

# PEMROGRAMAN KOMPUTER

C Language Series

## MODUL 5

(Operasi File)

Tim Pengajar Algoritma dan Pemrograman

## Operasi File

---

Nama Mata Kuliah	:	Praktikum Pemrograman Komputer
Bobot sks	:	1 SKS
Semester	:	II (Dua)
Koordinator MK	:	I Made Widiartha, S.Si.,M.Kom
Modul Praktikum	:	Modul 5 – Operasi File

---

### 1. Deskripsi

Pada praktikum ini akan diperlakukan tentang operasi-operasi file pada bahasa pemrograman C.

### 2. Tujuan Umum

Setelah mengikuti praktikum ini, mahasiswa mampu menggunakan berbagai operasi file yang terdapat pada bahasa pemrograman C dalam memecahkan suatu permasalahan.

### 3. Indikator Pencapaian

Adapun indikator pencapaian dari modul praktikum ini adalah mahasiswa dapat mengimplementasikan berbagai operasi file dalam penyelesaian permasalahan-permasalahan yang ada ke dalam sebuah program sederhana.

### 4. Teori

File adalah sebuah organisasi dari sejumlah record. Masing-masing record bisa terdiri dari satu atau beberapa field. Setiap field terdiri dari satu atau beberapa byte.

#### 4.1. Membuka File

Untuk membuka atau mengaktifkan file, fungsi yang digunakan adalah fungsi `fopen()`. File dapat berupa file biner atau file teks. File biner adalah file yang pola penyimpanan di dalam disk dalam bentuk biner, yaitu seperti bentuk pada memori

## Operasi File

---

(RAM) computer. File teks adalah file yang pola penyimpanan datanya dalam bentuk karakter. Penambahan yang perlu dilakukan untuk menentukan mode teks atau biner adalah “t” untuk file teks dan “b” untuk file biner. Prototype fungsi fopen () ada pada header fungsi “stdio.h”

Bentuk umum :

```
file *fopen(char *namafile, char *mode);
```

Keterangan :

- namafile adalah nama dari file yang akan dibuka/diaktifkan.
- mode adalah jenis operasi file yang akan dilakukan terhadap file.

### Jenis-jenis operasi file:

- r : menyarakan file hanya dapat dibaca (file harus sudah ada)
- w : menyatakan file baru akan dibuat/diciptakan (file yang sudah ada akan dihapus)
- a : untuk membuka file yang sudah ada dan akan dilakukan proses penambahan data (jika file belum ada, otomatis akan dibuat)
- r+ : untuk membuka file yang sudah ada dan akan dilakukan proses pembacaan dan penulisan.
- w+ : untuk membuka file dengan tujuan untuk pembacaan atau penulisan. Jika file sudah ada, isinya akan dihapus.
- a+ : untuk membuka file, dengan operasi yang akan dilakukan berupa perekaman maupun pembacaan. Jika file sudah ada, isinya akan dihapus.

Contoh :

```
pf = fopen ("COBA.TXT", "w");
```

## Operasi File

---

### 4.2. Menutup File

Untuk menutup file, fungsi yang digunakan adalah `fclose()`. Prototype fungsi `fclose()` ada di header file “`stdio.h`”.

Bentuk Umum :

```
int fclose(FILE *pf);
atau
int fcloseall(void);
```

### 4.3. Melaksanakan Proses File

#### Menulis Karakter

Untuk menulis sebuah karakter, bentuk yang digunakan adalah :

- ```
putc(int ch, file *fp)
```
- `fp` adalah pointer file yang dihasilkan oleh `fopen()`
  - `ch` adalah karakter yang akan ditulis.

```
#include "stdio.h"
#include "conio.h"

#define CTRL_Z 26

void main(){
    file *pf; // pointer ke file
    char kar;

    // ciptakan file
    if((pf = fopen("COBA.TXT", "w")) == NULL) {
        cputs("File tak dapat diciptakan !\r\n");
        exit(1);
    }

    while((kar=getche()) != CTRL_Z)
        putc(kar, pf); // tulis ke file

    fclose(pf); // tutup file
}
```

## Operasi File

---

### Membaca Karakter

Untuk membaca karakter dari file, fungsi yang digunakan adalah :

```
getc(file *fp);
```

- fp adalah pointer file yang dihasilkan oleh fopen ()
- Fungsi feof (), digunakan untuk mendeteksi akhir file.
- Pada saat membaca data foef(file \*fp)

```
#include "stdio.h"
#include "conio.h"

void main(){

    file *pf; // pointer ke file

    char kar;

    //membuka file
    if((pf = fopen("COBA.TXT", "r")) == NULL){
        cputs("File tak dapat dibuka !\r\n");
        exit(1);
    }

    while((kar=getc(pf)) != EOF)
        putch(kar); // tampilkan ke layar

    fclose(pf); // tutup file
}
```

### Membaca dan Menulis String

Fungsi untuk membaca dan menulis string adalah : fgets () dan fputs ()

Bentuk Umum :

```
fgets(char *str, int p, file *fp)
fputs(char *str, file *fp)
```

### Membaca dan Menulis Blok Data

Fungsi untuk membaca dan menulis blok data adalah : fread() dan fwrite()

Bentuk umum :

```
fread(void *buffer, int b_byte, int c, file *fp);
```

## Operasi File

---

```
fwrite(void *buffer, int b_byte, int c, file *fp);
```

Keterangan :

- buffer adalah pointer ke sebuah area di memori yang menampung data yang akan dibaca dari file.
- b\_byte adalah banyaknya byte yang akan dibaca atau ditulis ke file
- c adalah banyaknya item dibaca/ditulis.

```
#include "stdio.h"
#include "conio.h"

void main(){

    file *f_struktur;
    char jawaban;

    struct data_pustaka{
        char judul[26];
        char pengarang[20];
        int jumlah;
    } buku; // variabel buku bertipe struktur

    // buka file
    if((f_struktur = fopen("DAFBUKU.DAT", "wb")) == NULL){
        cputs("File tak dapat diciptakan !\r\n");
        exit(1);
    }

    do{
        cputs("Judul Buku : ");
        gets(buku.judul);

        cputs("Nama Pengarang : ");
        gets(buku.pengarang);

        cputs("Jumlah buku : ");
        scanf("%i", &buku.jumlah);

        fflush(stdin); // Hapus isi penampung keyboard

        //Rekam sebuah data bertipe struktur
        fwrite(&buku, sizeof(buku), 1, f_struktur);

        cputs("\r\nMau merekam data lagi (Y/T) ?");
        jawaban = getche();

    }while(jawaban == 'Y' || jawaban == 'y');

    fclose(f_struktur);
}
```

## Operasi File

---

### Membaca dan Menulis File yang Terformat

Jika diinginkan, data bilangan dapat disimpan ke dalam file dalam keadaan terformat. Fungsi yang digunakan adalah :

```
fprintf(ptr_file, "string control", daftar argument);
fscanf(pts_file, "string control", daftar argument);
```

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

void main(){

    FILE *pformat;
    char jawaban;

    struct{
        int x;
        int y;
    } koordinat;

    // Buka dan ciptakan file. Periksa kalau gagal dibuka
    if((pformat = fopen("KOORDINAT.TXT", "w")) == NULL){
        cputs("File tak dapat dibuka !\r\n");
        exit(1);
    }

    do{
        cputs("Masukkan data koordinat (bilangan integer)\r\n");
        cputs("Format : posisi x posisi y\r\n");
        cputs("Contoh : 20 30 [ENTER]\r\n");
        scanf("%i %i", &koordinat.x, &koordinat.y);

        fflush(stdin);

        // Rekam ke file
        fprintf(pformat, "%i %i\n", koordinat.x, koordinat.y);

        cputs("\r\nMenyimpan data lagi (Y/T) ??");
        jawaban = getch();
    }while(jawaban == 'y' || jawaban == 'Y');

    fclose(pformat);
    getch();
}
```

## Operasi File

---

### 4.4. File Sequensial

File sekuensial berisi rekord-rekord data yang tidak mengenal posisi baris atau nomor rekord pada saat aksesnya, dan setiap record dapat mempunyai lebar yang berbeda-beda. Akses terhadapnya selalu dimulai dari awal file dan berjalan satu persatu menuju akhir dari file. Dengan demikian, penambahan file hanya dapat dilakukan terhadap akhir file, dan akses terhadap baris tertentu harus dimulai dari awal file.

Fungsi baku yang terkait dengan file sekuensial ini antara lain adalah `fprintf`, `fscanf`, dan `rewind`. Program berikut menyajikan penanganan file sekuensial tentang data nasabah yang berisi tiga field, yaitu nomor identitas (account), nama (name), dan posisi tabungannya (balance) untuk (1) menyajikan yang tabungannya bernilai nol, (2) berstatus kredit, dan (3) berstatus debet. File data tersimpan dengan “nama klien.dat”.

```
#include <stdio.h>

void main() {

    int request, account;
    float balance;
    char name[25];

    FILE *cfPtr;

    if ( (cfPtr = fopen("klien.dat", "r+")) == NULL )
        printf("File could not be opened\n");

    else {

        printf ( "Enter request\n"
        "1 - List accounts with zero balances\n"
        "2 - List accounts with credit balances\n"
        "3 - List accounts with debit balances\n"
        "4 - End of run\n? " );
        scanf( "%d", &request );

        while (request != 4) {
            fscanf (cfPtr, "%d%s%f", &account, name, &balance);

            switch (request){
                case 1:
                    printf ("\nAccounts with zero balances:\n");
                    while ( !feof(cfPtr) ){
                        if (balance == 0)
                            printf ("%10d%-13s%7.2f\n", account, name, balance);
                        fscanf (cfPtr, "%d%s%f", &account, name, &balance);
                    }
                    break;
            }
        }
    }
}
```

## Operasi File

---

```

        case 2:
            printf ("\nAccounts with credit balances:\n");
            while ( !feof(cfPtr) ) {
                if (balance < 0)
                    printf ("%10d%13s%7.2f\n", account, name, balance);
                fscanf (cfPtr, "%d%s%f", &account, name, &balance);
            }
            break;

        case 3:
            printf ("\nAccounts with debit balances:\n");
            while ( !feof(cfPtr) ) {
                if (balance > 0)
                    printf ("%10d%13s%7.2f\n", account, name, balance);
                fscanf (cfPtr, "%d%s%f", &account, name, &balance);
            }
            break;

        }

        rewind(cfPtr);

        printf( "\n? ");
        scanf (" %d", &request);

    }

    printf ("End of run.\n");
    fclose(cfPtr);

}

```

## 5. Referensi

1. Munir, R. 1999. Algoritma dan Pemrograman Dalam Bahasa Pascal dan C. Bandung: Informatika.
2. Kadir, A dan Heriyanto. 2005. Algoritma Pemrograman Menggunakan C++. Yogyakarta: Penerbit Andi.
3. Tosin, R. 1997. Flowchart untuk Siswa dan Mahasiswa. Jakarta: DINASTINDO.

## Operasi File

---

### 6. Pelaksanaan Praktikum

#### 6.1. Tools

Tools yang dapat digunakan pada untuk menunjang praktikum kali ini antara lain:

1. Code :: Block IDE sebagai IDE bahasa pemrograman C

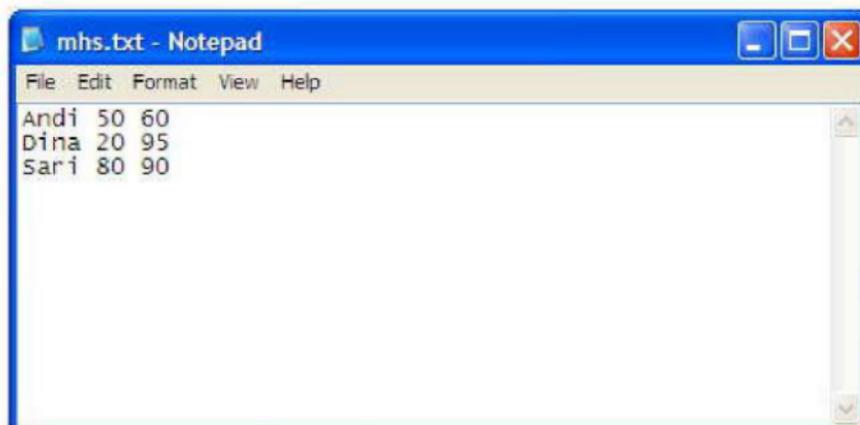
#### 6.4. Langkah-langkah Praktikum

Berikut akan dipaparkan langkah-langkah pembuatan program sederhana dengan menggunakan bahasa pemrograman C.

Studi kasus :

Buatlah sebuah program sederhana (berbasis console) dalam bahasa C yang mampu mengurutkan data nilai siswa secara descending berdasarkan rata-rata nilai pelajaran yang dimiliki. Data "nama" dan "nilai" siswa diimplementasikan ke dalam bentuk struct. Data nilai pelajaran masing-masing siswa terdapat dalam sebuah file berbentuk teks. Program akan meng-load data tersebut untuk diproses sehingga keluaran dari program adalah file berbentuk teks yang berisikan nilai-nilai siswa yang telah diurutkan.

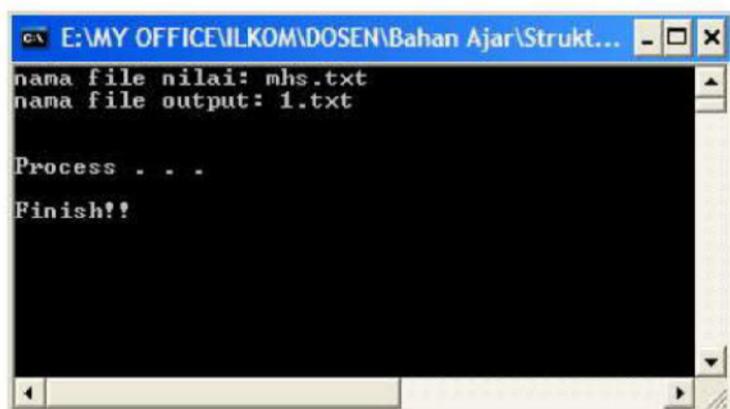
Ilustrasi :



Gambar 1. Daftar Nilai Siswa

## Operasi File

---



Gambar 2. Tampilan Program

The screenshot shows a Notepad window titled '1.txt - Notepad' containing two tables of student data. The first table is labeled 'Nilai tiap siswa (sebelum dilakukan sorting)' and the second is labeled 'Nilai tiap siswa (sesudah dilakukan sorting)'. Both tables have columns for Nama, Math, Bio, and Avg.

| Nama | Math | Bio  | Avg   |
|------|------|------|-------|
| Andi | 50.0 | 60.0 | 55.00 |
| Dina | 20.0 | 95.0 | 57.50 |
| Sari | 80.0 | 90.0 | 85.00 |

| Nama | Math | Bio  | Avg   |
|------|------|------|-------|
| Sari | 80.0 | 90.0 | 85.00 |
| Dina | 20.0 | 95.0 | 57.50 |
| Andi | 50.0 | 60.0 | 55.00 |

Gambar 3. Daftar Nilai Siswa setelah diurutkan (Output Program)

Langkah-langkah pembuatan program :

1. Inisialisasi jumlah siswa

```
#define NUM_OF_STUDENT 3 // banyaknya siswa
```

## Operasi File

---

### 2. Pendefinisan struct untuk nilai siswa

```
struct NILAI {
    char name[100]; // nama
    float math; // nilai math
    float biology; // nilai biology
    float average; // rata-rata nilai math & biology
};
```

### 3. Fungsi untuk membaca file nilai siswa

```
// Loading data

void load_data(FILE *fp, struct NILAI x[])
{
    int i, j;

    // membaca data dari file baris per baris

    for(i=0; i<NUM_OF_STUDENT; i++) {
        if(fscanf(fp, "%s %f %f", &x[i].name, &x[i].math, &x[i].biology) !=3) {
            printf("banyaknya item tidak sesuai\n");
            exit(1);
        }
    }
}
```

### 4. Fungsi untuk mencari nilai rata-rata siswa

```
// menghitung average nilai (nilai math+nilai biology)/2
void average(struct NILAI x[])
{
    int i;

    for(i=0; i<NUM_OF_STUDENT; i++)
        x[i].average= (x[i].math + x[i].biology)/2;
}
```

### 5. Fungsi untuk mengurutkan nilai-nilai siswa

```
void sort(struct NILAI x[])
{
    int i, j;

    struct NILAI tmp;

    for(i=0; i<NUM_OF_STUDENT-1; i++)
        for(j=NUM_OF_STUDENT-1; j>i; j--)
            if( x[j-1].average < x[j].average) {
                tmp=x[j];
                x[j]=x[j-1];
                x[j-1]=tmp;
            }
}
```

## Operasi File

---

6. Fungsi untuk mencetak nilai siswa yang telah diurutkan

```
// Tampilkan nama, nilai math, nilai biology dan average untuk tiap siswa
void print_result(FILE *fp, struct NILAI x[])
{
    int i;
    for(i=0; i<NUM_OF_STUDENT; i++) {
        fprintf(fp, "%s\t| %.1f | %.1f | %.2f\n", x[i].name, x[i].math, x[i].biology, x[i].average);
    }
}
```

7. Inisialisasi variable yang digunakan untuk proses membaca dan menulis sebuah file

```
FILE    *fpi, *fpo;
char    filename[100];
struct  NILAI p[NUM_OF_STUDENT];
```

8. Input nama file yang akan dibaca dan ditulis

```
printf("nama file nilai: "); scanf("%s",filename);

if((fpi=fopen(filename,"r"))==NULL) {
    printf("File %s tidak dapat dibuka \n",filename);
    exit(1);
}

printf("nama file output: "); scanf("%s",filename);

if((fpo=fopen(filename,"w"))==NULL) {
    printf("File %s tidak dapat dibuka \n",filename);
    exit(1);
}
```

9. Proses load data dari file

```
load_data(fpi,p); // membaca file data nilai
```

10. Proses menghitung nilai rata-rata siswa

```
average(p);           // menghitung nilai rata-rata
```

## Operasi File

---

### 11. Proses menulis ke file output

```

fprintf(fpo,"Nilai tiap siswa (sebelum dilakukan sorting)\n\n");
fprintf(fpo,"-----\n");
fprintf(fpo,"Nama\t\t| Math | Bio | Avg |\n");
fprintf(fpo,"-----\n");
print_result(fpo,p); // menampilkan nama, nilai math, nilai biology dan average untuk tiap siswa
fprintf(fpo,"-----\n");
fprintf(fpo,"\n\n"); // ganti baris 2 kali
sort(p); // sorting dilakukan berdasarkan rata-rata nilai tiap siswa
fprintf(fpo,"Nilai tiap sisua (sesudah dilakukan sorting)\n\n");
fprintf(fpo,"-----\n");
fprintf(fpo,"Nama\t\t| Math | Bio | Avg |\n");
fprintf(fpo,"-----\n");
print_result(fpo,p); // menampilkan nama, nilai math, nilai biology dan average untuk tiap sisua
fprintf(fpo,"-----\n");

```

### 12. Menutup file dan finishing proses

```

fclose(fpi);
fclose(fpo);
printf("\n\nProcess");
sleep(500);printf(" .");
sleep(500);printf(" .");
sleep(500);printf(" .");
printf("\n\nFinish!!\n\n");
getch();

```

## 7. Laporan

Buatlah sebuah program yang mampu membaca data dari suatu file yang berekstensi .txt lalu urutkan data pada file tersebut berdasarkan score mereka secara *ascending* dan *descending* dan data hasil pengurutannya disimpan pada suatu file (ascending.txt dan descending.txt)!

Laporan diketik rapih pada kertas A4 dan dikumpulkan pada saat demo program.

Format Penulisan Laporan :

- Font Tulisan : Times New Roman 12 pt
- Font Source Code : Courier New 10 pt
- Line Spacing : 1.5

## Operasi File

---

- Margin : 4, 3, 3, 3 (left, top, right, bottom)
- Page Numbering :
  - a). Cover = tanpa halaman
  - b). Kata Pengantar s/d sebelum BAB I = Center Bottom, angka romawi kecil (i,ii,iii)
  - c). BAB = Center Bottom, angka arab (1, 2, 3)
  - d). Bagian BAB = Top Right, angka arab (1, 2, 3)
- Isi Laporan :
  - BAB I PENDAHULUAN (Latar Belakang, Tujuan, Manfaat)
  - BAB II LANDASAN TEORI
  - BAB III PEMBAHASAN (Tugas Pendahuluan dan Tugas Praktikum)
  - BAB IV PENUTUP (Kesimpulan dan Saran)