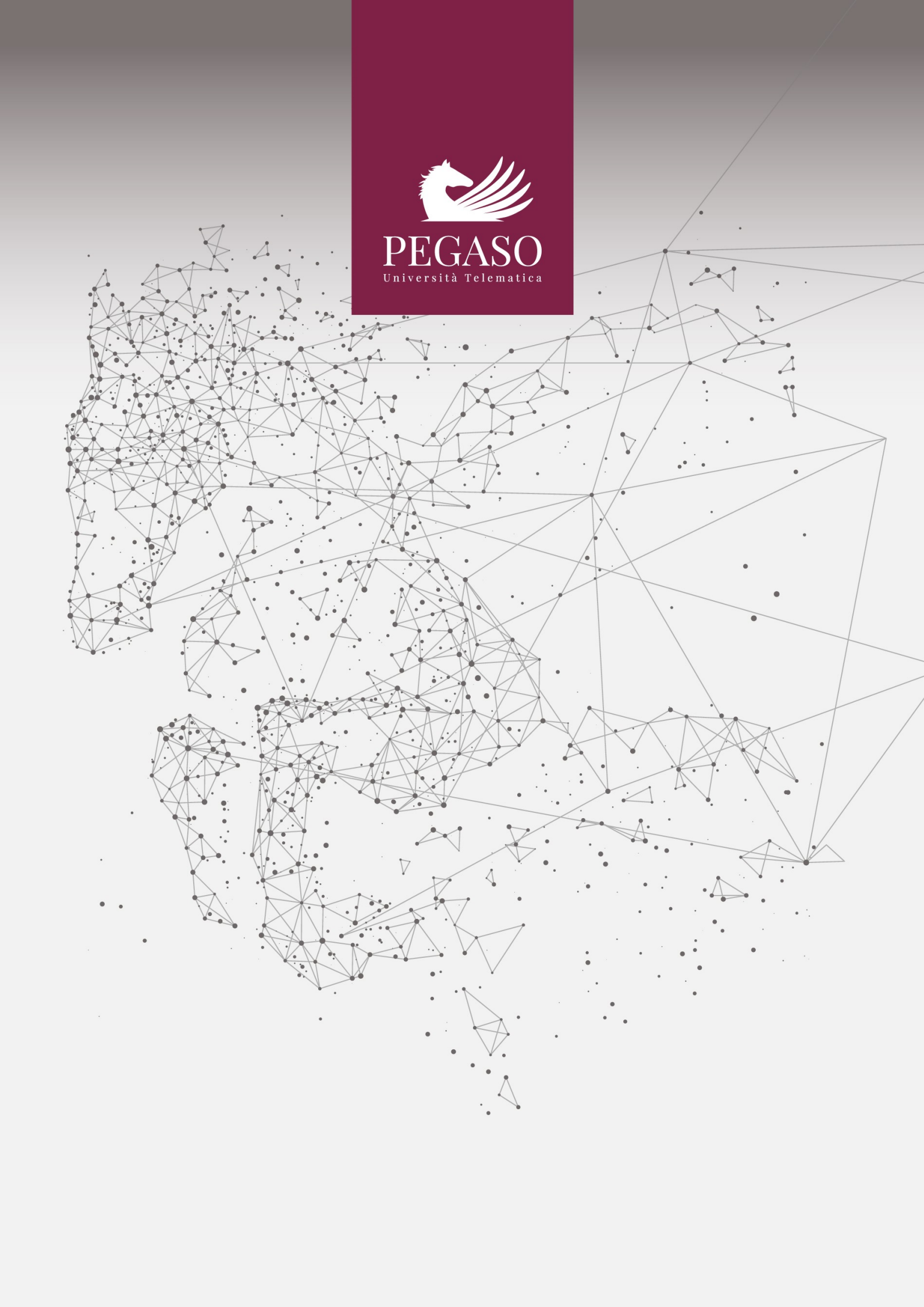

PEGASO
Università Telematica

Indice

1. Istruzioni do...while

L'istruzione di iterazione `do...while` è simile all'istruzione `while`.

Nell'istruzione `while` la condizione di continuazione del ciclo è verificata all'inizio del ciclo prima dell'esecuzione del corpo del ciclo.

L'istruzione `do...while` verifica la condizione di continuazione del ciclo dopo l'esecuzione del corpo del ciclo.

Pertanto, il corpo del ciclo sarà sempre eseguito almeno una volta.

Quando un `do...while` termina, l'esecuzione prosegue con l'istruzione dopo la clausola `while`.

L'istruzione do...while è scritta come segue:

```
do {
   istruzioni
} while (condizione); //qui e' richiesto il punto e virgola
```

Il seguente programma usa l'istruzione `do...while` per stampare i numeri da 1 a 10.

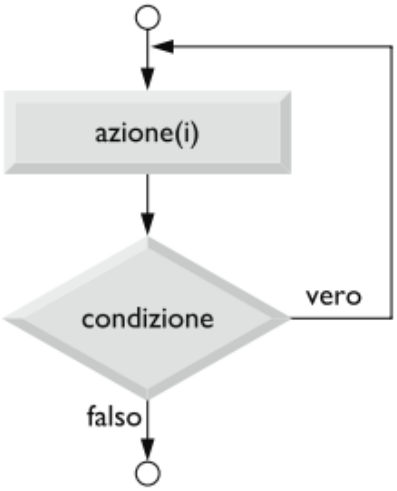
```
1  // Programma C
2  // Uso dell'istruzione di iterazione do...while.
3  #include <stdio.h>
4
5  int main(void)
6  {
7     unsigned int counter = 1; // inizializza il contatore
8
9     do {
10       printf("%u  ", counter);
11    } while (++counter <= 10);
12  }
```

```
1 2 3 4 5 6 7 8 9 10
```

Abbiamo deciso di preincrementare la variabile di controllo counter nel test di continuazione del ciclo (riga 11).

Diagramma di flusso dell'istruzione `do...while`

Il diagramma di flusso dell'istruzione `do...while` mostra che la condizione di continuazione del ciclo non viene testata finché l'azione non è eseguita almeno una volta.



2. Istruzione break

L'istruzione `break` si usa per alterare il flusso di controllo.

Abbiamo già visto come utilizzare l'istruzione `break` per terminare l'esecuzione di un'istruzione `switch`.

Questa volta esaminiamo come usare `break` in un'istruzione di iterazione.

L'istruzione `break`, quando è eseguita in un'istruzione `while`, `for`, `do...while` o `switch`, provoca un'uscita immediata da quell'istruzione.

L'esecuzione del programma continua con l'istruzione successiva dopo quel `while`, `for`, `do...while` o `switch`.

L'uso di `break` serve normalmente per uscire subito da un ciclo o per saltare il resto di un'istruzione `switch`.

Il seguente programma mostra l'istruzione `break` (riga 14) in un'istruzione di iterazione `for`.

Quando l'istruzione `if` scopre che x vale 5, viene eseguito il `break`.

Questo termina l'istruzione `for` e il programma continua con il `printf` dopo il `for`.

Il ciclo viene eseguito completamente solo quattro volte.

Abbiamo dichiarato x prima del ciclo in questo esempio, cosicché potremmo usare il suo valore finale dopo la fine del ciclo.

Ricordatevi che quando dichiarate la variabile di controllo in un'espressione di inizializzazione di un ciclo for, la variabile non esiste più dopo la fine del ciclo.

```
1  // Programma C
2  // Uso dell'istruzione break in un'istruzione for.
3  #include <stdio.h>
4
5  int main(void)
```

```
6  {
7     unsigned int x; // dichiarato per utilizzo dopo il ciclo
8
9     // ripeti 10 volte
10    for (x = 1; x <= 10; ++x) {
11
12       // se x e' 5, termina il ciclo
13       if (x == 5) {
14          break; // interrompi il ciclo solo se x e' 5
15       }
16
17       printf("%u ", x);
18    }
19
20    printf("\nBroke out of loop at x == %u\n", x);
21 }
```

```
1 2 3 4

Broke out of loop at x == 5
```

3. Istruzione continue

Come l'istruzione `break`, anche l'istruzione `continue` si usa per alterare il flusso di controllo.

L'istruzione `continue`, quando è eseguita in un'istruzione `while`, `for` o `do...while`, salta le istruzioni rimanenti nel corpo di quell'istruzione di controllo e fa eseguire la successiva iterazione del ciclo.

Nelle istruzioni `while` e `do...while` il test di continuazione del ciclo è valutato immediatamente dopo l'esecuzione dell'istruzione continue.

Nell'istruzione for viene eseguita l'espressione di incremento, quindi viene valutato il test di continuazione del ciclo.

Il seguente programma usa `continue` (riga 12) nell'istruzione `for` per saltare l'istruzione `printf` e iniziare l'iterazione successiva del ciclo.

```
01  // Programma C
02  // Uso dell'istruzione continue in un'istruzione for.
03  #include <stdio.h>
04
05  int main(void)
06  {
07     // ripeti 10 volte
08     for (unsigned int x = 1; x <= 10; ++x) {
09
10        // se x e' 5, continua con la successiva iterazione
11        if (x == 5) {
12           continue; // salta il restante codice del ciclo
13        }
14
15        printf("%u ", x );
16     }
17
18     puts("\nUsed continue to skip printing the value 5");
19  }
```

```
1 2 3 4 6 7 8 9 10

Used continue to skip printing the value 5
```

Osservazione di ingegneria del software

Alcuni programmatori pensano che `break` e `continue` violino le norme di programmazione strutturata.

Questi programmatori, pertanto, non usano `break` e `continue` nelle iterazioni.

Tuttavia, le istruzioni break e continue, quando usate correttamente, sono eseguite più velocemente delle corrispondenti tecniche strutturate che presto apprenderemo.

Vi è un contrasto tra l'obiettivo di realizzare un'ingegneria del software di qualità e quello di realizzare un software con le migliori prestazioni.

Spesso uno di questi obiettivi viene raggiunto a spese dell'altro.

In tutte le situazioni che non richiedono prestazioni estremamente spinte, osservate le seguenti linee guida: primo, rendete il vostro codice semplice e corretto; poi rendetelo più veloce e compatto, ma solo se necessario.

Il linguaggio C risulta essere particolarmente apprezzato, rispetto ad altri linguaggi quali Java o Python, proprio dove sono necessarie prestazioni spinte

Riferimenti bibliografici

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