UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Fondamenti di Informatica

Ing. Alba Amato, PhD

alba.amato@unina2.it

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

IL LINGUAGGIO C

Lucidi tratti da:

Alla scoperta dei fondamenti dell'informatica. Un viaggio nel mondo dei bit
di Angelo Chianese, Vincenzo Moscato, Antonio Picariello

capitolo 6 -par. 6.1, 6.2, 6.3, 6.4, 6.5, 6.6

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Premessa

- Fu creato agli inizi degli anni ’70 da Dennis Ritchie
- Porta un nome che deriva dal suo predecessore, il *linguaggio B*, progettato da Ken Thomson per implementare il sistema operativo Unix.
- Ritchie e Thomson, successivamente riscrissero il sistema *Unix* interamente in C.
- Si è diffuso molto rapidamente e nel 1989 l’American National Standards Institute (ANSI) completava la definizione del linguaggio producendo il documento noto come ANSI C, al quale fanno riferimento ormai tutti i compilatori per quel linguaggio.
- A differenza degli altri linguaggi ad alto livello consente un agevole accesso alla struttura hardware del sistema di elaborazione.

ANSI
American National Standards Institute

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Caratteristiche Generali

- Il C è un linguaggio:
  - ad alto livello
    - ... ma anche poco astratto
  - strutturato
    - ... ma con eccezioni
  - fortemente tipizzato
    - ogni oggetto ha un tipo
  - semplice (!?)
    - poche keyword
  - *case sensitive*
    - maiuscolo diverso da minuscolo negli identificatori!
  - portabile
  - standardizzato (ANSI)

Caratteristiche Generali

Le caratteristiche principali del linguaggio C sono:

- presenza dei fondamentali costrutti di controllo di flusso:
  - sequenza di istruzioni organizzate in blocchi;
  - decisioni o costrutti selettivi (**if**, **switch**);
  - cicli o costrutti iterativi (**while**, **for**, **do**).
- disponibilità di operazioni su tipi fondamentali (come i caratteri, gli interi, i reali in virgola mobile) e su tipi derivati, costruiti a partire dai tipi fondamentali o da altri tipi derivati, (come l'array, le funzioni, le strutture e le unioni).
- gestione dei puntatori con una aritmetica sugli indirizzi.
- invocazione ricorsiva di una funzione, con chiamate nidificate.
- linguaggio "case sensitive", nel senso che gli identificatori dipendono dal modo in cui sono scritti in quanto lettere maiuscole e minuscole sono considerate diverse.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Vocabolario

Il vocabolario del linguaggio C è costituito da sequenze di lunghezza finita di caratteri trattate come singole entità logiche (è quindi l'insieme delle parole costruite secondo le regole lessicali). In C ci sono sei classi di simboli:

- *separatori ed identificatori*,
- *simboli speciali* tra cui:

```
+ * / = < > ( ) [ ] { } . , ; : ” ? ! % # & ~ ^
```

- *parole chiave*,
- *costanti*,
- *stringhe*.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Separatori

- I caratteri “spazio” ed ENTER (fine linea) sono considerati separatori espliciti,
- gli operatori (aritmetici e di relazione), il punto e virgola e l'operatore dell'assegnazione (=) vengono implicitamente identificati come separatori.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Identificatori

- Gli ***identificatori*** permettono di indicare i nomi di programmi, costanti, variabili e funzioni.
- Un identificatore è una stringa di caratteri che deve soddisfare i seguenti vincoli:
  - deve essere composta da lettere, cifre e dal carattere '_';
  - il primo carattere deve essere una lettera;
  - deve essere costituito da un numero limitato di caratteri;
  - non deve ovviamente contenere al suo interno lo spazio.

A
a
alpha
ALPHA
Alpha
Sol1
Pi_Greco
eq2Grado

Si può usare una delle lettere dell'alfabeto inglese ed una delle dieci cifre arabe.
si possono usare sia lettere maiuscole che minuscole, con significato però diverso

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Simboli Speciali

- I ***simboli speciali*** sono o semplici caratteri o coppie adiacenti di essi ed indicano le operazioni presenti nel linguaggio e tutta la punteggiatura richiesta dalla sintassi.
- Essi sono:
  + - * / (operatori aritmetici)
  < > == <= >= ~= (operatori di confronto)
  , ; . ‘ “ : .. … (delimitatori e punteggiatura)
  ( ) [ ] { } (parentesi)
  && || ! (operatori logici)
  // /* */ (commento)
- il C distingue l’operatore di assegnazione ‘=’ da quello di confronto logico ‘==’.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Parole chiave

- Riservate!
- Nel C standard sono 32:

```
auto      double    int       struct
break     else      long      switch
case      enum      register  typedef
char      extern    return    union
const     float     short     unsigned
continue  for       signed    void
default   goto      sizeof    volatile
do        if        static    while
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Delimitatori di Istruzioni

- Le istruzioni devono essere scritte rispettando alcune regole sintattiche e di punteggiatura.
- L'istruzione deve sempre essere conclusa con un **;** (punto e virgola).
- Si può scrivere più di un'istruzione per riga purché ognuna sia conclusa col **;**.
- Un'istruzione può occupare più di una riga.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Commenti

- Sono testi liberi inseriti all'interno del programma dal programmatore per descrivere cosa fa il programma.
- Non sono processati dal compilatore: servono al programmatore, non al sistema!

- Formato:
  - Racchiuso tra i simboli: /* */
  - Non è possibile annidarli.

- Esempi:

```
/* Questo è un commento corretto! */
/* Questo /*          risulterà un          */  errore */
```

È possibile anche introdurre un commento in una unica riga, usando i caratteri "//", in questo caso è considerato commento tutto quello che inizia con "//" e termina con la fine del rigo

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Costanti

- Le ***costanti*** del linguaggio si dividono in *costanti numeriche* e *costanti di tipo carattere*.
- i numeri trattati dal linguaggio sono di due tipi: interi e reali

Sono costanti intere:
10
300
-1000
Sono costanti reali:
3.400
10.20
+1.99E+30
-30.008
-0.7E-10
9.02E3

Una costante di tipo carattere è racchiusa tra singoli apici. Il valore di una costante carattere è il valore numerico del carattere nel set di caratteri della macchina.
Alcuni caratteri non grafici sono rappresentati come di seguito:
newline \n
tab orizzontale \t
backspace \b
carriage return \r
form feed \f
**backslash **
apice singolo \'

X GLI INTERI: Se la sequenza inizia con '0' (cifra zero), la costante è un numero ottale, altrimenti è decimale. Se inizia con "0x" o "0X" rappresenta invece un intero esadecimale.
Una costante intera che supera il più grande intero rappresentabile dalla macchina, è considerata **long**. Una costante intera seguita da lettera "l" oppure "L" è una costante long.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Le stringhe

- Le ***costanti stringa di caratteri*** sono sequenze di caratteri racchiuse da una coppia di caratteri doppi apici ("), come in **"salve"**.
- Il valore della stringa è dato dalla sequenza di caratteri esclusi i doppi apici che fungono da parentesi.
- Una costante senza caratteri (una coppia di doppi apici) rappresenta la stringa a lunghezza nulla.
  - Tutte le stringhe, anche quando sono stringhe identiche, sono in realtà diverse.

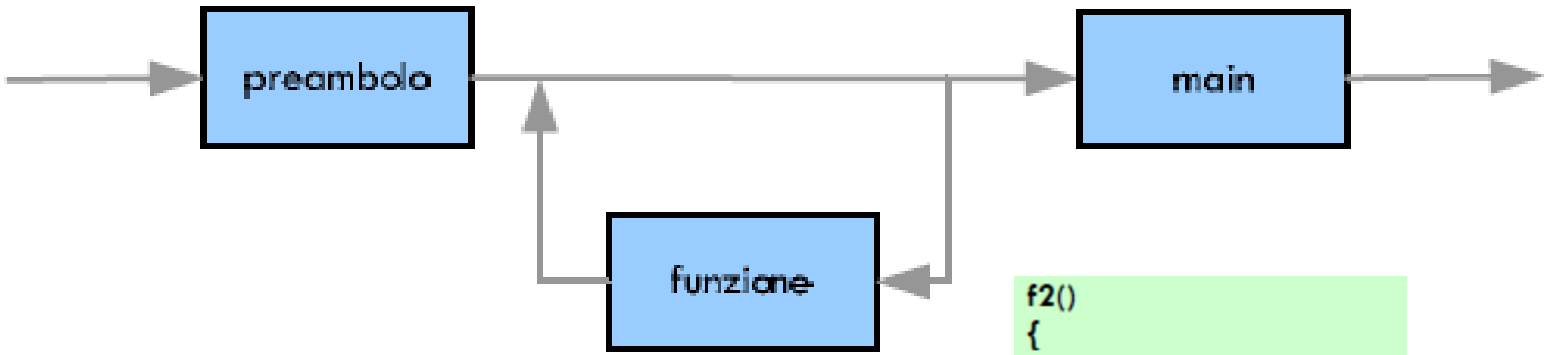
Sono costanti stringhe di caratteri:
"ALFABETO"
"ciao"
"Minuscolo"

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura di un programma C

```
direttive di compilazione
dichiarazione di variabili globali
dichiarazione alias di tipi
dichiarazione prototipi di funzione
```

```
main()
{
variabili locali
sequenza di istruzioni
}
```




```
f2()
{
variabili locali
sequenza di istruzioni
}
.
.
.
fN-1()
{
variabili locali
sequenza di istruzioni
}
```

```
<tipo della funzione> nome_funzione (parametri)
{
  blocco
}
```

Ad un programma C è infine associato un nome che coincide con il nome del file del codice sorgente (con estensione **.c**).

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura di un programma C

- Tutti gli oggetti, con le loro caratteristiche, che compongono il programma devono essere preventivamente dichiarati.

- main
  - è la parola chiave che indica il punto di "ingresso" del programma quando viene eseguito dal S.O.;

  - il suo contenuto è delimitato da parentesi graffe { … }

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura di un programma C

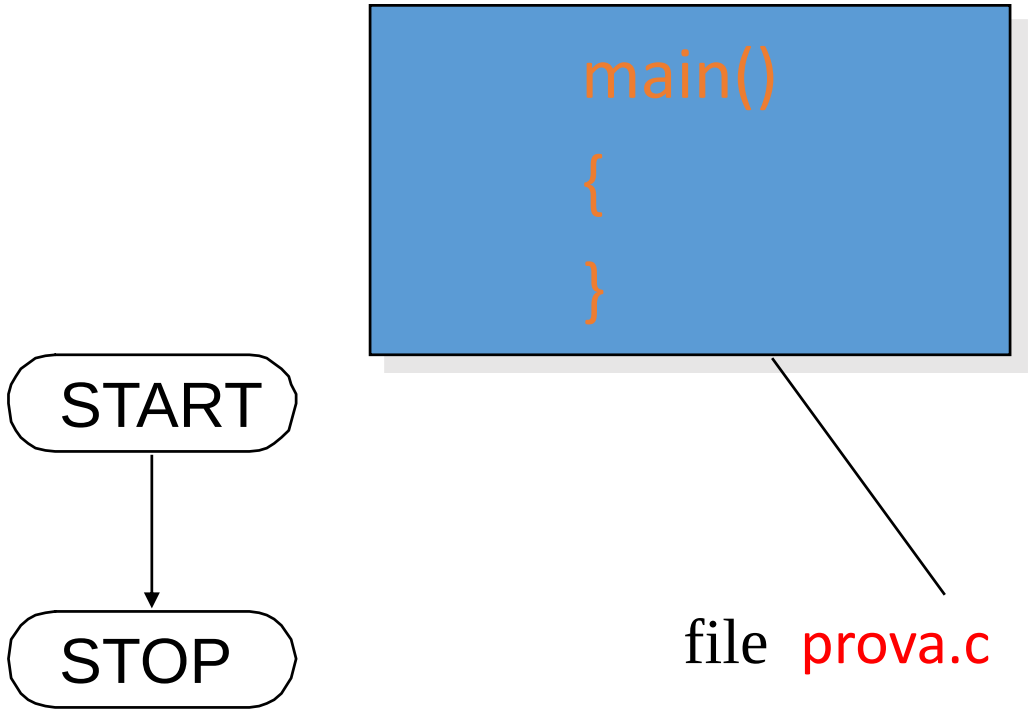
- Parte dichiarativa **locale**:
  - elenco degli oggetti che compongono il main ognuno con le proprie caratteristiche.

- Parte esecutiva:
  - sequenza di istruzioni, ovvero ciò che descriviamo con il diagramma di flusso oppure con lo pseudocodice!

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura di un programma C

- Programma minimo:



Direttive e parte dichiarativa globale

```
main ()
{
```

Parte dichiarativa locale

Parte esecutiva

```
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura di un programma C

```
#include <stdio.h>

int main()
{
  printf("Salve!\n");
  return 0;
}
```

La prima riga del preambolo è una direttiva al compilatore (**#include**): con essa si chiede l'inserimento del file "stdio.h" all'interno del programma, a partire dalla riga in cui si trova la direttiva stessa. "stdio.h" contiene non solo la definizione della funzione **printf()** utilizzata nel corpo del programma, ma di altre funzioni di input ed output.

La funzione **printf()** scrive a video la sequenza di caratteri, racchiusa tra virgolette, specificata tra le parentesi tonde.

La funzione return serve a ritornare un valore (nel caso in oggetto 0) alla fine dell'esecuzione del programma main.
Infatti, avendo dichiarato che il valore di ritorno della funzione deve essere di tipo intero, il compilatore si aspetta che la funzione main restituisca un valore intero alla fine dell'esecuzione.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura di un programma C

- In generale tutti i programmi C includono alcune direttive di inclusione di file .h, detti *include file o header file*, il cui contenuto è necessario per un corretto utilizzo delle funzioni di libreria.
- Il nome dell'include file è, in questo caso, racchiuso tra parentesi angolari (‘<’ e ‘>’): ciò significa che il compilatore deve ricercarlo solo nelle directory specificate nella configurazione del compilatore.
- Se il nome viene invece racchiuso tra virgolette (ad esempio: **"mialibreria.h"**), il compilatore lo cerca prima nella directory corrente, e poi in quelle indicate nella configurazione.
- Le direttive del compilatore non sono mai chiuse dal punto e virgola.
- Tutto quello che si trova tra le due parentesi graffe costituisce invece il *corpo della funzione* (*function body*) e definisce le azioni svolte dalla funzione stessa: può comporsi di definizioni di variabili, di istruzioni e di chiamate a funzione.
- Come già ricordato, tutti i programmi C hanno bisogno di una funzione principale main che fa partire l’esecuzione del programma.
- Un programma termina quando:
  - si raggiunge la fine del main;
  - si effettua una chiamata ad una funzione particolare detta exit();
  - il programma è interrotto in qualche modo;
  - il programma va in una condizione di errore detta anche crash.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Intestazione di una funzione

- L'intestazione di una funzione è costituita dai seguenti elementi:

  - il tipo della funzione corrispondente al valore da essa restituito alla sua terminazione; se la funzione non restituisce valori al posto della dichiarazione di tipo deve comparire **void**;

  - il nome mnemonico ad essa assegnato;

  - la lista di parametri di ingresso/uscita contenente la specifica (nome e tipo) dei parametri formali mediante i quali la funzione è capace scambiare informazioni con gli altri sottoprogrammi.

Sono esempi intestazioni di funzioni:

int main()
void somma (int a, int b, int *c)
int somma (int a, int b)

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Blocco di una funzione

- Il ***blocco*** è una entità sintattica che contiene la parte elaborativa di una qualsiasi unità di programma (programma principale, procedure e funzioni).
- Esso consiste di una serie di frasi del linguaggio costituenti la specifica dell'algoritmo racchiuse tra parentesi graffe.

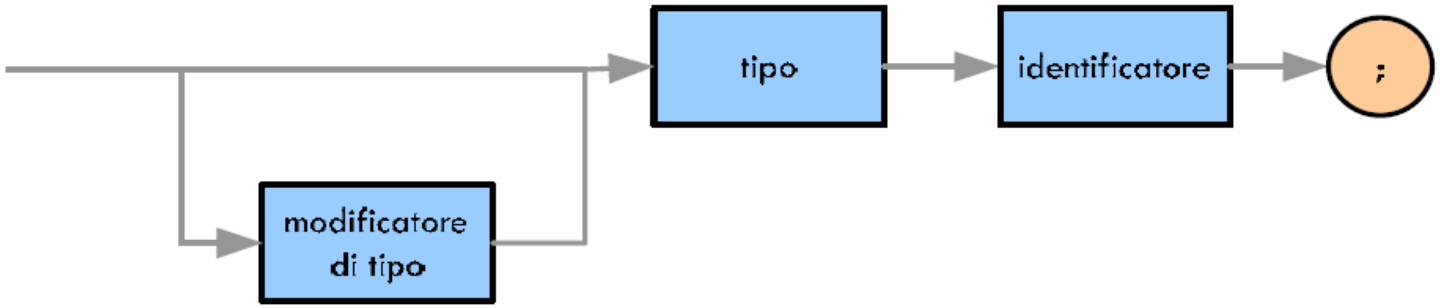
UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Definizione dei dati

- In C, tutti i dati devono essere *dichiarati e definiti* prima di essere usati!

- Definizione di un dato:
  1. riserva spazio in memoria;
  2. assegna un nome.
  3. identifica gli operatori leciti su quel dato

- Richiede l'indicazione di:
  - *nome* (identificatore);
  - *tipo*;
  - *modalità di accesso* (variabile/costante).



UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Tipi

- Il tipo definisce l'insieme dei valori che possono essere assunti, la rappresentazione interna e l'insieme degli operatori che possono agire su quel dato.

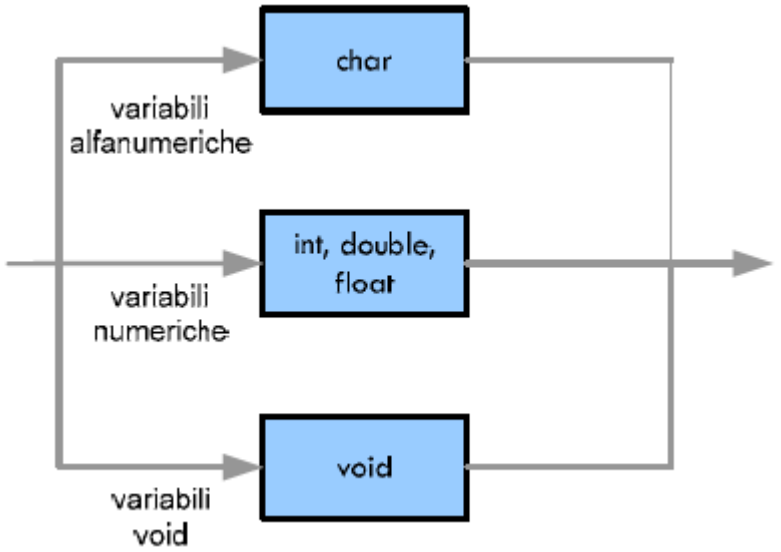
- Il linguaggio C richiede di definire il tipo dei dati e possiede regole rigide per la loro manipolazione (*tipizzazione forte*). Permette inoltre al programmatore di definire nuovi tipi astratti.

- Contemporaneamente permette di “vedere” gli oggetti interni al calcolatore: i registri, la memoria, gli indirizzi (puntatori), ecc.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Tipi base (primitivi)

- Sono quelli forniti direttamente dal C.



- Sono identificati da parole chiave!
  - char caratteri ASCII;
  - int interi (complemento a 2);
  - float reali (floating point singola precisione);
  - double reali (floating point doppia precisione).
- La dimensione precisa di questi tipi dipende dall'architettura (non definita dal linguaggio).
  - char = 8 bit sempre
- Attenzione: <u>le parole chiave dei tipi base vanno scritte in minuscolo!</u>

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

char

- Il tipo char *(character)* definisce un carattere (attenzione: un solo carattere!) espresso su 8 bit (1 byte) in codice ASCII.

- Si distinguono due tipi di char: il **signed char**, con l'ottavo bit usato come indicatore di segno, e l'**unsigned char**, che utilizza invece tutti gli 8 bit per esprimere il valore, e può dunque esclusivamente assumere valori positivi.
- Un carattere deve essere indicato tra apici, così:

'a'

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

int

- Il tipo int *(integer)* definisce i numeri interi può essere **signed** o **unsigned**.
- La rappresentazione interna e l'intervallo dei valori assunti dipende dal compilatore e dalla macchina usata.
- Generalmente si tratta del complemento a 2 e i valori assunti sono compresi nell'intervallo -32768 ÷ 32767 su 16 bit oppure, per le macchine a 32 bit, nell'intervallo -2.147.483.648 ÷ 2.147.483.647.

- Vanno indicati semplicemente così come siamo abituati a scriverli sulla carta, col loro valore (senza il punto decimale):

-2453

int

- Per evitare la differenza di rappresentazione tra macchina e macchina, lo standard definisce anche un tipo particolare detto **short int** che occupa 16 bit: spesso short int e int sono equivalenti.
- Per esprimere valori interi che variano in un range maggiore, si può usare il **long int**, che occupa 32 bit. Anche il long int può essere **signed** o **unsigned**.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

float e double

- Sia il tipo *float* che il tipo *double* sono rappresentazioni di numeri reali (frazionari).

- Sono rappresentati secondo la notazione *floating-point*, rispettivamente in singola (32 bit) e doppia (64 bit) precisione.

- I valori assunti (rappresentabili) sono:

float	± 3.4E+38	(7 cifre decimali)
double	± 1.7E+308	(16 cifre decimali)

- La rappresentazione è normalmente riferita allo standard IEEE 754.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

float e double

- I valori di tipo *float* o *double* vanno indicati con il punto decimale, ad esempio:

  14.8743

- E' ammessa anche una notazione simile alla notazione scientifica con il carattere E al posto di 10, così:

  0.148743E-02

- In alternativa, si può ancora scrivere il numero senza punto decimale ma seguito dal suffisso F oppure f (ad esempio, 10F, 10f e 10.0 sono equivalenti). Il compilatore concepisce questi valori sempre come di tipo *double*.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Valori Ammessi

TIPO	BIT	VALORI AMMESSI
character	8	da -128 a 127
unsigned character	8	da 0 a 255
short integer	16	da -32768 a 32767
unsigned short integer	16	da 0 a 65535
integer	16	da -32768 a 32767
unsigned integer	16	da 0 a 65535
long integer	32	da -2147483648 a 2147483647
unsigned long integer	32	da 0 a 4294967295
floating point	32	da $3.4*10^{-38}$ a $3.4*10^{38}$
double precision	64	da $1.7*10^{-308}$ a $1.7*10^{308}$
long double precision	80	da $3.4*10^{-4932}$ a $1.1*10^{4932}$

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Definizione di variabili

- Esempi:

```
int x;
char ch;
long int x1 ,x2, x3;
double pi;
short int stipendio;
long y,z;
```

- Usiamo nomi significativi!

  Esempi:

```
int RitenuteOperate, StipendioBase;
float OreLavorate;
```

- Esempi errati:

```
float Ore Lavorate;          /* c'è uno spazio   */
int Stip?base;               /* c'è un carattere speciale   */
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Definizione di variabili

- E' possibile "inizializzare" una variabile, ovvero attribuirgli un valore prima che venga utilizzata per la prima volta, in fase di dichiarazione della stessa.

- Esempio:

```
int x = 24;
char ch = 'm';
double pi = 124.654;
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Definizione di costanti

- Sintassi:

```
const  <tipo>  <nome della costante> = <valore>;
```

- Esempi:

```
const double pigreco =  3.14159;
const char  separatore = '$';
const float aliquota = 0.2;
```

- Convenzione:
  - Identificatori delle constanti tipicamente in MAIUSCOLO (ma è una convenzione!).

```
const double PIGRECO = 3.14159;
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Istruzioni di I/O

- Per ora consideriamo solo l'I/O interattivo, quello cioè che si realizza con tastiera e monitor.

- Sono disponibili diverse forme in base al tipo di informazione letta o scritta:

  - **I/O formattato**
  - **I/O di caratteri**
  - **I/O "per righe"**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I/O formattato

- Standard output (scrittura su monitor)
  istruzione printf

- Standard input (lettura da tastiera)
  istruzione scanf

- L'utilizzo di queste funzioni richiede l'inserimento di una *direttiva*

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

all'inizio del file sorgente il cui significato è: "includi il file *stdio.h* "

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

L'istruzione printf

- Visualizza sul monitor.

- La printf opera utilizzando una stringa, detta *<format>,* nella quale si devono inserire i comandi che descrivono come devono apparire i dati sul monitor.

- Al *<format>* deve seguire la lista di variabili che si vuol visualizzare.

- Sintassi:

printf (*<format>*,*<arg1>*,...,*<argn>*);

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

L'istruzione printf

- *<format>*: stringa che determina il formato di visualizzazione per ognuno dei vari argomenti.

- *<arg1>,...,<argn>*: lista degli argomenti da visualizzare. Gli argomenti (opzionali) possono essere costanti, variabili o espressioni.

- Se non ci sono argomenti (come quando si vuole visualizzare solo un messaggio) la funzione trasferisce sul video il testo della stringa di *<format>* e il cursore si posiziona subito dopo l'ultimo carattere.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

L'istruzione printf

- Affinché il cursore vada a capo, occorre inserire nella stringa di *format* il carattere new-line (\n).

- Esempio:

```
#include <stdio.h>

main()
{
printf ("Stampa di una riga\n");
printf ("Seconda riga\n");
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Specificatori di formato

- Generalmente nel *format* sono indicati gli specificatori di formato per le variabili della lista. I principali specificatori di formato sono:

  - %d o %i per il tipo int, stampa in notazione decimale;
  - %o per il tipo int, stampa in ottale senza segno;
  - %x per il tipo int, stampa in esadecimale senza segno;
  - %u per il tipo int, stampa in decimale senza segno;
  - %c per il tipo char, stampa un carattere;
  - %f per il tipo float, stampa nella notazione virgola mobile nel formato -d.dddddd (6 cifre dopo la virgola);
  - %e o %E per il tipo float, stampa nella notazione virgola mobile nel formato esponenziale -d.ddddde(E)±dd;
  - %s per le sequenze di caratteri (stringhe).

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Specificatori di formato

- Gli specificatori di formato sono costituiti dal carattere **%** seguito da un altro carattere che indica il formato da utilizzare per la stampa dell'argomento corrispondente (carattere, numero intero o reale, stringa, ecc.).

- Quando incontra il primo specificatore di formato il C preleva il primo argomento, effettua la conversione dal formato interno del dato ad una sequenza di caratteri ASCII seguendo le indicazioni del descrittore ed esegue infine l'ouput dei caratteri sul video.

- Prosegue poi con la stringa del format, ripetendo le azioni prima descritte per ogni specificatore incontrato e così fino ad esaurire l'intero format: il numero di specificatori di formato deve essere quindi pari al numero di argomenti.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Esempio

- L'associazione tra variabili e specificatori di formato è di tipo ordinale: 1° specificatore → 1ª variabile;
  2° specificatore → 2ª variabile, ecc.

- Esempi:

```
int x = 2;
float z = 0.5;
Char c = 'a';

printf ("%d %f %c\n", x, z, c);
```

```
2 0.500000 a
_
```

```
printf ("%f***%c***%d\n", z, c, x);
```

```
0.500000***a***2
_
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Specificatori di formato

- Tra il carattere % e quello di specificazione può esserci uno o più elementi aggiuntivi:
  - un intero, che fissa la larghezza minima (numero di caratteri) del campo su cui il dato è stampato;
  - un punto seguito da un intero, che stabilisce la precisione con cui visualizzare il dato (numero di cifre frazionarie);
  - uno di questi modificatori:
    - h (per indicare che si tratta di un tipo short),
    - l (per indicare che si tratta di un tipo long),
    - L (per indicare che si tratta di un tipo long double).

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Esempio (printf)

```
#include <stdio.h>
int  a=-57,  b=2,  c=450,  d=33;
float  e=1.22E7,  f=-0.1234567,  g=98765.4321,  h=1.0;

main()
{
printf ("a=%4d  b=%3d c=%8d d=%1d\n", a, b, c, d);
printf (“e=%9.3f f=%9.3f g=%9.3f h=%9.3f", e, f, g, h);
}
```

- Sul video apparirà:

  a= -57 b= 2 c=   450 d=33

  e=12200000.000 f=   -0.123 g=98765.432 h=   1.00

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Esercizio: Area

Scrivere un programma che calcoli l'area di un cerchio di raggio 3

Scrivere un programma che calcoli l'area di un triangolo di base 3 e altezza 7

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Esercizio: Area

```
#include <stdio.h>
#define pi 3.14
int main()
{
 //parte dicharativa
 int raggio;
 int area;
 //parte esecutiva
 raggio = 3;
 area = raggio*raggio*pi;
 printf("area= %d\n",area);
}
```

```
int main()
{
 //parte dicharativa
 int base;
 int altezza;
 int area;

 //parte esecutiva
 base = 3;
 altezza = 7;
 area = base * altezza;
 area = area/2;

 printf("area= %d\n",area);

}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Visibilità delle variabili

- Ogni variabile è definita all'interno di un preciso *ambiente di visibilità (scope).*

- Variabili *globali*
  - definite all'esterno al main sono visibili da tutti i moduli.

- Variabili *locali*
  - definite all'interno del main (sono visibili solo all'interno del main);
  - più in generale, definite all'interno di un blocco (sono visibili solo all'interno del blocco).

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Struttura a blocchi

- In C è possibile aggregare gruppi di istruzioni in **blocchi** racchiudendole tra parentesi graffe;

- significato: *delimitazione di un ambiente di visibilità di "oggetti" (variabili).*

- Esempio:

```
{
    int a=2;
    int b;

    b=2*a;
}
```

a e b sono definite

solo all'interno del blocco!

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Visibilità delle variabili - Esempio

```
int n;
double x;
main()
{
    int a,b,c;
    double y;
    {
        int d;
        double z;
    }
}
```

- n, x: visibili in tutto il file
- a, b, c ,y: visibili in tutto il main
- d, z: visibili solo nel blocco

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Assegnazioni

- Sintassi:

<*variabile*> = <*espressione*>;

- Non è un'uguaglianza!
  - Significato: il risultato di <***espressione***> viene assegnato a <***variabile***>;
  - <*variabile*> e <*espressione*> devono essere "compatibili" (ovvero dello stesso tipo);
  - <*variabile*> deve essere stata precedentemente definita!

- Esempio:

```
int x;
float y;
x = 3;
y = -323.9498;
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Assegnazioni

- In realtà l'*assegnazione* non è un'istruzione (come accade in tutti gli altri linguaggi);
- il simbolo **=** è un *operatore* che assegna alla variabile che si trova a sinistra il valore calcolato sull'espressione di destra;
- nel caso più semplice l'espressione di destra è un semplice valore.
- Sono pertanto lecite assegnazioni "multiple":

$$<var_1> = <var_2> = <espressione>;$$

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Alias di tipi

- Il C non permette di definire un nuovo tipo all'interno di un programma: tuttavia permette di introdurre un nome (***pseudonimo*** o ***alias*****)** che corrisponde ad uno dei tipi definiti.
- **typedef nomeTipo nomeNuovoTipo;**
- Il meccanismo degli alias può essere molto utile soprattutto per aumentare la leggibilità di un programma e per evitare espressioni complesse.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Il tipo enumerativo

- Tra i tipi fondamentali del linguaggio C va infine annoverato il cosiddetto ***tipo enumerativo***.
- L'enumerazione è un insieme di costanti intere rappresentate da identificatori.
- Le costanti sono dette anche costanti di enumerazione e sono delle costanti simboliche i cui valori sono impostati automaticamente: iniziano da 0 e sono incrementati di solito di 1.
- Un esempio di enumerazione è il seguente:
  - **enum GIORNI {LUN, MAR, MERC, GIOV, VEN, SAB, DOM }**
- creando un nuovo tipo **GIORNI** i cui identificatori sono associati con gli interi compresi tra 0 e 6, ovvero:
  - **enum GIORNI {LUN=0, MAR=1, MERC=2, GIOV=3, VEN=4, SAB=5, DOM=6 }**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

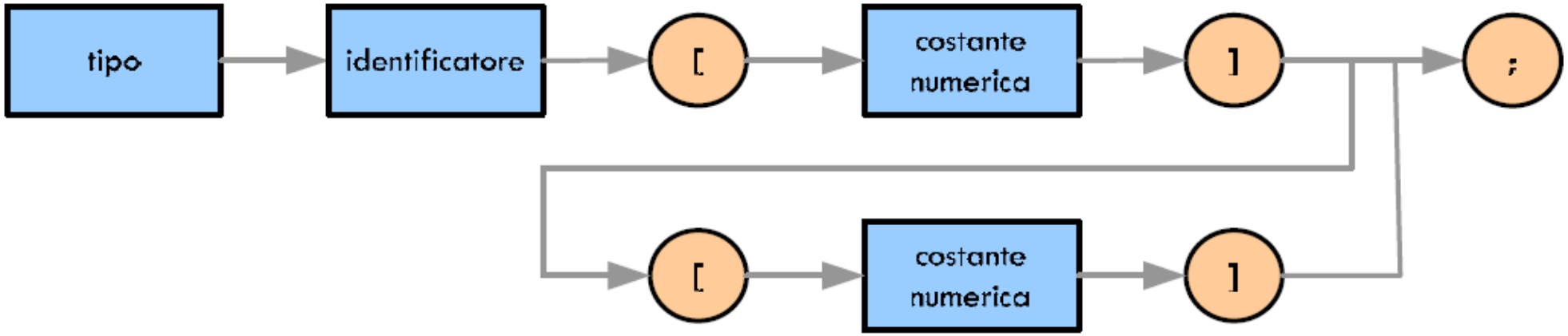
I tipi derivati

- Come noto, in un dato linguaggio, a patire dai tipi fondamentali è possibile costruire nuovi tipi, detti ***tipi derivati***.
- In C i tipi derivati sono vari: tra essi si prenderanno in considerazione gli *array*, le *strutture*, le *unioni* ed i *campi*.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I tipi derivati: array

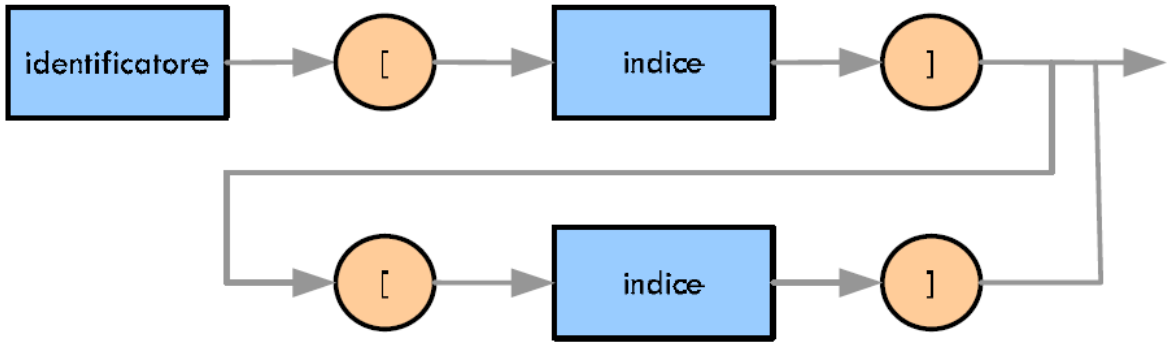
- La struttura ***array*** (o vettore) è composta da un insieme di elementi tutti dello stesso tipo e con un unico nome collettivo. Può avere una o più dimensioni fino ad un massimo prefissato.
- Ad ogni elemento dell'array è possibile accedere mediante un indice (o un numero di indici uguale alle dimensioni stabilite) che ne individua la posizione all'interno della struttura. Il numero di elementi dell'array è definito all'atto della dichiarazione e resta inalterato durante il corso del programma.



UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I tipi derivati: array

- La definizione di un array monodimensionale vet con cinque elementi di tipo intero è il seguente: **int vet[5];**
- Prima si dichiara il tipo (int), poi il nome dell'array (vet) e successivamente il numero massimo degli elementi dell'array. Per accedere ad un singolo elemento di un array, il linguaggio C mette a disposizione una *funzione d'accesso* consistente nel nome dell'array seguito dalla posizione dell'elemento (o coppia di posizioni) racchiusa tra parentesi quadre.
- La posizione è anche detta indice di accesso. Nel caso dell'array vet composto da 5 elementi l'indice può assumere i valori 0,1,2,3,4.
- Ad esempio le istruzioni:
  - **vet[1]=70;**
  - **vet[3]=1;**
- assegnano al secondo elemento dell'array vet il valore 70 e al quarto elemento dello stesso array il valore 1.



UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

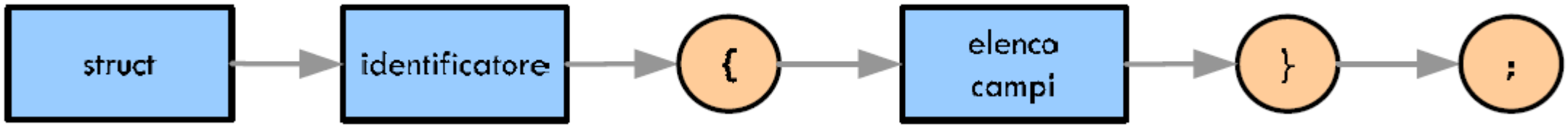
I tipi derivati: array

- Un esempio di definizione di array con più dimensioni (matrice) è il seguente:
  - **int mat[10][15];**
  - **float matReal[15][15];**
- L'accesso agli elementi di un array a due dimensioni è simile a quello di un array ad una sola dimensione, bisogna però in tal caso specificare due indici: l'*indice di riga* e quello *di colonna*.
- Ad esempio l'istruzione:
  - **mat[1][3]=3;**
- assegna il valore 3 all'elemento della matrice che si torva in corrispondenza della prima riga e della terza colonna.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I tipi derivati: struct

- Il linguaggio C mette a disposizione il tipo ***struct*** per la definizione dei *tipi record*.
- La struttura record è composta da un numero prefissato di componenti, anche di tipo differente. Ogni componente, detto anche *campo* del record, ha un suo nome e tipo.
- La definizione di un record prevede, allora, l'elencazione delle variabili, sia semplici che a loro volta strutturate, costituenti i campi componenti.
- Le operazioni consentite sul record sono quelle che si possono fare sui campi.



UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I tipi derivati: unioni e campi

- Il concetto di ***unione*** deriva direttamente da quello di struttura, ma con una importante differenza: i campi di una **union**, a differenza di quelli di una struct, hanno lo stesso indirizzo di memoria e vanno ad occupare, quindi, tutte le medesime locazioni di memoria.

- I campi sono una particolare struttura C che consente di far riferimento simbolicamente ai singoli bit di una variabile.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Stringhe di caratteri

- Il C non prevede la presenza di un tipo ***stringa*** predefinito. Tuttavia, è possibile utilizzare un array di caratteri con alcune convenzioni. L'array:
  - **char stringa[10];**
- dichiara un array di 10 caratteri, mentre la dichiarazione:
  - **char sentenza[ ]= "Salve mondo";**
- dichiara un array di caratteri in cui numero di elementi è dato dalla quantità di caratteri presenti nella stringa, più uno, un carattere speciale che indica la fine della stringa, detto *carattere null* ("\0"):

S	a	l	v	e		M	o	n	D	o	\0

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

I puntatori

- Un ***puntatore*** è una variabile che contiene un indirizzo di memoria: tale indirizzo rappresenta la posizione in memoria di un'altra variabile. Quando una variabile contiene l'indirizzo di un'altra, la prima è detta *puntare* alla seconda.
- Poiché una variabile possa essere usata come puntatore, bisogna che sia prima dichiarata come tale, attraverso la dichiarazione:
  - **tipo *var;**
- dove **tipo** è un tipo base del linguaggio e **var** è la variabile puntatore.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gli operatori del linguaggio

- ***Operatori Aritmetici***
- Gli ***operatori aritmetici*** sono i classici operatori binari

  +,- ,*,/ e l'operatore di modulo %.
- La divisione tra interi tronca la parte frazionaria, mentre l'espressione **x%y** fornisce il resto della divisione di x per y. L'operatore di modulo può essere applicato solo a tipi integrali.
- Gli operatori + e – hanno la stessa priorità, priorità inferiore a * e / (che invece hanno identica priorità).

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gli operatori del linguaggio

- ***Operatori Relazionali***
- Gli ***operatori relazionali*** sono:

  > >= < <=
- mentre gli operatori di eguaglianza (e disuguaglianza) sono:

  == !=
- Gli operatori di relazione hanno priorità maggiore rispetto a quelli di eguaglianza /disuguaglianza.
- Gli operatori relazionali hanno invece priorità minore rispetto agli operatori aritmetici.

  Aritmetici→relazionali→Uguaglianza/Disuguaglianza

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gli operatori del linguaggio

- ***Operatori Logici***
- Gli ***operatori logici*** sono:
  && (and), || (or) e ! (not)
- **&&** ha priorità maggiore di **||** ed entrambi hanno priorità inferiore agli operatori relazionali e di uguaglianza.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gli operatori del linguaggio

- ***Operatori di incremento e decremento***
- Il C fornisce due operatori inusuali nei linguaggi ad alto livello, utili per incrementare e per decrementare le variabili, ovvero l'operatore **++** e l'operatore -- **,** con notazione "prefissa" o "postfissa", come in esempio:

  x++

  ++x

  x- -

  - -x

- Sia nella notazione prefissa che postfissa si ha come effetto quello di incrementare (o decrementare) una variabile. Tuttavia **++x (- -x)** incrementa (decrementa) **x** prima di usarne il valore, mentre **x++ (x- -)** incrementa (decrementa) **x** dopo averne usato il valore.

Gli operatori sui puntatori

- Esistono due operatori per la manipolazione dei puntatori: **&** e *. Il primo è un operatore unario che restituisce l'indirizzo di memoria dell'operando cui è applicato. Ad esempio:

  indirizzo_x=&x;

- pone nella variabile **indirizzo_x** l'indirizzo di memoria della variabile **x**.
- Il secondo è un operatore unario che restituisce il valore della variabile che si trova all'indirizzo indicato nella variabile che lo segue. Ad esempio:

  valore_x=*indirizzo_x;

- ha l'effetto di porre nella variabile **valore_x** il valore della variabile che si trova in memoria all'indirizzo contenuto nella variabile **indirizzo_x**, ovvero il valore di **x**.

Gli operatori sui puntatori

- Esempio:

int x;
**int* indx=&x; ** restituisce l'indirizzo di memoria dell'operando cui è applicato
int valx=*indx;
printf("Benvenuti al corso di Fondamenti di Informatica\n");
printf("Esempio di programma sui puntatori\n");
printf("Inserisci il numero x: ");
scanf("%d",&x);
printf("\nIl valore di x e': %d\n",x);
printf("\nL'indirizzo di memoria (in esadecimale) a cui si trova x e': %x\n",indx);
printf("\nIl valore di x mediante puntatore e': %d\n",x);

Gli operatori sui puntatori

- Esempio: si vuole realizzare una funzione che effettua la somma e la differenza tra due numeri

```
void sommadiff(int a, int b, int *d, int *s) {
*d=a-b;
*s=a+b;
}
```

- mentre l'attivazione della funzione avverrà come segue:

```
int a,b,d,s;
sommadiff (a,b,&d,&s);
```

- L'attivazione della funzione ha come effetto quello di passare una copia dei valori delle variabili **a** ed **b** alla funzione (passaggio per valore) e quello di passare per le variabili **d** e **s** una copia dell'indirizzo di memoria (passaggio per riferimento) in cui esse si trovano per potere accedere al loro contenuto mediante l'operatore * all'interno della funzione.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gli operatori sui puntatori

- N.B. il tipo array viene sempre passato per riferimento, e quindi all'atto dell'attivazione di una funzione non viene utilizzato l'operatore & per la variabile ad esso relativo.
- Inoltre si ha che la sequenza:

  char str[80], *p1;
  p1=str;
- ha l'effetto di copiare nel puntatore **p1** l'indirizzo del primo elemento dell'array **str**.
- Ciò comporta che gli elementi di un array possono essere acceduti (in maniera sequenziale) anche mediante l'utilizzo dei puntatori. Ad esempio:

  str[4] e *(p1+4)
- sono equivalenti in quanto ogni elemento del vettore occupa un solo byte di memoria.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Operatori su bit

- Operatori di manipolazione dei bit:

Operazione	Operatore	tipo
AND bit a bit	&	Binario
OR bit a bit	|	Binario
XOR bit a bit	^	Binario
NOT bit a bit (complemento a 1)	~	Unario
Shift a sinistra	<<	Binario
Shift a destra	>>	Binario

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Statement

- La specificazione delle azioni elaborative di un sottoprogramma in C si traduce in un insieme di *enunciati* o *statement composti*. Lo statement composto è formato da una sequenza di *statement semplici* terminati dal carattere punto e virgola, mentre gli statement semplici che compongono la specificazione dell'algoritmo denotano le azioni elaborative vere e proprie che devono essere svolte dall'esecutore macchina.

  Tali statement si possono classificare in:

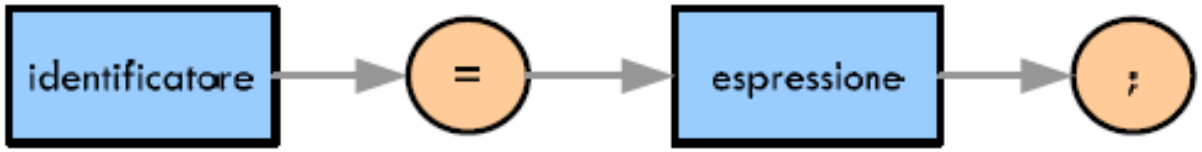
  - *istruzioni di assegnazione*
  - *richiamo di funzioni*
  - *costrutti selettivi (if … else, switch … case)*
  - *costrutti iterativi (while, for)*

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Istruzione di assegnazione

- L'*enunciato* o ***istruzione di assegnazione*** è la più elementare forma di enunciato.
- Esso prescrive che venga dapprima calcolato il valore dell'espressione che si trova a destra del segno di '=', e poi che tale valore venga poi assegnato alla variabile che si trova a sinistra dello stesso simbolo.

```
x = 1;
x = x+1;
x = a+b;
```



Espressione1 = Espressione1 operando Espressione2

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Richiamo della funzione

- Il ***richiamo della funzione*** determina l'esecuzione della funzione indicata. Affinché il richiamo sia corretto, si deve fornire una lista di parametri di I/O uguale, in numero e tipo, a quella specificata nella dichiarazione della funzione.

- La forma di richiamo per una funzione è la seguente:
- **variabile=nome_funzione(par_effettivo1,par_effettivo2,…., par_effettivon);**
- mentre per una procedura:
- **nome_procedura(par_effettivo1,par_effettivo2,…., par_effettivon);**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Costrutti selettivi

- L'enunciato **if-else** (se - altrimenti) permette di effettuare la scelta tra due alternative in funzione del valore di una espressione booleana.
- Se il valore dell'espressione è TRUE, si sceglie l'alternativa (insiemi di enunciati racchiusi tra parentesi graffe) introdotta dopo l'**if**, in caso contrario (insiemi di enunciati racchiusi tra parentesi graffe) quella aperta dall'**else**.
- In entrambi i casi, dopo l'esecuzione del blocco selezionato, l'esecuzione stessa continua con l'enunciato successivo all'**if**.
- È anche possibile usare una notazione abbreviata nel caso in cui non esista una delle due alternative, non specificando l'**else** dell'enunciato

```
if (n>10) {
        n=n+1;
}
else {
        n=n-1;
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Costrutti selettivi

- L'istruzione **switch** permette di scegliere tra più di due alternative (decisioni multiple) verificando se il valore di una espressione è uguale ad un valore tra quelli specificati in una lista.

Si noti che l'istruzione **break** causa l'uscita dallo **switch**. Il caso chiamato **default** viene eseguito quando non sono stati soddisfatti gli altri casi dello **switch**.

```
SWITCH(numero_mese) {
      CASE {1,3,5,7,8,10,12}
            printf("mese di 31 giorni");
            break;
      CASE {4,6,9,11}
            printf("mese di 30 giorni");
            break;
      CASE 2
            printf("mese di 28 o 29 giorni");
            break;
      DEFAULT:
            printf("mese non valido");
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Costrutti Iterativi

- L'enunciato **while** è composto da una espressione logica e da uno o più enunciati da ripetere in funzione del valore di tale espressione. L'esecuzione del **while** comporta la valutazione dell'espressione e l'esecuzione degli enunciati nel caso in cui il valore calcolato dell'espressione sia TRUE. Il ciclo ha termine quando l'espressione assume il valore FALSE per cui, se l'espressione risulta subito falsa, gli enunciati non vengono mai eseguiti.

```
while(x > 0)
{
        a = a+x;
        x- -;
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Costrutti Iterativi

- A differenza dei ciclo **while**, che verifica la condizione all'inizio del ciclo (loop) stesso, il costrutto **do-while** la verifica alla fine, con la conseguenza che esso viene **eseguito almeno una volta.**
- In particolare, l'istruzione viene eseguita, poi viene valutata l'espressione: se è vera, l'istruzione viene ancora eseguita e così via. Il ciclo termina quando l'istruzione diventa falsa.

```
do {
    scanf("%d",&n);
} while (n>0)
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Costrutti Iterativi

- Il costrutto **for** è un enunciato iterativo enumerativo o ciclico. Esso deve essere usato ogni qualvolta il numero di ripetizioni è noto a priori.
- Lo statement di *inzializzazione* coincide, nella sua forma più semplice, con un'istruzione di assegnazione con la quale viene fornito il valore iniziale alla variabile di controllo del ciclo. Lo statement di *condizione* rappresenta un'espressione booleana che serve a controllare la terminazione del ciclo, finché è TRUE il ciclo prosegue. Lo statement di *aggiornamento* definisce il modo in cui la variabile di controllo cambia il suo valore ad ogni ripetizione del ciclo. Di norma l'aggiornamento consiste in un *incremento* o *decremento* del valore della variabile di controllo.

```
for(int i=0; i<=100; i++)
        printf("%d ",i);
```

```
for(int i=100; i>=0; i- -)
{
        printf("%d ",i);
        somma = somma+i;
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Librerie di Funzioni

- Il C, per scelta di progetto, non supporta direttamente istruzioni di ingresso/uscita, né istruzioni particolari per le operazioni matematiche;
- non esistono nemmeno operazioni per trattare direttamente oggetti strutturati come stringhe di caratteri, insiemi, liste ed array. Il C affida tali tipologie di operazioni a ***librerie esterne*** di funzioni.
- È però possibile servirsi delle funzioni di una particolare libreria includendone il corrispettivo *file header* all'interno del programma, come già visto, mediante la direttiva di compilazione:
- **#include <file header della libreria>**

Un moderno compilatore C mette a disposizione una vasta gamma di librerie contenenti:

- **funzioni di uso generale ("stdlib.h"),**
- **gestione dell'I/O ("stdio.h"),**
- **calcolo matematico ("math.h"),**
- **gestione di stringhe ("string.h").**

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

La gestione dell'I/O

- Le funzioni comprese nel sistema di input/output del C possono essere raggruppate in tre grandi categorie: I/O su console (tastiera e monitor), I/O su file bufferizzato e I/O su file non bufferizzato (Unix-like).
- Per l'utilizzo di tutte le funzioni suddette è richiesta l'inclusione del file header **"stdio.h"**. Tale libreria fornisce alcune funzioni per le operazioni di lettura e scrittura dei valori delle variabili dai file standard INPUT e OUTPUT.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Apertura File

- In C la lettura da un file avviene aprendo un canale di comunicazione con la memoria di massa attraverso la connessione del file. In altre parole un file deve essere sempre connesso o aperto prima di prelevare dati da esso. L'apertura di un file avviene attraverso la funzione **fopen()** della libreria **"stdio.h"**:

  FILE* fopen (char* filename, char* permission)

- Tale funzione apre un file il cui nome (path su disco del file) è un insieme di caratteri "puntato" dalla variabile **filename**, mentre la variabile **permission** definisce la modalità di apertura del file (e.g., sola lettura, solo scrittura, lettura/scrittura). La funzione suddetta restituisce poi un identificatore del file noto anche con nome di *puntatore a file*, attraverso il quale il file viene referenziato. Se per vari motivi il file non può essere aperto viene restituito un puntatore nullo.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Lettura File

- Apertura in sola lettura di un file *pippo.txt* contenente un array di numeri interi (i numeri si trovano su righi differenti di testo separati tra loro dal carattere di fine rigo).

```
FILE *fid;
fid=fopen (“c:\pippo,txt”, “r”);
if (!fid) printf(“FILE NON APERTO”);
```

Di seguito è riportato un esempio di apertura di un file in scrittura:

```
FILE *fid;
fid=fopen (“c:\pippo,txt”, “w”);
if (!fid) printf(“FILE NON APERTO”);
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Lettura File

- esempio di lettura del file dove è memorizzato un array di interi (sul primo rigo del file di testo c'è il riempimento del vettore):

```
FILE * fp;
register int i;
int vettore[100];
int riemp, ret;
if (!(fp=fopen("pippo.txt","r"))) {
printf("\n Il file non puo' essere caricato\n");
}
else {
ret=fscanf(fp,"%d",&riemp);
if (ret!=EOF) {
printf("\nNumero di elementi nel vettore:
%d\n",riemp);
if (riemp>0) {
for (i=0; i<riemp; i++) fscanf(fp,"%d",vettore[i]);
}
}
else printf("\nfile vuoto!!!\n");
}
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Scrittura su File

- scrittura su file di un array di interi (sul primo rigo del file di testo viene inserito il riempimento del vettore):

```
register int i;
FILE*fp;
int count;
int tot=0;
int riemp=3;
int vettore[]= {1,2,3};
if (!(fp=fopen("pippo.txt","w"))) {
printf("\n Il file non puo' essere salvato\n");
}
else {
count=fprintf(fp,"%d\n",riemp);
tot=tot+count;
for (i=0; i<riemp; i++) {
count=fprintf(fp,"%d\n",vettore[i]);
tot=tot+count;
}
}
if (tot==riemp+1) printf("Scrittura avvenuta con successo");
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gestione delle Stringhe

- Il C possiede un ampio insieme di funzioni per il trattamento delle stringhe edei caratteri. Si ricorda che una stringa coincide con un array di caratteri terminato col carattere '\0'. Tutte le dichiarazioni richieste dalle funzioni per il trattamento delle stringhe sono contenute nel file header **"string.h"**.
- La funzione **strcat()** permette il concatenamento di due stringhe:

char *strcat(char *str1, char *str2)

```
char s1[]={'c','i','a','o','.','.','.', '\0'};
char s2[]={'c','o','m','e',' ','s','t','a','i','?', '\0'};
strcat(s1,s2);
printf("%s",s1);
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gestione delle Stringhe

- La funzione **strcpy()** permette la copia di stringhe:

  char *strcpy (char *str1, char *str2)

- Tale funzione copia il contenuto della stringa **str2** nella stringa **str1**, restituendo un puntatore a **str1**.
- La funzione **strcmp()** permette di verificare l'uguaglianza tra stringhe:

  int strcmp(char *str1, char *str2)

Tale funzione confronta secondo le regole lessicografiche due stringhe **str1** e **str2** terminate da carattere nullo e restituisce un intero il cui valore è determinato sulla base delle seguenti regole:

- 0 le stringhe sono uguali;
- minore di 0 le stringhe sono diverse ed in più **str1** ha meno caratteri di **str2**;
- maggiore di 0 le stringhe sono diverse e **str1** ha più caratteri di **str2**.

```
char str[100];
strcpy(str,"ciao");
printf("%s",str);
```

```
char s1[100];
char s2[100];
scanf("%s",s1);
scanf("%s",s2);
if (strcmp(s1,s2)==0)
printf("stringhe uguali");
```

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Gestione delle Stringhe

- La funzione **strlen()** permette di determinare la lunghezze di una stringa:

int strlen(char *str1)

```
char s1[100];
scanf("%s",s1);
int len=strlen(s1);
printf("stringa lunga %d caratteri",len);
```

- Tale funzione conta il numero di caratteri da cui è composta una stringa **str1** che termina col carattere di terminazione, escludendo tale carattere dal conteggio.
- Infine, le due funzioni:
- **char*strlwr(char *str1)**
- **char*strupr(char *str1)**
- convertono rispettivamente una stringa **str1** in minuscolo e maiuscolo, restituendo entrambe un puntatore alla nuova stringa.

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Funzioni per il calcolo matematico

- Le funzioni si possono dividere nelle seguenti categorie:
- funzioni trigonometriche
  - o **double sin(double x)** per il calcolo del seno di un numero reale **x**
  - o **double cos(double x)** per il calcolo del coseno di un numero reale **x**
  - o **double tan(double x)** per il calcolo del tangente di un numero reale **x**
- funzioni iperboliche
  - o **double sinh(double x)** per il calcolo del seno iperbolico di un numero reale **x**
  - o **double cosh(double x)** per il calcolo del coseno iperbolico di un numero reale **x**
  - o **double tanh(double x)** per il calcolo del tangente iperbolica di un numero reale **x**
- funzioni esponenziali e logaritmiche
  - o **double exp(double x)** per il calcolo dell'esponenziale di un numero reale **x**
  - o **double log10(double x)** per il calcolo del logaritmo in base 10 di un numero reale **x**
  - o **double log2(double x)** per il calcolo del logaritmo naturale di un numero reale **x**
- altre funzioni
  - o **double sqrt (double x)** per il calcolo della radice quadrata di un numero reale **x**
  - o **double fabs(double x)** per il calcolo del valore assoluto di un numero reale **x**
  - o **double ceil (double x)** per il calcolo dell'intero più piccolo non inferiore ad un numero reale **x**
  - o **double floor(double x)** per il calcolo dell'intero più grande non superiore ad un numero reale **x**
  - o **double fmod(double x, double y)** per il calcolo del resto in modulo della divsione di **x** per **y**
  - o **double pow(double base, double exp)** per il calcolo di **base^exp**