Jeffery Lansford
Febuary 1, 2019
Lab 3
To understand how the symbol table works and how it is impleatment.

symtable.c:

```
/* Jeffrey Lansford
 * CS 370
 * Lab 3
 * January 29, 2019
 * C program that simualtes a symbol table with that can insert, display, delete, search, and modify
 * Added indentation and comments into program that was pulled from http://forgetcode.com/C/101-Symbol-table
 */

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

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

struct SymbTab
{
   char label[10],symbol[10];
   int addr;
   struct SymbTab *next;

};

struct SymbTab *first,*last;

void main()
{
   int op,y;
   char la[10];
   do
   {
      // Reads User Input and uses the following five functions according to that input
      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);
```

```
        switch(op)
        {
           case 1:
              Insert();
              break;
           case 2:
              Display();
              break;
           case 3:
              Delete();
              break;
           case 4:
              // gets user input for label to be searched
              printf("\n\tEnter the label to be searched : ");
              scanf("%s",la);
              // searchs label in table prints results
              y=Search(la);
              printf("\n\tSearch Result:");
              // symbol found if 1
              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 */

/*
 * Inserts a new label into the Symbol Table if it does not already exist in the table
 * no return
 */
void Insert()
{
    // get new label name from user
    int n;
    char l[10];
    printf("\n\tEnter the label : ");
    scanf("%s",l);
    // check if new label alerady exit in symbol table
    n=Search(l);
    // label is found in table
    if(n==1)
```

```
        printf("\n\tThe label exists already in the symbol table\n\tDuplicate can.t be inserted");
    // label not found
    else
    {
        // make new node SymbTab and set data
        struct SymbTab *p;
        p=malloc(sizeof(struct SymbTab));
        strcpy(p->label,l);
        printf("\n\tEnter the symbol : ");
        scanf("%s",p->symbol);
        printf("\n\tEnter the address : ");
        scanf("%d",&p->addr);
        p->next=NULL;
        // sets first and last if new node is the only node.
        if(size==0)
        {
            first=p;
            last=p;
        }
        // more than zero nodes
        else
        {
            // inserts new node always last
            last->next=p;
            last=p;
        }
        size++;
    }
    printf("\n\tLabel inserted\n");
}// end Insert

/*
 * Displays all current labels in Symbol table
 * no return
 */
void Display()
{
    // starts at first then goes through each node through *next of each node
    int i;
    struct SymbTab *p;
    p=first;
    printf("\n\tLABEL\t\tSYMBOL\t\tADDRESS\n");
    // goes through syboml table and prints each label
    for(i=0;i<size;i++)
    {
        printf("\t%s\t\t%s\t\t%d\n",p->label,p->symbol,p->addr);
        p=p->next;
    }
}// end Display
```

```
/*
 * Search for a spefic label in the symbol table
 * returns 1 if founded, 0 if not
 */
int Search(char lab[])
{
   // goes through each node and compares the label vairable of each node to lab
   int i,flag=0;
   struct SymbTab *p;
   p=first;
   // goes through symbol table
   for(i=0;i<size;i++)
   {
      // compares label in sybmol table and lab
      if(strcmp(p->label,lab)==0)
         flag=1;
      p=p->next;
   }
   return flag;
}// end Search

/*
 * modifys a current label in the symbol table to change label or address
 * return void
 */
void Modify()
{
   // sets up for user input and temp pointer
   char l[10],nl[10];
   int add,choice,i,s;
   struct SymbTab *p;
   p=first;
   // gets User Input for what to change
   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);
   // changes either label, address, or both
   switch(choice)
   {
      case 1:
         // gets label to get from user
         printf("\n\tEnter the old label : ");
         scanf("%s",l);
         // label must be in Symbol table
         s=Search(l);
         // not in table
         if(s==0)
            printf("\n\tLabel not found\n");
```

```
    // is in table
    else
    {
       // gets new label
       printf("\n\tEnter the new label : ");
       scanf("%s",nl);
       // goes through table to the right label
       for(i=0;i<size;i++)
       {
          // compare the two strings to make sure you have right label
          // then copies string to the node's label
          if(strcmp(p->label,l)==0)
             strcpy(p->label,nl);
          p=p->next;
       }
       // display table to show change
       printf("\n\tAfter Modification:\n");
       Display();
    }
    break;

case 2:
    // gets label user wants to change
    printf("\n\tEnter the label where the address is to be modified : ");
    scanf("%s",l);
    // finds that label and makes sure that it is in the symbol table
    s=Search(l);
    // not in table
    if(s==0)
       printf("\n\tLabel not found\n");
    // is in table
    else
    {
       // gets new address
       printf("\n\tEnter the new address : ");
       scanf("%d",&add);
       // goes through table to the right label
       for(i=0;i<size;i++)
       {
          // compares strings to make sure that you have right label
          // then changes node's address to the new adress
          if(strcmp(p->label,l)==0)
             p->addr=add;
          p=p->next;
       }
       // display table to show change
       printf("\n\tAfter Modification:\n");
       Display();
    }
```

```
            break;

        case 3:
            // gets label to change from user
            printf("\n\tEnter the old label : ");
            scanf("%s",l);
            // checks if label is in Symbol Table
            s=Search(l);
            // not in table
            if(s==0)
                printf("\n\tLabel not found\n");
            // is in table
            else
            {
                // gets new label
                printf("\n\tEnter the new label : ");
                scanf("%s",nl);
                // gets new address
                printf("\n\tEnter the new address : ");
                scanf("%d",&add);
                // goes through table to the right label
                for(i=0;i<size;i++)
                {
                    // compares strings to make sure you have right label
                    if(strcmp(p->label,l)==0)
                    {
                        // copies string into node's label and changes address of node
                        strcpy(p->label,nl);
                        p->addr=add;
                    }
                    p=p->next;
                }
                // display table to show change
                printf("\n\tAfter Modification:\n");
                Display();
            }
            break;
    }
}// end Modify
/*
 * Deletes a node that the user selects
 * no return value
 */
void Delete()
{
    // sets up for user input and list transversal
    int a;
    char l[10];
    struct SymbTab *p,*q;
```

```
    p=first;
    printf("\n\tEnter the label to be deleted : ");
    scanf("%s",l);
    // checks if user input is in symbol table
    a=Search(l);
    // not in table
    if(a==0)
        printf("\n\tLabel not found\n");
    // is in symbol table
    else
    {
        // if node marked for deletion is first, then set first to the next node
        if(strcmp(first->label,l)==0)
            first=first->next;
        // if node marked for deletion is last, then set previous node to last
        else if(strcmp(last->label,l)==0)
        {
            // gets next node after p
            q=p->next;
            // get to label that user wants to delete
            while(strcmp(q->label,l)!=0)
            {
                p=p->next;
                q=q->next;
            }
            // sets p's next to null and make p last
            p->next=NULL;
            last=p;
        }
        // node is in the middle
        else
        {
            // gets next node after p
            q=p->next;
            // get label that user wants to delete
            while(strcmp(q->label,l)!=0)
            {
                p=p->next;
                q=q->next;
            }
            // sets p to the node that is next to q
            p->next=q->next;
        }
        // reduce size and display change
        size--;
        printf("\n\tAfter Deletion:\n");
        Display();
    }
}// end Delete
```

Makefile :

```
# Jeffrey Lansford
# Lab3
# Febuary 1, 2019
# compiles C program and makes exucuable lab3
all:
	gcc -o lab3 symtable.c
```

Show Insert

```
SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 1

Enter the label : x

Enter the symbol : int

Enter the address : 1

Label inserted

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 2

LABEL           SYMBOL          ADDRESS
x               int             1

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option :
```

Insert second label

```
SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 1

Enter the label : y

Enter the symbol : int

Enter the address : 2

Label inserted

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 2

LABEL           SYMBOL          ADDRESS
x               int             1
y               int             2
```

Search for y

```
SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 2

LABEL           SYMBOL          ADDRESS
x               int             1
y               int             2

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 4

Enter the label to be searched : y

Search Result:
The label is present in the symbol table
```

Delete x in table

```
LABEL           SYMBOL          ADDRESS
x               int             1
y               int             2

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 3

Enter the label to be deleted : x

After Deletion:

LABEL           SYMBOL          ADDRESS
y               int             2

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 2

LABEL           SYMBOL          ADDRESS
y               int             2
```

modify y to change label to z and address to 123

```
LABEL           SYMBOL          ADDRESS
y               int             2

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 5

What do you want to modify?

1.Only the label
2.Only the address
3.Both the label and address
Enter your choice : 3

Enter the old label : y

Enter the new label : z

Enter the new address : 123

After Modification:

LABEL           SYMBOL          ADDRESS
z               int             123

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 2

LABEL           SYMBOL          ADDRESS
z               int             123
```

The main structure for this code is SymTab structure. It is a linked list that each node holds data for a char array for label, another char arrray for symbol, an int for address, and a SymTab pointer that points to the next node or null. The structure is first built with the creation of a new node, then we set its data and its next pointer to null, and set first and last pointers, that are created outside of the structe to keep track of where we are, to that new node. When adding new nodes, we set the last's *next to the new node and set the last to that new node.

Malloc() allcoates memory in the heap and reurns a pointer to that memory. We use malloc since we cannot store our structure SymTab in any other segmentation in memory. We cannot store in the code, data, or the run-time stack, so we must store in the heap which we use malloc for.