PEMROSESAN FILE

TIM DOSEN

OUTLINE

1. Struktur File
2. Tahapan Operasi File
3. Operasi Penyimpanan dan Pembacaan Data
4. File Biner dan File Teks
5. Operasi Penyimpanan dan Pembacaan Data per int
6. Operasi Penyimpanan dan Pembacaan Data per Blok
7. Menyimpan dan Membaca Data String pada File
8. Menyimpan dan Membaca Data yang di Format
9. Pengaksesan File secara Acak
10. Menghapus File
11. Mengganti nama File
12. Stream, stdin dan stdout

BARANG SIAPA BELUM SEMPAT MERASAKAN PAHITNYA MENUNTUT ILMU WALAUPUN SESAAT, DIA HENDAK MENELAN HINANYA KEBODOHAN SEJAUH HIDUPNYA

" Imam Syafi' i.

STRUKTUR FILE

- File adalah Kumpulan data-data yang disimpan dalam disk dalam bentuk suatu kesatuan.
- Suatu file merupakan organisasi dari sejumlah record.
- Masing-masing record dapat terdiri dari satu atau beberapa field dan setiap field terdiri dari satu atau beberapa byte.

TAHAPAN OPERASI FILE

MEMBUKA / MENGAKTIFKAN FILE

MELAKSANAKAN OPERASI FILE

MENUTUP FILE

MEMBUKA DAN MENGAKTIFKAN FILE

- Untuk membuka / mengaktifkan file sebelum dapat diakses, digunakan fungsi open( )
- Penggunaan header file fstream
- Syntax : open(“namafile”, mode);

  ket :
  - namafile berupa nama dari file yang akan diaktifkan
  - mode berupa jenis operasi yang dilakukan terhadap file
- Berhasil-tidaknya operasi pengaktifan file dapat dilihat pada keluaran fungsi open ( ). Jika keluaran berupa NULL berarti operasi pengktifan gagal.

JENIS OPERASI (MODE) FILE

- ios::in → Baca input dari file, pasangan dari **ifstream**
- ios::out → Tulis output pada file, pasangan dari **ofstream**
- ios::app → Menambah teks pada akhir file dari **ofstream**

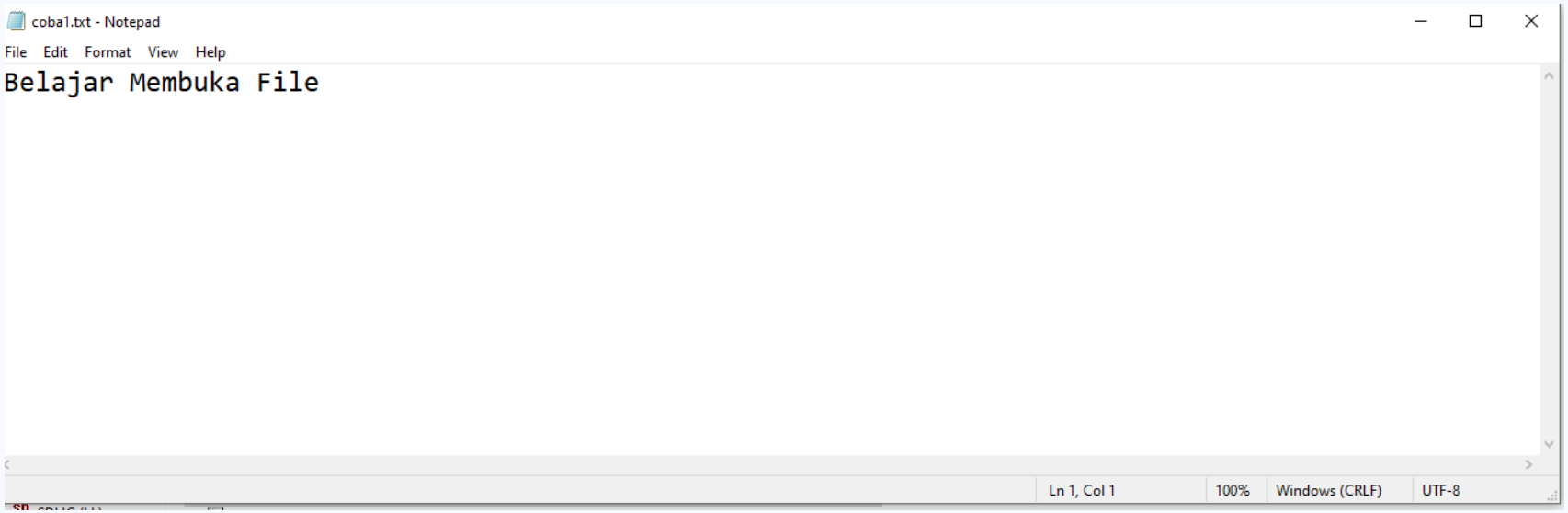
CONTOH PEMAKAIAN FUNGSI *OPEN()*

```
#include <iostream>
#include <conio.h>
#include <fstream>
using namespace std;

int main()
{
	//stream untuk menulis file
	ofstream myfile;

	//membuka file,
	//jika file tidak ditemukan maka file akan otomatis dibuat
	myfile.open("coba.txt", ios::app);

	cout<<"OPERASI FILE 1"<<endl;
	cout<<"--------------"<<endl;

	//fail() -> untuk memeriksa suatu kesalahan pada operasi file
	if(!myfile.fail())
	{
		//menulis ke dalam file
		myfile<<"Membuka FILE"<<endl;
		myfile.close(); //menutup file
		cout<<"Text telah ditulis ke dalam File"<<endl;
	}else{
		cout<<"File tidak ditemukan"<<endl;
	}

      return 0;
}
```

CONTOH PEMAKAIAN FUNGSI *OPEN()*

```
#include <fstream>
using namespace std;

int main () {
  ofstream myfile;
  myfile.open ("coba1.txt", ios::app);
  myfile << "Belajar Membuka File\n";
  myfile.close();
  return 0;
}
```

Output



MEMBACA FILE

```
#include <fstream>
#include <iostream>
using namespace std;
int main () {
       //stream untuk membaca file
       ifstream myfile;
       char teks;
       //membuka file yang telah ada
       myfile.open("Coba1.txt");
       //fail() -> untuk memeriksa suatu pada kesalahan operasi file
       if(!myfile.fail())
       {      cout<<"Isi dari File -> ";
        //ulang selama program belum mencapai akhir konten dari file
              while (!myfile.eof())
              {      myfile.get(teks);
                     cout<<sv_text;
              }
              myfile.close(); //menutup file
       }else{
        cout<<"File tidak ditemukan"<<endl;  }
  return 0;
}
```

REFERENSI

Bab 12, “Text and Binary File Processing”, *Problem Solving and Program Design in C*, Jeri R. Hanly dan Elliot B. Koffman, Addison Wesley, 2002