به نام خدا

دانشکده‌ی مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران

مبانی کامپیوتر و برنامه‌نویسی

نیمسال دوم ۹۹-۹۸	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
```

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، 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;
}
```

قسمت۱: قطعه کدهای داده شده را کامل کرده و آن‌ها را به همراه نتایج به دست آمده ، در کادر زیر بنویسید.

کد خواسته شده بصورت زیر خواهد بود:

```
#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"
#include <stdio.h>
#include <stdlib.h>
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);
}

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

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.
۴) 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؟

قسمت۲: موارد خواسته شده را انجام دهید. نتایج به دست آمده و همچنین پاسخ سوالات را در کادر زیر بنویسید.

اگر فایل `input.txt` را حذف کنیم و برنامه را اجرا کنیم آنگاه با `runtime error` مواجه خواهیم شد.

اگر از `fclose` استفاده نکنیم و فایل را نبندیم ، آنگاه تغییرات در فایل ذخیره نمیشود و برای ذخیره تغییرات باید آنرا بست. همچنین اگر فایل را نبندیم و بخواهیم آنرا در مرحله دیباگ حذف کنیم با این ارور مواجه خواهیم شد که نمیتوان یک فایلی رو که بازه پاک کرد.

۳. 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.

۱) 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 ۱۰۰ عبور نخواهد کرد.
۲) طول رشته‌ی درون فایل `input.txt` را تغییر دهید و برنامه را اجرا کنید.

قسمت۳: موارد خواسته شده را انجام دهید. نتایج به دست آمده را در کادر زیر بنویسید.

کد خواسته شده بصورت زیر خواهد بود که به ازای هر تعداد رشته کاراکتر در input.txt ، تمام آنرا معکوس میکند و در فایل "output.txt" (بدون هیچ کاراکتر اضافی و فقط تا پایان فایل) مینویسد:

```
#define ZERO 0
#define ONE 1
#define READ_CHAR_SIZE EOF_function()
#define WRITE_CHAR_SIZE EOF_function()
#define INPUT_TXT_ADDRESS "input.txt"
#define OUTPUT_FILE_ADDRESS "output.txt"
#include <stdio.h>
#include <stdlib.h>
int EOF_function() {
	FILE* input = fopen(INPUT_TXT_ADDRESS, "r");
	int ch = ZERO, count = - ONE;
	while (ch != EOF) {
		ch = fgetc(input);
		count++;}
	return count;
}
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 -
ONE - 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);
}
int main() {
	char* in_order_array = read_input_file();
	write_reversed_array_in_file(in_order_array);
	return 0;
}
```

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(در این مثال خطوط تیره باید حذف شوند). 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 خط‌تیره انتقال دهید.

قسمت۴: روند ذکر شده را طی کرده و نتیجه به دست آمده را در کادر زیر بنویسید.

همانند کد قسمت سوم مراحل را اینبار با استفاده از دستور fseek انجام میدهیم و هرگاه که به خط تیره رسدیم و \n امین خط تیره بود و n زوج بود ، آنگاه یک واحد جلوتر میرویم و کاراکتر بعدی را میخوانیم.

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 به دست آمده را در کادر زیر بنویسید.

```c
#define ZERO 0
#define ONE 1
#define READ_CHAR_SIZE EOF_function()
#define WRITE_CHAR_SIZE EOF_function()
#define INPUT_TXT_ADDRESS "input.txt"
#define OUTPUT_FILE_ADDRESS "output.txt"
#include <stdio.h>
#include <stdlib.h>

int EOF_function() {
	FILE* input = fopen(INPUT_TXT_ADDRESS, "r");
	int ch = ZERO, count = -ONE;
	while (ch != EOF) {
		ch = fgetc(input);
		count++;
	}
	return count;
}

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;
}

void* append_in_file(char* in_order_array) {
	FILE* output = fopen(OUTPUT_FILE_ADDRESS, "a");
	fwrite(in_order_array, sizeof(char), WRITE_CHAR_SIZE, output);
	fclose(output);
}

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

۶. 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.

قسمت۶: قطعه کد داده شده را مطالعه کرده و پس از تحلیل کردن آن ، نتیجه به دست آمده را در کادر زیر بنویسید.

عملکرد کلی کد داده شده به این شکل است که سه تا فایل - پوینتر تعریف میشه و از کاربر خواسته میشه تا آدرس آنها را وارد کند.(درون آرایه های از قبل تعریف شده). پس از وارد کردن آدرس ها ، دو فایل اول برای خواندن و فایل سوم برای نوشتن باز میشود. سپس در یک حلقه if بررسی میشود که اگر هر کدوم از فایل ها وجود ندارد برنامه را به پایان برساند.

در نهایت در یک حلقه while محتوای فایل اول را تا انتهای آن در فایل سوم کپی میکند. و سپس در یک حلقه دیگر محتوای فایل دوم را در فایل سوم میریزد. یعنی کل محتویات فایل اول و در ادامه‌ش کل محتویات فایل دوم در فایل سوم کپی خواهد شد.

۷.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.

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 فایل 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.

قسمت۷: موارد خواسته شده را انجام دهید ، نتایج به دست آمده را در کادر زیر نشان دهید.

حجم فایل جدید نسبت به اولی افزایش یافت.

موفق باشید