Developer document

My homework codes text into Morse, decodes Morse into text (binary tree must be used), gives statistics of Latin and Morse characters (dashes, dots). The coding/decoding should be controlled by command line switch and has infinitely long history of the results.

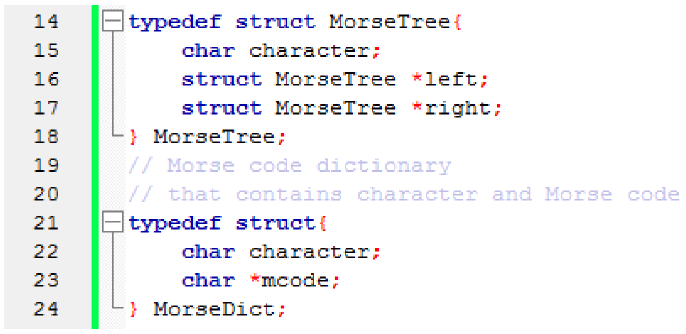
Libraries

I used 3 libraries to make this program.

- **<stdio>** to get data from user,
- **<string>** to modify strings, use string function such as strtok, strlen.
- **<stdlib>** to use null pointer to make dynamic data structures.

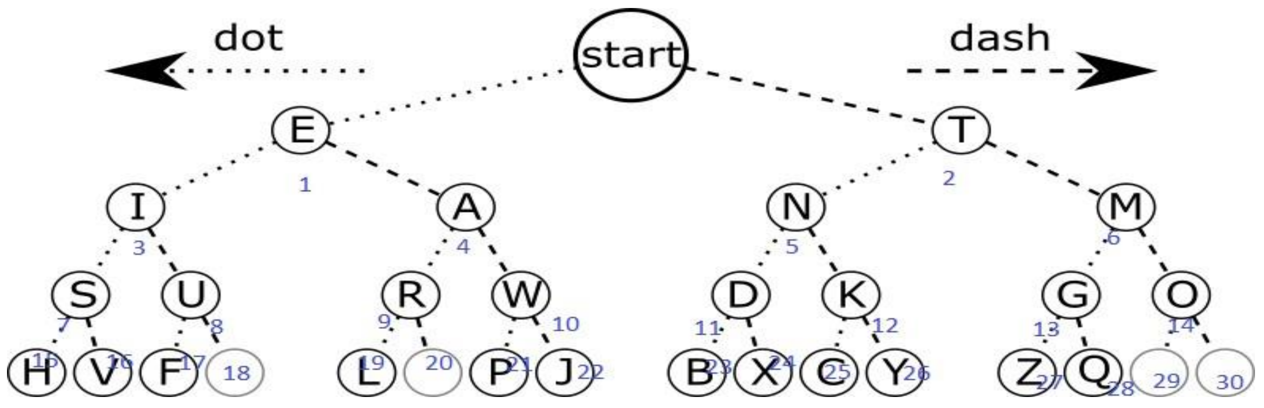
Data structures

My program has Binary search tree for search a character, hash table for character-Morse tree dictionary and file hander for store history.



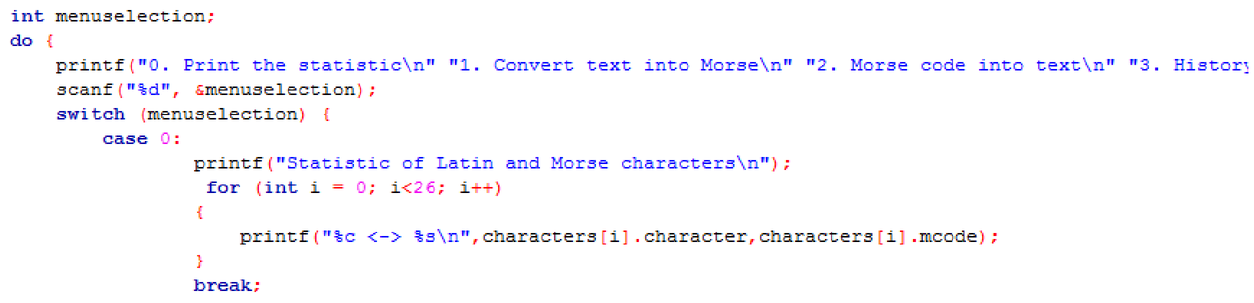
```c
typedef struct MorseTree{
    char character;
    struct MorseTree *left;
    struct MorseTree *right;
} MorseTree;
// Morse code dictionary
// that contains character and Morse code
typedef struct{
    char character;
    char *mcode;
} MorseDict;
```

This program searches a character using the fastest way which is binary search tree. It takes the most 5 steps to search the characters.



Conditions and statements

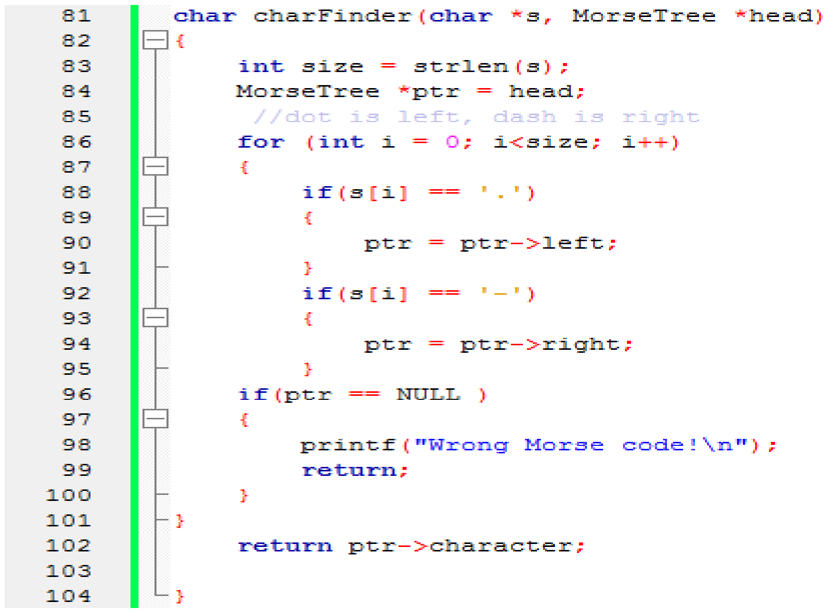
I used do-while loop, switch case to create a menu for my program and other if statements. Menu has 4 cases for user and default case rerun program. Program runs until user chooses the exits selection.



```
int menuselection;
do {
    printf("0. Print the statistic\n" "1. Convert text into Morse\n" "2. Morse code into text\n" "3. Histor
    scanf("%d", &menuselection);
    switch (menuselection) {
        case 0:
                printf("Statistic of Latin and Morse characters\n");
                 for (int i = 0; i<26; i++)
                {
                    printf("%c <-> %s\n",characters[i].character,characters[i].mcode);
                }
                break;
```

Functions

Essential function of my program is the charFinder function which is shown in the below picture.



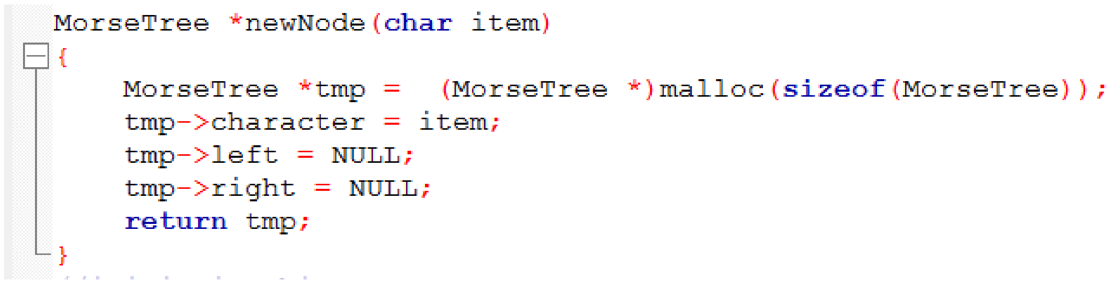
```
char charFinder(char *s, MorseTree *head)
{
     int size = strlen(s);
     MorseTree *ptr = head;
      //dot is left, dash is right
     for (int i = 0; i<size; i++)
     {
          if(s[i] == '.')
          {
               ptr = ptr->left;
          }
          if(s[i] == '-')
          {
               ptr = ptr->right;
          }
     if(ptr == NULL )
     {
          printf("Wrong Morse code!\n");
          return;
     }
}
     return ptr->character;

}
```

It takes a string and a Binary tree pointer and returns a character. It measures length of string and creates binary tree pointer to point a character, then checking component of the string using for loop. If the component of the string is dot pointer points left node of the binary tree or component is dash, it points to the right node. If function can't find the string it returns nothing and prints warning.

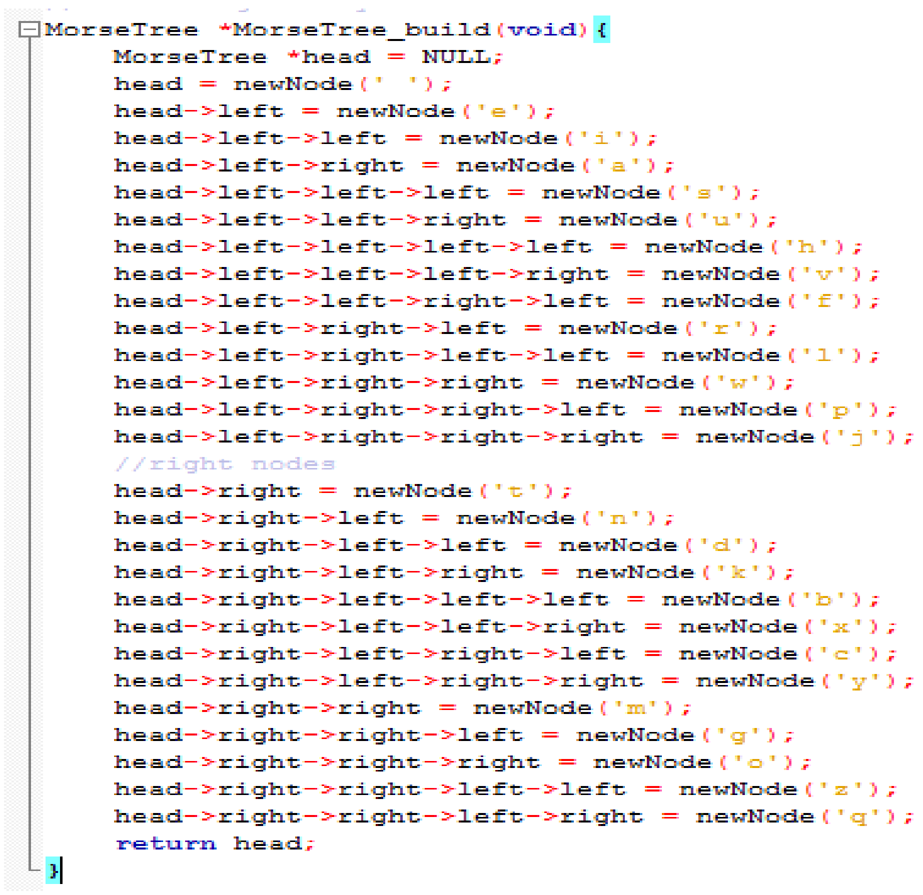
My program has a lot of other functions which are described in the comment section of my main c file.

newNode pointer function, it takes character, creates node consists of character and two null pointer and returns to the main function.



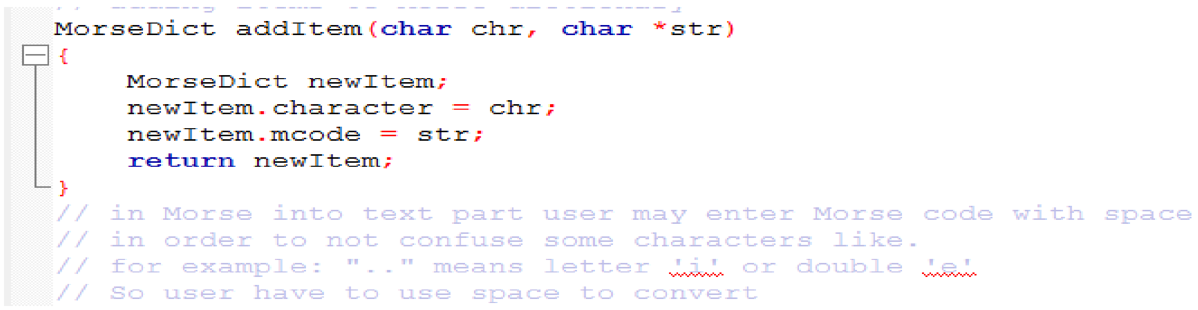
```
MorseTree *newNode(char item)
{
    MorseTree *tmp =  (MorseTree *)malloc(sizeof(MorseTree));
    tmp->character = item;
    tmp->left = NULL;
    tmp->right = NULL;
    return tmp;
}
```

MorseTree_build function, It uses newNode function and creates pointer that points root pointer and it creates binary search tree according to combination of dots, dashes and characters, which is shown in the picture and returns pointer that point root of binary search tree. newNode function makes node with character and this function builds binary search tree. Pointer, named head, is root of BST, points to leaves of characters.



```
MorseTree *MorseTree_build(void){
    MorseTree *head = NULL;
    head = newNode(' ');
    head->left = newNode('e');
    head->left->left = newNode('i');
    head->left->right = newNode('a');
    head->left->left->left = newNode('s');
    head->left->left->right = newNode('u');
    head->left->left->left->left = newNode('h');
    head->left->left->left->right = newNode('v');
    head->left->left->right->left = newNode('f');
    head->left->right->left = newNode('r');
    head->left->right->left->left = newNode('l');
    head->left->right->right = newNode('w');
    head->left->right->right->left = newNode('p');
    head->left->right->right->right = newNode('j');
    //right nodes
    head->right = newNode('t');
    head->right->left = newNode('n');
    head->right->left->left = newNode('d');
    head->right->left->right = newNode('k');
    head->right->left->left->left = newNode('b');
    head->right->left->left->right = newNode('x');
    head->right->left->right->left = newNode('c');
    head->right->left->right->right = newNode('y');
    head->right->right = newNode('m');
    head->right->right->left = newNode('g');
    head->right->right->right = newNode('o');
    head->right->right->left->left = newNode('z');
    head->right->right->left->right = newNode('q');
    return head;
}
```

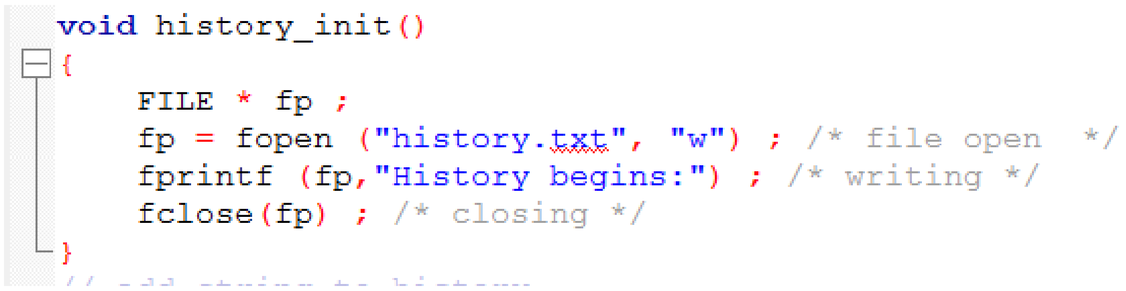
addItem function, it takes character and string, returns newItem node. I used this function to create Morse code – character dictionary. Character is letter and string is morse code and this function makes item for dictionary.



```
MorseDict addItem(char chr, char *str)
{
    MorseDict newItem;
    newItem.character = chr;
    newItem.mcode = str;
    return newItem;
}
// in Morse into text part user may enter Morse code with space
// in order to not confuse some characters like.
// for example: ".." means letter 'i' or double 'e'
// So user have to use space to convert
```

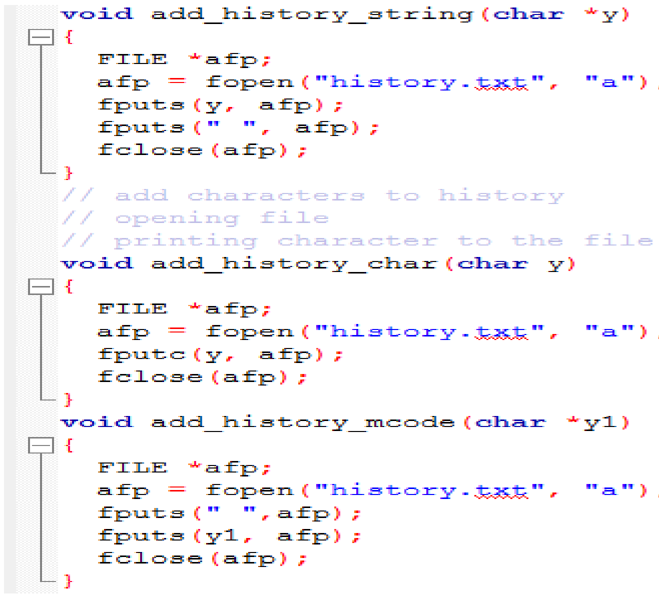
File hander functions

history_init function creates text file to story history.



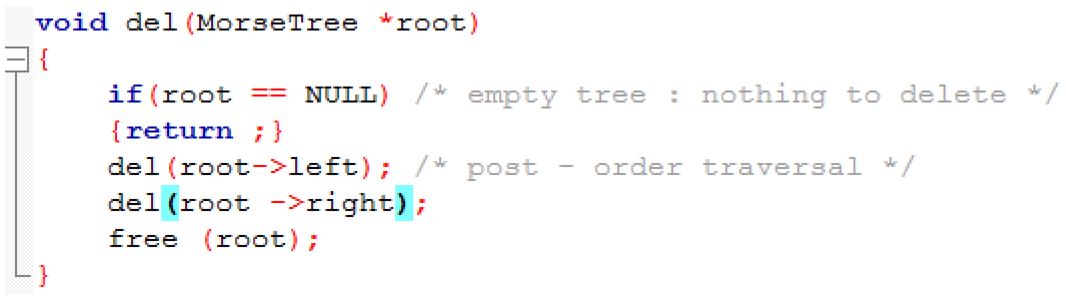
```
void history_init()
{
    FILE * fp ;
    fp = fopen ("history.txt", "w") ; /* file open  */
    fprintf (fp,"History begins:") ; /* writing */
    fclose(fp) ; /* closing */
}
```

Append functions. These functions take character and string from user and writes to history text file.



```
void add_history_string(char *y)
{
    FILE *afp;
    afp = fopen("history.txt", "a");
    fputs(y, afp);
    fputs(" ", afp);
    fclose(afp);
}
// add characters to history
// opening file
// printing character to the file
void add_history_char(char y)
{
    FILE *afp;
    afp = fopen("history.txt", "a");
    fputc(y, afp);
    fclose(afp);
}
void add_history_mcode(char *y1)
{
    FILE *afp;
    afp = fopen("history.txt", "a");
    fputs(" ",afp);
    fputs(y1, afp);
    fclose(afp);
}
```

Del function, it frees the binary search tree.



```
void del(MorseTree *root)
{
    if(root == NULL) /* empty tree : nothing to delete */
    {return ;}
    del(root->left); /* post - order traversal */
    del(root ->right);
    free (root);
}
```

Cases

Case 0: it prints statistics

Case 1: it takes string from user converts to check lowercase letter if strings are capital letter and converts string by characters to morse code, then writes the result to the file.

Case 2: it takes string from user, splits string by spaces using strtok function, converts string to the character and writes the character to the file.

Case 3: uses print_history function which prints everything from the text file.

Case: exits from program