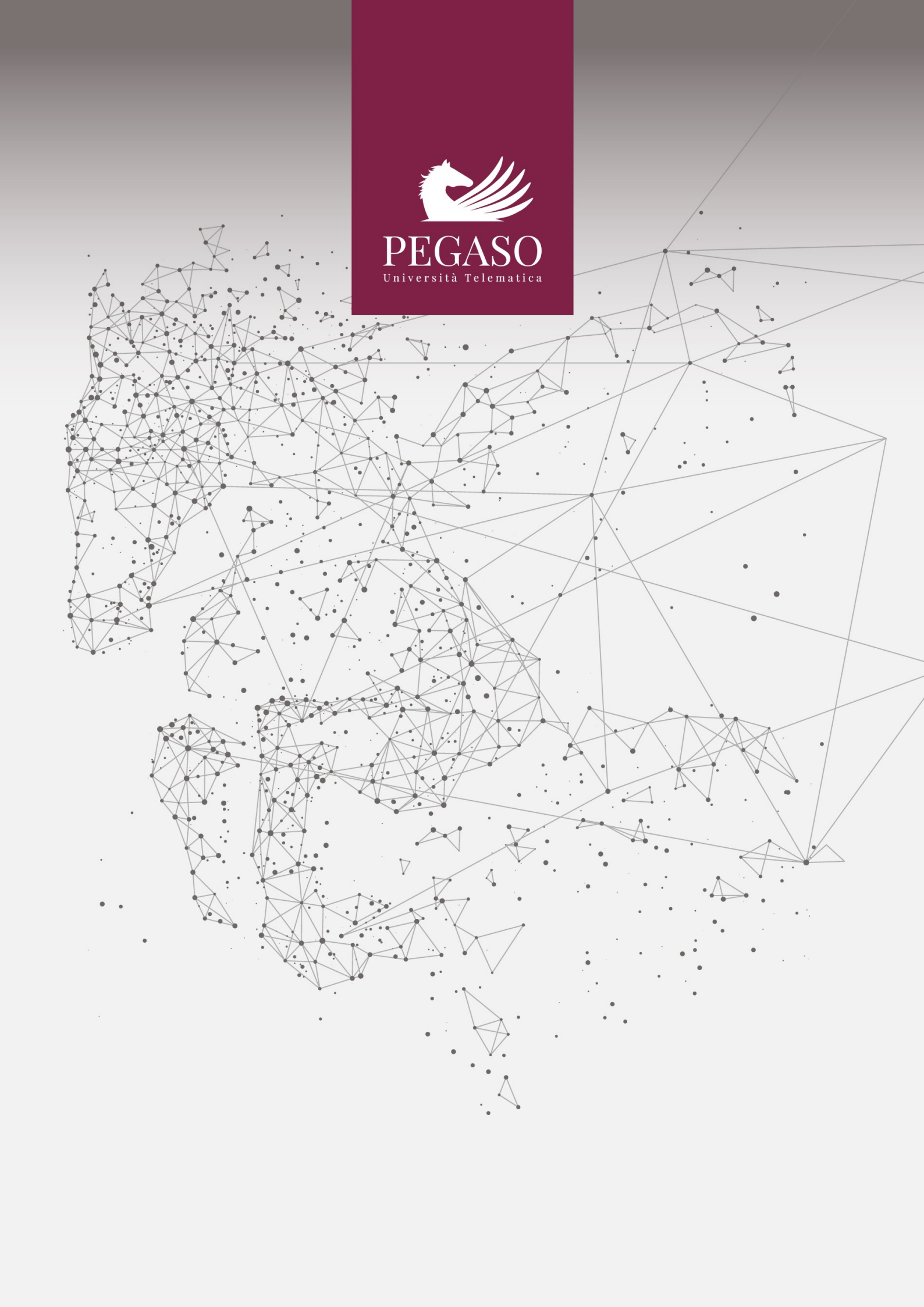
PEGASO
Università Telematica

Indice

1. Operatori relazionali e di uguaglianza

Le istruzioni eseguibili sono di due tipi:

- eseguono azioni (come calcoli, oppure input o output di dati)
- oppure prendono decisioni

Potremmo prendere una decisione in un programma, ad esempio, per stabilire se il voto di una persona a un esame sia maggiore o uguale a 18 e se il programma debba stampare il messaggio "Congratulazioni! Sei stato promosso."

Introduciamo l'istruzione if del C, che permette a un programma di prendere una decisione basata sulla verità o falsità di un'espressione chiamata condizione.

Se la condizione è vera (cioè la condizione è soddisfatta), l'istruzione nel corpo dell'istruzione if viene eseguita.

Se la condizione è falsa (cioè la condizione non è soddisfatta), l'istruzione nel corpo non viene eseguita.

A prescindere dal fatto che l'istruzione nel corpo venga eseguita o meno, al termine dell'istruzione if l'esecuzione procede con l'istruzione successiva all'istruzione if.

Le condizioni nelle istruzioni if vengono costruite usando gli operatori relazionali e gli operatori di uguaglianza riassunti di seguito

Operatore relazionale	Operatore relazionare in C	Esempio di condizione in C	Significato
>	>	x > y	x è maggiore di y
<	<	x < y	x è minore di y
≥	>=	x >= y	x è maggiore o uguale a y
≤	<=	x <= y	x è minore o uguale a y

Gli operatori relazionali hanno tutti lo stesso livello di precedenza e sono associativi da sinistra a destra.

Operatore di uguaglianza	Operatore di uguaglianza in C	Esempio in C	Significato
=	==	x == y	x è uguale a y
≠	!=	x != y	x non è uguale a y

Gli operatori di uguaglianza hanno un livello di precedenza più basso degli operatori relazionali e sono associativi da sinistra a destra.

Nota: in C una condizione può essere qualsiasi espressione che genera un valore zero (falso) o un valore diverso da zero (vero).

☹ *Errore comune di programmazione: Si commette un errore di sintassi se i due simboli in uno qualsiasi degli operatori ==, !=, >= e <= sono separati da spazi.*

☹ *Errore comune di programmazione: Confondere l'operatore di uguaglianza == con l'operatore di assegnazione =.*

Per evitare questa confusione, l'operatore di uguaglianza va letto "doppio uguale" e l'operatore di assegnazione va letto "assume il valore." Confondere questi operatori può causare un errore di compilazione non facile da riconoscere, ma può causare errori logici molto insidiosi.

2. Istruzione if

Il programma successivo usa sei istruzioni if per confrontare due numeri inseriti dall'utente. Se la condizione in una qualunque di queste istruzioni if è vera, l'istruzione printf a essa associata viene eseguita.

```
01  // 05_01.c
02  // Uso dell'istruzione if, degli operatori
03  // relazionali e degli operatori di uguaglianza.
04  #include <stdio.h>
05
06  // la funzione main inizia l'esecuzione del programma
07  int main( void )
08  {
09     printf( "Enter two integers, and I will tell you\n" );
10     printf( "the relationships they satisfy: " );
11
12     int num1; // primo numero inserito dall'utente
13     int num2; // secondo numero inserito dall'utente
14
15     scanf( "%d %d", &num1, &num2 ); // legge due interi
16
17     if ( num1 == num2 ) {
18        printf( "%d is equal to %d\n", num1, num2 );
19     } // fine di if
20
21     if ( num1 != num2 ) {
22        printf( "%d is not equal to %d\n", num1, num2 );
23     } // fine di if
24
25     if ( num1 < num2 ) {
26        printf( "%d is less than %d\n", num1, num2 );
27     } // fine di if
28
29     if ( num1 > num2 ) {
```

```
30          printf( "%d is greater than %d\n", num1, num2 );
31       } // fine di if
32
33       if ( num1 <= num2 ) {
34          printf( "%d is less than or equal to %d\n", num1, num2 );
35       } // fine di if
36
37       if ( num1 >= num2 ) {
38          printf( "%d is greater than or equal to %d\n", num1, num2 );
39       } // fine di if
40    } // fine della funzione main
```

```
Enter two integers, and I will tell you

the relationships they satisfy: 3 7

3 is not equal to 7

3 is less than 7

3 is less than or equal to 7
```

```
Enter two integers, and I will tell you

the relationships they satisfy: 22 12

22 is not equal to 12

22 is greater than 12

22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you

the relationships they satisfy: 7 7

7 is equal to 7

7 is less than or equal to 7

7 is greater than or equal to 7
```

Il programma usa scanf (riga 15) per leggere due interi nelle variabili int num1 e num2. Il primo %d converte il valore da memorizzare nella variabile num1 e il secondo %d converte il valore da memorizzare nella variabile num2.

😊 *Buona pratica di programmazione: Pur essendo permesso, è preferibile che non ci sia più di un'istruzione per riga in un programma.*

😕 *Errore comune di programmazione: Mettere virgole (quando non sono necessarie) tra specificatori di conversione nella stringa di controllo del formato di un'istruzione scanf.*

Confronto di numeri

L'istruzione if nelle righe 17–19

```
if ( num1 == num2 ) {
   printf( "%d is equal to %d\n", num1, num2 );
} // fine di if
```

confronta i valori delle variabili num1 e num2 per vedere se sono uguali.

Se i valori sono uguali, l'istruzione nella riga 18 stampa una riga di testo che afferma che i numeri sono uguali.

Se le condizioni sono vere in una o più delle istruzioni if nelle righe 21, 25, 29, 33 e 37, la corrispondente istruzione nel corpo dell'if stampa una riga di testo appropriata.

Indentare il corpo di ognuna delle istruzioni if e inserire righe vuote sopra e sotto di esse migliora la leggibilità del programma.

Una parentesi graffa sinistra, {, inizia il corpo di ognuna delle istruzioni if (es. la riga 17).

Una parentesi graffa destra corrispondente, }, termina il corpo dell'istruzione if (es. la riga 19).

Nel corpo di un'istruzione if si può inserire un numero qualsiasi di istruzioni.

☹ *Errore comune di programmazione: Mettere un punto e virgola immediatamente dopo la parentesi destra che segue la condizione in un'istruzione if.*

😊 *Buona pratica di programmazione: L'uso di parentesi graffe per delimitare il corpo di un'istruzione if è facoltativo quando il corpo contiene soltanto un'istruzione. È considerata una buona pratica usare sempre tali parentesi.*

😊 *Buona pratica di programmazione: Un'istruzione troppo lunga può essere distribuita su diverse righe. Se un'istruzione deve essere divisa fra più righe, scegliete punti di rottura che abbiano senso (come dopo una virgola in un elenco separato da virgole).*

Se un'istruzione è divisa fra due o più righe, indentate tutte le righe successive.

Non è corretto spezzare gli identificatori.

La tabella successiva elenca la precedenza degli operatori, dalla più alta alla più bassa. Gli operatori sono mostrati dall'alto in basso in ordine decrescente di precedenza.

Il segno di uguale è pure un operatore.

Tutti questi operatori, ad eccezione dell'operatore di assegnazione =, sono associativi da sinistra a destra. L'operatore di assegnazione (=) è associativo da destra a sinistra.

Operatori	Associatività
()	da sinistra a destra
* / %	da sinistra a destra
+ -	da sinistra a destra
< <= > >=	da sinistra a destra
== !=	da sinistra a destra
=	da destra a sinistra

Buona pratica di programmazione: Quando scrivete espressioni contenenti molti operatori, fate riferimento alla tabella di precedenza degli operatori.

Controllate che gli operatori nell'espressione siano applicati nell'ordine giusto.

Se non siete sicuri sull'ordine di calcolo in un'espressione complessa, usate le parentesi per raggruppare le espressioni o spezzate l'istruzione in diverse istruzioni più semplici.

Ricordatevi che alcuni operatori del C come l'operatore di assegnazione (=) sono associativi da destra a sinistra piuttosto che da sinistra a destra.

Parole chiave

Alcune delle parole che abbiamo usato nei programmi in C – in particolare int, if e void – sono parole chiave o parole riservate del linguaggio.

Queste parole hanno un significato speciale per il compilatore C, pertanto dovete fare attenzione a non usarle come identificatori, come per i nomi delle variabili.

Parole chiave:

auto break case char const continue

default do double else enum extern

float for goto if int long

register return short signed sizeof static

struct switch typedef union unsigned void

volatile while

Parole chiave aggiunte nello standard C99

_Bool _Complex _Imaginary inline restrict

Parole chiave aggiunte nello standard C11

_Alignas _Alignof _Atomic _Generic _Noreturn _Static_assert _Thread_local

3. Programmazione sicura in C

Il CERT (C Secure Coding Standard) definisce delle linee guida che aiutato a evitare pratiche di programmazione che espongono i sistemi ad attacchi.

Evitare printf con argomento singolo

Una tipica linea guida consiste nell'evitare l'uso di printf con una sola stringa come argomento. Se avete necessità di stampare una stringa che termina con un newline ("ritorno a capo"), usate la funzione puts, che stampa il suo argomento, una stringa, seguito da un carattere newline.

Ad esempio, l'istruzione

```
printf( "Welcome to C!\n" );
```

andrebbe scritta come:

```
puts( "Welcome to C!" );
```

Non abbiamo incluso \n nella stringa precedente perché puts lo aggiunge automaticamente.

Se avete l'esigenza di mostrare una stringa senza un carattere newline di terminazione, usate printf con due argomenti: una stringa di controllo per il formato "%s" e la stringa da stampare.

Lo specificatore di conversione %s serve a stampare una stringa.

Ad esempio, 'istruzione

```
printf( "Welcome " );
```

andrebbe scritta come:

```
printf( "%s", "Welcome " );
```

Sebbene le printf così come sono scritte fino ad ora non siano davvero insicure, queste modifiche sono buone pratiche di codifica che elimineranno certe vulnerabilità relative alla sicurezza. D'ora in poi tenderemo ad utilizzare queste pratiche.

Bibliografia

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

- CERT, www.securecoding.cert.org/