Understanding Text Files

What is in a Text File?

A text file contains combinations of single characters. Usually we are thinking of the alphanumeric characters, such as letters, numbers, and punctuation.

In addition, there are special characters that we don't normally use or notice.

The most common collection of characters is called the ASCII Character Set.

Computer Standards

When computers were being developed, there were no standards. Companies created their own methods for storing data, which made it difficult to write programs for different systems.

Plain text was one of the early ways to store data, and it was recognized that a standard was necessary for consistency in the computer industry.

The two most popular standards were ASCII and EBCDIC

The ASCII Standard

ASCII stands for “American Standard Code for Information Interchange”. It specifies the numeric codes (integers) that represent all of the characters that can be used in a text file.

For example, the letter 'A' is 65, 'B' is 66, etc. Similarly, 'a' is 97, 'b' is 98, etc.

The standard ASCII table has 128 codes, numbered from 0 to 127.

Characters <--> ASCII Codes

Even though we perform complex operations with computers using text, they really only understand numbers.

In fact, computers really only understand the numbers 1 and 0. This is the <u>binary</u> number system.

All computer information actually has a numerical representation, and the ASCII codes are one way to accomplish this for text files.

What about File I/O?

When we perform input or output using a text file, we are actually working with a sequence of ASCII codes.

The computer does not see the file the way we do. There are no letters, numbers, spaces, and new lines. Each of these concepts corresponds an ASCII code.

How We See a Text File

hello
how are you?

How The Computer Sees It

hello<LF><CR>how are you?<EOF>

Even this is not completely accurate, since the computer sees the codes as numbers.

ASCII Special Characters

There are many special characters in the ASCII table. We often put special codes in <>. The most important to text files are:

<CR> carriage return, <LF> line feed
These two characters combine to represent a new line (like hitting the “Enter” key)

<EOF> End of File
This represents the end of the text file.

Turing – the “get” command

```
get : file, word            % Ex.1
get : file, word :*         % Ex.2
```

We have encountered this subtle difference when reading text from a file.

By default, Turing assumes that values are separated by spaces or new lines (Ex.1).

We can force Turing to use everything, including spaces, up to the new line (CR+LF) with the “`:*`” at the end of the get (Ex.2)

Example: Input Data from a File

```
var fileNum : int
var data : array 1..10 of string
open : fileNum, "someFile.txt", get

for i : 1..10
   get : fileNum, data(i) : *
end for

close : fileNum
```

Input Data from a File

```
for i : 1..10
   get : fileNum, data(i) : *
end for
```

The previous example will only work if there are 10 or more data items in the file. If there are fewer than 10, we will go past the end of the file (EOF), which is called a data overrun.

Using EOF to Read Files

Going past the EOF is a problem. Thus it makes sense to look for EOF when reading data, and stop reading when we reach EOF. This can be accomplished using the EOF function.

```
i := 1
loop
   exit when eof(fileNum)
   get : fileNum, data(i) : *
   i := i + 1
end loop
```

Using EOF to Read Files

We can also limit based on the amount of data we want, or the size of the array.

```
i := 1
loop
   exit when eof(fileNum) or i > upper(data)
   get : fileNum, data(i) : *
   i := i + 1
end for
```