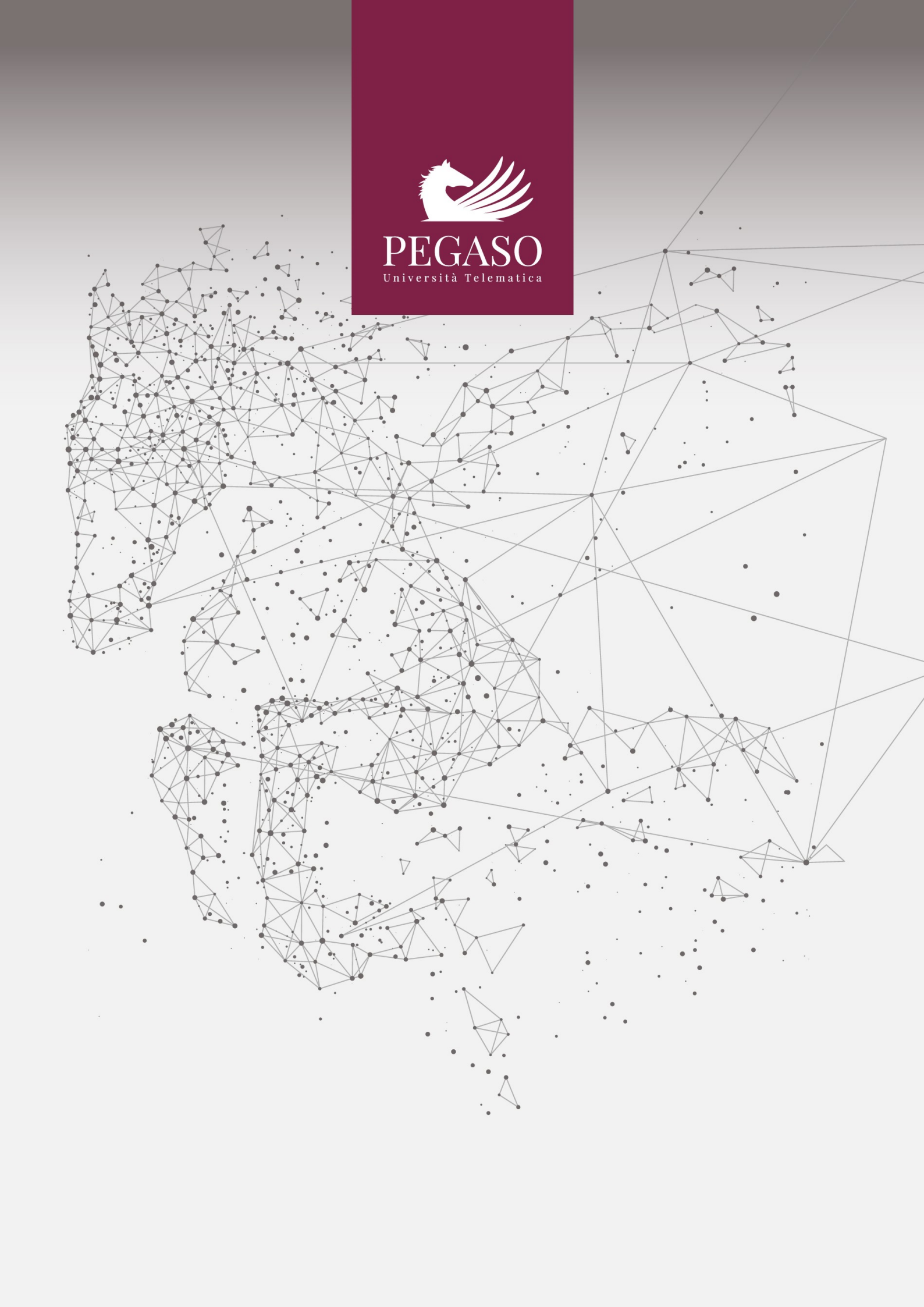
PEGASO
Università Telematica

Indice

1. Istruzioni di controllo annidate

Formuleremo ancora una volta l'algoritmo usando lo pseudocodice e l'affinamento graduale top-down, e scriveremo un corrispondente programma in C.

Abbiamo visto che le istruzioni di controllo possono essere accatastate una dopo l'altra (in sequenza) proprio come fa un bambino con i blocchi di costruzioni.

In questo caso pratico vedremo l'unico altro modo strutturato in cui le istruzioni di controllo possono essere connesse tra loro in C, ossia attraverso l'annidamento di un'istruzione di controllo all'interno di un'altra.

Considerate il seguente problema:

Un college vuole sapere come i 10 studenti di un corso si siano classificati all'esame.

Vi viene dato un elenco di questi 10 studenti. Accanto a ogni nome viene scritto un 1 se lo studente ha superato l'esame o un 2 se lo ha fallito.

Il vostro programma deve analizzare i risultati dell'esame come segue:

1. Leggere ogni risultato dell'esame (cioè un 1 o un 2). Stampare il messaggio di prompt "Enter result" ogni volta che il programma richiede un altro risultato dell'esame.
2. Contare il numero dei risultati dell'esame di ogni tipo.
3. Mostrare un riepilogo dei risultati dell'esame indicando il numero degli studenti
4. che lo hanno superato e il numero di quelli che sono stati respinti.
5. Se più di otto studenti hanno superato l'esame, stampare il messaggio "Bonus to
6. instructor!"

Dopo aver letto attentamente il testo del problema, facciamo le seguenti osservazioni:

1. Il programma deve elaborare 10 risultati di un esame. Sarà usato un ciclo controllato da contatore.
2. Ogni risultato dell'esame è un numero, un 1 o un 2. Ogni volta che legge un risultato, il programma deve stabilire se il numero è un 1 o un 2. Nel nostro algoritmo controlliamo se si tratta di un 1. Se il numero non è un 1, presumiamo che sia un 2. (Il testo garantisce che ogni risultato dell'esame sia un 1 o un 2.)
3. Sono usati due contatori, uno per contare il numero di studenti che hanno superato l'esame e uno per contare il numero di studenti che lo hanno fallito.
4. Dopo che il programma ha elaborato tutti i risultati, deve verificare se più di otto studenti hanno superato l'esame.

Procediamo con l'affinamento graduale top-down.

Iniziamo con una rappresentazione al livello top in pseudocodice:

Analizza i risultati dell'esame e decidi se l'istruttore deve ricevere un bonus

Ancora una volta è importante mettere in rilievo che il livello top è una rappresentazione completa del programma, ma è probabile che siano necessari diversi affinamenti prima che lo pseudocodice possa essere naturalmente tradotto in un programma in C.

Il nostro primo affinamento è:

Inizializza le variabili

Leggi i dieci voti dell'esame e conta le promozioni e le bocciature

Stampa un riepilogo dei risultati e decidi se l'istruttore deve ricevere un bonus

Anche qui, pur avendo una rappresentazione completa dell'intero programma, è necessario un ulteriore affinamento.

Ora individuiamo le variabili specifiche.

Sono necessari dei contatori per registrare le promozioni e le bocciature

Inoltre, un contatore sarà usato per controllare il processo di iterazione, ed è anche necessaria una variabile per memorizzare l'input dell'utente.

L'istruzione in pseudocodice

Inizializza le variabili

si può affinare come segue:

Inizializza il contatore delle promozioni a zero

Inizializza il contatore delle bocciature a zero

Inizializza il contatore del numero di studenti a uno

Si noti che vengono inizializzati solo il contatore e i totali.

L'istruzione in pseudocodice

Leggi i dieci voti dell'esame e conta le promozioni e le bocciature

richiede un ciclo che in successione legga ogni risultato dell'esame.

Qui si sa in anticipo che vi sono precisamente dieci risultati dell'esame, così l'iterazione controllata da contatore è quella appropriata.

Dentro il ciclo (ossia annidata all'interno del ciclo) un'istruzione di selezione doppia determinerà se il risultato di ogni esame è una promozione o una bocciatura e incrementerà di conseguenza i contatori corrispondenti.

L'affinamento della precedente istruzione in pseudocodice è allora

Finché il contatore del numero di studenti è minore o uguale a dieci

Leggi il risultato successivo dell'esame

Se lo studente ha superato l'esame

Aggiungi uno al contatore delle promozioni

altrimenti

Aggiungi uno al contatore delle bocciature

Aggiungi uno al contatore del numero di studenti

Si noti l'uso di righe vuote per evidenziare l'istruzione If...else (Se…altrimenti) al fine di migliorare la leggibilità del programma.

L'istruzione in pseudocodice

Stampa un riepilogo dei risultati e decidi se l'istruttore deve ricevere un bonus

può essere affinata come segue:

Stampa il numero di promozioni

Stampa il numero di bocciature

Se più di otto studenti sono stati promossi

Stampa "Bonus to instructor!"

Si ottiene pertanto il seguente pseudocodice completo:

1 Inizializza il contatore delle promozioni a zero

2 Inizializza il contatore delle bocciature a zero

3 Inizializza il contatore del numero di studenti a uno

```
4
5  Finché il contatore del numero di studenti è minore o uguale a dieci
6     Leggi il risultato successivo dell'esame
7
8     Se lo studente ha superato l'esame
9        Aggiungi uno al contatore delle promozioni
10    altrimenti
11       Aggiungi uno al contatore delle bocciature
12
13    Aggiungi uno al contatore del numero di studenti
14
15 Stampa il numero di promozioni
16 Stampa il numero di bocciature
17 Se più di otto studenti sono stati promossi
18    Stampa "Bonus to instructor!"
```

Questo pseudocodice è ora sufficientemente affinato per la conversione in C.

```
01  // 09_02.c
02  // Analisi dei risultati dell'esame.
03  #include <stdio.h>
04
05  // la funzione main inizia l'esecuzione del programma
06  int main( void )
07  {
08     // inizializza le variabili nelle definizioni
09     unsigned int passes = 0; // numero di promozioni
10     unsigned int failures = 0; // numero di bocciature
```

```
11    unsigned int student = 1; // contatore del numero di studenti
12    int result; // un risultato dell'esame
13
14    // processa 10 studenti usando un ciclo controllato da contatore
15    while ( student <= 10 ) {
16
17       // richiedi all'utente un valore in ingresso
18       printf( "%s", "Enter result ( 1=pass,2=fail ): " );
19       scanf( "%d", &result );
20
21       //  se il risultato e' 1, incrementa il numero di promozioni
22       if ( result == 1 ) {
23           passes = passes + 1;
24       } // fine di if
25       else { // altrimenti, incrementa il numero di bocciature
26          failures = failures + 1;
27       } // fine di else
28
29       student = student + 1; // incrementa il contatore
30    } // fine di while
31
32    //  fase di terminazione; stampa i risultati
33    printf( "Passed %u\n", passes );
34    printf( "Failed %u\n", failures );
35
36    // decidi se stampare "Bonus to instructor!"
37    if  ( passes > 8 {
38        puts( "Bonus to instructor!" );
39    } // fine di if
```

```
40    } // fine della funzione main
```

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4
```

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
```

```
Passed 9

Failed 1

Bonus to instructor!
```

Abbiamo tratto vantaggio da una caratteristica del C che permette che l'inizializzazione venga incorporata nelle definizioni (righe 9–11).

Tale inizializzazione avviene al momento della compilazione.

Notate anche che quando si invia in uscita un **unsigned int** si usa lo specificatore di conversione **%u** (righe 33–34).

Osservazione di ingegneria del software

Molti programmatori scrivono programmi senza mai usare strumenti di sviluppo dei programmi come lo pseudocodice. Essi pensano che il loro obiettivo finale sia quello di risolvere il problema su un computer e che scrivere lo pseudocodice ritardi semplicemente la realizzazione del prodotto finale.

L'esperienza dimostra che la parte più difficile del risolvere un problema su un computer è quella che riguarda lo sviluppo di algoritmi per la soluzione.

Una volta che è stato specificato un algoritmo corretto, il processo di produzione di un programma funzionante in C è di norma semplice.

Bibliografia

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