

Modul 10

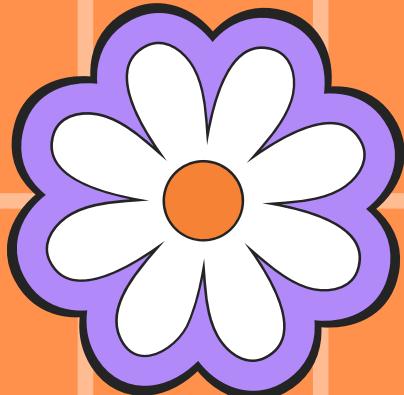
FILE SEKUENSIAL



Aliza Nurfitrian Meizahra - 1101223083



TOPIK UTAMA

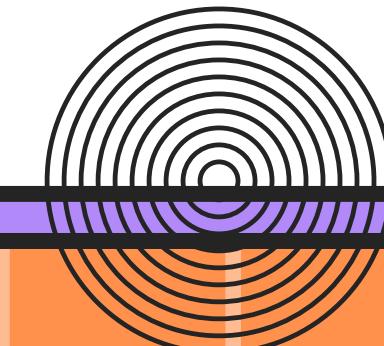
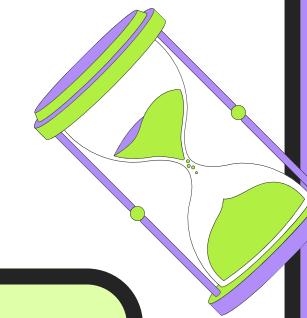
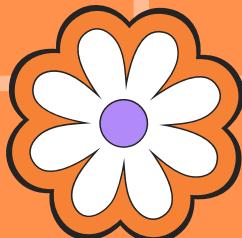


Tujuan Praktikum

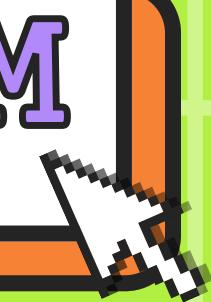
**Apa itu file
sekuensial ?**

**Tahapan operasi
file ?**

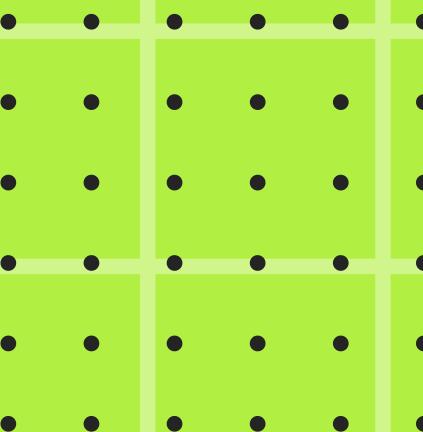
Contoh kasus o_o



TUJUAN PRAKTIKUM



- ★ Memahami bentuk bentuk file di dalam algoritma**
- ★ Memahami penyimpanan data dengan menggunakan file teks dan file biner**
- ★ Dapat membedakan antara file teks dan file biner**
- ★ Dapat membuka file, menutup file, menuliskan data ke file, serta membacanya untuk kedua file teks dan biner dalam algoritma maupun dalam bahasa C**



FILE SEKUENSIAL



File sekuensial adalah tempat penyimpanan yang bersifat permanen, bukan sementara menghilang ketika program ditutup atau dihentikan, yang bertujuan agar data yang sudah tersimpan dapat digunakan kembali.

Struktur FILE atau file handle yaitu tempat penyimpanan semua informasi yang dibutuhkan untuk mengendalikan aliran data suatu file dan dideklarasikan dalam file header standard stdio.h.

FILE SEKUENSIAL

FILE *file_pointer_identifier

FILE *fp

Struktur aliran data:

- Indikator posisi file
- Sebuah pointer buffer (jika ada)
- Sebuah indikator error yang menyimpan ketika proses kesalahan baca/tulis terjadi
- Sebuah indikator end-of-file yang menyimpan ketika akhir dari suatu file telah tercapai





OPERASI FILE

1. MEMBUKA FILE

2. MEMPROSES FILE

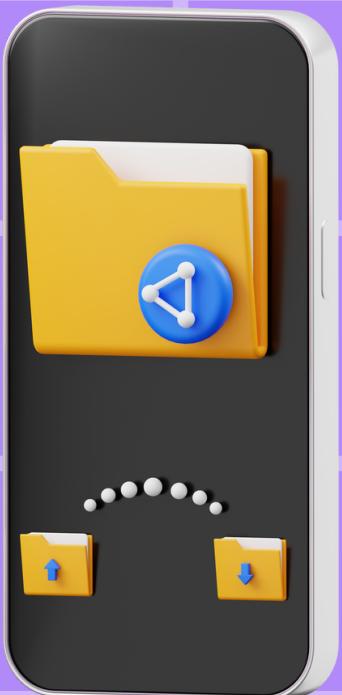
3. MENUTUP FILE

FILE TEKS

FILE BINER

MEMBUKA FILE

```
FILE *fopen(const char *restrict filename, const  
char *restrict mode)
```



- filename, berupa nama dari file (string) yang akan dibuka
- mode, berupa tipe operasi (string) yang akan dilakukan terhadap file
- Untuk selanjutnya bentuk dasar di dalam modul ini terdapat di dalam file stdio.h
- Fungsi tersebut akan mengembalikan pointer yang menunjuk pada objek (FILE) untuk mengendalikan aliran data suatu file. Jika operasi pembukaan file gagal, fopen akan mengembalikan pointer NULL (sebuah konstanta pointer null yang didefinisikan berdasarkan implementasi suatu platform).

Mode fopen



Membuka file untuk melakukan proses 'reading' (pembacaan)



Membuka file untuk melakukan proses 'writing' (penulisan)



Membuka file untuk melakukan proses 'appending' (penambahan)



Sama dengan 'r' akan tetapi mode ini dapat digunakan untuk proses penulisan



Sama dengan 'w' akan tetapi mode ini dapat digunakan untuk proses pembacaan



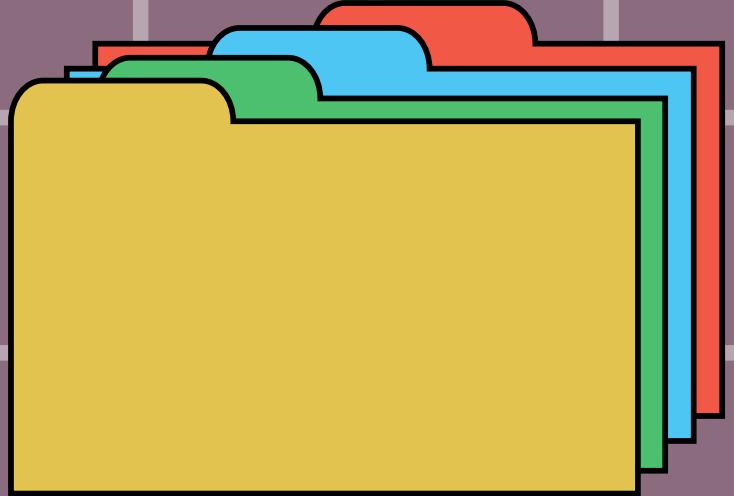
Sama dengan 'a' akan tetapi mode ini dapat digunakan untuk proses pembacaan

fopen

```
fp = fopen ("TES.TXT", "w");
```

- Jika argumen mode terdapat simbol '+', maka dinamakan mode update
- Proses penulisan tidak boleh langsung diikuti oleh proses penulisan tanpa adanya pemanggilan fungsi fflush atau fungsi pemosision file (fseek, fsetpos atau rewind), kecuali proses pembacaan telah mencapai akhir dari suatu file (end-of-file)

MEMPROSES FILE



File teks



file yang datanya akan diproses dalam bentuk karakter, file teks biasanya dipakai untuk menyimpan data bertipe karakter atau string

File biner



file yang datanya diproses dalam bentuk blok-blok memori yang terdiri dari byte, seperti memori pada RAM. Dipakai untuk menyimpan data kompleks, seperti struktur (struct).

File teks

1. Menulis/ membaca data string

mode update dengan r+, w+, a+

- fungsi fprintf --> menulis dilayar
- fungsi fgets --> membaca data string pada file

```
int fprintf(FILE * restrict stream, const char *  
           restrict format, ...);
```

```
char *fgets(char * restrict s, int n,  
           FILE * restrict stream);
```

- s adalah data string. stream adalah pointer FILE.
- n mendefinisikan berapa banyak karakter yang akan disimpan pada string s.
- Fungsi fprintf digunakan untuk menulis string s ke dalam file berdasarkan string yang ditunjuk oleh format untuk menentukan argumen-argumen (...) yang selanjutnya akan dikonversikan menjadi output pada file. Fungsi ini akan mengembalikan jumlah karakter yang ditulis, atau nilai negatif jika output atau error proses pengkodean karakter (encode) pada file terjadi.
- Fungsi fgets digunakan untuk membaca string dari file ke dalam string s. Jika akhir dari file telah tercapai maka s akan tetap tidak berubah dan fungsi akan mengembalikan nilai NULL. Namun jika error terjadi, isi dari string s tidak dapat ditentukan dan fungsi akan mengembalikan nilai NULL, Jika berhasil maka akan mengembalikan string s.



Contoh program

"%s\n" memiliki karakter baris baru (newline) di dalamnya. Karakter baris baru tersebut harus ditulis secara manual ke dalam file, tanpa itu, bukannya membuat file teks yang terdiri lebih dari satu baris dengan benar, file akan ditulis hanya dalam satu baris saja

```
#include <stdio.h>

char string[255];
int main(){
    FILE *f_teks;
    f_teks = fopen("contoh.txt", "w");
    gets(string);
    fprintf(f_teks, "%s\n", string);
    fclose (f_teks);
    return 0;
}
```

Contoh program

```
int fgetc(FILE *stream);
```

EOF adalah sebuah konstanta negatif int yang didefinisikan berdasarkan implementasi suatu platform, nilai ini biasanya dikembalikan oleh beberapa fungsi seperti fgetc untuk mengindikasikan akhir dari suatu file

```
#include <stdio.h>
int karakter;
int main(){

    FILE *f_teks;
    f_teks = fopen("contoh.txt", "r");
    while( (karakter=fgetc(f_teks)) != EOF)
        printf("%c", karakter);
    fclose (f_teks);

    return 0;
}
```

File teks

2. Menambah data

Untuk menambahkan data ke dalam file hampir sama dengan metode untuk menulis data. Perbedaannya terdapat pada argumen filemode di dalam fungsi fopen yang sedang dibuka oleh file.

```
#include <stdio.h>

char string[255];
int main(){

    FILE *f_teks;
    f_teks = fopen("contoh.txt","a");
    gets(string);
    fprintf(f_teks, "%s\n", string);
    fclose (f_teks);

    ...
    return 0;
}
```

Contoh: filemode 'w' atau 'w+' diubah menjadi 'a' atau 'a+'

FILE BINER

1. Menulis dan membaca data

Fungsi untuk menulis tipe data apapun seperti float atau struct

Fungsi yang bernama fread dan fwrite

```
size_t fread(void * restrict ptr, size_t size, size_t nmemb, FILE *  
            restrict stream);
```

```
size_t fwrite(const void * restrict ptr, size_t size, size_t nmemb, FILE *  
            restrict stream);
```

FILE BINER

Fungsi yang bernama fread dan fwrite



```
size_t fwrite(const void * restrict ptr, size_t size, size_t nmemb, FILE *  
             restrict stream);
```



```
size_t fread(void * restrict ptr, size_t size, size_t nmemb, FILE *  
            restrict stream);
```

- ptr adalah sebuah pointer yang menunjuk ke daerah memori yang akan digunakan sebagai tempat penyimpanan data sementara untuk membaca dari file (untuk fread) atau untuk menulis data ke dalam file (untuk fwrite).
- size adalah ukuran dari satu blok memori dalam byte yang akan dibaca atau ditulis
- nmemb adalah jumlah dari blok-blok memori yang akan dibaca atau ditulis dalam satu waktu
- stream adalah file pointer yang menunjuk ke FILE
- size_t, adalah tipe data integer spesial yang dapat menyimpan ukuran dari suatu objek (seperti variabel, array, tipe data dan apa pun yang dapat menyimpan nilai) yang disimpan dalam memori dengan satuan byte. Tipe data ini juga merupakan dari operator sizeof.



Contoh program

```
#include <stdio.h>

int main(){
    FILE *f_struktur;
    int n, i;

    struct {
        char Menu[30];
        double Harga;
    } daftar;

    f_struktur=fopen("Daftar Harga.dat","wb");
    printf("Banyaknya menu: ");
    scanf("%d", &n); getchar();

    for (i=1; i<=n; i++){
        printf("Menu : ");
        gets(daftar.Menu);
        printf("Harga : ");
        scanf("%lf", &daftar.Harga); getchar();
        fwrite(&daftar,sizeof(daftar),1,f_struktur);
    }

    fclose(f_struktur);
    return 0;
}
```

FILE BINER

2. Menambah data

Metode hampir sama dengan metode menulis data

filemode "wb", "wb+", atau "w+b" untuk menulis dan "ab", "ab+", atau "a+b" untuk menambahkan.

Contoh Program

```
#include <stdio.h>

int main(){
    FILE *f_struktur;
    int n, i;

    struct {
        char Menu [30];
        double Harga;
    } daftar;
    f_struktur = fopen("Daftar Harga.dat", "ab");

    printf("Silahkan masukkan banyaknya menu tambahan: ");
    scanf("%d", &n); getchar();

    for (i=1; i<=n; i++){
        printf("Menu : "); gets(daftar.Menu);
        printf("Harga : "); scanf("%lf", &daftar.Harga); getchar();
        fwrite(&daftar, sizeof (daftar), 1, f_struktur);
    }

    fclose(f_struktur);
    return 0;
}
```

FILE BINER

3. Mengakses secara acak

```
int fseek(FILE *stream, long int offset, int whence);
```

- Jika operasi gagal, maka akan mengembalikan nilai selain nol yang didefinisikan berdasarkan implementasi suatu platform
- stream adalah file pointer yang menunjuk ke FILE.
- offset adalah nilai penyesuaian byte terhadap referensi whence.
- whence adalah referensi indikator posisi file.

Note:

Setelah pemanggilan fseek yang berhasil, operasi selanjutnya dalam aliran data update dapat menjadi input (membaca dari file) atau output (menulis ke dalam file).

Referensi whence

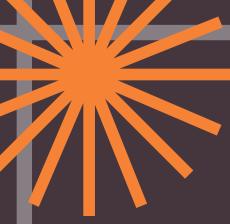
Macro-macro tersebut didefinisikan berdasarkan implementasi suatu platform yang merupakan konstanta bilangan bulat dengan nilai-nilai yang berbeda.

Macro	Referensi Lokasi File
SEEK_SET	Awal file
SEEK_CUR	Indikator posisi file saat ini
SEEK_END	Akhir file

1. fseek(fp, 3, SEEK_SET);	Indikator posisi file akan diubah 3 byte setelah awal dari file.
2. fseek(fp, -3, SEEK_END); <i>Perhatikan tanda negatif.</i>	Indikator posisi file akan diubah 3 byte sebelum akhir dari file.
3. fseek(fp, 3, SEEK_CUR);	Indikator posisi file akan diubah 3 byte setelah indikator posisi file saat ini.
4. fseek(fp, -sizeof(DATA), SEEK_CUR); <i>Perhatikan tanda negatif.</i>	Indikator posisi file akan diubah berdasarkan ukuran dari objek DATA sebelum indikator posisi file saat ini.

void rewind(FILE *stream);

- Fungsi rewind mengubah indikator posisi file untuk aliran data yang ditunjuk oleh stream menjadi bagian awal file. Hal tersebut sama saja dengan (void)fseek(stream, 0L, SEEK_SET) kecuali indikator error pada aliran data juga dihapus.
- Fungsi rewind tidak mengembalikan nilai.



≡

X

Contoh program

```
#include <stdio.h>

int main(){
    FILE *f_struktur;
    int no_struct;
    long int offset_byte;
    struct {
        char Menu [30];
        double Harga;
    } daftar;

    f_struktur = fopen("Daftar Harga.dat", "rb");
    printf("Silahkan masukkan nomor urutan yang ada pada menu: ");
    scanf("%d", &no_struct);
    offset_byte = (no_struct - 1) * sizeof(daftar);
    fseek(f_struktur, offset_byte, SEEK_SET);

    if (fread(&daftar, sizeof (daftar), 1, f_struktur) == 0){
        printf("Menu yang dicari tidak dapat ditemukan pada daftar.\n");
    } else {
        printf("Menu : %s\n",daftar.Menu);
        printf("Harga: %lf\n",daftar.Harga);
    }

    fclose(f_struktur);
    return 0;
}
```



FILE BINER

4. Menghapus data

Prosedur :

- Buka file utama dengan mode baca (read).
- Buatlah file cadangan dan buka file tersebut dengan mode tulis (write).
- Salin semua data dari file utama ke dalam file cadangan, kecuali data yang akan dihapus.
- Tutup file utama dan file cadangan.
- Hapus file utama.
- Ubah nama file file cadangan berdasarkan nama file file utama

```
int remove(const char *filename);
```

Fungsi remove menyebabkan file dengan nama string yang ditunjuk oleh filename, sehingga tidak dapat lagi diakses oleh file dengan nama tersebut. Dengan kata lain, file akan hilang.

```
int rename(const char *old, const char *new);
```

Fungsi rename menyebabkan file dengan nama string yang ditunjuk oleh old untuk kemudian dikenali menjadi nama yang diberikan oleh string yang ditunjuk oleh new.



Contoh program

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

int main(){
    FILE *f_struktur;
    FILE *f_struktur2;

    char dicari[50];

    struct {
        char Menu[30];
        double Harga;
    } daftar;

    f_struktur = fopen("Daftar Harga.dat", "rb");
    f_struktur2 = fopen("Daftar Harga2.dat", "wb");

    printf("Silahkan masukkan nama menu yang akan dihapus: ");
    gets(dicari);
    while (fread(&daftar, sizeof(daftar), 1, f_struktur)==1){
        if (strcmp(daftar.Menu, dicari)!=0){
            fwrite(&daftar, sizeof(daftar), 1, f_struktur2);
        }
    }

    fclose(f_struktur);
    fclose(f_struktur2);
    remove("Daftar Harga.dat");
    rename("Daftar Harga2.dat", "Daftar Harga.dat");

    return 0;
}
```

MENUTUP FILE

int fclose(FILE *stream);

- stream adalah file pointer yang menunjuk ke FILE.
- Jika fungsi fclose berhasil, maka akan menghilangkan aliran data file yang ditunjuk oleh stream dan menutup file yang bersangkutan. Data buffer untuk proses penulisan yang belum ditulis dalam aliran suatu data akan ditulis terlebih dahulu, tetapi data buffer untuk proses pembacaan yang belum dibaca akan dibuang.
- Fungsi fclose mengembalikan nilai nol jika aliran data berhasil ditutup, atau EOF jika ada error yang terdeteksi.





```
int fflush(FILE *stream);
```



Kak kalau untuk
mengosongkan buffer
gimana ?

- stream adalah file pointer yang menunjuk ke FILE.
- Data buffer untuk proses penulisan yang belum ditulis ke dalam aliran suatu data akan ditulis terlebih dahulu.
- Jika operasi terakhir bukan input (membaca dari file), fungsi fflush akan menyebabkan data buffer untuk proses penulisan yang belum ditulis dalam aliran suatu data akan segera ditulis ke dalam file. Selain itu, perilaku dari fungsi ini tidak terdefinisi (undefined).
- Fungsi fflush akan menetapkan indikator error untuk aliran data dan mengembalikan nilai EOF jika error proses penulisan terjadi. Selain itu akan mengembalikan nilai nol.



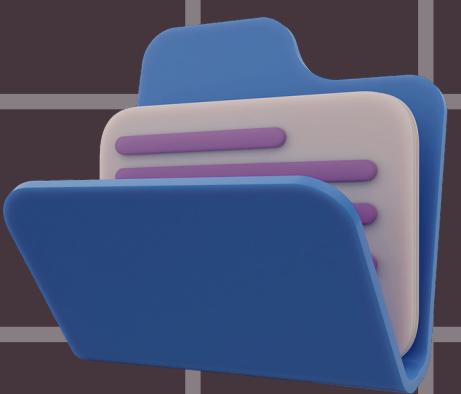
```
int fgetpos(FILE * restrict stream, fpos_t *  
           restrict pos);
```



```
int fsetpos(FILE *stream, const fpos_t  
            *pos);
```



Kak kalau untuk
menulis/
menempatkan
indikator posisi
file ?



- stream adalah file pointer yang menunjuk ke FILE.
- fpos_t adalah tipe data yang dapat menyimpan informasi yang dibutuhkan untuk menentukan setiap posisi secara unik di dalam suatu file.
- fgetpos akan menyimpan indikator posisi file di dalam objek yang ditunjuk oleh pos dan fsetpos akan mengubah indikator posisi file berdasarkan nilai dari objek yang ditunjuk oleh pos.
- Kedua fungsi tersebut akan mengembalikan bilangan bulat nol jika operasi berhasil dan selainnya yang didefinisikan berdasarkan implementasi suatu platform jika gagal

MODUL 10

THANK YOU

FILE SEKUENSIAL