BAB XII
FILE

Membuka / Mengaktifkan File

Sebelum file dapat diakses (dibaca atau ditulisi), mula-mula file harus diaktifkan lebih dulu. Untuk keperluan ini, fungsi yang digunakan yaitu **fopen()**. Bentuk deklarasinya adalah :

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

Keterangan :

namafile : menyatakan nama dari file yang akan diaktifkan

mode : jenis operasi yang akan dilakukan terhadap file

Jenis-jenis operasi adalah sebagai berikut :

r : Menyatakan file hanya akan dibaca, jika file belum ada maka tidak akan berhasil.

w : Menyatakan bahwa file baru diciptakan. Jika file tersebut sudah ada dalam disk, isinya yang lama akan terhapus.

a : Untuk membuka file yang sudah ada untuk ditambah dengan data, jika file belum ada akan dibuat yang baru.

r+ : Sama dengan “r” tetapi selain file dapat dibaca, file juga dapat ditulisi

w+ : Sama dengan “w” tetapi selain file dapat ditulisi, file juga dapat dibaca

Menutup File

Apabila suatu file tidak diproses lagi, file perlu ditutup. Hal ini sangat penting terutama jika melakukan pemrosesan file yang jumlahnya lebih dari satu. Untuk menutup file, fungsi yang digunakan adalah **fclose()**.

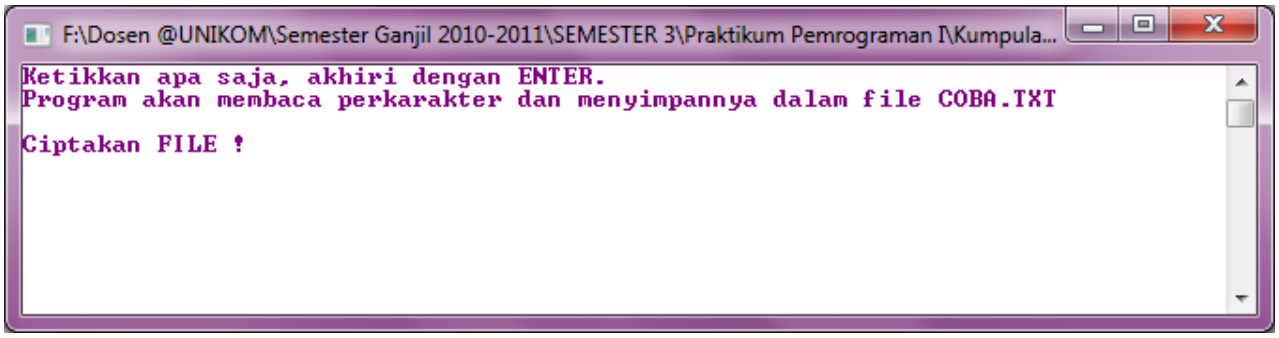
Bentuk deklarasinya adalah :

```
int fclose (FILE *pf);
```

Fungsi **fclose()** menghasilkan keluaran berupa 0 jika operasi penutupan file berhasil dilakukan.

```
1  /*
2   Program 12-1
3   Nama File  : Program 12-1.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12    FILE *pf;
13    char kar;
14    //membuka file
15    pf=fopen("COBA.TXT","a");
16    if(pf==NULL)
17    {
18       printf("File tidak bisa dibuka\n");
19    }
20    printf("Ketikkan apa saja, akhiri dengan ENTER.\n");
21    printf("Program akan membaca perkarakter");
22    printf(" dan menyimpannya dalam file COBA.TXT\n\n");
23    //memasukkan karakter per karakter
24    while((kar = getchar()) != '\n')
25    fputc(kar,pf);
26    //menutup file
27    fclose(pf);
28    system("PAUSE");
29    return 0;
30 }
```

```
1  /*
2    Program 12-1
3    Nama File  : Program 12-1.cpp
4    Programmer : Eko Budi Setiawan
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
10 using namespace std;
11
12 int main(int argc, char *argv[])
13 {
14    FILE *pf;
15    char kar;
16    //membuka file
17    pf=fopen("COBA.TXT","a");
18    if(pf==NULL)
19    {
20       cout<<"File tidak bisa dibuka\n";
21    }
22    cout<<"Ketikkan apa saja, akhiri dengan ENTER.\n";
23    cout<<"Program akan membaca perkarakter";
24    cout<<" dan menyimpannya dalam file COBA.TXT\n\n";
25    //memasukkan karakter per karakter
26    while((kar = getchar()) != '\n')
27    fputc(kar,pf);
28    //menutup file
29    fclose(pf);
30    system("PAUSE");
31    return EXIT_SUCCESS;
32 }
```



Gambar 12.1 Tampilan Gambar 12-1

Operasi Penyimpanan dan Pembacaan File

File dapat diisi dengan data berupa karakter, dimana proses pengaksesan data karakter di file dilakukan dengan dua cara, yaitu penyimpanan dan pembacaan data.

1. *Operasi Penyimpanan Karakter di File*

Sebuah karakter dapat disimpan dalam file dengan menggunakan fungsi fputc(). Bentuk deklarasi dari fungsi ini :

```
int fputc (char kar, FILE *ptr_file);
```

ptr_file adalah pointer ke FILE yang berisi keluaran dari fopen(), dan kar berupa karakter yang akan disimpan dalam file. Jika operasi **puts()** berjalan dengan sempurna maka keluaran fungsi akan sama dengan kar, jika tidak maka keluaran fungsi berupa **EOF** (-1). Hasil yang disimpan di file dengan ekstensi.TXT dapat dilihat dengan notepad.

2. *Operasi Pembacaan Karakter dari File*

Untuk melihat isi file yang telah diisi karakter, selain dapat menggunakan fasilitas notepad, dapat pula menggunakan program yang mempunyai fungsi fgetc(). Fungsi ini digunakan untuk membaca karakter disebuah file. Bentuk deklarasinya adalah sebagai berikut :

```
int fgetc (FILE, FILE *ptr_file);
```

```
1  /*
2   Program 12-2
3   Nama File  : Program 12-2.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12    FILE *pd;
13    char kar;
14    //Buka file
15    pd=fopen("COBA.TXT","r+");
```

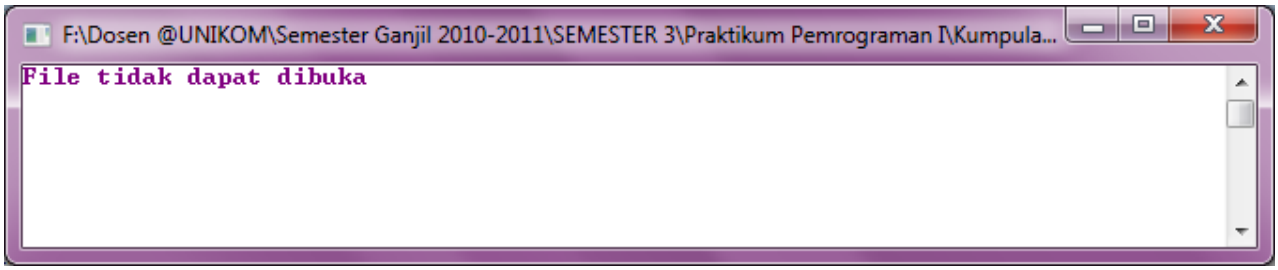
```
   if(pd==NULL)
   {
         printf("File tidak dapat dibuka \n");
   }
   //Baca karakter per karakter sampai ketemu End Of FIle
   while((kar=fgetc(pd)) != EOF)
   fputchar(kar);
   while((kar=getchar()) != '\n')
   fputc(kar,pd);
   fclose(pd); //tutup file
   system("PAUSE");
   return 0;
}
```

```
/*
  Program 12-2
  Nama File  : Program 12-2.cpp
  Programmer : Eko Budi Setiawan
*/

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
   FILE *pd;
   char kar;
   //Buka file
   pd=fopen("COBA.TXT","r+");
   if(pd==NULL)
   {
         cout<<"File tidak dapat dibuka \n";
   }
   //Baca karakter per karakter sampai ketemu End Of FIle
   while((kar=fgetc(pd)) != EOF)
   fputchar(kar);
   while((kar=getchar()) != '\n')
   fputc(kar,pd);
   fclose(pd); //tutup file
   system("PAUSE");
   return EXIT_SUCCESS;
}
```



Gambar 12.2 Tampilan Program 12-2

File Biner dan File Teks

Pada saat file dibuka, file bisa diperlakukan sebagai file biner atau file teks. File biner adalah file yang pola penyimpanan didalam disk berbentuk biner, yaitu seperti bentuk pada memori komputer. Misalnya data bertipe int selalu akan menempati ruang 2 byte, berapapun nilainya.

Sedangkan file teks merupakan file yang pola penyimpanannya dalam bentuk karakter. Bilangan bertipe int bisa saja menempati 1 byte, 2 byte dll, tergantung dari nilai bilangan. Sebagai contoh, bilangan 54 akan disimpan dalam 2 byte (berupa karakter 5 dan 4), tetapi bilangin 123 memerlukan 3 byte. File seperti ini bisa dilihat dalam editor bertipe text (disimpan sebagai file dengan esktensi.TXT).

Penambahan yang perlu dilakukan untuk menentukan mode teks atau biner berupa :

- t untuk mode teks
- b untuk mode biner

Contoh :

"rt" : mode file adalah teks dan file hendak dibaca

"rt+" : mode file adalah teks dan file bisa dibaca dan ditulisi. Bisa juga ditulis : "r+t"

"rb" : mode file adalah biner dan file hendak dibaca

Operasi Penyimpanan dan Pembacaan data Biner

Untuk keperluan menyimpan atau membaca data file bertipe int, C menyediakan fungsi **getw()** dan **putw()**. Bentuk deklarasinya :

```
int putw(intnilai, FILE *ptr_file);
int getw(FILE*ptr_file);
```

Kegunaan :

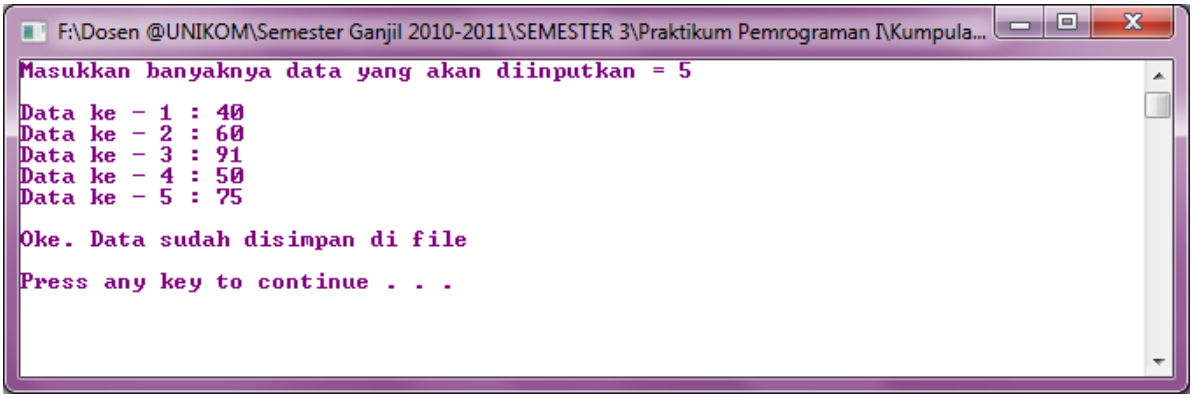
- **getw()** untuk membaca sebuah data bertipe int dari file
- **putw()** untuk menyimpan data bertipe int ke file

Contoh untuk menuliskan data pada file bertipe biner

```
1  /*
2   Program 12-3 Menulis data pada file bertipe biner
3   Nama File  : Program 12-3.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
10 using namespace std;
11
12 int main(int argc, char *argv[])
13 {
14    FILE *pf;
15          int i,data,nilai;
16          pf=fopen("Data.DAT","wb");
17          if(pf==NULL)
18          {
19                   cout<<"File tidak bisa dibuka\n";
20          }
21    cout<<"Masukkan banyaknya data yang akan diinputkan = ";
22    scanf("%d",&data);
23    cout<<"\n";
24    for(i=1;i<=data;i++)
25          {
26                   cout<<"Data ke - "<<i<<" = " ; cin>>nilai;
27                   putw(nilai,pf);
28          }
29    cout<<"\nOke. Data sudah disimpan di file\n\n";
30    fclose(pf);
31    system("PAUSE");
32    return EXIT_SUCCESS;
   }
```

```
1  /*
2   Program 12-3 Menulis data pada file bertipe biner
3   Nama File  : Program 12-3.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <cstdlib>
8  #include <iostream>
```

```
9
10  using namespace std;
11
12  int main(int argc, char *argv[])
13  {
14     FILE *pf;
15           int i,data,nilai;
16           pf=fopen("Data.DAT","wb");
17           if(pf==NULL)
18           {
19                    cout<<"File tidak bisa dibuka\n";
20           }
21     cout<<"Masukkan banyaknya data yang akan diinputkan = ";
22     scanf("%d",&data);
23     cout<<"\n";
24     for(i=1;i<=data;i++)
25           {
26                    cout<<"Data ke - "<<i<<" = " ; cin>>nilai;
27                    putw(nilai,pf);
28           }
29     cout<<"\nOke. Data sudah disimpan di file\n\n";
30     fclose(pf);
31     system("PAUSE");
32     return EXIT_SUCCESS;
33  }
```



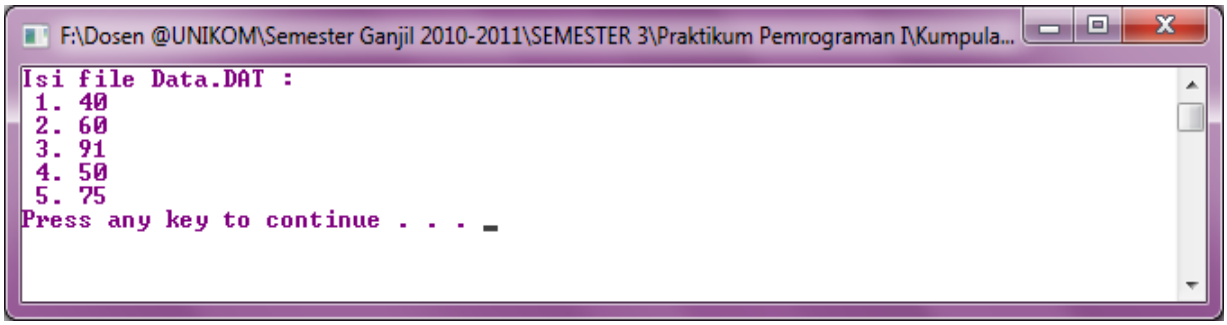
Gambar 12.3 Tampilan Program 12-3

Contoh untuk membaca isi data pada file bertipe biner

```
1  /*
2   Program 12-4 Membaca data pada file bertipe biner
3   Nama File  : Program 12-3.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12    FILE *pf; /* ptr ke file */
13          int nilai, nomor = 0;
14          /* Buka file biner untuk dibaca */
15          if((pf=fopen("Data.DAT","rb")) == NULL)
16          {
17          printf("File gagal dibuka.\n");
18          }
19                    printf("Isi file Data.DAT : \n");
20                    while(1) /* file berhasil dibuka */
21                    {
22                         nilai = getw(pf); /* Baca sebuah int dr file */
23                         if (feof(pf) != 0) break; /*Jika akhir file, keluar loop*/
24                         printf("%2d. %d \n", ++nomor, nilai); //Tampilkan ke layar
25                    }
26                    fclose(pf); /* Tutup file */
27    system("PAUSE");
28    return 0;
29 }
```

```
1  /*
2   Program 12-4 Mebaca data pada file bertipe biner
3   Nama File  : Program 12-4.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
10 using namespace std;
11
12 int main(int argc, char *argv[])
13 {
14    FILE *pf; /* ptr ke file */
15          int nilai, nomor = 0;
16          /* Buka file biner untuk dibaca */
17          if((pf=fopen("Data.DAT","rb")) == NULL)
```

```
18          {
19          cout<<"File gagal dibuka.\n";
20          }
21                  cout<<"Isi file Data.DAT : \n";
22                  while(1) /* file berhasil dibuka */
23                  {
24                      nilai = getw(pf); /* Baca sebuah int dr file */
25                      if (feof(pf) != 0) break; /*Jika akhir file, keluar loop*/
26                      cout<<++nomor<<"."<<nilai<<endl
27                  }
28                  fclose(pf); /* Tutup file */
29
30      system("PAUSE");
31      return EXIT_SUCCESS;
32  }
```



Gambar 12.4 Tampilan Gambar 12-4

Operasi Penyimpanan dan Pembacaan data String

Dua fungsi dipakai dalam penyimpanan dan pembacaan data string pada file, yaitu **fgets()** dan **fputs().** Bentuk deklarasinya :

```
int fputs(char*str, FILE *ptr_file);
char fgets(char*str, int n, FILE *ptr_file);
```

Kegunaan :

- **fputs()** untuk menyimpan string ke dalam file
- **fgets()** untuk membaca strinf dari file sampai ditemukannya karakter baris baru “\n”

Menghapus File

Sebuah file yang sudah terbentuk dapat dihapus secara manual atau melalui program yang dibuat dengan fungsi remove(). Bentuk deklarasinya :

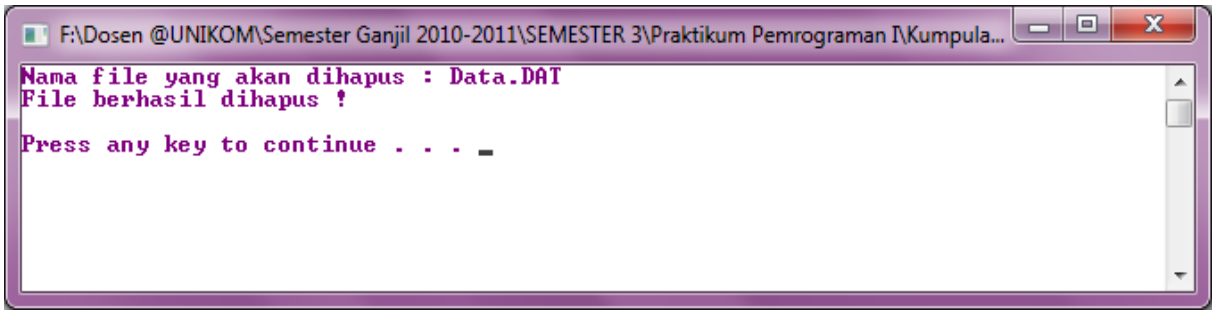
```
int remove (char *namafile);
```

Jika penghapusan file berhasil, akan didapatkan output = 0.

```
1  /*
2   Program 12-5 Menghapus file
3   Nama File  : Program 12-5.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #define PJG 65
10 int main(int argc, char *argv[])
11 {
12    int kode;
13          char namafile[PJG];
14          printf("Nama file yang akan dihapus : ");
15          gets(namafile);
16          kode = remove(namafile);
17          if(kode == 0)
18                printf("File berhasil dihapus !\n\n");
19          else
20                printf("Gagal dalam menghapus file\n");
21    system("PAUSE");
22    return 0;
23 }
```

```
1  /*
2   Program 12-5 Menghapus file
3   Nama File  : Program 12-5.c
4   Programmer : Eko Budi Setiawan
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9  #define PJG 65
10
11 using namespace std;
12
13 int main(int argc, char *argv[])
14 {
```

```
15          int kode;
16          char namafile[PJG];
17          cout<<"Nama file yang akan dihapus : ";
18          gets(namafile);
19          kode = remove(namafile);
20          if(kode == 0)
21                  cout<<"File berhasil dihapus !\n\n";
22          else
23                  cout<<"Gagal dalam menghapus file\n";
24    system("PAUSE");
25    return EXIT_SUCCESS;
26  }
```



Gambar 12.5 Tampilan Program 12-5

Mengganti nama File

Sebuah file yang sudah terbentuk dapat dihapus secara manual atau melalui program yang dibuat dengan fungsi remove(). Bentuk deklarasinya :

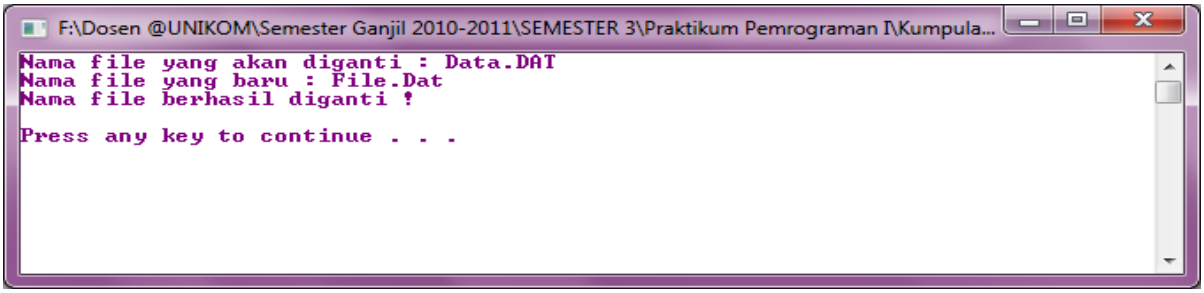
int rename(char*namafilelama, char *namafilebaru);

Jika operasi penggantian berhasil, akan diberikan output = 0.

```
1   /*
2    Program 12-6 Menghapus nama file
3    Nama File  : Program 12-5.c
4    Programmer : Eko Budi Setiawan
5   */
6   #include <stdio.h>
7   #include <stdlib.h>
8   #define PJG 65
9
10  int main(int argc, char *argv[])
11  {
12     int kode;
13     char namafilelama[PJG], namafilebaru[PJG];
```

```
14      printf("Nama file yang akan diganti : ");
15      gets(namafilelama);
16      printf("Nama file yang baru : ");
17      gets(namafilebaru);
18      kode = rename(namafilelama, namafilebaru);
19      if(kode == 0)
20                      printf("Nama file berhasil diganti !\n \n");
21      else
22            printf("Gagal dalam mengganti nama\n");
23      system("PAUSE");
24      return 0;
25  }
```

```
1   /*
2    Program 12-6 Mengganti nama File
3    Nama File  : Program 12-5.c
4    Programmer : Eko Budi Setiawan
5   */
6   #include <cstdlib>
7   #include <iostream>
8   #define PJG 65
9
10  using namespace std;
11  int main(int argc, char *argv[])
12  {
13      int kode;
14      char namafilelama[PJG], namafilebaru[PJG];
15      cout<<"Nama file yang akan diganti : ";
16      gets(namafilelama);
17      cout<<"Nama file yang baru : ";
18      gets(namafilebaru);
19      kode = rename(namafilelama, namafilebaru);
20      if(kode == 0)
21              cout<<"Nama file berhasil diganti !\n \n";
22      else
23        cout<<"Gagal dalam mengganti nama\n";
24      system("PAUSE");
25      return EXIT_SUCCESS;
26  }
```



Gambar 12.6 Tampilan Gambar 12-6