Esercizi con Array

Filippo Cugini

Lancio dadi con array

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 7
#define LANCI 60000000

int main(void)
{
   unsigned int frequency[SIZE] = {0}; // azzera

   srand(time(NULL)); //  seme del generatore casuale

   // lancia il dado 60.000.000 di volte
   for (unsigned int roll = 1; roll<=LANCI; ++roll) {
      size_t face = 1 + rand() % 6;
      ++frequency[face];
   }
```

Lancio dadi con array

```
   printf("%s%17s\n", "Face", "Frequency");

   // stampa gli elementi di frequency 1-6
   for (size_t face = 1; face < SIZE; ++face) {
      printf("%4d%17d\n", face, frequency[face]);
   }
}
```

Lancio dadi con array

Un output del programma:

```
Face        Frequency
   1          9997167
   2         10003506
   3         10001940
   4          9995833
   5         10000843
   6         10000711
```

Lancio dadi con array

Notate che abbiamo usato un array `frequency` di 7 elementi per contare le 6 facce del dado

Abbiamo infatti ignorato `frequency[0]`

E' infatti logico memorizzare le statistiche del lancio della faccia 1 nell'array all'indice 1

In questo modo, l'esito del lancio 1, incrementa `frequency[1]` invece di `frequency[0]`

Questo ci permette di usare ogni risposta direttamente come indice nell'array `frequency`, rendendo il programma più leggibile

```
++frequency[face];
```

Lancio dadi con array

Buona pratica di programmazione

Sforzatevi di rendere il programma il più chiaro possibile

A volte, ma non sempre, può valere la pena sacrificare l’uso efficiente della memoria (es. una locazione dell’array) o del tempo di esecuzione in favore di una scrittura di programmi più chiari

Stringhe con array

Un array di caratteri può essere inizializzato usando una stringa letterale:

```
char string1[] = "first";
```

inizializza gli elementi dell'array `string1` con i singoli caratteri nella stringa "first"

In questo caso, la dimensione dell'array `string1` è determinata dal compilatore in base alla lunghezza della stringa

Stringhe con array

La stringa "first" contiene cinque caratteri più un carattere speciale di terminazione di stringa chiamato **carattere nullo**

Così, l'array `string1` contiene in realtà sei elementi

La costante di tipo carattere che rappresenta il carattere nullo è

`'\0'`

```
char string1[] = "first";
```

Stringhe con array

Tutte le stringhe in C terminano con questo carattere nullo `'\0'`

Un array di caratteri che rappresenta una stringa deve essere sempre definito di dimensioni abbastanza grandi da contenere il numero di caratteri nella stringa e il carattere nullo che la termina

```
char string1[] = {'f','i','r','s','t','\0'};
```

Stringhe con array

Gli array possono anche essere inizializzati con costanti individuali di tipo carattere poste in una lista di valori di inizializzazione

Ciò può tuttavia risultare pesante

```
char string1[] = {'f','i','r','s','t','\0'};
```

Stringhe con array

Dal momento che una stringa è a tutti gli effetti un array di caratteri, possiamo accedere ai suoi caratteri individuali direttamente usando la notazione con indice per gli array

Ad esempio,

`string1[0]` è il carattere '`f`'

`string1[3]` è il carattere '`s`'

```
char string1[] = "first";
```

Stringhe con array

Possiamo anche inserire una stringa direttamente nell'array di caratteri dalla tastiera usando `scanf` e lo specificatore di conversione `%s`

Ad esempio:

```
char string2[20];
```

crea un array di caratteri che può memorizzare una stringa di al massimo *19 caratteri* e *un carattere nullo di terminazione*

```
scanf("%s", string2);
```

Manipolazione di stringhe con array

L'istruzione:

```
scanf("%19s", string2);
```

legge una stringa dalla tastiera e la memorizza in `string2`

Il nome dell'array è passato a `scanf` **non** preceduto dal carattere `&` che abbiamo visto usato con variabili che non sono stringhe

```
scanf("%d", &var);
```

Manipolazione di stringhe con array

Il carattere `&` si usa normalmente per fornire a `scanf` il riferimento alla locazione di memoria della variabile, così che vi si possa memorizzare un valore

Ma il valore del nome di un array è l'indirizzo dell'inizio dell'array

il carattere `&`, pertanto, con array **non** è necessario

```
scanf("%19s", string2);
```

Manipolazione di stringhe con array

La funzione `scanf` legge i caratteri finché non si incontra uno spazio, una tabulazione, un newline o un indicatore di end-of-file

La stringa `string2` non deve essere più lunga di 19 caratteri così da lasciare spazio al carattere nullo che la termina

Se l'utente scrive 20 o più caratteri, il programma può arrestarsi o creare un problema di vulnerabilità relativamente alla sicurezza noto come overflow del buffer

```
char string2[20];
scanf("%19s", string2);
```

Manipolazione di stringhe con array

Per questa ragione, abbiamo usato lo specificatore di conversione `%19s` in modo che `scanf` legga un massimo di 19 caratteri e non scriva caratteri in memoria oltre la fine dell'array `string2`

È vostra responsabilità assicurarvi che l'array nel quale la stringa letta viene memorizzata possa contenere qualunque stringa che l'utente scrive alla tastiera

La funzione `scanf` non controlla quanto è grande l'array, e di conseguenza `scanf` può anche scrivere oltre la fine dell'array

```
scanf("%19s", string2);
```

Manipolazione di stringhe con array

Un array di caratteri che rappresenta una stringa può essere inviato in uscita con `printf` con lo specificatore di conversione `%s`.

L'array `string2` è stampato con l'istruzione:

```
print("%s\n", string2);
```

La funzione `printf`, come `scanf`, non controlla quanto è grande l'array di caratteri

I caratteri della stringa sono stampati finché non si incontra un carattere nullo di terminazione

Manipolazione di stringhe con array

Consideriamo ora un esempio che riassuma:

- l'inizializzazione di un array di caratteri con una stringa letterale
- la memorizzazione di una stringa letta in un array di caratteri
- la stampa di un array di caratteri come una stringa
- l'accesso ai caratteri individuali di una stringa

Nota: In questo programma leggiamo solo stringhe che non contengono caratteri di spaziatura

Manipolazione di stringhe con array

```c
#include <stdio.h>
#define SIZE 20

int main(void)
{
   char string1[SIZE]; // riserva 20 caratteri
   char string2[] = "string literal"; // riserva 15 ch

   // memorizza la stringa inserita dall'utente
   printf("%s", "Enter a string (n<=19 ch): ");
   scanf("%19s", string1); // leggi <=19 caratteri

   // stampa le stringhe
   printf("string1 is: %s\n", string1);
   printf("string2 is: %s\n", string2);
   printf("string1 with spaces between characters is:\n");
```

Manipolazione di stringhe con array

```
    // stampa i caratteri
    // finche' non si raggiuge il carattere nullo
    for (size_t i = 0; i<SIZE && string1[i] != '\0'; ++i)
    {
       printf("%c ", string1[i]);
    }

    puts("");
}
```

Manipolazione di stringhe con array

Un output del programma:

```
Enter a string (no longer than 19 characters):
Hello there
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o
```