Manuale programmazione :: “Leggermente”

# Indice

[Indice 1](#_Toc378875123)

[Introduzione al linguaggio 2](#_Toc378875124)

[Scelte dei programmatori 2](#_Toc378875125)

[Sintassi del linguaggio 3](#_Toc378875126)

[I File 3](#_Toc378875127)

[Commenti 3](#_Toc378875128)

[Le funzioni 4](#_Toc378875129)

[I pacchetti 4](#_Toc378875130)

[Le varibili e vettori 5](#_Toc378875131)

[Comparatori e simboli 5](#_Toc378875132)

[Strutture di controllo 6](#_Toc378875133)

[Strutture condizionali 6](#_Toc378875134)

[Strutture d’iterazione 7](#_Toc378875135)

[Indice delle strutture d’iterazione 7](#_Toc378875136)

[Esempi 8](#_Toc378875137)

[Buble Sort 8](#_Toc378875138)

# Introduzione al linguaggio

Il linguaggio “LeggerMente” è nato dalla necessità di avvicinare le classi del biennio a un linguaggio di programmazione che fosse simile a quelli normalmente usati nell’ambito informatico, ma che allo stesso tempo non richiedesse ai ragazzi uno sforzo eccessivo per iniziare.  
Per questo motivo abbiamo iniziato lo studio di un linguaggio di programmazione che fosse in primo luogo “intuitivo” per i ragazzi e che permettesse anche una rapida comprensione e un veloce apprendimento.  
Il linguaggio LeggerMente è a tutti gli effetti un linguaggio tradotto, la sua sintassi viene convertita in Runtime in C#, infine lo stesso traduttore del linguaggio richiama a se il compilatore “CSharCompiler.exe” che produrrà l’eseguibile.

## Scelte dei programmatori

Dovendo semplificare un linguaggio, abbiamo dovuto prendere alcune decisioni, anche in relazione alle numerose interviste effettuate ai ragazzi del biennio informatico:

* Detipizzazione delle variabili, preso da numerosi linguaggi di Scripting quali il Java script, PHP e altri specializzati nell’ambiente web;
* Localizzazione del linguaggio, benché la maggior parte dei linguaggi di programmazione usi un dizionario di parole chiave inglese, abbiamo preferito contemplare la possibilità di usare un dizionario localizzato per semplificare la lettura del codice;
* Scelta .NET Framework, la scelta del linguaggio C# è dovuta alla nostra conoscenza di esso ampliatasi durante il periodo scolastico e per la possibilità

di utilizzare la programmazione ad oggetti;

* Riduzione simbolismi, per facilitare la comprensione sono stati rimosse alcuni simboli, comunemente usati nei linguaggi di programmazione, a favore di parole chiave intuitive ed equivalenti;
* Incapsulamento funzioni, oltre a permettere la scrittura delle funzioni senza un ordine fisso, abbiamo deciso di lasciare la possibilità di raggruppare le funzioni in pacchetti, esportabili singolarmente, per permettere il loro utilizzo;
* Continuità progettuale, per evitare la dispersione del progetto abbiamo contattato alcuni ragazzi frequentanti le classi quarte, che si sono dimostrati interessati al progetto e a una sua futura continuazione.

# Sintassi del linguaggio

Il linguaggio “LeggerMente” si ispira, per la sintassi, a differenti linguaggi di programmazione che ultimamente si stanno diffondendo anche nel mondo della programmazione non professionale, come il Lua e il Python.

La sintassi sfrutta la tabulazione per suddividere i comandi in regioni facilmente riconoscibili.

Inoltre abbiamo deciso di eliminare la possibilità di concatenare più comandi per permettere una più facile comprensione.

Le funzioni vengono indicate con il simbolo “#” seguite dal proprio nome, il codice delle funzioni deve essere scritto tra parentesi quadre per permetterne l’identificazione.

Qualunque funzione restituisce un valore non tipizzato e i parametri vengono inseriti senza specificazione di tipo.

I parametri dovranno essere indicati tra parentesi tonde con il loro nome e nell’utilizzo della funzione andranno inseriti nell’esatta sequenza.

La funzione principale, comunemente denominata “Main” qui verrà nominata come:

**#programma**[

…

]

## Commenti

In “LeggerMente” come tutti i linguaggi di programmazione implementa la possibilità di scrivere dei commenti direttamente nel codice.  
Viene considerato commento tutto ciò che è presente fra “|”

**#Funzione**[

**se** 3 **maggiore** 1

**ritorna** vero *|questo è un commento|*

**altrimenti**

**ritorna** falso *|anche questo|*

]

## Le funzioni

Di seguito presentiamo un esempio di funzione:

**#NumeroPari**(**numero**)[

**se** **MATE**.**Resto** **di** numero,2 **uguale** 0

**ritorna** vero

**altrimenti**

**ritorna** falso

]

Come si può osservare, la funzione viene dichiarata con il simbolo ‘#’, i parametri (all’interno delle parentesi tonde) sono separati da una virgola.

Infine si può specificare il valore di ritorno della funzione.

Le Procedure presentano una sintassi perfettamente uguale alle funzioni, senza però ritornare alcun valore.

Esempio:

**#Esempio**(**numero**)[

**se** **MATE**.**Resto** **di** numero,2 **uguale** 0

...

**altrimenti**

...

]

**N.B.** Se in una funzione o in un ramo di una struttura di controllo non viene specificato un valore di ritorno, viene restituito NULL.

**I parametri delle funzioni e procedure sono sempre passati per referenza e non per valore**.

## I pacchetti

Il pacchetto è un insieme di funzioni e/o procedure che viene definito su un file esterno e che può essere incluso in qualunque altro progetto.

Nella dichiarazione del pacchetto è necessario dichiarate il tag “pacchetto” e al suo interno non deve essere presente il tag “#programma”.

La dichiarazione avviene come nell’esempio:

**pacchetto** MATE

**#Pari**(**numero**)[

…

]

**#Media**(**vettore**)[

…

]

Il traduttore genererà automaticamente un file ”\*.plm” che potrà essere importato in un altro progetto, tramite il tag “aggiungi” come nell’esempio:

**Aggiungi** **MATE**

**#programma**[

…

]

## I File

In “LeggerMente” esistono due tipi di file:

* “*\*.lm*” , sono i sorgenti del nostro progetto, essi sono identificabili per la presenza del tag “#*programma*”, produrranno i file eseguibili.
* “*\*.plm*”, sono delle raccolte di funzioni a cui viene assegnato un nome, essi si possono riconoscere per il tag “pacchetto” al loro interno e per la mancanza del tag “#programma”.

## Le varibili e vettori

Nel linguaggio le variabili non sono state tipizzate, esse possono contenere un qualsiasi valore.

Sarà possibile sapere il tipo di dato contenuto mediante i metodi della libreria “VAR” come riportato nelle referenze in fondo al manuale.

I comandi standard nella gestione delle variabili sono:

**crea** pippo → Crea una variabile

**crea** pluto **a** 0 → Crea una variabile con un valore indicato

**cambia** pippo **a** 1 → Cambia la variabile al valore indicato

**scambia** pluto **con** pippo → Scambia due variabili fra di loro

Anche se non utilizzate all’atto pratico, il linguaggio ragiona su quattro tipi di variabili:

* Double, classico numero con la virgola
* String, stringa di testo
* Bool, variabile vera o falsa
* Vector, insieme di valori vettoriali.

Il vettore non è infatti altro che una lista di variabili accessibili tramite indice singolo.

Ogni variabile a sua volta può diventare un vettore, di conseguenza è possibile creare una matrice con gli appositi comandi:

**crea** **vettore** vet **di** 5 → Crea un vettore di 5

**cambia** vet(1) **a “gatto”** → Cambia il primo elemento del vettore a “gatto”

**crea** **vettore** vet(2) **a** 3 → Crea un vettore di 3 nel terzo elemento

**cambia** vet(2)(1) **a** 5 → Cambia il primo elemento del secondo a 5

## Comparatori e simboli

Oltre ai classici simboli quali:

+ → Somma

- → Sottrazione

/ → Divisione

\* → Moltiplicazione

“” → Definizione stringa

, → Concatenazione stringa

Abbiamo quelli in parole chiave, quali:

**maggiore** → Comparatore di maggioranza

**minore** → Comparatore di minoranza

**uguale** → Comparatore di eguaglianza

**diverso** → Comparatore di diseguaglianza

**non** → Negazione di un Booleano

# Strutture di controllo

Come in tutti i linguaggi, anche il LeggerMente sfrutta delle strutture di controllo per condizionare il codice.

## Strutture condizionali

Nel “LeggerMente” è possibile concatenare, nelle strutture condizionali, una sequenza di condizioni tramite l’utilizzo delle parentesi tonde e la concatenazione booleana fornita dai blocchi “**e**” ed “**o**” per ognuno dei seguenti casi:

* Se, condizionale semplice, permette di eseguire un determinato blocco di azioni se la condizione primaria indicata viene rispettata:

**se** (numero **maggiore** 0) **e** (x **diverso** -1)

**cambia** x **a** numero

* Se-altrimenti, condizionale composto, permette l’esecuzione di un blocco “se” con un annesso blocco nel caso la condizione primaria risulti falsa:

**se** numero **maggiore** 0

**cambia** x a 0

**altrimenti**

**cambia** x a 1

* Controlla, condizionale multiplo, in questo elemento è possibile confrontare una variabile con una serie di condizioni e eseguire diversi blocchi di codice in relazione alla veridicità di essi:

**controlla** numero

**se** **minore** 0

**cambia** x a 1

**se** **uguale** 0

**cambia** x a 2

**se** **maggiore** 0

**cambia** x a 3

Come visto negli esempi e precedentemente ognuna delle strutture contiene le proprie sequenze di comandi tramite la tabulazione.

## Strutture d’iterazione

Le strutture d’iterazione sono delle strutture nel linguaggio che permetto di ripetere un determinato blocco di comando in una specifica situazione.

Queste strutture sono:

* Ripeti volte, struttura che permette di ripetere un blocco di codice un determinato numero di volte senza una condizioni che ne determini l’interruzione o il procedimento:

**ripeti** 5 **volte**

**schermo** “LeggerMente\n”

* Quando ripeti, strutture che permette la ripetizione di un blocco di codice sotto una determinata condizione:

**ripeti quando** x **maggiore** 0

**schermo** “Minore”

* Ripeti quando, struttura che permette l’esecuzione di un blocco di codice e solo dopo il controllo di una condizione che ne determini la continuazione

**ripeti**

**schermo** “controllo minimo”

**quando** x **minore** 0

## Indice delle strutture d’iterazione

Nel “LeggerMente” ogni struttura d’iterazione durante l’esecuzione del suo blocco d’istruzioni, genera una variabile normalmente denominata “**Indice**” contenente il numero di esecuzioni che si sono svolte su quel blocco.  
E possibile ridefinire il nome di questa variabile mediante il tag “con”:

**ripeti** x **volte con** i

**cambia** vet(i) **a** vet(i)+5

# Esempi

Qui riportiamo alcuni esempi di codice scritto in “LeggerMente”.

## Buble Sort

|  |
| --- |
| **aggiungi** MATE    **#programma[**  **crea** n **a** 5  **crea vettore** v **a** n  **crea** sup  **crea** check    **ripeti** 5 **volte**  **cambia** v**(**indice**) a MATE.random di** 0**,**100    **ripeti**  **cambia** check **a falso**  **ripeti se minore di** n**-**1  **se** v**(**indice**) minore** v**(**indice**+**1**)**  **scambia** v(indice) **a** v(indice+1)  **cambia** check **a vero**  **cambia** n **a** n**-**1  **quando** check    **ripeti** 5 **volte con** i  **schermo** v**(**i**),”\n”**  **]** |