NAME

tinybasic − Tiny BASIC interpreter and compiler

SYNOPSIS

tinybasic [**options**] *program-file*

DESCRIPTION

Tinybasic is an implementation of the Tiny BASIC language. It conforms to the specification by Dennis Allison, published in People's Computer Company Vol.4 No.2 and reprinted in Dr. Dobb's Journal, January 1976.

The package provides both an interpreter and a compiler in the same executable. Both of these tools are non-interactive, and load their input from a source file written in a text editor. No interactive interpreter is provided.

Tinybasic provides additional features in the form of optional line numbers (also known as numeric line labels), and a configurable upper limit to the line labels in place of the 255 of Allison's specification.

OPTIONS

−n *value*, **−−line−numbers=***value*

Determines the handling of numeric line labels (known as line numbers in BASIC). An argument of **m** or **mandatory** causes **tinybasic** to require a numeric label for every program line, in ascending order. An argument of **i** or **implied** causes **tinybasic** to supply numeric labels internally for each line that lacks them; care must be taken when labelling lines so that there is room for a sequence of numbers between one label and the next. An argument of **o** or **optional** makes numeric labels completely optional; they need not be in sequence.

−N *limit*, **−−line−number−limit=***limit*

Specifies the largest numeric line label allowed in the BASIC program. The default is 255. The new limit may be up to 32767.

−o *comment-option*, **−−comments=***comment-option*

Enables or disables support for comments and blank lines in programs. *Comment-options* can be **e** or **enabled** to supports comments and blank lines, which is the default setting. It can be **d** or **disabled** to disable support for comments, which follows accurately the original Tiny BASIC specification.

PROGRAM FORMAT

Programs are text files loaded in on invoking **tinybasic**. Each line of the file consists of an optional numeric line label, a command keyword, and the command's arguments, if it has any. Lines may be blank, or the command keyword may be REM, which denotes that the rest of the program line is a comment.

That describes a program written using **tinybasic**'s additional features. In traditional Tiny BASIC program each line has a mandatory numeric line label, a command keyword which may not be REM, and the command's arguments. The original Tiny BASIC specification did not provide for comments or blank lines, and the numeric line labels were required by the language's interactive line editor.

Note that what this manual terms ´'numeric line labels But since tinybasic allows omission of these labels, errors are reported with the actual line number of the source file, which start at 1 and increment regardless of the numeric line labels in the program. It is to these source file line numbers that the term ´'line numbers

COMMANDS

LET *variable=expression*

Assigns a value, the result of *expression*, to a variable, *variable*. *Variable* must be a single letter, A..Z. *Expression* must evaluate to an integer in the range -32768 to 32767.

IF *condition* **THEN** *statement*

Conditional execution. If*condition* is true, then *statement* is e xecuted. *Statement* can be another **IF**, allowing conditions to be chained, effectively mimicking an AND operator.

GOTO *expression*

Transfer execution to another part of the program. *Expression* is evaluated, and program execution continues at the line marked with the appropriate numeric label.

GOSUB *expression*

Calls a subroutine. *Expression* is evaluated, and program execution transfers to the line marked with the appropriate label. The position of the **GOSUB** is remembered so that a **RETURN** can bring program execution back to the statement following the **GOSUB**.

RETURN

Return from a subroutine. Program execution returns to the statement following the **GOSUB** which called the present subroutine.

END Terminates program execution.

PRINT *output-list*

Produces output to the console. *Output-list* is a list of items separated by commas. Each item can be either a string literal enclosed in double quotation marks, or a numeric expression. An end of line sequence is output at the end, so that the next **PRINT** statement will put its output on a new line.

INPUT *variable-list*

Asks for input from the console. *Variable-list* is a list of variable names. For each variable given, a question mark is output and the value typed by the user is stored in that variable.

REM *comment-text*

Provides space for free-format comment text in the program. Comments have no effect on the execution of a program, and exist only to provide human-readable information to the programmer. Use of this command will raise an error if support for comments is disabled (see the **-o**/**--comment** option above).

EXPRESSIONS

Expressions in Tiny BASIC are purely arithmetic expressions, involving integers only. The four basic arithmetic operators are supported: multiplication (*****), division (**/**), addition (**+**) and subtraction (**-**). Unary operators for positive (**+**) and negative (**-**) are supported, as are parentheses for affecting the order of operations.

Standard operator precedence evaluates parentheses first, then unary signs, then multiplication and division, with addition and subtraction last.

CONDITIONS

The relational operators are **=**, **>**, **<**, **<>** or **><**, **>=**, and **<=**. They are not supported within arithmetic expressions, but can only be used as conditions in IF statements in the form: *expressionrelational-operator-expression*

VERSION INFORMATION

This manual page documents **tinybasic**, version 1.0.

AUTHORS

Tiny BASIC was originally designed by Dennis Allison. This implementation was written by Damian Gareth Walker.

EXAMPLE

This program prints out all of the numbers in the Fibonnaci series between 0 and 1000.

```
        LET A=0
        LET B=1
        PRINT A
100     PRINT B
        LET B=A+B
        LET A=B-A
        IF B<=1000 THEN GOTO 100
        END
```