BAB IX
PEMROSESAN FILE

Penyimpanan data dalam variabel dan array bersifat sementara, sehingga akan hilang saat program berhenti. File digunakan untuk menyimpan data secara permanen dalam ukuran yang lebih besar. Komputer menyimpan file dalam alat penyimpanan sekunder, misalnya disk, tape, dan sejenisnya.

HIRARKI DATA

Item data terkecil di dalam komputer diasumsikan berupa nilai 0 dan 1 yang disebut dengan **bit** (singkatan dari *binary digit*). Pemrogram bekerja dengan data yang mempunyai bentuk bit pada level rendah, yaitu bentuk digit desimal (0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9), huruf ('A' hingga 'Z', dan 'a' hingga 'z'), dan simbol-simbol khusus ('$', '@', '%' , '*', dan banyak lainnya). Ketiga bentuk tersebut, yaitu desimal, huruf, dan simbol khusus disebut sebagai **karakter**.

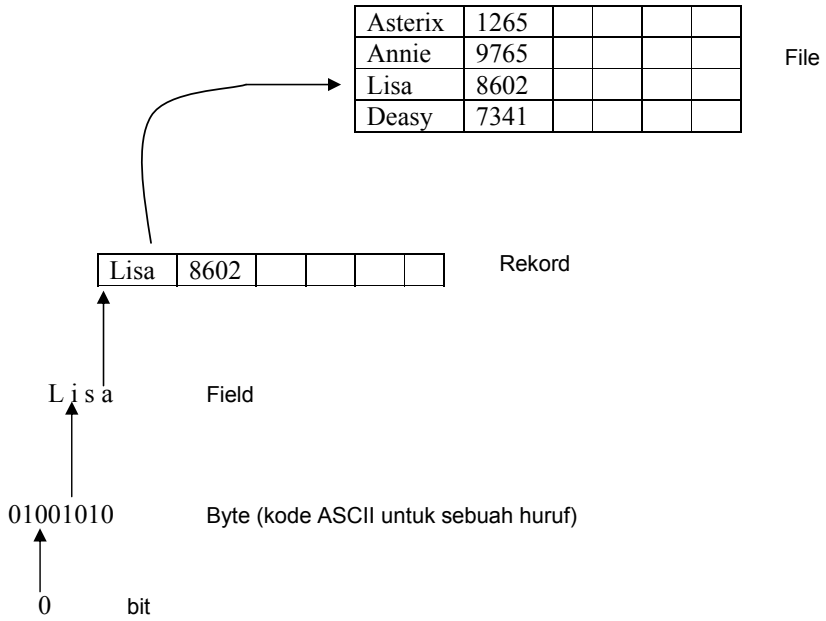
Karakter disusun dari sekumpulan bit yang sering disebut dengan **byte**, dan kumpulan karakter ini menyusun apa yang disebut dengan **field**. Field adalah sekumpulan karakter yang memiliki arti, misalnya field yang terdiri dari huruf besar dan kecil dapat digunakan untuk merepresentasikan nama orang. Dengan demikian, item data diproses oleh komputer sesuai hirarki data sehingga menjadi struktur yang lebih besar dan kompleks, dimulai dari bit, ke karakter (byte), ke field, dan seterusnya.

Rekord (yaitu **struct** dalam C) disusun dari beberapa field. Sebagai contoh, rekord mahasiswa yang didefinisikan sebagai **struct mhs** pada bahasan sebelumnya terdiri dari field-field berikut:

1. Nama mahasiswa
2. Nomor induk mahasiswa (NIM)
3. Nilai ujian tengah semester (UTS)
4. Nilai ujian akhir semester (UAS)
5. Nilai akhir, dan
6. Huruf mutu.

Sekumpulan dari rekord-rekord membentuk apa yang disebut dengan **file**.

Secara garis besar, hirarki data ini dapat digambarkan dengan skema sebagai berikut:



FILE DAN STREAM

C merepresentasikan file secara sederhana sebagai n bytes **stream**. Setiap file mempunyai identifikasi berupa tanda yang menunjukkan akhir dari suatu file. Pada saat membuka suatu file berarti memberi nilai pointer ke struktur **FILE** dalam pustaka baku <stdio.h>. Tiga file dan stream terkait secara otomatis dibuka pada saat program mulai diproses, yaitu *standard input*, *standard output*, dan *standard error*. Masing-masing dari ketiganya dimanipulasi dengan menggunakan pointer file **stdin**, **stdout**, dan **stderr**.

Pustaka baku menyediakan banyak fungsi untuk membaca dan menulis data dari dan ke file, antara lain seperti yang dicantumkan pada bagian di bawah ini.

FILE *fopen(const char *namafile, const char *modus);

merupakan fungsi untuk membuka file bernama *namafile* dengan modus tertentu, yaitu: **r** (membaca data dari file teks), **w** (menulis data ke dalam file teks), **a** (menambah ke file teks), **rb** (membaca file biner), **wb** (menulis data ke dalam file biner), dan **ab** (manambahkan ke file biner).

int fgetc (FILE *stream);

seperti fungsi **getchar**, yaitu membaca satu karakter dari file.

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

membaca string sebanyak n karakter yang ditunjuk oleh pointer **s** dari file.

int fputc (int c, FILE *stream);
seperti fungsi **putchar**, yaitu menuliskan sebuah karakter **c** ke dalam file

int fputs (const char *s, FILE *stream);
menuliskan string yang ditunjuk oleh pointer **s** ke dalam file

int fclose (FILE *stream);
menutup file yang sebelumnya telah dibuka

int fprintf (FILE *stream, const char *format, …..);
menuliskan output ke file dengan kendali string yang ditunjuk oleh pointer format

int fscanf (FILE *stream, const char *format, …..);
membaca input dari file dengan kendali string yang ditunjuk oleh pointer format

Untuk memahami konsep file, perhatikan program di bawah ini dan pelajari kira-kira apa yang dilakukannya.

```
#include <stdio.h>

FILE *in;

void BACA( int[ ] );
void CETAK( int[ ] );

main() {
     int tabel[26] = {0};
     BACA(tabel);
     CETAK(tabel);
     return 0;
}

void BACA ( int huruf[] ) {
     char c;
     if (( in = fopen("data.txt" , "r")) == NULL)
          printf ("File tidak bisa dibaca\n");
          else
          while ( (ch = fgetc(in)) != EOF ) {
               c = ( (( c >= 97) || ( c <= 122)) ? c - 32 : c );
               if ( (c >= 65) || (c <= 90) )
                    ++huruf [ c - 65 ];
          }
          fclose(in);
}

void CETAK ( int huruf[] ) {
     int counter;
     for ( counter = 0 ; counter <= 25 ; counter++ )
          printf ("\n%c%5d", counter + 65, huruf[counter] );
}
```


FILE SEKUENSIAL

File sekuensial berisi rekord-rekord data yang tidak mengenal posisi baris atau nomor rekord pada saat aksesnya, dan setiap rekord 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>

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%-13s7.2f\n", account, name, balance);
                              fscanf (cfPtr, "%d%s%f", &account, name, &balance);
                         }
                         break;

                    case 2:
                         printf ("\nAccounts with credit balances:\n");
                         while ( !feof(cfPtr) ) {
                              if (balance < 0)
                                   printf ("%-10d%-13s7.2f\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%-13s7.2f\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);
        }

        return 0;
    }
```

FILE ACAK (RANDOM)

File acak mempunyai rekord yang masing-masing lebarnya sama dan dapat diakses secara acak, artinya dikenal posisi rekordnya sehingga secara langsung dapat diakses pada nomor rekord tertentu dengan mudah. Berikut adalah fungsi pustaka baku yang terkait dengan manipulasi file acak, disamping pustaka yang telah dicantumkan pada bagian terdahulu.

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

menulis suatu array yang ditunjuk oleh **ptr** hingga **nmemb** elemen yang ukurannya ditentukan oleh **size** (dalam satuan bytes) ke dalam file yang ditunjuk oleh **stream**.

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

membaca data dari file yang ditunjuk oleh **stream**, dimasukkan ke dalam array yang ditunjuk oleh **ptr** hingga nmemb elemen yang ukurannya ditentukan oleh **size** (dalam satuan bytes).

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

menyimpan nilai yang menunjukkan posisi data dalam file yang saat ini ditunjuk oleh **pos**.

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

menetapkan posisi indikator suatu file pada nilai yang ditunjuk oleh **pos**.

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

memindahkan pointer file ke posisi sejauh **whence** dari nilai **offset**. Jika **whence** diisi dengan **SEEK_SET** berarti mulai dari awal file, **SEEK_CUR** adalah posisi saat ini, dan **SEEK_END** adalah akhir dari file.

Berikut adalah contoh program pemrosesan file yang lebih lengkap, untuk menangani data nasabah seperti sebelumnya, yang terdiri dari field-field nomor, nama, dan posisi tabungannya.

```
#include <stdio.h>

struct clientData {
    int acctNum;
    char name[25];
    float balance;
};
typedef struct clientData CLIENT;

int enterChoice(void);
void textFile(FILE *);
void updateRecord(FILE *);
void newRecord(FILE *);
void deleteRecord(FILE *);

main() {
    FILE *cfPtr;
    int choice;

    if ( (cfPtr = fopen("credit.dat", "r+")) == NULL )
        printf("File could not be opened.\n");
    else {
        while ( (choice = enterChoice() ) != 5 ) {
            switch (choice) {
                case 1:
                    textFile(cfPtr); break;
                case 2:
                    updateRecord(cfPtr); break;
                case 3:
                    newRecord(cfPtr); break;
                case 4:
                    deleteRecord(cfPtr); break;
            }
        }
    }
    fclose(cfPtr);
    return 0;
}
```

```
int enterChoice (void) {
    int menuChoice;
    printf  (  "\nEnter your choice\n"
               "1 - store a formatted text file of accounts for printing\n"
               "2 - update an account\n"
               "3 - add a new account\n"
               "4 - delete an account\n"
               "5 - end program\n? ");
    scanf ("%d", &menuChoice);
    return menuChoice;
}

void textFile(FILE *readPtr) {
    FILE *writePtr;
    CLIENT client;

    if ( (writePtr = fopen("account.txt", "w")) == NULL )
        printf ("File could not be opened.\n");
    else {
        rewind(readPtr);
        fprint(writePtr, "%-6d%-25s%10s\n", "Acct", "Name", "Balance");

        while ( !feof(readPtr) ) {
            fread (&client, sizeof(CLIENT), 1, readPtr);
            if (client.Num != 0)
                fprint(writePtr, "%-6d%-25s%10.2f\n",
                    client.acctNum, client.name, client.balance);
        }
    }
    fclose(writePtr);
}

void updateRecord (FILE *fPtr) {
    int account;
    float transaction;
    CLIENT client;

    printf( "Enter account to update (1-100) : " );  scanf ( "%d", &account );

    fseek( fPtr, (account - 1) * sizeof(CLIENT), SEEK_SET);
    fread( &client, sizeof(CLIENT), 1, fPtr);

    if (client.acctNum == 0)
        printf( "Account #%d has no information.\n", account);
    else {
        print( "%-6d%-25s%10.2f\n\n",
            client.acctNum, client.Name, client.balance );
        printf( "Enter charge (+) or payment (-) : " ); scanf( "%f", &transaction );
        client.balance += transaction;
        print( "%-6d%-25s%10.2f\n\n",
            client.acctNum, client.name, client.balance );
        fseek( fPtr, (account - 1) * sizeof(CLIENT), SEEK_SET);
        fwrite( &client, sizeof(CLIENT), 1, fPtr);
    }
}
```



```
void deleteRecord (FILE *fPtr) {
    CLIENT client, blankClient = { 0 , "" , 0 };
    int accountNum;

    printf( "Enter account number to delete (1-100) : " );  scanf( "%d", &accountNum );
    fseek( fPtr, (accountNum - 1) * sizeof(CLIENT), SEEK_SET);
    fread( &client, sizeof(CLIENT), 1, fPtr);

    if ( client.acctNum == 0 )
        printf( "Account #%d does not exist.\n", accountNum);
    else {
        fseek( fPtr, (accountNum - 1) * sizeof(CLIENT), SEEK_SET);
        fwrite( &blankClient, sizeof(CLIENT), 1, fPtr);
    }
}


void newRecord (FILE *fPtr) {
    CLIENT client;
    int accountNum;

    printf( "Enter new account number (1-100) : " );  scanf( "%d", &accountNum );
    fseek( fPtr, (accountNum - 1) * sizeof(CLIENT), SEEK_SET);
    fread( &client, sizeof(CLIENT), 1, fPtr);

    if ( client.acctNum != 0 )
        printf( "Account #%d already contains information.\n", client.acctNum);
    else {
        printf( "Enter name, balance\n? ");
        scanf( "%s%f", &client.name, &client.balance );
        client.acctNum = accountNum;
        fseek( fPtr, (client.acctNum - 1) * sizeof(CLIENT), SEEK_SET);
        fwrite( &client, sizeof(CLIENT), 1, fPtr);
    }
}
```