

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

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[])
```

parametri
di input

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

parametro
di output

gli array non hanno
valori in comune
(array **disgiunti**)

```

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 ; }
            }
        }
    }
}

```

versione 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

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

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)
```

↑
inizio I
metà

↑
inizio II
metà

↑
lunghezza
porzioni

```
#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);
}
```

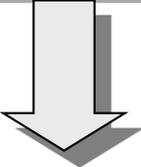
esercizio

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);
}
```

```
#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]);
}
```

passaggio di una
function a un'altra
function