PEMROGRAMAN LANJUT

15 | Operasi File

Team Teaching Pemrograman Lanjut

Fakultas Ilmu Komputer UB

Kenapa menggunakan file?

Amati program berikut:

```
main(){
   char nama[30];
   cout << "Input nama Anda : ";
   cin >> nama;
}
```

Penjelasan

- Pada program di atas, nama yang diinput ke dalam variabel akan kehilangan *reference*-nya ketika program selesai. Walaupun data tersebut "mungkin" masih berada di memory, tidak ada lagi cara untuk kita dapat menemukannya.
- Belum lagi bila komputer kita matikan, maka lenyaplah sudah keberadaan data tersebut. Ini dikarenakan data yang kita input tersimpan dalam variabel. Variabel berada di RAM (Random Access Memory) yang bersifat *volatile*, sehingga data akan "menguap" apabila tidak selalu "disegarkan" dengan aliran listrik.

Penjelasan

- Solusinya adalah kita dapat menyimpan data tersebut ke tempat penyimpanan (storage) yang bersifat *non-volatile*, misalnya *hard-disk*.
- C++ menyediakan cara yang mudah untuk menyimpan data ke dalam media penyimpanan selain RAM.
- Media penyimpanan *non-volatile* (kecuali virtual drive) biasanya memiliki nama drive dan path. Kita hanya perlu menyediakan informasi berupa nama drive, path,serta nama file tempat kita menuliskan datanya.

- Ada 2 cara kita dapat mengakses file, yaitu:
  - Menulis (write) ke dalam file
  - Membaca (read) isi dari file

Proses read dan write ditangani oleh 2 macam class yang berbeda. Anda dapat membayangkan class sebagai sebuah structure (record), tetapi di dalam "structure" tersebut dapat kita masukkan definisi function. Kemudian dari class tersebut, kita harus terlebih dahulu membuat object (objek) yang nantinya dapat digunakan untuk proses manipulasi file. Objek mirip dengan variabel structure.

Menulis dan Membaca File

- Ada 2 cara kita dapat mengakses file, yaitu:
  - Menulis (write) ke dalam file
  - Membaca (read) isi dari file

- Class-class tersebut adalah:
  - **`ofstream`** → untuk menulis ke file
  - **`ifstream`** → untuk membaca isi file
- Kedua class ini definisinya berada di dalam file header **`fstream`**.

Membuat objek dari class

```
namaClass    namaObjek;
```

Contoh:

```
ofstream   tulis;
ifstream   baca;
```

"tulis" merupakan objek dari class ofstream dan "baca" merupakan objek dari class ifstream

Fungsi di dalam class ofstream & ifstream

- Di dalam class ofstream dan ifstream terdapat banyak fungsi yang telah disediakan untuk mengakses file.
- Fungsi-fungsi tersebut hanya bisa diakses lewat objek yang telah dibuat dari class tersebut.
- Jadi, kita harus terlebih dahulu membuat objek dari class ofstream atau ifstream, seperti yang telah dijelaskan di atas.

Fungsi di dalam class ofstream & ifstream

- **`open()`** → digunakan untuk membuka/membuat file untuk ditulisi
- **`close()`** → digunakan untuk menutup file yang sebelumnya telah dibuka
- **`getline()`** → membaca isi file per baris
- **`eof()`** → mendeteksi akhir dari file (end of file)
- **`good()`** → mendeteksi keberhasilan dari suatu operasi file
- **`fail()`** → mendeteksi kegagalan hardware, proses baca-tulis, dan eksistensi file
- **`bad()`** → hanya mendeteksi kegagalan hardware dan eksistensi file

Fungsi di dalam class ofstream & ifstream

- **`setiosflags()`** → digunakan untuk menambahkan flag tertentu pada modus akses file
- **`resetiosflags()`** → digunakan untuk menghilangkan flag tertentu
- **`put()`** → digunakan untuk menulis satu karakter ke dalam file
- **`get()`** → digunakan untuk membaca satu karakter dari dalam file
- **`write()`** → digunakan untuk menulis data berformat biner ke dalam file
- **`read()`** → digunakan untuk membaca data dari dalam file yang berformat biner

Flag untuk modus akses file

- Kita dapat menambahkan flag (tanda) untuk akses file dengan proses manipulasi tertentu pada saat membuka file.
- Macam-macam flag tersebut adalah:

ios::app → penulisan isi file akan disisipkan/ditambahkan (append) pada akhir file, sehingga isi file sebelumnya tidak terhapus.

ios::ate → penunjuk file akan langsung ditempatkan pada akhir (at the end) dari file

Flag untuk modus akses file

ios::in → membuka file untuk ditulisi, otomatis di-set bila menggunakan class ifstream

ios::out → membuka file untuk dibaca, otomatis di-set bila menggunakan class ofstream

ios::nocreate → hanya untuk membuka file yang sudah ada, file tidak akan dibuat (no create) bila tidak ada

ios::noreplace → membuat file baru, file lama tidak akan ditimpa (no replace)

Flag untuk modus akses file

ios::trunc → membuat file baru dan menimpa (truncate) file yang lama

ios::binary → membuka file dengan format biner

ios::beg → menempatkan penunjuk file pada awal file

ios::end → menempatkan penunjuk file pada akhir file

ios::cur → penunjuk file yang berada pada posisinya saat ini (current)

Kita bisa menggabung beberapa flag dengan menggunakan operator bit-wise “OR” di C++, yaitu tanda pipeline “|”

Format File

- Secara garis besar format file terbagi menjadi 2, yaitu:
  - File berformat text
  - File berformat biner (binary)
- Kedua format file tersebut memiliki kelebihan dan kekurangannya masing-masing.

Format File

- File berformat text memiliki format yang sederhana sehingga memudahkan kita untuk menulis maupun membaca isi dari file tersebut dengan menggunakan sembarang editor teks.
- Ini menjadikan file tersebut tidak aman karena dapat dibaca oleh setiap orang yang memiliki file tersebut. Proses pencarian juga lambat karena isi dari file tidak memiliki index.

Format File

- File berformat text biner memiliki format yang lebih rumit dibandingkan format file text, sehingga hanya program yang mengetahui format file tersebut saja yang dapat membuka dan mengedit isi file tersebut.
- Ini menjadikan file tersebut aman karena tidak dapat dibaca dengan sembarang program editor. Proses pencarian dapat dilakukan lebih cepat karena isi file ini memiliki index.

Contoh Tulis-Baca File Berformat Text

Tulis File

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
  ofstream tulis;
  tulis.open("coba.txt");
  tulis << "saya mencoba menulis\n"
        << "ke dalam sebuah file teks";
  tulis.close();
}
```

Baca File

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
   ifstream baca;
   baca.open("coba.txt");
   char isi[80];
   while(!baca.eof()){
      baca.getline(isi,80);
      cout << isi << endl;
   }
}
```

Contoh Tulis-Baca File Berformat Biner

Tulis File Biner


```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
  ofstream tulis;
  int isi[] = {1, 2, 3, 4, 5};
  tulis.open("coba.bin", ios::binary);

  for(int i = 0;i < sizeof(isi)/sizeof(int);i++)
    tulis.write((char *)&isi[i],sizeof(int));
  tulis.close();
}
```

Baca File Biner

```
#include <fstream>
#include <iostream>
using namespace std;
main(){
  ifstream baca;
  baca.open("coba.bin", ios::binary);
  int isi;
  while(!baca.eof()){
    baca.read((char *)&isi,sizeof(int));
    if (!baca.eof()) cout << isi << endl;
  }
  baca.close();
}
```

Latihan

1. Buatlah sebuah program yang menampilkan masukan berupa Nama, Nim, Alamat yang mempunyai tipe data String.
2. Simpan hasil masukan user ke dalam sebuah sebuah file
3. Tampilkan Isi dari File
4. Simpan hasil beberapa kali masukan user ke dalam sebuah file. File yang sudah ada isinya akan ditambah di bagian bawah dengan isi yang baru