ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fondamenti di Informatica T

Esercizi extra per la pausa pasquale

Esercizio 1: Array

1. Si realizzi una funzione che legga da input una sequenza di numeri interi, di lunghezza non nota a priori (al più 10), e terminata da 0
   - Due parametri: un array di interi, un intero (dimensione fisica)
   - Valore di ritorno: un intero (quanti numeri sono stati letti)
2. Si realizzi una funzione che cerchi un elemento in un array di interi
   - Tre parametri: un array di interi, un intero (dimensione logica), un intero (elemento da cercare)
   - Valore di ritorno: intero (-1 se l'elemento non è presente nell'array, altrimenti il primo indice in cui è presente l'elemento cercato)
3. Si realizzi un main di prova che esegua i seguenti passi:
   1. Definizione di un array di interi
   2. Lettura dei valori (funzione 1)
   3. Lettura di un valore intero da terminale
   4. Ricerca del valore letto nell'array (funzione 2)
   5. Stampa del risultato della ricerca come stringa («Trovato in posizione x» o «Non trovato»)

Esercizio 1: Soluzione (1/2)

```
#include <stdio.h>

int leggi(int v[], int dim) {
  int i, num;
  i = 0;
  do {
    printf("Inserisci numero (posizione %d): ", i);
    scanf("%d", &num);
    if (num != 0) {
      v[i] = num;
      i++;
    }
  } while (num!=0 && i<dim);
  return i;
}

int trovaPos(int vet[], int dim, int el) {
  int i;
  for (i=0; i<dim; i++) {
    if (vet[i] == el)
      return i;
  }
  return -1;
}
```

Esercizio 1: Soluzione (2/2)

```
#define N 10

int main() {
  int dim_vett, val, pos;
  int vett[N];

  dim_vett = leggi(vett, N);

  printf("Inserisci un numero da cercare: ");
  scanf("%d", &val);

  pos = trovaPos(vett, dim_vett, val);

  if(pos == -1)
    printf("Non trovato\n");
  else
    printf("Trovato in posizione %d\n", pos);

  return 0;
}
```

Esercizio 2: Stringhe (1/3)

Uno dei più antichi sistemi di codificazione di messaggi segreti si basa sulla sostituzione, secondo un certo ordine, dei caratteri componenti il messaggio.

Ad esempio, dato un messaggio composto dalle lettere:

{a, b, c}

E data una chiave di sostituzione che, per ogni lettera ne associa un'altra:

a → x

b → y

c → z

Il messaggio originale può essere così riscritto:

{x, y, z}

Esercizio 2: Stringhe (2/3)

Si vuole costruire un sistema di codifica/decodifica di questo tipo, facendo le seguenti assunzioni:

1. Le lettere componenti il messaggio sono tutte minuscole, ed i messaggi non possono contenere altri caratteri che lettere (no spazi, no numeri, no punteggiatura, …)
2. Il codice di sostituzione è dato da un array di 26 caratteri, che viene interpretato nel seguente modo: nella posizione ad indice 0 vi è il carattere che deve sostituire la lettera ‘a’, in posizione con indice 1 vi è il carattere che deve sostituire la lettera ‘b’, etc.

Esercizio 2: Stringhe (3/3)

- Si strutturi la soluzione implementando due funzioni:

```
void crypt( char source[],
            int length,
            char code[DIM_ALPHA],
            char dest[]);
```

```
void decrypt( char source[],
              int length,
              char code[DIM_ALPHA],
              char dest[]);
```

- Si scriva un main di prova per invocarle

Esercizio 2: Soluzione (1/3)

```
#include <stdio.h>

#define DIM 16
#define DIM_ALPHA 26


void crypt(char source[], int length, char code[DIM_ALPHA], char dest[]) {
    int i;

    for (i=0; i<length; i++) {
      dest[i] = code[source[i] - 'a'];
    }
}
```

Esercizio 2: Soluzione (2/3)

```
void decrypt(char source[], int length, char code[DIM_ALPHA], char dest[])
{
    int i, j;
    int pos = -1;

    for (i=0; i<length; i++) {
        for (j=0; j<DIM_ALPHA && pos<0; j++) {
            if(source[i] == code[j])
                pos = j;
        }
        dest[i] = 'a' + pos;
        pos = -1;
    }
}
```

Esercizio 2: Soluzione (3/3)

```
int main()
{
    char source[DIM] = "civediamodomani";
    char codice[DIM_ALPHA] = "mvgadpliobrjxteqysuwcznhkf";
    char dest1[DIM];
    char dest2[DIM];

    printf("ORIGINALE: %s\n", source);

    crypt(source, DIM-1, codice, dest1);
    dest1[DIM-1] = '\0'; // Aggiungo il terminatore per la stampa
    printf("CRIPTATO: %s\n", dest1);

    decrypt(dest1, DIM-1, codice, dest2);
    dest2[DIM-1] = '\0'; // Aggiungo il terminatore per la stampa
    printf("DE-CRIPTATO: %s\n", dest2);

    return 0;
}
```

Esercizio 3: Strutture (1/3)

Si vuole implementare un programma per il **calcolo dell'inflazione su determinati prodotti commerciali**.

A tal scopo ogni prodotto è rappresentato tramite una **struttura *item***, definita da una stringa *name* con il nome del prodotto, e da due float *old_price* e *new_price* rappresentanti i prezzi.

Esercizio 3: Strutture (2/3)

1. Si scriva una funzione **lettura()** che riceva come parametri di ingresso un vettore *prezzi* di strutture *item*, la dimensione fisica *max* del vettore prezzi, e un puntatore a intero *num* che rappresenta la dimensione logica del vettore
   - La funzione deve leggere da standard input il nome del prodotto ed i due prezzi, e deve copiare tale informazione nella prima posizione libera nel vettore *prezzi*
   - La funzione deve terminare se l'utente inserisce come nome del prodotto il termine "fine", oppure se viene raggiunta la dimensione fisica del vettore
   - La dimensione logica del vettore *prezzi* così riempito deve essere restituita tramite il parametro *num* (passato appunto per riferimento). Al termine della lettura dei dati la funzione deve restituire il valore 0

Esercizio 3: Strutture (3/3)

2. Si scriva una funzione **main()** che esegua i seguenti passi:
   - definire un array di strutture item (di dimensione massima MAX_ITEM)
   - invocare la funzione **lettura()** per riempire tale array
   - stampare a video nome e tasso d'inflazione per ogni prodotto, utilizzando la formula:

$$infl_i = \left(\frac{new\_price_i}{old\_price_i} - 1 \right) * 100$$

Esercizio 3: Soluzione (1/3)

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

#define DIM 21
#define MAX_ITEM 100

typedef struct {
        char name[DIM];
        float old_price;
        float new_price;
} item;
```

Esercizio 3: Soluzione (2/3)

```
int lettura (item prezzi[], int max, int *num) {
    char name[DIM];
    *num = 0;

    printf("Inserire nome prodotto: ");
    scanf("%s", name);

    while ((strcmp(name, "fine")) && (*num < max)) {
        strcpy(prezzi[*num].name, name);
        printf("Inserire old price: ");
        scanf("%f", &prezzi[*num].old_price);
        printf("Inserire new price: ");
        scanf("%f%*c", &prezzi[*num].new_price);
        (*num)++;
        printf("Inserire nome prodotto: ");
        scanf("%s", name);
    }

    return 0;
}
```

Esercizio 3: Soluzione (3/3)

```
int main() {
    item V[MAX_ITEM];
    int num, i, result;
    float infl;

    result = lettura(V, MAX_ITEM, &num);

    if (result!=0) {
        printf("Problemi durante la lettura...\n");
        return -1;
    }

     for (i=0; i < num; i++) {
         infl = (V[i].new_price/V[i].old_price -1)*100;
         printf("Inflazione del prodotto %s: %6.2f%%\n", V[i].name, infl);
     }

     return 0;
}
```