**Project 1\_v2**

**<Jumble>**

**CSC-5 42829**

**Name: Cervantes, Jose**

**Date: 6/06/2016**

**Introduction**

Game Title: JUMBLE

This is a puzzle game that requires the ability to unscramble words that have been arranged in an unusual order. The player can require assistance bye typing in “hint” if he/she requires it. The game has two levels, in which the game gets difficult as he/she progresses in the game. The game JUMBLE is also a two player game in which player one inputs a four to eight letter word followed with a hint, to assist player two. Then player two has to figure out what player one has typed in. If player two is unable two solve the word, then he/she can type in “hint”. The hint that player one typed in will show on the screen and the user can repeat this game as many times as they wish. This program also has a file name called score.txt which will record all the attempts the user has tried.

**Summary**

Project Size: about 850 lines

The number of variables: about 16

The number of comments: about 156

**PROJECT 1V1 Summary**

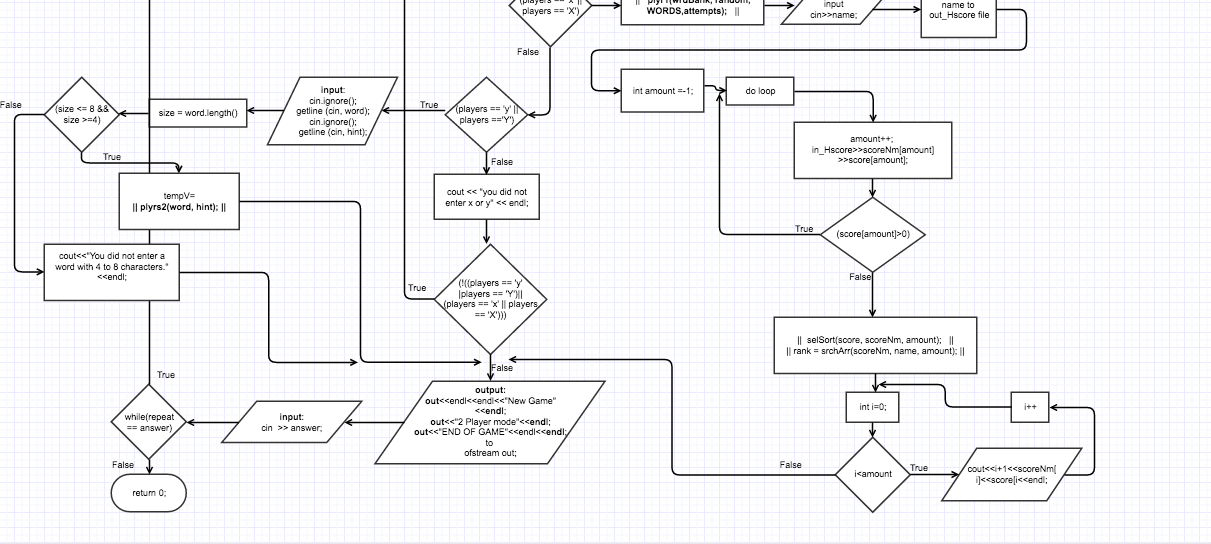
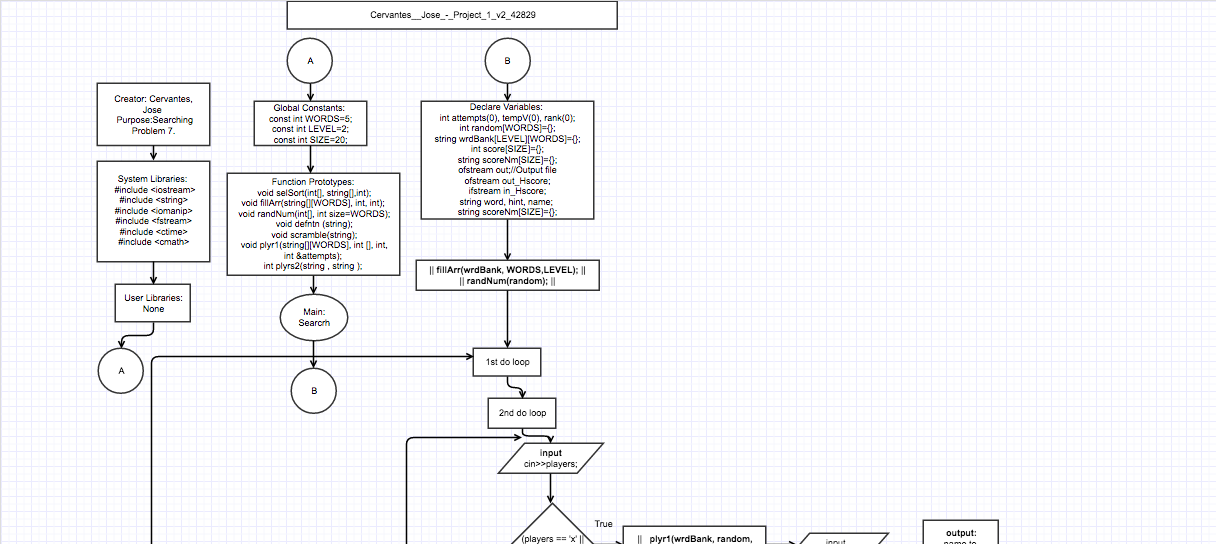
This project includes for loops, while loops, if and else, switch and many more concepts we learned. This project can be improved by using arrays to store more words in a word bank and generate random words from the array. If I were to do that with if and else statement the code will be incredibly long. And maybe create a time limit in how long the user has to solve the word. I spent about a week on this project. I had trouble unscrambling the word that a user would enter, but looked up a function called “ string.at(#) ” which will return a character at any position of the word.

**PROJECT 1V2 Summary**

This project includes functions that are passed by parameters, defaulted Parameters,

**Flow Chart**

**I Only flow charted the main function which contained all the main concepts because the functions would take a long time.**

****

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE** | **Variable Name** | **Description** | **Location** |
| **char** | players | Holds the number of players in a game | char players;  **Liine 60** |
| **unsigned short** | size(0) |  | unsigned short size(0), answer(0);  **Line 61** |
|  | answer(0) | Holds the the value if  The user wants to repeat the game | unsigned short size(0), answer(0);  **Line 61** |
| **int** | attempts(0) | Holds the number of  Attempts the user needed to solve the word | int attempts(0), tempV(0), rank(0);  **Line 58** |
| **ofstream** | out | Stores the score of previous games in a .txt file. | out.open("score.txt",ios::app);  **line 69** |
|  | **out\_Hscore** | Out puts the score of the player with the highest score | out\_Hscore.open("high\_score.txt",ios::app);  **Line 70** |
|  | **in\_Hscore** | Retrieves the score board of the players | in\_Hscore.open("high\_score.txt");  **Line 71** |
| **string** | Word | Take the user inputs for the word they want player 2 to solve. | string word, hint, name;  **Line 65** |
|  | Hint | Holds the key word hint if the user needs help | string word, hint, name;  **Line 65** |
|  | name | Stores the name of the player with the score they got | string word, hint, name;  **Line 65** |
|  | scoreNm[SIZE]={}; | Holds the names of the scoreboard and is a parallel arraya with score[]; | string scoreNm[SIZE]={};  **Line 66** |
|  | wrdBank[LEVEL][WORDS]={}; | Two dimensional array which has two levels and 5 words per level | String wrdBank[LEVEL][WORDS]  ={};  **Line 67** |
| **Int** | score[SIZE]={}; | Is a parallel array to scoreNm and stores the score next to the corresponding name | Int score[SIZE]={};  **Line 63** |
|  | tempV(0) | Holds the amount of attempts player 2 attempted in 2 player mode | int attempts(0), tempV(0), rank(0);  **Line 58** |
|  | random[WORDS]={}; | Generates random numbers from 1 through 5 to output a random order of the words that are being displayed. | Int random[WORDS]={};  **Line 59** |

**C++ Constructs**

|  |  |  |
| --- | --- | --- |
| **Chapter**  **Gaddis** | **New Syntax and Keywords** | **Location** |
| **2** | cout << “”; | Everywhere; |
|  | Unsigned short | unsigned short attempts(0);  **Line 61** |
|  | Int variable | for(int level =1; level <=2; level++)  **Line 275** |
|  | #include <string> | string answer, word1, hint;  **Line 20** |
|  | Equality operators and relational operators (==, !=, >, =, <=) | while(!((players == 'y' || players == 'Y') || (players == 'x' || players == 'X')))  **Line 206** |
|  | Arithmetic operators (+, -, \*, /) | attempt = attempt + 1;  **Line 642** |
|  |  |  |
| **3** | User Interactivity | cin >> answer;  **Line 217** |
|  | cin.ignore() | **Line 97** |
|  | getline (cin, variable) | getline (cin, word);  **Line 170** |
|  |  |  |
| **4** | Equality operators and relational operators (==, !=, >, =, <=) | if (answer == "hint")  **Line 539** |
|  | Bool | bool repeat = true;  **Line 222** |
|  | If/else function | if (players == 'x' || players == 'X')  **Line 454** |
|  | #include <iomanip> | Iomanip library  **Line 21** |
|  | Setw() | cout << setw(47)<<"\*" << endl;  **Line 550** |
|  | Nested If Else Statements | **Line 548** |
|  | And Or operators(|| &&) | else if(players == 'y' || players == 'Y')  **Line 158** |
|  | Comparing Strings | if ((!(answer == unWord)))  **Line 560** |
|  | Switch Statement | switch(size)  **Line 129** |
|  |  |  |
| **5** | Increment Operator | attempts++;  **Line 642** |
|  | While loop | while(repeat == answer)  **Line 138** |
|  | Do While loop | do { }while (!(answer == word1));  **Line 106** |
|  | For loop | for(int level =1; level <=2; level++)  **Line 250** |
|  | File storage | ofstream out;  out.open("hello.txt",ios::app);  **Line line 206** |
|  |  |  |
| **6** | Functions | void selSort(int[], string[],int);  **Line 40** |
|  | #include <fstream> | Fstream library  **Line 22** |
|  | #include <iostream> | Iostream library  **Line 19** |
|  | Passing a function by value, | fillArr(wrdBank, WORDS,LEVEL);  **line 84** |
|  | Passing a function by parameters, | plyr1(wrdBank, random, WORDS, attempts);  **line 121** |
|  | Passing a function by reference | void plyr1(string wrdBank[][WORDS], int order[], int words, int &attempts)  **Line 491** |
|  | returning primitive data types | int srchArr(string \*n, string answer, int size)  **Line 247** |
|  | Functions with defaulted parameters, | void randNum(int[], int size=WORDS);  **Line 42** |
|  |  |  |
| **7** | Arrays |  |
|  | 1 dimensional array | if(score[i] > score[x])  **Line 278** |
|  | 2 dimensional arrays | unWord = wrdBank[i][order[x]];  **Line 525** |
|  | Passing arrays through a function | void randNum(int[], int size=WORDS);  **Line 42** |
|  | Searching an array | int srchArr(string \*n, string answer, int size)  **Line 247** |
|  | Sorting an array | void selSort(int score[], string scoreNm[], int size)  **Line 271** |
|  |  |  |
| **9** | Pointers | int srchArr(string \*n, string answer, int size)  **line 247** |

**Pseudo code**

Declare players as char variable

Declare size, answer as unsigned short

Declare repeat as a true bool

Declare word, hint as strings

Do(do statement re-runs the whole game if the user wants do)

Do(do statement re-runs the menu for what option to enter if the player

enters a value that does not match the one given)

X = 1 player

Y = 2 players

//inputs

Ask user for amount of players

Input player;

If statement for player one is true

If (players == 'x' || players == 'X') //This Is ran

| Function call

| || plyr1(wrdBank, random, WORDS, attempts); ||

|

|

|

Else if (players == 'y' || players == 'Y')//Two Players

| Ask user for a 4 to 8 letter word

| Input word

| Ask user for a hint of the 4 to 8 letter word

| Input hint

| Calculate size by getting the number of letters in the word

|

| | if (size <= 8 && size >=4)

| | user inputs a four to eight letter word

| | function call

| | || **size = word.length();** ||

| |

| |

| |

| |

| |else You did not enter a word with 4 to 8

|

else the user did not enter x or y

While (!((players == 'y' || players == 'Y')|| (players == 'x' || players == 'X')))

Ask the user if the want to rerun the program again

Input answer

While repeat == answer