".
Dopo un certo numero di esecuzioni del primo e del secondo programma in sequenza, si verifichi che il file contenente le medie contenga un numero di valori pari al numero di esecuzioni della coppia di programmi (il file puo` essere visualizzato con l'editor del compilatore C della Borland oppure utilizzando qualsiasi altro editor).

Soluzione

La soluzione si compone di due sottoprogrammi distinti. Il primo inserisce i voti in un file, il secondo legge questi voti e calcola la media. Il primo programma e` implementato in due versioni. La prima versione aggiunge ad ogni esecuzione nuovi valori al file dei voti. La seconda versione ad ogni esecuzione distrugge il contenuto del file dei voti prima di inserire nuovi valori.

Programma per la scrittura dei voti sul file - prima versione

```
#include <stdio.h>

#define NOME_FILE_VOTI "voti.txt"
#define MAX_VOTI 20

/*
* Programma che permette all'utente di inserire
* all'interno di un file di testo un massimo di 20 voti.
* Ad ogni esecuzione il programma verifica il numero
* di voti gia' presenti nel file e permette all'utente
```

```
* di aggiungerne di nuovi in numero tale da non superare la soglia
* di 20
*/


void main()
{
FILE *file;

int votoEsame; /* per memorizzare un singolo voto da
                  leggere o scrivere */
int numeroEsami; /* per contare i voti gia' inseriti */
int nElementi; /* per memorizzare il numero
        di elementi letti o scritti su file
        ad ogni operazione di IO */

/***** conteggio dei voti gia' inseriti *****/
numeroEsami=0;
file=fopen(NOME_FILE_VOTI, "r");
if(file==NULL)
printf("Impossibile aprire il file: nessun voto inserito");
else
{
printf("Voti gia' inseriti: \n");
    nElementi=fscanf(file,"%d",&votoEsame);
while(nElementi>0)
{
            numeroEsami++;
            printf("%d ",votoEsame);
            nElementi=fscanf(file,"%d",&votoEsame);
}
    if(fclose(file)!=0 || ferror(file)==1)
    {
            printf("Errore di IO: impossibile continuare");
            return; /* questa istruzione provoca la
                    terminazione del programma */
    }
}

/***** inserimento nuovi voti *****/
file=fopen(NOME_FILE_VOTI,"a");
if(file==NULL)
printf("Impossibile aprire il file");
else
{
printf("\nInserimento nuovi voti. ");
printf("Immettere un numero negativo per terminare\n");
nElementi=1;
votoEsame=0;
numeroEsami++;
while(numeroEsami<=MAX_VOTI && votoEsame>=0 && nElementi>0)
{
    printf("%d: ",numeroEsami);
    scanf("%d",&votoEsame);
    if(votoEsame>=0)
    nElementi=fprintf(file,"%d\n",votoEsame);
    numeroEsami++;
}
if(fclose(file)!=0 || ferror(file)==1)
printf("Errore di IO");
}
```

```
}
```

Programma per la scrittura dei voti sul file - seconda versione

```
#include <stdio.h>

#define NOME_FILE_VOTI "voti.txt"
#define MAX_VOTI 20

/*
* Programma che permette all'utente di inserire
* all'interno di un file di testo un massimo di 20 voti.
* Ad ogni esecuzione il programma cancella tutti i voti
* eventualmente gia' presenti nel file (cioe' viene cancellato
* il risultato delle esecuzioni precedenti).
*/


void main()
{
FILE *file;

int votoEsame; /* per memorizzare un singolo voto da
        leggere o scrivere */
int numeroEsami; /* per contare i voti gia' inseriti */
int nElementi; /* per memorizzare il numero
        di elementi letti o scritti su file
        ad ogni operazione di IO */

file=fopen(NOME_FILE_VOTI,"w");
if(file==NULL)
printf("Impossibile aprire il file");
else
{
printf("\nInserimento nuovi voti.");
printf("Immettere un numero negativo per terminare\n");
nElementi=1;
votoEsame=0;
numeroEsami=1;
while(numeroEsami<=MAX_VOTI && votoEsame>=0 && nElementi>0)
{
    printf("%d: ",numeroEsami);
    scanf("%d",&votoEsame);
    if(votoEsame>=0)
    nElementi=fprintf(file,"%d\n",votoEsame);
    numeroEsami++;
}
if(fclose(file)!=0 || ferror(file)==1)
printf("Errore di IO");
}
}
```

Programma per il calcolo della media

```
#include <stdio.h>


#define NOME_FILE_IN "voti.txt"
#define NOME_FILE_OUT "media.txt"
```

```
/*
* Programma che legge una sequenza di voti da un file di testo
* e ne calcola la media. La media calcolata viene inserita alla fine
di
* un secondo file di testo.
*/

void main()
{
FILE *fileIn, *fileOut;
float media; /* per memorizzare la media */
int votoEsame; /* per memorizzare un voto letto da file */
int nEl; /* per le operazioni di lettura/scrittura*/
int totale; /* memorizza la somma di tutti i voti letti */
int numeroEsami; /* per il conteggio del numero di voti presenti nel
file*/

numeroEsami=0;
totale=0;

/***** lettura dei voti, calcolo del totale e del loro numero *****/
fileIn=fopen(NOME_FILE_IN, "r");
if(fileIn==NULL)
printf("Impossibile aprire il file di input");
else
{
nEl=fscanf(fileIn,"%d",&votoEsame);
while(nEl>0 && numeroEsami < 20)
{
    totale=totale+votoEsame;
    numeroEsami++;
    printf("%d ",votoEsame);
    nEl=fscanf(fileIn,"%d",&votoEsame);
}
if(ferror(fileIn)==1)
printf("Errore in lettura");
else if(numeroEsami!=0)
{
    /***** calcolo media e scrittura su file *****/
    media=(float)totale/(float)numeroEsami;
    /* andrebbe bene anche:
                1. (float) totale/numeroEsami;
                2. totale/ (float) numeroEsami;
       in ambedue i casi elencati, viene eseguita l'operazione di
       divisione tra float.

       Invece l'operazione,
                3. (float) (totale/numeroEsami);
        non sarebbe corretta perchè prima verrebbe eseguita la
        divisione fra interi (dunque, con conseguente troncamento) e
        il risultato verrebbe trasformato in float.
    */

    printf("\nMEDIA: %f\n",media);

    fileOut=fopen(NOME_FILE_OUT,"a");
    if(fileOut==NULL)
    printf("Impossibile aprire il file di output");
    else
    {
    fprintf(fileOut,"MEDIA: %f\n",media);
```

```
    if(ferror(fileOut)==1)
    printf("Errore in scrittura");
    if(fclose(fileOut)!=0)
    printf("Errore durante la chiusura di un file");
    }
}
if(fclose(fileIn)!=0)
printf("Errore durante la chiusura di un file");
}
}
```

Esercizio 6.06 Modifica di un dato presente in un file

Si scriva un sottoprogramma che modifica i dati relativi ad un rilievo altimetrico memorizzato su file. Il sottoprogramma individua il rilievo da modificare e interagisce con l'utente per ottenere i nuovi dati da inserire nel rilievo.

Soluzione

NB: per verificare il funzionamento del sottoprogramma richiesto, è stato costruito un main di prova ed altri sottoprogrammi di servizio.

```
#include <stdio.h>
#include <stdlib.h>  /* solo per invocare la funzione exit() */
typedef struct {
  int x;
  int y;
} punto;

typedef struct {
  int altezza;
  punto coord;
} rilievo;

typedef char string[20];
typedef enum {FALSO, VERO} bool;

void inserisciNuovoRilievo(rilievo r, string arch);
void modificaRilievo(punto p, string arch);
void stampaRilievi(string arch);

void main()
{
  string nfile = "arch.txt";
  rilievo r;
  int scelta;
  scelta=0;
  while(scelta!=3)
    {
      printf("*******************************************\n");
      printf("************ MENU PRINCIPALE **************\n");
      printf("*******************************************\n\n");

      printf("\t1) inserisci\n");
      printf("\t2) stampa\n");
      printf("\t3) ricerca\n");
      printf("\t4) modifica\n");
      printf("\t5) esci\n\n");
      printf("Cosa vuoi fare? ");

      scanf("%d",&scelta);
```

```
      switch(scelta)
                {
                case 1:
                  printf("coordinate punto: \n");
                  scanf("%d", &r.coord.x);
                  scanf("%d", &r.coord.y);
                  printf("Altezza: \n");
                  scanf("%d", &r.altezza);
                  inserisciNuovoRilievo(r, nfile);
                  break;
                case 2:
                  stampaRilievi(nfile);
                  break;
                case 3:
                  printf("Non implem.\n");
                  break;
                case 4:
                  printf("coordinate punto: \n");
                  scanf("%d", &r.coord.x);
                  scanf("%d", &r.coord.y);
                  modificaRilievo(r.coord, nfile);
                  break;
                case 5:
                  exit(0);
                  break;
                default:
                  printf("digita un'altra lettera\n");
                }
    }
}


void inserisciNuovoRilievo(rilievo ril, string arch)
{
  FILE *f;

  f=fopen(arch,"ab");
  if(f==NULL)
    printf("Impossibile aprire il file");
  else
    {
      if (fwrite(&ril, sizeof(rilievo), 1, f)!=1)
                printf("errore nella scrittura su file\n");
      fclose(f);
    }
}


void stampaRilievi(string arch)
{
  rilievo ril;
  FILE *f;

  f=fopen(arch,"rb");
  if(f==NULL)
    {
      printf("Impossibile aprire il file");
    }
  else
    {
      while(fread(&ril, sizeof(rilievo), 1, f)!=0)
```

```
                printf("Altezza alle coordinate (%d, %d): %d\n",
                       ril.coord.x, ril.coord.y, ril.altezza);
      fclose(f);
    }
}

void modificaRilievo(punto p, string arch)
{
  rilievo ril;
  FILE *f;
  bool trovato;
  f=fopen(arch,"rb+");
  if(f==NULL)
    {
      printf("Impossibile aprire il file");
    }
  else
    {
      trovato = FALSO;
      while(!trovato && fread(&ril, sizeof(rilievo), 1, f)!=0)
                {
                  if(ril.coord.x == p.x && ril.coord.y == p.y)
                    trovato = VERO;
                }
      if(trovato)
                {
                  printf("Altezza = %d\n", ril.altezza);
                  printf("Inserisci una nuova altezza\n");
                  scanf("%d", &ril.altezza);
                  fseek(f, -sizeof(rilievo), SEEK_CUR);
                  if (fwrite(&ril, sizeof(rilievo), 1, f)!=1)
                    printf("errore nella scrittura su file\n");
                }
      else printf("Elemento non trovato\n");
      fclose(f);
    }
}
```

7. Programmi completi

Esercizio 7.01 Numeri complessi

I numeri complessi non hanno in C un corrispondente tipo predefinito. Essi pero` possono essere rappresentati mediante la seguente struct:

```
typedef struct {
  double Re;
  double Imm;
} Complesso;
```

Si implementino i sottoprogrammi che effettuano le operazioni di somma e prodotto tra numeri complessi. Si assuma che i due numeri complessi su cui operare vengano sempre passati come parametro ai sottoprogrammi da implementare e che il risultato dell'operazione venga ritornato mediante il meccanismo del valore di ritorno.

Soluzione

```
#include <stdio.h>

typedef struct {
  double Re;
  double Imm;
```

```
} Complesso;

/* dichiarazione delle funzioni */
Complesso somma (Complesso c1, Complesso c2);
Complesso prodotto (Complesso c1, Complesso c2);

/* programma di prova per le funzioni */
void main()
{
  Complesso n1, n2, n3;
  n1.Re = 5;
  n1.Imm = 5;
  n2.Re = 10;
  n2.Imm = 16;

  n3 = somma(n1, n2);
  printf("n1 + n2 = Re %f Imm %f\n", n3.Re, n3.Imm);
  n3 = prodotto(n1, n2);
  printf("n1 * n2 = Re %f Imm %f\n", n3.Re, n3.Imm);
}

Complesso somma (Complesso c1, Complesso c2)
{
  Complesso ris;
  ris.Re = c1.Re + c2.Re;
  ris.Imm = c1.Imm + c2.Imm;
  return ris;
}

Complesso prodotto (Complesso c1, Complesso c2)
{
  Complesso ris;
  ris.Re = c1.Re * c2.Re - c1.Imm * c2 .Imm;
  ris.Imm = c1.Imm * c2.Re + c2.Imm * c1.Re;
  return ris;
}
```

Esercizio 7.02 Le matrici

Si implementi un programma che consenta all'utente di effettuare operazioni di somma e prodotto tra due matrici. Si utilizzino i sottoprogrammi per modularizzare la struttura del programma.

Soluzione

```
/* Il seguente programma implementa le funzioni per la lettura, la
stampa a video, la somma ed il prodotto di matrici 2x2. Non e` stato
implementato il sottoprogramma per il calcolo dell'inversa di una
matrice */

#include <stdio.h>

/* legge 4 numeri da standard input e li inserisce nella
matrice M passata come parametro */
void leggiMatrice(int M[2][2])
{ int i, j;

  for(i=0;i<2;i++)
     for(j=0;j<2;j++)
        scanf("%d",&M[i][j]);
}

/* effettua la somma tra le matrici Matr1 e Matr2 e
```

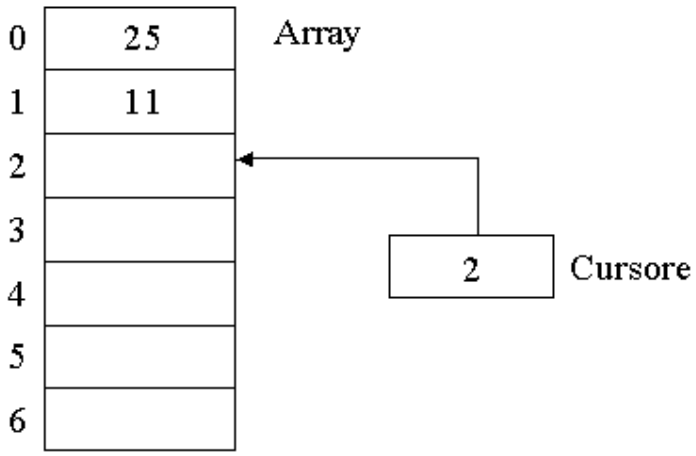
```
memorizza il risultato in Matr3 */
void somma(int Matr1[2][2], int Matr2[2][2], int Matr3[2][2])
{ int i, j;

  for(i=0;i<2;i++)
     for(j=0;j<2;j++)
        Matr3[i][j]=Matr1[i][j]+Matr2[i][j];
}

/* stampa i valori contenuti nella matrice passata come
parametro */
void stampaMatrice(int M[2][2])
{
  int i, j;
  for(i=0;i<2;i++){
     for(j=0;j<2;j++)
        printf("%d\t",M[i][j]);
     printf("\n");
  }
  printf("\n\n");
}


/* calcola il prodotto tra matrici secondo la formula:
Matr3[i][j] = sommatoria(Matr1[i][k] * Matr2[k][j]). Dove Matr1 e
Matr2 sono le matrici in input e Matr3 viene utilizzata
per restituire il risultato dell'operazione.
Si noti la necessita` di introdurre
tre cicli for innestati. I due cicli piu` esterni consentono di
accedere agli elementi di Matr3. Il ciclo piu` interno viene usato
per accedere agli elementi appartenenti ad una specifica riga di
Matr1 ed ad una specifica colonna di Matr2 */
void prodotto(int Matr1[2][2], int Matr2[2][2], int Matr3[2][2])
{
  int i, j, k;
  for (i = 0; i < 2; i++)
    {
      for (j = 0; j < 2; j++)
        {
          Matr3[i][j] = 0;
          for (k = 0; k < 2; k++)
            {
              Matr3[i][j] = Matr3[i][j] + (Matr1[i][k] *
Matr2[k][j]);
            }
        }
    }
}


/* questo main serve per testare i sottoprogrammi implementati
puo` essere modificato per offrire all'utente un'interfaccia piu`
amichevole */
void main()
{
  int M1[2][2];
  int M2[2][2];
  int M3[2][2];
  printf("Inserisci i 4 valori della prima matrice\n");
  leggiMatrice(M1);
  stampaMatrice(M1);
```

```
  printf("Inserisci i 4 valori della seconda matrice\n");
  leggiMatrice(M2);
  stampaMatrice(M2);
  somma(M1, M2, M3);
  printf("\nSomma di M1 e M2:\n");
  stampaMatrice(M3);
  prodotto(M1, M2, M3);
  printf("\nProdotto di M1 e M2:\n");
  stampaMatrice(M3);
}
```

Esercizio 7.03 La pila

Una pila e` un contenitore di dati che e` definito da due operazioni tradizionalmente chiamate push e pop. L'operazione di push inserisce un nuovo elemento nella pila in prima posizione rispetto a tutti gli altri elementi gia` contenuti nella pila stessa. L'operazione di pop preleva dalla pila l'elemento che si trova in testa alla pila stessa. Per avere una rappresentazione visiva di questa struttura dati immaginate una pila di piatti ed il modo con cui utilizzate questa pila (tipicamente non cercate ne` di prendere ne` di inserire un piatto in mezzo alla pila).

La pila nella programmazione viene di solito implementata mediante l'utilizzo di un array e di un intero che fa da cursore, tenendo traccia della prima posizione libera in testa alla pila. A questo proposito, si veda la seguente figura:
Si definisca la struttura dati nel caso in cui la pila contenga degli interi e si implementino i sottoprogrammi push e pop.



Soluzione

```
#include <stdio.h>

#define DIM 10 /* dimensione massima della pila */

/* definizione della struttura dati della pila. In questo
caso l'array che serve per memorizzare gli elementi
della pila ed il cursore sono incapsulati in una struct.
Questa soluzione rende piu` evidente il collegamento logico
esistente tra queste due parti */

typedef struct {
  int array[DIM];
  int cursore;
} pila;

/* Riceve un intero come parametro e lo inserisce nella pila. Il
parametro p e` il riferimento alla pila in cui vengono inseriti
```

```
gli elementi*/
void push(int elem, pila *p)
{
  /* si noti l'uso del simbolo -> per accedere agli elementi
     della struct puntata dal puntatore p. Questa e` una
     scorciatoia che ci permette di evitare l'uso della sintassi:
     (*p).cursore che risulta meno leggibile. E` sempre possibile
     utilizzare il simbolo -> per accerede agli elementi di una
     struct puntata da un puntatore */
   if (p->cursore < DIM)
     { /* e` possibile inserire un nuovo elemento nella pila */
       p->array[p->cursore] = elem;
       p->cursore = p->cursore+1; /* il cursore indica sempre
                                     la prima posizione
                                     libera in testa alla pila */
     }
   else printf("La pila e' piena\n");
}

/* Restituisce un intero che e` l'elemento in testa alla pila.
Il parametro p e` un riferimento alla pila */
int pop(pila *p)
{
  if (p->cursore > 0)
    {
      p->cursore = p->cursore-1;   /* il cursore viene decrementato
                                      in modo tale da liberare la
                                      posizione in testa alla pila */
      return p->array[p->cursore];
    }
  else return -1; /* la pila e` vuota. NB: si assume che i valori
                     inseriti
                     nella pila siano sempre positivi */
}

/* programma principale di prova */
void main()
{
  pila P; /* dichiarazione della pila */
  int scelta, numero; /* variabili utilizzate per la gestione dei
menu */

  P.cursore = 0;
  scelta=0;

  while(scelta!=3)
    {
      printf("*********************************************\n");
      printf("************ MENU PRINCIPALE ****************\n");
      printf("*********************************************\n\n");

      printf("\t1) Push\n");
      printf("\t2) Pop\n");
      printf("\t3) Esci\n\n");
      printf("Cosa vuoi fare? ");

      scanf("%d",&scelta);

      switch(scelta)
      {
         case 1:
```

```
            printf("digita l'intero da inserire nella pila\n");
            scanf("%d", &numero);
            if (numero > 0) push(numero, &P);
            break;
         case 2:
            numero = pop(&P);
            if (numero>0)
              printf("il numero prelevato dalla pila e' :%d\n",
numero);
            else printf("la pila e' vuota\n");
            break;
         case 3:
            printf("Programma terminato\n");
            break;
         default:
            printf("Devi inserire un numero compreso tra 1 e 3!");
            break;
      }
      printf("\n\n");
   }

}
```

Esercizio 7.04 La coda

Una coda è una struttura dati che ospita elementi secondo la politica FIFO (First In First Out). Questa politica prevede che l'ordine di prelievo degli elementi dalla coda concida con l'ordine con cui gli elementi sono inseriti nella coda. Una delle strutture dati che nella programmazione può essere usata per gestire una coda è l'array. Le operazioni che caratterizzano una coda sono l'inserimento e l'estrazione di elementi.
Si definisca una coda che contiene le informazioni relative alle pratiche da evadere nell'ufficio catastale di un comune di piccole dimensioni. Si implementino i sottoprogrammi che operano sulla coda. Infine, si sfruttino questi sottoprogrammi per costruire il programma che consente agli addetti del comune di aggiornare la coda delle pratiche.

Soluzione (parziale)

La seguente soluzione implementa una coda di interi. Estendere tale soluzione per risolvere il problema proposto nell'esercizio.
Nella soluzione dell'esecizio sarebbe utile avere a disposizione un'ulteriore operazione che consenta di capire se la coda e` piena oppure no. Provare a definire tale operazione mediante un sottoprogramma.

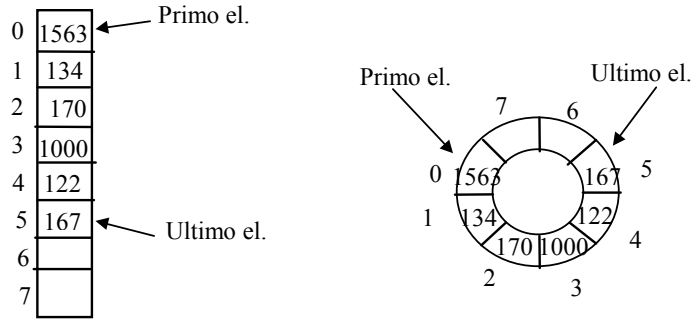
```
#include <stdio.h>

#define MAX_NUM_EL   5

typedef struct {
  int el[MAX_NUM_EL];
  int primoEl;
  int ultimoEl;
} coda;

/* inserisce nella posizione ultimoEl della coda l'elemento il
   cui valore si trova in elem. Se la coda e` piena, ritorna il
   valore zero che rappresenta un codice di errore  */
int inserisci(int elem, coda *c)
{
  if(c->ultimoEl < MAX_NUM_EL)
    {
      c->el[c->ultimoEl] = elem;
      c->ultimoEl = c->ultimoEl + 1;
```

```
      return 1;   /* l'inserimento e` stato effettuato con successo
*/
    }
  else return 0;
}

/* estrae l'elemento che si trova in prima posizione (primoEl)
   nella coda. Se la coda e` vuota, ritorna il valore 0 */
int estrai(coda *c)
{
  int elem;
  if(c->primoEl!=c->ultimoEl)
    {
      /* c'e` almeno un elemento nella coda */
      elem = c->el[c->primoEl];
      c->primoEl = c->primoEl+1;
    }
  else elem = 0;  /* si noti che se 0 rappresenta un codice di
                             errore, tale valore non deve mai
apparire nella
                             coda come valore significativo */
  return(elem);
}

/* assegna un valore iniziale ai campi primoEl e ultimoEl della
   coda. Questo sottoprogramma deve essere sempre richiamato
   per preparare la coda ad essere utilizzata in modo corretto. */
void inizializzaCoda(coda *c)
{
  c->primoEl = 0;
  c->ultimoEl = 0;
}

/* programma principale di prova */
void main()
{
  coda C; /* dichiarazione della coda */
  int scelta, numero; /* variabili utilizzate per la gestione dei
menu */
  int ris; /* variabile utilizzata per memorizzare il risultato delle
                   chiamate alle due funzioni di inserimento ed
estrazione */

  /* prepara la coda ad essere utilizzata */
  inizializzaCoda(&C);

  scelta=0;

  while(scelta!=3)
    {
      printf("*********************************************\n");
      printf("************ MENU PRINCIPALE ***************\n");
      printf("*********************************************\n\n");

      printf("\t1) Inserisci\n");
      printf("\t2) Estrai\n");
      printf("\t3) Esci\n\n");
      printf("Cosa vuoi fare? ");

      scanf("%d",&scelta);
```

```
      switch(scelta)
      {
         case 1:
            printf("digita l'intero da inserire nella coda\n");
            scanf("%d", &numero);
            if (numero != 0)
                     ris= inserisci(numero, &C);
                   if(ris == 0)
                     printf("Errore! La coda e` piena\n");
            break;
         case 2:
            numero = estrai(&C);
            if (numero!=0)
              printf("il numero prelevato dalla coda e' :%d\n",
numero);
            else printf("la coda e' vuota\n");
            break;
         case 3:
            printf("Programma terminato\n");
            break;
         default:
            printf("Devi inserire un numero compreso tra 1 e 3!");
            break;
      }
      printf("\n\n");
   }

}
```

Esercizio 7.05 La coda circolare

Si consideri la coda mostrata nella parte sinistra della seguente figura:



Un potenziale problema di questa coda è dato dal fatto che ci sono situazioni in cui nella coda non è possibile inserire elementi, anche se comunque ci sono posizioni libere. Si consideri per esempio la situazione in cui, a partire dalla situazione mostrata nella figura, si estraggono i primi cinque elementi dalla coda. In questa situazione, ci sono sette posizioni libere nella coda, ma solo due di esse (quelle numero 6 e 7) sono utilizzabili per l'inserimento di nuovi elementi. La politica di gestione della coda, infatti, impone che debba essere sempre mantenuto l'ordine di inserimento tra elementi, e, di conseguenza, non è possibile inserire elementi al di sopra dell'elemento in posizione 5.
Per risolvere tale problema è possibile utilizzare una *coda circolare* (si veda il disegno a destra in figura). Se un elemento viene estratto da una coda circolare, questo provoca uno spostamento (nella figura, in senso antiorario) dell'indice che individua il primo elemento della coda. In questo modo, la cella liberata dal prelievo rimane contigua alle celle già libere, e può essere sfruttata per successivi inserimenti.
Si definisca la struttura dati che implementa la lista circolare.
Si sviluppino i sottoprogrammi per l'inserimento e l'estrazione degli elementi dalla coda, nel caso in cui queste siano pratiche catastali (si veda l'esercizio precedente).

Suggerimento: poichè la memoria del calcolatore è lineare e non circolare, un'implementazione della coda circolare può essere ottenuta utilizzando un array e facendo in modo che l'indice che scandisce le celle dell'array assuma il valore 0 come valore successivo a quello corrispondente all'ultima posizione dell'array.

Soluzione (parziale)

Come nell'esercizio precedente, la seguente soluzione implementa una coda circolare di interi. Estendere tale soluzione per risolvere il problema proposto nell'esercizio.
La struttura dati che consente di rappresentare una coda circolare può essere quella già utilizzata nell'esercizio precedente per rappresentare una coda lineare. Anche le operazioni da implementare sono sostanzialmente uguali, a meno di qualche differenza dovuta al diverso trattamento degli indici che consentono la scansione della coda.
I problemi principali da affrontare per il trattamento degli indici di una coda circolare sono due:

1. Come fare in modo che gli indici che scandiscono le celle dell'array assumano il valore 0 come valore successivo a quello corrispondente all'ultima posizione dell'array. Per risolvere questo problema si può operare in due modi. Un primo modo consiste nell'utilizzare un frammento di programma del tipo mostrato sotto per incrementare ciascun indice.
   if (indice < MAX_DIM-1)
      indice++;
   else indice = 0;
   Un secondo modo più compatto è quello di utilizzare la seguente formula:
   indice = (indice + 1)%MAX_DIM
   Si ricorda che l'operatore % (modulo) consente di ottenere il valore del resto di una divisione intera. Nel codice si usa questo secondo approccio.
2. Come accorgersi che la coda è vuota. L'uguaglianza tra i due indici che si riferiscono al primo e all'ultimo elemento della coda può indicare in questo caso sia che la coda è vuota, sia che essa è completamente piena. Di conseguenza, è necessario aggiungere un'ulteriore variabile booleana che indica quale dei due casi si verifica effettivamente. Nel codice della soluzione questa variabile booleana è stata incapsulata nella struct che rappresenta la coda circolare. Essa viene aggiornata opportunamente nei sottoprogrammi di inizializzazione della coda, inserimento ed estrazione di elementi.

```
   #include <stdio.h>

   #define MAX_NUM_EL   5
typedef enum {Falso, Vero} boolean;

typedef struct {
  int el[MAX_NUM_EL];
  int primoEl;
  int ultimoEl;
  boolean vuota;
} codaCirc;

/* inserisce nella posizione ultimoEl della coda l'elemento il
   cui valore si trova in elem. Se la coda e` piena, ritorna il
   valore zero che rappresenta un codice di errore  */
int inserisci(int elem, codaCirc *c)
{
  if(c->vuota == Falso && c->ultimoEl == c->primoEl)
    return 0;
  else
    {
      c->el[c->ultimoEl] = elem;
      c->ultimoEl = (c->ultimoEl + 1)%MAX_NUM_EL;
      if (c->vuota == Vero)
       c->vuota = Falso;
      return 1;   /* l'inserimento e` stato effettuato con successo */
    }
}

/* estrae l'elemento che si trova in prima posizione (primoEl)
   nella coda. Se la coda e` vuota, ritorna il valore 0 */
int estrai(codaCirc *c)
```

```
{
  int elem;
  if(c->vuota == Falso)
    {
      /* c'e` almeno un elemento nella coda */
      elem = c->el[c->primoEl];
      c->primoEl = (c->primoEl+1)%MAX_NUM_EL;
      if(c->primoEl == c->ultimoEl) /* la coda si e` svuotata */
       c->vuota = Vero;
    }
  else elem = 0;  /* si noti che se 0 rappresenta un codice di
                     errore, tale valore non deve mai apparire nella
                     coda come valore significativo */
  return(elem);
}

/* assegna un valore iniziale ai campi primoEl, ultimoEl e vuota della
   coda. Questo sottoprogramma deve essere sempre richiamato
   per preparare la coda ad essere utilizzata in modo corretto. */
void inizializzaCoda(codaCirc *c)
{
  c->primoEl = 0;
  c->ultimoEl = 0;
  c->vuota = Vero;
}

/* programma principale di prova */
void main()
{
  codaCirc C; /* dichiarazione della coda */
  int scelta, numero; /* variabili utilizzate per la gestione dei menu */
  int ris; /* variabile utilizzata per memorizzare il risultato delle
              chiamate alle due funzioni di inserimento ed estrazione */

  /* prepara la coda ad essere utilizzata */
  inizializzaCoda(&C);

  scelta=0;

  while(scelta!=3)
    {
      printf("*********************************************\n");
      printf("************ MENU PRINCIPALE ****************\n");
      printf("*********************************************\n\n");

      printf("\t1) Inserisci\n");
      printf("\t2) Estrai\n");
      printf("\t3) Esci\n\n");
      printf("Cosa vuoi fare? ");

      scanf("%d",&scelta);

      switch(scelta)
      {
         case 1:
            printf("digita l'intero da inserire nella coda\n");
            scanf("%d", &numero);
            if (numero != 0)
             ris= inserisci(numero, &C);
          if(ris == 0)
            printf("Errore! La coda e` piena\n");
           break;
         case 2:
            numero = estrai(&C);
            if (numero!=0)
              printf("il numero prelevato dalla coda e' :%d\n", numero);
            else printf("la coda e' vuota\n");
            break;
```

```
         case 3:
            printf("Programma terminato\n");
            break;
         default:
            printf("Devi inserire un numero compreso tra 1 e 3!");
            break;
      }
      printf("\n\n");
   }

}
```

Esercizio 7.06 L'archivio degli impiegati

Si sviluppi un programma che consenta ad una azienda di mantenere un archivio dei propri impiegati. Il programma verrà utilizzato dal responsabile della gestione del personale dell'azienda oppure da un suo delegato e deve consentire:

- L'immissione dei dati di un nuovo impiegato.
- La cancellazione dei dati relativi ad un impiegato che non lavora più per l'azienda in questione
- La ricerca dei dati di un impiegato a partire dalla conoscenza del numero di matricola
- L'aggiornamento dello stipendio di ciascun impiegato. Si suppone che l'aggiornamento venga effettuato incrementando della stessa quantità percentuale lo stipendio degli impiegati.
- La stampa degli stipendi degli impiegati per la gestione delle buste paga.
- La stampa di tutte le informazioni in archivio.

Le informazioni relative agli impiegati e rilevanti per l'azienda sono il nome e l'indirizzo dell'impiegato, lo stipendio ed il numero di matricola. Lo stipendio viene assegnato all'impiegato al momento della sua assunzione e viene incrementato periodicamente (insieme a quello di tutti gli altri impiegati) utilizzando la funzionalità di aggiornamento degli stipendi. Il programma deve garantire che il numero di matricola sia univoco per ogni impiegato. Infine, il programma deve conservare in modo persistente le informazioni relative agli impiegati.

Soluzione

La soluzione presentata qui è una estensione della soluzione che è presentata con maggior dettaglio all'indirizzo
http://www.elet.polimi.it/upload/dinitto/Didattica/Info/InfoB0001/materiale%20didattico/impiegati.htm
Rispetto alla versione presente sul web, la presente versione contiene la parte di gestione dei file ed implementa il sottoprogramma di aggiornamento dello stipendio di un impiegato

```
/* direttive per il precompilatore */
#include <stdio.h>
#define MAX_NUM_CHAR 20
#define MAX_ARCH 100

#define ARCH_NAME "archImp.arc"

/* definizione dei tipi */
typedef char stringa[MAX_NUM_CHAR];

typedef struct
{
  stringa nome;
  stringa indirizzo;
  int matricola;
  int stipendio;
} impiegato;

typedef struct
{
  impiegato arch[MAX_ARCH];
  int numImp;
} archivioImp;
```

```
typedef enum {falso, vero} boolean;

/* intestazioni sottoprogrammi */
boolean inserisci(archivioImp *a, impiegato i);
boolean cancella(archivioImp *a, int m);
void aggiornaStipendi(archivioImp *a, float incr);
void stampaStipendi(archivioImp a);
void stampaArchivio(archivioImp a);

void stampaMenu();
impiegato leggiImpiegato();
void caricaArchivio(archivioImp *a);
void salvaArchivio(archivioImp a);

/* main */
void main()
{
  int scelta;
  archivioImp archivio;
  impiegato imp;
  boolean risultato;
  float stip;

  /* a->numImp = 0; */
  caricaArchivio(&archivio);

  /* stampa menu */
  stampaMenu();
  /* acquisizione scelta utente */
  scanf("%d", &scelta);
  /* interpretazione scelta ed esecuzione
     della relativa operazione */
  while(scelta!=6)
    {
      if(scelta == 1)
                {
                  imp = leggiImpiegato();
                  risultato = inserisci(&archivio, imp);
                  if(risultato == falso)
                    printf("Errore: archivio pieno\n");

                }
      else if(scelta == 2)
                printf("la scelta e` cancella\n");
      else if (scelta == 3)
                {
                  printf("Inserisci l'incremento percentuale di
stipendio: ");
                  scanf("%f", &stip);
                  aggiornaStipendi(&archivio, stip);
                }
      else if(scelta == 4)
                stampaStipendi(archivio);
      else if(scelta == 5)
                stampaArchivio(archivio);
      stampaMenu();
      scanf("%d", &scelta);
    }
  salvaArchivio(archivio);
}
/* implementazioni sottoprogrammi */
```

```
void stampaMenu()
{
  printf("Scegli una delle seguenti opzioni: \n");
  printf("1. Inserisci\n");
  printf("2. Cancella\n");
  printf("3. Aggiorna stipendi\n");
  printf("4. Stampa stipendi\n");
  printf("5. Stampa contenuto archivio\n");
  printf("6. Termina il programma\n");
}

impiegato leggiImpiegato()
{
  impiegato x;
  printf("Inserisci il nome: ");
  scanf("%s", x.nome);
  printf("Inserisci il indirizzo: ");
  scanf("%s", x.indirizzo);
  printf("Inserisci lo stipendio: ");
  scanf("%d", &x.stipendio);
  return x;
}

boolean inserisci(archivioImp *a, impiegato i)
{
  if(a->numImp == MAX_ARCH)
    return falso;
  else
    {
      i.matricola = a->numImp+1;
      /*  (*a).arch[(*a).numImp] = i; */
      a->arch[a->numImp] = i;

      /*  (*a).numImp = (*a).numImp +1; */
      a->numImp = a->numImp + 1;
      return vero;
    }
}

/* da implementare da soli */
boolean cancella(archivioImp *a, int m)
{
}

void aggiornaStipendi(archivioImp *a, float incr)
{
  int i;
  for(i=0; i < a->numImp; i++)
    a->arch[i].stipendio =
      a->arch[i].stipendio + a->arch[i].stipendio * incr;
}

/* da implementare da soli */
void stampaStipendi(archivioImp a)
{
}
void stampaArchivio(archivioImp a)
{
  int i;
  for(i=0;i<a.numImp;i++)
    {
```

```
      printf("%s, %s, %d, %d\n",
                    a.arch[i].nome, a.arch[i].indirizzo,
a.arch[i].stipendio, a.arch[i].matricola);
    }
}

void caricaArchivio(archivioImp *a)
{
  FILE *f; int numElem;

  f = fopen(ARCH_NAME, "rb");
  if(f == NULL)
    {
      printf("Archivio non esistente o corrotto, ne creo uno
nuovo\n");
      a->numImp = 0;
    }
  else
    {
      if(fread(&numElem, sizeof(int), 1, f)<1)
                {
                  printf("Archivio corrotto, ne creo uno nuovo\n");
                  a->numImp = 0;
                }
      else
                {
                  a->numImp = numElem;
                  if(fread(a->arch, sizeof(impiegato), numElem,
f)<numElem)
                    {
                      printf("Archivio corrotto, ne creo uno
nuovo\n");
                      a->numImp = 0;
                    }
                }
      fclose(f);
    }
}

void salvaArchivio(archivioImp a)
{
  FILE *f;

  f = fopen(ARCH_NAME, "wb");
  if(f==NULL)
    printf("Impossibile salvare i dati in archivio. I dati verranno
persi\n");
  else
    {
      if(fwrite(&a.numImp, sizeof(int), 1, f)!=1)
                printf("Errore di scrittura nel file. I dati potranno
essere persi\n");
      else if(fwrite(a.arch, sizeof(impiegato), a.numImp,
f)!=a.numImp)
                printf("Errore di scrittura nel file. I dati potranno
essere persi\n");
      fclose(f);
    }
}
```