# Materi 11: I/O File

Dalam pemrograman, operasi-operasi yang dapat dilakukan pada file adalah input/output file dan manipulasi file. Operasi I/O (input/output) melibatkan pembacaan file (input) dari file atau perekaman (output) ke file. Manipulasi file melibatkan operasi terhadap suatu file tanpa melakukan proes input atau output, misalnya mengecek keberadaan file di disk, mengganti nama file, menghapus file, dan sebagainya.

C++ menyediakan class-class berikut untuk melakukan output dan input karakter-kareakter dari/ke file-file.

* ofstream:class untuk menulis karakter ke file-file.
* ifstream:class untuk membaca karakter dari file-file.
* fstream:class untuk menulis dan membaca karakter dari/ke file-file.

Kelas-kelas tersebut berada di library fstream.h

Selanjutnya akan dibahas mengenai operasi-operasi pada file dan implementasinya dalam bahasa C.

* 1. **Membuka file**

1. Membuka file untuk di baca

Untuk membuka file untuk dibaca, pertama kali harus membuat objek dari kelas ifstream, dapat menggunakan sintaks:

**ifstream file\_objek;**

Setelah objek\_input diciptakan, maka file dibuka dengan cara

**file\_objek.open(“nama\_file”);**

Adapun dua pernyataan diatas dapat disederhanakan sebagai berikut:

**ifstream file\_objek(“nama\_file”);**

Contoh:

#include<iostream.h>

#include<fstream.h>

int main()

{

const int MAX = 80;

char buffer[MAX+1];

ifstream file\_objek;

file\_objek.open("latihan.txt");

cout<<"Membaca isi file latihan.txt\n";

while(file\_objek)

{

file\_objek.getline(buffer,MAX);

cout<<buffer<<endl;

}

}

1. Membuka file untuk ditulisi

Untuk membuka file untuk ditulisi, pertama kali harus membuat objek dari kelas ifstream, dapat menggunakan sintaks:

**ofstream file\_objek;**

Setelah objek\_output diciptakan, maka file dibuka dengan cara

**file\_objek.open(“nama\_file”);**

Adapun dua pernyataan diatas dapat disederhanakan sebagai berikut:

**ofstream file\_objek(“nama\_file”);**

Contoh:

#include<iostream.h>

#include<fstream.h>

int main()

{

ofstream file\_objek;

file\_objek.open("latihan.txt");

cout<<"Latihan menulis ke dalam sebuah file\n";

for(int i=1;i<11;i++)

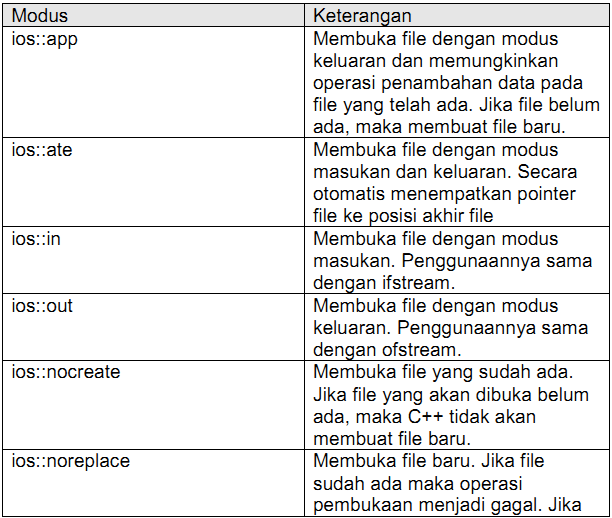
file\_objek<<"Ini adalah baris ke "<<i<<endl;

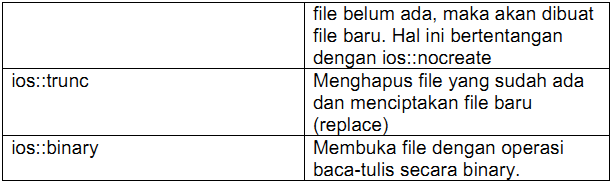
file\_objek.close();

}

1. Membuka File dengan Modus Tertentu

Pada suatu waktu mungkin anda memerlukan cara supaya file yang anda buka dapat dipergunakan untuk membaca dan menulis sekaligus, atau isi file yang sudah ada tidak dihapus jika anda ingin menambah isi file yang baru. Untuk keperluan itu, anda harus memformat modus pembukaan file. Adapun modus pembukaan file yang disediakan oleh C++ adalah sebagai berikut





Adapun contoh penggunaan dari modus – modus pembukaan file diatas adalah sebagai berikut :

**fstream file\_objek (“nama\_file”,ios::in | ios::out);**

Pernyataan diatas adalah dekalarasi file nama\_file dengan sehingga **nama\_file** dapat dibaca dan ditulisi.Contoh:

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

char FirstName[30], LastName[30];

int Age; char FileName[20];

cout << "Enter First Name: "; cin >> FirstName;

cout << "Enter Last Name: "; cin >> LastName;

cout << "Enter Age: "; cin >> Age;

cout << "\nEnter the name of the file you want to create: ";

cin >> FileName;

ofstream Students(FileName, ios::out);

Students << FirstName << "\n" << LastName << "\n" << Age;

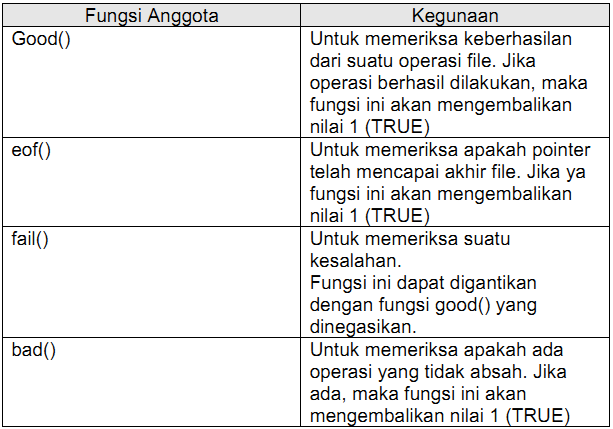
cout << "\n\n";

return 0;

}

1. Memeriksa kondisi file

C++ menyediakan sejumlah fungsi yang berguna untuk memeriksa kondisi-kondisi pada operasi file, sehingga kejadian kesalahan pada saat eksekusi dapat dikendalikan.



* 1. **Menutup file**

Setelah pemrosesan file berakhir, maka file perlu ditutup. Langkah ini dilakukan dengan cara:

**file\_objek.close();**

Pemakaian fungsi close() sifatnya optional. Bila anda tidak mempergunakannya, compiler tidak akan mengeluarkan pesan error.

* 1. **Binary File**

Binary file dapat digunakan untuk menyimpan blok data atau data yang ber-struktur. Misalnya dapat dilihat pada contoh berikut:

Program untuk menuliskan data structure ke binary file

/\* menuliskan data structure ke binary file \*/

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

struct inventory {

char item[20];

int quantity;

double cost;

};

int main()

{

ofstream fout("InvDat.dat", ios::out | ios::binary);

if(!fout) {

cout << "Cannot open file.\n";

}

inventory inv[3];

strcpy(inv[0].item,"A");

inv[0].quantity = 3;

inv[0].cost = 9.99;

strcpy(inv[1].item, "B");

inv[1].quantity = 12;

inv[1].cost = 7.85;

strcpy(inv[2].item, "C");

inv[2].quantity = 19;

inv[2].cost = 2.75;

for(int i=0; i<3; i++)

fout.write((const char \*) &inv[i], sizeof(inventory));

fout.close();

if(!fout.good()) {

cout << "A file error occurred.";

}

system("PAUSE");

return 0;

}

Setelah file biner telah terisi maka dapat dibaca dengan menggunakan objek **read** yang terdapat pada kelas **ifstream.**

Contoh program untuk membaca file biner yang telah terisi dengan kode diatas:

#include <iostream.h>

#include <fstream.h>

#include <conio.h>

using namespace std;

struct inventory {

char item[20];

int quantity;

double cost;

};

int main()

{

ifstream fin("InvDat.dat", ios::in | ios::binary);

if(!fin) {

cout << "Cannot open file.\n";

}

inventory inv[3];

for(int i=0; i<3; i++)

fin.read((char \*) &inv[i], sizeof(inventory));

fin.close();

if(!fin.good()) {

cout << "A file error occurred.\n";

}

for(int i=0; i < 3; i++) {

cout << inv[i].item << "\n";

cout << " Quantity on hand: " << inv[i].quantity;

cout << "\n Cost: " << inv[i].cost << "\n\n";

}

getch();

return 0;

}

Pada file biner juga dapat dilakukan operasi penambahan data (append), contoh:

#include <iostream>  
#include <fstream>  
  
**using**namespace std;  
  
**int**main()  
{  
  ofstream outFile("FileWrite.out");  
  
  **if**(outFile.fail()) {  
    cerr << "Unable to open file for writing." << endl;  
    exit(1);  
  }  
  
  outFile << "Hello!" << endl;  
  
  outFile.close();  
  
  ofstream appendFile("FileWrite.out", ios\_base::app);  
  
  **if**(appendFile.fail()) {  
    cerr << "Unable to open file for writing." << endl;  
    exit(1);  
  }  
  
  appendFile << "Append!" << endl;  
  
  appendFile.close();  
}