1 Файлтай ажиллах

Си хэлийг ашиглан файлтай харьцах нь доорх 3 алхмаас бүтнэ.

1. Файлыг нээх
2. Файлаас унших, бичих бүх ажлыг хийж дуусгах
3. Файлыг хаах

Файлыг хаалгүй орхих нь тухайн файлыг бусад хэрэглэгч хэрэглэх, мэдээлэл дутуу бичих, санах ой зэрэгт асуудал учруулдаг.

1.1 Файл нээх

Файлыг нээхдээ **stdio.h** санд байрлах *fopen* функцийг хэрэглэнэ.

```
fp = fopen(filename, mode);
```

Энд:

fp нь **FILE*** төрлийн хувьсагч (*FILE* төрлийн бүтцийг заах хаяган хувьсагч)

filename нь онгойлгох файлын нэр.

mode Файлыг хэрхэн онгойлгохыг заах *mode* тэмдэгтэн цуваа. Бид файл унших "r", бичих "w" *mode*-уудыг ихэвчлэн хэрэглэнэ.

fopen() функц нь *FILE** төрлийн хаяг буцаах бөгөөд энэ хаяг нь файлтай харьцахад хэрэглэгдэнэ. Хэрэв файл онгойх боломжгүй (онгойлгох зөвшөөрөлгүй, эсвэл тийм файл байхгүй г.м) байвал *NULL* утга буцаана.

```
FILE *fp;
const char *name = "input";
const char *mode = "r";

fp = fopen(name, mode);
if (fp == NULL) {
  printf("%s file-iig ongoilgoh bolomjgui", name);
  exit(1);
}
```

1.2 Файлаас унших эсвэл файлруу бичих

Файлаа онгойлгосны дараа *fscanf()* функцийг ашиглан уншиж, *fprintf()* функцийг ашиглан бичиж болно. Эдгээр хоёр функц яг л *scanf(), printf()* функцүүдтэй адилхан ажиллах ба зөвхөн уншиж, бичиж байгаа файлыг дүрслэх *FILE** төрлийн аргументыг хамгийн эхэнд нэмж өгөх шаардлагатай.

Жишээ нь: a, b хоёр тоог *output* нэртэй файлруу бичвэл.

```
FILE *fp = fopen("output", "w");
fprintf(fp, "%d%d\n", a, b);
```

fscanf(), fprintf() функцүүд нь хэдэн тоо уншиж бичсэнийг буцаадаг. Тэгэхээр дээрх *fprintf()* 2-ийг буцаана гэсэн үг.

Хэрэв тухайн файлын төгсгөл хүртэл тэмдэгтүүдийг уншихыг хүсвэл файлын төгсгөл хүрсэн эсэхийг шалгахад *feof()* функцийг хэрэглэнэ.

```
FILE *fp = fopen("input", "r");
char s[100];

while (!feof(fp)) {
  fscanf(fp, "%s", s);

  // unshisan temdegtei heregleh
}
```

1.2.1 Бусад функцүүд

1. *fread(), fwrite()* функцүүд нь төрөл харгалзахгүйгээр ямар ч санах ойд байгаа мэдээллийг яг тэр чигээр нь файлруу бичдэг.

   ```
   int fread(void *buf, int size, int count, FILE *fp);
   int fwrite(void *buf, int size, int count, FILE *fp);
   ```

   Эдгээр функц нь уншсан/бичсэн элементийн тоо *count*-ийг буцаана. Энд:

   buf Хаанаа өгөгдлийн хадгалах эсвэл хаанаас өгөгдлийн бичих хаяг.

   size Бичих өгөгдлийн нэг нэгжийн хэмжээ.

   count Тус бүр нь нэгж хэмжээтэй хэдэн ширхэг өгөгдөл унших/бичих.

   fp Файл заагч

   Жишээ: 10 сурагчийн мэдээллийн файлд бичих.

   ```
   struct Student a[10];
   //...
   fwrite(a, sizeof(struct Student), 10, fp);
   ```

2. *fseek(FILE *fp, long offset, int origin);* функц нь уншиж/бичиж байгаа файлын байршил заагчийг *origin* байрлалаас эхлэн *offset* байт зайд байрлуулдаг. Ингэснээр файлын яг тэр байрлалаас эхлэн уншиж бичиж болно.

1.3 Файлыг хаах

Тухайн файлыг хэрэглэж дуусаад заавал хаах шаардлагатай. Файлыг хаахдаа *fclose()* функцэд файлын заагчийг дамжуулна.

```
fclose(fin);
```

1.4 Тусгай файл заагч

Хүссэн файлаа онгойлгохоос гадна Си хэл тусгай файл заагчдыг агуулдаг. Эдгээр нь:

stdin Стандарт оролт

stdout Стандарт гаралт

stderr Стандарт алдаа

Эдгээр файлаас уншиж бичих нь жирийн *scanf()*, *printf()* функцүүдийг хэрэглэхтэй адил.

```
fprintf(stdout, "Hello World!\n");
```

2 Санах ой хувиарлах

Си хэлэнд *malloc()* функцийг ашиглан хүссэн хэмжээний санах ойг хуваарилж болдог. Эдгээр хуваарилсан санах ой хаана, ямар функцэд ашигласнаас үл хамааран *free()* функц ашиглан чөлөөлөх хүртэл идэвхтэй байдаг.

Уг функцийг ашиглахдаа хэдэн байт санах ой хуваарилахыг дамжуулж өгнө. Хэрэв амжилттай хуваарилсан тохиолдолд санах ойн хаягийг, үгүй бол *NULL* утгыг буцаана. Уг функц нь *stdlib.h* толгой файлд зарлагдсан байдаг. Жишээ нь:

```
#include <stdio.h>
#include <stdlib.h>
int * read(int n)
{
  int *p;
  p = malloc(sizeof(int) * n);
  if (p == NULL) {
    printf("Sanax oi xvrsengvi!\n");
    exit(1);
  }
  int i;
  for (i = 0; i < n; i++)
    scanf("%d", &p[i]);
  return p;
}
int main()
{
  int *a, i;
  a = read(5);
  for (i = 0; i < 5; i++)
    printf("%d ", a[i]);
  free(a); // sanax oig choloolno
  return 0;
}
```

3 Дасгалууд

3.1 Ангид

1. *input.txt* файл дахь хоёр бүхэл тоог уншиж, нийлбэрийг *output* файлд бич.

2. *n* хэмжээтэй, *value* утгаар дүүргэсэн хүснэгтийг буцаах доорх функцийг хэрэгжүүл. Хүснэгтийг *malloc()* функц ашиглан үүсгэнэ.

```
int *get_array(int n, int value);
```

3. Дээр бичигдсэн *read()* функцтэй адилаар *n* тоо дамжуулахад хуваагчуудыг нь *malloc()*-р хуваарилсан хүснэгтэд хадгалан буцаах доорх функцийг хэрэгжүүл. Хүснэгтийн хамгийн эхний элемент нь нийт хуваагчдын тоог хадгалах ёстой.

```
    int *find_divisors(int n);
```

4. "input.txt"файлын эхний мөрөнд *n* тооны утга, дараагийн мөрөнд *n* ширхэг бүхэл тоо агуулагдаж байгаа бол тэдгээр утгуудыг файлаас уншин нийлбэрийг хэвлэн харуул.

5. *n* ширхэг оюутны мэдээллийг гараас авч *struct Student* төрлийн хүснэгтэд хадгал. Уг мэдээллээ "students.txt"файлд хадгал.

```
struct Student {
    char fname[20], lname[20], id[10];
    float gpa;
};
```

6. Өмнөх дасгалаар үүсгэсэн "students.txt"файлаас оюутны мэдээллийг уншин дэлгэцэнд хэвлэ.

3.2 Гэрт

1. *fname* нэртэй файлаас, файлын төгсгөл хүртэл бүх тоог уншин *malloc()*-р үүсгэсэн хүснэгтэд хадгалан буцаах доорх функцийг хэрэгжүүл. Хүснэгтийн эхний элемент нь хүснэгтэд нийт хэдэн тоо байгааг хадгална.

```
int *get_array_from_file(char fname[]);
```

2. Дараах функцүүдийг хэрэгжүүл. Файлаас, уншиж бичихдээ хоёртын файлаар нээх ёстой бөгөөд *fwrite(), fread()* функцүүдийг хэрэглэгдэнэ.

```
struct Student {
    char fname[20], lname[20], id[10];
    float gpa;
};
typedef struct Student Student;
void read(Student a[], int n);
void print(Student a[], int n);
```

```c
/* a xusnegted baigaa n oyutnii medeellig fname file-d bichne */
void student_write(Student a[], int n, char fname[]);

/* fname file-d baigaa oyutnii medeellig neg neg unshin a xusnegted
   xadgalaad
   xeden oyutnii medeelel unshij awsanaa butsaana.
 */
int student_read(Student a[], char fname[]);
int main()
{
    Student a[100], b[100];
    int n, m;
    /* Oyutnii medeelliig garaas awna */
    scanf("%d", &n);
    read(a, n);
    print(a, n);

    student_write(a, n, "info.dat");

    m = student_read(b, "info.dat");
    print(b, m);
    return 0;
}
```