**−− −−− ∙−∙ ∙∙∙ ∙**

**MORSE CODE**

**−∙−∙ −−− −∙∙ ∙**

## Description of the Program

The program is a system that converts between plaintext and Morse code.

Plain text is language printed alphabetically (A, B, C, etc.), whereas Morse code uses patterns of dots and dashes to represent each letter in the alphabet:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Plaintext** | **Morse code** | **Plaintext** | **Morse code** | **Plaintext** | **Morse code** |
| A | **.-** | J | **.---** | S | **...** |
| B | **-...** | K | **-.-** | T | **-** |
| C | **-.-.** | L | **.-..** | U | **..-** |
| D | **-..** | M | **--** | V | **...-** |
| E | **.** | N | **-.** | W | **.--** |
| F | **..-.** | O | **---** | X | **-..-** |
| G | **--.** | P | **.--.** | Y | **-.--** |
| H | **....** | Q | **--.-** | Z | **--..** |
| I | **..** | R | **.-.** |  |  |

Each character is separated by a space, so the word HELLO is represented as follows:

**.... . .-.. .-.. ---**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| H | E | L | L | O |
| **....** | **.** | **.-..** | **.-..** | **---** |

Included within the pre-release material is a text file called ‘message.txt’. The contents of this file are as follows:

**===**⌂⌂⌂**=**⌂⌂⌂**=**⌂**===**⌂⌂⌂⌂⌂⌂⌂**===**⌂**=**⌂**=**⌂**===**⌂

***Note:*** *The* ⌂ *symbols are not included in the text file, they have been included in these notes to represent spaces,   
to make them more visible for this explanation. The message.txt file consists of spaces and equals symbols only.*

## Overview

The program has two subroutines that handle conversion between plaintext and Morse code:

**ReceiveMorseCode**

The subroutine ReceiveMorseCode reads Morse code from a text file and converts it to plain text. One of the key subroutines used to perform this conversion is Decode. The subroutine SendMorseCode takes plaintext from the user at the keyboard and converts it to Morse code.

ReceiveMorseCode consists of three main stages:

1. Extract text from a file. The file contains only spaces and equals symbols. A single equals (=) makes a dot. Three in a row (===) make a dash.   
     
   Below is an extract from the text file:

**===⌂=⌂=⌂===**

1. Convert the series of equals symbols to a series of dots and dashes. The sequence in the box above would become:

**-..-**

1. Convert the series of dots and dashes to plaintext, which is a letter between A and Z. The pattern in the box above would become:

**X**

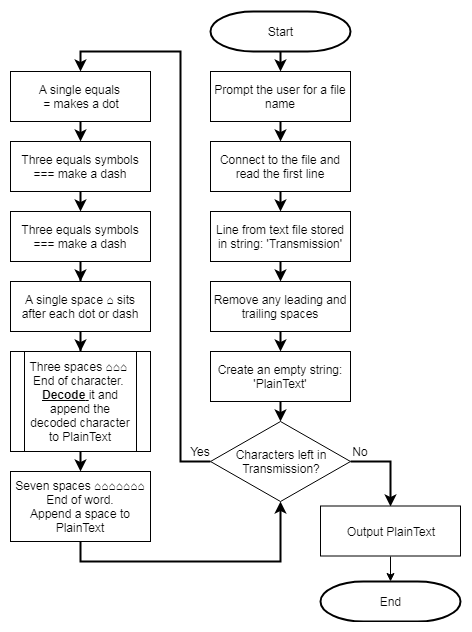
This process is repeated until the entire message has been translated into plaintext, at which point it is displayed in the console.

**SendMorseCode**

SendMorseCode is less involved. The user types uppercase plaintext at the console, which is converted into Morse code. The Morse code is then displayed on the console. Any spaces in the plaintext are represented as three spaces in Morse code.

|  |  |
| --- | --- |
| **Input** | **Output** |
| COMPUTING | **-.-. --- -- .--. ..- - .. -. --.** |
| AQA AS | **.- --.- .- .- ...** |

## ReceiveMorseCode Subroutine

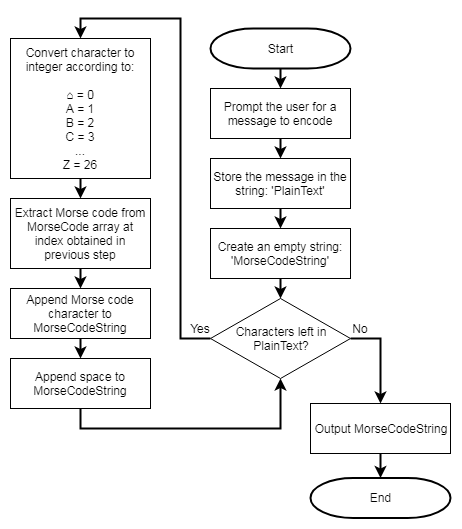


ReceiveMorseCode calls seven other subroutines, either directly or indirectly. These calls are not all included in the flowchart, as the flowchart exists only to provide a top-level understanding of the program.

## Decode Subroutine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element index in list:** | **Dot** | **Dash** | **Letter** | The subroutine Decode uses three lists in parallel;  Dot, Dash and Letter, whose contents remain the same throughout execution.  The flowchart below shows how Decode would translate the pattern **-.-** into the plaintext character ‘K’:  C:\Users\Septimus\Downloads\Untitled Diagram (4).png  If the first character is a dash (-), the program starts by looking at index 0 in the Dash list. If the first character is a dot (.), the starting point is index 0 of the Dot list. |
| 0 | 5 | 20 | ⌂ |
| 1 | 18 | 23 | A |
| 2 | 0 | 0 | B |
| 3 | 0 | 0 | C |
| 4 | 2 | 24 | D |
| 5 | 9 | 1 | E |
| 6 | 0 | 0 | F |
| 7 | 26 | 17 | G |
| 8 | 0 | 0 | H |
| 9 | 19 | 21 | I |
| 10 | 0 | 0 | J |
| 11 | 3 | 25 | K |
| 12 | 0 | 0 | L |
| 13 | 7 | 15 | M |
| 14 | 4 | 11 | N |
| 15 | 0 | 0 | O |
| 16 | 0 | 0 | P |
| 17 | 0 | 0 | Q |
| 18 | 12 | 0 | R |
| 19 | 8 | 22 | S |
| 20 | 14 | 13 | T |
| 21 | 6 | 0 | U |
| 22 | 0 | 0 | V |
| 23 | 16 | 10 | W |
| 24 | 0 | 0 | X |
| 25 | 0 | 0 | Y |
| 26 | 0 | 0 | Z |

## SendMorseCode Subroutine



Unlike ReceiveMorseCode, which calls several other subroutines, SendMorseCode is self-contained, and calls no other subroutines. The user enters a message, which is validated to ensure it contains only upper-case characters and spaces. The message is then translated, one character at-a-time, using Morse code equivalents taken from an list called MorseCode.

## The Text File (message.txt)

The contents of the text file are explained below:

**===**⌂⌂⌂**=**⌂⌂⌂**=**⌂**===**⌂⌂⌂⌂⌂⌂⌂**===**⌂**=**⌂**=**⌂**===**⌂

|  |  |
| --- | --- |
| **===**⌂⌂⌂ | This is a dash (===), followed by three spaces.  Three spaces signals the end of a character, but not the end of a word.  The character that is made up of a single dash is the letter T. |
| **=**⌂⌂⌂ | This is the second character, which is a single dot, making it the letter E. |
| **=**⌂**===** | This character is a dot followed by a dash.  A single space is used between them (instead of three spaces), because the character is not finished yet.  The Morse code comprising a dot followed by a dash is for the letter A. |
| ⌂⌂⌂⌂⌂⌂⌂ | This is then followed by seven spaces, which are used to form a gap between two words. |
| **===**⌂**=**⌂**=**⌂**===**⌂ | This is a character that is made up of a dash, followed by a dot, followed by a dot, followed by a dash, which make up the letter X. |

The whole message, therefore, is **TEA X**

## Subroutine Calls, Parameters and Return Values

The numbers to the left do **not** indicate the order in which subroutines are called, as there are multiple possible orders.   
Instead, these numbers relate to the numbers in the structure diagram.

| **Call** | **Parameters** | **Return** |
| --- | --- | --- |
| 1 Main calls SendReceiveMessages | - | - |
| 2 SendReceiveMessages calls DisplayMenu | - | - |
| 3 SendReceiveMessages calls GetMenuOption | - | MenuOption |
| 4 SendReceiveMessages calls ReceiveMorseCode | Dash  Letter  Dot | - |
| 5 SendReceiveMessages calls SendMorseCode | MorseCode | - |
| 6 ReceiveMorseCode calls GetTransmission | - | Transmission |
| 7 ReceiveMorseCode calls GetNextLetter | i  Transmission | i  SymbolString |
| 8 ReceiveMorseCode calls Decode | CodedLetter  Dash  Letter  Dot | Letter[Pointer]  This returns a string, but the string is always one character long, and is a character within the string Letter, at location Pointer.  If Letter contains the string “Hello” , then Letter[0] = H, Letter[1] = E, etc. |
| 9 GetTransmission calls StripLeadingSpaces | Transmission | Transmission |
| 10 GetTransmission calls StripTrailingSpaces | Transmission | Transmission |
| 11 GetTransmission calls ReportError | s | - |
| 12 GetNextLetter calls GetNextSymbol | i  Transmission | i  Symbol |
| 13 StripLeadingSpaces calls ReportError | s | - |
| 14 GetNextSymbol calls ReportError | s | - |

## Description of Subroutines

Each subroutine is described below.

| Subroutine Name | Description |  |
| --- | --- | --- |
| Decode  *Receives a coded letter (i.e. a letter in Morse code, such as -..-), and returns the corresponding plain text letter (‘X’ in this case)* | Parameters: CodedLetter  Dash  Letter  Dot  Returns: Letter[Pointer]  Called from: ReceiveMorseCode  Calls: - | 1. Initialise an integer variable CodedLetterLength to be equal to the length of the parameter CodedLetter  2. Initialise an integer variable Pointer to zero  3. Set up a loop to iterate through each character in CodedLetter, using the variable i  4. If i points to a space, this subroutine returns a space to ReceiveMorseCode  5. If i points to a dash, Pointer is changed to navigate the Morse code binary tree (see Preliminary Material, page 4), one step to the left  6. If i points to a dot, Pointer is changed to navigate the Morse code binary tree, one step to the right  7. By the time i has looped through each dot/dash in the encoded character, the value of Pointer should point (in the Letter list) to the letter that corresponds to the Morse code letter  8. If a space is not returned to ReceiveMorseCode in step 4 (above), the letter identified in step 7 is returned as a string |
| DisplayMenu  *Displays three options to the user – send Morse code, receive Morse code or end the program* | Parameters: -  Returns: -  Called from: SendReceiveMessages  Calls: - | 1. Output three menu options (R, S, X), one on each line |
| GetMenuOption  *Prompts the user for a desired menu option, returning it to SendReceiveMessages* | Parameters: -  Returns: MenuOption  Called from: SendReceiveMessages  Calls: - | 1. Declare a string variable to store the user’s desired menu option  2. Set up a loop that repeats until the user input is a single character in length  3. Within that loop, prompt the user for their desired option (R, S or X)  4. Return the user input to SendReceiveMessages |
| GetNextLetter  *A Morse code transmission usually consists of multiple letters. This subroutine extracts the next letter from a transmission.* | Parameters: i  Transmission  Returns: SymbolString  Called from: ReceiveMorseCode  Calls: GetNextSymbol | 1. Declare string variable SymbolString and initialise it to an empty string  2. Set up a loop to repeat until any **one** of these conditions is met:   * A space is returned from a call to GetNextSymbol (meaning the Morse character being parsed has ended) * The EOL character (#) is reached (meaning the end of the entire message has been reached) * The two characters after the current character are both spaces (meaning the letter has ended)   3. Within the loop, a call is made to GetNextSymbol, which will return a space, a dash or a dot. A space (see first bullet point) terminates the loop  4. If the call to GetNextSymbol returns a dash or a dot, that dash or dot is appended to the string variable SymbolString  5. At the end of the word (see bullet points), SymbolString is returned to ReceiveMorseCode |
| GetNextSymbol  *A Morse code letter can consist of multiple symbols (combinations of dots and dashes). There are also spaces, which are used to separate them. This subroutine determines whether the next symbol is a dot, a dash or a space.* | Parameters: i  Transmission  Returns: Symbol  Called from: GetNextLetter  Calls: ReportError | 1. When the parameter i is initially passed to this subroutine, its value is zero  2. Integer variable SymbolLength initialised to zero  3. i is used to point to characters within the string variable Transmission  4. If i points to the # character, ‘*End of transmission’* is written to the console, and an empty string is returned to GetNextLetter  5. Otherwise, i is incremented until it reaches either a space or the EOF character (#) within Transmission  6. As i is incremented, SymbolLength is also incremented  7. When i points to a space in Transmission, SymbolLength will have a value of either 0, 1 or 3  8. If SymbolLength is 0, a space is returned to GetNextLetter  9. If SymbolLength is 1, a dot is returned to GetNextLetter  10. If SymbolLength is 3, a dash is returned to GetNextLetter  11. Any other value for SymbolLength indicates an error, and ReportError is called with the parameter *‘Non-standard symbol received*’ |
| GetTransmission  *This subroutine prompts the user for a filename, then reads the first line of the corresponding file, passing it back to ReceiveMorseCode* | Parameters: -  Returns: Transmission  Called from: ReceiveMorseCode  Calls: StripLeadingSpaces  StripTrailingSpaces  ReportError | 1. Prompt the user for a file name  2. Create a FileHandle connected to the specified file  3. Read the first line of the file into the variable Transmission  4. Pass the variable Transmission to the subroutine StripLeadingSpaces, from which it should be returned  5. If the length of Transmission at this point is greater than zero, pass it to StripTrailingSpaces, from which it should be returned  6. Append the EOL symbol (currently #) to the variable Transmission  7. If any errors occur between steps 2 and 6, call ReportError (passing ‘*No transmission found*’ as a parameter) and set the variable Transmission to an empty string  8. Return the variable Transmission to the subroutine ReceiveMorseCode |
| Main  *This subroutine only exists to start  the program (by calling SendReceiveMessages)* | Parameters: -  Returns: -  Called from: -  Calls: SendReceiveMessages | 1. Call SendReceiveMessages |
| ReceiveMorseCode  *Calls other subroutines to manage the process of retrieving an encoded message (in Morse code), extracting each letter in turn and decoding each letter as it is extracted* | Parameters: Dash  Letter  Dot  Returns: -  Called from: SendReceiveMessages  Calls: GetTransmission  GetNextLetter  Decode | 1. Set string variables PlainText and MorseCodeString to contain empty strings  2. Set the string variable Transmission to contain the return value from a call to the subroutine GetTransmission  3. Set the integer variable LastChar to point to the index of the last character in Transmission  4. Initialise an integer variable i to contain zero  5. Loop through each character in Transmission, with the exception of the last one (which should be the EOL (#) character)  6. Set the string variable CodedLetter to contain the return value from a call to the subroutine GetNextLetter  7. Append a SPACE and CodedLetter to MorseCodeString  8. Set the character variable PlainTextLetter to contain the return  value from Decode  9. Append PlainTextLetter to PlainText  10. Print out the MorseCodeString and PlainText |
| ReportError  *Writes an error to the console between two asterisks* | Parameters: s  Returns: -  Called from: GetTransmission  StripLeadingSpaces  GetNextSymbol  Calls: - | 1. The error message arrives as a string parameter called s  2. Parameter s is displayed between two asterisks |
| SendMorseCode  *Accepts a plain text input from the user, translates it into Morse code and outputs the translation to the console* | Parameters: MorseCode  Returns: -  Called from: SendReceiveMessages  Calls: - | 1. Prompt the user for a message to be encoded  2. Store the message in the variable PlainText  3. Store the length of the message in the variable PlainTextLength  4. Initialise variable MorseCodeString as an empty string  5. Set up a loop to iterate through each character in PlainText  6. If the character is a space, the integer variable Index is set to 0  7. Otherwise, Index is set to a number that represents that letter’s position in the alphabet, e.g. if the letter is A, Index will be set to 1; if the letter is B, Index will be set to 2; etc.  8. The value of Index is used as an index in the MorseCode list that was passed in as a parameter. For example, if the letter being examined was A, the value of Index would be 1. Element 1 would then be retrieved from the MorseCode list.  9. The Morse code value retrieved from the list is appended to the variable MorseCodeString, followed by a space  10. Once steps 6–9 have been performed on each character in the variable PlainText, the value of the variable MorseCodeString is printed |
| SendReceiveMessages  *This contains the main program loop, which repeatedly displays the menu, prompts the user for an input, then calls the appropriate subroutine in response. This loop ends when the user indicates a desire to end the program.* | Parameters: -  Returns: -  Called from: Main  Calls: DisplayMenu  GetMenuOption  ReceiveMorseCode  SendMorseCode | 1. Initialise Dash list (to contain integer pointers that relate to the Morse code binary tree)  2. Initialise the Letter list (SPACE, ‘A’, B’, ‘C’ … ‘Z’)  3. Initialise the Dot list (to contain integer pointers that relate to the Morse code binary tree)  4. Initialise the MorseCode list (to contain the Morse equivalents of letters, in the same order as the Letter list)  5. Begin a loop that continues until the user indicates that they want to end the program  5. Call DisplayMenu to display the menu  6. Call GetMenuOption to get user input from menu  7. Either call ReceiveMorseCode, call SendMorseCode, or terminate the loop, depending on user input |
| StripLeadingSpaces  *Removes any spaces from the left of a string* | Parameters: Transmission  Returns: Transmission  Called from: GetTransmission  Calls: ReportError | 1. Store the length of the transmission in the integer variable TransmissionLength  2. Set up a loop that repeats while the first character of Transmission is a space, and while the length of Transmission is greater than zero  3. Within that loop, decrement the variable TransmissionLength and remove the first character of Transmission  4. If, after the loop, the length of Transmission is zero, call the subroutine ReportError, passing to it the string ‘*No signal received*’ as a parameter |
| StripTrailingSpaces  *Removes any space from the right of a string* | Parameters: Transmission  Returns: Transmission  Called from: GetTransmission  Calls: - | 1. Set the integer variable LastChar to point to the index of the last character in Transmission  2. If the last character is a space, remove it from the string variable Transmission, then decrement LastChar  3. Repeat step 2 until LastChar does not point to a space |

## Description of Variables, Constants and Parameters

The following table contains variables ⓥ, constants ⓒ and parameters ⓟ

| Name | Type | Description | Created in / Passed to |
| --- | --- | --- | --- |
| CodedLetter ⓟ | String | Contains a single Morse code letter that is about to be decoded (passed by value) | Decode |
| CodedLetter ⓥ | String | Contains a single Morse code letter that is about to be decoded or has just been encoded | ReceiveMorseCode  SendMorseCode |
| CodedLetterLength ⓥ | Integer | The number of Morse symbols in an encoded letter | Decode |
| Dash ⓟ | Integer list | Contains pointers to left branches of the binary tree seen on the Preliminary Material document, page 4 (passed by value) | Decode  ReceiveMorseCode |
| Dash ⓥ | Integer list | Contains pointers to left branches of the binary tree seen on the Preliminary Material document, page 4 | SendReceiveMessages |
| Dot ⓟ | Integer list | Contains pointers to right branches of the binary tree seen on the Preliminary Material document, page 4 (passed by value) | Decode  ReceiveMorseCode |
| Dot ⓥ | Integer list | Contains pointers to right branches of the binary tree seen on the Preliminary Material document, page 4 | SendReceiveMessages |
| EMPTYSTRING ⓒ | String | Constant to store an empty string: “” | (global) |
| EOL ⓒ | Char | Constant to store # symbol, which marks the end of a line | (global) |
| FileHandle ⓥ | File Handle | Used to store a reference to the text file containing the transmission | GetTransmission |
| FileName ⓥ | String | Name (which can include path) of a text file to be read | GetTransmission |
| FirstSignal ⓥ | Char | Contains the first character of a string that’s being trimmed | StripLeadingspaces |
| i ⓟ | Integer | The index of a character within a Morse code string currently being processed | GetNextSymbol  GetNextLetter |
| i ⓥ | Integer | The index of a character within a Morse code string currently being processed  Note there is another variable called i in SendMorseCode but it is only used as a loop variable | ReceiveMorseCode |
| Index ⓥ | Integer | Stores a pointer used to access the correct Morse code character within a list | SendMorseCode |
| LastChar ⓥ | Integer | Points to the index of the last character in Transmission | StripTrailingSpaces  ReceiveMorseCode |
| Letter ⓟ | String list | Contains a space in the first element, followed by the upper-case alphabet, with each letter in its own element (passed by value) | Decode  ReceiveMorseCode |
| Letter ⓥ | String list | Contains a space in the first element, followed by the upper-case alphabet, with each letter in its own element | SendReceiveMessages |
| LetterEnd ⓥ | Boolean | Set to true if the end of a Morse code letter has been reached while it is being parsed character by character | GetNextLetter |
| MenuOption ⓥ | String | Contains the user’s response when presented with the program’s main menu | GetMenuOption  SendReceiveMessages |
| MorseCode ⓟ | String list | Contains a space in the first element, followed by Morse code equivalents for each letter, with one such letter per element (passed by value) | SendMorseCode |
| MorseCode ⓥ | String list | Contains a space in the first element, followed by Morse code equivalents for each letter, with one such letter per element | SendReceiveMessages |
| MorseCodeString ⓥ | String | An entire Morse code message, which can contain any number of Morse code characters | ReceiveMorseCode |
| MorseCodeString ⓥ | String | Contains a Morse code message, constructed character by character | SendMorseCode |
| PlainText ⓥ | String | Contains a message that has been (or is about to be) decoded from its Morse code equivalent | ReceiveMorseCode  SendMorseCode |
| PlainTextLength ⓥ | Integer | The number of characters to be converted to Morse code | SendMorseCode |
| PlainTextLetter ⓥ | Char | Contains a single, upper-case letter that has been decoded, i.e. is no longer Morse code | ReceiveMorseCode |
| PlainTextLetter ⓥ | Char | Contains each letter in turn, as it is about to be converted to Morse code | SendMorseCode |
| Pointer ⓥ | Integer | Points to the list element that will be accessed next as individual dots and dashes are parsed | Decode |
| ProgramEnd ⓥ | Boolean | Set to true if the main program loop is about to terminate | SendReceiveMessages |
| s ⓟ | String | Parameter (passed by value) that contains an error message | ReportError |
| Signal ⓥ | String | Variable to examine each character of Transmission in turn | GetNextSymbol |
| SPACE ⓒ | Char | Constant to store a single space character | (global) |
| Symbol ⓥ | Char | Contains a dot, dash or space within a Morse code letter | GetNextSymbol  Decode |
| Symbol ⓥ | String | Contains the value returned from GetNextSymbol (i.e. a single dot, dash or space) that forms part of a Morse code letter | GetNextLetter |
| SymbolLength ⓥ | Integer | Stores the number of characters in a single Morse code letter | GetNextSymbol |
| SymbolString ⓥ | String | Built up, one dot or dash at a time, into a Morse code letter | GetNextLetter |
| Transmission ⓟ | String | Stores a sequence of Morse code letters (passed by value) | StripLeadingSpaces  StripTrailingSpaces  GetNextSymbol  GetNextLetter |
| Transmission ⓥ | String | Stores a sequence of equals signs and spaces, used to represent Morse code as described in the Preliminary Material Document | GetTransmission  ReceiveMorseCode |
| TransmissionLength ⓥ | Integer | Stores the length of the Transmission variable | StripLeadingSpaces |