



M Ű E G Y E T E M 1 7 8 2

Budapest University of Technology and Economics

Basics of Programming 1 (Course code: BMEVIEEAA00)

Academic year: 2020/21/1

Developer Documentation: Morse conversion program

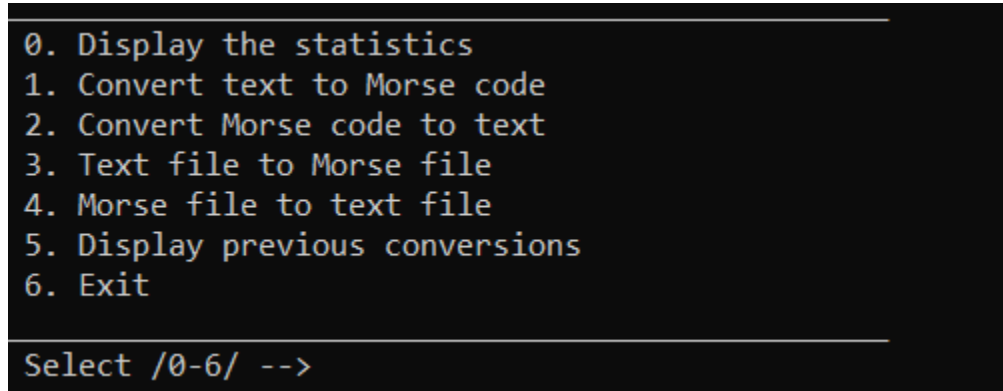
Neptun ID: QBI3JH

Name: Tushig Bat-Erdene

9th of December 2020

Morse conversion program:

The chosen project by myself was creating conversion program which is for code and decode text into the morse code or morse code into the text and storing them into the text files. This program is controlled by a command-line tool.



```
0. Display the statistics
1. Convert text to Morse code
2. Convert Morse code to text
3. Text file to Morse file
4. Morse file to text file
5. Display previous conversions
6. Exit

Select /0-6/ -->
```

Figure 1. Output of the program

According to the Figure 1, It displays the menu or switch case statements which is making possible to get the menu selection from the user. The details of the cases:

- Case 0: It displays the statistics of Morse alphabets and its conversion of Latin characters such as letters, numbers, and symbols.
- Case 1: It gets text from user to convert into the morse code
- Case 2: It gets morse code from user to convert into the text
- Case 3: It gets the text from user to create file.txt by storing given input to it and converting the given input to store in newfile.txt.
- Case 4: It gets the morse code from user to create file.txt by storing given input to it and converting the given input to store in newfile.txt.
- Case 5: It shows the previous conversions or history from case 1 to 4 and stores it into the generating history.txt.
- Case 6: It exits the program.

Requirements to build the program:

- Code Blocks or any other integrated development environment “IDE” which can compile C. (GNU C Compiler)
- Standard C library functions
 - o `#include<stdio.h>` “Standard Input/Output” → which is used for getting data from user
 - o `#include<stdlib.h>` “Standard Library” → which is used for defining data variable types, and functions. I used to null pointer for dynamic data structures.

- `#include<string.h>` “String”→ which defines variable type, and it is important to manipulate character arrays. I used string functions including `strlen`” used to find length of an array” and `strtok` “point to separate characters”.

Those were necessary to build my morse code conversion program.

Main part and the data structures (Binary tree, linked list, hash table, file handling):

I used binary tree for my morse program because it is the simplest to represent morse characters by connecting its nodes. It has at most 2 parent nodes (left, right) which is creating morse library easier.

As you can see from the Figure 2, I made left subtree of a nodes are the morse codes they are starting with dashes and right subtree of a nodes are starting with dots.

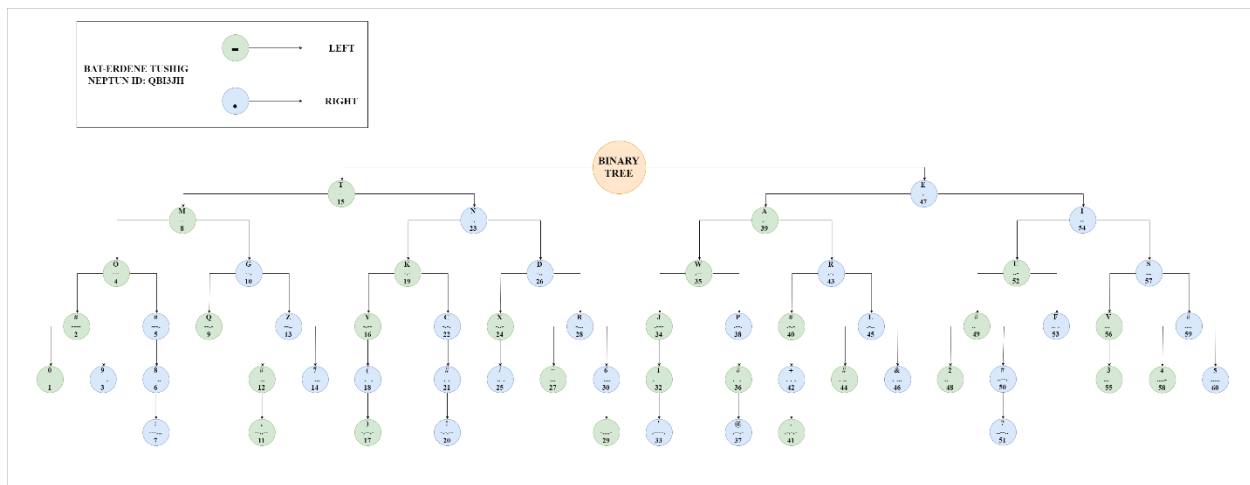


Figure 2. Binary tree diagram

Functions:

1. `typedef struct Binary_Tree`

I implemented this function by using linked list the sturture of `Binary_Tree` which is making it possible to search characters from library.

2. `typedef struct MorseLibrary`

I also implemented `MorseLibrary` function which has characters inside it by using linked list.

3. `MorseLibrary characters[]`

It is creating a global array which contains all characters and its translation in morse code.

4. `Binary_Tree* node(char item, int key)`

This pointer function is making it possible to put characters in the `Binary_Tree` by creating nodes for characters.

5. `Binary_Tree* insert_characters(Binary_Tree *start, char value, int key)`

This function is inserting characters to the Binary_Tree*. It can insert characters by depending on the key value which goes left or right.

6. Binary_Tree* CreateMorse(void)

This function is creating I created a well-ordered binary tree diagram (Figure 2.) which helped me a lot by defining characters and morse codes clearly. This function generating a pointer points start (root of binary search tree in my program) with the help of node function. It is inserting all available characters to tree using insert function.

It works like If you go left from the root (I used start to demonstrate root) you will find dash (- , which is character T), if you go to the right you will find dot (. , which is character E) and it will return to the start again. It decides whether go left or right by depending the key value which you can see it from Figure 2.

7. MorseLibrary InsertToML(char chr, char *string)

When I am writing the code of this program, I faced a problem that I couldn't get both text and morse code for only using a char but this function is making it possible to get both of them by character and string.

8. char Get_Morse_Code(char *str, Binary_Tree *start)

This function has a role to get string from a user to convert morse code into the text. It measures the size of string also creating the pointer from start root to get characters. As you can see the for loop to check whether it is left or right (left is dash, right is dot). If user inserts the morse code which is wrong or do not belong in the library of the program will print "Invalid Code!" in the debugged exe with the help of if function.

I also used file handling in case 3, case 4, and case 5 to store data's in the generated text files. In case 3 and case 4, I had to generate a text files to store my inputs and outputs in it.

9. void delete_node(Binary_Tree *root)

This function is deleting nodes from Binary tree until root will NULL.

10. void create_file(), void create_newfile(), void create_history_file()

Those three functions are creating files to allocate inputs and outputs.

11. void add_text(char *txt)

This function is adding text by appending char into the file.txt.

12. void add_morse_text(char *morsetxt)

This function is appending char into the newfile.txt.

13. void history_add_string(char *HistoryAdd)

This function is adding string to the generated history.file.

14. void history_line_break()

Printing lower dashes by separating previous conversions.

15. void TextToMorse(char *txt, int length)

This is the essential function for converting text into the Morse code. It is getting string and length of text to find translation. This function can add converted morse code into the newfile.txt which is essential for case 3.

16. void MorseTotext(char *str1, Binary_Tree *top)

This is the essential function for converting morse into the text. It has a big role to convert morse into the text. This function is getting string and calling its translation from CreateMorse function to convert. Also, I created file handling in this function to getting conversion to add into file.

Main function:

In the main function, I created a binary search tree and called MorseLibrary characters to print the statistics easily with using only for loop n the case 0. Also, it is making case 1 possible to easily convert the given text into the morse code by searching morse code from the tree. Case 3 is nearly similar to case 1 except called the file generation functions to place input to file.txt and output to newfile.txt

In case 2, it is calling function to get string from user to print it. Case 4 is similar to the case 2 except calling the file generation functions.

In case 5, I used file handling to print previous conversions in this section. It reads the contents from the history.txt file and making it possible to print in the debugged exe.

Testing if it is working correctly:

Testing case 0, It is successfully displaying the statistics.

Testing case 1 by entering input “test”, result correctly shows “- . . . -”.

Testing case 2 by entering input “- . . . -”, result correctly shows “test”.

Testing case 3 by entering input “testing”, result correctly shows “- . . . - .. -. --.” In the successfully generated files.

Testing case 4 by entering input ““- . . . - .. -. --.”, result correctly shows “testing-” In the successfully generated files. It automatically added “-” sign after the converted morse code in the newfile.txt.

Testing case 5, It is displaying without any errors and successfully inserted into the generated history.txt file.

Testing case 6, It successfully terminated the exe.

References:

[1] URL: <http://www.scphillips.com/morse/morse2.html>.

[2] URL: <http://www.hit.bme.hu/~ghorvath/bop>

[3] URL: <http://www-personal.acfr.usyd.edu.au/tbailey/ctext/ctext.pdf>