Corso di Informatica A.A. 2009-2010

Lezione 9

Il linguaggio C

- Istruzioni di input/output
- Arrays

La funzione `printf()`

La funzione `printf()` consente di mostrare sullo standard output (normalmente lo schermo) sia messaggi di testo che il valore di variabili definite nel programma

```
#include <stdio.h>
printf( format-string[,arguments, . . . ]);
```

La stringa di formato è obbligatoria, mentre l'uso degli argomenti è opzionale

La stringa di formato

La stringa di formato specifica come deve essere "formattato" l'output

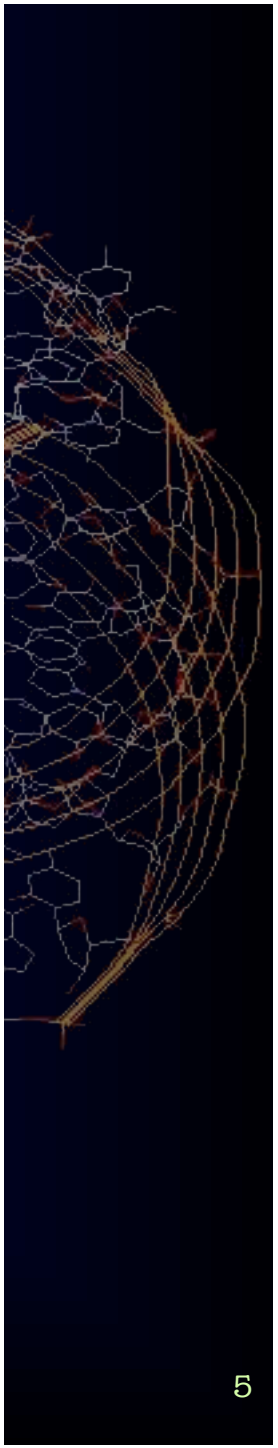
• **caratteri di testo**: sono stampati esattamente come sono stati scritti all'interno della stringa di formato

•**carattere di controllo:** un \ seguito da un singolo carattere

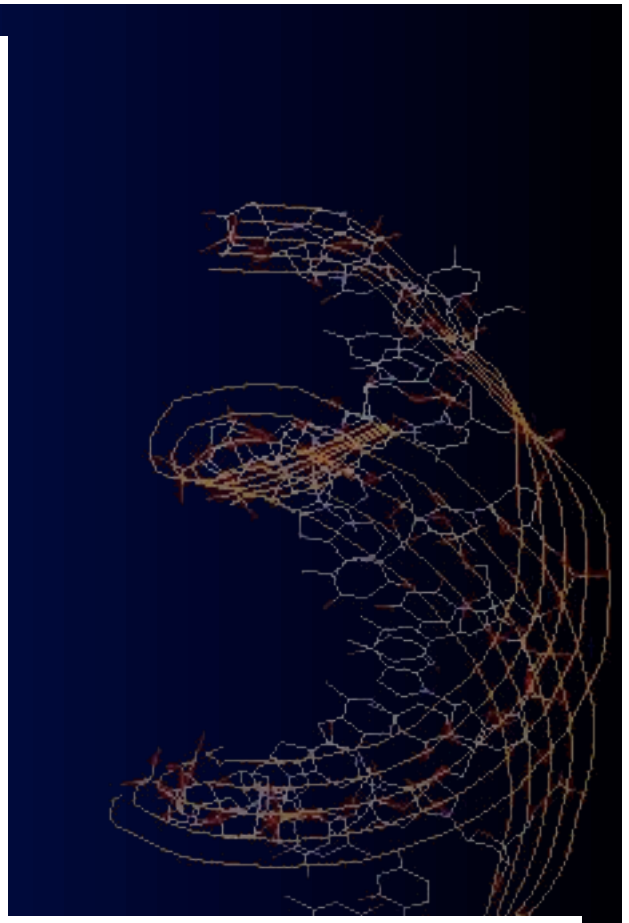
Sequence	*Meaning*
\a	Bell (alert)
\b	Backspace
\n	Newline
\t	Horizontal tab
\\	Backslash
\?	Question mark
\'	Single quotation

•**carattere di conversione:** un % seguito da un carattere

```c
/* Demonstration of frequently used escape sequences */

#include <stdio.h>

#define QUIT  3

int  get_menu_choice( void );
void print_report( void );

int main()
{
    int choice = 0;

    while (choice != QUIT)
    {
       choice = get_menu_choice();

       if (choice == 1)
          printf("\nBeeping the computer\a\a\a" );
       else
       {
          if (choice == 2)
              print_report();
       }
    }
    printf ("You chose to quit!\n");

    return 0;
}
```



```
int get_menu_choice( void )
{
     int selection = 0;

     do
     {
          printf( "\n" );
          printf( "\n1 - Beep Computer" );
          printf( "\n2 - Display Report");
          printf( "\n3 - Quit"):
          printf( "\n" );
          printf( "\nEnter a selection:" );

          scanf( "%d", &selection );

      }while ( selection < 1 || selection > 3 );

      return selection;
}
```



```
void print_report( void )
{
     printf( "\nSAMPLE REPORT" );
     printf( "\n\nSequence\tMeaning" );
     printf( "\n=========\t=======" );
     printf( "\n\\a\t\tbell (alert)" );
     printf( "\n\\b\t\tbackspace" );
     printf( "\n . . . \t\t . . . ");
}
```

Caratteri di conversione

La stringa di formato deve contenere un carattere di conversione per ogni variabile che deve essere stampata.

Il carattere di conversione deve corrispondere al tipo di Variabile da stampare

Specifier	*Meaning*	*Types Converted*
`%c`	Single character	`char`
`%d`	Signed decimal integer	`int, short`
`%ld`	Signed long decimal integer	`long`
`%f`	Decimal floating-point number	`float, double`
`%s`	Character string	`char` arrays
`%u`	Unsigned decimal integer	unsigned int, unsigned short
`%lu`	Unsigned long decimal integer	unsigned long

La funzione `printf()`

Una singola istruzione printf() può stampare un numero illimitato di variabili purché la stringa di formato contenga un carattere di conversione per ogni variabile.

```
/* Demonstration using printf() to display numerical values. */

#include <stdio.h>

int a = 2, b = 10, c = 50;
float f = 1.05, g = 25.5, h = -0.1;

int main()
{
    printf("\nDecimal values without tabs: %d %d %d", a, b, c);
    printf("\nDecimal values with tabs: \t%d \t%d \t%d", a, b, c);

    printf("\nThree floats on 1 line: \t%f\t%f\t%f", f, g, h);
    printf("\nThree floats on 3 lines: \n\t%f\n\t%f\n\t%f", f, g, h);

    printf("\nThe rate is %f%%", f);
    printf("\nThe result of %f/%f = %f\n", g, f, g / f);

    return 0;
}
```

La funzione `puts()`

La funzione `puts()` può essere utilizzata per visualizzare sullo schermo dei messaggi di testo, ma non può stampare variabili Numeriche.

La funzione `puts()` riceve una singola stringa come argomento e la visualizza sullo schermo andando a capo automaticamente

```
puts("Hello, world.");
```

```
printf("Hello, world.\n");
```

DO & DON'T

Utilizzare la funzione `puts()` ogni volta che è necessario stampare unicamente del testo

Non inserire più linee di testo in un'unica istruzione `printf()`

Non dimenticare il carattere di newline per stampare più linee mediante più `printf()`

Non utilizzare i caratteri di conversione con la funzione `puts()`

La funzione `scanf()`

Il modo più flessibile di leggere dati numerici da una tastiera si ottiene utilizzando la funzione `scanf()`

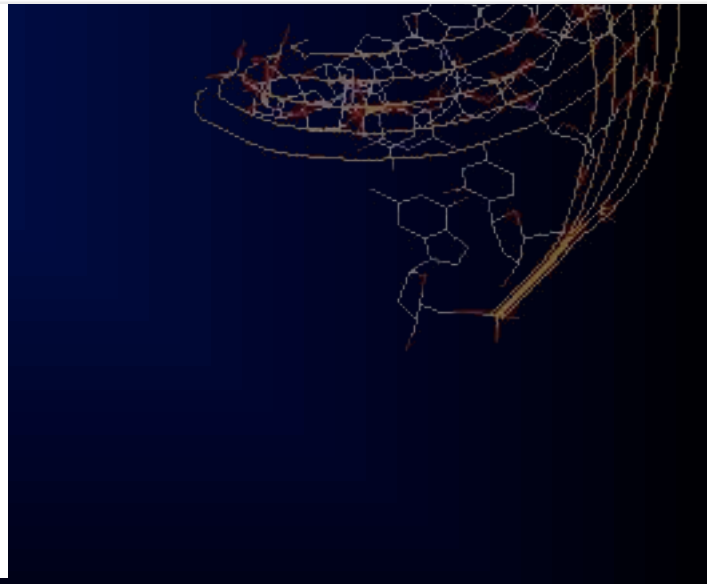
```
scanf("%f", &rate);
```

Una singola `scanf()` può gestire l'input di più variabili

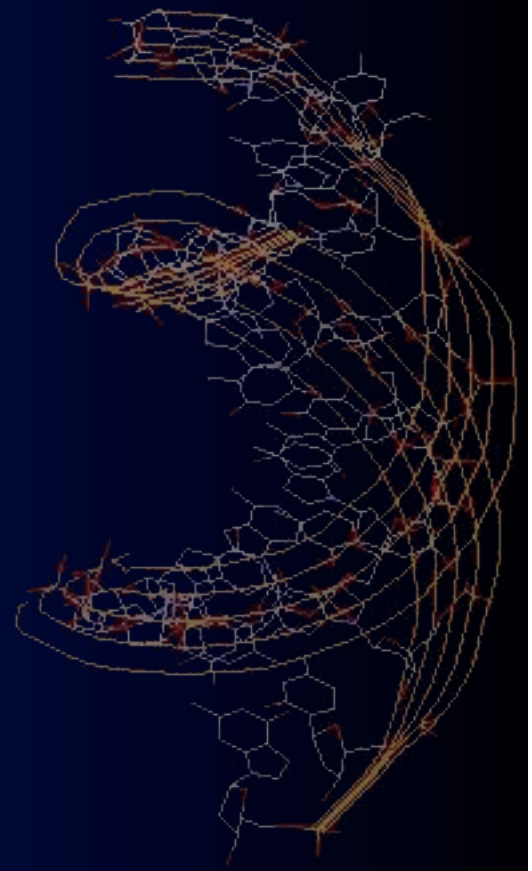
```
scanf("%d %f", &x, &rate);
```

Per fornire in input più variabili bisogna separarle con uno "spazio bianco" (spazio, tab, newline)

```c
/* Demonstration of using scanf() */
#include <stdio.h>

#define QUIT 4

int get_menu_choice( void );

int main()
 {
 int choice = 0;
 int int_var = 0;
 float float_var = 0.0;
 unsigned unsigned_var = 0;

 while (choice != QUIT)
 {
 choice = get_menu_choice();

 if (choice == 1)
 {
 puts("\nEnter a signed decimal integer (i.e. -123)");
 scanf("%d", &int_var);
 }
 if (choice == 2)
 {
 puts("\nEnter a decimal floating-point number (e.g. 1.23)");
 scanf("%f", &float_var);
 }
 if (choice == 3)
 {
 puts("\nEnter an unsigned decimal integer  (e.g. 1.23)" );
 scanf( "%u", &unsigned_var );
 }
 }
 printf("\nYour values are: int: %d float: %f unsigned: %u \n",
 int_var, float_var, unsigned_var );
 return 0;
 }
```

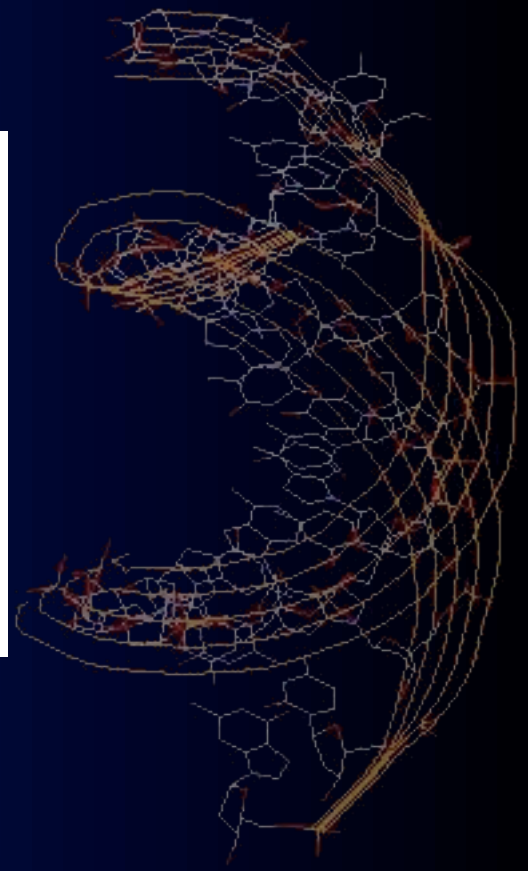


```c
int get_menu_choice( void )
{
int selection = 0;

do
{
puts( "\n1 - Get a signed decimal integer" );
puts( "2 - Get a decimal floating-point number" );
puts( "3 - Get an unsigned decimal integer" );
puts( "4 - Quit" );
puts( "\nEnter a selection:" );

scanf( "%d", &selection );

}while ( selection < 1 || selection > 4 );

return selection;
}
```



BUG BUSTER

```
int get_1_or_2( void )
{
     int answer = 0;
     while (answer < 1 || answer > 2)
     {
          printf(Enter 1 for Yes, 2 for No);
          scanf( "%f", answer );
     }
     return answer;
}
```



Arrays

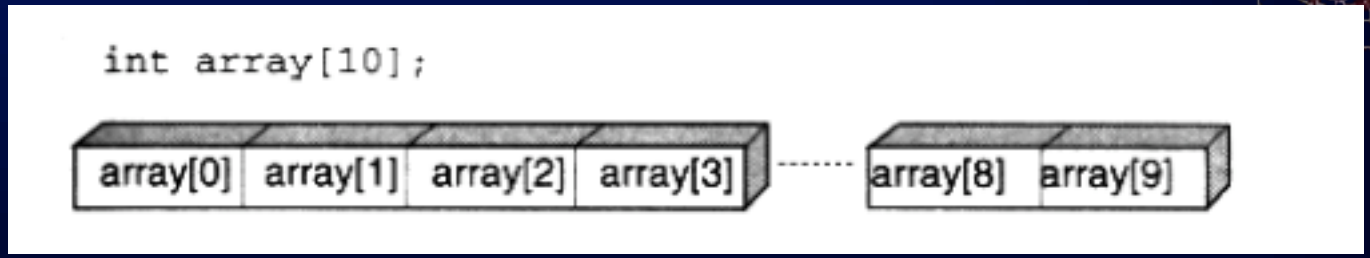
Un array è un insieme contiguo di celle di memoria tutte contenenti dati dello stesso tipo. Queste celle, che costituiscono in pratica un gruppo di variabili, hanno tutte lo stesso nome ed sono accessibili attraverso un indice.

```
int a[10], b[100];
float f[30];
char c[200];
```

La dichiarazione di un array (statico) è analoga a quella di una variabile, ad eccezione del fatto che il nome dell'array è seguito dal numero di elementi che lo compongono, racchiuso tra parentesi quadre.

Arrays

Quando viene dichiarato un array il compilatore riserva un "blocco" di memoria atto a contenere l'intero array.
Gli elementi dell'array sono "salvati" in locazioni di memoria contigue



Arrays

Reminder: se int *a[10]* e' un array, il primo elemento e' *a[0]* e l'ultimo e' *a[9]*.

Cosa accade se si tenta di referenziare un elemento che non fa parte dell'array, ad esempio `a[10]` ?

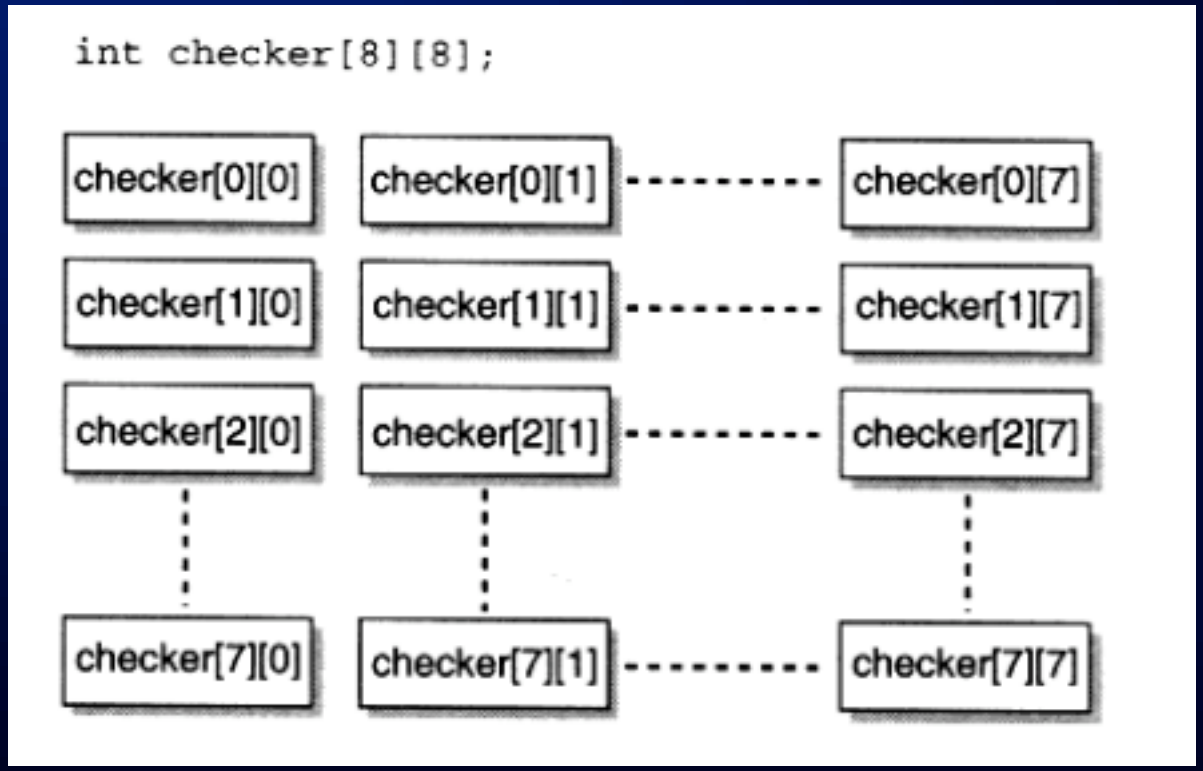
Il compilatore non puo' rivelare un tale errore: la memoria relativa ad `a[10]` può essere sempre letta e scritta, ma non fa parte dell'array!!
In caso di lettura, il valore letto non ha alcun significato logico ai fini del programma, mentre in caso di scrittura si rischia di sovrascrivere qualche altro dato importante che risiede nella locazione di memoria subito dopo le locazioni assegnate all'array.

Se la locazione di memoria referenziata non appartiene al programma, il programma stesso termina con un errore (segmentation fault).

```c
/* expenses.c - Demonstrates use of an array */

#include <stdio.h>

/* Declare an array to hold expenses, and a counter variable */

float expenses[13];
int count;

int main()
{
    /* Input data from keyboard into array */

    for (count = 1; count < 13; count++)
    {
        printf("Enter expenses for month %d: ", count);

        scanf("%f", &expenses[count]);
    }

    /* Print array contents */

    for (count = 1; count < 13; count++)
    {
        printf("Month %d = $%.2f\n", count, expenses[count]);
    }
    return 0;
}
```

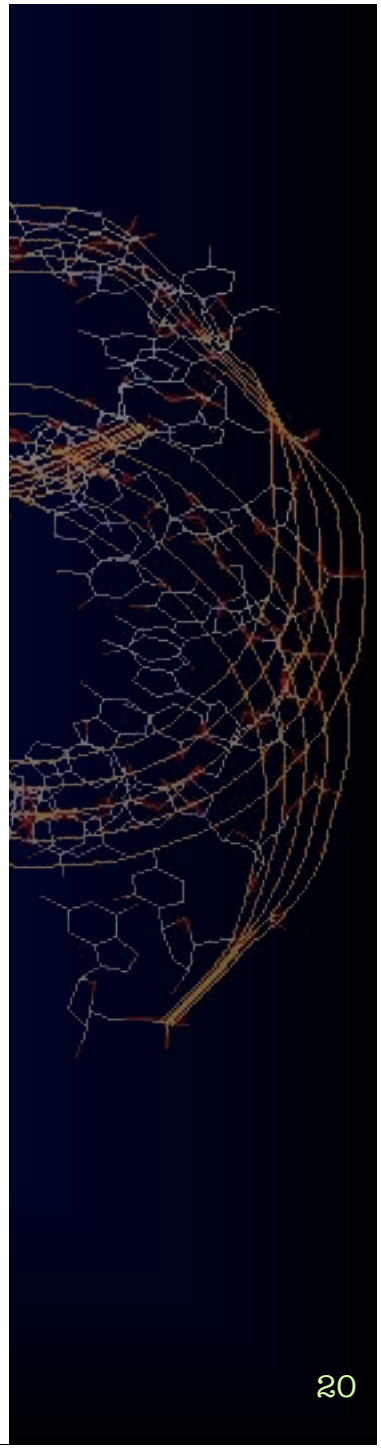
Arrays multidimensionali

Un array multidimensionale possiede più di un indice



In C non vi è alcun limite al numero di dimensioni di un array

```c
/*grades.c - Sample program with array */
/* Get 10 grades and then average them */

#include <stdio.h>

#define MAX_GRADE 100
#define STUDENTS  10

int grades[STUDENTS];

int idx;
int total = 0;            /* used for average */

int main()
{
    for( idx=0;idx< STUDENTS;idx++)
    {
        printf( "Enter Person %d's grade: ", idx +1);
        scanf( "%d", &grades[idx] );

        while ( grades[idx] > MAX_GRADE )
        {
            printf( "\nThe highest grade possible is %d",
                    MAX_GRADE );
            printf( "\nEnter correct grade: " );
            scanf( "%d", &grades[idx] );
        }

        total += grades[idx];
    }

    printf( "\n\nThe average score is %d\n", ( total / STUDENTS) );

    return 0;
}
```



DO & DON'T

Utilizzare l'istruzione #define per creare delle costanti per definire gli arrays

Evitare di utilizzare arrays di dimensione maggiore di tre

Non dimenticare che gli indici degli arrays partono da 0

Inizializzare gli arrays

All'atto della dichiarazione è possibile inizializzare gli elementi di un array

```
int array[4] = { 100, 200, 300, 400 };
```

```
int array[] = { 100, 200, 300, 400 };
```

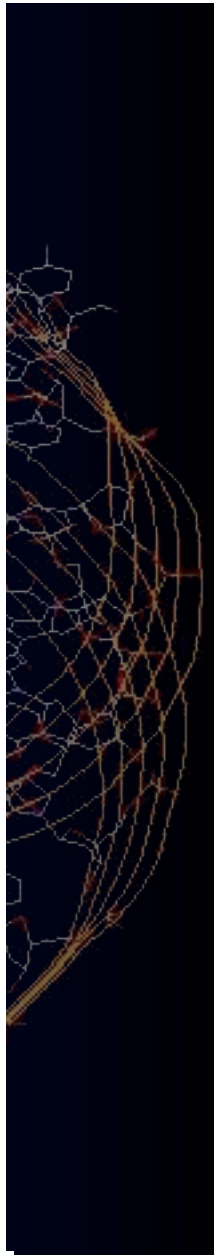
Anche gli arrays multidimensionali possono essere inizializzati

```
int array[4][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

```
array[0][0] is equal to 1
array[0][1] is equal to 2
array[0][2] is equal to 3
array[1][0] is equal to 4
array[1][1] is equal to 5
array[1][2] is equal to 6
. . .
array[3][1] is equal to 11
array[3][2] is equal to 12
```

```
int array[4][3] = { { 1, 2, 3 } , { 4, 5, 6 } ,
{ 7, 8, 9 } , { 10, 11, 12 } };
```

```c
/* random.c - Demonstrates using a multidimensional array */

#include <stdio.h>
#include <stdlib.h>
/* Declare a three-dimensional array with 1000 elements */

int random_array[10][10][10];
int a, b, c;

int main()
{
    /* Fill the array with random numbers. The C library */
    /* function rand() returns a random number. Use one */
    /* for loop for each array subscript. */

    for (a = 0; a < 10; a++)
    {
        for (b = 0; b < 10; b++)
        {
            for (c = 0; c < 10; c++)
            {
                random_array[a][b][c] = rand();
            }
        }

    }
```



```
    /* Now display the array elements 10 at a time */

    for (a = 0; a < 10; a++)
    {
        for (b = 0; b < 10; b++)
        {
            for (c = 0; c < 10; c++)
            {
                printf("\nrandom_array[%d][%d][%d] = ", a, b, c);
                printf("%d", random_array[a][b][c]);
            }
            printf("\nPress Enter to continue, CTRL-C to quit.");

            getchar();
        }
    }
    return 0;
}   /* end of main() */
```

La funzione getchar() legge un singolo carattere dalla tastiera.
In questo caso serve a mettere in pausa il programma fino a quando l'utente non digita Enter

Dimensioni di un array

Le dimensioni in byte di un array dipendono ovviamente dal numero e dal tipo degli elementi che lo costituiscono

BUG BUSTER

```
int x, y;
int array[10][3];
int main()
{
    for ( x = 0; x < 3; x++ )
        for ( y = 0; y < 10; y++ )
            array[x][y] = 0;
    return 0;
}
```

```
int array[10];
int x = 1;


int main()
{

    for ( x = 1; x <= 10; x++ )
        array[x] = 99;
.
    return 0;
}
```