LAMPIRAN
PUSTAKA I/O C++

Fasilitas I/O tidak merupakan bagian dari bahasa C++, tetapi diiplementasikan sebagai suatu library dalam C++. Pada bab ini akan dibahas pustaka **iostream** yang sering digunakan dalam pemrograman C++.

Pada level paling bawah, suatu file diinterpretasikan sebagai suatu barisan atau stream dari bytes. Pada level ini konsep tipe data tidak ada. Sedangkan pada level user, suatu file terdiri dari suatu barisan data dari satu atau lebih tipe karakter karakter, nilai-nilai numerik, dan obyek-obyek kelas.

Pustaka iostream menyadiakan sejumlah operasi untuk menangani baca dan tulis tipe-tipe data baku. Pemrograman dapat memperluas operasi-operasi itu untuk menangani tipe tipe kelas yang dibuatnya. Operasi input dan output didukung oleh kelas **istream** (input stream) dan kelas **ostream** (output stream). Operasi output dilakukan oleh operator left-shift atau operator insertion (<<), sedangkan operasi input dilakukan oleh operator right-shift atau operator extraction (>>).

Empat objek stream tersedia bagi pengguna:
1. **cin**, suatu objek dari kelas istream yang dikaitkan dengan standard-input.
2. **cout**, suatu objek dari kelas ostream yang dikaitkan dengan standard-output.
3. **cerr**, suatu objek dari kelas ostream yang dikaitkan dengan standard-error dengan *unbuffered output*
4. **clog**, suatu objek dari kelas ostream yang dikaitkan dengan standard-error dengan *buffered output*

Program yang memakai pustaka iostream harus meng-*include* file **iostream.h**. Pengguna dapat mengaitkan suatu file tertentu ke program dengan mendefinisikan suatu objek dari salah satu kelas berikut ini:
1. ifstream
   - turunan dari istream.
   - mengaitkan suatu file ke program untuk input.
2. ofstream
   - turunan dari ostream
   - mengaitkan suatu file ke program untuk output.
3. fstream
   - turunan dari iostream
   - mengaitkan suatu file ke program untuk input dan output.

Pustaka iostream juga mendukung pengaturan array karakter. Ada dua kelas untuk itu, yaitu:
1. istrstream
   - turunan dari ostream

- menyimpan karakter ke suatu array.

2. ostrstream
   - turunan dari ostream
   - menyimpan karakter ke suatu array.

Metode output secara umum adalah menerapkan operator << ke dalam cout. Operator << sudah di-overload untuk menangani tipe-tipe data built-in dan pointer (address objek). Secara otomatis, pointer ditampilkan dalam notasi heksadesimal. Untuk tampilan dalam notasi desimal harus dilakukan dengan cast eksplisit ke tipe long (unsigned). Contoh:

```
#include  <iostream.h>
main()
{
   int i =0; int *pi=&i;
   cout << "i: "<< i<< "\t &i:\t" << &i <<endl;
   cout << "*pi: " << *pi
        << "\t pi: \t" << pi << endl
        << "\t\t &pi :\t" << &pi << endl;
   cout << "\t\t pi: \t" << (unsigned long) pi << endl
        << "\t\t &pi: \t" << (unsigned long) &pi << endl;

   return 0;
}
```

Keluaran dari program tersebut adalah:

```
i : 10          &i : 0xfff4
*pi : 10        pi  : 0xfff4
                &pi : 0xfff2
                pi : 2372599796
                &pi : 2372599794
```

Operator << menginterpretasikan tipe c**har*** sebagai suatu string, bukan suatu nilai address. Perhatikan contoh berikut:

```
#include <iostream.h>
char  str[ ] = "hello ";
void main ()
{
     char *pstr =str;
     cout << "pstr:\t\t" << pstr <, endl
          << "(void ) pstr:\t; << (void)pstr <<endl;
}
```

Keluaran dari program tersebut adalah:

```
pstr :          hello
(void *)pstr: 0x00aa
```

Metode input dilakukan dengan menerapkan operator >> ke dalam cin. Seperti halnya dengan operator <<, operator >> dapat dikonkatenasikan. Contoh:

```
#include <iostream.h>
void main()
{
    int i1,i2;
    cout << "Ketikkan dua integer: ";
    cin >> i1 >>i2;
    cout  <<  Terbaca: < " << i1 << "," << i2 " >  " << endl;
}
```

Keluaran dari program tersebut adalah:

```
Ketikkan  dua integer : 5  12
Terbaca : <5,12>
```

Metode yang lebih umum untuk membaca dari input-stream adalah membuat operasi input sebagai kondisi dari suatu while loop seperti pada contoh berikut:

```
char ch;
while(cin >> ch) ….
```

yang membaca karakter satu persatu dari standar input_output. Apabila ketemu end-of-file, kondisi (cin >> ch) bernilai false dan loop akan terhenti.

Karakter-karakter:

a b c
d e

diperlakukan oleh operator >> sebagai barisan 5 karakter 'a','b','c','d','e'. White-space(spasi, newline,tab) berlaku sebagai pemisah nilai, tidak di baca sebagai karakter. Gunakan member-functions: get(),getline() dan read() kalau yang ingin dibaca termasuk white-space.

Operator input ">>" dapat juga dipakai untuk membaca suatu barisan sring dari input-stream. Menurut operator >>, string adalah barisan karakter yang dibatasi oleh white-space. Contoh:

```
char inBuf [panjangKata];
while(cin >> inBuf) …
```

akan membaca "S2 haruslah S1++" sebagai tiga string:

"S2
haruslah
S1++"

Manipulator setw() dapat dipakai untuk mencegah overflow pada array karakter input. Setw(ukuran) membagi suatu string yang sama atau lebih panjang dari ukuran menjadi dua atau lebih string dengan panjang maksimum ukuran –1.

Untuk memakai setw(), program perlu “include” header_file iomanip.h. Contoh:

```
#include<iostream.h>
#include<iomanip.h>

const ukuran = 24;
char buf[ukuran];
void(main)
{
     char *pbuf = buf;
     while(cin >> setw(sizeof(pbuf)) >> pbuf)
          cout << pbuf << endl;
}
```

Program ini akan memecah setiap string yang diinputkan yang panjangnya lebih dari sizeof(char*) menjadi dua atau lebih string.

I/O member-functions terdiri dari:

1. put(char ch), mengoutput karakter ch ke output-stream dan mengembalikan objek kelas ostream yang menginvokasinya.
2. get(char & ch), mengambil satu karakter dari input-stream dam menyimpannya di ch, serta mengembalikan obyek kelas iostream yang menginvokasinya. Contoh:

```
#include <iostream.h>
main()
{
     char ch;
     while(cin.get(ch))  cout.put(ch);
     return 0;
}
```

3. get(), mengambil dan mengembalikan satu nilai dari input-stream, termasuk nilai end-of file EOF yang didefinisikan di iostream.h. Contoh:

```
#include<iostream.h>
main()
{
     int ch;
     while( (ch = cin.get()) != EOF)
          cout.put(ch);
     return 0;
}
```

EOF biasanya diberi nilai –1. Oleh karena itu ch diberi tipe int.

4. write(const char *str, int length): menghasilkan sebuah string ke output stream dan mengembalikan object stream yang menginvokasinya, dan length menentukan jumlah karakter yang ditampilkan, mulai dari karakter yang ditunjuk dalah str.
5. getline(char *Buf, int Limit, char Delim = ‘\n’): mengambil satu blok karakter dan menyimpannya dalam array Buf’, jumlah karakter yang diambil ≤ (Limit –1), karakter null dicantolkan ke akhir Buf, jumlah karakter yang diambil <(Limit –1)

apabila ditemukan EOF atau karakter yang dikandung Delim, dan karakter yang dikandung Delim tidak ditaruh dalam Buf.

6. gcount(), memberikan jumlah karakter ynag diambil oleh invokasi getline() yang terakhir.
7. read(char *addr, int size): mengambil sebanyak size bytes berturutan dari input-stream dan menaruh mereka mulai pada alamat addr, dan gcount() juga mengembalikan jumlah bytes yang diambil oleh invokasi read() yang terakhir.
8. putback(char c), menaruh kembali karakter c ke input stream.
9. peek(), memberitahu karekter berikutnya (atom EOF) tetapi tidak mengambilnya.
10. ignore(int Limit = 1, int Delim = EOF): membuang karakter sebanyak <= Limit, dan stop jika ketemu karakter yang dikandung Delim. Contoh:

```
#include<iostream.h>
:
:
char ch,next,lookahead;
while(cin.get(ch))  {
     switch(ch)  {
     case '/':                              //kalau comment,buang
          next = cin.peek();
          if(next == '/')
               cin.ignore(panjangBaris,'\n');
          break;
     case '>':                              //cari >>=
          next = cin.peek();
          if(next == '>')  {
               lookahead = cin.get();
               next = cin.peek();
               if(next != '=')
                    cin.putback(lookahead);
          }
     }
}
```

Penggunaan suatu file dalam program untuk input dan/atau output memerlukan file header **fstream.h** disamping iostream.h. Contoh:

```
#include <stlib.h>
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
char *infile = "masuk";
char *outfile = "keluar";

void(main)
{
     ifstream iFile(infile, ios::in);
     ofstream oFile(outfile, ios::out);
```

```
    if(!iFile)  {
        cerr << "Tidak dapat buka <" << infile << "> untuk input.\n";
        exit(-1);
    }

    if (!oFile)  {
        cerr << "Tidak dapat buka <" << outfile << " > untuk output.\n";
        exit(-1);
    }

    int cnt = ∅;
    const ukuran = 125;
    char buf[ukuran];
    while(iFile >> setw(ukuran) >> buf)  {
        oFile << buf << endl;
        ++cnt;
    }

    cout << :["  << cnt << "]" << endl;
}
```

Baik obyek kelas ifstream maupun ofstream dapat didefinisikan tanpa menentukan suatu file tertentu. Suatu file dapat secara eksplisit dihubungkan nanti dengan objek itu melalui member-function open(). Perhatikan contoh berikut:

```
ifstream berkas;
    :
                        bertipe char*
  berkas.open(namaBerkas,ios::in)
  if(!berkas)  {
    :
    :
```

Suatu file dapat diputus hubungannya dengan program dengan menggunakan member-function close(). Contoh:

```
berkas.close();
```

Suatu objek dari kelas fstream dapat dibuka untuk: 1) input, atau 2) output, atau 3) input dan output. Contoh:

```
fstream  io("Berkas.IO",ios::in|ios::app);-→ append mode
```

Program berikut membaca dari suatu file tertentu, dimana setiap kali membaca karakter newline, menulis ukuran file (dalam byte) sejauh yang telah terbaca (termasuk newline) pada akhir file itu. Setelah membaca EOF, program akan menulis ukuran file total pada akhir file. Misalnya program akan mengubah file test:

```
Ilmu
Komputer
```

menjadi:

```
Ilmu
Komputer
5    14     19
```

```
#include <iostream.h>
#include <fstream.h>

void main ()
{
    fstream inOut (“coba.out”, ios::in\ios:app);
    int hitungan =∅;
    char ch;

    inOut.seekg (∅,ios:: beg);
    while (inOut.get(ch))
    {
        cout.put (ch); hitungan++;
        if (ch == ‘\n’)
        {
            streampos tanda = inOut.tellg();
            inOut << hitungan << ‘ ‘;
            inOut.seekg(tanda);
        }
    }

    inOut.clear (); // reset state flags: get out of end-of-file state
    inOut << hitungan << endl;
    count << “[“ << hitungan << “]” << endl;
}
```

Untuk pindah ke posisi tertentu dalam suatu file untuk baca, dapat kita gunakan member-function yang deklarasinya adalah:

```
typedef long streampos;
istream & seekg ( streampos p, seek_dir d = ios::beg);
```

di mana seek_dir adalah suatu enumeration yang mencakup ios:: beg (awal file), ios:: cur (posisi sekarang dalam file), dan ios:: end (akhir file). Perpindahan pointer dapat dilakukan juga dengan:

seekg (p,d) → pindah ke p bytes offset dari posisi d.

Misalnya:

seekg (-10, ios::end) → mundur 10 bytes dari akhir file.

Posisi sekarang dalam file dapat diperoleh dengan member-function: tellg().

Status suatu objek iostream dapat diketahui dengan fungsi-fungsi anggota:

1. eof(): true jika end-of-file telah ketemu, misalnya:

```
if(inOut.eof())  inOut.clear();
```

2 bad(): true jika operasi tidak wajar coba dilakukan, misalnya: “seeking” melewati end-of-file.

3 fail(): true jika suatu operasi tak sukses atau jika bad() memberi true, misalnya:

```
ifstream iFile(namaBerkas, ios:in);
if(iFile.fail())  error(…);
```

4 good(): true jika 1), 2), dan 3) false.

Program berikut menggunakan suatu objek **istrstream** anonim untuk mengubah suatu string ke integer:

```
#include<iostream.h>
#include<strstream.h>
#include<string.h>
int strkeInt(char *s)
{
     int val;
     istrstream(s,strlen(s)+1) >> val;
     return val;
}

char *tbl[5] = {“10”,”-5”,”400”,”15”,”1024”};

void main()
{
     int jumlah = ∅;
     char *p = tbl[∅];
     for(int i=∅;i<5 ;p = tbl[++i])
          jumlah += strkeInt(p);
     cout << “jumlah: “ << jumlah << endl;
}
```