OSAWARE

BASIC Language Reference Manual

The Geekprocessor v0.2a

Introduction

OSAWARE BASIC is a browser-based BASIC interpreter modelled on classic home computer BASIC dialects (Microsoft BASIC, BBC BASIC, GW-BASIC). Programs run in a terminal window with optional canvas graphics.

Program Structure

Every line of a BASIC program begins with a line number (1–4999). Lines execute in ascending numerical order. Blank line numbers are skipped. Use REM for comments.

```
10 REM This is a comment
20 PRINT "Hello, World!"
30 END
```

Line Numbers

Line numbers must be integers between 1 and 4999. Lines are stored in a sparse array — you can leave gaps for future edits. Conventional spacing is 10 lines apart.

⚠ *Line numbers 5000 and above are silently ignored. Keep all line numbers below 5000.*

Variables

Numeric variables: `single or multi-letter names (A, NR, MAXI, ZR2)`. Default value is 0.

String variables: `name must end with $ (A$, NAME$, C$)`. Default value is empty string.

💡 *Variable names are case-insensitive. A and a refer to the same variable.*

Data Types

BASIC supports two data types: numbers (floating point) and strings. There is no boolean type — conditions return 0 (false) or −1 / non-zero (true).

Operators

Arithmetic: + − * / ^ (power) % (modulo)

Comparison: = <> < > <= >=

Logical: AND OR NOT

String: + (concatenation)

Core Commands

PRINT — Output text or values to the terminal	
Syntax	`PRINT [expr] [; expr] [, expr]`
Example	`PRINT "Hello, World!"`
	`PRINT "X = "; X; " Y = "; Y`
	`PRINT TAB$(10); "Indented"`
	`PRINT ' blank line`
Idiom	*Use semicolons to suppress spacing between items. Trailing semicolon suppresses newline. PRINT with no args prints a blank line.*

INPUT — Accept input from the user	
Syntax	`INPUT ["prompt";] var [, var2]`
Example	`INPUT X`
	`INPUT "Enter your name: "; NAME$`
	`INPUT "Width, Height: "; W, H`
Idiom	*Always provide a prompt string so the user knows what to enter. INPUT halts execution until the user presses Enter.*

LET — Assign a value to a variable	
Syntax	`LET var = expr  (LET is optional)`
Example	`LET X = 42`
	`X = 42 ' LET is optional`
	`A$ = "hello"`
	`X = X + 1 ' increment`
	`L$ = L$ + C$ ' string concatenation`
Idiom	*LET is optional in OSAWARE BASIC. Direct assignment X=42 is preferred. Multiple assignments on one line use colons: X=1 : Y=2 : Z=3*

REM — Add a comment (ignored at runtime)	
Syntax	`REM comment text`
Example	`10 REM === MAIN PROGRAM ===`
	`100 REM Calculate Mandelbrot iteration`
	`200 X = X + 1  ' inline comment via REM not supported`
Idiom	*Use REM liberally to document your code. Conventional style uses === markers for section headers.*

END — Stop program execution	
Syntax	`END`
Example	`980 END`

	`IF ERR=1 THEN END`
Idiom	*Every program should have at least one END. Place END before subroutine definitions to prevent accidental fall-through.*

Flow Control

GOTO — Jump unconditionally to a line number	
Syntax	`GOTO linenum`
Example	`GOTO 1000`
	`GOTO 100`
Idiom	*Use GOTO to jump to subroutine sections or restart loops. Avoid using GOTO inside tight batch loops — prefer WHILE or FOR instead, as IF THEN GOTO inside nested loops can cause freezes.*

IF — Conditionally execute a statement	
Syntax	`IF condition THEN statement IF condition THEN GOTO linenum`
Example	`IF X=42 THEN PRINT "Found it!"`
	`IF A$="yes" THEN GOTO 500`
	`IF X<0 THEN X=0`
	`IF SPD<1 THEN SPD=2`
	`IF IT>=MAXI THEN C$="*"`
	`IF IT<MAXI AND DN=3 THEN C$="+"`
Idiom	*Use sequential IFs instead of IF THEN GOTO inside loops. For flag-based branching: set a flag variable with IF, then act on the flag with another IF on the next line.*

GOSUB / RETURN — Call a subroutine and return	
Syntax	`GOSUB linenum RETURN`
Example	`530 GOSUB 4800`
	`540 PRINT "GV = "; GV`
	`...`
	`4800 REM Subroutine`
	`4810 GV = 42`
	`4820 RETURN`
Idiom	*Place subroutines after END to avoid accidental fall-through. Subroutines communicate via global variables. GOSUB/RETURN can be nested.*

FOR / NEXT — Loop a fixed number of times	
Syntax	`FOR var = start TO end [STEP step]  ... NEXT var`
Example	`FOR I = 1 TO 10`
	`  S = S + I`
	`NEXT I`
	`FOR I = 0 TO N-1  ' expression limit`
	`  PRINT I`

	`NEXT I`
	`FOR I = 10 TO 1 STEP -1    ' count down`
	`  PRINT I`
	`NEXT I`
Idiom	*Start, end, and step can all be expressions or variables. Nested FOR loops are supported — each must use a different variable. The loop body is skipped entirely if start > end (with positive step).*

WHILE / WEND — Loop while a condition is true	
Syntax	`WHILE condition    ... WEND`
Example	`WHILE I < 10`
	`  I = I + 1`
	`WEND`
	`WHILE IT<MAXI AND ZR*ZR+ZI*ZI<4`
	`  TMP = ZR*ZR - ZI*ZI + CR`
	`  ZI = 2*ZR*ZI + CI`
	`  ZR = TMP`
	`  IT = IT + 1`
	`WEND`
Idiom	*The condition is evaluated at the top of each iteration. Compound conditions use AND and OR. Full arithmetic expressions are supported on both sides of the comparison.*

String Functions

LEN — Return the length of a string	
Syntax	`LEN(string$)`
Example	`PRINT LEN("Hello")  ' prints 5`
	`IF LEN(A$) = 0 THEN PRINT "empty"`
	`FOR I = 1 TO LEN(S$)`
Idiom	*Use LEN to validate input or iterate over characters.*

LEFT$ / RIGHT$ / MID$ — Extract substrings	
Syntax	`LEFT$(string$, n) RIGHT$(string$, n) MID$(string$, start, len)`
Example	`LEFT$("Hello", 3)  ' "Hel"`
	`RIGHT$("Hello", 3)  ' "llo"`
	`MID$("Hello", 2, 3)  ' "ell"`
	`IF LEFT$(CMD$,3)="RUN" THEN GOSUB 500`
Idiom	*MID$ is 1-based. MID$(S$,1,1) returns the first character. Use MID$ in a loop to process strings character by character.*

STR$ / VAL — Convert between numbers and strings	
Syntax	`STR$(number) VAL(string$)`
Example	`A$ = STR$(42)  ' "42"`
	`N = VAL("3.14")  ' 3.14`
	`N = VAL(INPUT$)`
Idiom	*VAL stops at the first non-numeric character and returns 0 for non-numeric strings.*

CHR$ / ASC — Convert between characters and ASCII codes	
Syntax	`CHR$(n) ASC(string$)`
Example	`PRINT CHR$(65)  ' A`
	`PRINT CHR$(42)  ' *`
	`PRINT ASC("A")  ' 65`
Idiom	*Use CHR$() to assign strings containing operator characters (* + - / etc.) without confusing the parser.*

INSTR — Find position of a substring	
Syntax	`INSTR(string$, search$) INSTR(start, string$, search$)`
Example	`INSTR("Hello World", "World")  ' 7`
	`INSTR("Hello", "xyz")  ' 0 (not found)`
Idiom	*Returns 0 if not found. Returns 1-based position if found.*

Math Functions

INT / ABS / SGN — Integer, absolute value, sign	
Syntax	`INT(n)  ABS(n)  SGN(n)`
Example	`INT(3.7)  ' 3`
	`INT(-3.7)  ' -4  (floor, not truncate)`
	`ABS(-5)  ' 5`
	`SGN(-3)  ' -1`
	`SGN(0)  ' 0`
	`SGN(5)  ' 1`
Idiom	*INT always rounds toward negative infinity. Use INT(X+0.5) for nearest-integer rounding.*

SQR / LOG / EXP — Square root, natural log, e^n	
Syntax	`SQR(n)  LOG(n)  EXP(n)`
Example	`SQR(16)  ' 4`
	`LOG(2.71828)  ' ~1`
	`EXP(1)  ' ~2.71828`

SIN / COS / TAN / ATN — Trigonometric functions (radians)	
Syntax	`SIN(r)  COS(r)  TAN(r)  ATN(r)`
Example	`SIN(3.14159/2)  ' ~1  (sin 90°)`
	`COS(0)  ' 1`
	`PI = 4 * ATN(1)  ' calculate pi`
	`RAD = DEG * PI / 180  ' degrees to radians`
Idiom	*All trig functions use radians. To convert degrees to radians: multiply by PI/180.*

RND / RANDOMIZE — Random number (0 to 1)	
Syntax	`RND RANDOMIZE [TIMER]`
Example	`X = RND  ' 0 to 0.999...`
	`D = INT(RND * 6) + 1  ' dice roll 1-6`
	`RANDOMIZE TIMER  ' seed from clock`
Idiom	*Always call RANDOMIZE TIMER at program start for truly random sequences. Without it, the same sequence repeats every run.*

Graphics Commands

Graphics commands draw on the canvas behind the terminal. The first graphics command automatically activates graphics mode, making the terminal background transparent. Use CLS to clear and return to black.

PSET / PRESET — Draw or erase a single pixel	
Syntax	`PSET x, y [, colour] PRESET x, y`
Example	`PSET 100, 100, 3 ' green pixel`
	`PSET PX, PY, INT(IT*15/MAXI)+1`
	`PRESET 100, 100 ' erase pixel`
Idiom	*PSET is the building block of all graphics. The Mandelbrot graphics mode uses PSET for every pixel. Colour 0 = white, 16 = black.*

LINE — Draw a straight line	
Syntax	`LINE x1, y1, x2, y2 [, colour]`
Example	`LINE 0, 0, 200, 150, 3`
	`LINE 0, 0, WIDTH, HEIGHT, 2 ' diagonal`
Idiom	*Use LINE for wireframe graphics, axes, and borders. Combine with SLEEP for animated drawing.*

CIRCLE — Draw a circle	
Syntax	`CIRCLE x, y, radius [, colour]`
Example	`CIRCLE 200, 150, 50, 4`
	`FOR R = 10 TO 100 STEP 10`
	`  CIRCLE 200, 150, R, 3`
	`NEXT R`
Idiom	*Draw concentric circles with a FOR loop. Use PAINT to fill after drawing.*

RECT — Draw a rectangle outline	
Syntax	`RECT x1, y1, x2, y2 [, colour]`
Example	`RECT 10, 10, 200, 100, 3`

PAINT — Flood fill from a point	
Syntax	`PAINT x, y [, colour]`
Example	`CIRCLE 200, 150, 50, 3`
	`PAINT 200, 150, 3 ' fill circle`
Idiom	*PAINT fills outward from x,y until it hits pixels of a different colour. Draw the outline first, then PAINT inside.*

COLOUR — Set the current text/graphics colour

Syntax	`COLOUR n`
Example	`COLOUR 3 ' green text`
	`COLOUR 2 ' red text`
	`COLOUR 7 ' light gray`
Idiom	*Colour 3 (green) is the classic terminal colour. Use COLOUR before PRINT to change text colour.*

`CLS` — Clear screen — terminal and canvas	
Syntax	`CLS`
Example	`CLS`
	`5 CLS ' clear on startup`
Idiom	*CLS at the top of every program for a clean start. CLS fills the canvas with black and clears the terminal.*

`DELAY / SLEEP` — Pause execution	
Syntax	`DELAY ms SLEEP ms`
Example	`DELAY 0 ' batch mode (fastest)`
	`DELAY 16 ' ~60fps`
	`SLEEP 80 ' pause 80ms per row`
	`SLEEP 300 ' pause 300ms then continue`
Idiom	*DELAY sets the per-line delay for the execution engine. DELAY 0 enables batch mode for maximum speed. SLEEP pauses once for the given milliseconds.*

System Commands

LOAD / RUN / SAVE — Load, run, or save a BASIC program	
Syntax	`LOAD filename RUN [filename] SAVE filename`
Example	`LOAD MANDEL`
	`RUN`
	`RUN MENU`
	`SAVE MYPROG`
Idiom	*LOAD reads a program from the virtual filesystem. RUN starts execution. RUN MENU loads and runs the menu. SAVE stores the current program in browser memory (not permanent).*

LIST — Display program lines	
Syntax	`LIST [start] [- end]`
Example	`LIST`
	`LIST 100`
	`LIST 100 - 200`
Idiom	*LIST is essential for debugging — view the loaded program to confirm it matches expectations.*

NEW — Clear the current program	
Syntax	`NEW`
Example	`NEW`
Idiom	*NEW erases the program in memory. Cannot be undone — save first if needed.*

MEM — Display memory usage	
Syntax	`MEM`
Example	`MEM`

TRON / TROFF — Enable/disable line number trace	
Syntax	`TRON TROFF`
Example	`TRON ' show [linenum] as each executes`
	`TROFF ' turn off tracing`
Idiom	*TRON is invaluable for debugging — it shows which line is executing in real time. Use TROFF to disable once the bug is found.*

AI Commands

OSAWARE BASIC includes built-in AI integration via the Anthropic Claude API.

AIKEY — Set the Anthropic API key	
Syntax	`AIKEY "your-api-key"`
Example	`AIKEY "sk-ant-..."`
Idiom	*Set once per session. The key is stored in memory only — never saved to disk.*

AI — Send a prompt and get a text response	
Syntax	`AI "prompt" [, RESULT$]`
Example	`AI "Tell me a joke"`
	`AI "What is the capital of France?", A$`
	`PRINT A$`
Idiom	*Without a variable, the response prints directly. With a variable, it is stored for further processing.*

AINUM — Send a prompt and get a numeric response	
Syntax	`AINUM "prompt", var`
Example	`AINUM "What is 2+2?", N`
	`PRINT N  ' 4`
Idiom	*Use AINUM when you need the AI to return a number for further calculation.*

Common Idioms & Patterns

Initialise on startup

```
5  CLS
10 COLOUR 3
20 PRINT "MY PROGRAM"
30 PRINT "==========="
```

Safe numeric input with default

```
100 INPUT "Speed [1-4]: "; SPD
110 IF SPD<1 THEN SPD=2
120 IF SPD>4 THEN SPD=2
```

Mode dispatch without IF THEN GOTO

```
60  INPUT "Mode: "; MODE
70  C$=" "
80  IF MODE=1 THEN C$="text"
90  IF MODE=2 THEN C$="graphics"
100 PRINT "Selected: "; C$
```

Avoid IF THEN GOTO inside loops. Use sequential IFs with flag variables instead.

Character selection from iteration count

```
1070 C$=" "
1071 IF IT>=MAXI THEN C$="*"
1072 IF IT<MAXI THEN DN=INT(IT*7/MAXI)
1073 IF IT<MAXI AND DN=1 THEN C$="."
1074 IF IT<MAXI AND DN=2 THEN C$=":"
1075 IF IT<MAXI AND DN=3 THEN C$="+"
```

Assign a default first, then overwrite with more specific conditions. Each IF is independent — no ELSE needed.

Mandelbrot core loop

```
WHILE IT<MAXI AND ZR*ZR+ZI*ZI<4
  TMP = ZR*ZR - ZI*ZI + CR
  ZI  = 2*ZR*ZI + CI
  ZR  = TMP
  IT  = IT + 1
WEND
```

Use TMP to hold the new ZR value while computing ZI — both depend on the old ZR.

Building a string row character by character

```
L$ = ""
FOR CL = 0 TO NC-1
  C$ = "*"                    ' set character
  L$ = L$ + C$                ' append
NEXT CL
PRINT L$
```

Graphics animation loop

```
DELAY 0
FOR PY = 0 TO PH-1
  FOR PX = 0 TO PW-1
    PSET PX, PY, COLOUR_VALUE
  NEXT PX
  SLEEP SYLD                 ' pause between rows
NEXT PY
```

Use nested FOR loops for graphics — not manual PX/PY counters with IF THEN GOTO.

GOSUB subroutine pattern

```
500 GOSUB 4000
510 PRINT "Result: "; RESULT
...
980 END                      ' prevent fall-through

4000 REM === Subroutine ===
4010 RESULT = 42
4020 RETURN
```

Prevent string literal confusion

```
C$ = "*"                     ' OK - * inside quotes
C$ = CHR$(42)                ' alternative - chr code
```

Strings containing operator characters (+ - / ^ %) are handled correctly in OSAWARE BASIC. Both forms work.*

Colour Palette

BASIC uses 1-based colour numbers (1–16). Use COLOUR n for text, or pass directly to graphics commands.

1	Lavender #a398ff	**2**	Red #FF0000	**3**	Green #00FF00

Colour codes: 1=Lavender 2=Red 3=Green 4=Yellow 5=Magenta 6=Cyan 7=LtGray
8=DkBlue 9=DkRed 10=DkGreen 11=LtYellow 12=DkMagenta 13=DkCyan
14=Orange 15=Gray 16=Black

💡 *Colour 16 (black) is used for the Mandelbrot set interior. Colour 3 (green) is the classic terminal colour.*

Quick Reference Card

Command	Usage
`PRINT x;y`	Print values, semicolon suppresses space/newline
`INPUT "p";V`	Prompt user, store in V
`LET V=expr`	Assign (LET optional)
`GOTO n`	Jump to line n
`IF c THEN s`	Execute s if c is true
`GOSUB n / RETURN`	Call subroutine at n, return here
`FOR I=a TO b STEP s`	Loop I from a to b in steps of s
`WHILE c / WEND`	Loop while c is true
`END`	Stop program
`REM text`	Comment
`CLS`	Clear screen
`COLOUR n`	Set colour (1-16)
`PSET x,y,c`	Draw pixel at x,y colour c
`LINE x1,y1,x2,y2,c`	Draw line
`CIRCLE x,y,r,c`	Draw circle
`PAINT x,y,c`	Flood fill
`DELAY ms`	Set engine delay (0=fastest)
`SLEEP ms`	Pause for ms milliseconds
`LOAD file`	Load program from VFS
`RUN [file]`	Run program
`LIST [n-m]`	List program lines
`TRON / TROFF`	Trace on/off
`MEM`	Show memory usage
`AI "prompt",V$`	Ask AI, store response in V$