TRIBELON
RIVISTA DI DISEGNO
UNIVERSITÀ DEGLI
STUDI DI FIRENZE

VOL. 1 | N. 1 | 2024
DISEGNO FRA TRADIZIONE E INNOVAZIONE
DRAWING BETWEEN TRADITION AND INNOVATION

Citation: G. Anzani, *Introduzione al linguaggio di programmazione AutoLISP*, in *Codici grafici*, *TRIBELON*, I, 2024, 1, pp. 116-122.
ISSN (stampa): 3035-143X
ISSN (online): 3035-1421
doi: https://doi.org/10.36253/tribelon-2865
Published: July, 2024

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.
Competing Interests: The Author(s) declare(s) no conflict of interest.
Journal Website: *riviste.fupress.net/tribelon*

INTRODUZIONE AL LINGUAGGIO DI PROGRAMMAZIONE AUTOLISP

GIOVANNI ANZANI
University of Florence
giovanni.anzani@unifi.it

Apriamo questo primo numero della rubrica codici grafici, dedicata ai linguaggi di programmazione per i programmi di grafica, con una prima puntata di introduzione al linguaggio AutoLISP associato ad AutoCAD, vedremo anche una semplice routine sviluppata sugli esempi di codice introdotti; seguiranno una seconda ed una terza puntata con livelli di approfondimento maggiori e con routine AutoLisp maggiormente complesse. La rubrica codici grafici tratterà in seguito anche altri linguaggi di programmazione dei quali verranno dati dapprima rudimenti introduttivi.
Creare routine personalizzate in AutoLISP consente di automatizzare ed estendere le potenzialità di AutoCAD secondo le proprie necessità. AutoLISP è un dialetto semplificato per AutoCAD del linguaggio LISP ideato nel 1958 da John McCarthy come linguaggio formale per studiare la computabilità di funzioni ricorsive su espressioni simboliche e consente uno stile di programmazione funzionale; il suo nome deriva dalla sua organizzazione in espressioni costituite da liste dinamiche di elementi racchiuse tra parentesi che vengono poi processate in fase di esecuzione del programma (LISt Processing); la diffusione più rilevante del linguaggio LISP è avvenuta realizzandone dialetti integrati in programmi di uso comune. In AutoCAD LISP viene introdotto nel 1986 con il nome di AutoLISP e diventando un linguaggio di programmazione molto versatile, altamente interoperabile con il programma: ha le caratteristiche base dell'originale LISP, unite ad una nutrita serie di funzioni specifiche dedicate al trattamento dei dati geometrici e delle entità grafiche di AutoCAD; nel 1997 Autodesk incorpora funzionalità aggiuntive e ne modifica il nome in Visual Lisp.

Gli elementi

Il codice AutoLISP sviluppato può essere salvato nel formato **.lsp**, per essere eseguito immediatamente da AutoCAD essendo un linguaggio interpretato; in alternativa può essere compilato da AutoCAD e salvato nel formato compilato **.fas** eseguibile anch'esso in autocad. In LISP come in AutoLISP, gli elementi racchiusi tra parentesi possono essere uno o più dei seguenti:

- **Funzioni:** di programmazione, matematica, di manipolazione di elenchi, ecc;
- **Valori:** numeri interi o reali, coordinate, stringhe di testo, comandi, variabili, ecc;
- **Elenchi:** raggruppamenti omogenei od eterogenei di valori.

Le espressioni

In AutoLISP l'espressione è il costrutto di base ed ha la seguente struttura:

- Inizia con una parentesi aperta "(";
- È costituita da un nome di funzione e da argomenti facoltativi per tale funzione. Ogni argomento può anche essere un'espressione nidificata; gli argomenti della funzione saranno elaborati in fase di esecuzione dalla funzione;
- Termina con una parentesi chiusa ")";
- Restituisce un valore che può essere utilizzato da un'espressione chiamante in cui essa è eventualmente nidificata; se contiene espressioni nidificate restituisce il valore dell'ultima di tali espressioni.

Creare routine personalizzate in AutoLISP consente di automatizzare ed estendere le potenzialità di AutoCAD secondo le proprie necessità.

Negli esempi che seguono viene mostrata la sintassi di generiche espressioni AutoLISP:

- (funz_1 (argom_1 argom_N ...))
- (fu_1 (fu_2 ar_1) (fu_3 ar_2))

Nel secondo esempio, la funzione fu_1 ha come argomenti due espressioni; i valori restituiti da tali espressioni vengono utilizzati da fu_1. Le altre funzioni fu_2 e fu_3 hanno ciascuna un argomento ar_1 ar_2. AutoLISP valuta prima le espressioni più interne fu_2 e fu_3 e procede verso l'esterno valutando fu_1.

Le funzioni

Una funzione è un programma già presente in AutoLISP (di sistema) o definito dal programmatore (utente) che può essere utilizzato indicandone il nome ed i suoi eventuali argomenti e può restituire un valore al termine dellaua esecuzione; in AutoLISP qualsiasi operazione che l'utente compie avviene processando espressioni che contengono funzioni ed eventuali argomenti. Negli esempi che seguono le espressioni contenenti la funzione per effettuare una somma e tre espressioni con funzioni alternative per creare una lista di valori; a destra il valore restituito dalla funzione:

- **Somma**: (+ 1 2 3 4) → 10
- **Lista**: (LIST 1 2 3 4) → (1 2 3 4)
- **Lista**: (quote (1 2 3 4)) → (1 2 3 4)
- **Lista**: ' (1 2 3 4) → (1 2 3 4)

In AutoLISP esistono oltre trecento funzioni di sistema, raggruppabili in cinque **categorie**: base; utilità; selezione e gestione entità AutoCAD; finestre di dialogo DCL; estensioni Visual LISP, ActiveX ed Express Tools; queste sono a loro volta suddivise in un totale di ventitré **sottocategorie**; ad esempio nella categoria **base** la sottocategoria **funzioni aritmetiche** contiene 26 funzioni di sistema.

Le variabili

Sono delle allocazioni di memoria a cui viene assegnato un nome identificativo in cui memorizzare dei dati in fase di esecuzione del programma, nel caso di AutoLISP è fondamentale distinguere tra **Variabili Utente** e **Variabili di Sistema**, le prime sono definite e quindi scritte in fase di stesura di un programma AutoLISP dal programmatore principalmente tramite la funzione **SETQ**[1] e possono essere lette semplicemente indicandone il nome; le seconde sono già presenti in memoria dall'apertura di Autocad e vengono utilizzate per settare alcuni parametri dell'interfaccia di AutoCAD, di alcuni comandi di AutoCAD o per contenere informazioni di utilizzo generale, possono essere lette con la funzione **GETVAR**[2] e, se modificabili, possono essere scritte con la funzione **SETVAR**[3].
In base al dato che viene immagazzinato in esse, le variabili possono essere distinte in quattro tipologie:

- **Numeri Reali**: (SETQ RA 13.00)
- **Numeri Interi**: (SETQ IA 13)
- **Stringhe di Testo**: (SETQ TA "casa")
- **Liste**: (SETQ LA ' ("si" 12 14.5))

Con SETQ si ottiene il duplice effetto, di creare la variabile e di assegnarle un valore; è anche possibile assegnare ad una variabile il valore contenuto in una variabile preesistente, sostanzialmente leggendo il valore della variabile sorgente in fase di scrittura della variabile destinazione:

1 **SETQ** – (setq sym expr [sym expr]...) – Imposta il valore di uno o più simboli sulle espressioni associate. Questa è la funzione di assegnazione di base in AutoLISP. La funzione setq può assegnare più simboli in una stessa chiamata alla funzione.
Sym – Tipo: simbolo – La variabile definita dall'utente a cui assegnare expr. Questo argomento non viene valutato.
Expr: Tipo: intero, reale, stringa, elenco, file, nome (nome entità), T o nil – Un'espressione.
Valori restituiti – Tipo: intero, reale, stringa, elenco, file, nome (nome entità), T o nil – Il risultato dell'ultima espressione valutata.

2 **GETVAR** – (getvar varname) – Recupera il valore di una variabile di sistema AutoCAD.
Varname – Tipo: Stringa di testo – Nome di una variabile di sistema. Consultare il sistema della Guida del prodotto per un elenco delle variabili di sistema AutoCAD correnti.
Valori restituiti – Tipo: intero, reale, stringa, elenco o zero – Il valore della variabile di sistema; Altrimenti zero, se varname non è una variabile di sistema valida.

3 **SETVAR** – (setvar varname value) – Imposta una variabile di sistema AutoCAD su un valore specificato. Alcuni comandi di AutoCAD leggono i valori di alcune variabili di sistema prima di emettere un qualsiasi prompt; usando setvar per impostare un nuovo valore mentre uno di tali comandi è in corso, il nuovo valore impostato potrebbe non avere effetto sul comando AutoCAD in corso. Quando si utilizza la funzione setvar per modificare le variabili di sistema ANGBASE e SNAPANG di AutoCAD, l'argomento valore viene interpretato come radianti.
Varname – Tipo: Stringa di testo – Nome della variabile di sistema. Consultare il sistema della Guida del prodotto per un elenco delle variabili di sistema AutoCAD correnti.
Value – Tipo: intero, reale, stringa, elenco, T o zero – Un atomo o un'espressione il cui risultato valutato deve essere assegnato a varname. Per le variabili di sistema con valori interi, il valore fornito deve essere compreso tra -32.768 e +32.767.
Valori restituiti – Tipo: intero, reale, stringa, elenco, T o zero – In caso di successo, setvar restituisce il valore assegnato alla variabile.

In AutoLISP esistono oltre trecento funzioni di sistema, raggruppabili in cinque categorie e suddivise in un totale di ventitré sottocategorie.

- **Var. sorgente** (SETQ IS 10)
- **Var. destinazione** (SETQ ID VS)

Potremo eventualmente verificare l'uguaglianza tra due variabili reali, intere o di testo, tramite la funzione **=**[4] o tramite la funzione **EQ**[5] che consente di valutare anche elenchi e nomi anche di entità AutoCAD o tramite la funzione **EQUAL**[6] che consente di indicare un range numerico di tolleranza nella valutazione:

- (= IS ID) → T
- (= IA ID) → nil
- (EQ IA ID) → nil
- (EQUAL IA ID 5) → T

Nelle espressioni sopra le verifiche danno come risultato il valore T (vero) o il valore nil (falso); nil indica più esattamente il dato nullo; è possibile settare una variabile sia a T che a nil, settare a nil una variabile preesistente equivale ad eliminare il suo contenuto e liberare la memoria.

La funzione COMMAND

Per comprendere il livello d'interoperabilità tra AutoLISP ed AutoCAD la funzione di sistema **COMMAND**[7] risulta molto esplicativa perché consente di eseguire i normali comandi AutoCAD rispondendo automaticamente alle domande che il comando stesso porrebbe al prompt dei comandi. Ad esempio, per disegnare in AutoCAD un segmento per due punti, scriveremmo sulla linea di comando le seguenti righe di testo dando poi invio ad ogni passaggio:

- **LINEA**
- **Specificare primo punto**: 1,1
- **Specificare punto successivo o [Annulla]**: 2,2
- **Specificare punto successivo o [Annulla]**:

Se volessimo trasformare questi quattro inserimenti in una singola istruzione AutoLISP (fig. 1), ci basterà passare i dati specificati manualmente sulla linea di comando nei quattro passaggi alla funzione COMMAND; scriveremmo direttamente sulla linea di comando la riga che segue dando poi invio per eseguirla:

- (COMMAND "linea" '(1 1) '(2 2) "")

L'esecuzione dell'espressione manda in esecuzione il comando AutoCAD linea, colloca il primo punto nelle coordinate 1,1, il secondo punto nelle coordinate 2,2, infine termina la linea eseguendo un invio; ogni comando ed ogni sua opzione sono racchiusi tra doppi apici e separati da uno spazio. La funzione command replica, in modo automatico, le operazioni che normalmente faremmo manualmente; è anche possibile utilizzare nel command l'istruzione speciale **PAUSE** che ne ferma l'esecuzione attendendo l'input dell'utente:

- (COMMAND "linea" '(1 1) PAUSE "")

In questo caso il secondo punto della linea viene richiesto all'utente in fase di esecuzione. È anche possibile utilizzare quali argomenti della funzione COMMAND delle variabili utente: possiamo ad esempio definire preventivamente con SETQ come variabili i punti iniziale **PA** e finale **PB** della linea per poi passare tali variabili al COMMAND che si occupa di disegnare la linea e terminando con un invio:

- (SETQ PA '(1 1) PB '(2 2))
- (COMMAND "linea" PA PB "")

Operando in questo modo potremmo utilizzare i due punti **PA PB** per disegnare non solo una linea, ma ad esempio anche un cerchio per due punti, avente per punti diametrali **PA PB** (fig. 1):

- (SETQ PA '(1 1) PB '(2 2))
- (COMMAND "linea" PA PB "")
- (COMMAND "cerchio" "2P" PA PB

Funzioni utente

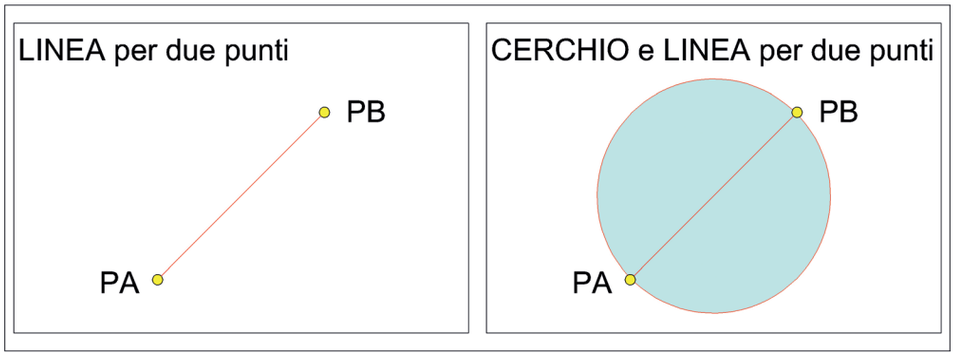
Fino ad ora ci siamo limitati a scrivere semplici linee di codice AutoLISP direttamente sulla linea di comando di AutoCAD: ora vedremo come sia possibile scrivere una nuova funzione AutoLISP. La sua struttura sarà quella di una serie di espressioni nidificate in una espressione che si occupa di definire la funzione; tale codice sarà poi richiamabile attraverso l'utilizzo di un identificatore che ne rappresenta il nome. È possibile definire funzioni utente principalmente tramite la funzione di sistema **DEFUN**[8]; i principali vantaggi nell'uso delle funzioni personalizzate sono tre:

[4] **=** – (= numstr [numstr...]) – Confronta gli argomenti valutandone l'uguaglianza numerica; se l'argomento è testuale lo confronta in termini ASCII.
Numstr – Tipo: intero, reale o stringa – Un numero o una stringa.
Valori restituiti – Tipo: T o nil – T, se tutti gli argomenti sono numericamente uguali; altrimenti nil.

[5] **EQ** – (eq expr1 expr2) – Determina se due espressioni sono identiche.
Expr1 – Tipo: intero, reale, stringa, elenco, nome (nome entità), T o nil – L'espressione da confrontare con expr2.
Expr2 – Tipo: intero, reale, stringa, elenco, nome (nome entità), T o nil – L'espressione da confrontare con expr1.
Valori restituiti – Tipo: T o nil – Se le espressioni sono identiche restituisce T; altrimenti nil.

[6] **EQUAL** – (equal expr1 expr2 [fuzz]) – Determina se due espressioni sono uguali. Nel confrontare due numeri reali apparentemente identici, essi possono differire leggermente se vengono utilizzati metodi diversi per calcolarli. Per questa ragione in equal è possibile specificare un importo fuzz per compensare tale eventuale differenza.
Expr1 – Tipo: intero, reale, stringa, elenco, nome (nome entità), T o nil – L'espressione da confrontare con expr2.
Expr2 - Tipo: intero, reale, stringa, elenco, nome (nome entità), T o nil – L'espressione da confrontare con expr1.
Fuzz – Tipo: intero o reale – Un numero reale che definisce l'importo massimo per il quale expr1 ed expr2 possono differire ed essere comunque considerati uguali.
Valori restituiti – Tipo: T o nil – Se le due espressioni sono uguali T; altrimenti nil.

[7] **COMMAND** – (command [arguments...]) – Esegue comandi di AutoCAD, valuta ciascun argomento e lo invia ad AutoCAD in risposta alle richieste successive. Invia i nomi dei comandi e le opzioni come stringhe, i punti 2D o 3D come elenchi di numeri reali. Può essere richiesto con PAUSE un input dell'utente in AutoCAD.
Arguments – Tipo: intero, reale, stringa o elenco – Comandi di AutoCAD e relative opzioni, come previsto dalla sequenza di prompt del comando eseguito. Una stringa nulla ("") equivale a premere Invio sulla tastiera.
Valori restituiti – Tipo: zero – Restituisce sempre zero.



1 | *L'uso alla riga di comando di AutoCAD delle funzioni AutoLISP COMMAND per l'esecuzione di comandi AutoCAD e della funzione SETQ per il settaggio di variabili:*
A sinistra il risultato dell'esecuzione di una singola espressione AutoLISP per il disegno di una linea per due punti di coordinate (1 1) e (2 2):
● *(COMMAND "linea" '(1 1) '(2 2) "")*
A destra il risultato dell'esecuzione di una serie di espressioni AutoLISP per il disegno di una linea e di un cerchio per due punti PA e PB preventivamente definiti come variabili e sempre di coordinate (1 1) e (2 2):
● *(SETQ PA '(1 1)" PB '(2 2))*
● *(COMMAND "linea" PA PB "")*
● *(COMMAND "cerchio" "2P" PA PB)*

● rendono più efficiente il lavoro di programmazione evitando di ripetere parti di programma più volte all'interno del codice complessivo di un programma;

● possono definire librerie di funzioni personalizzate riutilizzabili all'interno di più programmi senza doverle riscrivere ogni volta;

● permettono di rendere più chiara e fluida la struttura del codice di un programma dividendolo in piccole parti, ognuna delle quali svolge un determinato compito.

Finora, tutte le variabili che sono state dichiarate possono essere viste, ed utilizzate, in qualsiasi punto del nostro programma e per questo prendono il nome di **variabili globali**.
Le funzioni utente permettono invece di definire delle variabili visibili solo ed esclusivamente al loro interno che prendono il nome di **variabili locali**; una volta terminata la funzione queste variabili spariscono; tutto il codice all'esterno di tale funzione non può avere accesso a tali variabili a meno che non sia richiamato all'interno della funzione; viceversa potenzialmente le funzioni possono accedere ad eventuali variabili globali esterne ad esso in quanto residenti in memoria. AutoLISP permette al programmatore di creare particolari tipologie di funzioni che possono essere usate in maniera particolare e che possiamo indicare con un nome specifico:

● **funzioni predicato**: si distingue solamente per la sua specificità di avere il valore restituito in uscita limitato a due possibili valori: T nil (vero/falso);

● **funzioni comando**: possono essere utilizzate dall'operatore CAD nell'ambiente Autocad alla riga di comando come qualsiasi altro comando disponibile. Ciò che differenzia questa funzione dalle altre è l'indicazione "c:" prima del nome della funzione: tale prefisso trasforma qualsiasi funzione in un comando AutoCAD. In Tal caso, non è possibile avere dei parametri nella definizione della funzione, ma sarà possibile utilizzare le variabili locali;

● **funzioni nidificate**: la funzione nidificata sym2 viene definita all'interno del corpo della funzione ospitante sym1; sim2 viene dichiarata variabile locale della funzione sym1; in tal modo essa viene creata all'avvio della funzione contenitore e distrutta al suo termine, come avviene per una variabile locale. Si opera così per le procedure utilizzate raramente ed in specifici contesti per evitare l'accumularsi di procedure inutilizzate ed il degrado delle prestazioni del sistema.

Commento al codice

Con l'introduzione dei concetti e delle strutture espresse fin qua, è già possibile iniziare a scrivere procedure di una certa complessità (fig.2) e (fig.4). Il codice con molte funzioni e di notevole complessità, generalmente è di difficile rilettura e la comprensione del suo codice scritto, soprattutto se i programmi fatti vengono ripresi dopo lungo tempo per correzioni o modifiche, non è semplice.
Per poter rendere più chiaro e comprensibile un listato di programma, AutoLISP mette a disposizione la possibilità di inserire righe di commento per aiutare il programmatore nella descrizione delle operazioni svolte.
I commenti sono sempre inseriti dopo il carattere punto e virgola **(;)**: tutto ciò che, su quella stessa riga, viene digitato dopo il punto e virgola non viene considerato in fase di esecuzione del programma ed è quindi considerato commento al codice. Se il punto e virgola viene utilizzato all'interno di una coppia di doppi apici, verrà considerato

8 **DEFUN** – (defun [c:]sym ([arguments] [/ variables...]) expr...) – Definisce una funzione. Se non si dichiara alcun argomento o simbolo locale, si deve comunque fornire una serie vuota di parentesi dopo il nome della funzione.
Sym – Tipo: simbolo – Un simbolo che definisce il nome della funzione. Attenzione: non utilizzare mai il nome di una funzione incorporata o di un simbolo preesistente per l'argomento sym in defun. Ciò sovrascrive la definizione originale e rende inaccessibile la funzione o il simbolo integrato. Premettendo al nome della funzione c: la si rende utilizzabile direttamente nell'ambiente AutoCAD come comando personalizzato; è anche possibile ridefinire comandi di AutoCAD.
Arguments – Tipo: intero, reale, stringa, elenco, T o zero – I nomi degli eventuali argomenti attesi dalla funzione.
/ Variables – Tipo: simbolo – Gli eventuali nomi di una o più variabili locali per la funzione. La barra che precede i nomi delle variabili deve essere separata dal primo nome locale e dall'ultimo argomento, se presente, da almeno uno spazio.
Expr – Tipo: elenco – Qualsiasi numero di espressioni AutoLISP da valutare quando viene eseguita la funzione nidificate nell'espressione della funzion defun.
Valori restituiti – Il risultato dell'ultima espressione valutata.

> *Il listato del programma AutoLISP, salvato in formato di testo con estensione .lsp, può essere caricato in AutoCAD e mandato in esecuzione.*

parte di una semplice stringa di testo. È possibile utilizzare tale funzionalità anche per disabilitare parti di codice che non si vogliono mandare in esecuzione e che non si vogliono cancellare dal listato.

Un esempio di codice

Considerando l'esempio visto nella descrizione della funzione COMMAND, realizziamo del codice che permetta di aggiungere un comando AutoCAD per il disegno di un cerchio di centro PC e raggio rag e per il disegno del suo diametro per i punti PA PB che verranno chiesti in fase di esecuzione come input dal programma; per realizzare tale algoritmo, sarà necessario utilizzare altre funzioni AutoLISP che può essere utile introdurre:

- **GETPOINT**[9] consente, ponendo in pausa l'esecuzione, l'input utente di un punto e lo restituisce;
- **DISTANCE**[10] restituisce la distanza tra due punti;
- **ANGLE**[11] restituisce un angolo in radianti di un segmento per due punti;
- **POLAR**[12] restituisce il punto a un angolo e a una distanza specificati da un punto.

I programmi AutoLISP personalizzati possono essere scritti utilizzando un'ampia gamma di strumenti:

- editor di testo **ASCII** di base, come **Blocco note**, **Notepad++**, **TextEdit** o altri;
- ambiente di sviluppo integrato **Visual LISP (IDE)**; (fig. 2) nell'ambiente di sviluppo, l'editor di testo differenzia visivamente le varie componenti del codice con colorazioni differenti: in **rosso** le parentesi, in **blu** le funzioni di sistema, in **nero** le variabili e le funzioni utente, in **verde** i numeri, in **magenta** i testi, in **viola con sfondo grigio** i commenti al codice;
- **Microsoft Visual Studio Code** (fig. 4) con l'estensione AutoLISP installata;
- redattori di terze parti.

Nell'iniziare a sviluppare un programma in AutoLISP o in un altro linguaggio di programmazione, sarebbe opportuno considerare i seguenti passaggi:

- pensare ai compiti che deve svolgere;
- progettare il programma nella sua struttura sequenziale (algoritmo);
- scrivere il codice individuando le varie funzioni da definire;
- aggiungere commenti al codice per ricordare in momenti successivi come si intendeva procedere o anche per comunicare ad altri il proprio intento nella scrittura di quel codice;
- formattare il codice per facilitarne la leggibilità;
- testare il programma ed eseguirne il debug.

Tenendo conto dei passaggi appena descritti, possiamo delineare i compiti che il programma che vogliamo progettare dovrà eseguire (fig. 3):
Noti i punti **PA PB** forniti quale input (GETPOINT) in fase di esecuzione, sarà possibile determinare la loro distanza (DISTANCE) e direzione (ANGLE) e determinare il loro punto medio (POLAR); effettuati tali calcoli e salvati i risultati in opportune variabili (SETQ), sarà possibile procedere alla realizzazione del disegno del cerchio e del suo diametro (COMMAND).

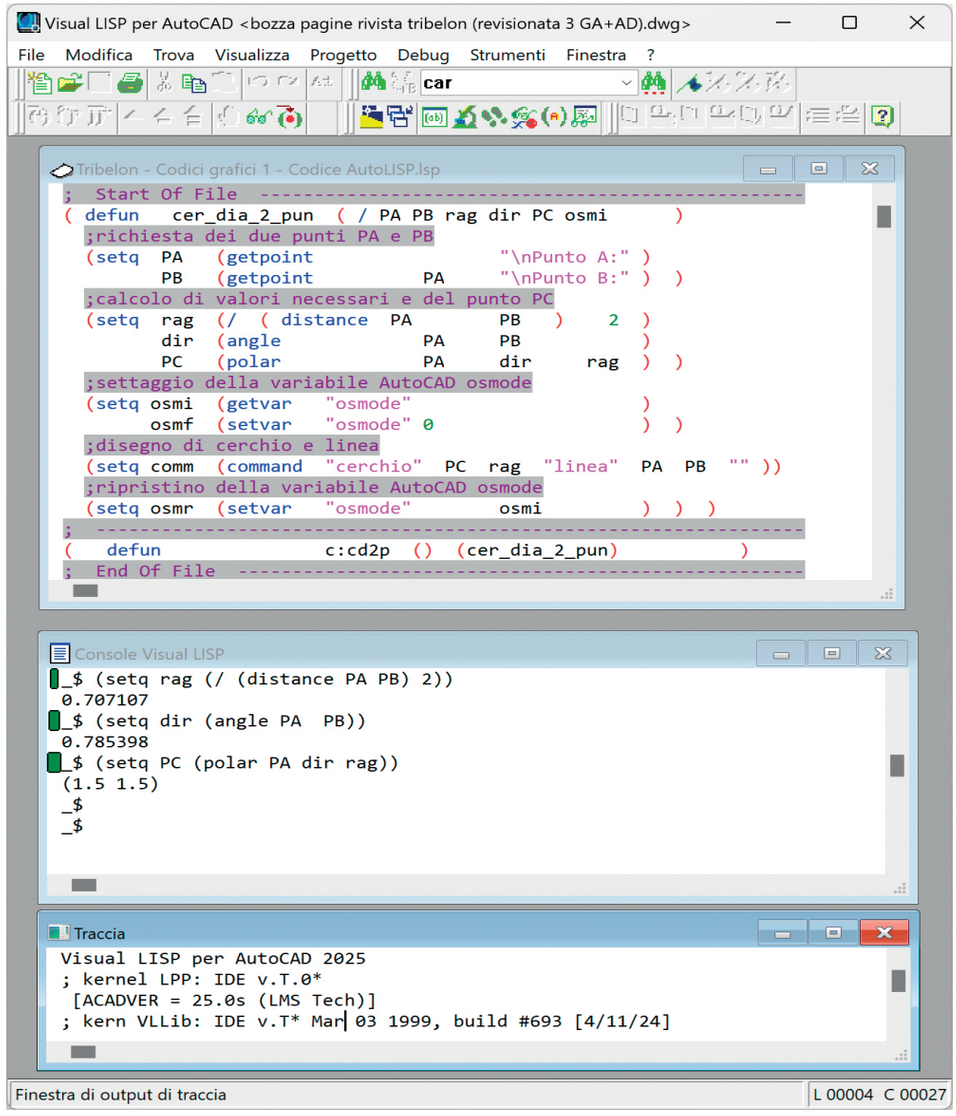
[9] **GETPOINT** – (getpoint [pt] [msg]) – Fa una pausa per l'input dell'utente di un punto e lo restituisce. L'utente può selezionare un punto o inserirne le coordinate 2D o 3D.
Pt – Tipo: elenco – Un punto base 2D o 3D nell'UCS corrente. Se l'argomento pt è presente, AutoCAD disegna e aggiorna in tempo reale una linea elastica da quel punto alla posizione corrente del cursore.
Msg – Tipo: stringa di testo – Messaggio da visualizzare quale richiesta all'utente.
Valori restituiti – Tipo: Elenco o zero – Un punto 3D di coordinate (x y z), espresso nell'UCS corrente.

[10] **DISTANCE** – (distance pt1 pt2) – Restituisce la distanza 3D tra due punti.
Pt1 – Tipo: elenco – Un elenco di coordinate di un punto 2D o 3D.
Pt2 – Tipo: elenco – Un elenco di coordinate di un punto 2D o 3D.Valori restituiti – Tipo: reale – La distanza. Se uno o entrambi i punti forniti sono un punto 2D, allora distance ignora le coordinate Z di eventuali punti 3D forniti e restituisce la distanza 2D tra i punti come proiettata nel piano di costruzione corrente.

[11] **ANGLE** – (angle pt1 pt2) – Restituisce un angolo in radianti di un segmento definito da due punti 2D o 3D. L'angolo viene misurato dall'asse X del piano di costruzione corrente, con angoli crescenti in senso antiorario.
Pt1 – Tipo: elenco – Un elenco di coordinate di un punto 2D o 3D.
Pt2 – Tipo: elenco – Un elenco di coordinate di un punto 2D o 3D.
Valori restituiti – Tipo: reale – Un angolo, in radianti.

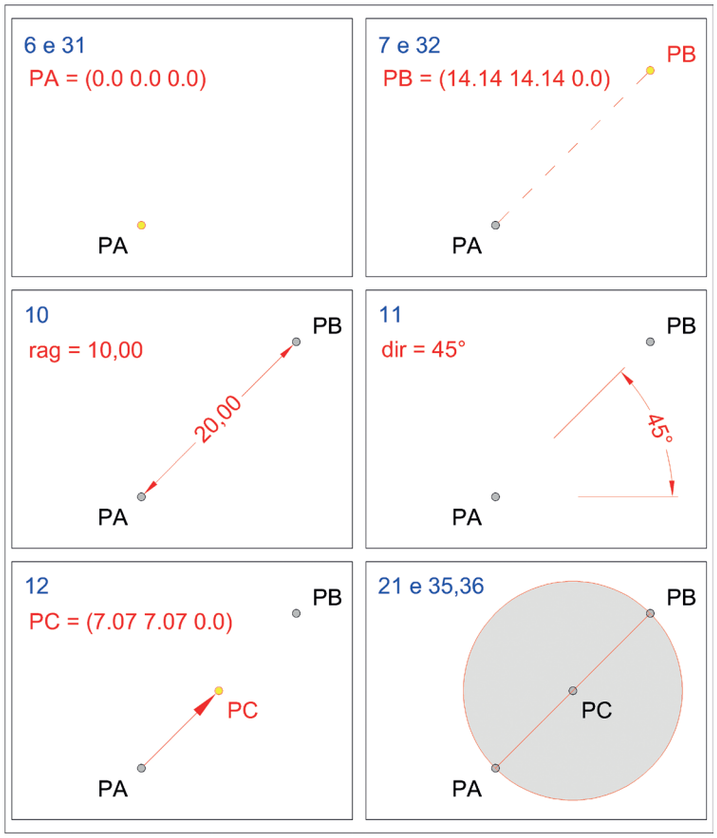
[12] **POLAR** – (polar pt ang dist) – Restituisce il punto UCS 3D a un angolo e a una distanza specificati da un punto.
Pt – Tipo: elenco – Un punto 2D o 3D.
Ang – Tipo: intero o reale – Un angolo espresso in radianti rispetto all'asse X dell'UCS. Gli angoli aumentano in senso antiorario, indipendentemente dal piano di costruzione corrente.
Dist – Tipo: intero o reale – Distanza dal punto specificato.
Valori restituiti – Tipo: elenco – Un punto 2D o 3D, a seconda del tipo di punto specificato da pt.

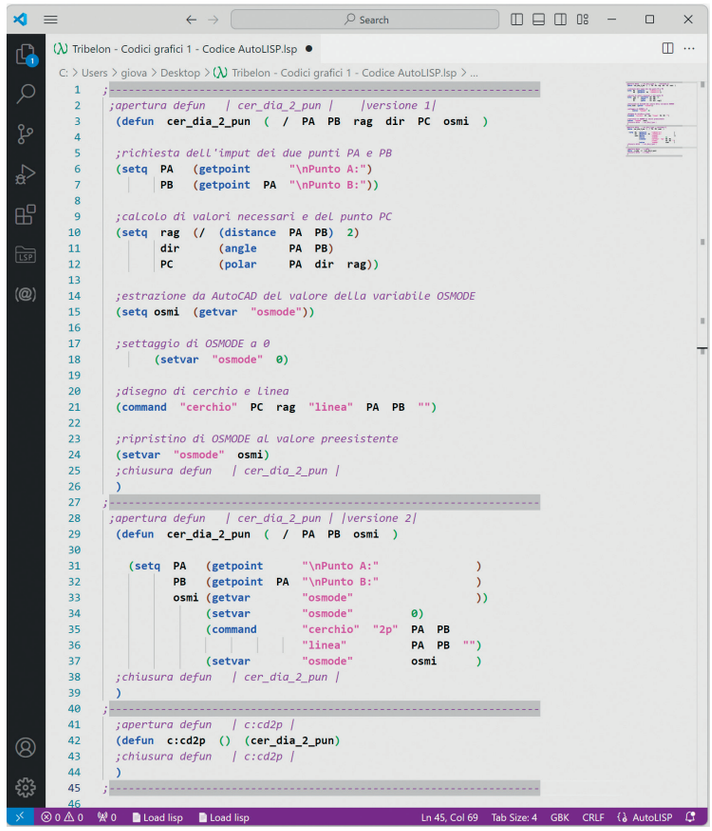
2 | *Gli elementi caratteristici dell'interfaccia utente di Visual LISP IDE inclusa in AutoCAD.*
Barra dei menu – Visualizzato nella parte superiore della finestra, fornisce l'accesso a tutti gli strumenti di modifica, visualizzazione e debug disponibili per lavorare sui file di progetto AutoLISP e DCL.
Barre degli strumenti – Finestre ancorate che forniscono un accesso rapido agli strumenti di uso comune.
L'editor di testo – Molto più di uno strumento di scrittura, è un componente centrale dell'ambiente di programmazione Visual LISP. Per apprezzare la versatilità e il valore dell'editor di testo, è necessario avere familiarità con il linguaggio di programmazione AutoLISP. L'editor di testo ha diverse funzionalità specifiche per la programmazione:
codifica colore dei file – Assegnare colori distinti a diverse parti di un programma AutoLISP. Ciò consente di trovare facilmente i componenti del programma e i nomi di variabili e aiuta a trovare errori tipografici;
formattazione del testo – Formattare il codice AutoLISP così come viene digitato, semplificandone la lettura. È possibile impostare la modalità di formattazione del codice AutoLISP selezionando tra diversi stili;
corrispondenza tra parentesi – Identifica le parentesi mancanti trovando la parentesi chiusa che accompagna una parentesi aperta;
esecuzione di espressioni AutoLISP – Testa espressioni e righe di codice senza uscire dall'editor di testo;
controllo della sintassi del codice AutoLISP – Valuta il codice AutoLISP ed evidenzia gli errori di sintassi.
Finestra della console – Versione di Visual LISP del prompt dei comandi di AutoCAD. È possibile immettere espressioni AutoLISP e valutarle direttamente da Visual LISP oppure eseguire comandi Visual LISP invece di utilizzare il menu o le barre degli strumenti. La console ha diverse funzionalità se ne riportano le principali:
È possibile valutare le espressioni AutoLISP e visualizzare il valore restituito. Premendo Invio viene valutata l'espressione AutoLISP immessa.
Le espressioni AutoLISP possono essere immesse su più righe premendo Ctrl+Invio per continuare con la riga successiva. Premendo Invio vengono valutate le espressioni AutoLISP immesse. È possibile valutare più espressioni contemporaneamente. Il testo può essere copiato e trasferito tra la console e le finestre dell'editor di testo.
Finestra di traccia – Finestra del messaggio informativo visualizzata ridotta a icona all'avvio. I messaggi visualizzati contengono informazioni sulla versione corrente di Visual LISP e su eventuali errori riscontrati durante l'avvio.
Barra di stato – Visualizzato nella parte inferiore della finestra Visual LISP e fornisce informazioni contestuali in base all'attività attualmente eseguita.



Dal punto di vista geometrico, i punti **PA** e **PB** sono gli estremi del diametro del cerchio, sarà possibile determinarne il raggio rag della circonferenza dimezzando la distanza tra i due punti **PA PB**, e posizionare il centro **PC** della circonferenza nel punto medio di **PA PB**, determinandolo da **PA** in direzione **dir** da **PA** a **PB** a distanza pari al raggio **rag**.

Nelle operazioni di disegno andranno temporaneamente disabilitate le funzionalità object snap salvandone la configurazione corrente rilevata nella variabile di sistema di AutoCAD **osmode** (GETVAR) in una variabile **osmi** al fine di ripristinarne la configurazione (SETVAR) ad operazioni di disegno completate.
Siamo ora in grado di scrivere il codice del programma (fig. 2) (fig. 4) contenente la funzione per disegnare il cerchio ed il suo diametro da due punti forniti in fase di esecuzione (**cer_dia_2_pun**) e la funzione del comando (**c:cd2p**) per usare in AutoCAD il programma realizzato.
Il listato del programma, scritto nell'ambiente di sviluppo che si ritiene opportuno o anche semplicemente in un editor di testo, può essere salvato in formato di testo con estensione **.lsp** e può essere, quando necessario, caricato in AutoCAD e mandato in esecuzione come un qualsiasi altro comando di autocad digitando **cd2p** sulla linea di comando di Autocad.
I testi riportati nella bibliografia che segue, sono idonei ad utenti non programmatori e che si avvicinano ora ad AutoLISP; nonostante la loro semplicità consentono comunque di approfondire gli argomenti fin qui trattati; altri testi più approfonditi verranno invece indicati successivamente.





3 | *Una graficizzazione in AutoCAD delle fasi di esecuzione del codice del programma. L'indicazione delle righe di comando mostrate in blu trova corrispondenza nell'interfaccia Microsoft Visual Studio Code mostrata nella figura successiva (fig. 4):*
righe 6 e 31: tramite la funzione GETPOINT viene richiesto e poi memorizzato dalla funzione SETQ il punto PA; righe 7 e 32: noto PA, tramite la funzione GETPOINT viene richiesto e poi memorizzato dalla funzione SETQ il punto PB; l'operzione viene agevolata dalla visualizzaizone in tempo reale di una linea elastica uscente dal punto PA; riga 10: noti i punti PA e PB, tramite la funzione DISTANCE viene calcolata e poi memorizzata dalla funzione SETQ la distanza rag da PA e PB del loro punto medio PC che verrà determinato successivamente; riga 11: noti i punti PA e PB, tramite la funzione ANGLE viene calcolata e poi memorizzata dalla funzione SETQ la direzione dir da PA in direzione e verso PB; riga 12: noti PA rag e dir, tramite la funzione POLAR viene calcolato e poi memorizzato dalla funzione SETQ il punto PC quale punto posto a distanza rag da PA in direzione dir; riga 21: noti PA PB PC e rag, tramite la funzione COMMAND vengono disegnati il segmento di vertici PA e PB e la circonferenza di centro PC e raggio rag; righe 35, 36: noti PA PB, tramite la funzione COMMAND vengono disegnati, in maniera alternativa, il segmento di vertici PA e PB e la circonferenza di punti diametrali PA e PB.

4 | *La visualizzazione del codice realizzato nell'interfaccia Microsoft Visual Studio Code con l'estensione AutoLISP installata; tra gli elementi caratteristici l'indicazione delle righe di codice a sinistra del codice ed una vista ridotta sull'intero codice a destra, utile soprattutto per programmi molto lunghi nei quali consente una navigazione per blocchi di codice. Il codice trova una corrispondenza in una graficizzazione in AutoCAD del disegno risultante ed è mostrato nella figura precedente (fig. 3).*
righe da 2 a 26 del codice: una prima versione della funzione cer_dia_2_pun commentata passo passo, formattata in modo da agevolarne la lettura e che segue i passaggi delineati nel progetto preliminare;
righe da 28 a 39 del codice: una seconda versione della funzione cer_dia_2_pun priva di commenti, con una formattazione alternativa e che ottimizza l'algoritmo eliminando dei passaggi che risultano inessenziali costruendo un cerchio per due punti invece che tramite la conoscenza del suo centro e del suo raggio. La nuova versione, venendo interpretata in successione della precedente la sostituisce a tutti gli effetti.
righe da 41 a 44 del codice: la funzione c:cd2p che predispone l'utilizzo quale comando in AutoCAD della funzione cer_dia_2_pun.

Bibliografia

Autodesk, AutoCAD: manuale di personalizzazione release 13, Autodesk development B.V., 1994.

M. Agosto, *AutoLISP: corso base per utenti non programmatori*, Tecniche nuove, Milano 1993.

T. Bousfield, *A practical guide to Autocad AutoLISP*, Longman, London 1998.

G.O. Head, *AutoLISP in plan english: A practical guide for non-programmers*, Ventana press, 1990.

C. Piccini, *LISP Trek: Guida all'uso del linguaggio LISP in ambiente CAD*, Lampi di stampa, Milano 2007.

R. Rossi, *Il mio Lisp*, http://redchar.net, Edizione 08.2016.1 (formato PDF).