Soluzioni

Esercizio 4:

Scrivere un programma 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:

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int N; /* numero di termini della serie */
    int nuovo_termine; /* nuovo termine della serie */
    int prec1, prec2; /* 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 */
    prec1 = 1;
    prec2 = 1;

    /* INIZIALIZZA A 1 IL PRIMO VALORE DELLA SERIE */
    nuovo_termine = 1;
    num_termini = 0;
    while(num_termini < N)
    {
        if(num_termini >= 2)
        {
            /* CALCOLA IL NUOVO TERMINE DELLA SERIE */
            nuovo_termine = prec1 + prec2;
            /* AGGIORNA IL VALORE DEI DUE TERMINI PRECEDENTI NELLA SERIE */
            prec2 = prec1;
            prec1 = nuovo_termine;
        }
        /* STAMPA UN NUOVO TERMINE DELLA SERIE */
        printf("%d ", nuovo_termine);
        /* INCREMENTA IL CONTATORE */
        num_termini = num_termini + 1;
   }
   printf("\n");
}
```

Esercizio 6:

Scrivere un programma tavola.c che:

- Chieda in ingresso un intero N;
- Stampi a video la tavola pitagorica per i fattori fino a N compreso.

Esempio:

```
"Inserisci il numero di righe della tavola pitagorica: 10"
// Stampa della tavola pitagorica
```

```
TAVOLA PITAGORICA
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

Soluzione:

```
#include <stdio.h>

main() {
  int n, riga, colonna, prodotto;
  printf("Inserire n: ");
  scanf("%d", &n);
  for(riga=0; riga<n; riga++) {
     for(colonna=0; colonna<n; colonna++){
         prodotto = (riga+1)*(colonna+1);
         printf("%d\t", prodotto);
     }
     printf("\n");
  }
}
```

Esercizio 8:

Scrivere un programma che stampi un rettangolo la cui cornice sia costituita da caratteri asterisco e la parte interna da un carattere immesso dall'utente. Anche il numero di righe e di colonne viene deciso dall'utente.
Esempio d'uso (dati di input sottolineati):

```
Inserisci quante righe vuoi: 10
Inserisci quante colonne vuoi: 30
Inserisci il carattere che riempira' il rettangolo: .

******************************
*............................*
*............................*
*............................*
*............................*
*............................*
*............................*
*............................*
*............................*
******************************
```

Soluzione:

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

```
int main() {
  int righe;
  do {
      printf("Inserisci quante righe vuoi: ");
      scanf("%d", &righe);
  } while (righe < 1);

  int colonne;
  do {
      printf("Inserisci quante colonne vuoi: ");
      scanf("%d", &colonne);
  } while (colonne < 1);

  char c;
  printf("Inserisci il carattere che riempira' il rettangolo:");
  scanf("%d", &c);

  for (int i = 1; i <= righe; i++)
    for (int j = 1; j <= colonne; j++)
      if (i == 1 || i == righe || j == 1 || j == colonne) {
            printf('*');
            if (j == colonne)
              printf('\n');
      }
      else
            printf(c);
}
```

Esercizio 9:

Scrivere un programma che legga in input un carattere c e un intero n. Il programma stampa un triangolo di altezza n e base lunga 2n-1, fatto con caratteri c. (Se il valore letto è negativo si consideri il suo valore assoluto)

Esempio di interazione con il programma:

```
Inserisci l'altezza: 6
     *
    ***
   *****
  *******
 *********
***********
```

Soluzione:

```
#include <stdio.h>

main() {

  int altezza;
  int riga,colonna;
  int ret;

  printf("Inserisci l'altezza: ");
  ret=scanf("%d", &altezza);
  if(ret!=1) { printf("Errore!\n"); return; }

  /* per ogni riga */
```

```
  for(riga=0 ; riga<altezza ; riga++) {

    /* prima gli spazi (che diminuiscono andando verso il basso) */
    for(colonna=0 ; colonna<(altezza-riga-1) ; colonna++)
      printf(" ");

    /* poi gli asterischi (che aumentano di 2 ogni volta andando verso il basso)
*/
    for(colonna=0 ; colonna<(2*riga+1) ; colonna++)
      printf("*");

    /* e poi termino la riga */
    printf("\n");
  }
}
```

Esercizio 11:

Modificare il programma 5 per stampare un triangolo isoscele di altezza lunga n (con n compreso tra 0 e 9) e base lunga 2n-1 fatto come sotto. Se il valore letto è non è compreso tra 0 e 9 si chieda all'utente un nuovo valore finché l'intero immesso non appartenga all'intervallo richiesto.

Esempio di interazione con il programma:

```
Dammi un numero intero positivo compreso tra 0 e 9: -2
Dammi un numero intero positivo compreso tra 0 e 9: 5

    1
   212
  32123
 4321234
543212345
```

Soluzione:

```
#include <stdio.h>

main() {

  int altezza=-1;
  int riga,colonna;
  int ret;

  do {
    printf("Dammi un intero positivo compreso fra 0 e 9: ");
    ret=scanf("%d", &altezza);
    if(ret!=1) { printf("Errore!\n"); return; }
  } while(altezza<0 || altezza>9);

  /* per ogni riga */
  for(riga=0; riga<altezza; riga++) {

    /* prima gli spazi (che diminuiscono andando verso il basso) */
    for(colonna=0;colonna<(altezza-riga-1);colonna++)
      printf(" ");

    /* poi le cifre decrescenti (parto dal numero di riga e scendo fino a 1) */
    for(colonna=riga+1;colonna>0;colonna--)
      printf("%d",colonna);
```

```
    /* quindi le cifre crescenti (riparto da dopo l'uno e salgo fino al numero
di riga) */
    for(colonna=1; colonna<riga+1; colonna++)
      printf("%d",colonna+1);

    /* e infine il fine riga */
    printf("\n");
  }

}
```

Esercizio 14:

Sulla base dell'esercizio visto durante la precedente esercitazione, scrivete un programma che calcola la media pesata dei voti di uno studente (la media pesata si calcola moltiplicando ogni voto per il suo peso in crediti, sommando tutti questi valori e dividendo per la somma del numero di crediti).
Quindi uno studente con due esami, fisica 1 (voto: 24, crediti:15) e informatica (voto: 28, crediti:6), avra' una media pesata circa uguale a (24*15 + 28*6)/(15+6) = 25,14.

Il programma deve quindi chiedere ad uno studente di inserire i voti degli esami e il loro peso in crediti, uno per volta. Lo studente dovrà inserire 0 per segnalare che ha terminato l'inserimento. Il programma quindi calcola e stampa la sua media pesata sui crediti. Nota: si tenga conto che la votazione del singolo esame e il numero di crediti sono interi. Inoltre sono votazioni valide per il superamento di un esame solo quelle comprese tra 18 e 30 (estremi inclusi) e il numero di crediti di un esame deve essere maggiore di 0.

Soluzione:

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

/*
  In questo esercizio calcolo la media pesata dei voti di uno studente e cerco
  di calcolare il voto che deve prendere nel prossimo esame per alzarla al
  successivo intero.
  Di nuovo (come nell'esercizio 6) non usare i break mi costringe ad eseguire
  i controlli di terminazione due volte.
  Inoltre li eseguo separatamente perche' voglio messaggi di errore diversi
  (o nessun errore, semplice terminazione).

*/

main() {

  /* somma parziale del prodotto fra ogni voto e il suo peso in crediti */
  float somma_voto_crediti=0.0;

  float somma_crediti=0.0;
  float media;
  float obiettivo;
  float nuovo_voto;

  /* valori di input */
  int voto, crediti;

  /* valore di ritorno di scanf */
  int ret;
```

```
  /* prima il ciclo di lettura dei voti preesistenti */
  do {
    printf("Inserisci il voto (0 per uscire): ");
    ret=scanf("%d", &voto);
    if(ret!=1) { printf("Errore!\n"); return; }

    /* controlli di terminazione e di coerenza per il voto */
    if(voto>0) {
      if(voto>=18 && voto<=30) {
        printf("Inserisci i crediti (0 per uscire): ");
        ret=scanf("%d", &crediti);
        if(ret!=1) { printf("Errore!\n"); return; }

        if(crediti>0) {
          /* sono qui solo se tutto e' a posto */
          somma_voto_crediti += voto*crediti;
          somma_crediti += crediti;
        }
        else
          printf("Errore: il numero di crediti deve essere maggiore di 0\n");
      }
      else
        printf("Errore: il voto deve essere compreso fra 18 e 30.\n");
    }
  } while(voto>=18 && voto<=30 && crediti>0);

  /* faccio il resto solo se ho letto almeno un voto valido */
  if(somma_crediti>0 && somma_voto_crediti>0) {

    media = somma_voto_crediti / somma_crediti;
    printf("La tua media e': %.2f\n", media);
    /* il programma per l'esercizio 7 finiva qui. */

    /* ovviamente voglio aumentare la media solo se e' possibile */
    if(media<30) {
      obiettivo = floor(media) + 1;
      printf("Vediamo se riesci a raggiungere %.0f\n", obiettivo);

      /* chiedo il numero di crediti in un ciclo: in questo caso permetto
      all'utente di inserire numeri non validi e in quel caso continuo a
chiedere.
      Potevo fare come prima che se il numero era sbagliato terminato e sarebbe
      andato bene lo stesso. */
      do {
        printf("Inserisci il numero di crediti del prossimo esame: ");
        ret=scanf("%d", &crediti);
        if(ret!=1) { printf("Errore!\n"); return; }

        if(crediti<=0)
          printf("Valore inserito non valido!\n");

      } while(crediti<=0);

      /* adesso ho tutti i dati, posso eseguire i conti */

      /* questa formula inverte la sommatoria per il voto pesato */
      nuovo_voto = (obiettivo * (somma_crediti + crediti) -
somma_voto_crediti) / crediti;

      /* questo non dovrebbe mai accadere, visto che il voto minore
      accettato precedentemente e' 18. */
      if(nuovo_voto<18) nuovo_voto = 18;

      if(nuovo_voto>30)
```

```
        printf("Mi dispiace, non e' possibile!\n");
      else
        printf("Per ottenere la media del %.0f devi prendere %.0f al prossimo
esame\n", obiettivo, ceil(nuovo_voto));
    }
  }
}
```

Esercizio 15:

Scrivere un programma alfabeto che chiede all'utente una sequenza di caratteri alfabetici minuscoli verificando che ogni carattere letto sia maggiore o uguale ai precedenti (secondo l'ordine alfabetico). Il primo carattere inserito può essere un qualsiasi carattere minuscolo. La sequenza termina quando l'utente immette un carattere non alfabetico o maiuscolo oppure se immette un carattere minore di uno di quelli letti precedentemente. Terminata la lettura dei caratteri il programma deve stampare il numero di caratteri minuscoli diversi appartenenti alla sequenza (il carattere che causa la terminazione non è considerato parte della sequenza). Se la sequenza è vuota, cioè non viene immesso alcun carattere minuscolo, allora il programma stampa solo un avvertimento.

Esempi di esecuzione:

```
Dammi un carattere: X
La sequenza di lettere minuscole e' vuota
Dammi un carattere: a
Dammi un carattere: r
Dammi un carattere: r
Dammi un carattere: 3
Totale lettere minuscole ordinate e diverse: 2

Dammi un carattere: a
Dammi un carattere: a
Dammi un carattere: a
Dammi un carattere: d
Dammi un carattere: z
Dammi un carattere: 4
Totale lettere minuscole ordinate e diverse: 3
```

Soluzione:

```
#include <stdio.h>

/*
  Lettura di una sequenza di lettere alfabetiche.
  Notate che durante il ciclo non mi serve ricordare molto, mi basta sapere
  quante lettere minuscole diverse ci sono state finora (all'inizio 0
ovviamente),
  l'ultimo carattere letto e quello appena precedente.
  Ovviamente devo inizializzare il carattere precedente ad un valore minore
  del primo valore lecito ('a').
  Notare che proseguo nel ciclo se leggo un carattere uguale al precedente ma
  aggiorno il contatore solo se ne ho letto uno maggiore.

*/

main() {
```

```
  int counter = 0;
  char prec, act;
  int ret;

  prec = 'a' - 1;

  printf("Dammi un carattere: ");
  ret=scanf(" %c", &act);
  if(ret!=1) { printf("Errore!\n"); return; }

  /* il carattere deve essere minuscolo e non minore del precedente */
  while('a'<=act && act <='z' && act>=prec) {
    /* se il carattere e' maggiore del precedente, lo conto e aggiorno prec */
    if(act>prec) {
      counter++;
      prec = act;
    }

    printf("Dammi un carattere: ");
    ret=scanf(" %c", &act);
    if(ret!=1) { printf("Errore!\n"); return; }
  }

  if(counter>0)
    printf("Totale lettere minuscole ordinate e diverse: %d\n", counter);
  else
    printf("La sequenza di lettere minuscole e' vuota.\n");

}
```

Esercizio 17

Nella morra due giocatori si sfidano scegliendo un simbolo ciascuno tra sasso, forbici e carta: due simboli uguali pareggiano, mentre il sasso batte le forbici, le forbici battono la carta, e la carta batte il sasso. Scrivere il programma morra che gestisce una sfida tra PC e utente: (a) generando un numero casuale da 1 a 3 così definiti: 1: sasso, 2: forbici, 3: carta (utilizzare il costrutto #define per rendere leggibile l'associazione tra il numero e il simbolo) (b) leggendo un carattere ('s': sasso, 'f': forbici, 'c': carta) (c) stampando l'esito del confronto. Se l'utente immette un carattere diverso da 's', 'f' e 'c' allora perde comunque.

Soluzione:

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

#define SASSO     's'
#define CARTA     'c'
#define FORBICE   'f'

/*
  In questo programma, il computer e il giocatore umano giocano alla morra
cinese. Entrambi scelgono uno dei simboli, il computer sceglie tirando un dado
da 0 a 2.
  Una volta che i due simboli sono noti, il computer calcola il risultato del
gioco secondo le regole classiche: sasso batte forbice, forbice batte carta,
carta batte sasso.
*/
```

```
main () {

  char computer;
  char umano;
  int ret, tirodado;

  printf("Inserisci %c, %c o %c\n", SASSO, CARTA, FORBICE);
  ret=scanf(" %c", &umano);
  if(ret!=1) { printf("Errore!\n"); return; } /* In caso di errore il programma
termina */

  /* se il giocatore ha inserito un simbolo non valido, perde di default. */
  if(umano!=SASSO && umano!=CARTA && umano!=FORBICE) {
    printf("Simbolo sbagliato: perdi automaticamente!\n");
    return;
  }

  /* Il computer tira un dado per ottenere un numero fra 0 e 2 e lo usa per
scegliere un simbolo */
  srand(time(NULL));
  tirodado=rand()%3;
  switch(tirodado) {
    case 0:
      computer=SASSO;
      break;
    case 1:
      computer=CARTA;
      break;
    case 2:
      computer=FORBICE;
      break;
    default:
      printf("Anche i computer sbagliano?\n");
      return;
  }

  /* Quando entrambi i simboli sono noti, il computer decide chi vince secondo
le regole classiche della morra */
  if(umano==computer)
    printf("Pareggiamo: Tu:%c, Io:%c\n", umano, computer);
  else if(umano==SASSO) {
    if(computer==CARTA)
      printf("Ho vinto io: Tu:%c, Io:%c\n", umano, computer);
    else
      printf("Hai vinto tu: Tu:%c, Io:%c\n", umano, computer);
  }
  else if(umano==FORBICE) {
    if(computer==CARTA)
      printf("Hai vinto tu: Tu:%c, Io:%c\n", umano, computer);
    else
      printf("Ho vinto io: Tu:%c, Io:%c\n", umano, computer);
  }
  else /* umano==CARTA */ {
    if(computer==SASSO)
      printf("Hai vinto tu: Tu:%c, Io:%c\n", umano, computer);
    else
      printf("Ho vinto io: Tu:%c, Io:%c\n", umano, computer);
  }

  printf("Ciao\n");

}
```

Esercizio 18:

Si dice che una sequenza di interi non negativi è bilanciata se contiene lo stesso numero di pari e di dispari. Si scriva una funzione iterativa che legga una sequenza di interi non negativi (senza memorizzarla) che termina quando vengono immessi due numeri uguali e che, supponendo di non considerare l'ultimo come parte della sequenza,
restituisca:

- 0 se la sequenza non è bilanciata, né ordinata in modo crescente;
- 1 se la sequenza è bilanciata, e non crescente;
- 2 se la sequenza non è bilanciata, ma è crescente;
- 3 la sequenza è bilanciata e crescente

Soluzione:

```
int bilanciataOrdinata() {
   int prec, act, res=0, pari=0, dispari=0;
   boolean ordinata=true;
   scanf("%d", &act);
   if(act%2==0) pari++;
   else dispari++;
   do {
      prec=act;
      scanf("%d", &act);
      if(act!=prec) {
            if(act<prec) ordinata = false;
            if(act%2==0) pari++;
            else dispari++;
            }
   } while(act!=prec);
   if(ordinata) res += 2;
   if(pari==dispari) res++;
   return res;
}
```