DS4TALKER

LAB 9 SECTION 1

SUBMITTED BY:

ETHAN GRIESMAN

SUBMISSION DATE:

11/29/2022

Problem:

The objective for this lab was to acquire skills/proficiency with top-down program design, work with two dimensional arrays and string manipulation within the C programming language, as well as further developing skills using loops.

Analysis:

The problem states that the user must be able to do certain functions with the program. It should be able to do these things multiple times in a row and allow for the user to add words with or without a space, or to delete the previous action. It must also use lncurses. I copied over the draw character function from lab 8, to draw my word pointer. I also created a function to clear an array of a given length.

Design

I went about solving this problem by using a function from my previous lab, drawCharacter. In addition to that, I wrote another function, clearAr, that sets any given array to 0, with a null character. These functions were called when necessary, and the main function used a series of if statements to determine which button was pushed. I used a tolerance comparison on each if statement to check if a certain time had passed before allowing the code inside to execute again. This helped to prevent unwanted calls and prevented the cursor from moving very quickly across the screen. I used lncurse to display the sentence, and the list of words in five columns.

RESULT:
Demo'd to TA 11/9/2022 and 11/16/2022

Testing

To verify my program worked properly, I compiled and ran the program several times in a lab. At first, I worked to have the words be displayed in the group correctly, then I tested the controller movement. When it moved too quickly, I remembered from lab8, adding a buffer to allow for the user to have more control when moving the avatar. I used a similar concept from moving this pointer. Adding words to the sentence with or without a space went smoothly, but removing words was more difficult. I ended up using a 2D array to store each change to the sentence, and if a user wanted to remove the last thing, the new sentence was the previous index of the array. This worked well for what it needed to do.

Comments

This lab was very fun for me, and I spent a lot of time tinkering with different conditions to make the game more interesting. When I first tried to stop the avatar when a barrier was encountered, I was only getting a message displaying “You have lost” at the specific line where the barrier was located, and not at the bottom of the screen like the “You have won” message. After struggling with this for a bit, I realized that I had to include in the endwin(); function, as that closed the window and was why our winning message appeared at the bottom. Doing this yielded the expected result.

Question 1:

To read words from the file, I used a method similar to what Dr. Daniels taught in class. I opened a file and while it wasn’t the end of the file, I trimmed the whitespace from the word, and added it to my array wordlist. Once it finished, the function would close the file and return the number of words from the file.

Question 2:

I keep track of the word selected on the screen with math. Because the words are spaced out 15 characters, and are in rows of five, I was able to calculate the index of the wordlist is the location of the pointer, (px/15) + (py*5). This didn’t work at first as I was OBOBing, so I added -1 to the end, and it worked perfectly from then on. I think this interface is somewhat reasonable, obviously there could be a better interface, for example requiring less movement from the user if it’s meant for somebody with a disability, but it works.

```
// Lab 9 DS4Talker Skeleton Code

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <ncurses/ncurses.h>
#define MAXWORDS 100
#define WORDLEN 11
#define DEBUG 1   // set to 0 to disable debug output

// reads words from the file
// into wl and trims the whitespace off of the end of each word
// DO NOT MODIFY THIS Prototype
int readWords(char* wl[MAXWORDS], char* filename);

//modifies s to trim white space off the right side
// DO NOT MODIFY THIS Prototype
void trimws(char* s) ;

int main(int argc, char* argv[]) {
       char* wordlist[MAXWORDS];
       int wordCount;
       int i;
       wordCount = readWords(wordlist, argv[1]);

for(int z = 0; z < MAXWORDS; z++){
       scanf("%s", wordlist[z]);
}
       if (DEBUG) {
              printf("Read %d words from %s \n", wordCount,
argv[1]);
              // add code to print the words to the screen here for
part 1

              for (i = 0; i < wordCount; i++) {
                     printf("%s\n", wordlist[i]);
              }
       }

// most of your code for part 2 goes here. Don't forget to include
the ncurses library

    return 0;
       }

int readWords(char* wl[MAXWORDS], char* filename) {
       char word[WORDLEN];
       int count = 0;
       FILE* f;
```

```
        f = fopen(filename, "r");


        while(fscanf(f, "%s", word) == 1) {
                trimws(word);
                wl[count] = (char*)malloc(sizeof(char)*(strlen(word))
+ 1);
                strcpy(wl[count], word);
                count++;
        }
        fclose(f);
        return count;
}

void trimws(char* s) {
        int index = 0;
        for(int i = 0; i < strlen(s); i++){
                while ((s[index] != ' ') && (s[index] != '\0')) {
                index++;
        }
        }
        s[index] = '\0';
}
```

```
// Lab 9 DS4Talker Skeleton Code

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <ncurses/ncurses.h>
#define MAXWORDS 100
#define WORDLEN 11
#define DEBUG 0   // set to 0 to disable debug output
#define MAXONEROW 5

// reads words from the file
// into wl and trims the whitespace off of the end of each word
// DO NOT MODIFY THIS Prototype
int readWords(char* wl[MAXWORDS], char* filename);

//modifies s to trim white space off the right side
// DO NOT MODIFY THIS Prototype
void trimws(char* s);
void draw_character(int y, int x, char use);
void rowPlace(int index, int *row, int *column);

//----//

int main(int argc, char* argv[]) {
       char* wordlist[MAXWORDS];
       int wordCount, rowCount, extra;
       int i, j;
       int index = 0;
       wordCount = readWords(wordlist, argv[1]);
       rowCount = wordCount / 5;
       extra = wordCount % 5;
       int totalRows = (rowCount + extra);
       int row, column;

       if (DEBUG) {
              // add code to print the words to the screen here for
part 1
       }

       // most of your code for part 2 goes here. Don't forget to
include the ncurses library
       initscr();
       int currWord = 0;
       for (i = 0; i < wordCount; i++) {
              for(j=0;j<5;j++) {
                     if(currWord < wordCount){
                     rowPlace(i, &row, &column);
                     mvprintw(i, j*15, "%15s",
wordlist[currWord++]);
                     refresh();
                     }
```

```
                }
                if (i > wordCount) {
                        break;
                }

        }

        int x = 15;
        int y = 0;
        draw_character(y, x, '*');

        int k, num = 0;
        int time, bT, bO, bX, bS, j1_PRESS, j2_PRESS, j1_x, j1_y,
j2_x, j2_y;
        int NA1, NA2, NA3, NA4, NA5, NA6;
        int timePrev = 0;
        int printWord = 0;
        int stringLengths[80];
        char *userString = malloc(sizeof(char) * 10);
        char *userSentence = malloc(sizeof(char) * 80);


        while (1) {
                scanf(" %d, %d,%d,%d,%d, %d,%d,%d,%d,%d,%d,%d,%d, %d,
%d, %d, %d", &time, &bT, &bO, &bX, &bS, &j1_PRESS, &j2_PRESS, &NA1,
&NA2, &NA3, &NA4, &NA5, &NA6, &j1_x, &j1_y, &j2_x, &j2_y);


                if ((time - timePrev) >= 150) {

                        refresh();
                        if ((j1_x < -60) && (!(x - 15 <= 0))) {
                                draw_character(y, x, ' ');
                                x = x - 15;
                                draw_character(y, x, '*');
                                printWord--;
                        }
                        if ((j1_x > 60) && (!(x + 15 > 75))) {
                                draw_character(y, x, ' ');
                                x = x + 15;
                                draw_character(y, x, '*');
                                printWord++;
                        }
                        if ((j1_y < -60) && (!(y - 1 < 0))) {
                                draw_character(y, x, ' ');
                                y = y - 1;
                                draw_character(y, x, '*');
                                printWord = printWord - 5;
                        }
                        if ((j1_y > 60) && (!(y + 1 >= totalRows))) {
                                draw_character(y, x, ' ');
                                y = y + 1;
                                draw_character(y, x, '*');
                                printWord = printWord + 5;
                        }

                        if (bT) {
                                if (strlen(userSentence) < 80) {
                                        strcpy(userString, " ");
                                        strcat(userString,
wordlist[printWord]);
                                        stringLengths[num] =
strlen(userString);
                                        strcat(userSentence,
userString);
                                        mvprintw(17, 5, "%s",
userSentence);
                                        num++;
                                }

                        }

                        if (bS) {

                                if(strlen(userSentence) < 80) {
                                        strcpy(userString,
wordlist[printWord]);
                                    stringLengths[num] =
strlen(userString);
                                    strcat(userSentence, userString);
                                    mvprintw(17, 5, "%s",
userSentence);
                                    num++;
                                }
                        }


                        if (bX) {

                                        int length;
                                        length = strlen(userSentence);
                                        if (!(length == 0)) {
                                        for (k = length; k > length -
stringLengths[num - 1]; k--) {
                                                userSentence[k] = '\0';
                                                mvprintw(17, k + 4, "
");
                                        }
                                        if(userSentence[k-1] == '0'){
                                                k--;
                                        }
                                        userSentence[k] = '\0';
                                        mvprintw(17, 5, "%s",
userSentence);
                                        num--;
                                }
                        }
                        if (j2_PRESS) {
                                int n = 0;
                                for(n = 0; n < 80; n++) {
                                        userSentence[n] = ' ';
                                        mvprintw(17, n + 5, " ");
                                }
                                userSentence[0] = '\0';
                                num = 0;
                        }

                        timePrev = time;
                }

        }
        endwin();
}
int readWords(char* wl[MAXWORDS], char* filename) {
        char word[WORDLEN];
        int count = 0;
        FILE* f;
        f = fopen(filename, "r");


        while (fscanf(f, "%s", word) == 1) {
                trimws(word);
                wl[count] = (char*)malloc(sizeof(char)*(strlen(word)
+ 1));
                strcpy(wl[count], word);
                count++;
        }
        fclose(f);
        return count;
}

void trimws(char* s) {
        int index = 0;
        while ((s[index] != ' ') && (s[index] != '\0')) {
                index++;
        }
        s[index] = '\0';
}

void draw_character(int y, int x, char use) {
        mvaddch(y, x, use);
        refresh();
}

void rowPlace(int index, int *row, int *column) {
        *row = index / 5;

        *column = (((index % 5) * (WORDLEN + 4)) + 1);
}




// Lab 9 DS4Talker Skeleton Code

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <ncurses/ncurses.h>
#define MAXWORDS 100
#define WORDLEN 11
#define DEBUG 0   // set to 0 to disable debug output
#define MAXONEROW 5

// reads words from the file
// into wl and trims the whitespace off of the end of each word
// DO NOT MODIFY THIS Prototype
int readWords(char* wl[MAXWORDS], char* filename);

//modifies s to trim white space off the right side
// DO NOT MODIFY THIS Prototype
void trimws(char* s);
void draw_character(int y, int x, char use);
void rowPlace(int index, int *row, int *column);

//----//

int main(int argc, char* argv[]) {
        char* wordlist[MAXWORDS];
        int wordCount, rowCount, extra;
        int i, j;
        int index = 0;
        wordCount = readWords(wordlist, argv[1]);
        rowCount = wordCount / 5;
        extra = wordCount % 5;
        int totalRows = (rowCount + extra);
        int row, column;

        if (DEBUG) {
               // add code to print the words to the screen here for
part 1
        }

        // most of your code for part 2 goes here. Don't forget to
include the ncurses library
        initscr();
        int currWord = 0;
```

```
	for (i = 0; i < wordCount; i++) {
		for(j=0;j<5;j++) {
			if(currWord < wordCount){
			rowPlace(i, &row, &column);
			mvprintw(i, j*15, "%15s",
wordlist[currWord++]);
			refresh();
			}
		}
		if (i > wordCount) {
			break;
		}

	}

	int x = 15;
	int y = 0;
	draw_character(y, x, '*');

	int k, num = 0;
	int time, bT, bO, bX, bS, j1_PRESS, j2_PRESS, j1_x, j1_y,
j2_x, j2_y;
	int NA1, NA2, NA3, NA4, NA5, NA6;
	int timePrev = 0;
	int printWord = 0;
	int stringLengths[80];
	char *userString = malloc(sizeof(char) * 10);
	char *userSentence = malloc(sizeof(char) * 80);


	while (1) {
		scanf(" %d, %d,%d,%d,%d, %d,%d,%d,%d,%d,%d,%d,%d, %d,
%d, %d, %d", &time, &bT, &bO, &bX, &bS, &j1_PRESS, &j2_PRESS, &NA1,
&NA2, &NA3, &NA4, &NA5, &NA6, &j1_x, &j1_y, &j2_x, &j2_y);


		if ((time - timePrev) >= 150) {

			refresh();
			if ((j1_x < -60) && (!(x - 15 <= 0))) {
				draw_character(y, x, ' ');
				x = x - 15;
				draw_character(y, x, '*');
				printWord--;
			}
			if ((j1_x > 60) && (!(x + 15 > 75))) {
				draw_character(y, x, ' ');
				x = x + 15;
				draw_character(y, x, '*');
				printWord++;
			}
			if ((j1_y < -60) && (!(y - 1 < 0))) {
				draw_character(y, x, ' ');
```

```
                                y = y - 1;
                                draw_character(y, x, '*');
                                printWord = printWord - 5;
                        }
                        if ((j1_y > 60) && (!(y + 1 >= totalRows))) {
                                draw_character(y, x, ' ');
                                y = y + 1;
                                draw_character(y, x, '*');
                                printWord = printWord + 5;
                        }

                        if (bT) {
                                if (strlen(userSentence) < 80) {
                                        strcpy(userString, " ");
                                        strcat(userString,
wordlist[printWord]);
                                        stringLengths[num] =
strlen(userString);
                                        strcat(userSentence,
userString);
                                        mvprintw(17, 5, "%s",
userSentence);
                                        num++;
                                }

                        }

                        if (bS) {

                                if(strlen(userSentence) < 80) {
                                        strcpy(userString,
wordlist[printWord]);
                                    stringLengths[num] =
strlen(userString);
                                    strcat(userSentence, userString);
                                    mvprintw(17, 5, "%s",
userSentence);
                                    num++;
                                }
                        }


                        if (bX) {

                                        int length;
                                        length = strlen(userSentence);
                                        if (!(length == 0)) {
                                        for (k = length; k > length -
stringLengths[num - 1]; k--) {
                                                userSentence[k] = '\0';
                                                mvprintw(17, k + 4, "
");
                                        }
                                        if(userSentence[k-1] == '0'){
                                                k--;
                                        }
                                        userSentence[k] = '\0';
                                        mvprintw(17, 5, "%s",
userSentence);

                                        num--;
                                }



                        }
                        if (j2_PRESS) {
                                int n = 0;
                                for(n = 0; n < 80; n++) {
                                        userSentence[n] = ' ';
                                        mvprintw(17, n + 5, " ");
                                }
                                userSentence[0] = '\0';
                                num = 0;
                        }

                        timePrev = time;
                }

        }
        endwin();
}
int readWords(char* wl[MAXWORDS], char* filename) {
        char word[WORDLEN];
        int count = 0;
        FILE* f;
        f = fopen(filename, "r");


        while (fscanf(f, "%s", word) == 1) {
                trimws(word);
                wl[count] = (char*)malloc(sizeof(char)*(strlen(word)
+ 1));
                strcpy(wl[count], word);
                count++;
        }
        fclose(f);
        return count;
}

void trimws(char* s) {
        int index = 0;
        while ((s[index] != ' ') && (s[index] != '\0')) {
                index++;
        }
        s[index] = '\0';
}
```

```
void draw_character(int y, int x, char use) {
	mvaddch(y, x, use);
	refresh();
}

void rowPlace(int index, int *row, int *column) {

	*row = index / 5;

	*column = (((index % 5) * (WORDLEN + 4)) + 1);
}
```