**מעבדה מספר 3 במת"מ –קבצים:**

תרגיל מספר 1:

**מגישים**: **אורי מלכא** 314862996 **ואלן ציפין** 313206062

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

typedef struct Person

{

char ID[10];

char F\_name[11];

char L\_name[16];

int Age;

char Addr[51];

}Person;

void Error\_Msg(char\*);

/\*

Function name:Error\_Msg

Input: string

Output: void

Function Algorithm: prints string using printf.

\*/

int main()

{

Person temp;

FILE\* in, \* out;

in = fopen("the\_first.txt", "rt"); // "r" is the default but we putted it anyway.

if (in == NULL)

{

Error\_Msg("Error we couldn't open the\_first.txt, aborting"); //error msg if we failed to open the txt file.

return 1;

}

out = fopen("the\_second.txt", "wt"); // open to write only the\_second file.

if (out == NULL)

{

Error\_Msg("Error we couldn't open the\_second.txt, aborting"); //error msg if we failed to open the txt file.

return 1;

}

// we read the information from the\_first in the temp person to structure and then we write it .

fscanf(in,"%s %s %s %d %s",temp.ID,temp.F\_name,temp.L\_name,&temp.Age,temp.Addr);

fprintf(out,"ID:%s\nFull name:%s %s\nAge:%d\nAddress:%s",temp.ID, temp.F\_name, temp.L\_name,temp.Age, temp.Addr);

fclose(in); // close the files.

fclose(out);

return 0;

}

void Error\_Msg(char\* msg)

{

printf("\n%s", msg);

exit(1);

}

תרגיל מס' 2:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

typedef struct

{

char name[20];

int grades[3];

}student;

void Error\_Msg(char\*);

/\*

Function name:Error\_Msg

Input:string

Output: void

Function Algorithm:prints a msg and exits the program

\*/

int InputData(student\*\*, FILE\*);

/\*

Function name:InputData

Input: pointer to student array and pointer to the file.

Output: int.

Function Algorithm: returns the number of students.

\*/

void OutputData(int, student\*, FILE\*);

/\*

Function name:OutputData

Input:int, pointer to a student structure and a pointer to a file.

Output: void

Function Algorithm: run over the file to check each student grades and keeps the highest between them.

\*/

int main()

{

FILE\* fp;

student\* arr;

int size;

if (!(fp = fopen("Students.txt", "rt")))

Error\_Msg("The input file is wrong");

size = InputData(&arr, fp);

fclose(fp);

if (!(fp = fopen("Students.txt", "wt")))

Error\_Msg("The output file is wrong");

OutputData(size, arr, fp);

fclose(fp);

free(arr);

return 0;

}

int InputData(student\*\* p\_array, FILE\* fp)

{

student\* arr;

int i = 1;

arr = (student\*)malloc(sizeof(student));

if (arr == NULL) {

Error\_Msg("No memory");

}

while (fscanf(fp, "%s %d %d %d", arr[i - 1].name, &arr[i - 1].grades[0], &arr[i - 1].grades[1], &arr[i - 1].grades[2]) != EOF)

{

arr = (student\*)realloc(arr, (++i) \* sizeof(student)); //reallocate new memo and one more spot.

if (arr == NULL) { //check if memo allocation succeeded

free(arr);

Error\_Msg("No memory");

}

}

\*p\_array = arr;

return i - 1; /\*return the number of students\*/

}

void OutputData(int arr\_size, student\* arr, FILE\* fp)

{

int i, j;

for (i = 0; i < arr\_size; i++) { //for loop that runs all over the student structure array

int tempmax = arr[i].grades[0];

for (j = 1; j < 3; j++) { // here we run on each student grades to find the max

if (arr[i].grades[j] > tempmax) tempmax = arr[i].grades[j];

}

fprintf(fp, "%s %d\n", arr[i].name, tempmax); //we write over the existed file the new grade (we keep the same name)

}

}

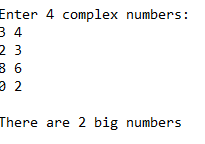
void Error\_Msg(char\* msg)

{

printf("\n%s", msg);

exit(1);

}

תרגיל מס' 3:  
  
  


#define \_CRT\_SECURE\_NO\_WARNINGS

#define N 4

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

typedef struct //Complex number structure Z=X+YI when r=Real part of Z and i=Imaginary part of Z.

{

float r, i;

}Complex;

float CRadius(Complex z);

/\*

Function name: CRadius

Input: complex structure

Output: float number

Function Algorithm: calcs the radius of the complex number given

\*/

void Error\_Msg(char\* str);

/\*

Function name:Error\_Msg

Input: string

Output:void

Function Algorithm: prints an error msg and exits the program.

\*/

void InputAndWriteToFile(FILE\* f);

/\*

Function name: InputAndWriteToFile

Input: file

Output: void

Function Algorithm: recives n complex numbers and writes them to the given text file togther with their radius

\*/

int CheckFile(FILE\* f, float m);

/\*

Function name:CheckFile

Input: file and float number

Output: int number

Function Algorithm: runs over the file and returns the number of complex numbers that have a bigger radius then "m" .

\*/

int main()

{

FILE\* f;

f = fopen("complex\_number.txt", "wt"); //open to write only

if (f == NULL) {

Error\_Msg("Error We couldn't open complex\_number.txt "); //check if we succeeded to open the txt file.

return 1;

}

InputAndWriteToFile(f);

fclose(f);

printf("\nThere are %d big numbers\n", CheckFile(f, 4));

return 0;

}

void Error\_Msg(char\* str)

{

printf("\n%s", str);

exit(1);

}

float CRadius(Complex z) {

float r = sqrt(pow(z.r, 2) + pow(z.i, 2)); // calcs the radius size of Z.

return r;

}

void InputAndWriteToFile(FILE\* f) {

int i;

Complex arr[N]; //array of structure complex numbers

printf("Enter %d complex numbers:\n", N);

for (i = 0; i < N; i++)

{

scanf("%f%f" ,&arr[i].r, &arr[i].i);// we recive from the user the complex number and we get the real and image part of the number

fprintf(f, "%.1f %.1f %.1f\n", arr[i].r, arr[i].i, CRadius(arr[i])); //we write on the txt file the real part image part and the radius using the function Cradius.

}

}

int CheckFile(FILE\* f, float m) {

int i, counter = 0;

f = fopen("complex\_number.txt", "rt");// we open the file in read only

if (f == NULL) {

Error\_Msg("Error We couldn't open complex\_number.txt "); //check if we succeeded to open the txt file.

return 1;

}

fseek(f, 2 \* sizeof(float), SEEK\_SET); //using fseek function to put the pointer to the third float (first radius)

for(i=1;i<N;i++){

float temp;

temp = getc(f)-48.0; //when we use getc we recive the ASCII number of the digit and we decrease 48 to display as a number

if(temp > m) counter++; //we count the radiuses that bigger then the recived "m"

fseek(f, 2 \* sizeof(float), SEEK\_CUR); //using fseek function agian we jump 2 sizeof(float) bytes from the current posting to the next raduis

}

fclose(f);

return counter;

}