Introduzione al C

Corso di Fondamenti di Informatica

Ingegneria Clinica

Esercitazione 9

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Raffaele Nicolussi

Esercizio :: DNA

- L'informazione genetica del DNA, é codificata nella sequenza di basi (adenina, guanina, citosina e timina) che lo formano.
- Per convenzione, sequenze di DNA sono rappresentate come liste di lettere 'A','G','C', 'T'.
- Vogliamo analizza sequenze di questo tipo, di lunghezza fissata DIM, che rappresentiamo come array di caratteri 'A','G','C', 'T'

Inizializzare l'array nel main

A[]={'A' ,'G', 'T', 'A', 'C', 'A', 'T', 'G', 'T', 'A'}

int DIM ***= 10***

DNA

1. Scrivere un programma che stampa quante volte ciascun carattere è presente
2. Scrivere un programma che, dato un array di caratteri ‘A’,’G’,’C’, ‘T’, elimina dall’array *la prima occorrenza di ‘A’, e stampa l’array* risultante.
3. Scrivere un programma che, dato un array di caratteri ‘A’,’G’,’C’, ‘T’, elimina dall’array *tutte le occorrenze di ‘A’, e stampa l’array* risultante.
4. Trasformare i programmi in funzioni

DNA

- **Da pensare**: cosa significa eliminare dall'array?
- **Attenzione**: non vogliamo lasciare "vuoti"...
- **Sugg**: la dimensione dell'array é fissa, ma possiamo tener conto, in una variabile dedicata, del numero di elementi significativi (la lunghezza che ci interessa), oppure del livello di riempimento (ultimo indice significativo).

DNA

- **Algoritmo semplice:**

1. *creo un secondo array di appoggio, B*
2. *scorro il primo e copio nel secondo solo gli elementi che mi interessano*
   - *Attenzione: avrò bisogno di un **indice_array1** e **indice_array2***
     - ***indice_array1 :: scorre A***
     - ***indice_array2 :: scorre B***
3. *Stampo l'array ottenuto*

DNA

- Algoritmo in versione **tosta**:

1. Fino a che non è finito A (ciclo su indice **i**)
2. A[**i**] va eliminato? (if)
    1. **SI**
        a. **sposta** tutto il resto dell'array **A** (da **i+1** fino a **DIM-1**) copiandolo nelle posizioni da **i** a **DIM-2**
        b. **DIM--;**
    2. **NO**
        1. non fare nulla (**if** senza **else**)

3. **sposta** ::
    1. for (k=i; k<DIM-1; k++)
        A[K] = A[K+1]

Rappresentazione binaria

Scrivere un programma che, dopo aver letto da standard input un intero positivo N, calcola la rappresentazione binaria di N. Per memorizzare le cifre binarie si utilizzi un array di 16 elementi. Il programma deve stampare il valore ottenuto se N è rappresentabile con 16 cifre binarie, altrimenti stampa un messaggio.

Rappresentazione binaria

```
Inserire un intero positivo: 23
23 espresso in binario = 10111
Premere un tasto per continuare . . .
```

```
Inserire un intero positivo: 65535
65535 espresso in binario = 1111111111111111
Premere un tasto per continuare . . .
```

```
Inserire un intero positivo: 65536
Il numero 65536 non e' rappresentabile con 16 bit
Premere un tasto per continuare . . .
```

Soluzione

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

#define NUM_BIT 16

int main( )
{
 unsigned short int vetbit[ NUM_BIT ] = {0};
 unsigned long int numero, quoziente;
 short int i;


 printf( "Inserire un intero positivo: " ) ;
 scanf("%ld", &numero);
 i=NUM_BIT - 1; /* si memorizza a partire dall'ultimo */
 quoziente = numero;
```

Soluzione

```
while (quoziente >= 1 && i >= 0) {
        vetbit[i] = quoziente % 2;
        quoziente = quoziente / 2;
        i--;
 }
 if (quoziente >= 1 )
    printf ("Il numero %ld non e' rappresentabile con 16 bit\n", numero);
 else {
    printf ("%ld espresso in binario = ", numero);
    for (i=i+1; i< NUM_BIT; i++)
        printf ("%hu", vetbit[i]);
    printf ("\n");
 }

 system( "PAUSE" ) ;
 return 0 ;
}
```

Esercizio calc array (1/2)

Progettare una funzione che, ricevuti:

1. un carattere ch
2. un array A di reali
3. un array B di reali
4. un array C di reali
5. un intero d indicante la dimensione di A,B,C,

interpreti ch come l'operazione da eseguire tra A[k] e B[k] (k=0,1,..,d-1) per produrre il valore da assegnare a C[k]. Le operazioni valide sono: somma ('+'), differenza ('-'), prodotto ('*'), e divisione ('/'). In caso di operatore valido, la funzione restituirà un carattere indicante l'operazione effettuata. In caso di operatore non valido, la funzione restituirà il carattere '?' e non effettuerà alcuna operazione tra gli elementi di A e B

Esercizio calc array(2/2)

ESEMPI

Se ch='+', d=4, *A* = [0.0, 1.0, 0.1, 1.1], B = [0.0, 1.0, 0.2, -1.7], verrà restituito '+' e sarà *C* = [0.0, 2.0, 0.3, -0.6]

Se ch='%', d=4, *A* = [0.0, 1.0, 0.1, 1.1], B = [0.0, 1.0, 0.2, -1.7], verrà restituito '?' e sarà *C* = [-, -, -, -]

Inserire la funzione all'interno di un programma contenente quanto necessario per verificare se il comportamento della funzione è corretto

Soluzione

- La soluzione proposta definisce 25 come massima dimensione array.
- E' possibile configurare la dimensione degli array A, B e C ed il contenuto degli array A e B.
- Dopo l'acquisizione del carattere che rappresenta l'operazione da effettuare sugli elementi degli array, si acquisisce la dimensione degli array e i valori degli elementi degli array A e B (funzione "arrayscan").
- Alla fine viene visualizzato un messaggio che segnala se il carattere inserito rappresenta un operatore valido e, in caso affermativo, viene stampato il contenuto dell'array C (funzione "arrayprint")

Soluzione

```
/* definizione funzione arrayscan */
void arrayscan( float v[ ] , unsigned int size )
{
    unsigned int i ;
    /* ciclo acquisizione elementi array */
    for( i = 0 ; i < size ; i++ )
    {
        printf( "Inserire elemento int di indice %u\n" , i ) ;
        scanf( "%f" , &v[ i ] ) ;
    }
}
```

Soluzione

```
/* definizione funzione arrayprint */
void arrayprint( float v[ ] , unsigned int size )
{
    unsigned int i ;

    /* ciclo stampa elementi array */
    for( i = 0 ; i < size ; i++ )
        printf( "%u\t%f\n" , i , v[ i ] ) ;
}
```

Soluzione

```
char arrayoper (char op, float v1[], float v2[], float v3[], int d)
{
   int i;

   switch (op) {
      case '+': for (i = 0; i < d; i++)  v3[i] = v1[i] + v2[i]; break;
      case '-': for (i = 0; i < d; i++)  v3[i] = v1[i] - v2[i]; break;
      case '*': for (i = 0; i < d; i++)  v3[i] = v1[i] * v2[i]; break;
      case '/': for (i = 0; i < d; i++)  v3[i] = v1[i] / v2[i]; break;
      default: return '?';
   }

   return op;
}
```

Soluzione

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

#define DIM 25

/* prototipo funzione arrayoper */
char  arrayoper(char op, float v1[], float v2[], float v3[], int d);

/* prototipo funzione arrayscan */
void arrayscan( float v[ ] , unsigned int size ) ;

/* prototipo funzione arrayprint */
void arrayprint( float v[ ] , unsigned int size ) ;
```

Soluzione

```
int main()
{
   float vA[DIM], vB[DIM], vC[DIM];
   unsigned int dim;
   char oper;


   printf("Inserire operatore (carattere): ");
   scanf("%c", &oper);

   printf( "\nInserire dimensione comune (max %u) array A e B:\n" , DIM ) ;
   scanf( "%u" , &dim ) ;

   printf( "\nAcquisizione elementi array A:\n" ) ;
   arrayscan( vA , dim ) ;

   printf( "\nAcquisizione elementi array B:\n" ) ;
   arrayscan( vB , dim ) ;
```

Soluzione

```
  if ( arrayoper(oper, vA, vB, vC, dim) != '?') {
      printf("Operatore  valido\n");
      printf("Stampa elementi array C:\n");
      arrayprint(vC, dim);
    }
    else
      printf ("Operatore non valido\n");

    system("PAUSE");
    return 0;
}
```

Esercizio Matrix (1/2)

Progettare una funzione che, ricevuti:

1. un array bidimensionale *A* di mxn interi (m righe, n colonne)
2. un array B di n interi
3. un array *C* di m interi
4. i due interi m e n,

assegni, per k = 0, 1, …, m-1, all'elemento C[k] il valore $\sum_i$ A[k][i] * B[i], i = 0, .., n-1

Dichiarare e definire la funzione specificando il massimo valore di m ed n usando la costante MAX_D

Esercizio Matrix (2/2)

ESEMPIO:

Se m=2, n=3,
A = [[1, 2, 3], [4, 5, 6]]
B = [2, 1, 0],
allora *C* = [4, 13]

Inserire la funzione all'interno di un programma contenente quanto necessario per verificare se il comportamento della funzione è corretto
Definire, tramite direttiva al compilatore, il valore 25 per la costante MAX_D

Soluzione

- La soluzione proposta definisce 25 come massima dimensione della matrice e degli array.
- E' possibile configurare le dimensioni della matrice A e di conseguenza degli array V1 e V2.
- Dopo l'acquisizione dei valori contenuti nella matrice (funzione "matrixscan") e dei valori contenuti nell'array (funzione "arrayscan") viene visualizzato il contenuto del vettore V2 dopo l'attivazione della funzione "matrixpervect“.

Soluzione

```
/* definizione funzione arrayscan */
void arrayscan( int v[ ] , unsigned int size )
{
    unsigned int i ;
    /* ciclo acquisizione elementi array */
    for( i = 0 ; i < size ; i++ )
    {
        printf( "Inserire elemento int di indice %u\n" , i ) ;
        scanf( "%d" , &v[ i ] ) ;
                                                    }
}
```

Soluzione

```
/* definizione funzione arrayprint */
void arrayprint( int v[ ] , unsigned int size )
{
    unsigned int i ;

    /* ciclo stampa elementi array */
    for( i = 0 ; i < size ; i++ )
        printf( "%u\t%d\n" , i , v[ i ] ) ;
}
```

Soluzione

```
/* definizione funzione matrixscan */
void matrixscan(int m[][DIM_MAX], unsigned int r, unsigned
     int c)
{
   unsigned int i, j;

   for (i = 0; i < r; i++)
      for (j = 0; j < c; j++)
      {
         printf( "Inserire elemento int di indice %u,%u\n" , i, j ) ;
         scanf("%d", &m[i][j]);

      }
   return;
}
```

Soluzione

```
/* definizione funzione matrixpervect */
void matrixpervect (int m[][DIM_MAX], int v1[], int v2[],
     unsigned int r, unsigned int c)
{
   unsigned int i, j;

   /* inizializzazione del vettore v2 */
   for (i = 0; i < r; i++)
      v2[i] = 0;

   for (i = 0; i < r; i++)
       for (j = 0; j < c; j++)
          v2[i] += m[i][j] * v1[j];

   return;
}
```

Soluzione

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

#define DIM_MAX 25

/* prototipo funzione matrixpervect */
void matrixpervect (int m[][DIM_MAX], int v1[], int v2[], unsigned int r,
     unsigned int c);

/* prototipo funzione matrixscan */
void matrixscan(int m[][DIM_MAX], unsigned int r, unsigned int c);

/* prototipo funzione arrayscan */
void arrayscan( int v[ ] , unsigned int size ) ;

/* prototipo funzione arrayprint */
void arrayprint( int v[ ] , unsigned int size ) ;
```

Soluzione

```
int main()
{
   int mat[DIM_MAX][DIM_MAX], vett1[DIM_MAX],
     vett2[DIM_MAX];
   unsigned int righe, colonne;

   printf( "\nInserire numero righe della matrice (max %u):\n" ,
     DIM_MAX ) ;
   scanf( "%u" , &righe ) ;

   printf( "\nInserire numero colonne della matrice (max %u):\n" ,
     DIM_MAX ) ;
   scanf( "%u" , &colonne ) ;

   printf( "\nAcquisizione elementi matrice :\n" ) ;
   matrixscan( mat , righe, colonne ) ;
```

Soluzione

```
  printf( "\nAcquisizione elementi vettore :\n" ) ;
   arrayscan( vett1 , colonne ) ;

   matrixpervect(mat, vett1, vett2, righe, colonne);

   printf( "\nStampa elementi array v2:\n" ) ;
   arrayprint( vett2 , righe ) ;

   system("PAUSE");
   return 0;
}
```

Per casa

- Dato un array A di interi scrivere le funzioni:
  - void carica_array (int ar[], int elem)
    - carica nell'array un numero di interi, chiesti all'utente, pari a elem
  - void stampa_array (int ar[], int elem)
    - Stampa elem elementi dell'array
  - int trova_max (int ar[] , int elem)
    - Trova e restituisce il valore massimo dell'array
  - float calcola_media (int ar[], int elem)
    - Calcola la media dei valori dell'array
  - int stampa_pari (int ar[], int elem)
    - Stampa gli elementi pari e restituisce il numero di elementi pari presenti nell'array

Per casa

- Dato anche un array B di interi di dimensioni uguali ad A scrivere le funzioni:
  - void somma_array (int A[], int B[], int elem)
    - Calcola A = A + B
  - int max_array (int A[], int B[], int elem)
    - Calcola il valore massimo tra i due array usando la funzione int trova_max (int ar[] , int elem)