```
/*
 * Name: Zachary Neeley
 * Date: 2/1/19
 * Description: This program will show how to use a symbol take, teaching how use and modify the
information using a linked list.
 *
 *          3) The main structure in the code is a symbol table which allows us to store values.
 *             The table is created by using a linked list.
 *             The fields are first, last, next, size they allow us to build up a structure in memory.
 *
 *          4) The malloc() allows us to dynamically allocate space in memory for the needed values.
 *             We use malloc in the program because this will allow us to store any values given to us.
 *
 */

#include <stdio.h>
/* #include<conio.h> */
#include <malloc.h>
#include <string.h>
#include <stdlib.h>

// init
int size = 0;
void Insert();
void Display();
void Delete();
int Search(char lab[]);
void Modify();

// Setup a struct with 2 arrays label and symbol.
struct SymbTab {
   char label[10], symbol[10];
   int addr;
   struct SymbTab * next;
};

// Create pointers for first and last
struct SymbTab * first, * last;

// Main creates the main for the program and calls the needed functions
void main() {
   // init
   int op, y;
   char la[10];

   // Loop until the number 6 is enter to exit the program
   do {
      printf("\n\tSYMBOL TABLE IMPLEMENTATION\n");
      printf("\n\t1.INSERT\n\t2.DISPLAY\n\t3.DELETE\n\t4.SEARCH\n\t5.MODIFY\n\t6.END\n");
```

```
        printf("\n\tEnter your option : ");
        scanf("%d", & op);

        // See what the input was and call the needed functions
        switch (op) {
        case 1:
           Insert();
           break;
        case 2:
           Display();
           break;
        case 3:
           Delete();
           break;
        case 4:
           printf("\n\tEnter the label to be searched : ");
           scanf("%s", la);
           y = Search(la);
           printf("\n\tSearch Result:");
           if (y == 1)
              printf("\n\tThe label is present in the symbol table\n");
           else
              printf("\n\tThe label is not present in the symbol table\n");
           break;
        case 5:
           Modify();
           break;
        case 6:
           exit(0);
        }
    } while (op < 6);

} // end of main

// Insert will add a new element in our linked list to the end of the list.
void Insert() {
    // init
    int n;
    char l[10];
    printf("\n\tEnter the label : ");
    scanf("%s", l);
    n = Search(l);

    // Check to see if the new label is already in the list if not add it to the end using malloc
    if (n == 1)
       printf("\n\tThe label exists already in the symbol table\n\tDuplicate can.t be inserted");
    else {
       // Create a temp value names p use malloc to create the need space for the new label
       struct SymbTab * p;
```

```
        p = malloc(sizeof(struct SymbTab));
        strcpy(p -> label, l);

        // Ask the user for the symbol and store it in the new label p for symbol
        printf("\n\tEnter the symbol : ");
        scanf("%s", p -> symbol);

        // Ask the user for the address and store it in the new label p for address
        printf("\n\tEnter the address : ");
        scanf("%d", & p -> addr);

        // Set the next pointer to for the new label to null because it is the end of the list
        p -> next = NULL;

        // See if the list is empty and if so then set the first and last to the mamory value of p
        if (size == 0) {
           first = p;
           last = p;

        // if the not the point the last to next and point last to p
        } else {
           last -> next = p;
           last = p;
        }

        // Add 1 to the list size
        size++;
    }
    printf("\n\tLabel inserted\n");
} // end INSERT

// Display will print out the table showing the label, symbol, address in the console
void Display() {
    // init
    int i;
    // Create a new p temp value
    struct SymbTab * p;

    // Set p to first
    p = first;
    printf("\n\tLABEL\t\tSYMBOL\t\tADDRESS\n");

    // Loop throught the list till printing out label, symbol and address
    for (i = 0; i < size; i++) {
        printf("\t%s\t\t%s\t\t%d\n", p -> label, p -> symbol, p -> addr);
        // Go to the next node
        p = p -> next;
    } // end for
} // end Display
```

```
// Search will look through the linked list till the incoming still is found
int Search(char lab[]) {
    // init
    int i, flag = 0;

    // Create a temp struct named p
    struct SymbTab * p;

    // Set p to the first value
    p = first;

    // Loop through the list looking for the lab array contents are found.
    for (i = 0; i < size; i++) {
        // If found set flag to 1
        if (strcmp(p -> label, lab) == 0)
            flag = 1;

        // Go to the next label
        p = p -> next;
    }
    return flag;
} // end Search

// Modify will go through the list nodes and change the needed value
void Modify() {

    // init
    char l[10], nl[10];
    int add, choice, i, s;

    // Create a temp struct p and point p to first
    struct SymbTab * p;
    p = first;

    printf("\n\tWhat do you want to modify?\n");
    printf("\n\t1.Only the label\n\t2.Only the address\n\t3.Both the label and address\n");
    printf("\tEnter your choice : ");
    scanf("%d", & choice);

    // Use the input to ask for the oold label and what to change.
    switch (choice) {
    case 1:
        printf("\n\tEnter the old label : ");
        scanf("%s", l);
        s = Search(l);

        // Check to see if the label is found
        if (s == 0)
```

```
            printf("\n\tLabel not found\n");

        // If the label is found ask for the new input and replace the old label with it
        else {
            printf("\n\tEnter the new label : ");
            scanf("%s", nl);

            // Loop still the end of the string adding the new label in p
            for (i = 0; i < size; i++) {
                if (strcmp(p -> label, l) == 0)
                    strcpy(p -> label, nl);

                // Move p to the next value
                p = p -> next;
            }
            printf("\n\tAfter Modification:\n");
            Display();
        } // end else
        break;

    // The user is wanting to change the address
    case 2:
        printf("\n\tEnter the label where the address is to be modified : ");
        scanf("%s", l);
        s = Search(l);

        // IF the label is found replace the address in the node with the user input
        if (s == 0)
            printf("\n\tLabel not found\n");
        else {
            // Get the new address from the user and replace the address in the current node
            printf("\n\tEnter the new address : ");
            scanf("%d", & add);

            // Replace the address with the incoming still
            for (i = 0; i < size; i++) {
                if (strcmp(p -> label, l) == 0)
                    p -> addr = add;
                p = p -> next;
            } // end for
            printf("\n\tAfter Modification:\n");
            Display();
        }
        break;
    // If the user is wanting to replace the entire node instead of just the label or address.
    case 3:
        printf("\n\tEnter the old label : ");
        scanf("%s", l);
        s = Search(l);
```

```
        // If the label is found get the input from the user and replace the needed material
        if (s == 0)
            printf("\n\tLabel not found\n");

        else {
            printf("\n\tEnter the new label : ");
            scanf("%s", nl);
            printf("\n\tEnter the new address : ");
            scanf("%d", & add);

            // Replace the values with the new input
            for (i = 0; i < size; i++) {
                if (strcmp(p -> label, l) == 0) {
                    strcpy(p -> label, nl);
                    p -> addr = add;
                }
                p = p -> next;
            }
            printf("\n\tAfter Modification:\n");
            Display();
        }
        break;
    } // end switch
} // end Modify

// This will remove a node from the list
void Delete() {
    // init
    int a;
    char l[10];

    // Create 2 new temp structs
    struct SymbTab * p, * q;
    p = first;

    // Get input
    printf("\n\tEnter the label to be deleted : ");
    scanf("%s", l);
    a = Search(l);

    // See if the label is found
    if (a == 0)
        printf("\n\tLabel not found\n");
    else {
        // If the label is found, if so take the label before the marked removed one to point to the next node
after the removed node.
        if (strcmp(first -> label, l) == 0)
            first = first -> next;
```

```
        else if (strcmp(last -> label, l) == 0) {
            q = p -> next;
            // Loop till the label is found
            while (strcmp(q -> label, l) != 0) {
                p = p -> next;
                q = q -> next;
            }
            p -> next = NULL;
            last = p;
        } else {
            // Point the next node to the correct node
            q = p -> next;
            while (strcmp(q -> label, l) != 0) {
                p = p -> next;
                q = q -> next;
            }
            p -> next = q -> next;
        }

        // Remove 1 from size
        size--;
        printf("\n\tAfter Deletion:\n");
        Display();
    } // end else
} // end delete
```