Ordinamento array

Titolo unità didattica: Strutture dati: array [07]

Titolo modulo : Function in C per problemi di base con array – parte 3 [11-C]

Sviluppo di function in C per problemi di base per array 1D ed esempi di utilizzo

Argomenti trattati:

- ✓ function in C per la fusione di array ordinati
- ✓ function in C per la determinare l'uguaglianza di array
- ✓ esempi di generazione di griglie 1D e campionamenti in C

Prerequisiti richiesti: AP-05-03-C, AP-07-04-T, AP-07-05-T, AP-07-07-T, AP-07-08-C

Esercizio 1

Realizzare una function in C per la fusione di due array 1D odinati.

```
void fusione(char a[], int n_a, char b[], int n_b, char c[])
```

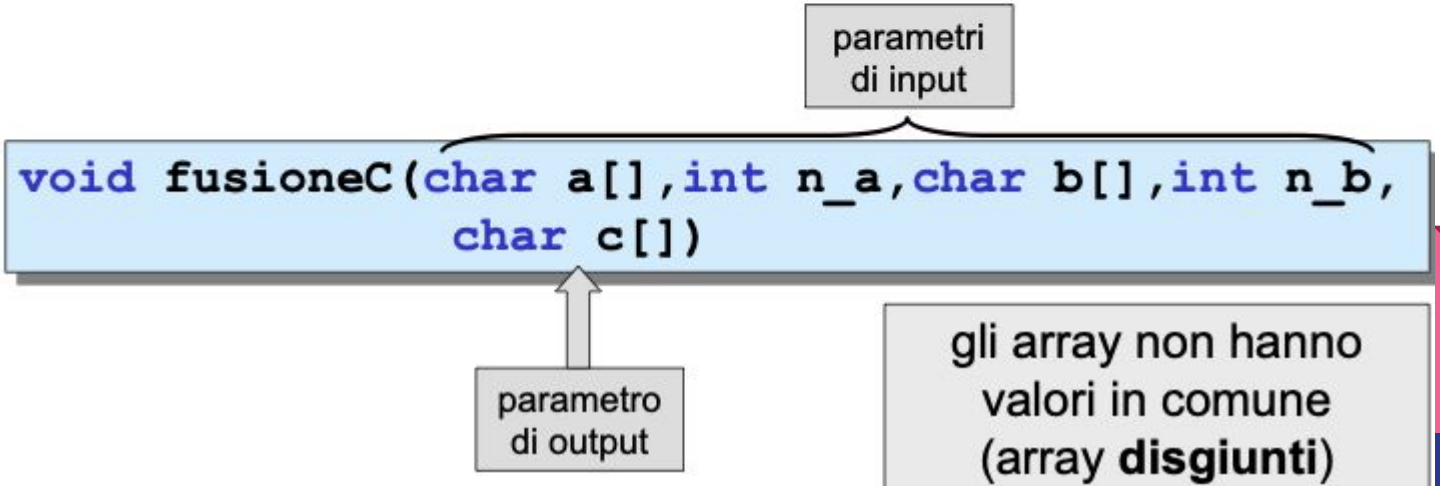
Fondere insieme i due array a[] e b[] nell'array c[].

```
void fusione(in: char a[],int n_a, char b[], int n_b;
out: char c[]) {
 int i_a, i_b, i_c;
 i_a = 0 ;
 i_b = 0 ;
 for (i_c = 0; i_c < n_a+n_b; i_c++) {
   if (i_a < n_a && i_b < n_b) {
/* ci sono sia elementi di a sia di b da
  prendere in considerazione */
                if (a[i_a] < b[i_b])
                  { c[i_c] = a[i_a] ;
                    i_a = i_a+1 ; }
             else}
                  {c[i_c] = b[i_b] ;
                   i_b = i_b+1 ; }
/* uno dei due array non deve essere più usato */
   else { if (i_b >= n_b) {
              /* considerare solo a  */
                   { c[i_c] = a[i_a];
                     i_a = i_a+1 ; }
                  else
              /* considerare solo b  */
                   { c[i_c] = b[i_b] ;
                     i_b = i_b+1 ; }
                }
              }
}
```

esercizio realizzare una function C per la fusione di due array 1D ordinati

algoritmo di **fusione** (***merge***) di array ordinati

```
void fusione(in: char a[], int n_a, char b[], int
n_b; out: char c[])
```



```
void fusioneC(char a[],int n_a,char b[],int n_b,char c[])
{
  int i_a=0,i_b=0,i_c;
  for (i_c=0; i_c < n_a+n_b; i_c++)
    {
     if(i_a<n_a && i_b<n_b)
      {
       if(a[i_a] < b[i_b]){
          c[i_c] = a[i_a];
          i_a++;}
       else {
          c[i_c] = b[i_b];
          i_b++;}
      }
      else if(i_b >= n_b) {
          c[i_c] = a[i_a];
          i_a++;}
       else {
          c[i_c] = b[i_b];
          i_b++;}
    }
}
```

```
void fusione(in: char a[],int n_a, char b[], int n_b;
out: char c[]) {
 int i_a, i_b, i_c;
 i_a = 0 ;
 i_b = 0 ;
 for (i_c = 0; i_c < n_a+n_b; i_c++) {
   if (i_a < n_a && i_b < n_b) {
/* ci sono sia elementi di a sia di b da
  prendere in considerazione */
                if (a[i_a] < b[i_b])
                  { c[i_c] = a[i_a] ;
                    i_a = i_a+1 ; }
            else}
                  {c[i_c] = b[i_b] ;
                   i_b = i_b+1 ; }
/* uno dei due array non deve essere più usato */
   else { if (i_b >= n_b) {
              /* considerare solo a  */
                   { c[i_c] = a[i_a];
                     i_a = i_a+1 ; }
                  else
              /* considerare solo b  */
                   { c[i_c] = b[i_b] ;
                     i_b = i_b+1 ; }
                }
              }
}
```

```
void fusioneC(char a[],int n_a,char b[],
              int n_b,char c[])
{
  int i_a=0,i_b=0,i_c=0;
  while (i_a < n_a && i_b <n_b)
    {
     if(a[i_a] < b[i_b])
         c[i_c++] = a[i_a++];
       else
         c[i_c++] = b[i_b++];
     }
  while (i_a < n_a)
     c[i_c++] = a[i_a++];
  while (i_b < n_b)
     c[i_c++] = b[i_b++];
}
```

viene **prima** usato il valore dell'indice e **poi** tale valore viene incrementato

```c
void fusione(int a[], int n_a, int b[], int n_b, int c[]);
void visualizza(int c[], int n_c);
//void visualizza_char(char c[], int n_c);

int main(void){

    int a[]={100,200,300,400,500}, b[]={15,25,30,45,50}, c[10];
    //char d[]={'a', 'c', 'd', 'f', 'h'}, e[]={'b', 'e', 'g', 'h', 'm'}, f[10];
    int n_a = 5, n_b = 5, n_c;

    fusione(a, n_a, b, n_b, c);

    n_c = n_a + n_b;
    visualizza(c, n_c);

    return 0;
}


void fusione(int a[], int n_a, int b[], int n_b, int c[]){

    int i_a =0, i_b= 0, i_c=0;
    while(i_a < n_a && i_b < n_b){
        if(a[i_a] < b[i_b])
            c[i_c++] = a[i_a++];
        else
            c[i_c++] = b[i_b++];
    }

    while(i_a <n_a)
        c[i_c++] = a[i_a++];

    while(i_b <n_b)
        c[i_c++] = b[i_b++];
```

Esercizio 2

Realizzare una function in C per determinare l'uguaglianza di due array 1D.

```
int uguaglianza(char a[], char b[], int n)
```

esercizio realizzare una function C per determinare l'uguaglianza di due array 1D

```
logical uguaglianza_array(char a[], char b[], int n)
```

```
int uguaglianza_arrayC(char a[],char b[],int n)
```

```
int uguaglianza_arrayC(char a[],char b[],int n)
{
 int i=0,uguale=1;
 while(uguale && i<n)
     {
      if(a[i] != b[i])
         uguale = 0;
      i++;
      }
 return uguale;
}
```

Esercizio 2

realizzare un main C che definisce un array testo e richiama la function “uguaglianza” per determinare l’uguaglianza della prima metà e della seconda metà dell’array testo

```
int uguaglianza_arrayC(char a[],char b[],int n)
```

```
uguaglianza_arrayC(testo,&testo[n_mezzi],n_mezzi)
```

esercizio

realizzare un main C che definisce un array `testo` e richiama la function

```
int uguaglianza_arrayC(char a[],
char b[],int n)
```

per determinare l'uguaglianza della prima metà e della seconda metà dell'array testo

```
int uguaglianza_arrayC(char a[],char b[],int n)
```

```
uguaglianza_arrayC(testo,&testo[n_mezzi],n_mezzi)
```

```c
#include <stdio.h>
int uguaglianza_arrayC(char a[],char b[],int n);
int epari(int x);
void main()
{
  char testo[]={'q','u','e','q','u','i'};
  int i,n_elem, n_mezzi;
  n_elem =  6;
  if (epari(n_elem))
     n_mezzi = n_elem/2;
   else
     printf("la lunghezza del testo deve essere un numero pari\n");
  for(i=0;i<n_elem;i++)
    printf("%c",testo[i]);
  printf("\n");
  if(uguaglianza_arrayC(testo,&testo[n_mezzi],n_mezzi))
     printf("le due meta sono uguali");
   else
     printf("le due meta non sono uguali");
}
int epari(int x)
{
 return !(x%2);
}
```

esercizio realizzare una function C che campiona su una griglia una funzione data come parametro

realizzare un main C che richiama la function C **campiona** per determinare il campionamento su una griglia uniforme di 50 punti sull'intervallo $[\pi,3\pi]$ della funzione

$$\frac{3\sin(x)}{x^2+10}$$

```
void campionaF(float fun(float),float a,
     float b,int n,float f_c[]);
```

$$y = \frac{3\sin(x)}{x^2 + 10}$$

```
float mia_fun(float x)
{
 return (3.0F*sin(x))/(pow(x,2)+10.F);
}
```

```
void campionaF(float fun(float),float a,
     float b,int n,float f_c[])
{
 float passo,p_griglia;
 int i;
 passo = (b-a)/(n-1);
 p_griglia = a;
 for (i=0;i<n;i++)
    {
      f_c[i] = fun(p_griglia);
      p_griglia = a + (i+1)*passo;
    }
}
float mia_fun(float x)
{
     return (3.0*sin(x))/(pow(x,2.)+10.0);
}
```

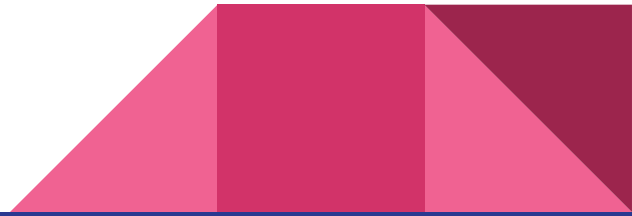
```c
#include <stdio.h>
#include <math.h>
void campionaF(float fun(float),float,float
     ,int ,float []);
float mia_fun(float );
void main()
{
  const float pi=3.1415926F;
  float mia_a,mio_b;
  int mio_n,i;
  float mio_f_c[100];
  mia_a = pi;
  mio_b = 3.F*pi;
  mio_n = 50;
  campionaF(mia_fun,mia_a,mio_b,mio_n,mio_f_c);
  for (i=0;i<mio_n;i++)
     printf(" %f",mio_f_c[i]);
}
```

passagio di una function a un'altra function

Esercizi

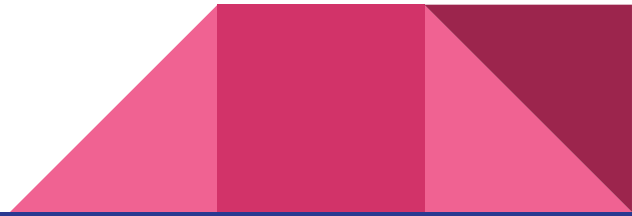
Esercizio 10 [✓]

Scrivere un programma che simuli un *salvadanaio*. L'utente può inserire e prelevare soldi. Visualizzare *salvadanaio vuoto* se non ci sono soldi. Il numero di operazioni di inserimento e prelievo sono decise dall'utente. L'inserimento e il prelievo devono avvenire con la chiamata a due procedure differenti.



Esercizio 11

Scrivere due funzioni che calcolino il massimo e il minimo di un insieme di valori inseriti dall'utente. Scrivere successivamente un programma che verifichi se la loro differenza è minore di un certo valore inserito.



I numeri casuali

Per la generazione di numeri pseudocasuali in C viene usata la funzione **rand()**. Essa genera solo numeri interi (tipo int) che appartengono all'intervallo $[0, \mathbf{RAND\_MAX}]$

e che sono uniformemente distribuiti in tale intervallo. **RAND_MAX** è definito nella libreria **stdlib.h**. Un semplice utilizzo è il seguente

```
int nc;
nc = rand();
```

Per generare a ogni esecuzione del programma una diversa sequenza di numeri pseudocasuali è necessario usare la funzione **srand()**

```
srand(nuovo_seed);
```

Ad esempio per una generazione automatica di un diverso seme si può usare la seguente chiamata a **srand**

```
srand((unsigned int)time(0));
```

dove **time** è una funzione della libreria **time.h**, che restituisce il valore dell'ora riportato dall'orologio interno del calcolatore.
Per avere dei numeri interi casuali uniformemente distribuiti in un qualsiasi intervallo $[A, B]$ viene usata l'operazione di modulo

```
int nc;

nc = A + rand()%(B + 1 - A);
```

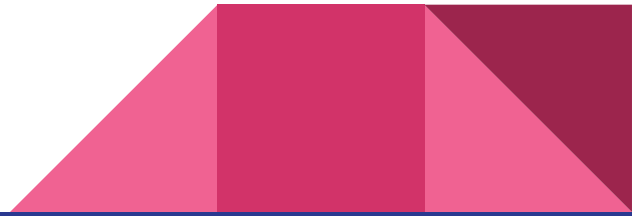
Per generare numeri pseudocasuali di tipo reale uniformemente distribuiti nell'intervallo $[a, b]$ usiamo invece la seguente relazione

```
nc_f = a+(b-a) * (float)rand()/(float)RAND_MAX;
```

Esercizi su rand()

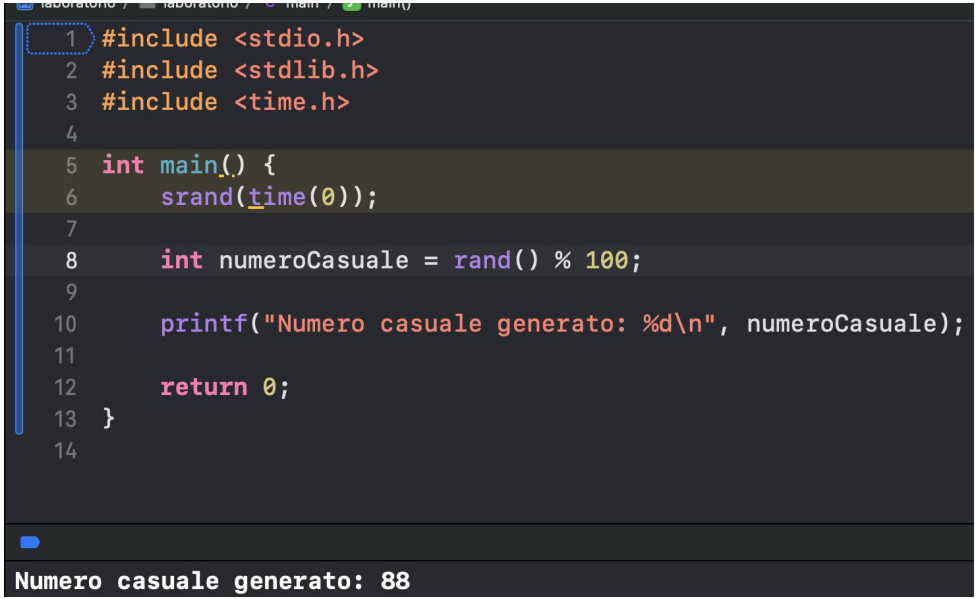
Esercizio 1

Genera e stampa un numero casuale tra 0 e 99



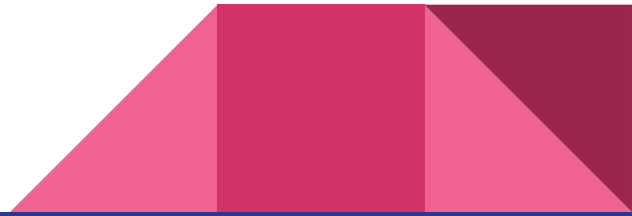
Esercizio 1

Genera e stampa un numero casuale tra 0 e 99



Esercizio 2

Popolare un array con numeri casuali da 0 a 99 e *ordinarli*.



Esercizio 12

Simulare per dieci volte il lancio di un dado e visualizzare la somma dei risultati di tutti i lanci.

Esercizio 13 [✓]

Due giocatori si sfidano lanciando un "dado truccato". Il dado ha dei valori interi nell'intervallo $[5, 15]$. A ogni turno vince il giocatore che ottiene un punteggio maggiore. In caso di parità il punto viene assegnato a entrambi. Simulare 10 sfide e visualizzare il giocatore che vince più volte.

Esercizio 14 [✓]

Scrivere una procedura che dato una ascissa x calcoli la seguente funzione matematica

$$(x^3 + 3x + 5)/(8x + 1) \tag{3.4}$$

Scrivere successivamente un programma che calcoli e visualizzi il risultato della funzione in 20 ascisse casuali scelte nell'intervallo $[0, 1]$.

Esercizio 12

Simulare per dieci volte il lancio di un dado e visualizzare la somma dei risultati di tutti i lanci.

Esercizio 13 [✓]

Due giocatori si sfidano lanciando un "dado truccato". Il dado ha dei valori interi nell'intervallo $[5, 15]$. A ogni turno vince il giocatore che ottiene un punteggio maggiore. In caso di parità il punto viene assegnato a entrambi. Simulare 10 sfide e visualizzare il giocatore che vince più volte.

Esercizio 14 [✓]

Scrivere una procedura che dato una ascissa x calcoli la seguente funzione matematica

$$(x^3 + 3x + 5)/(8x + 1) \tag{3.4}$$

Scrivere successivamente un programma che calcoli e visualizzi il risultato della funzione in 20 ascisse casuali scelte nell'intervallo $[0, 1]$.

laboratorio ⟩ laboratorio ⟩ main ⟩ lancia(primo, secondo)

```c
int main(void){
    int nc, i=0, a[10], sum;
    int pl_1, pl_2, punto_1=0, punto_2=0;

    srand((unsigned int) time(0));
    lancia(&pl_1, &pl_2);
    for(int i = 0; i<10 ; i++){
        lancia(&pl_1, &pl_2);
        printf(" %d ---- %d \n", pl_1, pl_2);

        if(pl_1 < pl_2)
            punto_2++;
        else if(pl_1 > pl_2)
            punto_1++;
        else{
            punto_1++;
            punto_2++;
        }
    }
    printf("Punteggio: \n Giocatore 1 = %d \n Giocatore 2 = %d\n", punto_1, punto_2);
    return 0;
}


void lancia(int *primo, int *secondo){
    *primo = 5 + rand() % 11;
    *secondo = 5 + rand() % 11;
}
```

```
11 ---- 9
9 ---- 14
15 ---- 8
7 ---- 13
9 ---- 14
9 ---- 9
10 ---- 12
Punteggio:
 Giocatore 1 = 4
 Giocatore 2 = 7
```

Soluzione esercizio esercitazione

```c
int main(void){

    int somma1, somma_gauss;
    int n;
    printf("inserisci il numero di cui fare la sommatoria \n");
    scanf("%d", &n);

    somma1 = sommatoria(n);
    printf("sommatoria = %d \n", somma1);
    somma_gauss = gauss(n);

    printf("sommatoria gauss = %d \n", somma_gauss);

    if(somma1 == somma_gauss)
        printf("Ben fatto!\n");
    else
        printf("Hai sbagliato qualcosa\n");
    return 0;
}

int sommatoria(int n){
    int k, somma1 =0;
    for(k = 0; k<=n; k++)
        somma1 += k;
    return somma1;
}

int gauss(int n){
    int somma2;
    somma2 =  n*(n+1)/2;
    return somma2;
}
```