نیمسال اول	عنوان:	استاد : دکتر مرادی،
۰۰-۰۱	کار با فایل در زبان c	دکتر هاشمی

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 پوشه محل قرارگیری پروژه و برنامه است.

۱. 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 یک فایل با نام `input.txt` بسازید و یک رشته با طول حداکثر در نظر بگیرید؛ سپس قطعه کدهای زیر را کامل کنید:

```
#define ZERO 0
#define ONE 1
#define READ_CHAR_SIZE 100
#define WRITE_CHAR_SIZE 100
#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;
}
```

قسمت۱: نتیجه را به دستیاران آموزشی نشان دهید.

۲. انجام دهید!

۱) در قسمت قبل فایل `input.txt` را از پوشه‌ی محل پروژه حذف کرده و سپس دوباره برنامه را اجرا کنید. چه اتفاقی می‌افتد؟

۲) فایل `input.txt` را دوباره در محل پروژه قرار دهید.

۳) در مورد مشکلاتی که در صورت عدم استفاده از `fclose` ممکن است اتفاق بیفتد، در اینترنت تحقیق کنید.

قسمت۲: نتایج را با دستیاران آموزشی در میان بگذارید.

۳. انجام دهید! (EOF)

در کار با فایل‌ها، انتهای فایل با مقدار ثابتی (یک کاراکتر) به نام EOF معرفی می‌شود. همواره می‌توان با بررسی برابری آخرین کاراکتر دریافت شده و ثابت EOF رسیدن به انتهای فایل را بررسی کرد.
همچنین میتوان با تابع feof، که ورودی آن اشاره‌گر به فایل مورد نظر است، 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، همچنان فرض کنید که طول رشته ورودی خوانده شده هیچ‌گاه از ۱۰۰ عبور نخواهد کرد.

```
int feof(FILE *stream)
```

۱) طول رشته‌ی درون فایل input.txt را تغییر دهید و برنامه را اجرا کنید.

قسمت۳: نتیجه را به دستیاران آموزشی نشان دهید.

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.

۴. انجام دهید!

هدف تغییر کد قسمت اول به طریقی است که علاوه بر معکوس‌سازی متن ورودی، حروف یکی در میان حذف شوند برای مثال:

Abcdefg به gecA تبدیل شود .برای این کار :

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

قسمت۴: نتیجه را به دستیاران آموزشی نشان دهید.

۵. انجام دهید!

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.

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

قسمت۵: نتیجه را به دستیاران آموزشی نشان دهید.

۶. 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 ۸ بیتی که نمایانگر مقادیر RGB هستند نشان داده می‌شوند.

۱۵۴ کاراکتر اول در واقع اطلاعات مربوط به نوع فایل (هدر فایل) است. این ۱۵۴ مقدار باید همانطور به فایل جدید منتقل شود، بقیه اطلاعات اطلاعات مرتبط با هر پیکسل و هر رنگ هستند.

۱) فایل input2.bpm را با استفاده از دستور fopen و در حالت "rb" باز کنید.
۲) آرایه ای از کاراکتر به طول ۱۵۴ بسازید و به همین اندازه از ابتدای فایل خوانده و در آرایه ذخیره کنید.
۳) آرایه‌ی سه‌بعدی به ابعاد [3][50][50] بسازید (طول*عرض*سه مقدار RGB) و با استفاده از دو حلقه بلوک‌های ۳ تایی از فابل خوانده و در این آرایه ذخیره کنید.
۴) تمامی مقادیر آرایه‌ی سه بعدی را ۱۰۰ عدد اضافه کنید.
۵) فایل جدیدی به نام out.bmp ایجاد کرده و نوع باز کردن آن را "wb" انتخاب کنید.
۶) ابتدا مقادیر آرایه‌ی به طول ۱۵۴ را در فایل ذخیره کنید.
۷) مقادیر آرایه‌ی سه بعدی را به همان روش خواندن در فایل جدید ذخیره کنید.
۸) فایل جدید ایجاد شده را مشاهده کنید.

قسمت۶: نتیجه را به دستیاران آموزشی نشان دهید.

موفق باشید