Submitted by:

1. 1805066
2. 1805078
3. 1805090
4. 1805102
5. 1805114

20/20

Experiment on file: Opening in various mode and observing its behavior

It is possible to open the currently opened file with same or different FILE pointer and works well. (fflush() is to be used.)

Function behaviors in wb mode:

fputs():

1. fputs works fine in binary mode.

```
FILE *fp = fopen("file.bin", "wb");
char buffer[] = "WB MODE IN FILE I/O\n";
fputs(buffer, fp);
fputs("Added line 1\n", fp);
fputs("Added line 2\n", fp);
fclose(fp);
```

2. It is possible to replace characters using fseek(). It doesn't insert, rather it replaces the characters.

```
FILE *fp = fopen("file.bin", "wb");
char buffer[] = "WB MODE IN FILE I/O\n";
fputs(buffer, fp);
fseek(fp, 20, SEEK_SET);
fputs(" file", fp); //replaces 'FILE' with 'file'
fseek(fp, 0, SEEK_END);
fputs("Added line 1\n", fp);
fputs("Added line 2\n", fp);
fclose(fp);
```

fgets(): It can read only the first string from the file. But, it doesn't work perfectly and wherever the pointer is, it reads only the first line.

fputc() works fine as it can write a single byte in the file.

fgetc() function doesn't work. It returns EOF.

```
FILE *fp = fopen("file.bin", "wb");
fputc('a', fp);
fputc('b', fp);
```

```
    fseek(fp, 0, SEEK_SET);

    printf("%c ", fgetc(fp));
    printf("%c ", fgetc(fp));

    printf("\n%d", ftell(fp));
    fclose(fp);
```

fprintf() function works perfectly as it stores data character by character.

fscanf() function doesn't work. It returns 0.

```
    int n=0;
    FILE *fp = fopen("file.bin", "wb");
    fprintf(fp, "%d", 86 + 2);
    int ret = fscanf(fp, "%d", &n);
    printf("%d %d", ret, n);
    fclose(fp);
```

fwrite() function works perfectly as it is supposed to and returns the quantity of data it writes.

fread() doesn't work and returns 0.

fseek() function works pretty fine.

```
    FILE *fp = fopen("file.bin", "wb");
    int ara[5] = {1, 2, 3, 4, 5}, n=101, data[5], ret;
    ret = fwrite(ara, sizeof(int), 5, fp);
    fseek(fp, sizeof(int), SEEK_SET);
    fwrite(&n, sizeof(int), 1, fp);
    printf("%d\n", ret);
    fseek(fp, 0, SEEK_END);
    ret = fread(data, sizeof(int), 5, fp);
    printf("%d\n", ret);
    fclose(fp);
```

Thus all the write functions perform very well, whereas the reading functions don't work in wb mode.

Unexpected behavior of "r+" mode: At first I printed 0 to 1000 space separated integers in a file. Then I open the file in r+ mode and write "Hello numbers!".

I observed that the string is written at the first of the file. Then there is a long gap. After that numbers are printed. The starting number was 75. 0 to 74 are lost.

I use fseek to set the origin forward and observe that the initial numbers are written. Then the string. And after that a long space and lose of numbers.

Copying a image: Opening a image file in “rb” mode and then reading all the bytes by fgetc() and then opening another file in “wb” mode and putting those bytes by fputc() results in copying the image.

```
FILE *fp1=fopen("image.bmp","rb"),*fp2=fopen("copy.bmp","wb");
    while(1){
        c=fgetc(fp1);
        if(c==EOF)break;
        fputc(c,fp2);
    }
```

QUESTION: Is it possible to add any special effect in the image or resize or crop the image by file operation?

I took the following step to get the negative of the image but I failed.

```
fputc(255-c,fp2);
```

But this approach works,

```
    int c;
    int i=0;
    while(1){
        i++;
        c=fgetc(fp1);
        if(c==EOF)break;
        if(i>60)c=255-c;
        fputc(c,fp2);
    }
```

Binary Facebook: In our practice problem we faced a log in problem. To solve that an approach is comparing all the usernames and the passwords with the input username and password. But if our database is too big then it will take long time to log in. To solve this problem we can use prefix tree. It is an efficient method of string matching. In that case we have to save information of the prefix tree in a file.

QUESTION: Is it possible to save the information of a prefix tree in a file in binary mode and then retrieve those information?

In this purpose following procedure can be taken to read and write the information of a prefix tree from a file in binary mode.

Reading a trie from a file:

```
//call root=trieread(root,fp);
node* trieread(node *root,FILE *fp)
{
    root=new node();
    fread(root,sizeof(node),1,fp);
    for(int i=0;i<26;i++){
        if(root->nxt[i]){
            root->nxt[i]=trieread(root->nxt[i],fp);
        }
    }
    return root;
}
```

Writing a trie in a file:

```
void trieprint(node *root,FILE *fp)
{
    fwrite(root,sizeof(node),1,fp);
    for(int i=0;i<26;i++){
        if(root->nxt[i]){
            trieprint(root->nxt[i],fp);
        }
    }
}
```