

Il linguaggio C

Variabili, costanti, tipi scalari

```
main()
{
/*definizioni variabili: */
char y='a'; /*codice(a)=97*/
int x,X,Y;
unsigned int Z;
float SUM;
double r;

/* parte istruzioni: */
X=27;
Y=343;
Z = X + Y -300;
X = Z / 10 + 23;
Y = (X + Z) / 10 * 10;      /* qui X=30, Y=100, Z=70 */
X = X + 70;
Y = Y % 10;
Z = Z + X -70;
SUM = Z * 10;              /* X=100, Y=0, Z=100 , SUM=1000.0*/
x=y;                      /* char -> int: x=97*/
x=y+x; /*x=194*/
r=y+1.33; /* char -> int -> double*/
x=r; /* coercizione -> troncamento: x=98*/
}
```

Fondamenti di Informatica L- A

Breve storia e caratteristiche

Progettato nel **1972** da D. M. Ritchie presso i laboratori AT&T Bell, per poter riscrivere in un linguaggio di alto livello il codice del sistema operativo UNIX.

Definizione formale nel **1978** (B.W. Kernigham e D. M. Ritchie)

Nel **1983** è stato definito uno standard (**ANSI C**) da parte dell'American National Standards Institute.

Alcune caratteristiche:

- Elevato potere espressivo:
 - **Tipi di dato** primitivi e tipi di dato definibili dall'utente
 - **Strutture di controllo** (programmazione strutturata, funzioni e procedure)
- Caratteristiche di basso livello
 - Gestione della memoria, accesso alla rappresentazione

Fondamenti di Informatica L- A

Esempio di programma in C

```
#include <stdio.h>
main() {
    // dichiarazione variabili
    int A, B;
    // input dei dati
    printf( "Immettere due numeri: " );
    scanf( "%d %d", &A, &B );
    // eseguo semisomma e mostro risult.
    printf( "Semisomma: %d\n", (A+B)/2 );
}
```

Fondamenti di Informatica L- A

Elementi del testo di un programma C

Nel testo di un programma C possono comparire:

- **parole chiave:** sono parole riservate che esprimono istruzioni, tipi di dato, e altri elementi predefiniti nel linguaggio
- **identificatori:** nomi che rappresentano oggetti usati nel programma (ad esempio: variabili, costanti, tipi, funzioni, etc.)
- **costanti:** numeri (interi o reali), caratteri e stringhe
- **operatori:** sono simboli che consentono la combinazione di dati in espressioni
- **commenti**

Fondamenti di Informatica L- A

Elementi del programma in C

```

#include <stdio.h>
main() {
    // dichiarazione variabili
    int A, B;
    // input dei dati
    printf( "Immettere due numeri: " );
    scanf( "%d %d", &A, &B );
    // eseguo semisomma e mostro result.
    printf( "Semisomma: %d\n", (A+B)/2 );
}

```

Parole chiave

Esprimono istruzioni, tipi di dato, e altri elementi predefiniti nel linguaggio
 Sono parole riservate (cioè non possono essere utilizzate come identificatori)

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

Identificatori

Un identificatore è un nome che denota un oggetto usato nel programma (es.: **variabili, costanti, tipi, funzioni**).

- Deve iniziare con una lettera (o con il carattere '_'), alla quale possono seguire lettere e cifre in numero qualunque:

<identificatore> ::= <lettera> { <lettera> | <cifra> }

- distinzione tra maiuscole e minuscole (**case-sensitive**)

Es.: Alfa , beta , Gamma1 , X3 sono identificatori validi
3X , int non sono identificatori validi

Regola Generale:
 prima di essere *usato*, ogni identificatore deve essere già stato **definito** in una parte di testo precedente.

Costanti

Valori interi : Rappresentano numeri relativi (quindi con segno):

	2 byte	4 byte
base decimale	12	70000, 12L
base ottale	014	0210560
base esadecimale	0xFF	0x11170

Valori reali :

24.0 2.4E1 240.0E-1

Suffissi:	l, L, u, U	(interi - long, unsigned)
	f, F	(reali - floating)
Prefissi:	0	(ottale)
		0x, 0X (esadecimale)

Caratteri : Insieme dei caratteri disponibili (è dipendente dalla realizzazione). In genere, ASCII esteso (256 caratteri). Si indicano tra apici singoli:

'a' 'A' ' '

'1' ';' '\\'

Costanti

Caratteri speciali: sono caratteri ai quali non è associato alcun simbolo grafico, ai quali è associato un significato predefinito

<i>newline</i>	'\n'	se stampato, provoca l'avanzamento alla <i>linea</i> successiva
<i>backspace</i>	'\b'	se stampato, provoca l'arretramento al <i>carattere</i> precedente
<i>form feed</i>	'\f'	se stampato, provoca l'avanzamento alla <i>pagina</i> successiva
<i>carriage return</i>	'\r'	se stampato, provoca l'arretramento all'inizio della linea corrente

ecc.

Stringhe: Sono sequenze di caratteri tra doppi apici.

"a" "aaa" "" (stringa nulla)

Fondamenti di Informatica L- A

Commenti:

Sono sequenze di caratteri ignorate dal compilatore:

- vanno racchiuse tra /* e */ ...
- ...oppure precedute da //

```
/* questo codice non deve essere eseguito:
int X, Y;
*/
int A, B; // ho cambiato i nomi alle variabili
```

- I commenti vengono generalmente usati per introdurre **note esplicative** nel codice di un programma.

Fondamenti di Informatica L- A

Struttura di un programma C

Nel caso più semplice, un programma C consiste in:

```

<programma> ::= [ <parte-dich-globale> ]
                { <dich-funzione> }
                <main>
                { <dich-funzione> }
<main>       ::= main() {
                <parte-dichiarazioni>
                <parte-istruzioni> }
  
```

dichiarazioni: oggetti che verranno utilizzati dal main (variabili, tipi, costanti, etichette, etc.);

istruzioni: implementano l'algoritmo risolutivo utilizzato, mediante istruzioni del linguaggio.

Formalmente, il main è una funzione

Fondamenti di Informatica L- A

Esempio:

```

#include <stdio.h>
main() {
    // dichiarazione variabili
    int A, B;
    // input dei dati
    printf( "Immettere due numeri: " );
    scanf( "%d %d", &A, &B );
    // eseguo semisomma e mostro risult.
    printf( "Semisomma: %d\n", (A+B)/2 );
}
  
```

dichiarazioni

istruzioni

Fondamenti di Informatica L- A

Variabili

Una **variabile** rappresenta un dato che può cambiare il proprio valore durante l'esecuzione del programma.

In generale: nei linguaggi di alto livello una variabile è caratterizzata da un **nome** (*identificatore*) e 4 attributi base:

1. **scope** (*campo d'azione*), è l'insieme di istruzioni del programma in cui la variabile è nota e può essere manipolata;
C, Pascal, determinabile staticamente
LISP, dinamicamente
2. **tempo di vita** (o *durata* o *estensione*), è l'intervallo di tempo in cui un'area di memoria è legata alla variabile;
FORTRAN, allocazione statica
C, Pascal, allocazione dinamica
3. **valore**, è rappresentato (secondo la codifica adottata) nell'area di memoria legata alla variabile;
4. **tipo**, definisce l'insieme dei valori che la variabile può assumere e degli operatori applicabili.

Fondamenti di Informatica L- A

Variabili in C

- Ogni variabile, per poter essere utilizzata dalle istruzioni del programma, deve essere preventivamente definita.
- La definizione di variabile associa ad un identificatore (nome della variabile) un tipo.

```
<def-variabili> ::=  
  <identificatore-tipo> <identif-variabile> {, <identif-variabile> }  
  ;
```

Esempi:

```
int  A, B, SUM; /* Variabili A, B, SUM intere */  
float  root, Root; /* Variab. root, Root reali */  
char   C ;      /* Variabile C carattere */
```

Effetto della definizione di variabile:

- La definizione di una variabile provoca come effetto l'**allocazione in memoria** della variabile specificata (allocazione automatica).
- Ogni istruzione successiva alla definizione di una variabile A, potrà utilizzare A

Fondamenti di Informatica L- A

Assegnamento

- L'assegnamento è l'operazione con cui si attribuisce un (nuovo) valore ad una variabile.
- Il concetto di variabile nel linguaggio C rappresenta un'astrazione della cella di memoria.
- L'assegnamento, quindi, è l'astrazione dell'operazione di scrittura nella cella che la variabile rappresenta.

Esempi:

```

/* a, X e Y sono variabili */
int a ;
float X , Y ;
...
/* assegnamento ad a del valore 100: */
a = 100 ;
/* assegnamento a Y del risultato di una espr.
aritmetica: */
Y = 2 * 3.14 * X ;

```

Tipo di dato

Il **tipo di dato** rappresenta una categoria di dati caratterizzati da proprietà comuni.

In particolare:

un tipo di dato T è definito:

- dall'insieme di **valori (dominio)** che le variabili di tipo T possono assumere;
- dall'insieme di **operazioni** che possono essere applicate ad operandi del tipo T.

Per esempio:

Consideriamo i numeri naturali

Tipo_naturali = [N, {+, -, *, /, =, >, <, ... }]

- **N** è il *dominio*
- {+, -, *, /, =, >, <, ... } è l'insieme delle operazioni effettuabili sui valori del dominio.

Il Tipo di dato

Un linguaggio di programmazione è **tipato** se prevede **costrutti specifici** per attribuire tipi ai dati utilizzati nei programmi.

Se un linguaggio è tipato:

- Ogni dato (variabile o costante) del programma deve appartenere ad uno ed un solo tipo.
- Ogni operatore richiede operandi di tipo specifico e produce risultati di tipo specifico.

Vantaggi:

- **Astrazione:** l'utente esprime e manipola i dati ad un livello di astrazione più alto della loro organizzazione fisica (bit!).
Maggior portabilità.
- **Protezione:** Il linguaggio protegge l'utente da combinazioni errate di dati ed operatori (controllo statico sull'uso di variabili, etc. in fase di compilazione).
- **Portabilità:** l'indipendenza dall'architettura rende possibile la compilazione dello stesso programma su macchine profondamente diverse.

Fondamenti di Informatica L- A

Il tipo di dato in C

Il C è un linguaggio tipato.

Classificazione dei tipi di dato in C: due criteri di classificazione "ortogonali"

1. Si distingue tra:

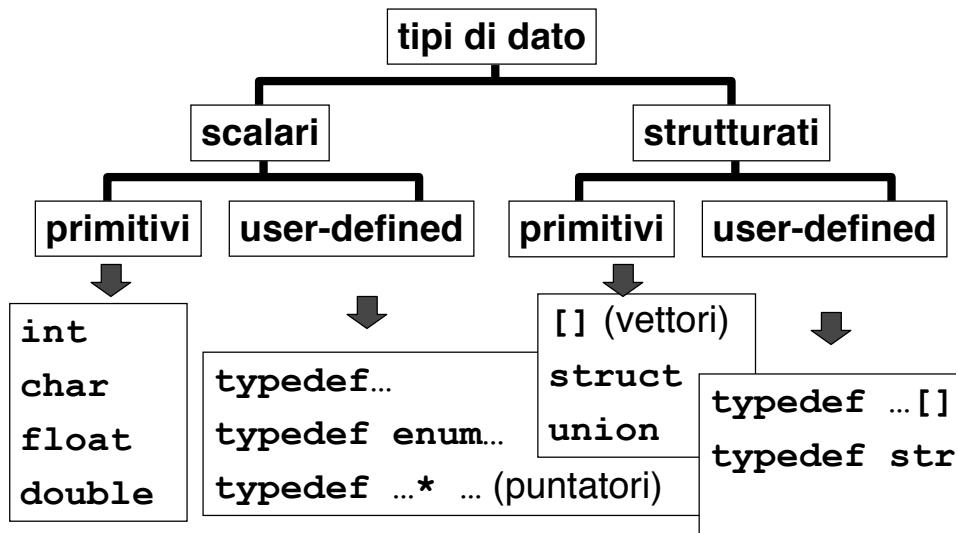
- **tipi scalari**, il cui dominio è costituito da elementi atomici, cioè logicamente non scomponibili.
- **tipi strutturati**, il cui dominio è costituito da elementi non atomici (e quindi scomponibili in altri componenti).

2. Inoltre, si distingue tra:

- **tipi primitivi:** sono tipi di dato previsti dal linguaggio (*built-in*) e quindi rappresentabili direttamente.
- **tipi non primitivi:** sono tipi definibili dall'utente (mediante appositi costruttori di tipo, v. `typedef`).

Fondamenti di Informatica L- A

Classificazione dei tipi di dato in C



Fondamenti di Informatica L- A

Tipi scalari primitivi

Il C prevede quattro tipi scalari primitivi:

- **char** (caratteri)
- **int** (interi)
- **float** (reali)
- **double** (reali in doppia precisione)

Fondamenti di Informatica L- A

Il tipo `int`

Dominio:

Il dominio associato al tipo `int` rappresenta l'insieme dei numeri interi con segno: ogni variabile di tipo `int` è quindi l'astrazione di un intero.

Esempio: definizione di una variabile intera.

(È la frase mediante la quale si *introduce* una nuova variabile nel programma.)

```
int A; /* A è una variabile intera */
```

- ⇒ Poiché si ha sempre a disposizione un numero finito di bit per la rappresentazione dei numeri interi, il dominio rappresentabile è di estensione finita.
- ⇒ Es: se il numero n di bit a disposizione per la rappresentazione di un intero è 16, allora il dominio rappresentabile è composto di:

$$(2^n) = 2^{16} = 65.536 \text{ valori}$$

Fondamenti di Informatica L- A

Quantificatori e qualificatori

A dati di tipo `int` è possibile applicare i quantificatori *short* e *long*: influiscono sullo spazio in memoria richiesto per l'allocazione del dato.

- `short`: la rappresentazione della variabile in memoria può utilizzare un numero di bit ridotto rispetto alla norma.
- `long`: la rappresentazione della variabile in memoria può utilizzare un numero di bit aumentato rispetto alla norma.

Esempio:

```
int X; /* se X e` su 16 bit..*/  
long int Y; /*..Y e` su 32 bit */
```

I quantificatori possono influire sul dominio delle variabili:

- Il dominio di un `long int` può essere più esteso del dominio di un `int`.
- Il dominio di uno `short int` può essere più limitato del dominio di un `int`.

Gli effetti di `short` e `long` sui dati dipendono strettamente dalla realizzazione del linguaggio.

In generale:

$$\text{spazio}(\text{short int}) \leq \text{spazio}(\text{int}) \leq \text{spazio}(\text{long int})$$

Fondamenti di Informatica L- A

Quantificatori e qualificatori

A dati di tipo `int` e' possibile applicare i qualificatori *signed* e *unsigned*:

- **signed**: viene usato un bit per rappresentare il segno. Quindi l'intervallo rappresentabile e':
 $[-2^{n-1}-1, +2^{n-1}-1]$
- **unsigned**: vengono rappresentati valori a priori positivi. Intervallo rappresentabile:
 $[0, (2^n - 1)]$

I qualificatori condizionano il dominio dei dati.

Fondamenti di Informatica L- A

Il tipo int

Operatori:

Al tipo `int` (e tipi ottenuti da questo mediante qualificazione/quantificazione) sono applicabili gli operatori *aritmetici*, *relazionali* e *logici*.

Operatori aritmetici:

forniscono risultato intero:

- | | |
|---|--|
| <code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> | somma, sottrazione, prodotto, divisione intera. |
| <code>%</code> | operatore modulo: calcola il resto della divisione intera.
<code>10%3 → 1</code> |
| <code>++</code> , <code>--</code> | incremento e decremento: richiedono un solo operando (una variabile) e possono essere postfixi (<code>a++</code>) o prefissi (<code>++a</code>) (v. espressioni) |

Fondamenti di Informatica L- A

Operatori relazionali

Si applicano ad operandi interi e producono risultati *logici* (o “*booleani*”).

Booleani: sono grandezze il cui dominio e` di due soli valori (valori logici): {*vero*, *falso*}

Operatori relazionali:

$==, !=$ uguaglianza ($==$), disuguaglianza ($!=$):
 $10==3 \rightarrow falso$
 $10!=3 \rightarrow vero$

$<, >, <=, >=$ minore, maggiore, minore o uguale, maggiore o uguale
 $10>=3 \rightarrow vero$

Fondamenti di Informatica L- A

Booleani

Sono dati il cui dominio e` di due soli valori (valori logici):

{*vero*, *falso*}

→ in C **non esiste un tipo primitivo per rappresentare dati booleani !**

Come vengono rappresentati i risultati di espressioni relazionali ?

Il C prevede che i valori logici restituiti da espressioni relazionali vengano rappresentati attraverso gli interi {0,1} secondo la convenzione:

- 0 equivale a *falso*
- 1 equivale a *vero*

Ad esempio:

l'espressione $A == B$ restituisce:

- 0, se la relazione non e` vera
- 1, se la relazione e` vera

Fondamenti di Informatica L- A

Operatori logici

Si applicano ad operandi di tipo logico (in C: `int`) e producono risultati *booleani* (cioe' interi appartenenti all'insieme {0,1}).

In particolare l'insieme degli operatori logici e':

- `&&` operatore AND logico
- `||` operatore OR logico
- `!` operatore di negazione (NOT)

Definizione degli operatori logici:

a	b	a&&b	a b	!a
falso	falso	falso	falso	vero
falso	vero	falso	vero	vero
vero	falso	falso	vero	falso
vero	vero	vero	vero	falso

Operatori Logici in C

In C, gli operandi di operatori logici sono di tipo `int`:

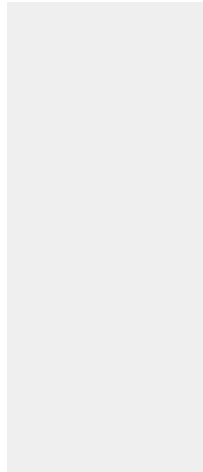
- se il valore di un operando e' **diverso da zero**, viene interpretato come **vero**.
- se il valore di un operando e' **uguale a zero**, viene interpretato come **falso**.

Definizione degli operatori logici in C:

a	b	a&&b	a b	!a
■	■	■	■	!
■	■ ■	■	!	!
■ ■	■	■	!	■
■ ■	■ ■	!	!	■

Operatori tra interi: esempi

```
37 / 3
37 % 3
7 < 3
7 >=3
5 >=5
0 || 1
0 || -123
15||0
12 && 2
0 && 17
! 2
```



Fondamenti di Informatica L- A

I tipi reali: float e double

Dominio:

Concettualmente, è l'insieme dei numeri reali \mathbb{R} .

In realtà, è un sottoinsieme di \mathbb{R} :

- limitatezza del dominio (limitato numero di bit per la rappresentazione del valore).
- precisione limitata: numero di bit finito per la rappresentazione della parte frazionaria (*mantissa*)

Lo spazio allocato per ogni numero reale (e quindi l'insieme dei valori rappresentabili) dipende dalla realizzazione del linguaggio (e, in particolare, dal metodo di rappresentazione adottato).

Differenza tra float/double:

<code>float</code>	singola precisione
<code>double</code>	doppia precisione (maggiore numero di bit per la <i>mantissa</i>)

→ Alle variabili `double` e' possibile applicare il quantificatore `long`, per aumentare ulteriormente la precisione: `spazio(float) <= spazio(double) <= spazio(long double)`

Esempio: definizione di variabili reali

```
float x; /* x e` una variabile reale "a singola precisione"*/
double A, B; /* A e B sono reali "a doppia precisione"*/
```

Fondamenti di Informatica L- A

I tipi float e double

Operatori: per dati di tipo reale sono disponibili operatori aritmetici e relazionali.

Operatori aritmetici:

`+, -, *, /` si applicano a operandi reali e producono risultati reali

Operatori relazionali: hanno lo stesso significato visto nel caso degli interi:

`==, !=` uguale, diverso
`<, >, <=, >=` minore, maggiore etc.

Operazioni su reali: esempi

`5.0 / 2` → 
`2.1 / 2` → 
`7.1 < 4.55` → 
`17 == 121` → 

→ A causa della rappresentazione finita, ci possono essere errori di conversione. Ad esempio, i test di uguaglianza tra valori reali (in teoria uguali) potrebbero non essere verificati.

$$(x / y) * y == x$$

Meglio utilizzare "un margine accettabile di errore":

`(X == Y)` → `(X <= Y + epsilon) && (X >= Y - epsilon)`

dove, ad esempio: `float epsilon = 0.000001;`

Il tipo char

Dominio:

È l'insieme dei caratteri disponibili sul sistema di elaborazione (set di caratteri).

Comprende:

- le lettere dell'alfabeto
- le cifre decimali
- i simboli di punteggiatura
- altri simboli di vario tipo (@, #, \$ etc.)
- caratteri speciali (backspace, carriage return, ecc.)
- ...

Tabella dei Codici

Il dominio coincide con l'insieme rappresentato da una **tabella dei codici**, dove, ad ogni carattere viene associato un intero che lo identifica univocamente (il **codice**).

- Il dominio associato al tipo char e' **ordinato**: l'ordine dipende dal codice associato ai vari caratteri.

La tabella dei codici ASCII

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
0	(null)	NUL	32	(space)	64	@	96	a
1	☉	SOH	33	!	65	A	97	b
2	☉	STX	34	"	66	B	98	c
3	♥	ETX	35	#	67	C	99	d
4	♦	EOT	36	\$	68	D	100	e
5	♣	ENQ	37	%	69	E	101	f
6	▲	ACK	38	&	70	F	102	g
7	(beep)	BEL	39	'	71	G	103	h
8	■	BS	40	(72	H	104	i
9	(tab)	HT	41)	73	I	105	j
10	(line feed)	LF	42	*	74	J	106	k
11	(home)	VT	43	+	75	K	107	l
12	(form feed)	FF	44	,	76	L	108	m
13	(carriage return)	CR	45	-	77	M	109	n
14	♪	SO	46	.	78	N	110	o
15	☉	SI	47	/	79	O	111	p
16	▲	DLE	48	0	80	P	112	q
17	▲	DC1	49	1	81	Q	113	r
18	↕	DC2	50	2	82	R	114	s
19	!!	DC3	51	3	83	S	115	t
20	π	DC4	52	4	84	T	116	u
21	\$	NAK	53	5	85	U	117	v
22	■	SYN	54	6	86	V	118	w
23	↕	ETB	55	7	87	W	119	x
24	↕	CAN	56	8	88	X	120	y
25	↕	EM	57	9	89	Y	121	z
26	←	SUB	58	:	90	Z	122	{
27	←	ESC	59	;	91	[123	
28	(cursor right)	FS	60	<	92	\	124	}
29	(cursor left)	GS	61	=	93]	125	~
30	(cursor up)	RS	62	>	94	^	126	␣
31	(cursor down)	US	63	?	95	_	127	␣

Il tipo char

Definizione di variabili di tipo char: esempio

```
char C1, C2;
```

Costanti di tipo char:

Ogni valore di tipo char viene specificato tra singoli apici.

```
'a' 'b' 'A' '0' '2'
```

Rappresentazione dei caratteri in C:

Il linguaggio C rappresenta i dati di tipo char come degli interi su 8 bit:

- ogni valore di tipo char viene rappresentato dal suo **codice** (cioè, dall'intero che lo indica nella tabella ASCII)
- ➔ Il dominio associato al tipo `char` è **ordinato**: l'ordine dipende dal codice associato ai vari caratteri nella tabella di riferimento.

Fondamenti di Informatica L- A

Il tipo char: Operatori

I char sono rappresentati da interi (su 8 bit):

- ➔ sui char è possibile eseguire tutte le operazioni previste per gli interi. Ogni operazione, infatti, è applicata ai codici associati agli operandi.

Operatori relazionali:

`==, !=, <, <=, >=, >` per i quali valgono le stesse regole viste per gli interi

Ad esempio:

```
char x, y;
```

```
x < y se e solo se codice(x) < codice(y)
```

```
'a' > 'b' falso perché codice('a') < codice('b')
```

Operatori aritmetici:

sono gli stessi visti per gli interi.

Operatori logici:

sono gli stessi visti per gli interi.

Fondamenti di Informatica L- A

Operazioni sui char: esempi

Esempi:

'A' < 'C' →

'"' + '!' →

!'A' →

'A' && 'a' →

Fondamenti di Informatica L- A

OVERLOADING

- In C (come in Pascal, Fortran e molti altri linguaggi) operazioni primitive associate a tipi diversi possono essere denotate con lo stesso simbolo (ad esempio, le operazioni aritmetiche su reali o interi).
- In realtà l'operazione è diversa e può produrre risultati diversi. Per esempio:

```
int X,Y;  
se X = 10 e Y = 4;  
X/Y vale ...
```

```
int X; float Y;  
se X = 10 e Y = 4.0;  
X/Y vale ...
```

```
float X,Y;  
se X = 10.0 e Y = 4.0;  
X/Y vale ...
```

Fondamenti di Informatica L- A

CONVERSIONI DI TIPO

In C è possibile combinare tra di loro operandi di tipo diverso:

- espressioni **omogenee**: tutti gli operandi sono dello stesso tipo
- espressioni **eterogenee**: gli operandi sono di tipi diversi.

- **Regola adottata in C:**

- sono eseguibili le espressioni eterogenee in cui tutti i tipi referenziati risultano **compatibili** (cioè: dopo l'applicazione della **regola automatica di conversione implicita** di tipo del C risultano omogenei).

Conversione implicita di tipo

Data una espressione $x \text{ op } y$.

1. Ogni variabile di tipo **char** o **short** viene convertita nel tipo **int**;
2. Se dopo l'esecuzione del passo 1 l'espressione è ancora eterogenea, rispetto alla seguente gerarchia

int < long < float < double < long double

 si converte temporaneamente l'operando di tipo *inferiore* al tipo *superiore* (**promotion**);
3. A questo punto l'espressione è **omogenea**. L'operazione specificata può essere eseguita se il tipo degli operandi è compatibile con il tipo dell'operatore. Il risultato è di tipo uguale a quello prodotto dall'operatore effettivamente eseguito.

Conversione implicita di tipo: esempio

```
int x;  
char y;  
double r;  
(x+y) / r
```



Hp: La valutazione dell'espressione
procede da sinistra verso destra

- **Passo 1:** $(x+y)$
 - y viene convertito nell'intero corrispondente
 - viene applicata la somma tra interi
 - **risultato intero:** tmp
- **Passo 2**
 - tmp / r tmp viene convertito nel double corrispondente
 - viene applicata la divisione tra reali
 - **risultato reale (double)**

Fondamenti di Informatica L- A



Conversione esplicita di tipo

In C si può forzare la conversione di un dato in un tipo specificato, mediante l'operatore di cast:

(<nuovo tipo>) <dato>

→ il <dato> viene convertito esplicitamente nel <nuovo tipo>.

Esempio:

```
int A, B;  
float C;  
C=A/(float)B;
```

→ viene eseguita la divisione tra reali.

Fondamenti di Informatica L- A

Definizione / inizializzazione di variabili di tipo semplice

Definizione di variabili

Tutti gli identificatori di tipo primitivo descritti fin qui possono essere utilizzati per definire variabili.

Ad esempio:

```
char lettera;  
int, x, y;  
unsigned int P;  
float media;
```

Inizializzazione di variabili

E' possibile specificare un valore iniziale di una variabile in fase di definizione.

Ad esempio:

```
int x =10;  
char y = 'a';  
double r = 3.14*2;  
int z=x+5;
```

Fondamenti di Informatica L- A

Dichiarazioni e Definizioni

Nella parti dichiarative di un programma C possiamo incontrare:

- definizioni (di variabile, o di funzione)
- dichiarazioni (di tipo o di funzione)

Definizione:

Descrive le proprietà dell'oggetto definito e ne determina l'esistenza.

Ad esempio:

```
int V; /* definizione della variabile intera V */
```

Dichiarazione:

Descrive soltanto delle proprietà di oggetti, che verranno (eventualmente) creati mediante definizione.

Ad esempio: dichiarazione di un tipo di dato non primitivo:

```
typedef .... newT; /* newT è un tipo non primitivo*/
```

Fondamenti di Informatica L- A

Dichiarazione di tipo

La dichiarazione di tipo serve per introdurre tipi non primitivi.

```
typedef <descrizione-nuovo-tipo> newT;
```

- si utilizza la parola chiave `typedef`.
- la dichiarazione associa ad un tipo di dato non primitivo un identificatore arbitrario (`newT`)
- le caratteristiche del nuovo tipo sono indicate in `<descrizione-nuovo-tipo>`

L'introduzione di tipi non primitivi aumenta la **leggibilità** e **modificabilità** del programma.

Fondamenti di Informatica L- A

Tipi scalari non primitivi

In C sono possibili dichiarazioni di tipi scalari non primitivi:

- tipi **ridefiniti**
- [tipi **enumerati**] (non li tratteremo)

Tipo ridefinito:

Si ottiene associando un nuovo identificatore a un tipo già esistente (primitivo o non).

Sintassi:

```
typedef <id-tipo-esistente> <id-nuovo-tipo> ;
```

Esempio:

```
typedef int MioIntero; /* MioIntero è un tipo non
                        primitivo che ridefinisce il
                        tipo int*/
MioIntero X;          /* X è di tipo MioIntero */
int Y;                /* Y è di tipo int */
```

→ X e Y rappresentano entrambi valori interi (**strutturalmente** equivalenti), ma **nominalmente** sono di tipo diverso.

Fondamenti di Informatica L- A

Il linguaggio C

Istruzione di assegnamento e operatori

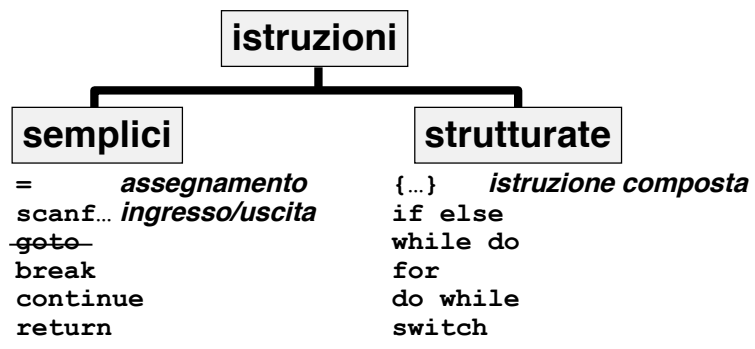
```
main()
{
  /*definizioni variabili: */
  char y='a'; /*codice(a)=97*/
  int x,X,Y;
  unsigned int Z;
  float SUM;
  double r;

  /* parte istruzioni: */
  X=27;
  Y=343;
  Z = X + Y -300;
  X = Z / 10 + 23;
  Y = (X + Z) / 10 * 10;      /* qui X=30, Y=100, Z=70 */
  X = X + 70;
  Y = Y % 10;
  Z = Z + X -70;
  SUM = Z * 10;              /* X=100, Y=0, Z=100 , SUM=1000.0*/
  x=y;                       /* char -> int: x=97*/
  x=y+x; /*x=194*/
  r=y+1.33; /* char -> int -> double*/
  x=r; /* coercizione -> troncamento: x=98*/
}
```

Istruzioni: classificazione

In C, le istruzioni possono essere classificate in due categorie:

- istruzioni **semplici**
- istruzioni **strutturate**: si esprimono mediante composizione di altre istruzioni (semplici e/o strutturate).



Istruzione di Assegnamento

- È l'istruzione con cui si modifica il valore di una variabile. Mediante l'assegnamento, si scrive un nuovo valore nella cella di memoria che rappresenta la variabile specificata.

Sintassi:

```
<istruzione-assegnamento> ::=  
  <identificatore-variabile> = <espressione> ;
```

Ad esempio:

```
int A, B;  
  
A=20;  
B=A*5; /* B=100 */
```

Compatibilità di tipo ed assegnamento:

In un assegnamento, l'identificatore di variabile e l'espressione devono essere dello stesso tipo (eventualmente, conversione implicita oppure **coercizione**).

Fondamenti di Informatica L- A

Assegnamento e Coercizione

Facendo riferimento alla gerarchia dei tipi semplici:

```
int < long < float < double < long double
```

consideriamo l'assegnamento:

```
V=E;
```

Quando la variabile V è di un tipo di *rango inferiore* rispetto al tipo dell'espressione E l'assegnamento prevede la conversione forzata (**coercizione**) di E nel tipo di V.

Ad esempio:

```
float k=1.5;  
int x;  
x=k; /*il valore di k viene convertito forzatamente nella  
sua parte intera: x assume il valore 1 ! */
```

Fondamenti di Informatica L- A

Esempio:

```
main()
{
  /*definizioni variabili: */
  char y='a'; /*codice(a)=97*/
  int    x,X,Y;
  unsigned int Z;
  float SUM;
  double r;

  /* parte istruzioni: */
  X=27;
  Y=343;
  Z = X + Y -300;
  X = Z / 10 + 23;
  Y = (X + Z) / 10 * 10;
  X = X + 70;
  Y = Y % 10;
  Z = Z + X -70;
  SUM = Z * 10;
  x=y;
  x=y+x;
  r=y+1.33;
  x=r;
}
```

Fondamenti di Informatica L- A

Assegnamento come operatore

Formalmente, l'istruzione di assegnamento è un'espressione:

- Il simbolo = è un operatore:
 - l'istruzione di assegnamento è una espressione
 - ritorna un valore:
 - il valore restituito è quello assegnato alla variabile a sinistra del simbolo =
 - il tipo del valore restituito è lo stesso tipo della variabile oggetto dell'assegnamento

Ad esempio:

```
int valore=122;
int K, M;
```

```
K=valore+100;
```

```
M= (K=K/2)+1;
```

Fondamenti di Informatica L- A

Assegnamento abbreviato

In C sono disponibili operatori che realizzano particolari forme di assegnamento:

- operatori di incremento e decremento
- operatori di assegnamento abbreviato
- operatore sequenziale

• **Operatori di incremento e decremento:**

Determinano l'incremento/decremento del valore della variabile a cui sono applicati.

Restituiscono come risultato il valore incrementato/decrementato della variabile a cui sono applicati.

```
int A=10;

A++; /*equivale a: A=A+1; */
A--; /*equivale a: A=A-1; */
```

Differenza tra notazione prefissa e postfissa:

- Notazione **Prefissa**: (ad esempio, ++A) significa che l'incremento viene fatto prima dell'impiego del valore di A nella espressione.
- Notazione **Postfissa**: (ad esempio, A++) significa che l'incremento viene effettuato dopo l'impiego del valore di A nella espressione.

Incremento & decremento: esempi

```
int A=10, B;
char C='a';
```

```
B=++A;
B=A++;
C++;
```

```
int i, j, k;
k = 5;
i = ++k;
j = i + k++;
j = i + k++;
```

- In C l'ordine di valutazione degli operandi non e' indicato dallo standard: si possono scrivere espressioni il cui valore e' difficile da predire:

```
k = 5;
j = ++k * k++; /* quale effetto ?*/
```

Operatori di assegnamento abbreviato

E' un modo sintetico per modificare il valore di una variabile.

Sia *v* una variabile, *op* un'operatore (ad esempio, +,-,/, etc.), ed *e* una espressione.

$$v \text{ op } = e$$

è quasi equivalente a:

$$v = v \text{ op } (e)$$

Ad esempio:

<code>k += j;</code>	<code>/* equivale a k = k + j */</code>
<code>k *= a + b;</code>	<code>/* equivale a k = k * (a + b) */</code>

Le due forme sono **quasi** equivalenti perchè:

- in `v op = e` *v* viene valutato una sola volta;
- in `v = v op (e)` *v* viene valutato due volte.

Operatore sequenziale

Un'espressione sequenziale (o di **concatenazione**) si ottiene concatenando tra loro più espressioni con l'operatore virgola (,).

`(<espr1>, <espr2>, <espr3>, .. <esprN>)`

- Il risultato prodotto da un'espressione sequenziale è il risultato ottenuto dall'ultima espressione della sequenza.
- La valutazione dell'espressione avviene valutando nell'ordine testuale le espressioni componenti, **da sinistra verso destra**.

Esempio:

```
int A=1;
char B;
A=(B='k', ++A, A*2);
```