

STREAM FILE

TIB21

PENDAHULUAN

Stream Input/Output:

- **Alur karakter antara perangkat I/O dengan program:**
 - **Contoh input device: keyboard**
 - **Contoh output device: monitor**

- **Input stream: flow dari input device ke program**
- **Output stream: flow dari output device ke program**
- **Stream I/O bersifat sementara: begitu program mati maka data/nilai akan hilang**
- **Variable dalam program dapat menyimpan data/nilai, tapi bersifat sementara begitu program mati maka nilai yang tersimpan akan hilang**

PENDAHULUAN

//Program TulisNama

//Membaca nama dari keyboard dan menuliskan ke layar

#include <iostream>

using namespace std;

int main ()

{

string nama;

cout << "Tuliskan namamu: " << endl;

cin >> nama ;

cout << "Namamu adalah : " << nama << endl;

return 0;

}

Contoh
Variabel

Stream Input

Stream Output

PENDAHULUAN

- Pada banyak kasus dibutuhkan agar nilai input/output atau nilai variable disimpan sehingga masih dapat dipakai walaupun program selesai
 - Untuk itu digunakan **file [eksternal]**
- File:
 - Bentuk penyimpanan **eksternal** dalam suatu media penyimpanan, misalnya ***harddisk***
 - bentuk penyimpanan sementara (untuk data variable dan stream I/O) adalah **memory**

FILE TEKS DAN FILE BINER

- File Teks:
 - File yang isinya bisa dibaca dan dibuat langsung dengan menggunakan editor teks biasa
 - Contoh editor teks biasa: notepad, wordpad
- File biner (*binary file*):
 - File yang memiliki format khusus yang hanya bisa dibaca dengan program khusus
 - Contoh: Coba buka file *.pdf dengan editor teks biasa. Apa yang terlihat?
- Yang akan digunakan pada kuliah ini hanya **file teks**

NAMA FISIK VS NAMA LOGIKA

Dalam program setiap file mempunyai 2
nama:

➤ Nama fisik : nama file dalam media penyimpanan

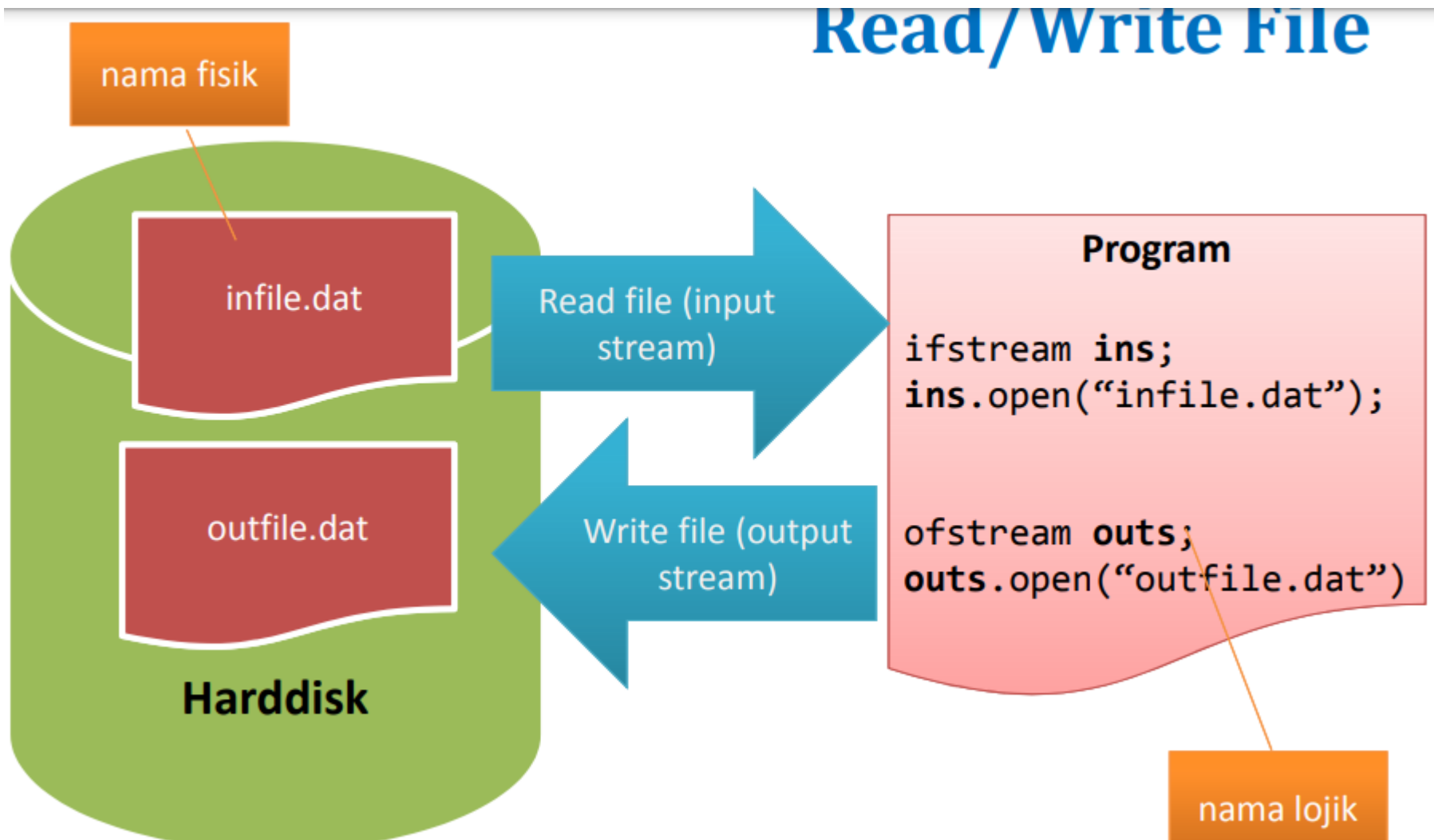
Contoh: myfile.txt, mydata.dat

Nama Logika : nama variabel yang digunakan untuk

➤ Nama Logika : nama variabel yang digunakan untuk
menggantikan nama fisik file dalam program

- ifstream : type variable untuk membaca input stream dari file
- ofstream: type variable untuk menuliskan output stream ke file

NAMA FISIK VS NAMA LOGIKA



CONTOH

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

```
int main () {
```

```
    //KAMUS
```

```
    ifstream in_stream;
```

```
    ofstream out_stream;
```

```
    //ALGORITMA
```

```
    in_stream.open("infile.dat");
```

```
    out_stream.open("outfile.dat");
```

```
    ...
```

```
    in_stream.close();
```

```
    out_stream.close();
```

```
}
```

Contoh Variable untuk
membaca input stream dari
file

Contoh Variable untuk
menuliskan output stream ke
file

Membuka file untuk
membaca input stream dari
file **infile.dat**

Membuka file untuk
menuliskan output stream ke
file **outfile.dat**

Menutup file

PEMROSESAN FILE

- Membuka file
 - Membuka file untuk membaca isinya (read only)
 - Membuka file untuk menulis isinya (rewrite)
- Membaca isi file
- Menulis isi file
- Menutup file
- End of File (EOF)

MEMBUKA FILE UNTUK MEMBACA ISINYA

- Mempersiapkan file untuk dibaca (read-only)
- Input stream dari file ke program

//KAMUS

string FILE_NAME = "infile.txt";

ifstream fin;

// ALGORITMA

fin.open(FILE_NAME); //buka file dengan modus read-only

//sama dengan fin.open("infile.txt")

MEMBUKA FILE UNTUK MENULIS ISI FILE

- Output stream dari program ke file
- Mempersiapkan file untuk siap ditulis (rewrite)
 - Jika file fisik belum ada, file di-create
 - Jika file tidak kosong, maka isi yang lama dihapus dan akan ditimpa dengan isi yang baru

// KAMUS

string FILE_NAME = "outfile.txt";

ofstream fout;

// ALGORITMA

fout.open(FILE_NAME); //buka file dengan modus rewrite

...

MEMBACA ISI FILE

- Membaca data dalam file dan menampung isinya ke suatu variable
- Hati-hati dengan deklarasi variable → type harus sesuai dengan isi file

// KAMUS

```
string FILE_NAME = "infile.txt";
```

```
ifstream fin;
```

```
string s1;
```

```
int i1;
```

// ALGORITMA

```
fin.open(FILE_NAME); //buka file dengan modus read-only
```

```
fin >> s1;
```

```
fin >> i1;
```

```
// bisa disingkat : fin >> s1 >> i1;
```

```
...
```

infile.txt

Hello
123

MENULIS FILE

- Menulis nilai-nilai ke dalam file

// KAMUS

string FILE_NAME = "outfile.txt";

ofstream fout;

// ALGORITMA

fout.open(FILE_NAME); //buka file dengan modus rewrite

fout << "Hello" << endl;

fout << 123;

// bisa disingkat: fout << "Hello" << endl; << 123;

...

outfile.txt

Hello

123

MENUTUP FILE

- Menutup file: file tidak dapat dibaca/ditulis lagi
- Jika **membuka** harus **menutup!!**

```
// KAMUS
ifstream in_stream;
ofstream out_stream;

//ALGORITMA
in_stream.open("infile.dat");
out_stream.open("outfile.dat");

...

in_stream.close();
out_stream.close();
```

Biasakan selalu menulis **close** segera setelah menulis **open!!**
Kode lain sisipkan di antaranya

END OF FILE (EOF)

- Sebuah fungsi yang digunakan untuk menyatakan bahwa pembacaan isi file sudah mencapai akhir file
→ lihat kegunaannya pada pembahasan berikutnya

```
// KAMUS
ifstream in_stream;

//ALGORITMA
in_stream.open("infile.dat");

if (in_stream.eof()) {
    cout << "File kosong" << endl;
} else ...

in_stream.close();
```

`in_stream.eof()` berarti berada di akhir file (artinya sudah tidak ada yang bisa dibaca dari file)



PEMROSESAN FILE

FILE SEKUENSIAL

- File yang dibaca secara sekuensial dari awal sampai akhir:
 - Tidak ada akses di tengah file
 - Akses hanya bisa maju, tidak bisa mundur, atau lompat
- Untuk itu file harus diproses juga secara sekuensial
- Data yang tersimpan dalam file memiliki type yang sama:
 - *file text, file of integer, file of float, dll.*

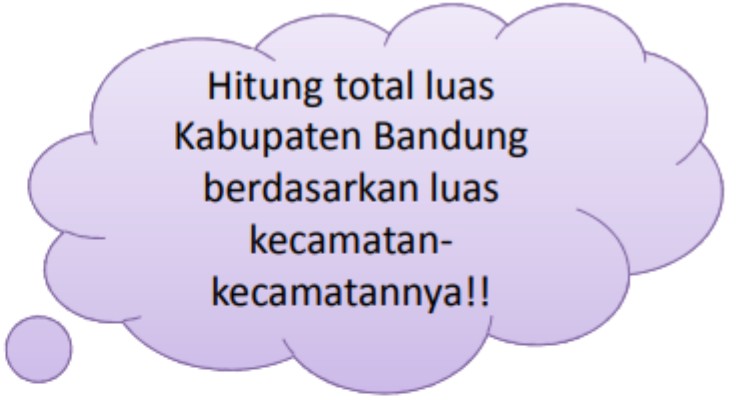
MEMBACA DATA SECARA SEKUENSIAL HINGGA AKHIR FILE

- Pada banyak kasus, program diharapkan membaca data secara sekuensial sampai akhir file, contoh:
 - File berisi nilai mahasiswa satu kelas (skala 0 s.d. 100). harus dihitung rata-rata nilai mahasiswa
 - File berisi luas wilayah setiap kecamatan suatu kabupaten. harus dihitung total wilayah kecamatan untuk mendapatkan luas kabupaten
 - File berisi data tinggi badan pasien. harus dicari pasien yang tertinggi
- Banyaknya data yang tersimpan di file tidak bisa diketahui:
 - File juga mungkin kosong!!

CONTOH

4846.92
14837.01
23957.65
5500.03
19540.93
15207.37
9193.97
5456.52
5102.91
4013.63
3599.23
4930.30
4524.83
2536.46
2400.66
4617.57
4155.54
6497.79
4291.79
2461.06
1462.32

...
1572.46
2550.68
4730.26
1834.50
1054.33
1102.91
2781.23
3157.51
3011.95
5308.34



Hitung total luas
Kabupaten Bandung
berdasarkan luas
kecamatan-
kecamatanannya!!

Data luas kecamatan di Kab. Bandung (2009) dlm. Hektar
(diakses dari http://bapeda.bandungkab.go.id/index2.php?option=com_docman&task=doc_view&gid=79&temid=37 pada 29 Mei 2013)



CONTOH

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

int main () {
// KAMUS
    ifstream in_stream;
    float luas; // variable utk luas kec yg sdg dibaca
    float luaskab; // variable utk luas kabupaten total
// ALGORITMA
    in_stream.open("datakecamatan.dat");

    luaskab = 0; //inisialisasi
    while (!in_stream.eof()) {
        in_stream >> luas;
        luaskab = luaskab + luas;
    }
    cout << "Luas wilayah kabupaten Bandung tahun 2009 adalah
" << luaskab << " hektar";

    in_stream.close();

    return 0;
}
```

Loop akan berhenti, jika sudah sampai pada akhir file, termasuk jika file kosong

MENULIS FILE

```
int main ()  
{  
// KAMUS  
// ALGORITMA  
ofstream out_stream;  
int nr;  
out_stream.open("datanilai.dat");  
cin >> nr;  
while (nr != -999)  
{  
out_stream << nr << endl;  
cin >> nr; }  
out_stream.close();  
Return 0;  
}
```

Data dibaca dari keyboard
sampai pengguna
mengetikkan -999

Contoh isi file:

100
20
12
54
66
67
78
99

FILE RANDOM

Random access files

- Akses individual record tanpa mencari melalui record-record yang lain
- Akses ke record di dalam file secara instant
- Data dapat disisipkan tanpa merusak data yang lain
- Data yang ada dapat disimpan dan dapat diupdate atau dihapus tanpa overwriting

FILE RANDOM

Data yang ada di Random access files

- **Unformatted (stored as "raw bytes")**
- **Semua data yang sama tipenya menggunakan nilai memori yang sama**
- **Semua record data yang sama tipenya memiliki lebar data yang tetap**

FILE RANDOM

Unformatted I/O functions

- **fwrite**
 - Transfer bytes dari sebuah lokasi memori ke suatu file
- **fread**
 - Transfer bytes dari sebuah file ke lokasi di dalam memory
- Example:

```
fwrite( &number, sizeof( int ), 1, myPtr );
```

 - **&number** – Untuk transfer bytes dari sebuah lokasi
 - **sizeof(int)** – jumlah bytes untuk di transfer
 - **1** – For arrays, number of elements to transfer
 - In this case, "one element" of an array is being transferred
 - **myPtr** – File untuk ditransfer transfer ke atau dari

FILE RANDOM

Menulis Structs

```
fwrite( &myObject, sizeof (struct myStruct) ,  
1, myPtr );
```

- **sizeof** – mengembalikan ukuran dalam bentuk bytes

Menulis beberapa element Array

- Pointer to array as first argument
- Number of elements to write as third argument

FILE RANDOM

```
1  /* Fig. 11.11: fig11 11.c
2     Creating a randomly accessed file sequentially */
3  #include <stdio.h>
4
5  struct clientData {
6     int acctNum;
7     char lastName[ 15 ];
8     char firstName[ 10 ];
9     double balance;
10 };
11
12 int main()
13 {
14     int i;
15     struct clientData blankClient = { 0, "", "", 0.0 };
16     FILE *cfPtr;
17
18     if ( ( cfPtr = fopen( "credit.dat", "w" ) ) == NULL )
19         printf( "File could not be opened.\n" );
20     else {
21
22         for ( i = 1; i <= 100; i++ )
23             fwrite( &blankClient,
24                 sizeof( struct clientData ), 1, cfPtr );
25
26         fclose( cfPtr );
27     }
28
29     return 0;
30 }
```

1. Define struct

1.1 Initialize variable

1.2 Initialize struct

2. Open file

2.1 Write to file using unformatted output

3. Close file

FILE RANDOM

fseek

- Sets file position pointer to a specific position
- **fseek** (*pointer*, *offset*, *symbolic_constant*) ;
 - *pointer* – pointer to file
 - *offset* – file position pointer (0 is first location)
 - *symbolic_constant* – specifies where in file we are reading from
 - **SEEK_SET** – seek starts at beginning of file
 - **SEEK_CUR** – seek starts at current location in file
 - **SEEK_END** – seek starts at end of file

```

1  /* Fig. 11.12: fig11_12.c
2     Writing to a random access file */
3  #include <stdio.h>
4
5  struct clientData {
6     int acctNum;
7     char lastName[ 15 ];
8     char firstName[ 10 ];
9     double balance;
10 };
11
12 int main()
13 {
14     FILE *cfPtr;
15     struct clientData client = { 0, "", "", 0.0 };
16
17     if ( ( cfPtr = fopen( "credit.dat", "r+" ) ) == NULL )
18         printf( "File could not be opened.\n" );
19     else {
20         printf( "Enter account number"
21             " ( 1 to 100, 0 to end input )\n? " );
22         scanf( "%d", &client.acctNum );
23
24         while ( client.acctNum != 0 ) {
25             printf( "Enter lastname, firstname, balance\n? " );
26             fscanf( stdin, "%s%s%lf", client.lastName,
27                 client.firstName, &client.balance );
28             fseek( cfPtr, ( client.acctNum - 1 ) *
29                 sizeof( struct clientData ), SEEK_SET );
30             fwrite( &client, sizeof( struct clientData ), 1,
31                 cfPtr );
32             printf( "Enter account number\n? " );

```

FILE RANDOM

1. Define struct

1.1 Initialize variables

2. Open file

2.1 Input data

2.2 Write to file

```
33         scanf( "%d", &client.acctNum );
34     }
35
36     fclose( cfPtr );
37 }
38
39 return 0;
40 }
```

```
Enter account number (1 to 100, 0 to end input)
? 37
Enter lastname, firstname, balance
? Barker Doug 0.00
Enter account number
? 29
Enter lastname, firstname, balance
? Brown Nancy -24.54
Enter account number
? 96
Enter lastname, firstname, balance
? Stone Sam 34.98
```

FILE RANDOM

3. Close file

Program Output