Struct

Dati strutturati in C

Dati omogenei e non...

L'array è un contenitore di dati omogenei.

Per unire dati non omogenei, ma tutti riferiti ad una entità, utilizziamo la struttura:

STRUCT

Struct

Una struct viene dichiarata, con tutte le parti che contiene, poi si può dichiarare una variabile di tipo 'struct' e poi può essere utilizzata richiamando le singole parti di cui è composta con un punto...

...Vediamo un esempio pratico!

Struct studente

Prendiamo ad esempio una entità studente.

struct

Nome struct

Parentesi {}

```
struct studente {
  char nome[40];
  char cognome[40];
  int eta;
  char classe[3];//"3B" "3C"
};
```

Dati elementari

; per ogni variabile definita all'interno della struct

Parentesi } e ; per chiudere

```c
#include <stdio.h>

struct studente {
  char nome[40];
  char cognome[40];
  int eta;
  char classe[3];//"3B" "3C"
};

int main(){

  struct studente stud;

  printf("Inserire il nome dello studente: ");
  gets(stud.nome);
  printf("\nInserire il cognome dello studente: ");
  gets(stud.cognome);
  printf("\nInserire il età dello studente: ");
  scanf("%d",&stud.eta);
  printf("\nInserire la classe dello studente: ");
  scanf(" %s",stud.classe);

  printf("\nDati Immessi: %s - %s - %d -%s", stud.nome,
stud.cognome, stud.eta, stud.classe);
  return 0;
}
```

Struct definizione globale

Variabile stud di tipo studente

stud.attributo per accedere ai singoli attributi di studente

Inizializzazione struct

Una struct può essere anche inizializzata così:

```
struct studente stud2 = {"Mario", "Rossi", 16, "4A"};
printf("\nDati Immessi: %s - %s - %d -%s", stud2.nome,
stud2.cognome, stud2.eta, stud2.classe);
```

Esercitazione

Utilizzando una struct classica:

```
struct punto{
   int x;
   int y;
};
```

Scrivere un programma che legge in input le coordinate di due punti in un piano cartesiano e in output restituisce la loro distanza.

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$