نیمسال دوم 99-98	عنوان: آزمایشگاه هفتم (کار با فایل در زبان c)	استاد : دکتر مرادی

در این جلسه شما ابتدا به نحوه‌ی تبادل اطلاعات با فایل‌ها در زبان c آشنا می‌شوید.

کار با فایل :

برای کار با فایل‌ها در زبان c باید ابتدا یک اشاره‌گر از نوع FILE بسازیم. پس از آن با استفاده از دستور fopen می‌توانیم یک فایل از حافظه‌ی کامپیوتر را باز کرده و به محتوای آن دسترسی پیدا کنیم. مقدار بازگشتی این تابع اشاره‌گر از نوع FILE است. به مثال زیر توجه کنید:

```
FILE *myfile = fopen(“out.txt”, “wb”);
```

تابع fopen دو ورودی دریافت کرده که ورودی اول آدرس و نام فایل با فرمت char* و ورودی دوم نوع رفتار با فایل را مطابق جدول زیر تعیین می‌کند:

“r”	خواندن از فایل متنی
“w”	نوشتن در فایل متنی
“a”	اضافه کردن به انتهای فایل متنی
“rb”	خواندن از فایل به صورت باینری
“wb”	نوشتن در فایل به صورت باینری
“ab”	اضافه کردن به انتهای فایل به صورت باینری

برای نوشتن در فایل توابعی مانند fprintf و fwrite و برای خواندن توابعی مانند fscanf و fread وجود دارند. توابع fprintf و fscanf همانند توابع printf و scanf عمل می‌کنند با این تفاوت که ورودی اول آن‌ها از نوع FILE* است. از این رو به بررسی توابع fwrite و fread می‌پردازیم. این توابع یک قطعه (block) از اطلاعات را در فایل می‌نویسند یا می‌خوانند. به این منظور این توابع به عنوان ورودی اول یک اشاره‌گر به ابتدای یک آرایه، ورودی دوم اندازه‌ی هر قسمت از block ، ورودی سوم طول قطعه و ورودی چهارم اشاره‌گر از نوع FILE دریافت می‌کنند. سپس به اندازه‌ی اندازه‌ی هر قسمت * طول از آدرس اشاره‌گر به آرایه آغاز کرده و در فایل می‌نویسند(یا می‌خوانند). به قطعه کد زیر توجه کنید.

```
FILE *myfile = fopen(“out.txt”, “wb”);
char *str = “Hello!?”;
fwrite(str, sizeof(char), 5, myfile);
fclose(myfile);
```

نکته : حتما باید در انتهای برنامه فایل‌های باز شده را با استفاده از دستور fclose ببندیم.

نکته: پس از استفاده از توابع fread و fwrite پیمایش‌کننده‌ی فایل در محل جدیدی قرار می‌گیرد. این محل اولین محل پس از محتوای خوانده یا نوشته شده است.

نکته : در صورتی که مانند کد بالا در قسمت آدرس تنها اسم فایل را ذکر کنیم، مرجع آدرس فایل آدرس زیر است :

```
Documents -> Visual Studio 201x -> Projects -> project name -> project name
```

1. انجام دهید!

هدف نوشتن برنامه‌ای است تا یک فایل را بخواند و متن داخل آن را به صورت معکوس در فایل دیگری بنویسید. به این منظور، قطعه کدهای زیر را کامل کنید:

```
#define ZERO 0
#define ONE 1
#define READ_CHAR_SIZE 78
#define WRITE_CHAR_SIZE 78
#define INPUT_TXT_ADDRESS "input.txt"
#define OUTPUT_FILE_ADDRESS "output.txt"

char* read_input_file() {
	char* in_order_array = (char*)malloc(READ_CHAR_SIZE *
sizeof(char));
	FILE* input = fopen(INPUT_TXT_ADDRESS, /*Fill the gap.(It is a
known fact that you are going to read from a .txt file…)*/ );
	fread(/* Complete this part with cogent reasoning */);

	/* Possibly your mind is rife with an assumption about completing
	the function. I would like to, if I may, state that you're
	missing an item. */

	return in_order_array;
}

char* reverse_array(char* in_order_array) {
	char *reversed_array = (char*)malloc(READ_CHAR_SIZE *
sizeof(char))


	for (int i = ZERO; i < READ_CHAR_SIZE; i++){
		// Write down a code to reverse the input array. While you
		may already have considered that it is just an easy task,
		and you're almost right, but be careful about the indexes.
```

```
	}
	return reversed_array;
}
void write_reversed_array_in_file(char* in_order_array) {
	char *reversed_array = reverse_array(in_order_array);
	FILE* output = fopen(OUTPUT_FILE_ADDRESS, "w");
	fwrite(/* Complete this part with cogent reasoning */);

	/* Possibly your mind is rife with an assumption about completing
	the function. I would like to, if I may, state that you're
	missing an item. */

}

int main() {
	char* in_order_array = read_input_file();
	write_reversed_array_in_file(in_order_array);
	return 0;
}
```

قسمت1: قطعه کدهای داده شده را کامل کرده و آن‌ها را به همراه نتایج به دست آمده ، در کادر زیر بنویسید.

```
char* read_input_file() {
	char* in_order_array = (char*)malloc(READ_CHAR_SIZE *  sizeof(char));
	FILE* input = fopen(INPUT_TXT_ADDRESS, "r");
	fread(in_order_array , sizeof(char) , READ_CHAR_SIZE , input);
	fclose(input);
	return in_order_array;
}
char* reverse_array(char* in_order_array) {
	char *reversed_array = (char*)malloc(READ_CHAR_SIZE * sizeof(char));
	for (int i = ZERO; i < READ_CHAR_SIZE; i++){
		reversed_array[i] = in_order_array[READ_CHAR_SIZE - 1 - i];
	}
	return reversed_array;
}
void write_reversed_array_in_file(char* in_order_array) {
	char *reversed_array = reverse_array(in_order_array);
	FILE* output = fopen(OUTPUT_FILE_ADDRESS, "w");
	fwrite(reversed_array , sizeof(char) , WRITE_CHAR_SIZE , output);
	fclose(output);
}
```

فایل input.txt خوانده شده و برعکس شده عبارت داخل آن در output.txt ذخیره می شود. فقط حتما باید به این نکته توجه شود که در این کد فرض بر این بوده است که محتویات داخل فایلی که از آن خوانده می شود و فایلی که در آن نوشته می شود اندازه ای برابر 78 دارند و اگر این اندازه بیشتر و یا کم تر باشد مقادیر نامتعارف چاپ شده و یا متن برعکس شده کامل نشان داده نمی شود.

2. انجام دهید!

1) در قسمت قبل فایل `input.txt` را از پوشه‌ی محل پروژه حذف کرده و سپس دوباره برنامه را اجرا کنید. چه اتفاقی می‌افتد؟
2) فایل `input.txt` را دوباره در محل پروژه قرار دهید.
3) در مورد مشکلاتی که در صورت عدم استفاده از `fclose` ممکن است اتفاق بیفتد، در اینترنت تحقیق کنید.
4) قبل از دستورات `fclose` در کد قسمت قبل `breakpoint` بگذارید و برنامه را در حالت `debug` اجرا کنید. پس از توقف برنامه در محل `breakpoint` سعی کنید فایل‌های `input.txt` یا `out.txt` را از محل پروژه حذف کنید. چه اتفاقی می‌افتد؟

قسمت2: موارد خواسته شده را انجام دهید. نتایج به دست آمده و همچنین پاسخ سوالات را در کادر زیر بنویسید.

با حذف کردن فایل input.txt برنامه دچار خطای اجرا می شود و دلیل آن این است که وقتی که برنامه می خواهد از روی فایل بخواند آن را در محل مورد نظر که پوشه محل پروژه است پیدا نمی کند و به همین دلیل پیدا نشدن فایل، نمی تواند متنی را بخواند و به runtime error می خورد.

یکی از مشکلاتی که می تواند ایجاد کند این است که اگر fopen را صدا زده و fclose را استفاده نکنیم بعد از چندین بار اجرای برنامه fopen به مشکل خورده و به درستی کار نمی کند و به خطا می خورد. یکی دیگر از مشکلات هدر دادن حافظه خواهد بود به دلیل این که با این که از فایل استفاده ای نمی شود ولی همچنان در حال اجرا است. همچنین یکی دیگر از این مشکلات این است که با نبستن فایلی که آن را باز کرده ایم بعد از چندین بار اجرا تعداد فایل هایی که برنامه می تواند اداره کند پر شده و با این که فقط یک فایل را باز کرده ایم برنامه قابلیت اداره کردن فایل دیگری را نخواهد داشت و در باز کردن فایل های دیگر با fopen به مشکل خواهیم خورد.

اجازه حذف کردن فایل به ما داده نمی شود و دلیل آن این است که فایل مد نظر درون برنامه باز شده است و اگر فایلی باز باشد امکان حذف کردن آن فایل برای ما وجود ندارد و با توجه به این که fclose را هنوز صدا نزده ایم برنامه بسته نشده است پس در نتیجه نمی توانیم آن را حذف کنیم.

3. انجام دهید! (EOF)

در کار با فایل‌ها، انتهای فایل با مقدار ثابتی به نام EOF معرفی می‌شود. همواره می‌توان با بررسی برابر بودن آخرین کاراکتر دریافت شده و ثابت EOF و یا با استفاده از تابع `feof`، که ورودی آن اشاره‌گر به فایل مورد نظر است، رسیدن به انتهای فایل را بررسی کرد.

1) برنامه‌ی قسمت اول را به گونه‌ای تغییر دهید تا با متغیر بودن طول رشته‌ی درون فایل `input.txt` عملیات معکوس‌سازی را همانند قبل انجام دهد. برای سادگی، فرض کنید که طول رشته ورودی خوانده شده هیچ‌گاه از `100` عبور نخواهد کرد.
2) طول رشته‌ی درون فایل `input.txt` را تغییر دهید و برنامه را اجرا کنید.

قسمت3: موارد خواسته شده را انجام دهید. نتایج به دست آمده را در کادر زیر بنویسید.

```c
#include <stdio.h>
#include <stdlib.h>

#define ZERO 0
#define ONE 1
#define DEFAULT_SIZE 100
#define INPUT_TXT_ADDRESS "input.txt"
#define OUTPUT_FILE_ADDRESS "output.txt"

void write_reversed_array_in_file(char* in_order_array, int size_array) {
	char *reversed_array = (char*)malloc(size_array * sizeof(char));
	for (int i = ZERO; i < size_array; i++){
		reversed_array[i] = in_order_array[size_array - 1 - i];
	}
	FILE* output = fopen(OUTPUT_FILE_ADDRESS, "w");
	fwrite(reversed_array, sizeof(char), size_array, output);
	fclose(output);
}

void read_input_file() {
	char* in_order_array = (char*)malloc(DEFAULT_SIZE *  sizeof(char));
	FILE* input = fopen(INPUT_TXT_ADDRESS, "r");

	int size_array = 0;
	while (!feof(input)){
		fread(in_order_array + size_array, sizeof(char), 1, input);
		size_array++;
	}
	fclose(input);
	write_reversed_array_in_file(in_order_array, size_array - 1);;
}

int main() {
	read_input_file();
	return 0;
}
```

با این کد دیگر به مشکلی که در کد بخش اول داشتیم که حتما باید محتویات داخل فایل اندازه ای برابر 78 داشتند برنمی خوریم.

تابع fseek :

همانطور که ذکر شد برای کار با فایل‌ها یک اشاره‌گر از نوع FILE که به فایل مورد نظر اشاره می‌کند تعریف می‌کنیم. برای تغییر محل پیمایش‌کننده‌ی فایل، می‌توانیم از تابع fseek استفاده کنیم. ورودی اول این تابع اشاره‌گر به فایل مورد نظر، ورودی دوم مقدار تغییر مکان پیمایش‌کننده و ورودی سوم مرجع تغییر است؛ که با استفاده از SEEK_SET به ابتدای فایل و با استفاده از SEEK_CUR به مکان فعلی پیمایش‌کننده اشاره می‌کند.

4. انجام دهید!

هدف تغییر کد قسمت اول به طریقی است که علاوه بر معکوس‌سازی متن ورودی، حروف یکی در میان حذف شوند(در این مثال خطوط تیره باید حذف شوند). برای این کار :

1) از کد قسمت سوم استفاده کنید تا کاراکتر های ورودی را تک تک دریافت کنید.
2) پس از دریافت هر کاراکتر (با استفاده از دستور fread یا fgetc) با استفاده از دستور fseek مکان پیمایش‌کننده را به محل بعد از خط‌تیره انتقال دهید.

قسمت4: روند ذکر شده را طی کرده و نتیجه به دست آمده را در کادر زیر بنویسید.

```
#include <stdio.h>
#include <stdlib.h>

#define ZERO 0
#define ONE 1
#define DEFAULT_SIZE 100
#define INPUT_TXT_ADDRESS "input.txt"
#define OUTPUT_FILE_ADDRESS "output.txt"

void write_reversed_array_in_file(char* in_order_array, int size_array) {
	char *reversed_array = (char*)malloc(size_array * sizeof(char));
	for (int i = ZERO; i < size_array; i++){
		reversed_array[i] = in_order_array[size_array - 1 - i];
	}
	FILE* output = fopen(OUTPUT_FILE_ADDRESS, "w");
	fwrite(reversed_array, sizeof(char), size_array, output);
	fclose(output);
}

void read_input_file() {
	char* in_order_array = (char*)malloc(DEFAULT_SIZE *  sizeof(char));
	FILE* input = fopen(INPUT_TXT_ADDRESS, "r");

	int size_array = 0;
	while (!feof(input) && fread(in_order_array + size_array, sizeof(char), 1, input)){
		fseek(input, 1, SEEK_CUR);
		size_array++;
	}
	fclose(input);
	write_reversed_array_in_file(in_order_array, size_array);;
}

int main() {
	read_input_file();
	return 0;
}
```

همانطور که در قسمت اول نیز ذکر شد، حالت‌های مختلفی در خواندن و نوشتن فایل‌های ورودی و خروجی قرار دارد. یکی از این حالات append کردن است. در این حالت وقتی می‌خواهیم اطلاعاتی را در فایل بنویسیم این اطلاعات را به انتهای یک فایل که موجود است اضافه کنیم. برای این کار باید حالت باز کردن فایل را 'a' قرار دهیم.

1) برنامه‌ای بنویسید که اطلاعات درون فایل input.txt را به انتهای فایل out.txt که از قسمت قبل به‌دست آمده است اضافه کند.

قسمت5: نتیجه به دست آمده را در کادر زیر بنویسید.

```
#include <stdio.h>
#include <stdlib.h>

#define ZERO 0
#define ONE 1
#define DEFAULT_SIZE 100
#define INPUT_TXT_ADDRESS "input.txt"
#define OUTPUT_FILE_ADDRESS "output.txt"

void read_input_file_and_append() {
	char* in_order_array = (char*)malloc(DEFAULT_SIZE *  sizeof(char));
	FILE* input = fopen(INPUT_TXT_ADDRESS, "r");

	int size_array = 0;
	while (!feof(input) && fread(in_order_array + size_array, sizeof(char), 1,
input)){
		size_array++;
	}
	fclose(input);
	FILE* output = fopen(OUTPUT_FILE_ADDRESS, "a");
	fwrite(in_order_array, sizeof(char), size_array, output);
	fclose(output);
}

int main() {
	read_input_file_and_append();
	return 0;
}
```

با تغییر مقدار "w" در fwrite به "a" محتویات فایل input.txt در انتهای فایل output.txt قرار می گیرد.

6. انجام دهید!

در این سوال شما میبایست قطعه کدی که در فایل Q6.c قرار دارد را بدون اجرا کردن و تنها با دنبال کردن کد تحلیل کنید و نتیجه را دریابید.

همچنین قطعه کدی نیز در فایل Q6_Additional.c وجود دارد که بررسی آن اختیاری میباشد ولی نمره ای ندارد . توصیه میشود آن را نیز تحلیل کرده و با دستورات موجود در آن آشنا شوید و روند پیشروی کد را تحلیل کنید.

قسمت6: قطعه کد داده شده را مطالعه کرده و پس از تحلیل کردن آن ، نتیجه به دست آمده را در کادر زیر بنویسید.

در این کد 3 فایل که در برنامه با نام های sourceFile1 و sourceFile2 و destFile مشخص شده اند وجود دارند. در ابتدا با اجرای برنامه آدرس هر کدام از این فایل ها از کاربر گرفته شده و به ترتیب در متغیر های sourcePath1 و sourcePath2 و destPath قرارمی گیرد حال ابتدا چک می شود که آیا فایل باز شده است و به مشکلی نخورده است اگر به مشکلی خورده باشد پیامی متناسب با آن چاپ شده و برنامه متوقف می شود اگر فایل پیدا شده باشد و مشکلی در باز شدن آن نباشد برنامه ادامه پیدا می کند حال در دو حلقه ابتدا محتویات داخل sourceFile1 به داخل فایل destFile کاراکتر به کاراکتر منتقل می شود و در حلقه بعدی با توجه به این که نشانه گر در انتهای فایل destFile مانده است محتویات داخل sourceFile2 کاراکتر به کاراکتر در ادامه فایل destFile قرار می گیرد و در واقع محتویات دو فایل sourceFile1 و sourceFile2 ادغام شده و در destFile قرار می گیرد سپس پیام موفقیت آمیز بودن عملیات در کنسول چاپ شده و با بستن هر سه فایل برنامه به پایان می رسد.

7.انجام دهید!(امتیازی)

یکی از حالات ذکر شده در نوشتن و خواندن فایل‌ها حالت دودویی است. از این حالت برای خواندن و نوشتن فایل های غیر متنی استفاده می‌کنیم. در این قسمت می‌خواهیم یک فایل تصویری را باز کرده، تغییراتی در آن داده و ذخیره کنیم. پیش از آن به این نکته توجه کنید که هر فایل در ابتدای خود دارای تعاریفی است که نوع و اطلاعات فایل را را تعیین می‌کند. همچنین در فایل از نوع bmp که در این قسمت با آن کار می‌کنیم هر پیکسل از تصویر توسط یه مقدار 8 بیتی که نمایانگر مقادیر RGB هستند نشان داده می‌شوند.

1) فایل input2.bpm را با استفاده از دستور fopen و در حالت "rb" باز کنید.

2) آرایه ای از کاراکتر به طول 154 بسازید و به همین اندازه از ابتدای فایل خوانده و در آرایه ذخیره کنید.
3) آرایه‌ی سه‌بعدی به ابعاد [3][50][50] بسازید (طول*عرض*سه مقدار RGB) و با استفاده از دو حلقه بلوک‌های 3 تایی از فابل خوانده و در این آرایه ذخیره کنید.
4) تمامی مقادیر آرایه‌ی سه بعدی را 100 عدد اضافه کنید.
5) فایل جدیدی به نام out.bmp ایجاد کرده و نوع باز کردن آن را "wb" انتخاب کنید.
6) ابتدا مقادیر آرایه‌ی به طول 154 را در فایل ذخیره کنید.
7) مقادیر آرایه‌ی سه بعدی را به همان روش خواندن در فایل جدید ذخیره کنید.
8) فایل جدید ایجاد شده را مشاهده کنید.

قسمت7: موارد خواسته شده را انجام دهید ، نتایج به دست آمده را در کادر زیر نشان دهید.

موفق باشید