Qualche esempio di programmi in C

E Mumolo - DIA

- Passaggio di strutture a una funzione per valore

```
#include <stdio.h>
#include <string.h>

typedef struct
{
            int num_mat;
            char nome[20];
            float media;
}studente;

void func(studente record);

int main()
{
            studente record;

            record.num_mat=1;  strcpy(record.nome, "Giovanni");  record.media = 27.5;
            func(record);
            return 0;
}

void func(studente elem)
{
            printf(" Nr matricola: %d \n", elem.num_mat);
            printf(" Nome: %s \n", elem.nome);
            printf(" Media: %f \n", elem.media);
}
```

- Passaggio di strutture a una funzione per indirizzo

```
#include <stdio.h>
#include <string.h>

typedef struct
{
            int num_mat;
            char nome[20];
            float media;
}studente;

void func(studente *record);

int main()
{
            studente record;

            record.num_mat=1;  strcpy(record.name, "Giovanni");  record.media = 27.5;
            func(&record);
            return 0;
}

void func(studente *elem)
{
          printf(" Nr matricola: %d \n", elem->num_mat);
          printf(" Nome: %s \n", elem->nome);
          printf(" Media: %f \n", elem->media);
}
```

Operatore -> per recuperare I campi tramite l'indirizzo

- Strutture come variabili globali

```
#include <stdio.h>
#include <string.h>

typedef struct
{
            int num_mat;
            char nome[20];
            float media;
}studente ;
studente record; /* dichiarazione globale */

void func();

int main()
{
            record.id=1;
            strcpy(record.name, "Raju");
            record.percentage = 86.5;

            structure_demo();
            return 0;
}

void func()
            printf(" Nr matricola: %d \n", record.num_mat);
            printf(" Nome: %s \n", record.nome);
            printf(" Media: %f \n", record.media);
}
```

- Puntatori a strutture

```
#include <stdio.h>
#include <string.h>

typedef struct
{
     int num_mat;
     char nome[20];
     float media;
}studente;

int main()
{
     int i;
     studente record1 = {1, "Giorgio", 27.5};
     studente *ptr;

     ptr = &record1;


     printf("  num_mat: %d \n", ptr->num_mat);
     printf("  Nome: %s \n", ptr->nome);
     printf("  Media: %f \n\n", ptr->media);

     return 0;
}
```

- Tre metodi per copiare strutture

```
#include <stdio.h>
#include <string.h>
typedef struct
{
    int num_mat;
    char nome[30];
    float media;
}studente;

int main(){
    int i;

    studente record1 = {1, "Giorgio", 27.5};
    studente record2, *record3, *ptr1, record4;

    // primo metodo di copia
    record2=record1;

    // secondo metodo: funzione memcpy(dest, puntatore sorg, numero byte)
    ptr1 = &record1;   memcpy(record3, ptr1, sizeof(record1));

    // terzo metodo di copia
    record4.num_mat=record1.num_mat; strcpy(record4.nome, record1.nome);
    record4.media = record1.media

    return 0;
}
```

- Allocazione dinamica di stringhe

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
     char *mem;
     /* allocazione dinamica in heap */
     mem = malloc( 20 * sizeof(char) );

     if( mem == NULL )
        printf("Errore di allocazione malloc");
     else
        strcpy( mem,"stringastringa");

     printf("Contenuto della memoria allocata : %s\n", mem);

     free(mem); /*rimozione e recupero della memoria allocata*/
}
```

Array come membri di struct

```
#include <stdio.h>
#include <string.h>
typedef struct
{
    char nome [30];
    int  voti[ 5];
    int  totale;
    float media;
}studente;

int main()
{
    studente std;
    int i; char temp[30];

    printf("Scrivi il nome: ");
    scanf("%s",temp);
    strcpy(std.nome,temp);

    printf("scrivi 5 voti:\n");
    std.totale=0;
    for(i=0;i< 5;i++){
        printf("scrivi voto dell'esame %d : ",i+1);
        scanf("%d",&std.voti[i]);
        std.totale+=std.voti[i];
    }
    std.media=(float)std.totale/5;

    printf("\nNome: %s \nTotale: %d \nmedia: %.2f\n",std.nome,std.totale,std.media);

    return 0;
}
```

Struttura dati Union

```
#include <stdio.h>
typedef union
{
    char nome[32];
    float salario;
    int nlav;
} ulavoro;

typedef struct
{
    char nome[32];
    float salario;
    int nlav;
} slavoro;

int main()
{
    printf("size della union = %ld", sizeof(ulavoro));
    printf("\nsize della struct = %ld\n", sizeof(slavoro));
    return 0;
}
```

- Risultato:

```
size della union = 32
size della struct = 40
```

Struttura dati Union

- La dimensione della Union è pari alla variabile componente più grande
- Accesso alle variabili della Union: come la struct, operatore
- Solo una variabile contenuta nella Union può essere usata alla volta. Esempio:

```
                        #include <stdio.h>
                #include <string.h>
                union Data {
                      unsigned int i;
                      float f;
                      char str[20];
                };
                int main( ) {
                      union Data data;

                      data.i = 10;
                      data.f = 220.5;
/*”AAAA”=0x41414141=0100 0001 0100 0001 0100 0001 0100 0001=1094795585 in base 10*/
                      strcpy( data.str, "AAAA");

                      printf( "data.i : %d\n", data.i);
                      printf( "data.f : %f\n", data.f);
                      printf( "data.str : %s\n", data.str);

                      return 0;
                 }
```

- Risultato:

```
                 data.i : 1094795585
                 data.f : 12.078431
                 data.str : AAAA
```

Union come componente di una struct

```
#include <stdio.h>
typedef union {
        float coords[3];
        char info[20];
} data;

typedef struct {
        char *descr;
        int tipo;
        data stuff;
} data1;

int main() {
        data1 vet[3];
        int i;

        printf ("Hello World\n");

        for (i=0; i<3; i++) {
                vet[i].descr = "stringastringa";
                vet[i].tipo = 1;
                vet[i].stuff.coords[0] = 3.14;
        }
        vet[2].stuff.info[2] = 'X';

        for (i=0; i<3; i++) {
                printf("%s\n",vet[i].descr);
                printf("%5.2f\n",vet[i].stuff.coords[0]);
         }

}
```