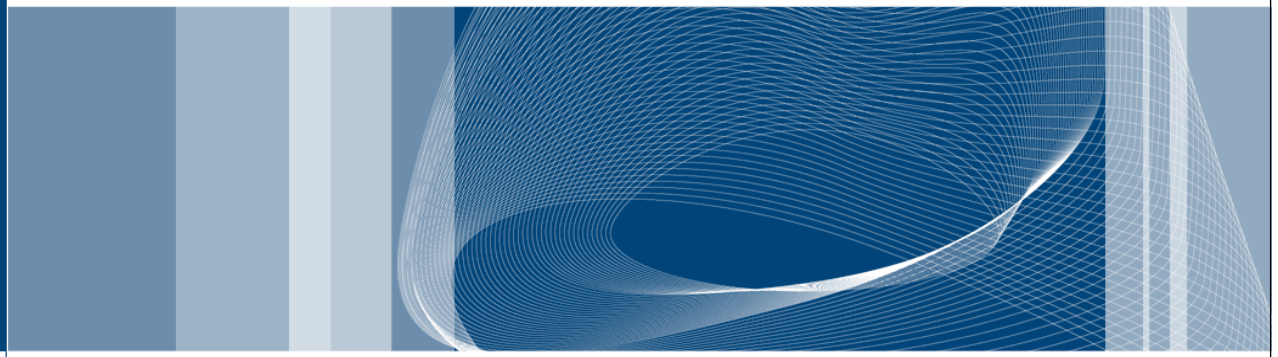
POLITECNICO DI MILANO



Fondamenti di Informatica 2013-2014

Strutture

Paola Mussida
Area Servizi ICT

Definire due strutture ognuna delle quali in grado di memorizzare i seguenti dati relativi ad una persona:

- ✓ Cognome
- ✓ Nome
- ✓ Età
- ✓ Altezza

Memorizzare i dati nelle due strutture e mostrare nome e cognome della persona più giovane.

```
struct {
        char cognome[40];
        char nome[40];
        int  eta;
        int  altezza;
} persona1, persona2;
```

Esercizio S_1 - Persona

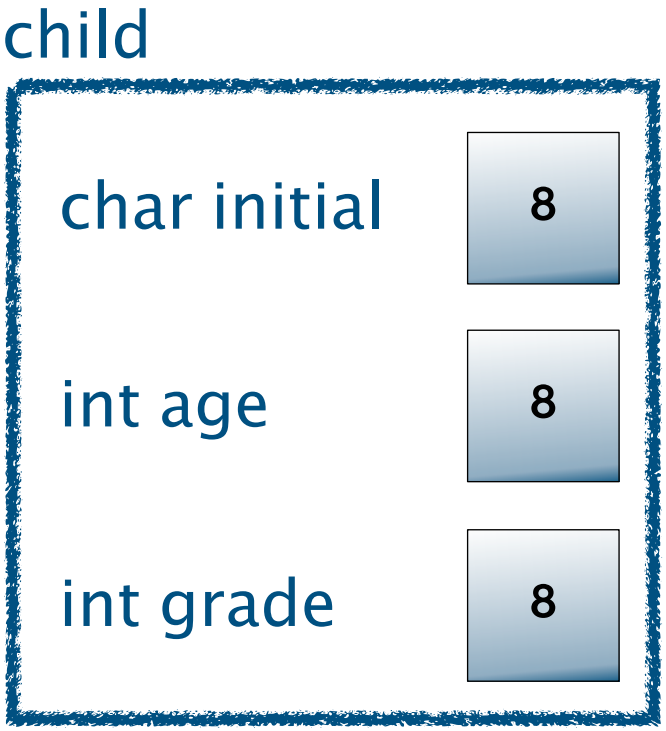
```
void main() {

        persona1.eta=25;
        strcpy(persona1.cognome,"ROSSI");
        strcpy(persona1.nome,"MARIO");
        persona1.altezza=175;

        printf("Inserisci l'eta':\n");
        scanf("%d", &persona2.eta);

        printf("\nInserisci il cognome:");
        scanf("%s", persona2.cognome);

        printf("\nInserisci il nome:");
        scanf("%s", persona2.nome);

        persona2.altezza= persona1.altezza - 5;
                                                //-->
```

Esercizio S_1 - Persona

```
    if (   persona1.eta < persona2.eta   )

       printf("\nLa persona piu' giovane e' %s %s",
                    persona2.cognome, persona2.nome);


    else

       printf("\nLa persona piu' giovane e' %s %s",
                    persona1.cognome, persona1.nome);


}
```

Dichiarazione

```
struct child{
    char initial;     /* iniziale del cognome     */
    int age;          /* età                      */
    int grade;        /* voto di informatica      */
};
struct child boy;

typedef struct{
    char initial;
    int age;
    int grade;
} child;

child boy;
child boy[12];
```



Basandosi sulla struttura definita nell'esercizio S_1, definire un archivio di 25 persone ED una persona singola.

Acquisire i dati delle 25 persone e successivamente trovare il più giovane, memorizzandolo nella struttura singola e mostrandolo a video.

```
#define MAX_PERS 25

typedef struct {
        char cognome[40];
        char nome[40];
        int  eta;
        int  altezza;
} anagrafe;
```

Esercizio S_1_1 - Giovane

```
anagrafe persone[MAX_PERS], giovane;

int i;

void main() {
        strcpy(giovane.cognome,"");
        strcpy(giovane.nome,"");
        giovane.eta=999;
        giovane.altezza=0;

        for (i=0; i< MAX_PERS; i++)     {

                scanf("%s", persone[i].cognome);
                scanf("%s", persone[i].nome);
                scanf("%d", &persone[i].eta);
                scanf("%d", &persone[i].altezza);

        }                                                    //-->
```

Esercizio S_1_1 - Giovane

```
    for (i=0; i< MAX_PERS; i++)
        if ( persone[i].eta < giovane.eta)

            giovane=persone[i];



    printf("\nLa persona piu' giovane e' %s %s",
                giovane.cognome, giovane.nome);


}
```

Moltiplicare 2 numeri razionali (definiti da numeratore e denominatore) semplificando il risultato.

Esercizio S_2 - Prodotto di razionali

```
#include <stdio.h>
#include <stdlib.h> /* per la funzione abs() */

typedef struct
{
  int numeratore;
  int denominatore;
} Razionale;

typedef enum {false, true} boolean;

Razionale p,q,r;
int min,i;
boolean trovato = false;

main()
{
```

Esercizio S_2 - Prodotto di razionali

```
p.numeratore = -51;
p.denominatore = 4;   //printf("\nDammi il numeratore: ");
                      //scanf("%d", &p.numeratore);
q.numeratore = 18;    //...
q.denominatore = 34;


r.numeratore = p.numeratore * q.numeratore;
r.denominatore = p.denominatore * q.denominatore;


if(abs(r.numeratore) < abs(r.denominatore)){
  min = abs(r.numeratore);
}
else{
  min = abs(r.denominatore);
}
```

Esercizio S_2 - Prodotto di razionali

```
i = min;
  while (i > 1 && (trovato==false))
  {
    if ( r.numeratore % i == 0   /* divisibile per i */
                 && r.denominatore % i == 0    )
    {
       r.numeratore = r.numeratore / i;
       r.denominatore = r.denominatore / i;
       trovato = true;
    }
    i = i - 1;
  }


   printf("\n%d|%d\n", r.numeratore, r.denominatore);

}
```

Esercizio S_3 - Gestione vettore

Scrivere un menu che permetta l'inserimento, la visualizzazione e l'eliminazione di interi da un elenco di massimo 10 elementi, utilizzando le strutture.

Esercizio S_3 - Gestione vettore

```c
#include <stdio.h>

#define MAX 10


typedef enum {VUOTA, PIENA} dispon;

typedef struct {
        int elemento;
        dispon stato;
} cella;


cella vett[MAX];

int i, operaz=-1, numero;
```

Esercizio S_3 - Gestione vettore

```
void main()
{
        //inizializzazione

        for (i=0;i<MAX;i++)
             vett[i].stato=VUOTA;


         while (operaz!=0)
         {
                  /* richiesta operazione da eseguire */
                  printf("\nInserisci l'operazione
                            (0 fine, 1 aggiungi,
                            2 elimina, 3 visualizza): ");

                  scanf("%d",&operaz);
```

Esercizio S_3 - Gestione vettore

```
                /* elaborazione */

switch (operaz)
{
        case 0:
                printf("\nFine Programma!");
                break;

        case 1:
                /* richiesta numero sul quale operare */
                printf("\nInserisci il numero positivo: ");
                scanf("%d",&numero);
```

Esercizio S_3 - Gestione vettore

```
/* ricerca cella disponibile */
i=0;

while(i<MAX && vett[i].stato==PIENA)
        i++;
/* verifica inserimento */
if (i==MAX)
{
        printf("\nNon ci sono celle disponibili!");
}
else
{
        vett[i].elemento=numero;
        vett[i].stato=PIENA;
}

break;
```

```
case 2:
        /* richiesta numero sul quale operare */
        printf("\nInserisci il numero positivo: ");
        scanf("%d",&numero);

        /* ricerca numero nell'array */
        i=0;
        while(i<MAX && (vett[i].stato==VUOTA) ||
                (vett[i].stato==PIENA
                        && (vett[i].elemento!=numero))   )
                i++;
        if (i==MAX)
                printf("\nNumero non presente!");
        else
                vett[i].stato=VUOTA;

        break;
```

Esercizio S_3 - Gestione vettore


```
  case 3:
          /* visualizzazione array */

         for(i=0;i<MAX;i++)
         {
         if (vett[i].stato==PIENA)
                printf("\n vettore %d-esimo (PIENA) = %d",
                                        i, vett[i].elemento);
         else
                printf("\n vettore %d-esimo (VUOTA)", i);
         }

          break;
  default:
          printf("\nOperzione non supportata!");
          break;
   }
  }
}
```

Esercizio S_4 - Hangman

Implementare il "gioco dell'impiccato".

Esercizio S_4 - Hangman


```c
#include <stdio.h>
#include <String.h>

#define MAX 15
#define TENTATIVI 10


typedef enum {NASCOSTO, INDOVINATO, ND} stato;

typedef struct {
        stato situazione;
        char  lettera;
} posizione;


posizione indovina[MAX];
char  parola[MAX], lettera;


int   fine=0, i, iter=0, lungh;
```

Esercizio S_4 - Hangman

```
int main()
{
  fflush(stdin);
  printf("\nInserisci una parola: ");
  fflush(stdin);
  scanf("%s", parola);
  lungh=strlen(parola);

  /* inizializza le lettere come nascoste o non disponibili */

  for (i=0; i< MAX; i++)
          if (i < lungh)
          {
                  indovina[i].situazione = NASCOSTO;
                  indovina[i].lettera    = parola[i];
          } else {
                  indovina[i].situazione = ND;
                  indovina[i].lettera    = 0;
          }
```

Esercizio S_4 - Hangman

```
do {

        /* stampa la parola nella situazione attuale */
        printf("\n");
        for (i=0; i<lungh; i++)
                if (indovina[i].situazione==INDOVINATO)
                        printf("%c ", indovina[i].lettera);
                else if (indovina[i].situazione==NASCOSTO)
                        printf(". ");


        /* visualizzazione tentativi */
        printf("\t");
        for (i=0; i<iter; i++)
                printf("-");
        for (i=iter; i< TENTATIVI; i++)
                printf("*");
```

Esercizio S_4 - Hangman

```
        /* inserimento lettera */
        printf("\tInserisci lettera: ");
        fflush(stdin);
        scanf("%c", &lettera);

        /* aggiornamento situazione */
        for (i=0; i<lungh; i++)
                if ((indovina[i].situazione!=INDOVINATO)
                                && (indovina[i].lettera==lettera))
                    indovina[i].situazione=INDOVINATO;
        /* verifica termine gioco */
        fine=1;
        for (i=0; i<lungh; i++)
                if (indovina[i].situazione!=INDOVINATO)
                        fine=0;
        iter++;

} while(  (iter < TENTATIVI) && (fine==0)   );
```

Esercizio S_4 - Hangman

```
        if (fine==1)
                printf("\nHai vinto (%s)!", parola);
        else
                printf("\nHai perso (%s)!", parola);

}
```

Si scriva un programma che calcoli i giorni trascorsi fra la data predefinita 08/03/1972 e una data successiva inserita dall'utente, trascurando gli anni bisestili.

Esercizio S_5 - Giorni

```c
#include <stdio.h>

 typedef struct
 {
   unsigned giorno;
   unsigned mese;
   unsigned anno;
 } Data;

 int deltaAnni = 0;  /* tutti i delta sono in giorni */
 int deltaMesi = 0;
 int deltaGiorni = 0;
 int segno = 1;
 int i;

 Data fine, inizio = {8,3,1972};
```

Esercizio S_5 - Giorni

```
int main() {

      do {
        printf("\nGiorno: ");
        scanf("%d", &fine.giorno);
      } while(fine.giorno < 1 || fine.giorno > 31);

      do {
        printf("\nMese: ");
        scanf("%d", &fine.mese);
      } while(fine.mese < 1 || fine.mese > 12);

      do {
        printf("\nAnno: ");
        scanf("%d", &fine.anno);
      } while(fine.anno < inizio.anno);
```

Esercizio S_5 - Giorni

```
deltaAnni = (fine.anno - inizio.anno)*365;

if (fine.mese < inizio.mese){segno = -1;}

for (i=inizio.mese; i!=fine.mese; i=i+segno){
  switch(i){
  case 4:
  case 6:
  case 9:
  case 11: //mesi con 30 gg
    deltaMesi = deltaMesi + segno*30;
    break;
```

```
      case 2: //febbraio ha 28 gg
        deltaMesi = deltaMesi + segno*28;
        break;
      default: //31 gg
        deltaMesi = deltaMesi + segno*31;
      }
   }

   deltaGiorni = fine.giorno - inizio.giorno;

   printf("La differenza e` di %d giorni\n",
                          deltaAnni+deltaMesi+deltaGiorni);
}

//A casa provare a considerare gli anni bisestili
//(Quali anni sono bisestili? Uno ogni 4 o...?)
```

Si utilizzi un vettore di strutture per memorizzare gli orari delle lezioni nel formato:

<nome corso, giorno sett, orario>

Il programma faccia uso di un tipo enumerato GIORNI associato ai giorni della settimana. Si evidenzino i giorni nei quali lo studente segue le lezioni di un corso inserito da tastiera.

Esercizio S_6 - Orario lezioni


```
#include <stdio.h>
#define MAX_LEZIONI 20

typedef enum {LUNEDI, MARTEDI, MERCOLEDI, GIOVEDI,
              VENERDI, SABATO, DOMENICA} g_sett;

typedef struct {
        g_sett  giorno;
        char    corso[50];
        char    ora[50];
} lezione;
lezione orario[MAX_LEZIONI];

char risp, ricerca[50];

int cont=0, i;
g_sett gg;
```

Esercizio S_6 - Orario lezioni

```
void main() {

  for (gg=LUNEDI; gg<=DOMENICA; gg++) {

     do {
          printf("Vuoi inserire una lezione per ");
          switch (gg) {
               case LUNEDI:
                         printf("\tLunedi    ");
               break;
               /* .... */
               case DOMENICA:
                         printf("\tDomenica ");
               break;
          }
          printf("(y/n)?");
```

```
            scanf("%c", &risp);
            fflush(stdin);

            if (risp=='y') {
                    orario[cont].giorno=gg;

                    printf("\nCorso: ");
                    gets(orario[cont].corso);

                    printf("Ora: ");
                    gets(orario[cont].ora);

                    cont++;
                }
  }while (risp!='n');
} //end for
```

Esercizio S_6 - Orario lezioni

```
printf("\nLezioni del MERCOLEDI': ");


for (i=0; i<cont; i++)
        if ( (orario[i].giorno==MERCOLEDI))
                printf("\t corso: %s \t ora: %s",
          orario[i].corso, orario[i].ora);
```

Esercizio S_6 - Orario lezioni

```
  printf("\nInserisci corso da ricercare: ");
  gets(ricerca);
  for (i=0; i<cont; i++)
    if ( strcmp(orario[i].corso,ricerca)==0 )
       switch (orario[i].giorno) {
          case LUNEDI:
             printf("\n giorno: Lunedi' \t ora: %s",
                                          orario[i].ora);
            break;

             /* .... */

          case DOMENICA:
             printf("\n giorno: Domenica \t ora: %s",
                                          orario[i].ora);
            break;
       }
}
```

Provare a strutturare il campo ora, introducendo un nuovo tipo ora

<int ore; int minuti>

e inserendo nella struct prima definita l'ora di inizio e l'ora di fine.

Esercizio S_7 - Bubble Sort

Ordinare un array di strutture utilizzando il ''bubble sort''.

Esercizio S_7 - Bubble Sort

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

#define MAX 10


typedef struct {
        char  codice[10];
        char  descrizione[100];
        int   qta;
        int   prezzo;
} articolo;


articolo magazzino[MAX], tmp;

int i, j, k=1, scambio, risp;
```

Esercizio S_7 - Bubble Sort

```
void main() {
      for(j=0; j<MAX; j++) { /* inserimento articoli */
                  printf("\nArticolo %d: ", j);
                  printf("\n\t codice: ");
                  scanf( "%s", magazzino[j].codice      );

                  printf("\n\t descrizione: ");
                  scanf( "%s", magazzino[j].descrizione );

                  printf("\n\t quantita': ");
                  scanf( "%d", &magazzino[j].qta        );

                  printf("\n\t prezzo: ");
                  scanf( "%d", &magazzino[j].prezzo     );
        }
```

Esercizio S_7 - Bubble Sort

```
/* scelta dell'ordinamento */

printf("\nOrdinare per: 0 quantita', 1 prezzo, 2 codice");
scanf("%d", &risp);
switch (risp) {
    case 0: //ordinamento in base alla quantita'
            //...(prossima pagina)
      break;
    case 1: //ordinamento in base al prezzo decrescente
            //..(provare a casa e controllare sul sorgente)
      break;
    case 2: //ordinamento in base al codice
            //..(provare a casa e controllare sul sorgente)
      break;
    default:
             printf("\nOpzione non riconosciuta!");
}
```

Esercizio S_7 - Bubble Sort

```
do
{
    scambio=0;
    for(i=0; i< (MAX-k); i++)
    {
        if (magazzino[i].qta > magazzino[i+1].qta)
        {
                tmp=magazzino[i+1];
                magazzino[i+1]=magazzino[i];
                magazzino[i]=tmp;
                scambio=1;
        }
    }
    k++;

} while (scambio==1);
```

Esercizio S_7 - Bubble Sort

```
/* visualizzazione risultato */
printf("\n\n");
for(j=0; j<MAX; j++) {

   printf("\nArticolo %d: ", j);

   printf( "\n\t codice: %s", magazzino[j].codice    );

   printf( "\n\t descrizione: %s",
                          magazzino[j].descrizione   );

   printf( "\n\t quantita': %d", magazzino[j].qta    );

   printf( "\n\t prezzo: %d", magazzino[j].prezzo    );
  }
}
```