Esercizi con Array e Funzioni

Filippo Cugini

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

a) Un array può memorizzare molti tipi differenti di valori

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

a) Un array può memorizzare molti tipi differenti di valori

Falso. Un array può memorizzare solo valori dello stesso tipo

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

b) L’indice di un array può essere del tipo di dati `double`

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

b) L’indice di un array può essere del tipo di dati `double`

Falso. L’indice di un array deve essere un intero o un’espressione intera

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

c) Se in una lista di inizializzatori vi sono meno inizializzatori del numero degli elementi nell'array, il C automaticamente inizializza i restanti elementi all'ultimo valore nella lista degli inizializzatori

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

c) Se in una lista di inizializzatori vi sono meno inizializzatori del numero degli elementi nell'array, il C automaticamente inizializza i restanti elementi all'ultimo valore nella lista degli inizializzatori

Falso. Il C inizializza automaticamente i restanti elementi a zero

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

d) È un errore se una lista di inizializzatori contiene più inizializzatori di quanti sono gli elementi nell’array

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

d) È un errore se una lista di inizializzatori contiene più inizializzatori di quanti sono gli elementi nell'array

Vero

Revisione concetti di base

Stabilite se vero o falso. Se falso, spiegate il motivo:

e) L'elemento individuale di un array che viene passato a una funzione come un argomento della forma `a[i]` e viene modificato nella funzione chiamata conterrà poi il valore modificato nella funzione chiamante

Falso. Gli elementi individuali di un array sono passati per valore. Se invece è passato l'intero array a una funzione, allora le modifiche agli elementi saranno riflesse nell'originale

Revisione concetti di base

Eseguite l’istruzione riguardante un array chiamato `fractions`:

a) Definite una costante simbolica `SIZE` da sostituire con il testo di sostituzione 10

Eseguite l’istruzione riguardante un array chiamato `fractions`:

a) Definite una costante simbolica `SIZE` da sostituire con il testo di sostituzione 10

```
#define SIZE 10
```

Revisione concetti di base

Eseguite l’istruzione riguardante un array chiamato `fractions`:

b) Definite un array con un numero uguale a `SIZE` di elementi di tipo `double` e inizializzate gli elementi a 0

Eseguite l'istruzione riguardante un array chiamato `fractions`:

b) Definite un array con un numero uguale a `SIZE` di elementi di tipo `double` e inizializzate gli elementi a 0

```
double fractions[SIZE] = { 0.0 };
```

Revisione concetti di base

Eseguite l’istruzione riguardante un array chiamato `fractions`:

c) Assegnate il valore 1.667 all’elemento nove dell’array

Revisione concetti di base

Eseguite l'istruzione riguardante un array chiamato `fractions`:

c) Assegnate il valore 1.667 all'elemento nove dell'array

```
fractions[9] = 1.667;
```

Revisione concetti di base

Eseguite l’istruzione riguardante un array chiamato `fractions`:

d) Stampate gli elementi 6 e 9 dell’array con due cifre di precisione alla destra del punto decimale e mostrate l’output che viene stampato sullo schermo

Revisione concetti di base

Eseguite l'istruzione riguardante un array chiamato `fractions`:

d) Stampate gli elementi 6 e 9 dell'array con due cifre di precisione alla destra del punto decimale e mostrate l'output che viene stampato sullo schermo

```
printf("Primo:%.2f\nSecondo:%.2f\n",
       fractions[6], fractions[9]);
```

Eseguite l'istruzione riguardante un array chiamato `fractions`:

e) Stampate tutti gli elementi dell'array usando un'istruzione di iterazione `for`. Supponete che la variabile intera `x` sia stata definita come una variabile di controllo per il ciclo

Revisione concetti di base

Eseguite l'istruzione riguardante un array chiamato `fractions`:

e) Stampate tutti gli elementi dell'array usando un'istruzione di iterazione `for`. Supponete che la variabile intera `x` sia stata definita come una variabile di controllo per il ciclo

```
for (x=0; x<SIZE; x++) {
   printf("fractions[%d] = %f\n", x, fractions[x]);
}
```

Esercizi con array bidimensionali

Esercizi con array bidimensionali

Eseguite l’istruzione per effettuare le seguenti operazioni:

a) Definite `table` come un array intero con 3 righe e 3 colonne. Supponete che la costante simbolica `SIZE` sia stata definita come valore 3

Esercizi con array bidimensionali

Eseguite l'istruzione per effettuare le seguenti operazioni:

a) Definite `table` come un array intero con 3 righe e 3 colonne. Supponete che la costante simbolica `SIZE` sia stata definita come valore 3

```
int table[SIZE][SIZE];
```

Esercizi con array bidimensionali

Eseguite l'istruzione per effettuare le seguenti operazioni:

b) Quanti elementi contiene l'array `table`? Assegnate a ciascun elemento la somma dei sui indici

Esercizi con array bidimensionali

Eseguite l'istruzione per effettuare le seguenti operazioni:

b) Quanti elementi contiene l'array `table`? Assegnate a ciascun elemento la somma dei sui indici

L'array `table` contiene `SIZE*SIZE` elementi

```
for(x = 0; x < SIZE; x++) {
   for(y = 0; y < SIZE; y++) {
      table [x][y] = x + y;
   }
}
```

Esercizi con array bidimensionali

Eseguite l’istruzione per effettuare le seguenti operazioni:

c) Stampate ciascun elemento

Esercizi con array bidimensionali

Eseguite l'istruzione per effettuare le seguenti operazioni:

c) Stampate ciascun elemento

```
for(x = 0; x < SIZE; x++) {
   for(y = 0; y < SIZE; y++) {
      printf("table[%d][%d] = %d\n",
           x, y, table [x][y]);
   }
}
```

```
table[0][0] = 1
table[0][1] = 8
table[0][2] = 0
table[1][0] = 2
table[1][1] = 4
table[1][2] = 6
table[2][0] = 5
table[2][1] = 0
table[2][2] = 0
```

Esercizi con array bidimensionali

d) Presupponete:

```
int a[2][2] = { { 1, 2 }, { 3, 4 } };
```

E’ corretto scrivere la seguente istruzione?

```
int a[1, 1] = 5;
```

Esercizi con array bidimensionali

d) Presupponete:

```
int a[2][2] = { { 1, 2 }, { 3, 4 } };
```

E’ corretto scrivere la seguente istruzione?

```
int a[1, 1] = 5;
```

Errore: indicizzazione dell’array fatta in modo scorretto

Correzione: cambiate l’istruzione in:

```
int a[1][1] = 5;
```

Esercizi su ricorsione con array

Palindromi

Un palindromo è una stringa che si scrive e si legge allo stesso modo in avanti e all'indietro

Alcuni esempi di palindromi sono: "radar", "able was i ere i saw elba" e, se ignorate gli spazi, "a man a plan a canal panama"

Scrivete una funzione ricorsiva `testPalindrome` che restituisca 1 se la stringa memorizzata in un array è un palindromo e altrimenti 0

La funzione deve ignorare gli spazi e la punteggiatura nella stringa

Esercizi su ricorsione con array

```c
#include <stdio.h>
#define SIZE 80

// prototipo di funzione
int testPalindrome(char array[], int left, int right);

int main(void)
{
   char string[SIZE]; // stringa originale
   char copy[SIZE]; // copia della stringa senza spazi
   puts("Enter a sentence:");
   char c; // tiene temporaneamente l'input
   unsigned int count = 0; // lunghezza della stringa
   // ricevi la frase da testare dall'utente
   while ((c = getchar() ) != '\n' && count < SIZE) {
      string[count++] = c;
   }
```

Esercizi su ricorsione con array

```
    string[count] = '\0'; // termina la stringa
    unsigned int copyCount = 0;
    // crea una copia della stringa senza spazi
    for (unsigned int i = 0; string[i] != '\0'; ++i) {
       if (string[i] != ' ' && string[i] != ',' &&
             string[i] != '.' && string[i] != '!') {
          copy[copyCount++] = string[i];
       }
    }
    // stampa se la sentenza e' o non e' un palindromo
    if (testPalindrome(copy, 0, copyCount - 1)) {
       printf("\"%s\" is a palindrome\n", string);
    }
    else {
       printf("\"%s\" is not a palindrome\n", string);
    }
}
```

Esercizi su ricorsione con array

```
// funzione per vedere se la frase e' un palindromo
int testPalindrome(char array[], int left, int right)
{
   // testa l'array per vedere se e' un palindromo
   if (left == right || left > right) {
      return 1;
   }
   else if (array[left] != array[right]) {
      return 0;
   }
   else {
      return testPalindrome(array, left + 1, right - 1);
   }
}
```