Problem solving elementare su dati scalari

Esercizi risolti

1 Esercizio: “Fattoriale”

Si realizzi un programma che, letto un numero n, stampi il valore del fattoriale per tutti i numeri da 0 a n. Si ricordi che 0!=1.

Soluzione

```
#include <stdio.h>

void main (void) {
  int i, n, fatt;

  printf ("Valore (>=0) di cui calcolare il fattoriale: ");
  scanf ("%d", &n);

  if (n>=0) {
    /* calcolo del fattoriale */
    fatt = 1;
    for (i=2; i<=n; i++)
      fatt = fatt * i;

    printf ("\n%d! = %d \n", n, fatt);
  }
}
```

Osservazione

Per ottenere la visualizzazione dei fattoriali di tutti i numeri minori o uguali a n (0 e 1 compresi), il programma può essere modificato come segue.

```
#include <stdio.h>

void main (void)
{
  int i, n, fatt=1;     /* 0! = 1 */

  printf ("Introdurre il limite ((<0)=FINE): ");
  scanf ("%d", &n);

  if (n>=0)
  {
    printf ("0! = 1\n");

    /* inizio del ciclo da 1 per comprendere la stampa di 1! */
    for (i=1; i<=n; i++)
    {
      fatt = fatt * i;
      printf ("%d! = %d\n", i, fatt);
    }
  }
}
```


2 Esercizio: “Area di triangolo rettangolo dati tre lati”

Si scriva un programma che, letti da tastiera 3 numeri interi, che questi rappresentano le dimensioni dei 3 lati di un triangolo rettangolo

- determini quale dei 3 lati rappresenta l’ipotenusa del triangolo
- calcoli e visualizzi l’area del triangolo stesso.

Soluzione

Il problema richiede di determinare in via preliminare quali siano i cateti del triangolo, prima di procedere al calcolo dell’area. Va poi notato che, mentre i tre lati vengono considerati all’introduzione come valori interi, il valore dell’area potrà essere un valore reale. Occorre quindi effettuare un’opportuna operazione di cast durante il calcolo stesso dell’area.

```
#include <stdio.h>

void main (void)
{
  int a, b, c;
  float area;

  /* lettura lati da tastiera */
  printf ("Scrivi il primo lato: ");
  scanf ("%d", &a);
  printf ("Scrivi il secondo lato: ");
  scanf ("%d", &b);
  printf ("Scrivi il terzo lato: ");
  scanf ("%d", &c);

  /* selezione dei tre casi */
  if (a>b && a>c)
  {
      printf ("L’ipotenusa è il primo lato \n");
      area = ((float)(b * c)) / 2.0;
  }
  else if (b>a && b>c)
  {
    printf ("L’ipotenusa è il secondo lato \n");
    area = ((float) (a * c)) / 2.0;
  }
  else if (c>b && c>a)
  {
    printf ("L’ipotenusa è il terzo lato \n");
    area = (float) (b * a)) / 2.0;
  }

  /* visualizzazione dell’area */
  printf ("Area = %f\n", area);
}
```

La soluzione precedente riconosce i tre casi mediante tre costrutti `if` basati ognuno su 2 confronti. Una soluzione alternativa (con esecuzione di un minor numero di confronti) è la seguente:

```
#include <stdio.h>

void main (void)
{
  int a, b, c;
  float area;

  /* lettura lati da tastiera */
```

```
  printf ("Scrivi il primo lato: ");
  scanf ("%d", &a);
  printf ("Scrivi il secondo lato: ");
  scanf ("%d", &b);
  printf ("Scrivi il terzo lato: ");
  scanf ("%d", &c);
  /* confronti */
  if ( a > b ) /* L'ipotenusa non e' b */
    if ( a > c )
    {
      printf ("L'ipotenusa è il primo lato \n");
      area = ((float) (b * c)) / 2.0;
    }
    else
    {
      printf ("L'ipotenusa è il terzo lato \n");
      area = ((float) (a * b)) / 2.0;
    }
  else /* L'ipotenusa non e' a */
    if ( b > c )
    {
      printf ("L'ipotenusa è il secondo lato \n");
      area = ((float) (a * c)) / 2.0;
    }
    else
    {
      printf ("L'ipotenusa è il terzo lato \n");
      area = a * b / 2;
    }

  /* Visualizzazione dell'area */
  printf ("Area = %f\n", area);
}
```

3 Esercizio: "Media aritmetica"

Si scriva un programma che esegua le seguenti operazioni:

- legga una serie di numeri da terminale sino all'introduzione del numero 0
- calcoli e visualizzi la media dei numeri introdotti (lo 0 teminale non va conteggiato)

Soluzione

```
#include <stdio.h>

void main (void) {
  int  numero,
       somma=0,
       termini=0;

  printf ("Scrivi un numero: <0 per finire>: ");
  scanf ("%d", &numero );
  while (numero != 0) {
    somma += numero;
    termini++;
    printf ("Scrivi un numero: <0 per finire>: ");
    scanf ("%d", &numero);
  }

  if (termini!=0)
    printf ("Media = %f\n", ((float) somma)/((float) termini));
  else
    printf ("Nessun numero inserito.\n");
}
```

4 Esercizio: “Numeri triangolari, quadrati e pentagonali”

Realizzare tre programmi che:

- leggano da tastiera un numero intero n
- visualizzino ciascuno una delle seguenti grandezze:
  - il numero *Triangolare* T_n, definito come:

$$T_n = 1 + 2 + 3 + ... + n = \sum_{i=1}^{n} i$$

  - il numero *Quadrato* Q_n, definito come:

$$Q_n = 1 + 3 + 5 + 7 + ... + (2n - 1) = \sum_{i=1}^{n} (2i - 1)$$

  - il numero *Pentagonale* P_n, definito come:

$$P_n = 1 + 4 + 7 + 10 + ... + (3n - 2) = \sum_{i=1}^{n} (3i - 2)$$

Soluzione

```
/* NUMERO TRIANGOLARE */
#include<stdio.h>

void main (void) {
  int n, x=0;

  printf ("Introduci n: "); scanf ("%d",&n);
  while (n>0) {
    x += n;
    n--;
  }
  printf ("Risultato: %d\n",x);
}
```

```
/* NUMERO QUADRATO*/
#include <stdio.h>

void main (void) {
  int n, x=0;

  printf ("Introduci n: "); scanf ("%d", &n);
  while(n>0) {
    x += 2*n-1;
    n--;
  }
  printf ( "Risultato: %d\n", x);
}
```

```
/* NUMERO PENTAGONALE*/
#include <stdio.h>

void main (void) {
  int n, x=0;

  printf ("Introduci n: "); scanf ("%d", &n);
  while (n>0) {
    x += 3*n-2;
    n--;
  }
  printf ( "Risultato: %d\n", x);
}
```

5 Esercizio: “Codici ASCII”

Stampare una tabella riportante, per ciascun carattere alfabetico (minuscolo e maiuscolo), il rispettivo codice ASCII come valore decimale, ottale e esadecimale. La tabella dovrà essere costituita da un insieme di righe, ciascuna delle quali contiene otto colonne. Il carattere ASCII minuscolo appare nella prima colonna, seguito dal rispettivo valore decimale, ottale e esadecimale. Il corrispondente carattere maiuscolo compare, invece, nella quinta colonna, anch’esso seguito dai valori decimale, ottale e esadecimale.

Soluzione

Si noti come si utilizzi l’istruzione:

```
if ( (num%N)==0 )
```

per interrompere la visualizzazione dopo N righe consecutive. La visualizzazione non riprende sino a quando non viene introdotto il carattere desiderato (a-capo):

```
do { scanf ("%c", &c);
} while (c!='\n');
```

```
#include<stdio.h>

#define N 10

void main (void) {
  char c, c1, c2;
  int num, i, j;

  /* Stampa della intestazione della tabella */
  printf ("char\tDec\tOtt\tEsa \t\t Char\tDec\tOtt\tEsa\n");
  num = 1;
  for (c1='a'; c1<='z'; c1++) {
    c2 = c1 - 'a' + 'A';
    i = (int) (c1); j = (int) (c2);
    /* stampa di una riga della tabella */
    printf ("%c\t%d\t%o\t%x \t\t %c\t%d\t%\t%x\n", c1, i, i, i, c2, j, j,
j);
    /* ogni N righe premere return per continuare */
    if ((num%N)==0) {
      printf ("\n <return> per continuare\n");
      do { scanf ("%c", &c);
      } while (c!='\n');
    }
    num++;
  }
}
```

6 Esercizio: “Numeri romani”

Realizzare un programma che legga da terminale un numero intero, lo converta nel corrispondente numero romano e lo visualizzi.

Si ricordi che:

1 = I

5 = V

10 = X

50 = L

100 = C

500 = D

1000 = M

e si utilizzi un meccanismo di traduzione semplificata come indicato dall'esempio seguente.

Esempio

Il numero 4 venga tradotto come IIII invece che come IV; il numero 48 venga tradotto come XXXXVIII (traduzione corretta), il numero 49 come XXXXVIIII e non come IL, e così via. Non è cioè richiesta la gestione della regola che prevede la "sottrazione di un valore minore" al successivo "valore maggiore".

Si scriva una funzione in grado di contare il numero di giorni intercorsi tra due date, ricevute come parametri in ingresso. Si faccia riferimento ad un tipo data_t, realizzato mediante struct, quale quello utilizzato nell'esercizio 3.

Soluzione

```
#include <stdio.h>

void main (void)
{
  int numero, i;
  /* memorizzazione, nel vettore romano, dei caratteri "base" romani */
  char romano[7] = {'I', 'V', 'X', 'L', 'C', 'D', 'M'};
  /* memorizzazione, nel vettore arabo, dei corrispondenti valori decimali
*/
  int arabo[7] = {1, 5, 10, 50, 100, 500, 1000};

  printf ("Dammi il numero: ");
  scanf("%d", &numero);

  /* il programma cerca iterativamente il più grande numero
     valore di arabo[], lo sottrae al numero e ne stampa il
     corrispettivo romano */
  i = 6; /* 6 e' l'indice dell'ultima casella dei vettori */
  while (numero != 0)
  {
    if (numero >= arabo[i])
    {
      printf ("%c", romano[i]);
      numero = numero - arabo[i];
    }
    else
      i--;
  }

  printf ("\n");
}
```

7 Esercizio: “Triangolo di Floyd”

Si realizzi un programma che:

- legga un numero intero n
- visualizzi le prime n righe del *Triangolo di Floyd*, così come definito nella figura seguente:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

...

Soluzione

Supponendo non si debbano visualizzare numeri con più di tre cifre il seguente programma produce un corretto incolonnamento con l’utilizzo della direttiva `%3d`.

```
#include <stdio.h>

void main (void)
{
  int n, righe, colonne, prog;

  prog = 1;
  printf ("Scrivi il numero di righe: ");
  scanf ("%d", &n);

  for (righe=1; righe<=n; righe++)
  {
    for (colonne=1; colonne<=righe; colonne++)
      printf ("%3d ", prog++);
    printf ("\n");
  }
}
```

8 Esercizio: “Riconoscimento di una sequenza”

Si scriva un programma che legga una serie di caratteri (singoli) da tastiera (ognuno seguito da `invio`) sino a quando si rileva nel flusso di ingresso la presenza della parola `ciao` (ossia la sequenza dei quattro caratteri consecutivi `'c'`, `'i'`, `'a'`, `'o'`).

Soluzione

L’idea fondamentale è quella di utilizzare una variabile `stato` per indicare “lo stato” del programma, ovvero a quale punto del riconoscimento ci si trova in ciascun istante. Il programma si trova inizialmente in uno stato caratterizzato da `stato = 0`. Se in tale situazione si introduce un carattere `'c'` si passa a `stato = 1`, altrimenti si rimane in `stato = 0`. Quando `stato = 1` se si introduce il carattere `'i'` si passa a `stato = 2`, altrimenti si ritorna a `stato = 0`. Analogamente da `stato = 2` si passa a `stato = 3` introducendo un carattere `'a'`, e così via. La prima soluzione presentata utilizza istruzioni `if` per effettuare i controlli richiesti.


```
#include <stdio.h>

void main (void) {
  char c;
  int stato;   /* stato: se
                  ==0 nulla di riconosciuto
                  ==1 riconosciuto carattere 'c'
                  ==2 riconosciuti caratteri 'c', 'i'
                  ==3 riconosciuti caratteri 'c', 'i', 'a'
                  ==4 riconosciuti caratteri 'c', 'i', 'a', 'o' */

  printf ("Introduci il testo un carattere alla volta.\n\n");

  stato = 0;
  do {
    scanf ("%c%*c", &c);
    /* %*c per leggere e saltare il carattere di invio */

    if ( c=='c' )
      stato = 1;
    else
      if (c=='i') {
        if (stato==1)
          stato = 2;
        else
          stato = 0;
      }
      else
        if (c=='a') {
          if (stato==2)
            stato = 3;
          else
            stato = 0;
        }
        else
          if ( c=='o' ) {
            if (stato==3)
              stato = 4;
            else
              stato = 0;
          }
          else
            stato = 0;
  } while (stato < 4);

  printf ("\n\nSequenza riconosciuta.\n\n");
}
```

La soluzione seguente utilizza un costrutto `switch` e riconosce tanto caratteri minuscoli quanto caratteri maiuscoli (`"ciao"`, `"CIAO"`, `"CiaO"`, …):

```
#include <stdio.h>

void main (void) {
  char c;
  int stato;   /* stato: se
                  ==0 nulla di riconosciuto
                  ==1 riconosciuto carattere 'c'
                  ==2 riconosciuti caratteri 'c', 'i'
                  ==3 riconosciuti caratteri 'c', 'i', 'a'
                  ==4 riconosciuti caratteri 'c', 'i', 'a', 'o' */

  printf ("Introduci il testo un carattere alla volta.\n\n");

  stato = 0;
  do {
```

```
    scanf ("%c%*c", &c);
    /* %*c per leggere e saltare il carattere di invio */

    switch (c) {
      case 'c':
      case 'C': stato = 1;
                break;
      case 'i':
      case 'I': if (stato==1)
                  stato = 2;
                else
                  stato = 0;
                break;
      case 'a':
      case 'A': if (stato==2)
                  stato = 3;
                else
                  stato = 0;
                break;
      case 'o':
      case 'O': if (stato==3)
                  stato = 4;
                else
                  stato = 0;
                break;
      default : stato = 0;
                break;
    }
  } while (stato < 4);

  printf ("\n\nSequenza riconosciuta.\n\n");
}
```

9 Esercizio: “Numeri primi”

Si scriva un programma che effettui:

- la lettura di un numero
- il controllo per verificare se il numero è primo
- la visualizzazione di un opportuno messaggio a seconda che il numero sia primo oppure no.

Soluzione

Utilizzando la definizione per la quale un numero è primo solo se è divisibile unicamente per il numero 1 e per se stesso, si ottiene il programma seguente.

```
#include <stdio.h>

void main (void)
{
  int n, i, primo;

  /* lettura del parametro di ingresso */
  printf ("Numero: ");
  scanf ("%d", &n);

  if (n==0)
    printf ("\n\nNumero NULLO.\n\n");
  else {
    /* se n e' negativo se ne prende il valore assoluto */
    if (n<0) n = -n;
```

```
    /* verifica iterativa: divisibilità per tutti i numeri inferiori a n */
    flag = 1;
    /* N.B. la variabile intera primo viene utilizzata
       come un valore booleano per segnalare lo strato del numero:
       primo/non-primo. Se il numero non e' primo si interrompono
       le iterazioni */

    for (i=2; ((i<n)&&(primo==1)); i++)
      if ((n%i)!=0) primo = 0;

    if (primo==1)
      printf ("\n\nNumero PRIMO.\n\n");
    else
      printf ("\n\nNumero NON primo.\n\n");
  }
}
```