Array et al.

17/10/2024

6.5. Trovate l'errore in ognuno dei seguenti segmenti di programma e correggetelo.

a)
```
#define SIZE 100;
```

b)
```
SIZE = 10;
```

c)
```
int b[10] = {0}; int i;
for (size_t i = 0; i <= 10; ++i) {
b[i] = 1; }
```

d)
```
#include <stdio.h>;
```

e)
```
int a[2][2] = {{1, 2}, {3, 4}};
a[1, 1] = 5;
```

f)
```
#define VALUE = 120
```

6.5. a) Errore: il punto e virgola alla fine della direttiva per il preprocessore `#define`.

Correzione: eliminate il punto e virgola.

b) Errore: assegnare un valore a una costante simbolica usando un'istruzione di assegnazione.
Correzione: assegnate un valore alla costante simbolica in una direttiva per il preprocessore `#define` senza usare l'operatore di assegnazione, come in `#define SIZE 10`.

c) Errore: fare riferimento a un elemento di un array al di fuori dei confini dell'array (`b[10]`).
Correzione: cambiate il valore finale della variabile di controllo a `9` o cambiate `<=` in `<`.

d) Errore: il punto e virgola alla fine della direttiva per il preprocessore `#include`.
Correzione: eliminate il punto e virgola.

e) Errore: indicizzazione dell'array fatta in modo scorretto.
Correzione: cambiate l'istruzione in `a[1][1] = 5;`

f) Errore: il valore di una costante simbolica non si definisce usando l'operatore `=`.
Correzione: cambiate la direttiva per il preprocessore in `#define VALUE 120`.

6.8. Scrivete istruzioni per effettuare ognuna delle seguenti operazioni.

a) Stampare il valore del settimo elemento di un array di caratteri `f`.
b) Inserire un valore nell'elemento 4 di un array `b` unidimensionale di elementi in virgola mobile.
c) Inizializzare a `8` ognuno dei cinque elementi di un array intero unidimensionale `g`.
d) Sommare gli elementi dell'array `c` di 100 elementi in virgola mobile.
e) Copiare l'array `a` nella prima porzione dell'array `b`. Supponete che `a` abbia 11 elementi, `b` ne abbia 34 ed entrambi abbiano lo stesso tipo di elementi.

```c
#include <stdio.h>

int main() {
    // a) Stampare il valore del settimo elemento di un array di caratteri f
    char f[] = "Esempio di array";  // Supponiamo un array di caratteri di esempio
    printf("Il settimo elemento di f è: %c\n", f[6]);  // Gli indici partono da 0, quindi il settimo elemento è f[6]

    // b) Inserire un valore nell'elemento 4 di un array b unidimensionale di elementi in virgola mobile
    float b[10];  // Supponiamo un array di 10 elementi
    b[3] = 5.5;  // L'indice 4 corrisponde a b[3] (gli indici partono da 0)

    // c) Inizializzare a 8 ognuno dei cinque elementi di un array intero unidimensionale g
    int g[5];
    for (int i = 0; i < 5; i++) {
        g[i] = 8;  // Inizializzo ogni elemento dell'array g a 8
    }

    // d) Sommare gli elementi dell'array c di 100 elementi in virgola mobile
    float c[100];  // Supponiamo un array di 100 elementi
    float somma = 0;
    for (int i = 0; i < 100; i++) {
        somma += c[i];  // Sommo ciascun elemento dell'array c
    }
    printf("La somma degli elementi dell'array c è: %.2f\n", somma);
```

```c
    // e) Copiare l'array a nella prima porzione dell'array b
    float a[11];  // Supponiamo un array di 11 elementi
    // Inizializziamo l'array a con alcuni valori (come esempio)
    for (int i = 0; i < 11; i++) {
        a[i] = i + 1;  // Assegniamo valori da 1 a 11 per esempio
    }
    // Ora copiamo l'array a nella prima parte dell'array b
    for (int i = 0; i < 11; i++) {
        b[i] = a[i];
    }

    // Stampa per verifica
    printf("Array b dopo la copia: ");
    for (int i = 0; i < 11; i++) {
        printf("%.1f ", b[i]);
    }
    printf("\n");

    return 0;
}
```

6.4. Scrivete istruzioni 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.
b) Quanti elementi contiene l'array `table`? Stampate il numero totale di elementi.
c) Usate un'istruzione di iterazione `for` per inizializzare ogni elemento di `table` alla somma dei suoi indici. Usate variabili `x` e `y` come variabili di controllo.
d) Stampate i valori di ogni elemento dell'array `table`. Supponete che l'array sia stato inizializzato con la definizione:

```
int table[SIZE][SIZE] = {{1, 8}, {2, 4, 6}, {5}};
```

6.4. a) `int table[SIZE][SIZE];`

b) Nove elementi. `printf("%d\n", SIZE * SIZE);`

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

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

Output:
```
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
```

6.12. Scrivete dei cicli che effettuino ognuna delle seguenti operazioni su array unidimensionali:

a) Inizializzare a zero i 10 elementi dell'array intero `counts`.
b) Aggiungere 1 a ognuno dei 15 elementi dell'array intero `bonus`.
c) Leggere i 12 valori dell'array `monthlyTemperatures` di tipo `float` dalla tastiera.
d) Stampare i cinque valori dell'array intero `bestScores` nel formato a colonne.

```c
int main() {
    // a) Inizializzare a zero i 10 elementi dell'array intero counts
    int counts[10];
    for (int i = 0; i < 10; i++) {
        counts[i] = 0;  // Inizializzo ogni elemento dell'array counts a 0
    }

    // b) Aggiungere 1 a ognuno dei 15 elementi dell'array intero bonus
    int bonus[15];
    for (int i = 0; i < 15; i++) {
        bonus[i] += 1;  // Aggiungo 1 a ciascun elemento dell'array bonus
    }

    // c) Leggere i 12 valori dell'array monthlyTemperatures di tipo float dalla tastiera
    float monthlyTemperatures[12];
    printf("Inserisci 12 valori per monthlyTemperatures:\n");
    for (int i = 0; i < 12; i++) {
        printf("Valore %d: ", i + 1);
        scanf("%f", &monthlyTemperatures[i]);  // Leggo i valori inseriti dalla tastiera
    }

    // d) Stampare i cinque valori dell'array intero bestScores nel formato a colonne
    int bestScores[5] = {100, 90, 85, 80, 75};  // Supponiamo di avere alcuni valori per bestScores
    printf("\nBest Scores:\n");
    for (int i = 0; i < 5; i++) {
        printf("%d\t", bestScores[i]);  // Stampo i valori in formato a colonne
    }
    printf("\n");
```

6.10. ***(Commissioni sulle vendite)*** Usate un array unidimensionale per risolvere il seguente problema.
Un'azienda paga i suoi agenti di vendita su commissione. Gli agenti di vendita ricevono $200 alla settimana più il 9 % delle loro vendite lorde per quella settimana. Per esempio, un agente di vendita che ottiene un introito lordo sulle vendite di $3000 in una settimana riceve $200 più il 9 % di $3000 , ovvero un totale di $470. Scrivete un programma in C (usando un array di contatori) che determini quanti agenti di vendita hanno avuto i loro guadagni in ognuno dei seguenti intervalli (supponete che il guadagno di ogni agente di vendita sia troncato a una quantità intera):

a) $200-299
b) $300-399
c) $400-499
d) $500-599
e) $600-699
f) $700-799
g) $800-899
h) $900-999
i) $1000 e oltre

```c
#include <stdio.h>

int main(void) {
   int salaries[11] = {0}; // array to hold salary counts
   double i = 0.09; // commission percentage

   // prompt user for gross sales
   printf("%s", "Enter employee gross sales (-1 to end): ");
   int sales; // current employee's sales
   scanf("%d", &sales);

   // while sentinel value not read from user
   while (sales != -1) {
      // calculate salary based on sales
      double salary = 200.0 + sales * i;
      printf("Employee Salary is $%.2f\n", salary);

      // update appropriate salary range
      if (salary >= 200 && salary < 1000) {
         ++salaries[(int) salary / 100];
      }
      else if (salary >= 1000) {
         ++salaries[10];
      }

      // prompt user for another employee sales amount
      printf("%s", "\nEnter employee gross sales (-1 to end): ");
      scanf("%d", &sales);
   }

   // display table of ranges and employees in each range
   printf("%s", "\nEmployees in the range:\n");
   printf("$200-$299 : %d\n", salaries[2]);
   printf("$300-$399 : %d\n", salaries[3]);
   printf("$400-$499 : %d\n", salaries[4]);
   printf("$500-$599 : %d\n", salaries[5]);
   printf("$600-$699 : %d\n", salaries[6]);
   printf("$700-$799 : %d\n", salaries[7]);
   printf("$800-$899 : %d\n", salaries[8]);
   printf("$900-$999 : %d\n", salaries[9]);
   printf("Over $1000: %d\n", salaries[10]);
}
```

6.15. ***(Eliminazione di duplicati)*** Usate un array unidimensionale per risolvere il seguente problema. Memorizzate 20 numeri, ognuno dei quali compreso tra 10 e 100 , estremi inclusi. Quando un numero viene letto, stampatelo solo se non è un duplicato di un numero già letto. Tenete conto del “caso peggiore”, in cui tutti i 20 numeri sono differenti. Usate l’array più piccolo possibile per risolvere questo problema.

```c
#include <stdbool.h>
#include <stdio.h>
#define MAX 20

int main(void) {
   int a[MAX] = {0}; // array for user input
   int k = 0; // number of values currently entered

   printf("Enter 20 integers between 10 and 100:\n");

   // get 20 integers from user
   for (int i = 0; i <= MAX - 1; ++i) {
      bool duplicate = false;
      int value; // current value
      scanf("%d", &value);

      // test if integer is a duplicate
      for (int j = 0; j < k; ++j) {
         // if duplicate, raise flag and break loop
         if (value == a[j]) {
            duplicate = true;
            break;
         }
      }

      // if number is not a duplicate, enter it in array
      if (!duplicate) {
         a[k++] = value;
      }
   }

   puts("\nThe nonduplicate values are:");

   // display array of nonduplicates
   for (int i = 0; a[i] != 0; ++i) {
      printf("%d  ", a[i]);
   }

   puts("");
}
```

6.35. ***(Stampare un array)*** Scrivete una funzione ricorsiva `printArray` che riceva come argomenti un array e la dimensione dell'array, stampi l'array e non restituisca niente. La funzione deve arrestare l'elaborazione e tornare alla funzione chiamante quando essa riceve un array di dimensione zero.

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

#define SIZE 10

// function prototype
void printArray(int array[], int low, int high);

int main(void) {
   srand(time(NULL));

   int array[SIZE]; // array to be printed

   // initialize array elements to random numbers
   for (int loop = 0; loop < SIZE; ++loop) {
      array[loop] = 1 + rand() % 500;
   }

   puts("Array values printed in main:");

   // print array elements
   for (int loop = 0; loop < SIZE; ++loop) {
      printf("%d  ", array[loop]);
   }

   puts("\n\nArray values printed in printArray:");
   printArray(array, 0, SIZE - 1);
   puts("");
}

// function to recursively print an array
void printArray(int array[], int low, int high) {
   // print first element of array passed
   printf("%d  ", array[low]);

   // return if array only has 1 element
   if (low == high) {
      return;
   }
   else { // call printArray with new subarray
      printArray(array, low + 1, high);
   }
}
```

6.36. ***(Stampare una stringa all'indietro)*** Scrivete una funzione ricorsiva `stringReverse` che riceva un array di caratteri come argomento, lo stampi all'indietro e non restituisca niente. La funzione deve arrestare l'elaborazione e tornare alla funzione chiamante quando incontra il carattere nullo di terminazione della stringa.

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

#define SIZE 10

// function prototype
void printArray(int array[], int low, int high);

int main(void) {
   srand(time(NULL));

   int array[SIZE]; // array to be printed

   // initialize array elements to random numbers
   for (int loop = 0; loop < SIZE; ++loop) {
      array[loop] = 1 + rand() % 500;
   }

   puts("Array values printed in main:");

   // print array elements
   for (int loop = 0; loop < SIZE; ++loop) {
      printf("%d  ", array[loop]);
   }

   puts("\n\nArray values printed in printArray:");
   printArray(array, 0, SIZE - 1);
   puts("");
}

// function to recursively print an array
void printArray(int array[], int low, int high) {
   // print first element of array passed
   printf("%d  ", array[low]);

   // return if array only has 1 element
   if (low == high) {
      return;
   }
   else { // call printArray with new subarray
      printArray(array, low + 1, high);
   }
}
```

6.31. ***(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.

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

// function prototype
int testPalindrome(char array[], int left, int right);

int main(void) {
   char string[SIZE]; // original string
   char copy[SIZE]; // copy of string without spaces

   puts("Enter a sentence:");
   char c; // temporarily holds keyboard input
   int count = 0; // length of string

   // get sentence to test from user
   while ((c = getchar()) != '\n' && count < SIZE) {
      string[count++] = c;
   }

   string[count] = '\0'; // terminate string
   int copyCount = 0;

   // make a copy of string without spaces
   for (int i = 0; string[i] != '\0'; ++i) {
      if (string[i] != ' ' && string[i] != ',' && string[i] != '.'
         && string[i] != '!') {
         copy[copyCount++] = string[i];
      }
   }

   // print whether or not the sentence is a palindrome
   if (testPalindrome(copy, 0, copyCount - 1)) {
      printf("\"%s\" is a palindrome\n", string);
   }
   else {
      printf("\"%s\" is not a palindrome\n", string);
   }
}

// function to see if the sentence is a palindrome
int testPalindrome(char array[], int left, int right) {
   // test array to see if a palindrome
   if (left == right || left > right) {
      return 1;
   }
   else if (array[left] != array[right]) {
      return 0;
   }
   else {
      return testPalindrome(array, left + 1, right - 1);
   }
}
```

6.19. ***(Lancio dei dadi)*** Scrivete un programma che simuli il lancio di due dadi. Il programma deve usare due volte `rand` per lanciare, rispettivamente, il primo dado e il secondo dado, poi deve calcolare la somma dei due valori. Poiché ogni dado può mostrare sulla faccia superiore un valore intero da 1 a 6, la somma dei due valori varierà allora da 2 a 12, con 7 che è la somma più frequente e 2 e 12 che sono le somme meno frequenti. Il diagramma seguente mostra 36 possibili combinazioni dei due dadi.

Il vostro programma deve lanciare i due dadi 36.000 volte. Usate un array unidimensionale per annotare il numero delle volte in cui compare ogni possibile somma. Stampate i risultati in un formato tabellare. Stabilite inoltre se i totali sono ragionevoli: per esempio, vi sono sei modi di ottenere un risultato 7, quindi approssimativamente un sesto di tutti i lanci deve avere come somma 7.

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

```c
// Exercise 6.19 Solution
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
   srand(time(NULL)); // seed random number generator
   int sum[13] = {0}; // count occurrences of each combination

   // array expected contains counts for the expected
   // number of times each sum occurs in 36 rolls of the dice
   int expected[13] = {0, 0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1};

   // roll dice 36,000 times
   for (int i = 1; i <= 36000; ++i) {
      int x = 1 + rand() % 6;
      int y = 1 + rand() % 6;
      ++sum[x + y];
   }

   printf("%10s%10s%10s%10s\n", "Sum", "Total", "Expected", "Actual");

   // display results of rolling dice
   for (int j = 2; j <= 12; ++j) {
      printf("%10u%10u%9.3f%%%9.3f%%\n", j, sum[j],
         100.0 * expected[j] / 36, 100.0 * sum[j] / 36000);
   }
}
```