

Folder src

24 printable files

(file list disabled)

src\Events\KeyEvents\CtrlC\CtrlC.c

```
1 //includes
2 #include "CtrlC.h"
3 #include "../../SafeExit/SafeExit.h"
4
5 //definition
6 void CtrlC(){
7     SafeExit();
8 }
```

src\Events\KeyEvents\CtrlC\CtrlC.h

```
1 #ifndef CTRL_C_H_
2 #define CTRL_C_H_
3
4 //declarations
5 void CtrlC();
6
7 #endif
```

src\Events\SafeExit\SafeExit.c

```
1 //include
2 #include "../../Globals.h"
3 #include "SafeExit.h"
4
5 //definition
6 void SafeExit(){
7     shouldExit = 1;
8 }
```

src\Events\SafeExit\SafeExit.h

```
1 #ifndef EXIT_H_
2 #define EXIT_H_
3
4 //declarations
5 void SafeExit();
6
7 #endif
```

src\Globals.h

```
1  #ifndef GLOBALS_H_
2  #define GLOBALS_H_
3
4  //global variable declarations
5  extern char* fileLocation;
6  extern char* content;
7  extern int shouldExit;
8  extern int size;
9  extern int jump;
10 extern int foot;
11
12 #endif
```

src\Input\InputHandler\Foot\Foot.c

```
1  //includes
2  #include <stdlib.h>
3  #include "../..../Math/Min/Min.h"
4  #include "Foot.h"
5
6  //definition
7  void Foot(char tokenInput[64][16], int* footPointer, int size){
8      //get foot location address and set to foot variable
9      *footPointer = strtol(tokenInput[1], NULL, 16);
10
11      //ensure jump is less than size + 1
12      *footPointer = Min(*footPointer, size+1);
13 }
```

src\Input\InputHandler\Foot\Foot.h

```
1  #ifndef FOOT_H_
2  #define FOOT_H_
3
4  //declaration
5  void Foot(char tokenInput[64][16], int* footPointer, int size);
6
7  #endif
```

src\Input\InputHandler\InputHandler.c

```
1  //include
2  #include <stdio.h>
3  #include <string.h>
4  #include "../..../Globals.h"
5  #include "InputHandler.h"
6  #include "Save/Save.h"
7  #include "Write/Write.h"
8  #include "Jump/Jump.h"
9  #include "Foot/Foot.h"
10
```

```

11 //definition
12 void InputHandler(char tokenInput[64][16]){
13     //command hand selection and hand off
14     if(strcmp(tokenInput[0], "save\n") == 0){
15         Save(fileLocation, content, size);
16     }else if(strcmp(tokenInput[0], "write") == 0){
17         Write(content, tokenInput);
18     }else if(strcmp(tokenInput[0], "jump") == 0){
19         Jump(tokenInput, &jump);
20     }else if(strcmp(tokenInput[0], "foot") == 0){
21         Foot(tokenInput, &foot, size+1);
22     }else{
23         printf("Invalid command\n");
24     }
25 }

```

src\Input\InputHandler\InputHandler.h

```

1 #ifndef INPUT_HANDLER_H_
2 #define INPUT_HANDLER_H_
3
4 //declarations
5 void InputHandler(char tokenInput[64][16]);
6
7 #endif

```

src\Input\InputHandler\Jump\Jump.c

```

1 //includes
2 #include <stdlib.h>
3 #include "../Math/Max/Max.h"
4 #include "Jump.h"
5
6 //definition
7 void Jump(char tokenInput[64][16], int* jumpPointer){
8     //get jump location address and set to jump variable
9     *jumpPointer = strtol(tokenInput[1], NULL, 16);
10
11     //ensure jump is 0 or more
12     *jumpPointer = Max(*jumpPointer, 0);
13 }

```

src\Input\InputHandler\Jump\Jump.h

```

1 #ifndef JUMP_H_
2 #define JUMP_H_
3
4 //declaration
5 void Jump(char tokenInput[64][16], int* jumpPointer);
6
7 #endif

```

src\Input\InputHandler\Save\Save.c

```
1 //includes
2 #include <stdio.h>
3 #include "Save.h"
4
5 //definition
6 void Save(char* location, char* content, int size){
7     //open file
8     FILE* fp = fopen(location, "wb");
9
10    //write data to file
11    fwrite(content, size, 1, fp);
12
13    //close file
14    fclose(fp);
15
16    //respond to input
17    printf("Saved to file\n");
18 }
```

src\Input\InputHandler\Save\Save.h

```
1 #ifndef SAVE_H_
2 #define SAVE_H_
3
4 //declaration
5 void Save(char* location, char* content, int size);
6
7 #endif
```

src\Input\InputHandler\Write\Write.c

```
1 //includes
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "Write.h"
5
6 //definition
7 void Write(char* content, char chunkedInput[64][16]){
8     //get lower address line
9     int lowerAddr = strtol(chunkedInput[1], NULL, 16);
10
11    //get upper address line
12    int upperAddr = strtol(chunkedInput[2], NULL, 16);
13
14    //get lower byte
15    int lowerByte = atoi(chunkedInput[3]);
16
17    //get upper byte
18    int upperByte = atoi(chunkedInput[4]);
19 }
```

```

20 //get line and data size
21 int lineSize = upperAddr - lowerAddr;
22 int dataSize = upperByte - lowerAddr;
23
24 //get data and store to data array
25 int* data = malloc(dataSize);
26 for (int i = 0; i < dataSize; i++){
27     sscanf(chunkedInput[5+i], "%hhd", &(data[i]));
28 }
29
30 //write data
31 for (int line = 0; line < lineSize; line++){
32     for (int byte = 0; byte < dataSize; byte++){
33         int location = (lowerAddr+line)*16+lowerByte+byte;
34         content[location] = data[byte];
35     }
36 }
37 }

```

src\Input\InputHandler\Write\Write.h

```

1 #ifndef WRITE_H_
2 #define WRITE_H_
3
4 //declaration
5 void Write(char* content, char chunkedInput[64][16]);
6
7 #endif

```

src\Input\Tokenize\Tokenize.c

```

1 //include
2 #include <string.h>
3 #include "Tokenize.h"
4
5 //definition
6 int Tokenize(char* content, char tokenedOutput[64][16], char* delimiter, int numTokens, int
tokenSize){
7     int count = 0;
8     char* token = strtok(content, delimiter);
9     while (token != NULL && count < numTokens){
10         strncpy(tokenedOutput[count], token, tokenSize);
11         count += 1;
12         token = strtok(NULL, delimiter);
13     }
14     return count;
15 }

```

src\Input\Tokenize\Tokenize.h

```

1 #ifndef TOKENIZE_H_
2 #define TOKENIZE_H_

```

```
3 |
4 | //declarations
5 | int Tokenize(char* content, char tokenedOutput[64][16], char* delimiter, int numTokens, int
   | tokenSize);
6 |
7 | #endif
```

src\Math\Max\Max.c

```
1 | //include
2 | #include "Max.h"
3 |
4 | //definitions
5 | int Max(int a, int b){
6 |     if (a > b){
7 |         return a;
8 |     }
9 |     return b;
10 | }
```

src\Math\Max\Max.h

```
1 | #ifndef MAX_H_
2 | #define MAX_H_
3 |
4 | //declarations
5 | int Max(int a, int b);
6 |
7 | #endif
```

src\Math\Min\Min.c

```
1 | //include
2 | #include "Min.h"
3 |
4 | //definitions
5 | int Min(int a, int b){
6 |     if (a < b){
7 |         return a;
8 |     }
9 |     return b;
10 | }
```

src\Math\Min\Min.h

```
1 | #ifndef MIN_H_
2 | #define MIN_H_
3 |
4 | //declarations
5 | int Min(int a, int b);
6 |
```

```
7 | #endif
```

src\Output\DisplayContent\DisplayContent.c

```
1 | //include
2 | #include <stdio.h>
3 | #include "DisplayContent.h"
4 |
5 | //definition
6 | void DisplayContent(char* content, int size, int jump, int foot){
7 |     //display header
8 |     printf("          LA_X LA_Y RA_X RA_Y D_UP D_DW D_LF D_RH D_BA D_BB D_BX D_BY D_L1 D_L2 D_R1
D_R2\n");
9 |
10 |     //loop through each line of content
11 |     //one line for every 16 bytes
12 |     for (int line = 0; line < (int)((foot-jump)/16); line++){
13 |         //print line address
14 |         printf("$%04x ", line*16+jump);
15 |
16 |         //loop through each byte of line
17 |         for (int byte = 0; byte < 16; byte++){
18 |             //get value in int8 form
19 |             __int8 value = (__int8) content[line*16+byte];
20 |
21 |             //print value with leading sign(special manipulation for positive)
22 |             if (value >= 0){
23 |                 printf("+%03i ", value);
24 |             }else{
25 |                 printf("%04i ", value);
26 |             }
27 |         }
28 |
29 |         //create new line
30 |         printf("\n");
31 |     }
32 | }
```

src\Output\DisplayContent\DisplayContent.h

```
1 | #ifndef DISPLAY_CONTENT_H_
2 | #define DISPLAY_CONTENT_H_
3 |
4 | //declarations
5 | void DisplayContent(char* content, int size, int jump, int foot);
6 |
7 | #endif
```

src\main.c

```
1 | //include
2 | #include <stdio.h>
```

```
3  #include <stdlib.h>
4  #include <string.h>
5  #include <signal.h>
6  #include "Globals.h"
7  #include "Events/KeyEvents/CtrlC/CtrlC.h"
8  #include "Events/SafeExit/SafeExit.h"
9  #include "Output/DisplayContent/DisplayContent.h"
10 #include "Input/Tokenize/Tokenize.h"
11 #include "Input/InputHandler/InputHandler.h"
12
13 //initialization of globals
14 char* fileLocation;
15 char* content;
16 int shouldExit = 0;
17 int size = 0;
18 int jump = 0;
19 int foot = 0;
20
21 //main method
22 int main(int argc, char** argv){
23     //set ctrl c signal
24     signal(SIGINT, CtrlC);
25
26     //check if location passed
27     if (argc <= 1){
28         //location not passed, error out
29         printf("Error, no file passed");
30         return 1;
31     }
32
33     //set file location
34     fileLocation = argv[1];
35
36     //load file into file pointer
37     FILE* filePointer = fopen(argv[1], "rb");
38
39     //check if file exists
40     if (filePointer == NULL){
41         printf("Cannot find %s: No such file", argv[1]);
42         return 1;
43     }
44
45     //seek file size, store to size
46     fseek(filePointer, 0 , SEEK_END);
47     size = ftell(filePointer);
48
49     //ensure size is a multiple of 16
50     if (size % 16 != 0){
51         printf("Error, missing data, size is not a multiple of 16");
52         return 1;
53     }
54
55     //create content buffer with allocation of size + 1 -needed for end char-
56     //read file into content buffer
57     content = malloc(size+1);
58     content[size] = '\0';
```



```
59     fseek(filePointer, 0, SEEK_SET);
60     fread(content, size, 1, filePointer);
61
62     //close file for safety
63     fclose(filePointer);
64
65     //set jump and foot variables
66     jump = 0;
67     foot = size;
68
69     //output file contents
70     DisplayContent(content, size, jump, foot);
71
72     //create input buffer of size 1024(64*16)
73     //create token input buffer of size 1024(64*16)
74     char inputBuffer[1024];
75     char tokenInput[64][16];
76
77     //input loop
78     while (shouldExit == 0){
79         //print line leader
80         printf("Editor ~ ");
81
82         //get input and push into input buffer
83         fgets(inputBuffer, sizeof(inputBuffer), stdin);
84
85         //tokenize input and store number of tokens
86         int numTokens = Tokenize(inputBuffer, tokenInput, " ", 64, 16);
87
88         //handle input
89         InputHandler(tokenInput);
90
91         //re-output file contents
92         DisplayContent(content, size, jump, foot);
93     }
94
95     //return successful
96     return 0;
97 }
```