- 2° Modulo – parte 1^
  - La rappresentazione dell'Informazione
  - Struttura dei programi C
  - Tipi Semplici del C
  - Costanti Variabili, Operatori
  - Espressioni
  - Operatori aritmetici, logici, bitwise

http://www.elet.polimi.it/upload/martucci/index.html

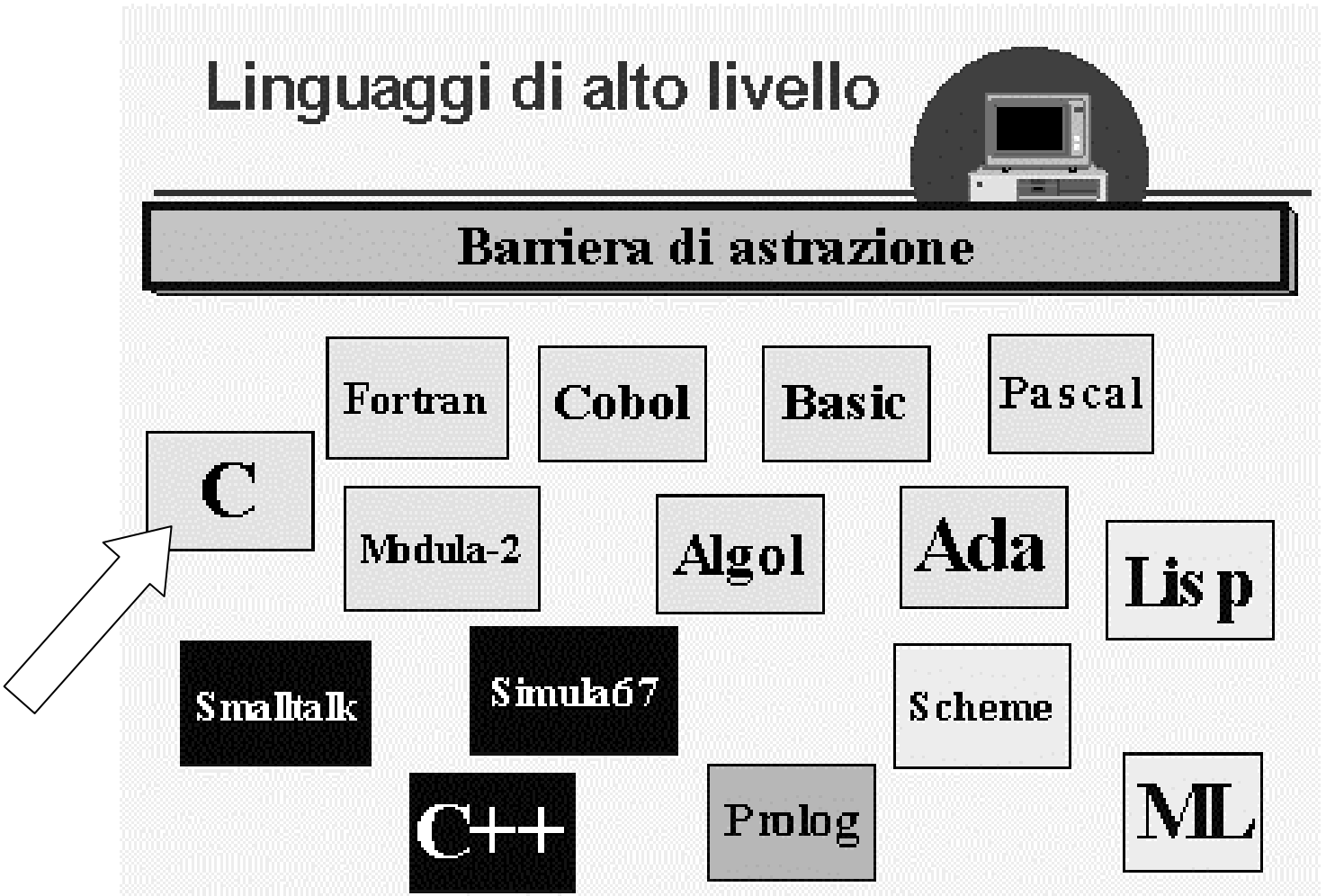
Passi fondamentali del C

- Definito nel 1972 (AT&T Bell Labs) per sostituire l’assembler nella programmazione di sistemi operativi: in pratica, nato per creare **UNIX**

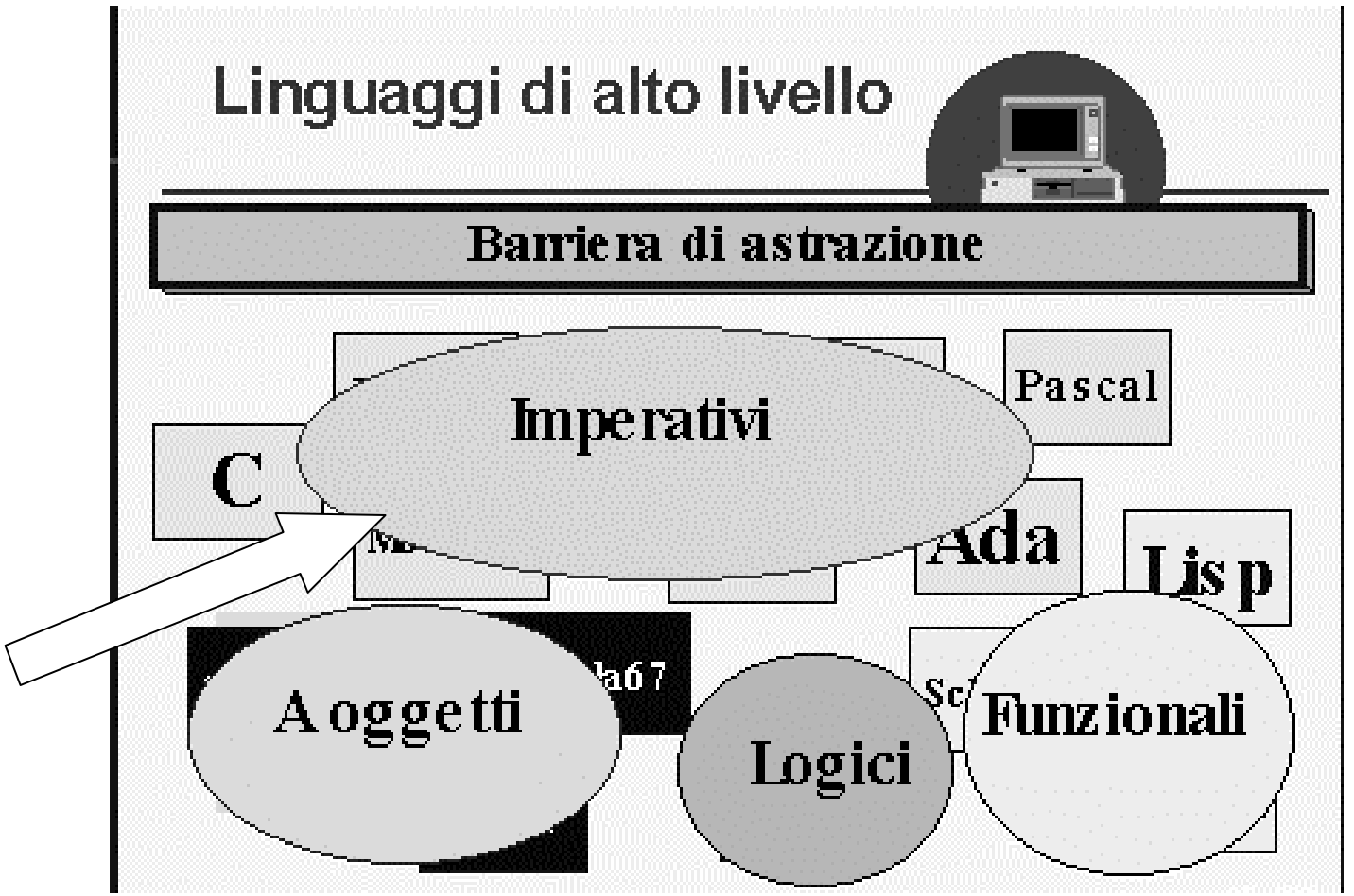
- Prima definizione precisa: Kernigham & Ritchie (1978)

- Prima definizione ufficiale: **ANSI C** (1983)

C: linguaggio di alto livello



C: linguaggio imperativo



Caratteristiche del C

- Linguaggio *sequenziale*, ***imperativo***, ***strutturato a blocchi***
- Usabile anche come linguaggio di sistema → adatto a software di base, sistemi operativi, compilatori, ecc.
- Portabile, efficiente, sintetico (ma a volte poco leggibile...)

- Basato su pochi concetti elementari:
  - **espressione**
  - **dichiarazione / definizione**
  - **istruzione / blocco**
  - **funzione**
- tuttavia, viene arricchito da un vasto insieme di librerie di funzioni (per operazioni matematiche, di input/output, su stringhe, ecc.)

Caratteristiche del C (*cont*.)

SET DI CARATTERI

Dipendente dalla implementazione (in genere ASCII più estensioni)

IDENTIFICATORI

<Identificatore> ::= <Lettera> { <Lettera> | <Cifra> }

- **Lettera** include tutte le lettere, maiuscole e minuscole, e l'underscore **"_"** (utilizzabile all'inizio solo per *oggetti di sistema*)

- Maiuscole e minuscole sono *diverse* (linguaggio C è *case-sensitive*)

PAROLE RISERVATE

- Esempio: **int, float, if, for, do, ...**

  { } delimitatore di blocco

COMMENTI

/* commento, anche su più righe */ (non possono essere innestati)

ELEMENTI DEL LINGUAGGIO C

parole chiave *(riservate)* ⇒ proprie del linguaggio
- sono le istruzioni oppure hanno un significato particolare (tipi)

identificatori ⇒ costituiti da sequenze di lettere ...
- rappresentano nomi di variabili, costanti, (tipi nuovi), funzioni, procedure
- definiti dall'utente oppure «di sistema» (di libreria)

operatori (unari o binari)
- di assegnamento =
- aritmetici + - * /
- relazionali (confronto) > < <= == ...
- logici ! (not) && ||
- di chiamata di funzione/procedura ( )
- di dereferenziazione &
- dipendenti dal costruttore di tipo * [ ] ....
- altri

separatori (delimitatori)
- di identificatori di variabili e costanti ,
- di istruzioni ;
- commento /* */
- di blocco di istruzioni { }
- in espressioni ( )

direttive al preprocessore C
- # include (parola chiave) .....

valori costanti (cifre o caratteri)

THE ‘C’ ALPHABET

The characters in ‘C’ are divided into 3 groups

- Digits: 0 - 9
- Letters: A - Z, a - z, $, -
- Special Characters:

b Blank	/ Slash	‘ Apostrophe
+ Plus	! Exclamation	* Asterisk
? Question Mark	^ Circumflex	(Let Parent
: Colon	| Stroke	) Right Par
< Less than	, Comma	[Left Bracket
= Equal	% Percent	] Right Bracket
> Greater than	$ Dollar sign	{ Left Brace
~ Tilde	. Period	} Right Brace
& Ampersand	_ Underscore	# Pound sign
- Hyphen	“ Quotation mark	; Semicolon
\ Backslash		

Reserved Words

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

Sintassi C: Frasi

Summary of Statements

statement :

compound-statement
expression-statement
selection-statement
iteration-statement
jump-statement

jump-statement :

continue;
break;
return *expression* $_{opt}$ **;**

compound-statement :
{ *declaration-list* $_{opt}$ *statement-list* $_{opt}$ **}**

declaration-list :
declaration
declaration-list declaration

statement-list :
statement
statement-list statement

expression-statement :
expression $_{opt}$ **;**

iteration-statement :
while (*expression* **)** *statement*
do *statement* **while (** *expression* **);**
for (*expression* $_{opt}$ **;** *expression* $_{opt}$ **;** *expression* $_{opt}$ **)** *statement*

selection-statement :
if (*expression* **)** *statement*
if (*expression* **)** *statement* **else** *statement*
switch (*expression* **)** *statement*

labeled-statement :

case *constant-expression* **:** *statement*
default : *statement*

Sintassi C: Frasi

Summary of Statements

statement :

compound-statement
expression-statement
selection-statement
iteration-statement
jump-statement

jump-statement :

continue;
break;
return *expression* $_{opt}$ **;**

compound-statement :
{ *declaration-list* $_{opt}$ *statement-list* $_{opt}$ **}**

declaration-list :
declaration
declaration-list declaration

statement-list :
statement
statement-list statement

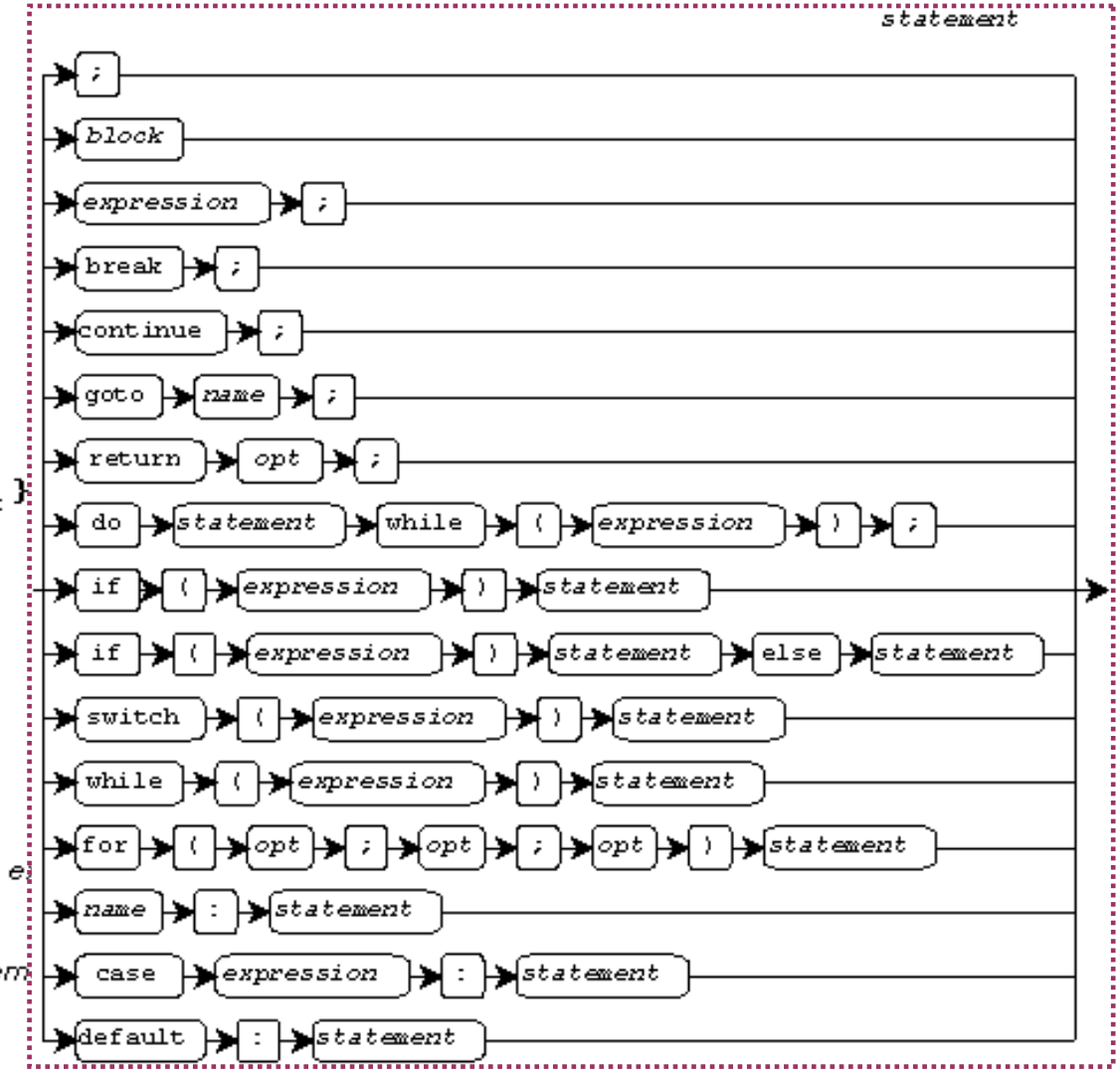
expression-statement :
expression $_{opt}$ **;**

iteration-statement :
while (*expression* **)** *statement*
do *statement* **while (** *expression* **);**
for (*expression* $_{opt}$ **;** *expression* $_{opt}$ **;** *e*

selection-statement :
if (*expression* **)** *statement*
if (*expression* **)** *statement* **else** *statem*
switch (*expression* **)** *statement*

labeled-statement :

case *constant-expression* **:** *statement*
default : *statement*



Struttura di un Programma C

Un programma C ha in linea di principio la seguente forma:

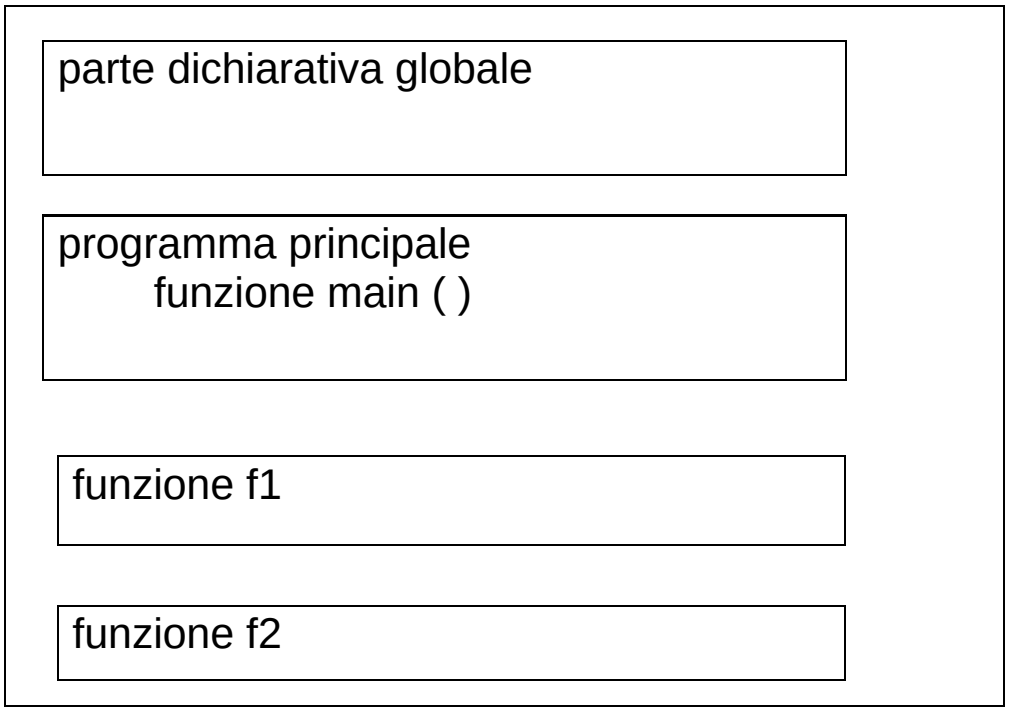
- **Direttive per il preprocessore**
- **Definizione di tipi**
- **Prototipi di funzioni**, con dichiarazione dei tipi delle funzioni e delle variabili passate alle funzioni)
- **Dichiarazione delle Variabili Globali**
- **Dichiarazione Funzioni**, dove ogni dichiarazione di una funzione ha la forma:

```
Tipo NomeFunzione(Parametri)
{
   Dichiarazione Variabili Locali
   Istruzioni C
}
```

```
#include <stdio.h>

typedef struct point {
            int   x;    int   y;
            };
int       f1(void);
void    f2(int i, double g);


int   sum;


int main(void)
{
      int j;
      double g=0.0;
      for(j=0;j<2;j++)
            f2(j,g);
      return(2);
}

void f2(int i, double g)
{
      sum = sum + g*i;
}
```

STRUTTURA DI UN PROGRAMMA C - 1

programma C



Stile di scrittura di un programma ⇒ **leggibilità**

- scelta degli identificatori
- indentazione: «struttura grafica» che rispecchia la struttura logica

- uso di commenti

STRUTTURA DI UN PROGRAMMA C - 2

Parte dichiarativa globale:

- servizi (funzioni) importate da altri moduli (file), cioè definite e codificate in altri file

- «oggetti» (tipi di dati, variabili, costanti simboliche, prototipi di funzioni) visibili (utilizzabili) da tutto il programma, cioè da main e dalle altre funzioni.

Programma principale:

```
main ()

{

  parte dichiarativa
  locale



  parte esecutiva

}
```

parola riservata (identificatore di funzione)
appare una e una sola volta nel programma
definisce l'inizio dell'esecuzione
è (formalmente) una funzione

definisce l'insieme di «oggetti» usati dal programma principale per l'esecuzione. sono oggetti visibili (locali) a main.

insieme di istruzioni che costituiscono il programma principale

STRUTTURA DI UN PROGRAMMA C - 3

Parte dichiarativa locale

1. dichiarazione di costanti
2. definizione di «nuovi» tipi definiti dall'utente (ridenominazione)
3. dichiarazione di variabili
4. prototipi di funzioni

«Regole» sintattiche sulle dichiarazioni sia locali che globali:

- ogni identificatore usato deve essere prima definito
- ogni variabile usata deve essere prima dichiarata

Parte esecutiva: istruzioni (per tipologia)

- istruzioni di assegnamento
- istruzioni composte
- costrutti di (modifica del flusso di) controllo (costrutti condizionali, costrutti ciclici)
- «istruzioni» di ingresso e uscita
- chiamate di sottoprogrammi (funzioni)

ESEMPIO 1 - Dichiarazioni e istruzioni

```
#include <stdio.h>


main ( )

{
 int potenza, base;
 int esponente, i;          /* devono essere >=0*/

 printf("Inserisci il valore della base:");
 scanf("%d",&base);
 printf("Inserisci il valore dell'esponente:");
 scanf("%d", &esponente);

 potenza=1;
 i=0;

 while (i<esponente)
   {
   potenza=potenza*base;
   i=i+1;
    }

 printf("Potenza=%d \n", potenza);
}
```

ESEMPIO 1 - Dichiarazioni e istruzioni

```
#include <stdio.h>


main ( )

{
 int potenza, base;
 int esponente, i;          /* devono essere >=0*/

 printf("Inserisci il valore della base:");
 scanf("%d",&base);
 printf("Inserisci il valore dell'esponente:");
 scanf("%d", &esponente);

 potenza=1;
 i=0;

 while (i<esponente)
    {
    potenza=potenza*base;
    i=i+1;
     }

 printf("Potenza=%d \n", potenza);
}
```