Esercitazione Programmazione Procedurale con Laboratorio

Esercitazione aggiuntiva

Esercizio 1 Massimo comune divisore

Si scriva un programma in linguaggio C per calcolare il massimo comune divisore (MCD) di due numeri interi positivi (letti da tastiera). Il MCD è definito come il massimo tra i divisori comuni ai due numeri. Suggerimento. Si considerino due numeri interi N1 e N2. Il MCD di N1 e N2 è il massimo tra i numeri che sono divisori (con resto uguale a zero) sia di N2 che di N1. In particolare, si supponga che sia N1 minore di N2. Il MCD è il massimo tra i numeri compresi tra 1 e N1 che sono divisori (con resto uguale a zero) sia di N1 che di N2.

Soluzione dell'esercizio 1

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

int main(void){

int numero1, numero2;  /* numeri inseriti */
int minimo; /* valore minimo tra numero1 e numero2 */
int divisore; /* divisore del numero. E' un contatore per scandire tutti i valori tra 1 e "minimo" */
int mcd; /* massimo comun divisore */

/* leggi i due numeri */
printf("inserisci il primo numero: ");
scanf("%d", &numero1);
printf("inserisci il secondo numero: ");
scanf("%d", &numero2);

/* controlla se entrambi i numeri sono positivi */
if ( numero1 <= 0 || numero2 <= 0 )
  printf("errore: hai inserito un numero nullo o negativo\n");
else
{
  /* calcola il valore inferiore tra i due numeri inseriti*/
  if ( numero1 < numero2 )
    minimo = numero1;
  else
    minimo = numero2;

  /* per calcolare il massimo comun divisore considera
  tutti i numeri compresi tra 1 e "minimo". il massimo comun divisore e' il massimo tra i valori
    compresi tra 1 e "minimo" che e' divisore sia di "numero1" che di "numero2" */
  divisore= 1;
  mcd= 1;

  while ( divisore <= minimo )
  {
    /* verifica se il numero rappresentato in "divisore"
    e' divisore, con resto uguale a 0, sia di "numero1" che di "numero2" */
    if ( numero1%divisore == 0 && numero2%divisore == 0 )
    {
      /* poiche' il resto e' uguale a 0, il valore di "divisore" e' un possibile massimo comune
      divisore. aggiorna il valore del massimo comum divisore */
      mcd = divisore ;
      printf("%d e' divisore \n", mcd);
    }

    /* incrementa il valore di "divisore" */
```

```
    divisore = divisore + 1;
  }

  /* stampa il risultato */
  printf("\n");
  printf("il massimo comun divisore per i numeri %d e %d e' %d\n", numero1, numero2, mcd);
}
exit(0);
}
```

Esercizio 2 Stampa triangolo di Floyd

Scrivere un programma in linguaggio C per la rappresentazione del triangolo di Floyd. Il triangolo di Floyd è un triangolo rettangolo che contiene numeri naturali, definito riempiendo le righe del triangolo con numeri consecutivi e partendo da 1 nell'angolo in alto a sinistra. Si consideri ad esempio il caso N=5. Il triangolo di Floyd è il seguente:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Il programma riceve da tastiera un numero intero N. Il programma visualizza le prime N righe del triangolo di Floyd. Suggerimento. Si osserva che il numero di valori in ogni riga corrisponde all'indice della riga: 1 valore sulla prima riga, 2 sulla seconda, 3 sulla terza.

Soluzione dell'esercizio 2

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

int main(void) {
int numero; /*N*/
int riga, colonna; /* riga e colonna del triangolo */
int cifra; /* numero da stampare nel triangolo di Floyd */

/* leggi un numero */
printf("Inserisci il numero ") ;
scanf("%d",&numero);

/* controlla se il numero e' maggiore di 0 */
if ( numero <=0 )
  printf("Errore, il lato deve essere maggiore di zero\n") ;
else
{
  /* il ciclo piu' esterno scandisce la righe del triangolo */
  /* inizializza la variabile per la scansione delle righe del triangolo */
  riga= 0;

  /* la prima cifra da stampare nel triangolo e' 1 */
  cifra= 1;

  while ( riga < numero )
  {
    /* il ciclo piu' interno scandisce le colonne del triangolo */
    /* per ogni riga stampa il valore in "cifra" solo se colonna <= riga */
    /* inizializza la variabile per la scansione delle colonne del triangolo */
    colonna = 0;

      while ( colonna <= riga )
      {
        /* stampa "cifra" */
        printf("%d ", cifra);

        /* incrementa "colonna" per passare alla colonna successiva */
        colonna = colonna + 1;

        /* incrementa "cifra" */
        cifra= cifra + 1;
      }
```

```
43
44      /* terminata la stampa di una riga si riporta il cursore al margine sinistro dello schermo */
45      printf("\n");
46
47      /* incrementa "riga" per passare alla riga successiva */
48      riga = riga + 1;
49    }
50  }
51
52  exit(0);
53  }
```

Esercizio 3 Numero di Fibonacci

Scrivere un programma in linguaggio C che calcoli e stampi i primi N numeri della serie di Fibonacci, con N inserito da tastiera. La serie di Fibonacci inizia con 1, 1 ed ogni numero successivo è dato dalla somma dei due precedenti: 1, 1, 2, 3, 5, 8, 13, 21...

Soluzione dell'esercizio 3

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

int main(void) {

int n;
int nuovo_termine;
int prec_1, prec_2;

/* due termini precedenti nella serie */
int num_termini;

/* contatore per scandire i termini della serie */

/* leggi il numero termini della sequenza */
printf("inserisci il numero di termini della serie di fibonacci: ") ;
scanf("%d", &n);

/* inizializza a 1 i primi due termini della serie */
prec_1 = 1;
prec_2 = 1;

/* inizializza a 1 il primo valore della serie */
nuovo_termine = 1;

/* inizializza a 0 il contatore che scandisce i termini della serie */
num_termini = 0;

while ( num_termini < n )
{

/* i primi due termini della serie sono uguali a 1.
i termini successivi sono calcolati come somma dei due termini precedenti */
  if ( num_termini >= 2 )
  {
    /* calcola il nuovo termine della serie */
    nuovo_termine = prec_1 + prec_2;

    /* aggiorna il valore dei due termini precedenti nella serie */
    prec_2 = prec_1;
    prec_1 = nuovo_termine;
  }

  /* stampa un nuovo termine della serie */
  printf("%d ", nuovo_termine);
  /* incrementa il contatore "num_termini" */
  num_termini = num_termini + 1;
}

/* riporta a capo il cursore al termine della stampa della serie */
```

```
printf("\n");

exit(0);
}

```

Esercizio 4 Ricerca di un elemento in un vettore

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. I numeri sono memorizzati in un vettore. Il valore N è inserito dall'utente, ma il vettore puoò contenere al massimo 30 numeri. Terminato l'inserimento della sequenza di numeri, l'utente inserisce un valore di riferimento. Il programma deve indicare se tale valore di riferimento è contenuto nel vettore. Trovare inoltre

- l'elemento massimo del vettore,
- l'elemento minimo del vettore.

Soluzione dell'esercizio 4

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

int main(void) {
const int MAXN = 30; /* dimensione massima del vettore */
int N; /* occupazione effettiva del vettore */
int vet[MAXN]; /* sequenza di numeri interi */
int i; /* indice dei cicli */
int numero; /* numero da ricercare nella sequenza */
int trovato; /* flag per indicare se la sequenza contiene il numero inserito */

/* leggi le dimensioni del vettore */
do
{
  printf("Quanti numeri saranno inseriti? ");
  scanf("%d",&N);

  /* la dimensione massima del vettore e' compresa tra 1 e maxn */
  if ( N > MAXN || N <=0 )
    printf("Errore: il numero deve essere compreso tra %d e 0\n", MAXN) ;

while ( N > MAXN || N <=0 );

/* leggi una sequenza di n numeri interi, memorizzandoli in un vettore */
printf("Inserisci una sequenza di %d numeri\n", N);

for ( i=0; i<N; i++ )
{
  printf("Elemento %d: ", i+1);
  scanf("%d", &vet[i]);
}

printf("\n");

/* stampa il vettore di interi */
printf("La sequenza inserita e' la seguente\n");

for ( i=0; i<N; i++ )
  printf("Elemento %d: %d\n", i+1, vet[i]);
printf("\n");

/* leggi il numero da ricercare nella sequenza */
printf("Inserisci il numero da cercare nella sequenza: ");
scanf("%d", &numero);

/* verifica se la sequenza di numeri contiene il numero inserito */

/* inizializza il flag "trovato". il flag assume i valori
-- "trovato" e' uguale a 0 se il vettore "vet" non contiene il valore "numero"
-- "trovato" e' uguale a 1 se il vettore "vet" contiene il valore "numero" */
```

```
trovato = 0 ;

/* il ciclo for scandisce il vettore "vet" e verifica se contiene
il valore "numero". La ricerca termina quando si trova una cella "vet[i]"
uguale a "numero" o quando sono state considerate tutte le celle del vettore */

for ( i=0; i<N && trovato==0; i++ )
{
  if ( vet[i] == numero )
    /* se "vet" contiene il valore in "numero", aggiorna il flag "trovato" */
    trovato = 1;
}

/* stampa il risultato */
if ( trovato == 0 )
  printf("Il numero %d non e' contenuto nella sequenza inserita\n", numero);
else
  printf("Il numero %d e' contenuto nella sequenza inserita\n", numero);

exit(0);
}
```

L'elemento massimo e minimo si trovano scorrendo l'array nello stesso modo, salvando l'elemento massimo fini a quella iterazione, invece di *trovato*.

Esercizio 5 Verifica vettore ordinato

Scrivere un programma in linguaggio C che riceve in ingresso una sequenza di N numeri interi. I numeri sono memorizzati in un vettore. Il valore N é inserito dall'utente, ma il vettore puó contenere al massimo 30 numeri. Terminato l'inserimento della sequenza di numeri, il programma deve verificare se il vettore contiene una sequenza di numeri ordinata in modo strettamente crescente.

Soluzione dell'esercizio 5

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

int main(void) {

const int maxn = 30; /* dimensione massima del vettore */
int n; /* occupazione del vettore */
int vet[maxn];  /* sequenza di numeri interi */
int i; /* indice dei cicli */
int crescente; /* flag per indicare se la sequenza e' crescente */

/* leggi le dimensioni del vettore */
do
{
  printf("quanti numeri saranno inseriti? ");
  scanf("%d",&n) ;

  /* la dimensione massima del vettore e' compresa tra 1 e maxn */
  if ( n > maxn || n <=0 )
    printf("errore: il numero deve essere compreso tra %d e 0\n", maxn);

while ( n > maxn || n <=0 );

/* leggi una sequenza di n numeri interi, memorizzandoli in un vettore */
printf("inserisci una sequenza di %d numeri\n", n);

for ( i=0; i<n; i++ )
{
  printf("elemento %d: ", i+1);
  scanf("%d", &vet[i]);
}

printf("\n");

/* stampa il vettore di interi */
printf("la sequenza inserita e' la seguente\n");

```

```
38 for ( i=0; i<n; i++ )
39   printf("elemento %d: %d\n", i+1, vet[i]);
40
41 printf("\n");
42
43 /* verifica se la sequenza di numeri e' ordinata in modo crescente */
44 /* inizializza il flag "crescente". il flag assume i valori
45 -- "crescente" e' uguale a 1 se la sequenza e' crescente
46 -- "crescente" e' uguale a 0 se la sequenza non e' crescente */
47 crescente = 1;
48
49 /* il ciclo for scandisce il vettore "vet" e controlla se la sequenza
50 memorizzata nel vettore e' crescente. la ricerca termina quando si verifica
51 che la sequenza non e' crescente o quando sono state considerate tutte
52 le celle del vettore */
53
54 /* nel ciclo for si confronta ogni cella del vettore con la cella precedente.
55 si osserva che la cella con indice 0 (vet[0]) non puo' essere confrontata
56 con la cella precedente (con indice -1). pertanto l'indice "i" del ciclo
57 assume i valori tra 1 e n-1 */
58 for ( i=1; i < n && crescente==1; i++ )
59 {
60   if ( vet[i] <= vet[i-1] )
61   /* sequenza non crescente, aggiorna il flag "crescente" */
62   crescente = 0;
63 }
64
65 /* stampa il risultato */
66 if ( crescente == 0 )
67   printf("la sequenza non e crescente\n");
68 else
69   printf("la sequenza e crescente\n");
70
71 exit(0);
72 }
```