BAB 12
MANIPULATION FILE

1. Tujuan Instruksional Umum

a. Mahasiswa mampu melakukan perancangan aplikasi menggunakan Struktur File
b. Mahasiswa mampu melakukan analisis pada File yang dibuat
c. Mahasiswa mampu mengimplementasikan File pada sebuah aplikasi secara tepat dan efisien

2. Tujuan Instruksional Khusus

a. Mahasiswa mampu menjelaskan mengenai File
b. Mahasiswa mampu membuat dan mendeklarasikan struktur File
c. Mahasiswa mampu menerapkan dan mengimplementasikan File

PENGANTAR FILE

File digunakan sebagai penyimpanan data eksternal selain memori komputer. Tempat penyimpanan eksternal ini bersifat permanen (non-volatile) dan biasanya berukuran besar dengan tujuan untuk dibaca kembali data-datanya. Operasi-operasi terhadap file pasti berkaitan dengan Input (menulis) dan Output (menampilkan) serta berbagai hal lain seperti mengecek keberadaan file, ukuran file, dan lain-lain. Operasi untuk mengolah file membutuhkan buffer untuk menampung informasi yang berkenaan dengan file tersebut.

JENIS FILE

File Teks adalah file yang berisi data-data ASCII sehingga dapat ditampilkan ke layar apa adanya. Satu karakter ASCII dalam file teks berukuran 2 bytes. Misalnya sebuah file teks berisi 1000 karakter berarti berukuran 2000 bytes.

File Biner yang data-datanya berupa data biner dan berupa byte stream sehingga tidak dapat ditampilkan apa adanya dilayar. Satu karakter berukuran 1 byte, sedangkan nilai bukan karakter akan disimpan sesuai dengan ukuran microprocessor.Ukuran tergantung pada ketentuan microprocessor, bukan pada jumlah digit bilangan.

MANIPULASI FILE DENGAN stdio.h

FILE dalam C dimanipulasi dengan menggunakan header file stdio.h atau fstream.h. Deklarasikan FILE seperti berikut:

```
FILE  *f
```

Berarti menyiapkan buffer area dan di-assign ke pointer bernama f

File Input Output yang digunakan dalam C secara default:

1. stdin = untuk input file standar, yang biasanya diasosiasikan dengan inputan dari keyboard, fungsi scanf dan getchar
2. stdout = untuk output file, biasanya diasosiakan dengan screen monitor, fungsi putchar dan printf
3. stderr = standar error, biasanya diasosiasikan dengan screen, fungsi

```
fprintf(stderr, "error!");
```

FILE adalah sebuah struct yang didefinisikan pada stdio.h sebagai berikut:

```
typedef struct {
unsigned char *curp;     /* Current active pointer */
unsigned char *buffer;   /* Data transfer buffer */
int level;               /* fill/empty level of buffer */
int bsize;               /* Buffer size */
unsigned short istemp;   /* Temporary file indicator */
unsigned short    flags;  /* File status flags  */
short token;             /* Used for validity checking  */
char fd;                 /* File descriptor*/
unsigned  char  hold;        /*  Ungetc  char  if  no  buffer
*/
}FILE;                 /* This is the FILE object     */
```

MEMBUKA FILE

Suatu file dalam disk harus dalam keadaan terbuka terlebih dahulu baru dapat diakses.

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

Kembalian dari fungsi fopen adalah nilai pointer file atau NULL jika pembukaan file gagal, dimana
Filename adalah nama file
Dengan parameter mode adalah:
r - open for reading
w - open for writing (file need not exist)
a - open for appending (file need not exist)
r+ - open for reading and writing, start at beginning
w+ - open for reading and writing (overwrite file)
a+ - open for reading and writing (append if file exists)

File dapat dibuka sebagai file teks (t) atau file biner (b)
File teks menggunakan parameter tambahan menjadi rt, wt, at, r+t, w+t, dan a+t
File biner menggunakan parameter tambahan menjadi rb, wb, ab, r+b, w+b, dan a+b
Defaultnya adalah mode teks (t)

Contoh penggunaan:

```
FILE  *fp;
fp=fopen("c:\\test.txt", "r");
if  (fp==NULL) printf("Error, file tidak dapat dibuka!");
```

MENUTUP FILE

```
int fclose(FILE  *a_file);
int fcloseall(void);
```

Nilai kembalian int akan berisi 0 jika berhasil atau -1 (EOF) jika gagal!
fcloseall akan menutup seluruh stream yang aktif kecuali stdin, stdout, stdprn dan stdaux. Pada Windows, kembalian dari nilai ini tidak terlalu diperlukan karena di Windows tidak diperlukan untuk mengetahui status suatu program. Contoh penggunaan:

```
FILE  *fp;
fp=fopen("c:\\test.txt", "r");
if  (fp==NULL) printf("Error, file tidak dapat dibuka!");
fclose(fp);
```

Contoh program untuk menulis file pada harddisk

```
#include  <stdio.h>
#include  <stdlib.h>
void main(){
FILE  *fp;
if((fp=fopen("C:\\anton.txt","w"))==NULL){
printf("error!");
exit(1);
}
fputs("ABCDE\n",fp);
printf("alamat file  :  %p\n",fp->buffer);
printf("ukuran file  :  %d byte  \n",fp->bsize);
printf("posisi file  :  %p\n",fp->curp);
printf("isi file  :");
for(int i=0;i<=4;i++){
printf("%c",*(fp->buffer+i));
}
printf("\n");
printf("no pengenal file  :  %d\n",fp->fd);
printf("status file  :\n");
if((fp->flags &  1)==1) printf("readonly\n");
if((fp->flags &  2)==2) printf("writeonly\n");
if((fp->flags &  3)==3) printf("read/write\n");
if((fp->flags &  8)==8) printf("file line\n");
if((fp->flags &  16)==16) printf("error\n");
if((fp->flags &  32)==32) printf("akhir file\n");
if((fp->flags &  64)==64) printf("file biner\n");
else printf("file teks\n");
```

```
if((fp->flags &  128)==128)
      printf("data dari file\n");
if((fp->flags &  256)==256)
      printf("data ke file\n");
if((fp->flags &  512)==512)
      printf("file ada diterminal\n");
else
      printf("file di disk");
fclose(fp);
}
```

Untuk menulis ke file dalam format tertentu:

```
int fprintf(fp, "Testing...\n");
```

jika berhasil akan dikembalikan jumlah byte yang dituliskan sedangkan jika gagal dikembalikan EOF Untuk membaca dari file dalam format field tertentu:

```
int fscanf(fp, "Testing...\n");
```

jika berhasil akan dikembalikan jumlah field yang dibaca sedangkan jika gagal dikembalikan EOF Untuk menulis karakter ke file teks:

```
int fputc( int c, FILE  *fp  );
```

jika berhasil akan dikembalikan karakter c sedangkan jika gagal dikembalikan EOF Untuk membaca file teks perkarakter:

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

jika berhasil akan dikembalikan karakter c sedangkan jika gagal dikembalikan EOF Untuk meletakkan nilai integer ke file:

```
int putw(int w, FILE  *fp);
```

jika berhasil akan dikembalikan integer w sedangkan jika gagal dikembalikan EOF Untuk membaca nilai integer:

```
int getw(FILE  *fp);
```

jika berhasil akan dikembalikan integer w sedangkan jika gagal dikembalikan EOF Untuk menulis string ke file tanpa ada karakter NULL dan newline:

```
int fputs(const char  *s,FILE  *fp);
```

jika berhasil akan dikembalikan string s sedangkan jika gagal dikembalikan EOF Untuk membaca string dari file sebanyak n karakter atau bertemu karakter '\n':

```
char  *fgets(const char  *s,int n,FILE  *fp);
```

jika berhasil akan dikembalikan string s sedangkan jika gagal dikembalikan EOF Untuk mengetahui akhir sebuah file stream:

```
int feof(FILE  *fp);
```

jika berhasil akan dikembalikan nilai integer selain 0.

Contoh penggunaan menulisi file dari inputan keyboard menggunakan stdout:

```
#include  <stdio.h>
#include  <stdlib.h>
void main(){
char data[100],*fp;
printf("masukkan teks  : ");
gets(data);
printf("alamat file  :  %p\n",stdin->buffer);
printf("ukuran file  :  %d byte  \n",stdin->bsize);
printf("posisi file  :  %p\n",stdin->curp);
printf("isi file  :");
for(fp=stdin->buffer;fp<stdin->curp;fp++){
printf("%c",*fp);
}
printf("\n");
printf("no pengenal file  :  %d\n",stdin->fd);
printf("status file  :\n");
if((stdin->flags &  1)==1) printf("readonly\n");
if((stdin->flags &  2)==2) printf("writeonly\n");
if((stdin->flags &  3)==3) printf("read/write\n");
if((stdin->flags &  8)==8) printf("file line\n");
if((stdin->flags &  16)==16) printf("error\n");
if((stdin->flags &  32)==32) printf("akhir file\n");
if((stdin->flags &  64)==64) printf("file biner\n");
else printf("file teks\n");
if((stdin->flags &  128)==128) printf("data dari file\n");
if((stdin->flags &  256)==256) printf("data ke file\n");
if((stdin->flags   &       512)==512)   printf("file   ada
diterminal\n"); else
printf("file di disk");
}
```

Contoh program menyalin data dari file a.text ke file b.text:
**)file harus dibuat dan diisi data terlebih dahulu(a.text)*

```
#include  <stdio.h>
int main()
{
char ifilename[]  = "c:/a.txt";
char ofilename[]  = "c:/b.txt";
char name[30];
int idNum;
FILE  *ofp,  *ifp;
/* Open file for input  */
ifp  = fopen(ifilename,"r");
/* Read data  */
fscanf(ifp,"%s  %d",name,&idNum);
/* Open file for output  */
ofp  = fopen(ofilename,"w");
/* Write out data  */
fprintf(ofp,"%d  %s\n",idNum, name);
/* Close Files  */
fclose(ifp);
fclose(ofp);
return  0;
}
```

Contoh program menyalin data dari file a.text ke file b.text:

```
#include  <stdio.h>
int main()
{
char ifilename[]  = "c:/a.txt";
char ofilename[]  = "c:/b.txt";
char name[30];
int idNum;
FILE  *ofp,  *ifp;
/* Open file for input  */
ifp  = fopen(ifilename,"r");
/* Open file for output  */
ofp  = fopen(ofilename,"w");
/* Read and write data  */
while  (fscanf(ifp,"%s  %d",name,&idNum)  != EOF)
{
/* Write out data  */
fprintf(ofp,"%d  %s\n",idNum, name);
}
/* Close Files  */
fclose(ifp);
fclose(ofp);
return  0;
}
```

MANIPULASI FILE DENGAN fstream.h

Membuka File

Sebelum file digunakan/diproses harus dibuka terlebih dahulu. Kita perlu mendefinisikan obyek file. Salah satu bentuk pernyataannya adalah sebagai berikut:

```
ofstream nama_variabel_file;
nama_variabel_file_output.open("nama_file.txt");
```

Menulis ke File

Untuk menulis ke file kita perlu menggunakan nama_variabel file dan tanda "<<" seperti berikut ini:

```
nama_variabel_file_output  <<  "Hallo  "<<endl;
nama_variabel_file_output  <<"Baris kedua  "<<endl;
```

Menutup File

Setelah diproses, file perlu ditutup sebagai berikut:

```
nama_variabel_file_output.close();
```

Contoh program lengkap:

```
#include  <iostream.h>
#include  <fstream.h>
void main(){
ofstream tulis;
tulis.open("c:\\coba.txt");
cout<<"Penulisan dimulai..."<<endl;
tulis<<"hallo"<<endl;
tulis<<"PTIers"<<endl;
cout<<"Penulisan selesai."<<endl;
tulis.close();
}
```

FILE MASUKKAN (INPUTSTREAM)

File masukkan digunakan untuk membaca data yang ada di dalam sebuah File. Membuka file

```
ifstream nama_variabel_file_input;
nama_variabel_file_input.open("nama_file.txt");
```

Contoh pembacaan:

```
#include  <iostream.h>
#include  <fstream.h>
void main(){
ifstream baca("c:\\coba.txt");
const int MAKS  =  80;
```

```
char buf[MAKS+1];
while(baca){
baca.getline(buf,MAKS);
cout<<buf<<endl;
}
baca.close();
}
```

Pendeteksian akhir pada pembacaan inputstream dilakukan dengan cara:

```
while(nama_variabel_file_input){ }
```

Atau sebagai berikut:

```
while(!nama_variabel_file_input.eof()){ }
```

Penambahan Data pada File
Gunakan fungsi :

```
ofstream
nama_variabel_file_output("nama_file.txt",ios::app);
```

Sehingga data dapat ditambahkan pada baris bawah dari akhir file

Perhatikan contoh berikut ini:

```
#include  <iostream.h>
#include  <fstream.h>
void main(){
ofstream tulis;
tulis.open("c:\\coba.txt",ios::app);
cout<<"Penulisan dimulai..."<<endl;
tulis<<"Hallo lagi"<<endl;
tulis<<"PTIers"<<endl;
cout<<"Penulisan selesai."<<endl;
tulis.close();
}
```

LATIHAN

Dibawah ini terdapat sepasang script untuk menuliskan dan membaca hasil proses dari suatu file.

1.Script menulis ke file:

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>

class Kelas
{
private:
    char nama[20];
    char nim[12];
    char email[20];
    char hp[15];
public:
    void entri_kelas();
};

void rekam_kelas(Kelas pti);

void main()
{
Kelas kelas_pti;
rekam_kelas(kelas_pti);
}

void Kelas::entri_kelas()
{
cout<<"===Merekam Data Kelas==="<<endl<<endl;
cout<<"Nama :";
cin.getline(nama, sizeof(nama));
cout<<"NIM  :";
cin.getline(nim, sizeof(nim));
cout<<"Email:";
cin.getline(email, sizeof(email));
cout<<"No hp:";
cin.getline(hp, sizeof(hp));
}

void rekam_kelas(Kelas pti)
{
char jawab;
ofstream file_kelas("KELAS.DAT",ios::app);
for(;;)
{
pti.entri_kelas();
```

```
48  file_kelas.write((char *)&pti, sizeof(pti));
49  cout<<endl<<"Masukkan data lagi(Y/T):";
50  cin>>jawab;
51  if ((jawab=='t')||(jawab=='T'))
52  break;
53  }
54  file_kelas.close();
55  }
```

2.Script membaca file:

```
1   #include<iostream.h>
2   #include<fstream.h>
3   #include<conio.h>
4
5   class Kelas
6   {
7   private:
8       char nama[20];
9       char nim[12];
10      char email[20];
11      char hp[15];
12
13  public:
14      void info_kelas();
15  };
16
17  void baca_kelas(Kelas pti);
18
19  void main()
20  {
21  Kelas kelas_pti;
22  baca_kelas(kelas_pti);
23  }
24
25  void Kelas::info_kelas()
26  {
27  cout<<"Nama:"<<nama<<endl;
28  cout<<"NIM:"<<nim<<endl;
29  cout<<"Email:"<<email<<endl;
30  cout<<"No hp:"<<hp<<endl<<endl;
31  }
32
```

```
33  void baca_kelas(Kelas pti)
34  {
35  ifstream file_kelas("KELAS.txt");
36  cout<<"===Info Data Kelas==="<<endl<<endl;
37  file_kelas.read((char *)&pti, sizeof(pti));
38  while (!file_kelas.eof())
39  {
40  pti.info_kelas();
41  file_kelas.read((char *)&pti, sizeof(pti));;
42  }
43  file_kelas.close();
44  }
45
```

Tugas :

1. Buatlah program untuk memasukkan sebuah nilai integer ke dalam suatu file bernama a.txt dan sebuah nilai integer ke dalam file b.txt, serta kemudian melakukan operasi-operasi seperti penjumlahan, perkalian, pengurangan, dan pembagian dari kedua bilangan tersebut dan dituliskan ke dalam file ketiga c.txt
2. Buatlah program untuk mengkopi isi file dengan pilihan nama sama atau berbeda!