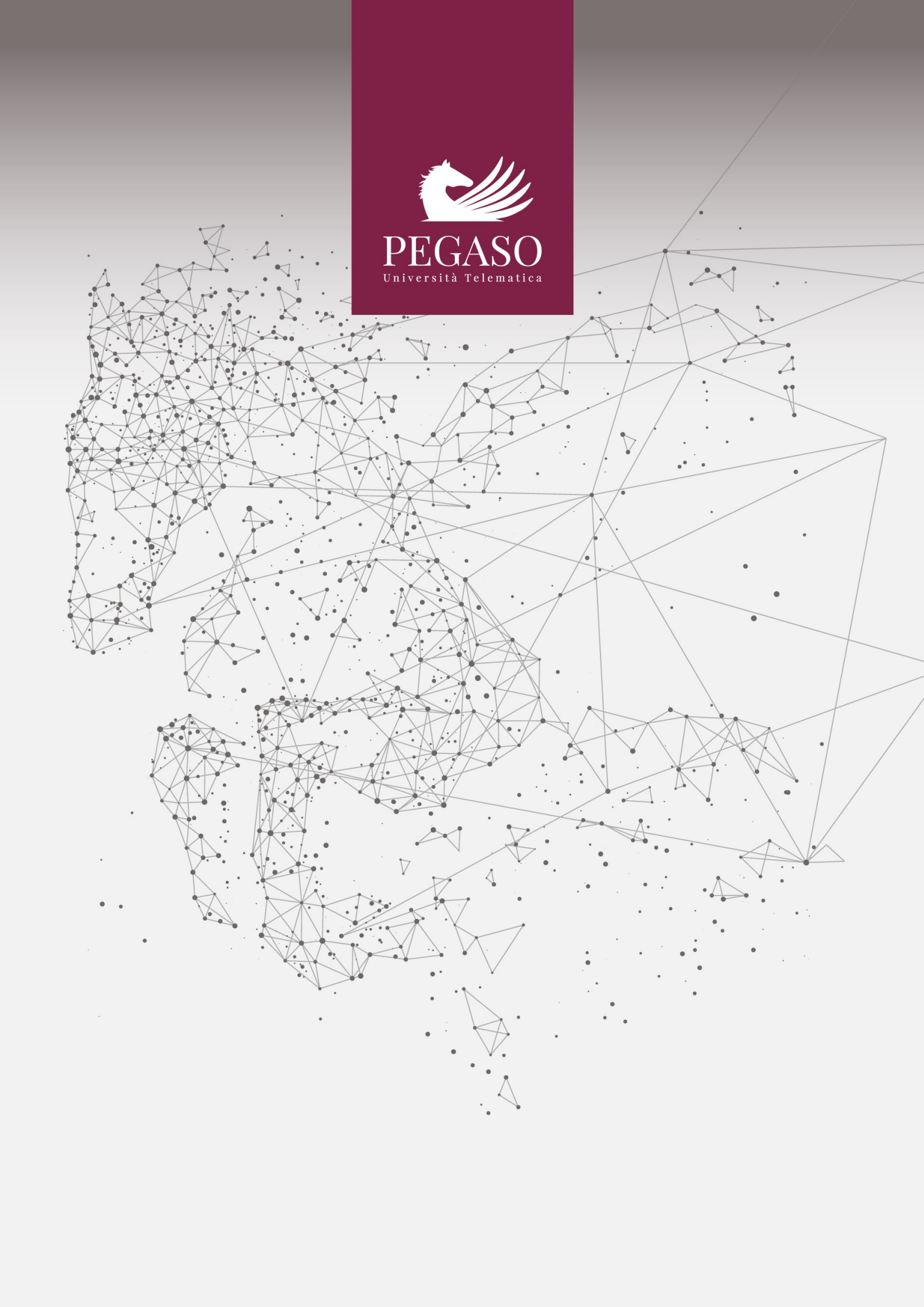
PEGASO
Università Telematica

Indice

1. Concetti relativi alla memoria

Nomi di variabili come integer1, integer2 e sum corrispondono in realtà a locazioni nella memoria del computer.

Ogni variabile ha un nome, un tipo e un valore.

Quando viene eseguita questa istruzione

scanf("%d", &integer1); // legge un intero

il valore inserito dall'utente viene posto in una locazione di memoria alla quale è stato assegnato il nome integer1.

Supponete che l'utente inserisca il numero 45 come valore per integer1.

Il computer porrà il valore 45 nella locazione integer1.

Ogni volta che un valore è posto in una locazione di memoria, tale valore sostituisce quello precedente in quella locazione e il valore precedente va perso; per tale motivo, questo processo si dice *distruttivo*.

Quando viene eseguita l'istruzione:

scanf("%d", &integer2); // legge un intero

il valore inserito dall'utente viene posto in una locazione di memoria differente, alla quale è stato assegnato il nome integer2.

Supponete che l'utente inserisca il numero 72 come valore per integer2.

Il computer porrà il valore 72 nella locazione integer2.

Queste locazioni non sono necessariamente adiacenti.

L'istruzione successiva

sum = integer1 + integer2; // assegna il totale a sum

addiziona i valori di integer1 e integer2 e memorizza il totale nella variabile sum.

Qualsiasi valore che era stato memorizzato in precedenza in sum viene perso.

I valori di integer1 e integer2 rimangono esattamente come erano prima di essere usati nel calcolo.

Durante il calcolo sono stati usati, ma non distrutti. Pertanto, quando un valore viene semplicemente letto dalla locazione di memoria, il processo si dice *non* distruttivo.

2. Operatori aritmetici in C

La maggior parte dei programmi in C esegue calcoli usando gli operatori aritmetici del C. Notate l'uso di vari simboli speciali non usati comunemente in algebra.

L'asterisco (*) indica la moltiplicazione e il segno di percentuale (%) indica l'operatore di resto.

In algebra, per moltiplicare a per b, mettiamo semplicemente l'uno accanto all'altro questi nomi di variabile costituiti da singole lettere, come in ab. Se facessimo questo in C, ab sarebbe interpretato come un nome singolo di due lettere (o identificatore).

Quindi il C (così come molti altri linguaggi di programmazione) richiede che la moltiplicazione sia indicata esplicitamente usando l'operatore *, come in a * b.

Gli operatori aritmetici sono tutti operatori binari.

Ad esempio, l'espressione 3 + 7 contiene l'operatore binario + e gli operandi 3 e 7.

Operazione in C	Operatore aritmetico	Espressione algebrica	Espressione in C
Addizione	+	f + 7	f + 7
Sottrazione	-	p - c	p - c
Moltiplicazione	*	bm	b * m
Divisione	/	x/y	x / s
Resto	%	r mod s	r % s

Divisione intera e operatore di resto

La divisione intera restituisce un risultato intero.

Ad esempio, l'espressione 7 / 4 restituisce il valore 1 e l'espressione 17 / 5 restituisce il valore 3.

Il C ha l'operatore di resto, %, che calcola il resto di una divisione intera.

L'operatore di resto è un operatore intero che può essere usato soltanto con operandi interi.

L'espressione x % y calcola il resto della divisione di x per y.

Così, 7 % 4 restituisce il valore 3 e 17 % 5 restituisce il valore 2.

Anticipiamo che esistono molte interessanti applicazioni dell'operatore di resto.

☹ *Errore comune di programmazione: Un tentativo di dividere per zero porta a un risultato indefinito sui computer e generalmente provoca un errore* irreversibile*, ossia un errore che causa l'interruzione immediata di un programma, senza che questo porti a termine con successo il suo lavoro. Gli errori* non irreversibili *permettono ai programmi di completare l'esecuzione, producendo spesso risultati scorretti.*

Espressioni aritmetiche in forma lineare

Le espressioni aritmetiche in C devono essere scritte in forma lineare per facilitare l'inserimento di programmi nel computer.

Così, espressioni come "a diviso b" devono essere scritte come a / b,

in modo che tutti gli operatori e gli operandi compaiano in una configurazione lineare.

3. Espressioni algebriche in C

Parentesi per raggruppare sottoespressioni

Le parentesi sono usate nelle espressioni in C allo stesso modo che nelle espressioni algebriche.

Ad esempio, per moltiplicare per a la quantità b + c, scriviamo a * (b + c).

Regole di precedenza degli operatori

Il C applica gli operatori nelle espressioni aritmetiche in una precisa sequenza, determinata dalle seguenti regole di precedenza degli operatori, che generalmente sono le stesse di quelle dell'algebra:

1. Gli operatori nelle espressioni contenute entro una coppia di parentesi sono applicati per primi. Le parentesi si considerano al "massimo livello di precedenza". In caso di parentesi annidate, come ((a + b) + c) gli operatori nella coppia di parentesi più interne sono applicati per primi.
2. Le operazioni di moltiplicazione, divisione e resto sono calcolate subito dopo. Se un'espressione contiene diverse operazioni di moltiplicazione, divisione e resto, il calcolo procede da sinistra a destra. Si dice che la moltiplicazione, la divisione e il resto hanno lo stesso livello di precedenza.
3. Segue il calcolo delle operazioni di addizione e sottrazione. Se un'espressione contiene diverse operazioni di addizione e sottrazione, il calcolo procede da sinistra a destra. L'addizione e la sottrazione hanno anche lo stesso livello di precedenza, che è più basso di quello delle operazioni di moltiplicazione, divisione e resto.
4. L'operatore di assegnazione (=) è valutato per ultimo.

Le regole di precedenza degli operatori specificano l'ordine che viene usato per calcolare le espressioni in C.

Quando diciamo che il calcolo procede da sinistra a destra, ci riferiamo all'associatività degli operatori.

Vedremo che alcuni operatori sono associativi da destra a sinistra.

Esempi di espressioni algebriche e di espressioni in C

La seguente espressione calcola la media aritmetica di cinque termini.

m = (a + b + c + d + e) / 5;

Le parentesi sono richieste per raggruppare le addizioni, perché la divisione ha una precedenza più alta dell'addizione.

L'intera quantità (a + b + c + d + e) deve essere divisa per 5.

Se le parentesi sono erroneamente omesse, otteniamo che solo e verrà diviso per 5

La seguente espressione è l'equazione di una linea retta:

y = m * x + b;

Non sono richieste parentesi.

La moltiplicazione è calcolata per prima, perché la moltiplicazione ha una precedenza più alta dell'addizione.

La seguente espressione contiene le operazioni di resto (%), moltiplicazione, divisione, addizione, sottrazione e assegnazione:

z = p * r % q + w / x - y;

6 1 2 4 3 5

I numeri sottolineati indicano l'ordine in cui vengono applicati gli operatori in C.

La moltiplicazione, il resto e la divisione sono calcolati per primi nell'ordine da sinistra a destra (cioè sono associativi da sinistra a destra), perché hanno una precedenza più alta dell'addizione e della sottrazione.

L'addizione e la sottrazione sono calcolate successivamente, anch'esse da sinistra a destra. Alla fine, il risultato è assegnato alla variabile z.

Non tutte le espressioni con diverse coppie di parentesi contengono parentesi annidate. Ad esempio, l'espressione seguente non contiene parentesi annidate; le parentesi sono "allo stesso livello".

a * (b + c) + c * (d + e)

Calcolo di un polinomio di secondo grado

Per avere una comprensione migliore delle regole di precedenza degli operatori, vediamo come viene calcolato un polinomio di secondo grado in C.

y = a * x * x + b * x + c;

6 1 2 4 3 5

I numeri sottolineati sotto l'istruzione indicano l'ordine in cui vengono eseguite le operazioni in C.

Non c'è alcun operatore aritmetico per l'esponente in C, così abbiamo scritto x^2 come x * x.

Supponete che le variabili a, b, c e x nel precedente polinomio di secondo grado siano inizializzate come segue: a = 2, b = 3, c = 7 e x = 5.

y = 2 * 5 * 5 + 3 * 5 + 7; (moltiplicazione delle cifre più a sinistra)

y = 10 * 5 + 3 * 5 + 7; (moltiplicazione delle cifre più a sinistra)

y = 50 + 3 * 5 + 7; (moltiplicazione prima dell'addizione)

y = 50 + 15 + 7; (addizione delle cifre più a sinistra)

y = 65 + 7; (ultima addizione)

y = 72; (ultima operazione: assegnare 72 a y)

Utilizzare parentesi per chiarezza

Come in algebra, è accettabile inserire parentesi non necessarie in un'espressione per renderla più chiara.

Queste sono chiamate parentesi ridondanti.

Ad esempio, l'istruzione precedente potrebbe essere scritta con parentesi come segue:

```
y = ( a * x * x ) + ( b * x ) + c;
```

Bibliografia

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.