Università degli Studi di Cagliari
Corso di Laurea in Ingegneria Biomedica

ELEMENTI DI INFORMATICA

http://people.unica.it/gianlucamarcialis/

A.A. 2020/2021

Docente: **Gian Luca Marcialis**

LINGUAGGIO C
Tipi di dati strutturati

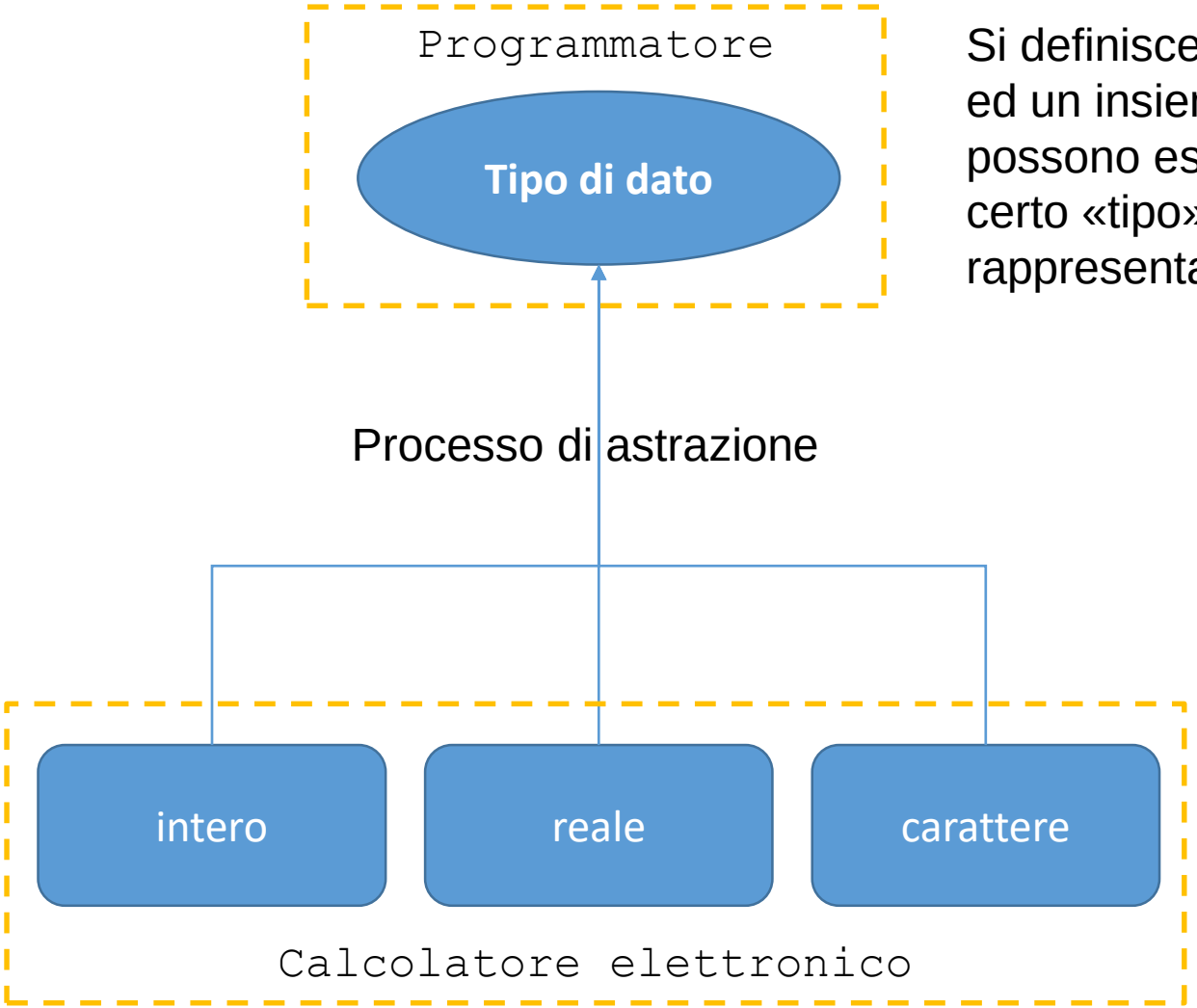
Sommario

- Typedef e struct
- Accesso alle variabili strutturate
- Esercizi

Classificazione dei tipi di dati

- Tipi semplici
  - L'uso di una variabile numerica è naturale
    - Es. una velocità, una temperatura
- Tipi strutturati
  - Indicano "raggruppamenti" di tipi semplici e strutturati
  - Es. Data: giorno.mese.anno
- In C esistono tipi semplici predefiniti
- Ma l'utente ha la libertà di definirne nuovi, sia semplici, sia strutturati

Perché i «tipi di dato»

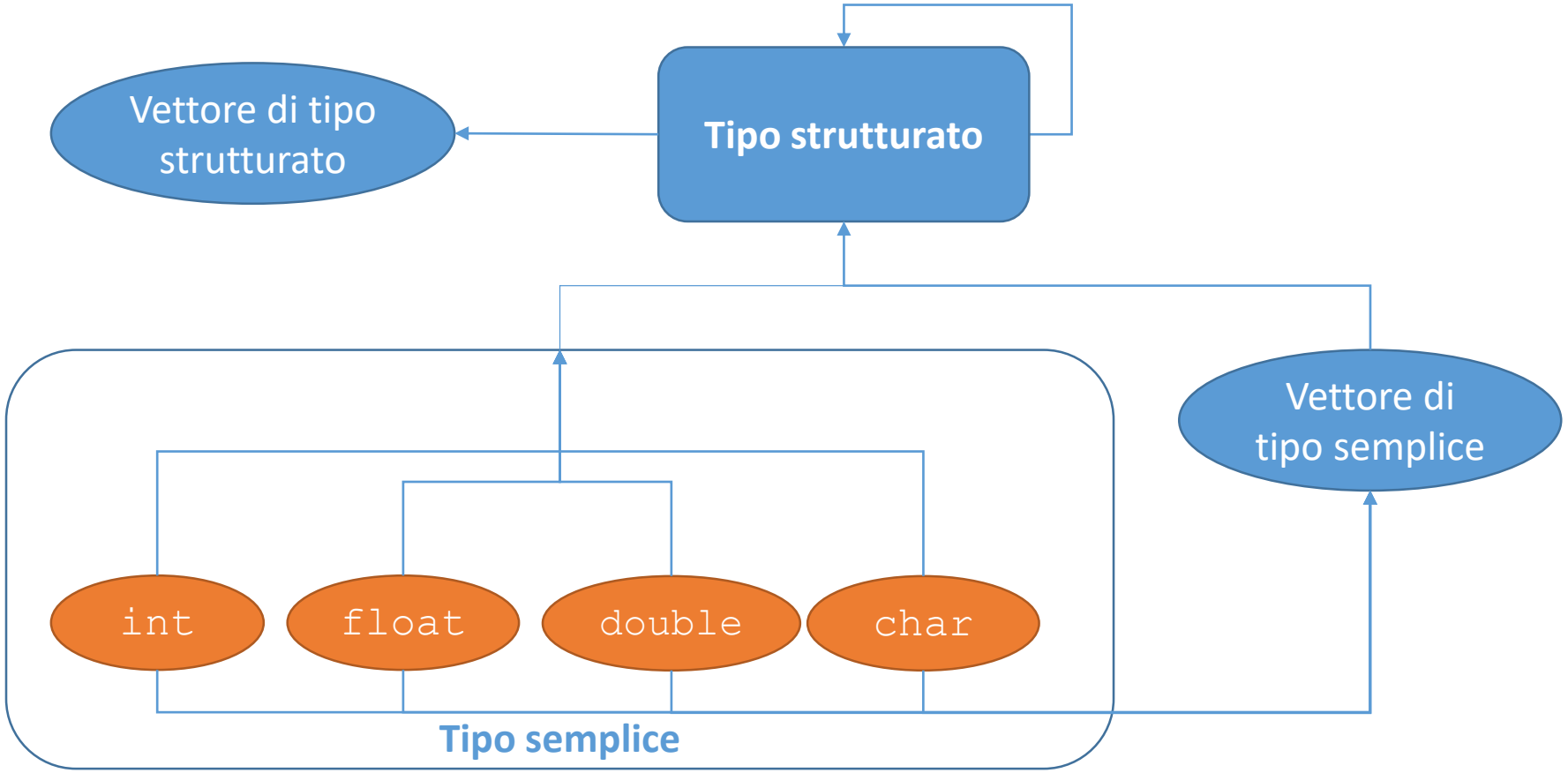


Si definisce un **dominio di esistenza** ed un insieme di **operazioni** che possono essere svolte sui dati di un certo «tipo» indipendentemente dalla rappresentazione «fisica»

Necessitano di rappresentazione binaria opportuna dentro il calcolatore

- Intero: segno e valore complemento a 2, eccesso
- Reale: virgola mobile, virgola fissa
- Carattere: ASCII…
- Quanti bit?

Visione grafica di un tipo strutturato



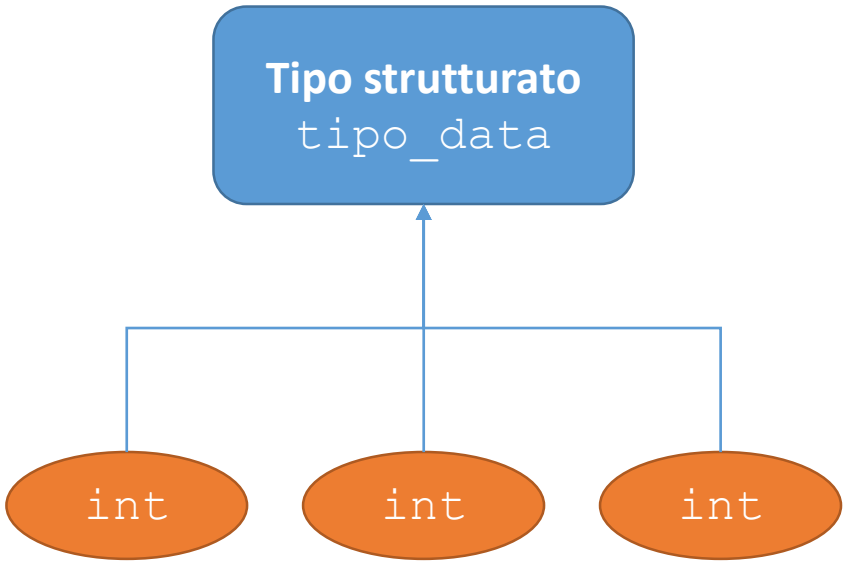
Il costruttore `struct`

- Il costruttore struct serve a definire tipi “strutturati”, cioè composti da tipi semplici (o da ulteriori tipi strutturati)
- Questi tipi “incapsulano” i dati a disposizione, e su essi possono essere definite delle operazioni implementando così dei veri e propri “tipi di dati astratti” (es. liste, code, pile o “stack”)
- Sintassi:
  - **`typedef struct`**
  - `{`
    - `Componenti della struttura;`
  - `} NomeTipo;`

Qualche esempio

- Si voglia definire un nuovo tipo `tipo_data` attraverso la composizione di tre variabili intere `giorno`, `mese`, `anno`.
- Usando il costrutto `struct`:

```
typedef struct
{
  int giorno, mese, anno;
} tipo_data;
```



Accesso ai valori di `tipo_data`

- Con la **definizione** precedente, possiamo **dichiarare** una variabile di tipo `tipo_data`:

```
tipo_data data1, data2;
```

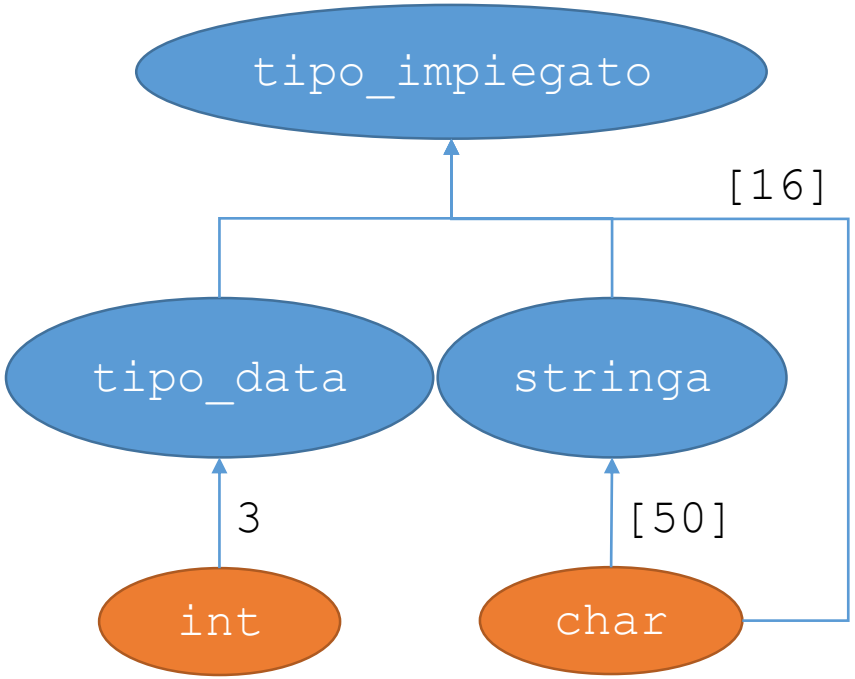
- Su essa possiamo definire operazioni più o meno complesse (per ora non ci interessa…)
- L'accesso ad una delle variabili "incapsulate" dalla nostra struttura dati avviene intercalando il nome della variabile dal nome della struttura con un punto '.' :
  - `x = data1.giorno;`
    - assegna ad `x` il valore della variabile `giorno` di `data1`
  - `data2.mese = y;`
    - assegna alla variabile `mese` di `data2` il valore di `y`
- Nota bene:

definizione di **tipo** `!=` dichiarazione di **variabile**

Definizione di strutture complesse

- Definiamo il tipo `tipo_impiegato`:

```
typedef struct
{
    stringa nome, cognome;
    tipo_data data_di_assunzione;
    char codicefiscale[16];
} tipo_impiegato;
```



Esercizi

- Definire il tipo `tipo_dipartimento`, caratterizzato:
  - Da una stringa che ne indichi il nome, di max 100 caratteri;
  - Dal numero corrente di impiegati;
  - Da un vettore di impiegati, con un max di 50 impiegati;
  - Dalla data rispetto a cui le altre informazioni sono aggiornate.
- Definire il tipo `tipo_cartella` caratterizzata:
  - Da una variabile di tipo `tipo_dati_personali` con i dati del possessore (nome, età, anno di nascita)
  - Da una variabile di tipo `tipo_info_biometriche` che indica i valori biometrici basati su determinati esami (emocromo, formula leucocitaria, biririubinemia, transaminasi)

Esercizio sui tipi

- Scrivere un programma C in cui vengano assegnati gli slot di una variabile strutturata di tipo `tipo_impiegato` che vengano poi stampati a video. In particolare:
  - Si scriva una funzione che restituisca una variabile di tipo `tipo_impiegato` con tutti i suoi slot assegnati da tastiera
  - Si scriva una funzione che, ricevendo a video una variabile di tipo `tipo_impiegato`, stampi a video tutti i suoi slot

Soluzione
Intestazione e definizione dei tipi

```
/*Inserimento di dati in una variabile
strutturata*/

#include <stdio.h>


/*Le strutture dati si definiscono
fuori dalla main*/

typedef char stringa[20];


typedef struct
{
        int giorno, mese, anno;
} tipo_data;
```

```
typedef struct
{
        stringa nome, cognome;
        tipo_data data_di_assunzione;
        stringa codicefiscale[16];
} tipo_impiegato;
```

Funzione main

```
int main()
{
    tipo_impiegato impiegato;

    impiegato=leggiImpiegato();
    stampaImpiegato(impiegato);

    return 0;
}
```

Soluzione
Corpo della main: stampa

```
tipo_impiegato leggiImpiegato(void)
{
    tipo_impiegato i;
    /*
        lettura degli
        slot
    */
    return i;
}
```

```
void stampaImpiegato(tipo_impiegato i)
{
    /*scrittura degli
    slot di i
    */

    return;
}
```

Esercizio n. 1

- Definire il tipo strutturato `tipo_studente` composto da una stringa indicante il nome, una stringa indicante il cognome (entrambe di 20 caratteri), un intero indicante il numero di matricola, un reale indicante la media voti, un intero indicante il numero di esami dati.
- Scrivere un programma C che, dichiarata una variabile di tipo `tipo_studente`, assegni a ciascuno dei suoi slot un valore da tastiera.
- Ricevuto poi da tastiera il voto relativo ad un nuovo esame, aggiorni lo slot relativo alla media secondo la formula:

$$mediaNuova = \frac{mediaCorrente \cdot numeroEsami + votoNuovoEsame}{numeroEsami + 1}$$

- Aggiorni inoltre lo slot relativo al numero di esami dati.
- Stampi infine tutti i dati dello studente presenti nella variabile assegnata, indicando sia la media vecchia che la media aggiornata.
- Si richiede modularizzazione del codice definendo le funzioni che possono ritenersi utili per risolvere il problema.

Esercizio n. 2

- Sostituire, nella definizione del tipo `tipo studente`, lo slot relativo alla media con un vettore di due reali per cui la posizione 0 corrisponda alla media «vecchia» (precedente l'ultimo esame dato) e la posizione 1 alla media «corrente» (con l'ultimo esame dato).
- Riscrivere il programma dell'esercizio n.1 alla luce di questa modifica.
- Suggerimenti:
  - Ad inizio programma la posizione 0 dello slot sarà inizializzata a 0 e la media letta sarà assegnata alla posizione 1
  - Dopo la lettura del nuovo voto, la media in posizione 1 sarà assegnata alla posizione 0 e la media alla posizione 1 sarà riassegnata secondo la formula di aggiornamento

Esercizio n. 3

- Definire un tipo di dato `triangolo_rettangolo` caratterizzato da tre valori reali relativi ai suoi lati (i due cateti e l'ipotenusa).
- Dichiarato un vettore di massimo 50 variabili di tipo `triangolo_rettangolo`, gli si assegnino i valori degli slot da file «dati.txt».
- Per ogni triangolo memorizzato, stampare a video i lati compatibili con il teorema di Pitagora.

Vincolo

- Modularizzare il codice in modo che la `main` utilizzabile sia la seguente:

```
int main()
{
	triangolo_rettangolo tr[N];
	int i, n;

	n=assegnaTriangoli(tr,"dati.txt");

	for(i=0; i<n; i++)
		if(rettangolo(tr[i]))
			stampaLati(tr[i]);

	return 0;
}
```

Esercizio n. 4

- Scrivere un programma che legge un file di testo `geometri.txt` così formattato:

  `<Forma Geometrica> v1 [v2 v3]`

- Dove
  - `<Forma Geometrica> = Triangolo | Quadrato | Rettangolo`
  - `v1 [v2 v3]` sono al massimo tre valori reali

- Per ogni riga di ingresso, il programma calcola e scrive, nel file di uscita `risultati.txt`, ed in modalità “append”, la forma della figura e il suo perimetro
- Si sviluppi la soluzione in forma **modulare**
  - Attraverso procedure e funzioni

Esempio di file geometri.txt

```
Triangolo 34.2 45.1 90.4
Rettangolo 40.1 20.9
Rettangolo 12.1 55.5
Quadrato 45.0
```

Vincoli per scrivere la soluzione

- Si definisca una struttura dati `Forma_Geometrica`, caratterizzata da:
  - un vettore di caratteri `nome_forma` indicanti il tipo (“Triangolo”; “Quadrato”; “Rettangolo”);
  - un vettore di **`float`** `lati` di max 3 posizioni con i valori dei lati
  - un valore **`float`** `perimetro`;
  - un intero `numero_lati` tale che:
    - Se 1 la forma è “Quadrato”
    - Se 2 la forma è “Rettangolo”
    - Se 3 la forma è “Triangolo”
    - Altrimenti la forma non è definita, e contenga il valore 0;
- Si scriva inoltre:
  - una funzione `perimetro_forma` che calcoli il perimetro per ognuna delle tre forme: essa riceve in ingresso il vettore dei lati ed il numero di lati, e restituisce un valore reale pari al perimetro
  - una funzione `leggi_forma` che legga il tipo di forma da file, calcoli il numero opportuno di lati restituisca una variabile di tipo `Forma_Geometrica` aggiornata
  - una funzione `stampa_forma` che, ricevendo in ingresso una variabile di tipo `Forma_Geometrica`, scriva la forma e stampi il perimetro nel file su file in modalità “append”

Definizione della struttura dati richiesta

```
typedef struct
{
 char nome_forma[50];
 float lati[3];
 float perimetro;
 int numero_lati;
} Forma_Geometrica;
```

Implementazioni delle funzioni richieste: perimetro della forma

```
float perimetro_forma(float *lati, int num_lati)
{
  switch(num_lati)
  {
      case 1: return 4*lati[0];
      case 2: return 2*(lati[0]+lati[1]);
      case 3: return lati[0]+lati[1]+lati[2];
  }
  return 0.0;
}
```

Aggiornamento della forma

```
Forma_Geometrica leggi_forma(FILE *fp)
{
  int i;
  Forma_Geometrica forma;

  fscanf(fp,"%s",&(forma.nome_forma[0]));
  if(!strcmp(forma.nome_forma,"Triangolo"))
        forma.numero_lati=3;
  else
        if(!strcmp(forma.nome_forma,"Rettangolo"))
                forma.numero_lati=2;
        else
                if(!strcmp(forma.nome_forma,"Quadrato"))
                        forma.numero_lati=1;
                else
                {
                        forma.numero_lati=0;
                        printf("\nForma non definita\n");
                }
  for(i=0; i<forma.numero_lati; i++)
        fscanf(fp,"%f",&(forma.lati[i]));

  return forma;
}
```

Stampa della forma col perimetro

```
void stampa_forma(Forma_Geometrica forma)
{
  FILE *out;
  out=fopen("risultati.txt","a");
  if(out==NULL) return;

  fprintf(out,"%s %f\n",forma.nome_forma,forma.perimetro);

  fclose(out);
}
```

Soluzione: vista top-down

```
/*Programma per la stampa del perimetro di una forma geometrica su file*/

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

int main()
{
  FILE *in;
  Forma_Geometrica forma;
  /*Apri il file geometri.txt */
  while (!feof(in))        /* Finché il file non è finito*/
  {
        /*Leggi la forma geometrica*/
        /*Se la forma è definita:
                Calcola il perimetro della forma letta
                Stampa il perimetro della forma su file risultati.txt
          Altrimenti
                Esci dal ciclo*/
  }
  /*Chiudi il file geometri.txt*/

  return 0;
}
```

/*Apri il file geometri.txt*/

```
in=fopen(“geometri.txt”,”r”);
if(in==NULL) return 0;
```

/*Chiudi il file geometri.txt*/

```
fclose(in);
```

Ritorniamo al main

```
/*Programma per la stampa del perimetro di una forma geometrica su file*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
   FILE *in;
   Forma_Geometrica forma;
   in=fopen("geometri.txt","r");
   if(in==NULL) return 0;
   while (!feof(in)) /* Finché il file non è finito*/
   {
    forma=leggi_forma(in);
    if(forma.numero_lati!=0)
    {
        forma.perimetro=perimetro_forma(forma.lati,forma.numero_lati);
        stampa_forma(forma);
    }
    else
        break;
   }
   fclose(in);
   return 0;
}
```

Esercizio n. 5 (file, stringhe)

- Scrivere un programma C che letto un numero intero n da tastiera, generi n file caratterizzati dal nome «file1.txt», «file2.txt», …, «filen.txt». In ciascuno di questi file si scriva un numero intero casuale compreso tra 0 e 100. Ciascun file conterrà un numero diverso.
- Per esempio se n=10, si avranno 10 file «file1.txt»… «file10.txt»
- Suggerimenti
  - Per generare l'intero casuale richiesto, si usi la funzione `rand()` che genera un intero casuale compreso tra 0 e il massimo intero rappresentabile espresso tramite l'identificatore costante `RAND_MAX`. La funzione è presente in «stdlib.h».
  - Si ricordi la funzione `sprintf()` per alterare una stringa.

Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
	FILE *f;
	int r, n, i;
	char nomefile[50];

	printf("Inserire un numero: ");
	scanf("%d",&n);

	for (i=1; i<=n; i++)
	{
		sprintf(nomefile,"file%d.txt",i);
		f=fopen(nomefile,"w");
		r= rand() % 101; /* il resto è compreso tra 0 e 100 */
		fprintf(f,"%d",r);
		fclose(f);
	}

	return 0;
}
```

Esercizio n. 6
(tipi strutturati, file, funzioni, stringhe)

- Si scriva un programma che cerchi e legga, finché esistono, i file chiamati «file1.txt», …, «filen.txt», memorizzi le loro proprietà in un vettore di tipo `FILE_PROPERTIES` e stampi a video la loro somma.
- Il tipo strutturato `FILE_PROPERTIES` è caratterizzato da:
  - Una stringa di 20 caratteri rappresentante il nome di un file, chiamata `nomeFile`;
  - Un puntatore a file impostato a NULL se il file non è aperto, chiamato `fp`;
  - Un intero rappresentante l'intero contenuto nel file, chiamato `x`.
- Si scrivano le seguenti funzioni:
  - `void generaNome(int i, char*nome)`: immesso un valore `i`, assegna alla stringa `nome` il valore «file`i`.txt», con `i` valore immesso.
  - `FILE_PROPERTIES crea(char*nome)`: restituisce una variabile di tipo `FILE_PROPERTIES` il cui slot `nomeFile` corrisponde alla stringa passata in ingresso. Essa apre il file con il nome indicato e assegna i corrispondenti slot della variabile restituita.
  - `int calcolaSomma(FILE_PROPERTIES* files, int n)`: calcola la somma degli `n` interi presenti in ciascuno degli slot corrispondenti del vettore di variabili di tipo `FILE_PROPERTIES` passato in ingresso.

Per saperne di più

- Ceri, Mandriola, Sbattella, *Informatica – arte e mestiere*, Cap. 5, McGraw-Hill

- Kernighan, Ritchie, *Il linguaggio C*, Cap. 2, Pearson-Prentice Hall

Curiosità: array di array

- Cos'è questo?

```
char mesi[12][30];
```

- Si tratta di un array di 12 stringhe di 30 caratteri ciascuna

Esercizio

- Dato il seguente array di stringhe

```
char mesi[12][15]={"Gennaio", "Febbraio", "Marzo",
"Aprile", "Maggio", "Giugno", "Luglio", "Agosto",
"Settembre", "Ottobre", "Novembre", "Dicembre"};
```

- Scrivere un programma C che, richiesto da tastiera il numero del mese, compreso tra 1 e 12, stampi a video l'espressione «`Sei nato in <nomeDelMeseCorrispondente>!`»

- Per esempio, se il numero introdotto è 1, il sistema risponderà con «`Sei nato in Gennaio!`»

Soluzione

```
/***Array di stringhe: in quale mese sei nato?***/


#include <stdio.h>
int main()
{               /* assegnazione di un array di stringhe */
        char mesi[12][15]={"Gennaio", "Febbraio", "Marzo",    "Aprile", "Maggio",
"Giugno", "Luglio", "Agosto",     "Settembre", "Ottobre", "Novembre", "Dicembre"};
        int mese;

        printf("In che mese dell\' anno sei nato? (1-12)\n");
        scanf("%d",&mese);

        printf("Sei nato in %s!\n",mesi[mese-1]); /*indicizzazione*/
        return 0;
}
```