ESERCIZIO 1 **(Costrutti iterativi + uso di contatori e accumulatori + Algoritmi)**

Dato un numero positivo Q, scrivere la sua rappresentazione in binario naturale, applicando il tradizionale algoritmo per divisioni successive (l'output sarà inteso corretto se letto da destra a sinistra) e indicando anche il minimo numero di bit utilizzato.
Es: Input: 19 in decimale, Output: (LSB) 11001 (MSB) in binario.

Stesura informale dell'algoritmo: (lasciata come esercizio)

```
#include <iostream.h>/*inclusione della libreria standard       */
#include <stdlib.h>
int main( )
{
 int n;        /* contatore                                      */
 int Q;        /* variabile per il numero letto da tastiera      */
 /* ciclo di controllo per l'immissione dati che garantisce Q>0 */
 do {
   cout << "\nInserisci un numero positivo Q: ";
   cin >> Q;
 }while (Q <= 0);

 cout << "\nIn binario: ";
 n = 0;
 do {
   n = n+1;
   cout << (Q % 2);
   Q = Q / 2;
 } while (Q != 0);

 cout << "\nnumero di bit n = " << n;
 system("PAUSE");
 return 0;
}
```

ESERCIZIO 2 **(Approfondimento Costrutti iterativi + uso di contatori + uso accumulatori + Algoritmi)**

Dato un numero positivo Q, scrivere la sua rappresentazione in binario naturale, indicando anche il minimo numero di bit utilizzato. Il programma dovrà esibire un comportamento come nell'esempio seguente:

Input: 19 in decimale, Output: con 5 bit = 10011 in binario.

1ma idea di soluzione

```
#include <iostream.h>
#include <stdlib.h>
int main () {
   unsigned int Q, Qaux, current, i, n;
   cout << "\n Numero intero Q = "; cin >> Q;
   Qaux = Q;
   n = 0;
   do {
       Qaux = Qaux / 2;
       n++;
   } while (Qaux != 0);
   cout << "\n Codifica di Q = " << Q;
   cout << "con " << n << " bit = " ;
   for (current = n; current >0; current--) {
      Qaux = Q;
      for (i=1; i < current; i++)
         Qaux = Qaux / 2;
      cout << Qaux % 2;
   }
   system("PAUSE");
   return 0;
}
```

2da: Idea di soluzione:

```
#include <iostream.h>
#include <stdlib.h>
int main() {
   unsigned int dec, Q, n,pot;
   cout << "\n Numero intero Q > 0  = "; cin >> Q;
   dec = Q%2; Q = Q/2; n=1; pot = 1;
   while (Q > 0){
       pot = pot *10;
       dec = dec + (Q%2)*pot;
       cout << "\n dec = " << dec;
       Q = Q / 2 ;
       n = n + 1 ;
   }
   cout << "\nNum. decimale che emula la sequenza di bit voluta:";
   cout << dec << "\nlunghezza in bit = " << n;
   system("PAUSE");
   return 0;
}
```

3za: Idea di soluzione: se $Q = q_{n-1}2^{n-1}+q_{n-2}2^{n-2}+...+q_1 2+q_0$,
posso confrontare Q con le successive potenze di 2: $1,2,2^2,2^4,...$ finché risulta $Q >= 2^j$. Quando si verificherà la condizione per cui $Q < 2^n$, l'esponente di tale potenza sarà proprio il numero di bit necessario a rappresentare Q.
Osservando che:

$q_{n-1} = 1$ perché per costruzione $Q >= 2^{n-1}$, rappresentato su n bit

$q_{n-2} = 1$ sse $(Q - q_{n-1}2^{n-1}) >= 2^{n-2}$ altrimenti $q_{n-2} = 0$

$q_{n-3} = 1$ sse $(Q - q_{n-1}2^{n-1} - q_{n-2}2^{n-2}) >= 2^{n-3}$ altrimenti $q_{n-3} = 0$

.....

$q_0 = 1$ sse $(Q-q_{n-1}2^{n-1} - q_{n-2}2^{n-2}-...- q_1 2^1) >= 1$ altrimenti $q_0 = 0$

la stesura informale dell'algoritmo risolutore del problema proposto, può scriversi come segue:

1. inizio programma
2. leggi il numero intero *Q* da convertire
3. inizializza un contatore *n* al valore *0*
4. inizializza un accumulatore *d*, per le potenze di *2*, al valore *1*
5. Finché (il numero da convertire è >= *d*) esegui
   5.1. incrementa di uno il contatore n
   5.2. moltiplica per *2* la variabile *d*
   5.3. torna a 5.
6. stampa a video il valore della variabile *n*
7. inizializza un contatore *i* con *n-1*
8. Finché (*i >=0*) esegui
   8.1. dividi per *2* la variabile *d*
   8.2. se (*Q >= d*) allora
      8.2.1. stampa a video un carattere "1"
      8.2.2. assegna a *Q* il valore *Q-d*
   8.3. altrimenti stampa a video il carattere "0"
   8.4. decrementa di *1* il contatore *i*
   8.5. torna a 4.
9. Fine programma

```
#include <iostream.h> /* inclusione della libreria standard     */
#include <stdlib.h>
int main( )
{
 int i,d,n;    /* contatore                                      */
 int Q;        /* variabile per il numero letto da tastiera      */

 /* ciclo di controllo per l'immissione dati che garantisce Q>0 */
 do
 {
   cout << "\n Inserisci un numero positivo Q: ";
   cin >> Q;

 }while (Q <= 0);

 n = 0;
 d = 1;
 while (Q >= d)
 {
   n = n+1;
   d = d*2;
 }
 /* d contiene la piu' piccola potenza di 2 maggiore di Q */
 /* n ha il valore dell'esponente della potenza di 2      */
 /*   memorizzata in d e coincide con il numero di bit    */
 /*   minimo per rappresentare Q.                         */

 /* n-1 è il                                               */
 /* valore dell’esponente più significativo presente nella */
 /* rappresentazione binaria del numero Q.                 */
 /* Infatti Q = 2^(n-1) + …                                */

 cout << "\n” << Q << “ in decimale, con “ << n << “ bit = ",Q,n;
 i = n-1;
 while(i >= 0)
 {
    d = d / 2;
    if (Q >= d)
    {
      cout << "1";
      Q = Q - d;
    }
    else
    {
      cout << "0";
    }
    i = i-1;
  }
  system(“PAUSE”);
  return 0;
}
```

ESERCIZIO 3 (Manipolazione di caratteri)

Si scriva un programma in linguaggio C che risolva il problema seguente.
Ricevere dallo standard input due caratteri alfabetici, convertirli in maiuscolo e stampare a video ordinatamente tutti i caratteri dell'alfabeto fra essi compresi, estremi inclusi.
Esempio: dati 'g' e 'M' stampa a video la sequenza: GHIJKLM.

Stesura informale dell'algoritmo

Ricordando che nella tabella ASCII si hanno le seguenti corrispondenze tra i caratteri alfanumerici e le loro codifiche:

'0' - '9' : 48 - 57
'A' - 'Z' : 65 - 90
'a' - 'z' : 97 - 122

seguendo quanto richiesto nella traccia del problema è possibile formulare la soluzione come segue:

1. inizio programma
2. leggere due caratteri alfabetici in due variabili x, y
3. converte in maiuscolo i caratteri inseriti
4. se ($x > y$) allora scambia il valore delle due variabili
5. inizializza un contatore i con il valore del codice ASCII del carattere x
6. finché il contatore è minore o uguale al codice ASCII del carattere y
   6.1 stampa il carattere con codice ASCII uguale al valore del contatore
7. fine programma

```
#include <iostream.h> /* inclusione della libreria standard     */
#include <stdlib.h>
int main( )
{           /* dichiarazione variabili                        */
 char x,y;  /* variabili per memorizzare i dati in ingresso */
 char t;    /* variabile ausiliaria utilizzata per scambiare*/
            /*           i valori di due variabili.          */
 int  i;    /* contatore                                      */

 /* ciclo per il controllo dell'acquisizione dati           */
 do
 {
   cout << “\n Inserisci il carattere alfabetico x: ";
   cin >> x;

 }while (! ((x >= 'a')&&(x <= 'z') || (x >= 'A')&&(x <= 'Z')) );

 /* converte il carattere x nel corrispondente maiuscolo    */
 if ((x >= 'a')&&(x <= 'z')) x = x - ('a'-'A');

 /* ciclo per il controllo dell'acquisizione dati */
 do
 {
   cout << “\n Inserisci il carattere alfabetico y: ";
   cin >> y;

 }while (!(((y >= 'a')&&(y <= 'z'))||((y >= 'A')&&(y <= 'Z'))));

 /* converte il carattere y nel corrispondente maiuscolo    */
 if ((y >= 'a')&&(y <= 'z'))  y = y - ('a'-'A');

 /* ordina i due caratteri x,y in ordine crescente          */
 if (x > y)
 {
    t = x;
    x = y;
    y = t;
 }
cout << "\n La sequenza di caratteri richiesta e': ";

 i = (int) x;
 while (i <= (int) y)
 {
    cout << (char)i;
    i = i+1;
 }
 system(“PAUSE”);
 return 0;
}
```

ESERCIZIO 4 (Manipolazione caratteri)

Si scriva un programma in linguaggio C che risolva il problema seguente. Leggere una sequenza di caratteri alfanumerici dallo standard input terminata dal carattere ‘0’; quindi stampare sullo standard output la media dei numeri interi corrispondenti ai caratteri numerici inseriti.

Esempio:

input: ‘A’ ‘b’ ‘C’ ‘1’ ‘F’ ‘4’ ‘G’ ‘T’ ‘6’ ‘Y’ ‘6’ ‘3’ ‘s’ ‘0’
output: media = 4

Soluzione:

```
#include <iostream.h> /* inclusione della libreria standard     */
#include <stdlib.h>
int main( )
{           /* dichiarazione variabili                        */
 char  c;
 int accumulator;
 int  count;
 float mean;

 /* ciclo per il controllo dell'acquisizione dati            */
 accumulator = 0;
 count = 0;
 do
 {
   cout << "\n Inserisci un carattere: ";
   cin >> c;
   if ((c >= '1')&&(c <= '9')) {
      accumulator = accumulator + ((int)c – 48);
      count = count + 1;
   }

 } while (c!=’0’);
 if (count > 0) {
    mean = ((float)accumulator) / ((float)count);
    cout << "\n media = " << mean;
 }
 else
   cout << "\n non ci sono caratteri numerici!";

 system(“PAUSE”);
 return 0;
} /* end main */
```

ESEMPI DI PROGRAMMI IN C USANDO GLI ARRAY

ESERCIZIO 1

Realizzare un programma che legga 10 caratteri, uno alla volta, memorizzandoli progressivamente in un array di caratteri da 10 posizioni. Una volta terminata la sequenza di introduzione, il programma deve stampare in ordine inverso, uno alla volta e di seguito, i caratteri introdotti, dopodiché termina.

```
#include <iostream.h>
#include <stdlib.h>
#define NCAR 10

int main()
{
  int i;
  char c, parola[NCAR];


  for (i = 0; i < NCAR; i++)
  {
    cout << "\nInserire il carattere n. “ << i + 1 << “ “;
    cin >> parola[i];
  }

  cout << endl << endl ;
  for (i = NCAR - 1; i >=0; i--)
    cout << parola[i];

  system(“PAUSE”);
  cout << "\n\nFINE ELABORAZIONE --------------------------\n\n";
  return 0;
}
```

ESERCIZIO 2

Realizzare un programma che legga una stringa lunga al massimo 100 caratteri. Una volta terminata l'introduzione, il programma deve stampare la lunghezza della stringa, il numero di vocali contenute nella stringa e la stringa stessa al contrario.

```
#include <iostream.h>
#include <stdlib.h>
#include <string.h>
#define MAXLEN 100 + 1
// evidenzio il fatto che per contenere una stringa di 100 caratteri
// una array di char deve avere 101 elementi in modo da accogliere anche
// il carattere di terminazione della stringa ("\0")

int main()
{
  int i, len, nvoc;
  char frase[MAXLEN];

  cout << "\nInserire una frase (massimo 100 caratteri):\n";
  cin.getline(frase,MAXLEN,'\n'); // converte il '\n' in '\0' automaticamente
  len = strlen(frase);
  //cin.getline ( char buffer [], int length, char delimiter = ' \n');

  nvoc = 0;
  for (i = 0; i < len; i++)
  {
    switch (frase[i])//è certamente possibile utilizzare una istruzione if,
    {// sebbene la sua espressione condizionale risulti relativamente complessa.
        case 'a':// Provare a implementare questa parte con una istruzione if
        case 'A':
        case 'e':
        case 'E':
        case 'i':
        case 'I':
        case 'o':
        case 'O':
        case 'u':
        case 'U': nvoc++;
                  break;
     }
  }
  cout << "\n\nLa frase e' lunga " << len << "caratteri.";
  cout << "\nLa frase contiene " << nvoc <<" vocali.";
  cout << "\nLa frase invertita e':\n";

  // In questa versione la stringa viene semplicemente stampata al contrario.
  // Una soluzione più interessante prevede l'inversione in loco della stringa
  // e la sua successiva stampa tramite una semplice cout << frase;
  // Si lascia come esercizio l'individuazione di un algoritmo che inverta sul
  // posto la stringa contenuta nell'array frase[MAXLEN].

  for (i = len - 1; i >= 0; i--)
        cout << frase[i];

  system("PAUSE");
  cout << "\n\nFINE ELABORAZIONE -------------------------\n\n";
  return 0;
}
```

ESERCIZIO 3

Scrivere un programma in linguaggio C che risolva il problema seguente.
Leggere un polinomio di grado n a coefficienti reali e valutarlo in un dato punto x.

Idea di soluzione:
Il primo modo che sicuramente si può individuare è quello di sostituire ad x un certo valore, valutare le diverse potenze di x e moltiplicare per il corrispettivo coefficiente accumulando man mano il risultato in una variabile.

Si suppone di leggere i coefficienti del polinomio ordinatamente da quello di grado massimo a quello di grado nullo nelle celle di un vettore a partire dalla prima (cella 0).

Es: n = 3, $f(x) = 3x^3 + 5x^2 + 2x + 1$ lo rappresento come V = < 3, 5, 2, 1>

Iniziando a leggere il polinomio dal termine noto, quindi dall'ultimo elemento del vettore (avrà n+1 elementi) cioè dalla posizione n-esima, si ha:

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = (a_n x^n + (a_{n-1}x^{n-1} + \ldots + (a_1 x + (a_0))\ldots))$$

es: $f(x) = 3x^3 + 5x^2 + 2x + 1 = ((((1) + 2x) + 5x^2) + 3x^3)$

Al variare di un indice tra 0 e il grado del polinomio, si utilizzeranno due variabili accumulatore:

- Una, per costruire man mano il valore della potenza di x opportuna
- l'altra usata per sommare man mano i valori parziali della f

1. legge n e il vettore V dallo standard input
2. legge x dallo standard input
3. f = 0;
4. pot = 1;
5. i = n;
6. finché (i >= 0)
   6.1. f = V[i]*pot + f;
   6.2. pot = pot * x
   6.3. i = i - 1;

```
#include <iostream.h>  /* inclusione della libreria standard */
#include <stdlib.h>
#define MAX_LEN 100 /* dichiarazione di costante simbolica */

int main ( )
{
 float V[MAX_LEN];
 float pot,f,x;
 int i,n;

 do {

  cout << "\ninserire il grado del polinomio, n = ";
  cin >> n;

 }while ((n <= 0)||(n >= MAX_LEN));

 cout<<"\ninserire i coeff. del pol. V_" << n << " … V_0:  \n";
 i = 0;
 do{

   cin >> V[i];
   i++;

 }while (i<=n);

cout << "\ninserire il valore in cui valutare il polinomio,x =");
cin >> x;


 f = 0;
 pot = 1;
 for (i = n; i >=0; i--) {

    f = f + V[i]*pot;

    pot = pot * x;
 }

 cout << "\n f(" << x << ") = " << f << ".\n ";
 system("PAUSE") ;
 return 0;
}
```

Un secondo metodo, più efficiente, che utilizza un minor numero di moltiplicazioni, consiste nell' effettuare la valutazione secondo la seguente espressione:

$$a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 + a_0 = ((((0*x+a_n)x + a_{n-1})x + a_{n-2})x + \ldots + a_1)x + a_0$$

$$3x^3 + 5x^2 + 2x + 1 = ((0x+3)x + 5)x + 2)x + 1$$

Nota: è così necessaria una sola variabile per sommare i risultati parziali, e si fanno esattamente n+1 moltiplicazioni (una per ogni coefficiente del polinomio).

1. legge n e il vettore V dallo standard input
2. legge x dallo standard input
3. f = 0;
4. i = 0;
5. finché (i <=n)
   5.1 f = f*x + V[i];
   5.1 i = i + 1;

```
#include <iostream.h>  /* inclusione della libreria standard   */
#include <stdlib.h>
#define MAX_LEN 100 /* dichiarazione di costante simbolica  */

int main ( )
{
 float V[MAX_LEN];
 float f,x;
 int i,n;

 do {

  cout << “\ninserire il grado del polinomio, n = ”;
  cin >> n;

 }while ((n <= 0)||(n >= MAX_LEN));

 cout<<“\ninserire i coeff. del pol. V_“ << n << “ … V_0: \n”;
 i = 0;
 do{

   cin >> V[i];
   i++;

 }while (i<=n);


cout << “\ninserire il valore in cui valutare il polinomio,x = ”;
cin >> x;

 f = 0;
 for (i = 0; i <=n; i++) {

    f = f*x + V[i];

 }


 cout << “\n f(“ << x << “) = “ << f << “.\n ”;
 system(“PAUSE“) ;
 return 0;
}
```