**Project 1: Hangman**

By: Hamza Khan

Class: CSC 11

Date Due: November 3, 2014

Teacher: Dr. Lehr

Table of Contents

1. Introduction………………………………………………………………………………………..3
2. Summary………………………………………………………………………………………….....3
3. Description………………………………………………………………………………………....3
4. Sample Inputs and Outputs…………………………………………………………….3
5. Flowchart……………………………………………………………………………………...4
6. Variables……………………………………………………………………………………….4
7. Reference……………………………………………………………………………………………5
8. Code…………………………………………………………………………………………………...5

I. Introduction:

The game I created for the first project is a game of hangman. I came up the idea while my brother and me were playing a came of hangman and from that I got the idea for to create the game. The reason I chose the word I did is because I was talking to friend and we were joking about an episode of The Walking Dead so that is why the word is the way it is. The amount of time it took me to create the project was about three to four days because I tried doing it piece by piece on a daily basis and minimizes the workload.

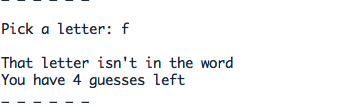
II. Summary:

|  |  |
| --- | --- |
| Number of actual code | 133 |
| Number of blanks | 37 |
| Number of comment lines | 6 |
| Total | 175 |

III. Description:

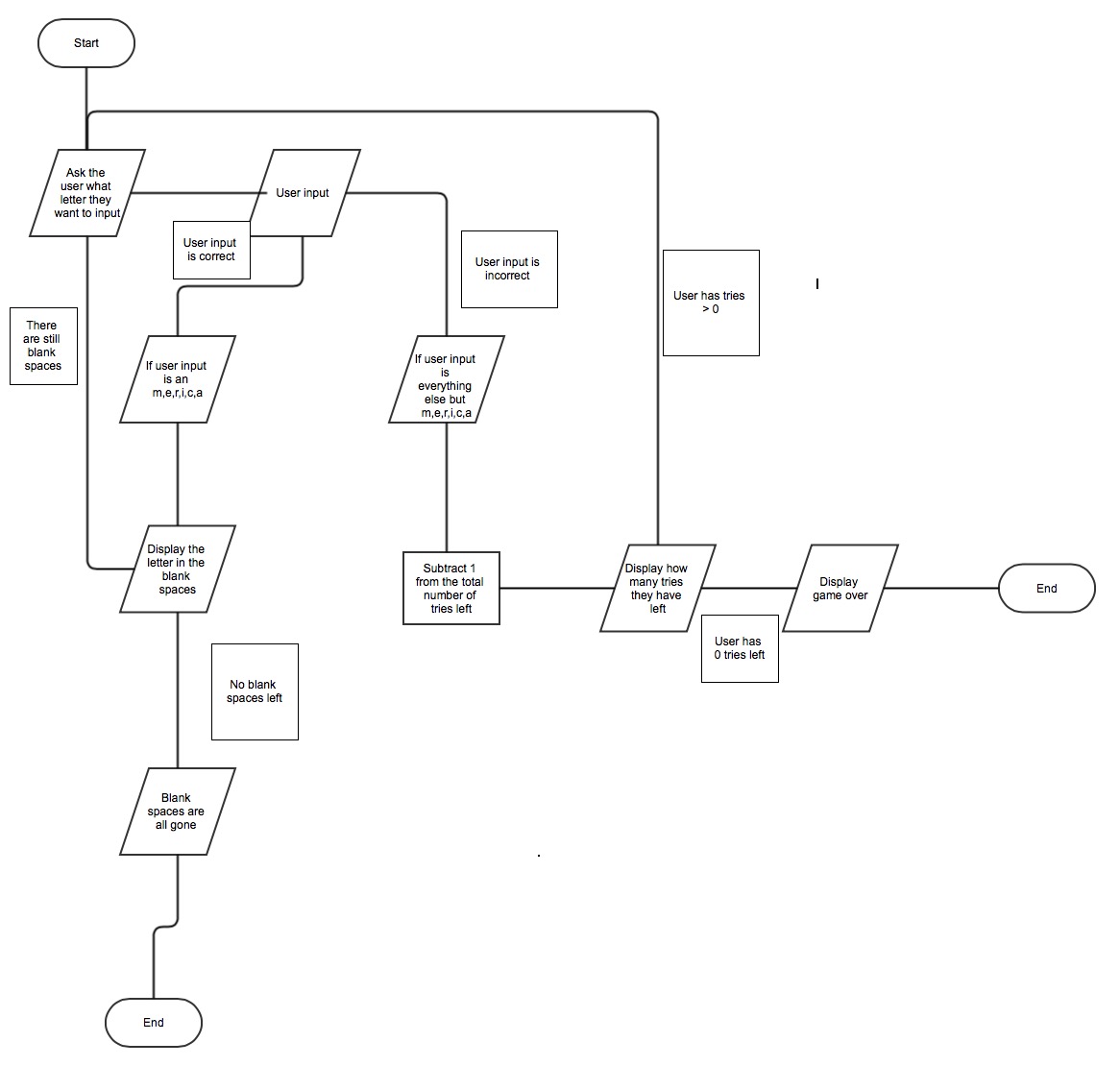
A. Sample Input and outputs

Input and output if you enter in a wrong letter



Input and output if you enter in the right letter



B. Flowchart

C. Variables

a. Branching

Throughout the project I used branching to go different parts of the program. The purpose of the branch is to allow the program to access different parts of the program and allow it to read and collect data from them if it meets a certain requirement. I used branching from lines 76 – 92 to allow the system to read data from it.

b. Addition, Subtraction

I also used the addition and subtract to allow the project to do basic math. In the program it has a number of tries system and the subtract was used to allow the game to remove a try if the user enters in an invalid or incorrect character. I used this in lines 128, 144, 135, 121, 114, and 107.

IV. Reference:

For this project all tfhe code is originally mine. Everything in it is code that you have taught us how to use. So with that information I was able to create the game. If you want to be considered a reference, because you kind of are.

V. Code:

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.data

message1:

.asciz "%c "

message2:

.asciz "\n\nPick a letter: "

message3:

.asciz "\nThat letter isn't in the word. Try again\n"

message4:

.asciz "You have %d guesses left\n"

message5:

.asciz "\nYou already used that letter. Try again\n"

format:

.asciz " %c"

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.text

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.global main

main:

STR lr, [sp,#-4]! @ Push lr onto the top of the stack

SUB sp, sp, #4 @ Make room for one 4 byte integer in the stack

MOV R4, #95 @Cover word with "\_" ascii code = 95

MOV R5, #95

MOV R6, #95

MOV R7, #95

MOV R8, #95

MOV R9, #95

MOV R10, #6 @Length of the word

MOV R11, #6 @Guesses left

loop: @Print the word one character at a time (no arrays yet)

MOV R1, R4

LDR R0, address\_of\_message1 @Set &message1 as the first parameter of printf

BL printf @Call printf

MOV R1, R5

LDR R0, address\_of\_message1

BL printf

MOV R1, R6

LDR R0, address\_of\_message1

BL printf

MOV R1, R7

LDR R0, address\_of\_message1

BL printf

MOV R1, R8

LDR R0, address\_of\_message1

BL printf

MOV R1, R9

LDR R0, address\_of\_message1

BL printf

LDR R0, address\_of\_message2

bl printf

LDR R0, address\_of\_format @Set &format as the first parameter of scanf

MOV R1, sp @Set the top of the stack as the second parameter of scanf

BL scanf @Call scanf

LDR R1, [sp] @Load character read into r1

MOV R12, R1

CMP R12, #109 @Letter c, ascii code = 109

BEQ letter\_m

CMP R12, #101 @Letter h, ascii code = 101

BEQ letter\_e

CMP R12, #114 @Letter i, ascii code = 114

BEQ letter\_r

CMP R12, #105 @Letter n, ascii code = 105

BEQ letter\_i

CMP R12, #99 @Letter c, ascci code = 99

BEQ letter\_c

CMP R12, #97 @Letter a, ascii code = 97

BEQ letter\_a

b wrong @If none of this, the letter is not in the word

letter\_m: @Do this for every single character in the word

CMP R12, R4 @Check if letter is already used in the word

BEQ repeated

MOV R4, R12 @Replace '\_' with the correct letter

SUB R10, R10, #1 @Decrease size of word by one (letter is correct)

b test @Test the condition to repeat the loop

letter\_e:

CMP R12, R5

BEQ repeated

MOV R5, R12

SUB R10, R10, #1

b test

letter\_r:

CMP R12, R6

BEQ repeated

MOV R6, R12

SUB R10, R10, #1

b test

letter\_i:

CMP R12, R7

BEQ repeated

MOV R7, R12

SUB R10, R10, #1

b test

letter\_c:

CMP R12, R8

BEQ repeated

MOV R8, R12

SUB R10, R10, #1

b test

letter\_a:

CMP R12, R9

BEQ repeated

MOV R9, R12

SUB R10, R10, #1

b test

repeated: @Letter already used

LDR R0, address\_of\_message5

BL printf

b loop

wrong:

SUB R11, R11, #1 @If wrong decrease guesses left by one

CMP R11, #0 @If gueses left = 0, user loses

BEQ lose

LDR R0, address\_of\_message3

BL printf

MOV R1, R11

LDR R0, address\_of\_message4

BL printf

test:

CMP R10, #0 @if size of word = 0, no more letters to guess (win)

BNE loop

MOV R0, #0 @if win, return 0

b end

lose:

MOV R0, #1 @if lose return 1

end:

ADD sp, sp, #4 @Discard the integer read by scanf

LDR lr, [sp], #+4 @Pop the top of the stack and put it in lr

bx lr @Leave main

@\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

address\_of\_message1: .word message1

address\_of\_message2: .word message2

address\_of\_message3: .word message3

address\_of\_message4: .word message4

address\_of\_message5: .word message5

address\_of\_format: .word format