Array e Funzioni - 2

Mattia Curri - m.curri8@studenti.uniba.it

Dipartimento di Informatica, Università di Bari

Informazioni sul tutorato (ufficiali):
https://www.uniba.it/it/ricerca/dipartimenti/informatica/orientamento-pcto-tutorato/tutorato/info-per-studenti/informazioni-per-utenza

Esercizio 3

Sposta tutti gli elementi dell'array di una posizione verso destra (l'ultimo diventa il primo).

```c
#include <stdio.h>

int main() {
    int arr[] = {1,2,3,4,5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int temp = arr[n-1];
    int i = n-1;
    while(i > 0) {
        arr[i] = arr[i-1];
        i--;
    }
    arr[0] = temp;
    // Se vogliamo stampare l'array risultante
    i = 0;
    while(i < n) {
        printf("%d ", arr[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

Esercizio 4

Dato un array, rimuovi gli elementi duplicati.

```c
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 2, 3, 4, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int newArr[n]; // Può contenere al massimo n elementi unici

    // Inizializziamo newArr a 0
    int i = 0;
    while (i < n) {
        newArr[i] = 0;
        i++;
    }

    i = 0;
    int k = 0;
    while(i < n) {
        int found = 0;
        int j = 0;
        while(j < k && !found) {
            if(newArr[j] == arr[i]) {
                found = 1;
            }
            j++;
        }
        if(!found) {
            newArr[k] = arr[i];
            k++;
        }
        i++;
    }
    i = 0;
    while(i < k) {
        printf("%d ", newArr[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

E se volessimo farlo *inplace*, senza usare un array ausiliario?

```c
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 2, 3, 4, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int k = 0; // Ci serve salvare la nuova VERA dimensione dell'array
    int i = 0;
    while(i < n) {
        int found = 0;
        int j = 0;
        while(j < k && !found) {
            if(arr[j] == arr[i]) {
                found = 1;
            }
            j++;
        }
        if(!found) {
            arr[k] = arr[i];
            k++;
        }
        i++;
    }
    i = 0;
    while(i < k) { // Attenzione: k è la nuova dimensione dell'array
        printf("%d ", arr[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

Esercizio 5

Dato un array, stampa per ogni valore quante volte compare.

Vediamo cosa combina Gemini. Una soluzione molto complessa. Possiamo farlo in maniera più elegante?

```c
#include <stdio.h>

int main() {
    int arr[] = {10, 5, 20, 5, 10, 5, 30};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Frequenza degli elementi nell'array:\n");

    for (int i = 0; i < n; i++) {
        int visitato = 0; // Flag per marcare se l'abbiamo già contato

        // 1. Controlla se abbiamo GIA' contato questo elemento
        // Scorrendo all'indietro da i-1 fino a 0
        for (int j = i - 1; j >= 0; j--) {
            if (arr[i] == arr[j]) {
                visitato = 1; // Trovato!

                // Ha usato il break! Questa NON è programmazione strutturata!
                break; // Non serve continuare
            }
        }

        // Se 'visitato' è 1, significa che l'abbiamo già stampato.
        // Passa al prossimo elemento del ciclo 'i'.
        if (visitato) {
            continue;
        }

        // 2. Se non è stato visitato, conta le sue occorrenze
        int count = 1; // Inizia a 1 (conta se stesso)
        for (int j = i + 1; j < n; j++) { // Scorre in avanti da i+1
            if (arr[i] == arr[j]) {
                count++;
            }
        }

        // 3. Stampa il risultato
        printf("-> %d compare %d volte\n", arr[i], count);
    }

    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int arr[] = {1,2,2,3,4,4,4};
    int n = sizeof(arr)/sizeof(arr[0]);

    // Prendiamo il valore massimo da arr
    int max = arr[0];
    int i = 1;
    while(i < n) {
        if(arr[i] > max) {
            max = arr[i];
        }
        i++;
    }

    // Gli indici dell'array count rappresenteranno i numeri da 1 a max
    // Il valore in count[i] rappresenterà quante volte i+1 compare in arr
    int count[max];
    i = 0;
    while(i < max) {
        count[i] = 0;
        i++;
    }

    // Conta le occorrenze
    i = 0;
    while(i < n) {
        count[arr[i]-1]++;
        i++;
    }

    // Stampa
    i = 0;
    while(i < max) {
        if(count[i] > 0) {
            printf("%d: %d volte\n", i+1, count[i]);
        }
        i++;
    }
    return 0;
}
```

Molto più elegante, no?

Ma purtroppo presenta alcuni problemi:

- Funziona solo con numeri positivi
- Occupa molta memoria se il valore massimo è molto grande

Come risolviamo il primo problema?

```c
#include <stdio.h>

int main() {
    int arr[] = {-3, -1, -2, -3, -3, -1, 0, 1, 2, 2, 11, 9};
    int n = sizeof(arr)/sizeof(arr[0]);

    // Prendiamo sia il massimo che il minimo da arr
    int max = arr[0];
    int min = arr[0];
    int i = 1;
    while(i < n) {
        if (arr[i] < min) {
            min = arr[i];
        }
        if (arr[i] > max) {
            max = arr[i];
        }
        i++;
    }

    // Gli indici dell'array count rappresenteranno i numeri da 1 a max
    // Il valore in count[i] rappresenterà quante volte i+1 compare in arr
    int count[max - min + 1];

    // Inizializza count a 0
    i = 0;
    while(i < max - min + 1) {
        count[i] = 0;
        i++;
    }

    // Conta le occorrenze
    i = 0;
    while(i < n) {
        count[arr[i]-min]++;
        printf("%d %d\n", arr[i] - min, count[arr[i]-min]);
        i++;
    }

    // Stampa
    i = 0;
    while(i < max - min + 1) {
        if(count[i] > 0) {
            printf("%d: %d volte\n", i+min, count[i]);
        }
        i++;
    }
    return 0;
}
```

Qual è la migliore soluzione?

- Dipende dai vincoli del problema
- Se abbiamo solo numeri positivi e un valore massimo contenuto, la soluzione con l'array di conteggio è molto efficiente.
- Se i numeri possono essere negativi o il valore massimo è molto grande, la soluzione con il conteggio delle occorrenze è più adatta. Come mai?

Esercizio 6

Un numero intero è un numero perfetto se i suoi fattori primi, compreso 1 (ma non il numero stesso), hanno come somma il numero stesso. Per esempio, 6 è un numero perfetto perché 6 = 1 + 2 + 3.

Scrivete una funzione `isPerfect` che determini se il parametro `number` è un numero perfetto.

Usate questa funzione in un programma che determini e stampi tutti i numeri perfetti tra 1 e 1.000. Stampate i fattori di ogni numero perfetto per confermare che il numero è effettivamente perfetto. Sfidate la potenza del vostro computer provando numeri molto più grandi di 1.000.

```c
int main() {
    int i = 1;
    while (i <= 1000) {
        // A
        int sum = 0;
        int j = 1;
        while (j <= i / 2) {
            if (i % j == 0) {
                sum = sum + j;
            }
            j = j + 1;
        }

        if (sum == i) {
            printf("%d è un numero perfetto\n", i);
        }

        i = i + 1;
    }
    return 0;
}
```

```c
#include <stdio.h>

int isPerfect(int number) {
    int sum = 0;
    int i = 1;
    while (i <= number / 2) {
        if (number % i == 0) {
            sum += i;
        }
        i++;
    }
    return sum == number;
}

int main() {
    int i = 1;
    while (i <= 1000) {
        int x = isPerfect(i);
        if (x == 1) { // Oppure direttamente if(isPerfect(i))
            printf("%d è un numero perfetto\n", i);
        }
        i++;
    }
    return 0;
}
```

https://codeforces.com/problemset/problem/263/A

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

int main() {
    int a[5][5];
    int x, y;

    // Leggi la matrice 5x5 in input
    int i = 0;
    while (i < 5) {
        int j = 0;
        while (j < 5) {
            scanf("%d", &a[i][j]);
            // Trova la posizione dell'1
            if (a[i][j] == 1) {
                x = i;
                y = j;
            }
            j++;
        }
        i++;
    }

    // Calcola il numero di mosse per portare l'1 al centro (2,2)
    printf("%d\n", abs(2 - x) + abs(2 - y));
    return 0;
}
```

Extra

Esercizio 7

Implementa una funzione che cerchi un elemento specifico in un array di interi e restituisca l'indice della prima occorrenza, oppure -1 se non trovato.

Esercizio 8

Scrivi il codice per stampare la seguente matrice:

```
1 2 3
4 5 6
7 8 9
```

- Array multidimensionale
- Tip: usa due cicli annidati per righe e colonne