Titolo unità didattica: Strutture dati: record [15]

Titolo modulo : Record in C: il tipo **`struct`** [02-C]

Argomenti trattati:

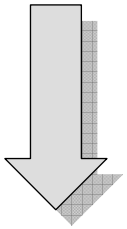
- ✓ tipo strutturato **`struct`** in C
- ✓ campi e accesso ai campi
- ✓ inizializzazione di una variabile struct
- ✓ array di struct in C
- ✓ puntatori a variabili struct in C
- ✓ passaggio di variabili struct (per valore e per riferimento) a una function C
- ✓ function C che restituiscono un dato struct

Prerequisiti richiesti: AP-05-04-C, AP-07-08-C, AP-08-04-C, AP-15-01-T

linguaggio C : tipo strutturato **`struct`**

il tipo strutturato **record** viene realizzato
in C mediante il
meccanismo delle strutture

il meccanismo delle strutture
(uso delle **`struct`**)
consente di dichiarare specifici tipi derivati



aggregati di variabili di tipo non omogeneo

strutture (tipi **struct**) in C

dichiarazione di una struttura (**tipo `struct`**)

```
struct <etichetta> {
    <campo_1>;
    <campo_2>;
    ……
    <campo_n>;
 };
```

```
struct <etichetta> {
    <tipo> <variabile>;
    <tipo> <variabile>;
    ……
    <tipo> <variabile>;
 };
```

tipi struct in C

dichiarazione di una struttura (**tipo `struct`**)

```
struct punto_2D {
    double ascissa;
    double ordinata;
 };
```

il nome del tipo derivato è **`struct punto_2D`**

```
struct data {
    int giorno;
    char mese[10];
    int anno;
 };
```

il nome del tipo derivato è **`struct data`**

tipi struct in C

dichiarazione di una struttura (**tipo `struct`**)

```
struct punto_2D {
    double ascissa;
    double ordinata;
 };
```

il nome del tipo derivato è **`struct punto_2D`**

la **dichiarazione** di una **`struct`** **non alloca memoria**

deve essere vista come uno schema che indica l'organizzazione di un generico dato strutturato di tipo **`struct punto_2D`**

tipi struct in C

dichiarazione di una variabile struttura (di uno specifico **tipo `struct`**)

```
struct punto_2D vertice,estremo_sin;
```

la **dichiarazione alloca memoria**
per le due variabili **`vertice` e `estremo_sin`**
di tipo **`struct punto_2D`**

```
struct data data_nascita,partenza;
```

la **dichiarazione alloca memoria**
per le due variabili **`data_nascita` e `partenza`**
di tipo **`struct data`**

dichiarazione di una variabile struttura (di uno specifico **tipo** **`struct`**)

```
struct data {
   int giorno;
   char mese[10];
   int anno;
} data_nascita,partenza;
```

le variabili sono dichiarate insieme alla struttura **`struct data`**

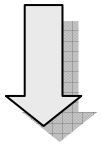
```
struct {
   int giorno;
   char mese[10];
   int anno;
} data_nascita,partenza;
```

le variabili sono dichiarate insieme alla struttura, che **non ha etichetta** e quindi **non** può essere più utilizzata per dichiarare altre variabili

dichiarazione di una struttura (**tipo `struct`**)

```
typedef struct data Data;
```

`Data` è un nuovo tipo di dato



```
Data data_nascita, partenza;
```

dichiarazione di una struttura (tipo `struct`)

```
struct indirizzo {
   struct {
      char *toponimo;
      int n_civico;
   } sede;
   int CAP;
   char *comune;
   char *provincia;
};
```

```
struct domicilio {
    char *toponimo;
    int n_civico;
};
struct indirizzo {
 struct domicilio sede;
 int CAP;
 char *comune;
 char *provincia;
};
```

```
typedef struct indirizzo Indirizzo;
```

accesso ai campi di una variabile di un tipo `struct`

```
Data data_nacita;
struct punto_2D vertice,estremo_sin;
double a;
Indirizzo ind_UniPrath;
data_nascita.giorno = 21;
data_nascita.mese = “settembre”;
vertice.ascissa = 2.1;
vertice.ascissa = estremo_sin.ascissa;
a = max_D(vertice.ascissa,vertice.ordinata);
ind_UniParth.sede.toponimo=“Via Acton”;
ind_UniParth.sede.n_civico = 33;
ind_UniParth.CAP = 80133;
ind_UniParth.comune = “Napoli”;
ind_UniParth.provincia = “NA”;
```

operatore di accesso al campo

```c
#include <stdio.h>
#include <string.h>
void main()
{
 struct indirizzo {
   struct {
     char toponimo[15];
     int n_civico;
   } sede;
   int CAP;
   char comune[10];
   char provincia[2];
 };
 struct indirizzo a,mio_ind;
 a.CAP = 80133;
 strcpy(a.provincia,"SA”);
 strcpy(mio_ind.provincia,"NA”);
 strcpy(mio_ind.sede.toponimo,“Piazza Roma”);
 printf(" provincia di a= %s
        CAP= %d\n",a.provincia,a.CAP);
 printf("mia provincia=%s sede in = %s\n",
       mio_ind.provincia,mio_ind.sede.toponimo);}
```

in alternativa:
stringhe dichiarate come array

attenzione:
il tipo **`struct indirizzo`** è **locale** al main

```c
#include <stdio.h>
#include <string.h>
 struct indirizzo {
   struct {
     char toponimo[15];
     int n_civico;
   } sede;
   int CAP;
   char comune[10];
   char provincia[2];
 };
void main()
{
 struct indirizzo a,mio_ind;
 a.CAP = 80133;
 strcpy(a.provincia,“SA”);
 strcpy(mio_ind.provincia,“NA”);
 strcpy(mio_ind.sede.toponimo,“Piazza Roma”);
 printf(" provincia di a= %s
        CAP= %d\n",a.provincia,a.CAP);
 printf("mia provincia=%s sede in = %s\n",
       mio_ind.provincia,mio_ind.sede.toponimo);
}
```

attenzione:
il tipo **`struct indirizzo`** è **globale**

```c
typedef enum{picche,fiori,quadri,cuori}Semi_Fr;
struct carta {
   int valore;
   Semi_Fr seme;
 };
 typedef struct carta Carta;
 Carta mia_carta,tua_carta;
```

```c
 mia_carta.valore = 10;
 mia_carta.seme = quadri;
 if(mia_carta.seme > tua_carta.seme)
  ……;
```

```c
 tua_carta = mia_carta;
```

ogni campo di **`mia_carta`** viene assegnato al corrispondente campo di **`tua_carta`**

inizializzazione di una variabile di un tipo `struct`

```
typedef enum{picche,fiori,quadri,cuori}Semi_Fr;
struct carta {
   int valore;
   Semi_Fr seme;
 };
 typedef struct carta Carta;
 Carta mia_carta = {10,quadri},
       tua_carta = {13,cuori};
```

```
stuct data data_nascita = {21,
                           ”settembre”,
                            1974};
```

array 1D di strutture

```
typedef enum{picche,fiori,quadri,cuori}Semi_Fr;
struct carta {
   int valore;
   Semi_Fr seme;
 };
 typedef struct carta Carta;
```

```
 Carta mazzo_francese[52];
```

```
 mazzo_francese[12].valore = 10;
 mazzo_francese[12].seme = quadri;
```

array 1D di strutture

```
struct punto_2D {
   double ascissa;
   double ordinata;
 };
 typedef struct punto_2D Triangolo[3];
```

```
 Triangolo mio_triangolo;
```

```
 mio_triangolo[0].ascissa = 0.0;
 mio_triangolo[0].ordinata = 0.0;
 mio_triangolo[1].ascissa = 2.0;
 mio_triangolo[1].ordinata = 0.0;
 mio_triangolo[2].ascissa = 2.0;
 mio_triangolo[2].ordinata = 3.0;
```

inizializzazione di un array 1D di strutture

```
Carta mano_di_poker[5]=
     {{11,cuori},
      {2,quadri},
      {11,fiori},
      {1,fiori},
      {3,fiori}};
```

```
Triangolo mio_triangolo = {{0.0,0.0},
                           {2.0,0.0},
                           {2.0,3.0}};
```

puntatori a variabili di un tipo struct

```
struct punto_2D * ppunto,
                  mio_punto={10.3,20.1};
Carta * pcarta,mia_carta={1,cuori};
ppunto = &mio_punto;
pcarta = &mia_carta;
```

indirizzo base della struttura

accesso ai campi attraverso puntatori a variabili di un tipo struct

```
(*ppunto).ascissa = 100.2;
(*ppunto).ordinata = 200.1;
```

sono necessarie

puntatori a variabili di un **tipo** `struct`

```
struct punto_2D * ppunto,
                  mio_punto={10.3,20.1};
Carta * pcarta,mia_carta={1,cuori};
ppunto = &mio_punto;
pcarta = &mia_carta;
```

indirizzo base della struttura

accesso ai campi attraverso puntatori a variabili di un **tipo** `struct`

```
ppunto -> ascissa = 100.2;
ppunto -> ordinata = 200.1;
```

operatore di accesso

puntatori ad array di strutture

```
Carta mano_di_poker[5]=
      {{11,cuori},
       {2,quadri},
       {11,fiori},
       {1,fiori},
       {3,fiori}};
Carta *pcarta;
pcarta = &mano_di_poker[0];
```

```
pcarta = mano_di_poker;
```

pcarta punta all'indirizzo base dell'array di **struct**

```c
#include <stdio.h>
void main()
{
 typedef enum {picche,fiori,quadri,cuori}Semi_Fr;
 struct carta {
   int valore;
   Semi_Fr seme;
 };
 typedef struct carta Carta;
 Carta mano_di_poker[5]={{11,cuori},{2,quadri},
                  {11,fiori},{1,fiori},{3,fiori}};
 Carta *pcarta;
 pcarta = &mano_di_poker[0];
 for(i=0;i<5;i++)
   printf ("carta n. %2d : valore= %d  seme=
      %d\n",i+1,(pcarta+i)->valore,(pcarta+i)->seme);
}
```

```
carta n. 1 : valore= 11  seme=  3
carta n. 2 : valore= 2   seme=  2
carta n. 3 : valore= 11  seme=  1
carta n. 4 : valore= 1   seme=  1
carta n. 5 : valore= 3   seme=  1
```

passaggio di variabili di un **tipo** `struct` come **parametri** di function

il passaggio di una variabile di un **tipo** `struct` a una function avviene **per valore**

```
double area_triangolo(Triangolo triang)
{
 double a,b,c,s;
 a = distanza_2d(triang[1],triang[2]);
 b = distanza_2d(triang[2],triang[3]);
 c = distanza_2d(triang[3],triang[1]);
 s = (a+b+c)/2.0;
 return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

```c
#include <stdio.h>
#include <math.h>
struct punto_2D {
   double ascissa;
   double ordinata;
 };
typedef struct punto_2D Triangolo[3];
double area_triangolo(Triangolo triang);
double distanza_2D(struct punto_2D p1, struct punto_2D p2);
void main()
{
  Triangolo mio_triang={{0.0,0.0},{2.0,1.0},{1.0,4.0}};
  double area;
  area = area_triangolo(mio_triang);
  printf("area del triangolo=%lf\n",area);
}
double distanza_2D(struct punto_2D p1,struct punto_2D p2)
{
 return sqrt(pow((p1.ascissa - p2.ascissa),2) +
                        pow((p1.ordinata - p2.ordinata),2));
}
```

una function C può restituire un dato di un **tipo `struct`**

```
#include <stdio.h>
struct carriera {
   char * cognome;
   char * nome;
   int voti_esami[20];
};
typedef struct carriera Carriera;
Carriera aggiorna_carriera(Carriera carr_da_agg);
void main()
{
 Carriera mia_carriera = {“Rossi”,“Ugo”,{30,30,28,30,27,28,30,28,
                         28,30,30,30,28,26,30,30,30,30,30,29}};
 int i;
 mia_carriera = aggiorna_carriera(mia_carriera);
 printf(“Cognome:%s\n,Nome:%s\n”,mia_carriera.cognome,
        mia_carriera.nome);
 for(i=0;i<20;i++)
    printf(“esame:%d-simo voto:%d”,i+1,mia_carriera.voti_esami[i]);
}
Carriera aggiorna_carriera(Carriera carr_da_agg)
{
  int i_esame,nuovo_voto;
  printf(“inserire n. Esame e nuovo voto:”);
  scanf(“%d %d”,&i_esame,&nuovo_voto);
  carr_da_agg.voti_esami[i_esame-1] = nuovo_voto;
  return carr_da_agg;
}
```

passaggio per riferimento simulato di una variabile di un **tipo `struct`**

```c
#include <stdio.h>
struct carriera {
   char * cognome;
   char * nome;
   int voti_esami[20];
};
typedef struct carriera Carriera;
void aggiorna_carriera(Carriera *pcarr_da_agg);
void main()
{
 Carriera mia_carriera = {“Rossi”,“Ugo”,{30,30,28,30,27,28,30,28,
                         28,30,30,30,28,26,30,30,30,30,30,29}};
 int i;
 aggiorna_carriera(&mia_carriera);
 printf(“Cognome:%s\n,Nome:%s\n”,mia_carriera.cognome,
        mia_carriera.nome);
 for(i=0;i<20;i++)
    printf(“esame:%d-simo voto:%d”,i+1,mia_carriera.voti_esami[i]);
}
void aggiorna_carriera(Carriera *pcarr_da_agg)
{
  int i_esame,nuovo_voto;
  printf(“inserire n. Esame e nuovo voto:”);
  scanf(“%d %d”,&i_esame,&nuovo_voto);
  pcarr_da_agg->voti_esami[i_esame-1] = nuovo_voto;
}
```