MODUL PRAKTIKUM
ALGORITMA & PEMROGRAMAN II

Oleh : Drs. Muh. Alhan, ST., M.Eng.

PROGRAM STUDI MANAJEMEN INFORMATIKA
POLITEKNIK PRATAMA MULIA
SURAKARTA 2018

Daftar Isi

Daftar Isi....2
Larik (Array) 4
  Larik Dimensi Satu 4
  Larik Dimensi Dua....6
  Larik Multi Dimensi....7
Matriks 10
  Dasar 10
  Macam-Macam Matriks 11
  Operasi Matematika Matriks 12
Pointer 17
  Pengertian....17
    Mendefinisikan variabel pointer 17
  Pointer void 18
  Operasi Pointer 19
    Operasi Penugasan 19
    Operasi Aritmatika....19
    Operasi Logika....20
  Pointer dan String 21
  Pointer dan Larik 22
  Pointer Menunjuk Pointer 23
Operasi File 26
  Pengantar File....26
  Pengertian....26
  Operasi Dasar....26
    Membuka File....26
    Menulis ke file 27
    Menutup File....27
    Membaca File 28
  Menambah Data....29
  Memformat String....30
  Operasi Berbasis Obyek....31
    Membaca 33
Struktur....36
  Konsep....36
    Mendefinisikan variabel struktur 36
    Struktur di dalam struktur....36
    Mengakses Anggota Struktur 37
    Penugasan struktur 38
  Union....39
    Inisialisasi Union....40
Pencarian 42
  Konsep....42
  Metode Pencarian Beruntun 42
  Metode Pencarian bagi Dua (Binary Search) 45
Pengurutan....50

Metode Pengurutan Gelembung (Bubble Sort) 51
Metode Pengurutan Pilih (selection Sort) 53
Metode Pengurutan Sisip (Insertion Sort).....54
Daftar Pustaka 56

Larik (Array)

Larik merupakan kumpulan dari nilai-nilai data yang bertipe sama dalam urutan tertentu yang menggunakan nama yang sama. Letak atau posisi dari elemen larik ditunjukkan oleh suatu indeks. Dilihat dari dimensinya larik dapat dibagi menjadi Larik Dimensi Satu, Larik Dimensi Dua, dan Larik Multi-Dimensi.

Larik Dimensi Satu

Larik dimensi satu merupakan tipe data yang sering digunakan pada pendeklarasian variable yang sama tapi memiliki indeks yang berbeda, serta pengisian elemen larik dilakukan melalui indeks. Indeks larik secara default dimulai dari 0.

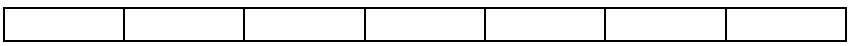
Bentuk umum penulisan :

Type_data variabel1[jumlah_elemen];

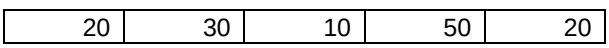
Contoh :

```
int data1[7];
int data2[5] = {20,30,10,50,20];
```

artinya :



Data 1 = Elemen kosong



Data 2 = Elemen tidak kosong

Contoh program :

```
//Program untuk memasukkan data dan menampilkan data
//Nama File Ini.cpp

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

```
void main()
{
      float x[5];    //Deklarasi larik dengan lima  elemen
      int d;
      clrscr();

      for(int i = 1; i <= 5; i++)   //mengisi larik
      {
            cout<<"Isi data ";
            cin>>x[i];
       }
      for( i = 1; i <= 5; i++)
      {
            cout<<”Tampilkan hasil “<<x[i]<<endl; //tampilan
setelah diisi
      }
}

//Program menghitung suhu rata-rata
//Nama file suhu.cpp

#include <iostream.h>
#include <conio.h>

const int JUM_DATA = 5;

void main()
{
      float suhu[JUM_DATA];
      float total;

      clrscr();

      cout<<"Masukkan data suhu "<<endl;
      for (int i = 0; i < JUM_DATA; i++)
      {
            cout<< i + 1 << " : ";
            cin>>suhu[i];
      }

      total = 0;

      for (i  = 0 ; i < JUM_DATA; i++)
            total += suhu[i];
      cout<<"Suhu rata - rata = " <<total/JUM_DATA<<endl;
}
```

Larik Dimensi Dua

Larik dimensi dua merupakan tipe data yang sering digunakan pada pendeklarasian variabel yang sama tapi memiliki dua indeks yang berbeda, serta pengisian elemen larik dilakukan melalui indeks. Indeks larik secara default dimulai dari 0,0. Jumlah elemennya adalah indeks1 x indeks 2.

Bentuk umum penulisan :

Type_data variabel1[jumlah_elemen1][jumlah_elemen2];

Contoh :

```
int y[2][2];
```

artinya di dalam memori computer terdapat alokasi sebagai berikut :

Data 1 = Elemen kosong terdapat 2 baris dan 2 kolom

Contoh program :

```
//Program pemakaian larik dimensi dua
//Nama File lulus.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
int data_lulus[3][4]; //deklarasi larikk dimensi dengan tiga
                        elemen baris dan
     int tahun, jurusan;    //empat elemen kolom

     clrscr();

     data_lulus[0][0] = 35; //mengisi larik
     data_lulus[0][1] = 45;
     data_lulus[0][2] = 90;
     data_lulus[0][3] = 120;
     data_lulus[1][0] = 100;
     data_lulus[1][1] = 110;
     data_lulus[1][2] = 70;
     data_lulus[1][3] = 101;
     data_lulus[2][0] = 10;
     data_lulus[2][1] = 15;
     data_lulus[2][2] = 20;
```

```
      data_lulus[2][3] = 17;

      while (1)
      {
            cout<<"Jurusan (0 = T I, 1 = MI, 2 = TK) ";
            cin>>jurusan;

            if((jurusan == 0) || (jurusan == 1 ) || (jurusan == 2))
                  break;
      }

      while (1)
      {
            cout<< "tahun (1992 - 1995); ";
            cin>>tahun;

            if((tahun >= 1992) && (tahun <= 1995))
            {
                  tahun -= 1992;
                  break;
            }
      }

      cout<<"jumlah yang lulus = "
             <<data_lulus[jurusan][tahun]<<endl;
}
```

Larik Multi Dimensi

Larik Multi Dimesi adalah larik yang banyak memiliki dimensi tidak terbatas pada satu atau dua dimensi. Larik tersesebut memiliki dimensi sesuai dengan kebutuhan, walaupun sebenarnya jarang melebihi dimensi tiga.

Bentuk umum penulisan :

Type_data variabel1[jumlah_elemen1][jumlah_elemen2]…[jumlah_elemen*n*]

Contoh :

```
int  x[2][2][2];
int  y[4][5][8][2];
```

Contoh program :

```
//Program larik dimensi tiga
```

```
//Nama file hurufab.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      //pendefinisian larik dimensi tiga
      //dan pemberian nilai awal
      int huruf [2][8][8] =
      {
            {{0,1,1,1,1,1,0,0},
             {0,1,0,0,0,1,0,0},
             {0,1,0,0,0,1,1,0},
             {1,1,1,1,1,1,1,0},
             {1,1,0,0,0,0,1,0},
             {1,1,0,0,0,0,1,0},
             {1,1,0,0,0,0,1,0},
             {0,0,0,0,0,0,0,0},
            },
             {{1,1,1,1,1,1,0,0},
              {1,0,0,0,0,1,0,0},
              {1,0,0,0,0,1,0,0},
              {1,1,1,1,1,1,1,0},
              {1,1,0,0,0,0,1,0},
              {1,1,0,0,0,0,1,0},
              {1,1,1,1,1,1,1,0},
              {0,0,0,0,0,0,0,0},
             }
            };
            int i,j,k;
            clrscr();
            for (i=0;i<2;i++)
            {
                  for(j=0;j<8;j++)
                  {
                        for (k = 0; k <8; k++)
                        {
                              if (huruf[i][j][k] == 1)
                                    cout<<'\xDB';
                              else
                                    cout<<'\x20';
                        }
                        cout <<endl;
                  }
            }
}
```

Pertanyaan

1. Jika suatu larik memiliki 100 elemen, sebutkan range subscript yang diperoleh ?
2. Apa perbedaan dari ekspresi a4 dan a[4] ?
3. Suatu larik dengan nama day dideklarsikan sebagai berikut :

   int hari[] = {mon, tue, wed, thu, fri}

   Berapa banyak elemen dari larik day ?

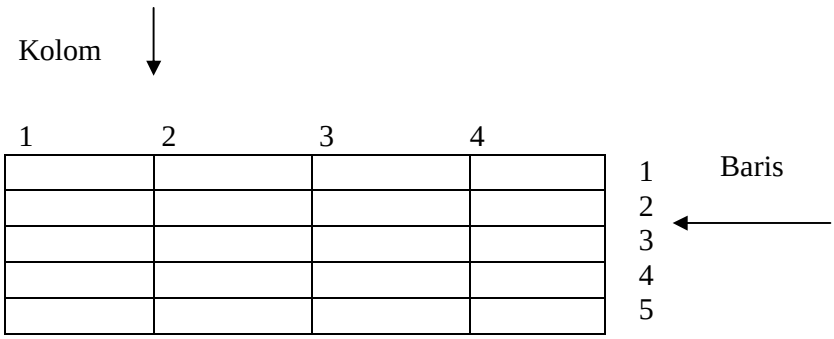
   Bagaiman kalau deklarasi diganti menjadi

   int day[7] = {mon, tue,wed,thu,fri};
4. Buat program dengan larik untuk menghitung rata-rata deret bilangan ganjil 1. d.. 100 yang habis dibagi 5 .

Matriks

Dasar

Matriks adalah sekumpulan informasi yang setiap individu elemennya diacu dengan menggunakan dua buah indeks (yang biasanya dikonotasikan dengan baris dan kolom). Gambar 1 memperlihatkan matriks yang terdiri dari lima buah baris dan empat buah kolom. Angka 1, 2, 3, 4 (dan 5) menyatakan indeks baris dan indeks kolom. Karena adanya dua buah indeks tersebut, matriks disebut juga larik dimensi dua.



Gambar 1 *Matriks yang terdiri dari lima baris dan empat kolom*

contoh matriks $A_{(3x3)}$:

A = 3 4 5
3 0 8
5 1 2

Karena matriks sebenarnya adalah larik, maka konsep umum dari larik juga berlaku untuk matriks yaitu :

1. Kumpulan elemen yang bertipe sama.
2. Setiap elemen dapat diakses secara acak (random) jika indeksnya (baris dan kolom) diketahui, yang dalam hal ini ini indeks menyatakan posisi relatif di dalam kumpulannya.
3. Merupakan struktur data yang statik, artinya jumlah elemennya sudah ditentukan terlebih dahulu di dalam deklarasi dan tidak bisa diubah selama pelaksanaan program.

Struktur matriks praktis untuk dipakai (pengaksesnya cepat) tetapi memakan banyak tempat di memori. Misalnya matriks integer berukuran 100 x 100, membutuhkan 10000 x tempat penyimpanan integer (2 byte).

Jumlah baris dan jumlah kolom disebut juga *matra* atau ukuran matriks. Matriks pada gambar 1 berukuran 5 x 4. Karena matriks adalah struktur statik, maka ukuran matriks harus sudah diketahui sebelum pelaksanaan program. Seperti pada larik biasa, kita menuliskan matra matriks sebagai penomoran indeks baris dan indeks kolom, mulai dari indeks terendah sampai indeks tertinggi. Sebagai contoh, misalkan matriks pada gambar 1 bernama Mat, maka Mat[1..5, 1..4] menyatakan matriks berukuran 5 x 4, dengan indeks baris 1 sampai 5 dan indeks kolom dari 1 sampai 4. Indeks matriks tidak harus dimulai dari 1, tetapi juga boleh dari 0 atau dari bilangan negatif, seperti Mat[0..4, 1..3], Mat[-2..2, 0..3]. Bahkan, indeks juga dapat bertipe karakter, seperti P[‘a’..’f’,’s’..’w’] atau tipe lain yang mempunyai keterurutan.

Macam-Macam Matriks

Matriks Identitas adalah matriks yang diagonalnya bernilai 1.

1	0	0
0	1	0
0	0	1

Matriks baris adalah matriks yang memiliki hanya satu baris.
Contoh data berikut merupakan hasil pengukuran suhu kota Jakarta. (satuan dalam derajat Celcius) selama lima hari.

37	36	35,6	36,1	35,9

Matriks kolom adalah matriks yang hanya memiliki satu kolom
Contoh data berikut merupakan data pengukuran tangki minyak pertamina (satuan dalam m) tiap 30 menit.

57
58
59
60

61
62

Matriks bujur sangkar adalah matriks yang memiliki jumlah baris dan kolom yang sama

Contoh data berikut menyatakan hubungan diplomatik antara negara-negara.

	Indonesia	Portugal	Jepang	Nepal
Indonesia	-	FALSE	TRUE	FALSE
Portugal	FALSE	-	TRUE	TRUE
Jepang	TRUE	TRUE	-	TRUE
Nepal	FALSE	TRUE	TRUE	-

ket : true artinya ada hubungan diplomatik dan false artinya tidak ada hubungan diplomatik

Operasi Matematika Matriks

Penjumlahan

Contoh

Matriks A = $\begin{matrix} 3 & 5 & 3 \\ 4 & 1 & 0 \\ 3 & 9 & 6 \end{matrix}$ Matriks B = $\begin{matrix} 1 & 2 & 1 \\ 0 & 4 & 2 \\ 1 & 1 & 0 \end{matrix}$

maka C = A + B

$= \begin{matrix} 3+1 & 5+2 & 3+1 \\ 4+0 & 1+4 & 0+2 \\ 3+1 & 9+1 & 6+0 \end{matrix}$

$= \begin{matrix} 4 & 7 & 4 \\ 4 & 5 & 2 \\ 5 & 10 & 6 \end{matrix}$

Pengurangan

Contoh

Matriks A = $\begin{matrix} 3 & 5 & 3 \\ 4 & 1 & 0 \\ 3 & 9 & 6 \end{matrix}$ Matriks B = $\begin{matrix} 1 & 2 & 1 \\ 0 & 4 & 2 \\ 1 & 1 & 0 \end{matrix}$

maka C = A - B

$= \begin{matrix} 3-1 & 5-2 & 3-1 \\ 4-0 & 1-4 & 0-2 \\ 3-1 & 9-1 & 6-0 \end{matrix}$

```
      = 2    3    2
        4   -3   -2
        2    8    6
```

Perkalian

Contoh

```
Matriks A = 3   5   3      Matriks B =  1   2   1
            4   1   0                   0   4   2
            3   9   6                   1   1   0
```

maka C = A x B

```
      = 3x1+5x0+3x1   3x2+5x4+3x1   3x1+5x2+3x0
        4x1+1x0+0x1   4x2+4x4+0x1   4x1+1x2+0x0
        3x1+9x0+6x1   3x2+9x4+6x1   3x1+9x2+6x0

      = 6   29   3
        4   28   6
        9   48   21
```

Contoh Program

```
//Program mengisi nilai matriks
//Nama File Mat1.cpp

#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
#include <stdlib.h>

void main()
{
      int jb, jk, i, j;
      float arr1[10][10];

      clrscr();

      cout<<"Jumlah baris : ";cin>>jb;
      cout<<"Jumlah kolom : ";cin>>jk;
      cout<<"Data matriks ";
      cout<<"\n";
      arr1[0][0]=0;

      for (i = 1; i <=jb; i++)
      {
            for (j = 1; j <=jk; j++)
            {
                  cout<<"Data ke "<<i<<"-"<<j ;cin>> arr1[i][j];
```

```
            }
       }

       cout<<setprecision(2);
       cout<<setiosflags(ios::fixed);
       for (i = 1; i <=jb; i++)
       {
            for (j = 1; j <=jk; j++)
             cout<<setw(8)<<arr1[i][j];
             //cout<<setw(3);
                  cout<<"\n";
        }

       cout<<resetiosflags(ios::fixed);
}

//Program menjumlahkan dua buah matriks
//Nama file jumlah_mat.cpp

#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
#include <stdlib.h>

void main()
{
     int jb, jk, i, j;
     float arr1[10][10];
     float arr2[10][10];
     float arr3[10][10];

     clrscr();

     cout<<"Jumlah baris : ";cin>>jb;
     cout<<"Jumlah kolom : ";cin>>jk;
     cout<<" Data matriks A ";
     cout<<"\n";

     arr1[0][0]=0;
     arr2[0][0]=0;
     arr3[0][0]=0;

     for (i = 1; i <=jb; i++)
     {
          for (j = 1; j <=jk; j++)
          {
               cout<<"Data ke "<<i<<"-"<<j ;cin>> arr1[i][j];
          }
      }
      for (i = 1; i <=jb; i++)
     {
          for (j = 1; j <=jk; j++)
               cout<<setw(8 )<<arr1[i][j];
```

```
                cout<<"\n";
        }

        cout<<"\n";
        cout<<"Data matriks B\n ";
        for (i = 1; i <=jb; i++)
        {
             for (j = 1; j <=jk; j++)
             {
                        cout<<"Data ke "<<i<<"-"<<j ;
                        cin>> arr2[i][j];
             }
         }
         for (i = 1; i <=jb; i++)
        {
             for (j = 1; j <=jk; j++)
                  cout<<setw(8 )<<arr2[i][j];
                  cout<<"\n";
        }
    for (i = 1; i <=jb; i++)
    {
             for (j = 1; j <=jk; j++)
                  arr3[i,j] = arr1[i,j]+arr2[i,j];
                  cout<<"\n";
    }

         for (i = 1; i <=jb; i++)
        {
             for (j = 1; j <=jk; j++)
                  cout<<setw(8)<<"Data ke "<<i<<"-"<<j<<arr3[i][j];
        }
}
```

Pertanyaan

1 Buatlah program untuk menghitung pengurangan dua buah matriks.

2. Buatlah program untuk menghitung perkalian dua buah matriks.

Pointer

Pengertian

Pointer (variabel penunjuk) adalah suatu variabel yang berisi alamat memori dari suatu variabel lain. Alamat ini merupakan lokasi dari obyek lain (biasanya variabel lain) di dalam memori. Contoh, jika sebuah variabel berisi dari variabel lain, variabel pertama dikatakan menunjuk ke variabel kedua.

Mendefinisikan variabel pointer

Suatu variabel pointer didefinisikan dengan bentuk sebagai berikut :

Tipe_data *nama_variabel;

- **tipe_data** dapat berupa sembarang tipe seperti halnya pada pendefinisian variabel bukan pointer.
- **nama_variabel** adalah nama variabel pointer.

Beberapa contoh pendifinisian variabel pointer :

```
int *pint       // Pointer ke int
char *pch       //Pointer ke char
float *pfl      //Pointer ke float
```

variabel pointer pint dapat diatur agar menunjuk ke vint dengan cara sebagai berikut :

```
pint = &vint
```

Pernyataan di atas berarti : “pint diisi dengan alamat dari vint”. Operator & menyatakan “alamat dari”.

Sekarang, marilah kita lihat program yang mendefiniskan variabel pointer, mengisinya dengan alamat suatu variabel dan menampilkan isi dari pointer.

Contoh program :

```
//Program pointer pertama
//Nama file ptr1.cpp

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

```
void main()
{
      int vint = 55;
      int *pint;

      clrscr();

      pint = &vint;

      cout<<"Alamat vint = "<<&vint<<endl;
      cout<<"pint = "<<pint<<endl;
}
```

Pointer void

Pada bagian sebelumnya diberikan contoh variabel pointer yang menunjuk ke tipe data tertentu. Namun sebenarnya juga dimungkinkan untuk membuat pointer yang tak bertipe. Caranya yakni dengan meletakkan kata kunci *void* pada bagian penentu tipe pointer, Contoh :

v*oid *ptr;*

Merupakan pernyataan untuk mendefinisikan *ptr* sebagai variabel pointer void.

Suatu pointer void adalah pointer yang dapat menunjuk ke sembarang tipe data. Misalnya, Anda dapat mengatur agar pointer ini menunjuk ke tipe data int, tetapi di saat lain diperlukan untuk menunjuk data bertipe float

Catatan : selain pointer void juga terdapat istilah pointer null. Pointer null mempunyai makna yang berbeda dengan pointer void. Pointer null adalah pointer yang beris nol (NULL).

Contoh program

```
//Program pointer void
//Nama file void.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      clrscr();

      void *ptr; //pointer tak bertipe

      int vint = 50;
```

```
      float vfl = 51.5;

      ptr = &vint; //menunjuk ke int, diperkenankan

      cout<<"Nilai yang ditunjuk oleh ptr : "
          <<*(int*)ptr<<endl;

      ptr = &vfl;  //menunjuk ke float, diperkenankan

      cout<<"Nilai yang ditunjuk oleh ptr : "
           <<*(float *)ptr<<endl;
}
```

Operasi Pointer

Operasi Penugasan

Suatu variabel pointer seperti hanya variabel yang lain, juga bisa mengalami operasi penugasan. Nilai dari suatu variabel pointer dapat disalin ke variabel pointer yang lain.

Contoh program :

```
//Program penugasan pointer
//Nama file tugas.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      float *x1,*x2,y;

      clrscr();

      y = 13.45;
      x1 = &y;
      x2 = x1;

      cout<<"Nilai variabel y = "<<y<<" ada di alamat "<<x1<<endl;
      cout<<"Nilai variabel y = "<<y<<" ada di alamat "<<x2;
      getch();
}
```

Operasi Aritmatika

Suatu variabel pointer hanya dapat dilakukan operasi aritmatika dengan nilai integer saja. Operasi yang biasa dilakukan adalah operasi penambahan dan pengurangan. Operasi penambahan dengan suatu nilai menunjukkan lokasi data berikutnya (indeks selanjutnya) dalam memori. Begitu juga operasi penguranganya.

Contoh program :

```
//Program pointer aritmatika
//Nama file arit.cpp

#include "iostream.h"
#include "conio.h"

void main()
{
      int nilai[3], *penunjuk;
      clrscr();
      nilai[0] = 125;
      nilai[1] = 345;
      nilai[2] = 750;
      penunjuk = &nilai[0];
      cout<<"Nilai "<<*penunjuk<<" ada di alamat
                                  "<<penunjuk<<endl;
      cout<<"Nilai "<<*(penunjuk+1)<<" ada di alamat
                                       "<<(penunjuk+1)<<endl;
      cout<<"Nilai "<<*(penunjuk+2)<<" ada di alamat
                                       "<<(penunjuk+2)<<endl;
      getch();
}
```

Operasi Logika

Suatu pointer juga dapat dikenai operasi logika.

```
//Program aritmatika pointer
//Nama file arit.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      int a = 100, b = 200, *pa,*pb;

      clrscr();

      pa = &a;
      pb = &b;
```

```
      if(pa<pb)
            cout<<"pa menunjuk ke memori lebih rendah dari
                  pb"<<endl;
      if(pa==pb)
            cout<<"pa menunjuk ke memori sama dengan pb"<<endl;
      if(pa>pb)
            cout<<"pa menunjuk ke memori lebih tinggi dari
                  pb"<<endl;
      getch();
}
```

Pointer dan String

Contoh hubungan pointer dan string ditunjukkan oleh program berikut :

```
#include "stdio.h"
#include "conio.h"

char *nama1 = "SPIDERMAN";
char *nama2 = "GATOTKACA";

void main()
{
      //char namax;

      clrscr();

      puts("SEMULA :");
      printf("Saya suka >>%s\n ",nama1);
      printf("Tapi saya juga suka >>%s\n ",nama2);

      printf("SEKARANG : ");
      printf("Saya suka >>%s\n ",nama1);
      printf("Dan saya juga masih suka>>%s\n ",nama2);

      getch();
}
```

Pada contoh di atas :

```
Char *nama1 = “SPIDERMAN”;
Char *nama2 = “GATOTKACA”;
```

Akan menyebabkan C++ :

- mengalokasikan nama1 dan nama2 sebagai variabel pointer yang menunjuk ke data bertipe char dan menempatkan konstanta string “SPIDERMAN” dan “GATOTKACA” ke suatu lokasi di memori komputer.

- Kemudian nama1 dan nama2 akan menunjuk ke lokasi string “SPIDERMAN” dan “GATOTKACA”

Pointer dan Larik

Pointer dan larik mempunyai hubungan yang dekat. Secara internal larik juga menyatakan alamat. Misalnya, didefinisikan :

int tgl_lahir[] = {24, 6, 1965};

dan :

*int *ptgl;*

agar ptgl menunjuk ke larik, diperlukan pernyataan berupa :

ptgl = tgl_lahir;

Perhatikan dengan seksama pernyataan di atas. Tidak ada tanda & di depan menggunakan format :

ptr = &variabel

Bila variabel bukan larik. Ini disebabkan nama larik sebenarnya sudah menyatakan alamat. Oleh karena itu tanda & tidak diperlukan.

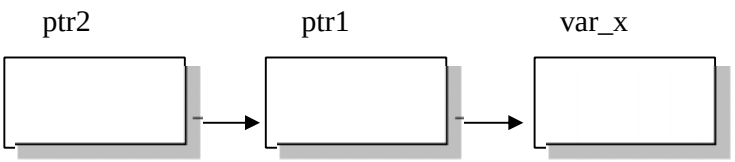
Contoh program :

```
//Program pointer dan larik
//Nama file ptrlar.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      clrscr();
      int tgl_lahir[] = {24, 6, 1965};
      int *ptgl;

      ptgl = tgl_lahir;

      cout<<"Nilai yang ditunjuk oleh ptgl : "
            <<*ptgl<<endl;

      cout<<"Nilai dari tgl_lahir[0]   : "
            <<tgl_lahir[0]<<endl;
}
```

```
//Program mengakses elemen larik melalui pointer
//ptrlar2.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      clrscr();

      int tgl_lahir[] = {24, 6, 1965};
      int *ptgl;

      ptgl = tgl_lahir; //ptgl menunjuk ke
                                          //elemen pertama dari larik
      for(int i = 0; i < 3; i++)
      {
            cout<<"ptgl = "<<ptgl<<endl;
            cout<<"*ptgl = "<<*ptgl<<endl;

            ptgl++;    //menunjuk ke elemen berikutnya
      }
}
```

Pointer Menunjuk Pointer

Suatu pointer bisa saja menunjuk ke pointer lain. Gambaran mengenai hal ini ditunjukkan pada gambar 1 di bawah



gambar 1 pointer menunjuk ke pointer

Untuk membentuk rantai pointer seperti gambar 1 diperlukan pendifinisian sebagai berikut :

int var_x;

*int *ptr1; // satu tanda * di depan ptr1*

*int **ptr2; // dua tanda * di depan ptr2*

Pada pendifinisian di atas :

- *var_x* adalah variabel bertipe int.
- *ptr1* adalah variabel pointer yang menunjuk ke data bertipe int.
- *ptr2* adalah variabel pointer yang menunjuk ke pointer int.

agar *ptr1* menunjuk ke variabel *var_x*, perintah yang diperlukan berupa :

ptr1 = &var_x; sedangkan supaya ptr2 menunjuk ke *ptr1*, diperlukan perintah berupa :

ptr2 = &ptr1;

Contoh program yang memberikan gambaran pengaksesan nilai pada *var_x* melalui pointer *ptr1* dan *ptr2* dapat dilihat di bawah ini.

```
//Program pointer yang menunjuk ke pointer lain
//Nama file ptrptr.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
     clrscr();

     int var_x = 273;
     int *ptr1; //Pointer ke variabel bukan pointer
     int **ptr2;     //Pointer ke pointer

     cout<<"var_x = "<<var_x<<endl;

     //Penugasan Alamat

     ptr1 = &var_x;  //ptr1 menunjuk ke var_x
     ptr2 = &ptr1;   //ptr2 menunjuk ke ptr1

     //Mengakses nilai var_x melalui ptr1

     cout<<"*ptr1 = "<<*ptr1<<endl;

     //Mengakses nilai var_x melalui ptr2

     cout<<"**ptr1 = "<<**ptr2<<endl;
}
```

Pertanyaan

1. Apa yang dimaksud dengan pointer ?
2. Apa persamaan larik dengan pointer ?
3. Jelaskan jenis data dari definisi berikut :

   a. `float *measures[250];`

   b. `int *volume[50];`
4. carilah kesalahannya.

   Jika diberikan definisi variabel

   ```
   int i;
   long int j;
   int *ip1;
   int *ip2;
   ```

   Mengapa berikut ini tak akan bekerja ?

   ```
   ip1 = &i;
   jp2  = &j;
   ```
5. Tuliskan program yang menyimpan nama-nama bahan pokok makanan dalam larik pointer karakter. Tampilkan nama-nama bahan pokok makanan tersebut

Operasi File

Pengantar File

Berhubungan dengan komputer memang tidak pernah lepas dari file. Program yang kita tulis biasa kita letakkan ke dalam file. Saat lain bila kita perlukan kita bisa memanggilnya, menyunting dan kemudian menyimpan kembali. Bagaimana halnya kalau kita bermaksud menulis, membaca atau melakukan hal-hal lain yang berhubungan dengan file melalui C++. Pada bagian ini akan menngupasnya dari hal yang mendasar.

Pengertian

File adalah sebuah organisasi dari sejumlah record. Masing-masing record bisa terdiri dari satu atau beberapa field. Setiap field terdiri dari satu atau beberaoa byte.

Operasi Dasar

Operasi dasar pada file pada dasarnya terbagi menjadi tiga tahap. yaitu :

- Membuka atau mengaktifkan file
- Melaksanakan pemrosesan file
- Menutup file

Membuka File

Sebelum suatu file dapat diproses, file harus dibuka terlebih dahulu. Namun, sebelum hal ini dapat dilakukan perlulah untuk mendefinisikan obyek file. Salah satu bentuk pernyataan yang diperlukan :

```
ofstream nama_obyek;
```

Dalam hal `ofstream` adalah nama kelas yang disediakan C++ untuk menangani operasi keluaran.

Setelah suatu obyek file diciptakan, file dapat dibuka dengan cara semacam berikut :

```
file_keluaran.open(“Mahasiswa.TXT”);
```

dalam hal ini :

- `file_keluaran` adalah nama obyek file.
- `Mahasiswa.TXT` adalah nama file yang hendak dibuat pada harddisk.

Menulis ke file

Salah satu jenis pemrosesan pada file adalah menulis atau merekam data ke file. Contoh operasi untuk merekam ke file :

```
file_keluaran<<”Mahasiswa, mahasiswa “<<endl;
```

bentuk di atas menyerupai :

```
cout<<”Mahasiswa, mahasiswa “<<endl;
```

dengan bagian cout diganti oleh file_keluaran. Hasilnya, tulisan :

```
Mahasiswa, mahasiswa.
```

dikirim ke file_keluaran (dalam hal ini yaitu file `Mahasiswa.TXT`)

Menutup File

Setelah pemrosesan file berakhir, file perlu ditutup. Langkah ini dilakukan dengan memanggil fungsi anggota bernama **close().**

Contoh :

```
file_keluaran.close();
```

merupakan pernyataan untuk menutup `file_keluaran`.

Sesudah keempat perintah dasar yang berhubungan dengan file telah diketahui, sekarang saatnya untuk mencoba membuat program yang utuh.

```
//Program untuk merekam string ke file
//Nama file file1.cpp

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

void main()
{
     ofstream file_keluaran;
```

```
    file_keluaran.open ("Mahasiswa.TXT");

    cout<<"Sedang  merekam ..  "<<endl;
    file_keluaran<<" Haloo, Halloo " <<endl;
    file_keluaran<<" Saya Adi  " <<endl;
    file_keluaran<<" Saya seorang mahasiswa " <<endl;
    file_keluaran<<" Teknik Informatika " <<endl;

    file_keluaran.close();
}
```

Membaca File

Apabila kita bermaksud membaca sebuah file, pertama-tama kita harus menciptakan obyek berkelas *ifstream*.

Contoh :

```
    ifstream file_masukan;
```

Setelah obyek file_masukan diciptakan, file dapat dibuka dengan cara memanggi fungsi anggota **open()**:

```
    file_masukan.open(“Mahasiwa.TXT”);
```

Pada contoh ini, `Mahasiswa.TXT` dibuka dengan modus masukan (sebab kelas yang digunakan adalah *ifstream*).

Seperti halnya pada kelas *ofstream*, penciptaan obyek berkelas *ifstream* dan pembukaan file dapat disederhakan dengan penulisan pernyataan :

```
    ifstream file_masukan (“Mahasiswa.TXT”);
```

Setelah file berhasil dibuka, file dapat dibaca. C++ menyediakan sejumlah fungsi anggota untuk membaca file. Jenis fungsi yang digunakan tergantung tipe data yang ada pada file. Pada contoh sebelumnya, tipe data berupa string. Data seperti ini dapat dibaca dengan menggunakan fungsi anggota **getline().** Sebagai contoh dapat dilihat pada program berikut :

```
//Program untuk membaca file
//Nama file Baca.cpp

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

void main()
```

```
{
      clrscr();

      const int MAKS = 80;
      char penyangga[MAKS+1];

      ifstream file_masukan ("Mahasiswa.TXT");

      while (file_masukan) //Baca seluruh data
      {
            file_masukan.getline(penyangga, MAKS);
            cout<<penyangga<<endl;
      }

      file_masukan.close();
}
```

Menambah Data

Telah disinggung di depan bahwa apabila program yang mengandung pernyataan :

```
ofstream file_keluaran (“Mahasiswa.TXT”);
```

Dijalankan lebih dari satu kali, maka isi semula dari file akan dihapus. Kemudian, apakah ini berarti tidak ada mekanisme untuk menambahkan data di kemudian hari ? Tentu saja ada. Agar data yang ada pada file tidak terhapus, pada bagian **open()** perlu ditambahkan argumen kedua berupa :

```
ios::app
```

misalnya :

```
file_keluaran.open(“Mahasiswa.TXT”, ios::app);
```

atau :

```
ofstream file_keluaran(“Mahasiswa.TXT”, ios ::app);
```

Contoh Program :

```
//Program menambah data dalam file
//Nama File datafile.cpp

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

void main()
{
      ofstream file_keluaran("Mahasiswa.TXT",  ios::app);

      file_keluaran<<"Saya sangat suka program "<<endl;
      file_keluaran<<"Saya sedang belajar C++ "<<endl;
```

```
      file_keluaran<<"Saya ingin menjadi programer "<<endl;

      file_keluaran.close();
}
```

Memformat String

String yang disimpan ke file dapat diformat. Pemformatan dapat dilakukan dengan memanfaatkan manipulator yang biasa digunkan pada cout.

Adapun contoh berikut menunjukkan cara menyimpan data dengan format sebagai berikut :

Nama	Honor
Edi Sumeno	150000
Andi Sentanu	90000
Ria Karunia	8000
Ali Akbar	25000

Contoh Program :

```
//Program memformat string ke file
//Nama File FileHonor.cpp

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <conio.h>

void main()
{
      struct
      {
            char nama[45];
            long honor;
            } gaji_honorer[] = { {"Edi Sumeno ", 150000 },
                                 {"andi Sentanu ",
                                 90000},
                                 {"Ria  karunia  ",
                                 8000},
                                 {"Ali   Akbar   ",
                                 25000}
                               };
            clrscr();

            ofstream fhonor("HONOR.TXT");
```

```
            fhonor<<setiosflags(ios::left)<<setw(16)
                        <<"N A M A "<<"   Honor    "<<endl;

            for(int i = 0; i <4; i++)
                  fhonor<<setiosflags(ios::left)   <<setw(16)
                              <<gaji_honorer[i].nama
                              <<resetiosflags(ios::left)
                              <<setiosflags(ios::right)<<setw(8)
                              <<gaji_honorer[i].honor<<endl;

                  fhonor.close();
      }
```

Operasi Berbasis Obyek

Suatu obyek juga dapat disimpan ke dalam file. Pada bagian ini membahas cara merekam dan juga cara membacanya kembali.

Merekam

Obyek dapat direkam ke dalam file dengan menggunkan fungsi **write().** Bentuk pernyataannya :

```
obyek.write((char *)&obyek, sizepf(obyek));
```

Contoh Program :

```
//Program untuk menyimpan obyek ke file
//Nama File rekamObyek.cpp

#include <iostream.h>
#include <fstream.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

class Buku
{
      private :
            char kode[10];
            char judul[35];
            char pengarang[25];
            int jumlah;
      public :
            void entri_buku();
};

void rekam_buku(Buku buku);
```

```
void main()
{
      Buku buku_perpustakaan;
      rekam_buku(buku_perpustakaan);
}

void Buku ::entri_buku()
{
      char tmp[15];

      clrscr();

      cout<<"<<Merekam Data Buku >>"<<endl;
      cout<<endl;

      cout<<"Kode           :  ";
      cin.getline(kode, sizeof(kode));
      cout<<"Judul          :  ";
      cin.getline(judul, sizeof(judul));
      cout<<"Pengarang      :  ";
      cin.getline(pengarang, sizeof(pengarang));
      cout<<"Jumlah         :  ";
      cin.getline(tmp, sizeof(tmp));
      jumlah = atoi(tmp);
}

//--Untuk merekam data buku ke file

void rekam_buku(Buku buku)
{
      char jawab;

      // Buka dengan modus penambahan

      ofstream file_buku("BUKU.DAT", ios::app);

      for(; ; )
      {
            buku.entri_buku();

            file_buku.write((char *)&buku, sizeof(buku));

            //Pertanyaan untuk mengulang

            cout<<endl;
            cout<<"Mau memasukkan data lagi (Y/T) ? ";
            do
            {
                  jawab = toupper (getch());
             }
            while (!((jawab == 'Y') ||(jawab == 'T')));
```

```
            cout<<jawab<<endl;

            if(jawab == 'T')
                  break;
      }

      file_buku.close(); //Tutup file
}
```

Membaca

Membaca data obyek yang ada pada file dapat dilakukan dengan menggunakan fungsi anggota **read().** Bentuk pernyataannya.

```
obyek_read((char *)&obyek, sizeof(obyek));
```

Program berikut menunjukkan cara membaca file BUKU.DAT yang terbentuk oleh program sebelumnya.

```
//Program untuk membaca data obyek dari file
//Nama File bacabuku.cpp

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

class Buku
{
      private :
            char kode[10];
            char judul[35];
            char pengarang[25];
            int jumlah;
      public :
            void info_buku();
};

void baca_buku(Buku buku);

void main()
{
      Buku buku_perpustakaan;
      baca_buku(buku_perpustakaan);
}

void Buku ::info_buku()
{
      cout<<"Kode           :  " <<kode<<endl;
      cout<<"Judul          :  "<<judul<<endl;
      cout<<"Pengarang      :  "<<pengarang<<endl;
      cout<<"Jumlah         :  "<<jumlah<<endl;
      cout<<endl;
```

```
}

//--Untuk merekam data buku ke file

void baca_buku(Buku buku)
{
      ifstream file_buku("BUKU.DAT");

      cout<<"<<Daftar Buku >>"<<endl;
      cout<<endl;

      file_buku.read((char *)&buku, sizeof(buku));
      while (!file_buku.eof())
      {
            buku.info_buku();
            file_buku.read((char *)&buku, sizeof(buku));
      }

      file_buku.close();
}
```

Pertanyaan

1. Bagaimana cara Anda membuat file pada C++ ?
2. Apa yang terjadi terhadap pointer file sewaktu Anda membaca dari sebuah file ?
3. Benar atau salah: **read()** yang membaca file data yang dihasilkan oleh **write()** biasanya berisi argumen-argumen yang sama dengan **write().** Berikan penjelasan?
4. Perusahaan Anda ingin memberikan penghargaan terhadap lima peringkat tertinggi untuk bulan lalu dari para penjualanya (anggaplah bahwa tak ada yang namanya lebih dari 10 karakter). Kemudian program harus menuliskan nama-nama tersebut ke dalam file disk ?

Struktur

Konsep

Struktur bermanfaat untuk mengelompokkan sejumlah data dengan tipe yang berlainan. Sebuah contoh pendeklarasian struktur dapat dilihat dibawah ini :

```
struct data_tanggal
{
      int tahun;
      int bulan;
      int tanggal;
};
```

Pada contoh ini, dideklarasikan sebuah struktur bernama data_tanggal yang terdiri dari tiga buah anggota berupa :

```
tahun
bulan
tanggal
```

Mendefinisikan variabel struktur

Apabila suatu struktur telah dideklarasikan, struktur ini dapat digunakan untuk mendefinisikan suatu variabel. Misalnya,

```
data_tanggal tanggal_lahir;
```

Merupakan pendefinisian variabel struktur bertipe struktur tanggal lahir. Dengan adanya pendefinisian ini, `tanggal_lahir` memiliki tiga buah anggota yaitu :

```
tahun
bulan
tanggal
```

Struktur di dalam struktur

Suatu struktur juga bisa mengandung struktur yang lain. Sebagai gambaran ditunjukkan di bawah ini.

```
struct data_pegawai
{
```

```
      int nip;
      char nama[25];
      data_tanggal tanggal_lahir;
}     rec_peg;
```

Mengakses Anggota Struktur

Anggota struktur diakses dengan menggunakan bentuk :

```
variabel_struktur.nama_anggota
```

Tanda titik diberikan diantara nama variabel struktur dan nama anggota. Misalnya :

```
tanggal_lahir.tanggal = 1;
```

merupakan pernyataan penugasan untuk memberikan nilai1 ke anggota tanggal pada variabel struktur `tanggal_lahir`.

Bagaimana halnya untuk mengakses anggota bernama bulan pada variabel struktur `rec_peg` seperti pada contoh di depan ?

Misalnya

```
rec_peg.tanggal_lahir.bulan = 9;
```

Merupakan contoh untuk memberikan nilai terhadap anggota bulan pada variabel struktur `rec_peg`.

Contoh program

```
//Program pendeklarasian, pendefinisian dan
//pengaksesan struktur
//Nama File struk1.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      struct data_tanggal   //pendeklarasian
      {
            int tahun;
            int bulan;
            int tanggal;
      };

      data_tanggal tanggal_lahir;//pendefinisian struktur

      //pengaksesan anggota struktur
```

```
      tanggal_lahir.tanggal = 1;
      tanggal_lahir.bulan = 9;
      tanggal_lahir.tahun = 1964;

      cout<<tanggal_lahir.tanggal<<'/'
            <<tanggal_lahir.bulan<<'/'
            <<tanggal_lahir.tahun<<endl;

}
```

Penugasan struktur

Pemberian nilai terhadap suatu struktur dapat dilakukan dengan bentuk :

```
      var1 = var2;
```

Sepanjang kedua variabel adalah variabel struktur bertipe sama. Misalnya terdapat pendefinisian :

```
      data_tanggal tgl1 tgl2;
```

Penugasan seperti berikut :

```
      tgl2 = tgl1;
```

Diperkenankan. Dalam hal ini, seluruh anggota pada variabel tgl2 diisi dengan anggota terkait yang ada pada tgl1. Pernyataan di atas merupakan penyederhanaan dari tiga pernyataan berikut :

```
      tgl2.bulan = tgl1.bulan;
      tgl2.tahun = tgl1.tahun;
      tgl2.tanggal = tgl1.tanggal;
```

Contoh program :

```
//Program penugasan struktur
//Nama File Struk2.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      struct data_tanggal   //pendeklarasian
      {
            int tahun;
```

```
            int bulan;
            int tanggal;
      };

      data_tanggal tgl1, tgl2;   //pendefinisian struktur

      //penugasan per anggota

      tgl1.tanggal = 1;
      tgl1.bulan = 9;
      tgl1.tahun = 1964;

      //penugasan antarvariabel struktur

      tgl2 = tgl1;

            cout<<tgl2.tanggal<<'/'
             <<tgl2.bulan<<'/'
             <<tgl2.tahun<<endl;
}
```

Union

Union menyerupai struktur (termasuk dalam hal pengaksesannya), namun mempunyai perbedaan yang nyata. Union biasa dipakai untuk menyatakan suatu memori dengan nama lebih dari saru Sebagai gambaran, sebuah `union` dideklarsikan sebagai berikut :

```
      Union bil_bulat
      {
            unsigned int di;
            unsigned char dc[2];
      }
```

Pada pendeklarsian seperti ini, `did` dan `dc` menempati memori yang sama.

Contoh program :

```
//Program union
//Nama File union.cpp

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

union bil_bulat
{
      unsigned int di;
      unsigned char dc[2];
};
```

```
void main()
{
      clrscr();

      bil_bulat bil_x; //pendefinisian union

      bil_x.di = 0x2345;

      cout<<setiosflags(ios::showbase);
      cout<<hex<<"di   :    "<<bil_x.di<<endl;
      cout<<hex<<"dc[0]:    "<<int(bil_x.dc[0])<<endl;
      cout<<       "dc[1]:   "<<int(bil_x.dc[1])<<endl;
}
```

Inisialisasi Union

Seperti halnya struktur, variabel union juga dapat diinisialisasi saat didefinisikan. misalnya :

```
      union bil_bulat bil x = { 0x2728 );
```

Memberikan nilai 0x2728 ke anggota di pada variabel union bil_bulat.

Contoh program

```
//Program inisialisasi union
//Nama file union2.cpp

#include <iostream.h>
#include <conio.h>


union bil_bulat
{
      unsigned int di;
      unsigned int dc[2];
};

void main()
{
      clrscr();

      bil_bulat bil_x = { 0x2345 }; //inisialisasi


      cout<<hex<<"di   :    "<<bil_x.di<<endl;
      cout<<hex<<"dc[0]:    "<<int(bil_x.dc[0])<<endl;
      cout<<       "dc[1]:   "<<int(bil_x.dc[1])<<endl;
}
```

Pertanyaan

1. Apa yang dimaksud dengan struktur ?

   Diketahui definisi struktur dibawah ini (soal 2-6 berdasarkan yang detahui) :

```
struct S
{
      int I;
      char *c;
      char c2[100];
      float x;
      long int l;
}
```

2. Berapakah struktur yang didefinisikan dari yang diketahui di atas ?
3. Berapakah anggota yang ddidefinisikan ?
4. Berapakah jumlah variabel struktur yang didefinisikan ? Jika ada, apa saja namanya ?
5. Berapakah nama struktur yang didefinisikan ? Jika ada, apa saja namanya ?
6. Apakah mungkin Anda menempatkan kode ini secara lokal atau global ?

Pencarian

Penerapan dari konsep pemrograman sebelumnya, dapat digunakan untuk berbagai macam permasalahan. Pada bagian ini akan dibahas penerapan ke dalam bentuk pencarian.

Konsep

Pencarian merupakan proses yang fundamental dalam pemrograman, guna menemukan data (nilai) tertentu di dalam sekumpulan data yang bertipe sama. Fungsi pencarian itu sendiri adalah memvalidasi (mencocokan) data. Sebagai contoh, untuk menghapus atau mengubah sebuah data di dalam sekumpulan nilai, langkah pertama yang harus ditempuh adalah mencari data tersebut, lalu menghapus atau mengubahnya. Contoh lain adalah penyisipan data ke dalam kumpulan data, jika data telah ada, maka data tersebut tidak akan disisipkan, selainnya akan disisipkan ke dalam kumpulan data tersebut.

Ada sebuah kasus sederhana, misalkan terdapat 10 data yang bertpe integer, terangkum di dalam variabel larik L. Terdapat data X di dalam larik L tersebut. Bagaimana proses pencarian data X tersebut ? Jika ketemu maka akan mengeluarkan pesan teks “ **Data ditemukan !** “ atau jika tidak ditemukan akan mengeluarkan pesan teks “ **Data tidak ditemukan** “. Serta menampilkan di elemen ke beberapa elemen tersebut ditemukan, dan berapa jumlah data X di larik L.

Ada beberapa metode mencari data di dalam sekumpulan data yang bertipe sama yaitu :

1. Metode Pencarian Beruntun (Sequential Search)
2. Metode Pencarian Bagi Dua (Binary Search)

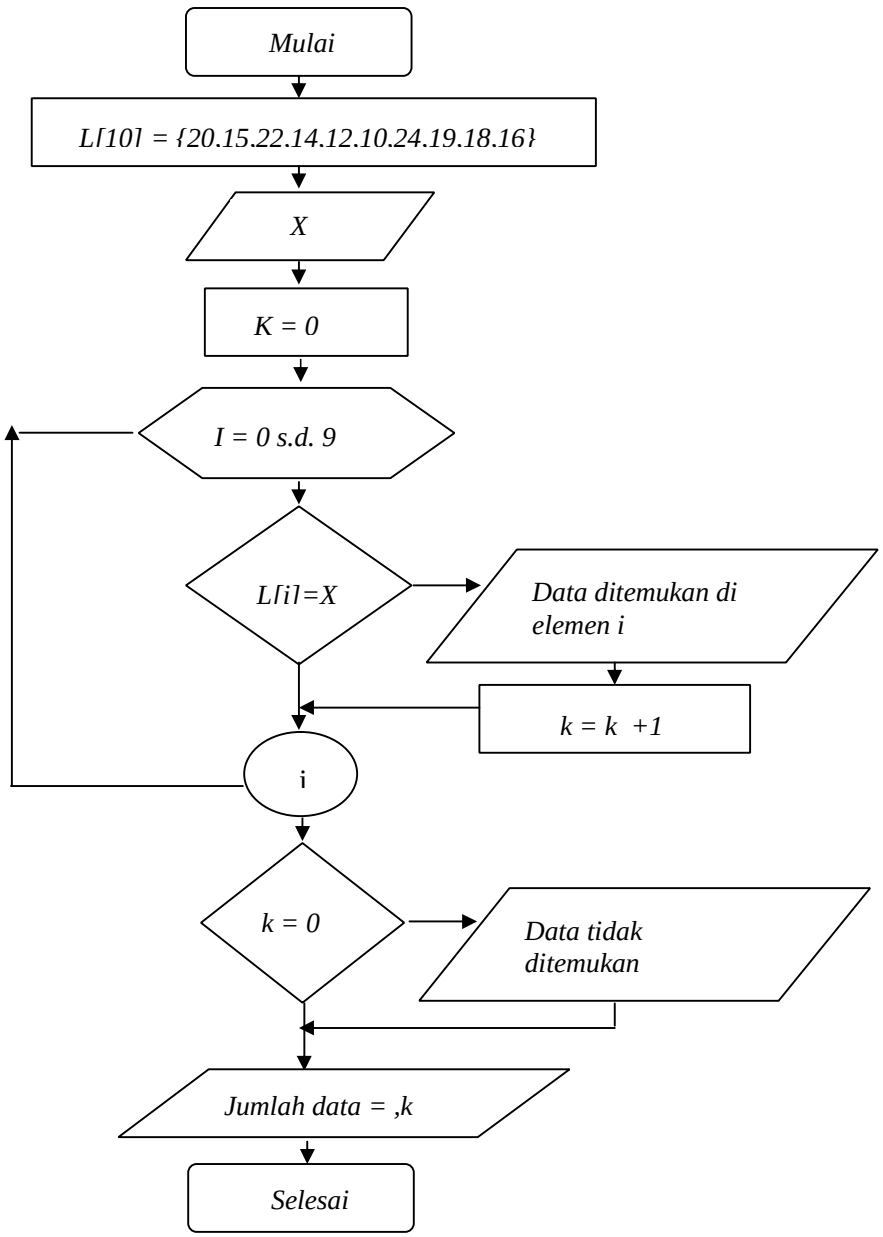
Metode Pencarian Beruntun

Konsep yang digunakan dalam metode ini adalah membandingkan data-data yang ada dalam kumpulan tersebut, mulai dari elemen pertama sampai elemen ditemukan, atau sampai elemen terakhir. Perhatikan alur di bawah ini :

20	15	22	14	12	10	24	19	18	16

Data yang akan dicari adalah X, misal X = 10, maka elemen yang diperiksa adalah elemen yang bernilai 10.

Flowchart



Keterangan

Dari flowchart di atas dapat dilihat bahwa pada saat terjadi pengulangan, nilai L[i] akan diperiksa, apakah sama dengan nilai X yang diinput. Jika nilainya sama,maka akan mengeluarkan pesan "**Data ditemukan di elemen i**" dan langsung menambahkan variabel k dengan nilai 1. Apa fungsi variabel k disini ?. Variabel k berfungsi untuk menjumlah ada berapa banyak nilai X yang terdapat di larik L, karena itulah ada nilai awal, yaitu k = 0. Setelah perulangan selesai, maka akan diperiksa apakah jumlah k = 0 ? Jika bernilai benar, maka akan mengeluarkan pesan "**Data tidak ditemukan**". Dan sebelum flowchart berakhir, program akan mengeluarkan jumlah nilai X dilarik L.

Contoh program :

```
//Program pencarian data
//Nama File Cari.cpp

#include <iostream.h>
#include <conio.h>

void main()
{
      int X,i,k;
      int L[10] = {20,15,22,14,12,10,24,19,18,16};

      cout<<"data yang akan dicari = ";cin>>X;
      k=0;
      for(i=0;i<=9;i++)
      {
            if(L[i]==X)
            {
                  cout<<"Data ditemukan di elemen "<<i<<endl;
                  k++;
            }
      }
      if(k==0)
      {
            cout<<"Data tidak ditemukan \n";
      }
      cout<<"Jumlah data yang ditemukan = "<<k<<endl;

      getch();

}
```

Secara umum, pencarian beruntun kinerjanya relatif lambat, dikarenakan adanya proses pengulangan dalam metode tersebut. Bayangkan jika ada lebih dari 100.000 data, artinya akan ada 100.000 kali pengulangan. Jika dalam proses satu pengulangan membutuhkan 0,01 detik, maka akan memakan waktu sekitar 1000 detik atau 16,7 menit. Karena itulah, untuk pencarian dengan data yang besar, metode ini tidak digunkan oleh programmer.

Metode Pencarian bagi Dua (Binary Search)

Metode ini diterapkan pada sekumpulan data yang sudah terurut (menaik atau menurun). Metode ini lebih cepat dibandingkan metode pencarian beruntun. Data yang sudah terurut menjadi syarat mutlak untuk menggunakan metode ini.

Konsep dasar metode ini adalah membagi 2 jumlah elemennya, dan menentukan apakah data yang berada pada elemen paling tengah bernilai sama, maka langsung data tengah dicari ditemukan. Jika data di elemen terurut naik, maka jika data yang berada di tengah kurang dari data yang dicari, maka pencarian selanjutnya berkisar di elemen tengah ke kanan, dan begitu seterusnya sampai ketemu atau tidak sama sekali. Dan sebaliknya untuk nilai data yang berada di tengah lebih dari data yang dicari, maka pencarian selanjutnya berkisar di elemen tengah ke kiri, dan begitu seterusnya sampai ketemu atau tidak sama sekali. Dan demikian sebaliknya untuk data yang terurut menurun. Dalam hal ini tentukan indeks paling awal dan indeks paling akhir, untuk membagi 2 elemen tersebut.

Indeks awal = i, dimana nilai i, pada awalnya bernilai 0;
Indeks akhir =j, dimana nilai j, pada awalnya bernilai sama dengan jumlah elemen.

Langkah-langkah untuk metode pencarian bagi dua.

1. Asumsikan data terurut secara horizontal dari indeks 0 samapi n-1, untuk menggunkan istilah kanan dan kiri.
2. Misalkan kumpulan data yang berjumlah n adalah larik L, dan data yang akan dicari adalah X.
3. Tentukan nilai indeks awal i=0 dan indeks akhir j = n-1.

4. Tentukan apakah data terurut menurun atau meniak dengan menggunakan membandingkan apakah elemen paling kiri L[0] lebih dari atau kurang dari eleemn paling kanan L[n-1].
   - Jika data di elemen paling kiri L[0] > data di elemen paling kanan L[n-1], maka data terurut menurun.
   - Jika data elemen paling kiri L[0] < data di elemen paling kanan L[n-1], maka data terurut menaik.
5. Asumsikan bahwa data terurut menaik (tergantung hasil nomor 3).
6. Misalkan variabel k adalah indeks paling tengah, diperoleh dengan rumus :

$$K = (I + j) \text{ div } 2$$

7. Periksa, jika L[k] = x, maka data dicari langsung ketemu di elemen k.
8. Jika nomor 7 tidak terpenuhi, periksa jika L[k]<X, maka pencarian berikutnya dilakukan di sisi kanan indeks k, lakukan proses seperti pada nomor 6, dimana nilai indeks I sekarang sama dengan nilai indeks sebelumnya.

$$I=k$$

$$K=(i+j)\text{div}2$$

   Dan seterusnya sampai nilai X dicari ketemu atau tidak sama sekali.
9. Jika nomor 8 tidak terpenuhi, maka tidak pasti nilai Lk]>X, maka pencarian berikutnya dilakukan di sisi kiri indeks k, lakukan proses seperti pada nomor 6, dimana nilai indeks j sekarang sama dengan nilai indeks k sebelumnya.

$$J=k$$

$$K=(i+j)\text{div}2$$

   Dan seterusnya sampai nilai X dicari ketemu atu tidak sama sekali.
10. Jika data terurut menurun, maka tukar kondisi yang ada di nomor 8 dan 9.

Contoh :

Diberikan 10 data terurut L[10] = {12,14,15,17,23,25,45,67,68,70}. Cari nilai X = 14 di elemen tersebut.

Solusi :

1. Menentukan apakah data terurut menaik atau menurun.

$$L[0] = 12$$

$$L[9] = 70$$

Karena L[0] < L[9], maka data tersebut terurut menaik.

2. Misal indeks paling kiri adlah I = 0 dan indeks paling kanan adalah j = 9, maka indeks tengahnya adalah :

$$K = (i+j) \text{ div } 2$$
$$= (0+9) \text{ div } 2$$
$$= 4.$$

   Elemen tengah sekarang adalah 4 dengan L[4] = 23.

3. Karena data di indeks tengah lebih dari nilai data yang dicari (L[4] > X), maka pencarian berikutnya dilakukan pada sisi kiri indeks k, maka nilai j sekarang sama dengan k, lalu lakukan proses sama seperti nomor 2.

$$J=k$$
$$= 4$$
$$K= (i+j) \text{ div } 2$$
$$= (0 +4) \text{ div } 2$$
$$= 2$$

   Elemen tengah sekarang adalah 2 dengan L[2] = 15.

4. Karena data di indeks tengah lebih dari nilai data yang dicari (L[2] >X), maka pencarian berikutnya dilakukan pada sisi kiri indeks k, maka nilai j sekarang sama dengan k, lalu lakukan proses sama seperti nomor 2.

$$J=k$$
$$= 2$$
$$K = (i+j) \text{ div } 2$$
$$= (0 +2) \text{ dic } 2$$
$$= 1$$

   Elemen tengah sekarang adlah 1 dengan L[1} = 14

5. Karena nilai data di elemen tengah sama dengan nilai data yang dicari X, maka pencarian berakhir. Data X ditemukan di iNdeks ke-1.

Contoh program :

```
//Program pencarian bagi dua
//Nama file bagidua.cpp
#include <stdio.h>
#include <conio.h>
```

```
int main()
{
      int X,i,j,k,p;
      int L[10] = {12,14,15,17,23,25,45,67,68,70};

      if(L[0]<L[9])
      {
            printf("Data terurut menaik \n");
            p=0;
      }
      else
      {
            printf("Data terurut menurun \n");
            p=1;
      }

      printf("Data yang akan dicari = ");scanf("%d",&X);

      i = 0;
      j = 0;

      do
      {
            k = (i+j)/2;
            if (p==0)
            {
                  if(L[k]==X)
                  {
                        printf("data ditemukan di elemen %d",k);
                        getch();
                        return 0;
                  }
                  else if (L[k]<X)
                  {
                        i=k;
                  }
                  else
                  {
                        j=k;
                  }
            }
            else
            {
                  if(L[k]==X)
                  {
                        printf("Data ditemukan di elemen %d",k);
                        getch();
                        return 0;
                  }
                  else if(L[k]>X)
                  {
                        i=k;
                  }
```

```
                else
                {
                     j=k;
                }
           }
      }
      while(k!=0);

      printf("Data tidak ditemukan !");

      getch();
      return 0;
}
```

Terlihat dari contoh di atas, bahwa perulangan terjadi tidak sebanyak di metode pencarian beruntun, karena perulangan hanya dilakukan apabila data tidak ketemu, pada bagian indeks elemen. Artinya pada metode pencarian beruntun, perulangan dilakukan sebanyak jumlah data, sedangkan di metode bagi dua, perulangan dilakukan $^{2}\log(n)$ kali, sehingga metode pencarian bagi dua lebih cepat dibandingkan dengan pencarian beruntun.

Pertanyaan

1. Jelaskan konsep pencarian Beruntun
2. Jelaskan konsep pencarian Bagidua
3. Buat program untuk mencari bilangan terbesar dari suatu data random yang berasal dari komputer

Pengurutan

Pengurutan (*sorting*) adalah proses mengatur sekumpulan obyek menurut urutan atau susunan tertentu. Urutan tersebut dapat menaik (ascending) atau menurun (descending). Jika diberikan n buah elemen disimpan di dalam larik L, maka :

- pengurutan menaik adalah $L[0] \leq L[1] \leq L[2] \leq \ldots \leq L[n-1]$
- pengurutan menaik adalah $L[0] \geq L[1] \geq L[2] \geq \ldots \geq L[n-1]$

Pengurutan berdasarkan jenisnya, dibagi dua kategori, yaitu :

1. Pengurutan Internal, yaitu pengurutan terhadap sekumpulan data disimpan di dalam memori utama komputer.
2. Pengurutan Eksternal, yaitu pengurutan data yang disimpan di dalam memori sekunder, biasanya data bervolume besar sehingga tidak mampu dimuat semuanya dalam memori computer, disebut juga pengurutan arsip (file), karena struktur eksternal yang dipakai adalah arsip.

Karena pengaksesan memori utama lebih cepat daripada memori sekunder, maka pengurutan internal lebih cepat daripada pengurutan eksternal.

Bermacam-macam metode yang dipakai untuk melakukan pengurutan, antara lain :

- Bubble Sort
- Selection Sort
- Insertion Sort
- Heap Sort
- Shell Sort
- Quick Sort
- Merge Sort
- Radix Sort
- Tree Sort

Pada bagian ini hanya akan dibahas mengenai tiga buah metode sederhana yang mendasar, yaitu :

1. Metode Pengurutan Gelembung (Bubble Sort)
2. Metode Pengurutan Pilih (Selection Sort)
3. Metode Pengurutan Sisip (Insertion Sort)

Metode Pengurutan Gelembung (Bubble Sort)

Metode ini diinspirasi oleh gelembung sabun yang berada di permukaan air. Karena berat jenis gelembung sabun lebih ringan dibandingkan dengan berat jenis air, sehingga gelembung sabun selalu terapung di permukaan air. Prinsip pengapungan inilah yang diterapkan ke metode ini, dimana nilai yang paling rendah berada di posisi paling atas, melalui proses pertukaran.

Konsep dasar dari metode ini adalah setiap data yang ada di kumpulan, dibandingkan dengan data-data lainnya, artinya jika jumlah data sebnayak 5, maka akan terjadi perbandingan sebanyak $(5-1)^2 = 16$ kali. Untuk satu data, akan dibandingkan sebanyak 4 kali terhadap data yang lainnya.

Atau secara umum dapat ditarik rumus, untuk jumlah data sebanyak n buah, maka :

Jumlah iterasi pembandingan = $(n-1)^2$

Jika data-data tersebut disimpan di dalam larik L, maka :

1. Untuk pengurutan menaik, pembandingnya sebagai berikut :
   L[n] < L{n-1]
2. Untuk pengurutan menurun, pembandingnya sebagai berikut :
   L[n] > L[n-1]

Jika kondisi diatas terpenuhi, maka nilai data yang ada di indeks n-1 akan ditukar dengan nilai data yang ada di indeks n.

Perhatikan program pengurutan menaik di bawah ini :

Contoh program

```
//Program  pengurutan metode gelembung
//Nama file bubble.cpp

#include <stdio.h>
#include <conio.h>

int main()
{
      int i,j;
      int temp;
      int L[10]={20,15,22,14,12,10,24,19,18,16};

      for(i=1;i<=9;i++)
      {
            for(j=9;j>=1;j--)
            {
                  if(L[j]<L[j-1])
                  {
                        temp=L[j];
                        L[j]=L[j-1];
                        L[j-1]=temp;
                  }
            }
      }
      for (i=0;i<=9;i++)
      {
            cout<<L[i]<<endl;
      }
      getch();
      return 0;
}
```

Terlihat dari program di atas, bahwa terjadi pengulangan di dalam pengulangan (nested looping). Hal ini menunjukkan bahwa ada proses pembandingan sebanyak $(10-1)^2 = 81$ kali. Dimana setiap data dibandingkan dengan 8 data lainnya. Jka kondisi perbandingan terpenuhi, maka akan menukarkan posisi (indeks) data di larik L. Karena perbandingan menggunakan operator pembanding kurang dari (<), maka akan menghasilkan data larik L yang terurut menaik.

Metode ini relatif lebih lambat, karena banyaknya iterasi perbandingan yang dilakukan, namun lebih mudah dipahami

Metode Pengurutan Pilih (selection Sort)

Metode ini memiliki konsep memilih data yang maksimum/minimum dari suatu kumpulan data larik L, lalu menempatkan data tersebut ke elemen paling akhir atau paling awal sesuai pengurutan yang diinginkan. Data maksimum/minimum yang diperoleh, diasingkan ke tempat lain, dan tidak diikutsertakan pada proses pencarian data maksimum/minimum berikutnya. Perhatikan ilustrasi berikut :

Misalkan ada sekumpulan data acak berjumlah n elemen yang disimpan di dalam larik L, akan diurut menaik, maka langkah-langkah yang harus dilakukan adalah :

1. Menentukan jumlah iterasi, yaitu pass = n-2
2. Untuk setiap pass ke-I = 0,1,2, … , pass, lakukan
    a. Cari elemen terbesar (maks) dari elemen ke-i sampai ke-(n-1)
    b. Pertukaran maks dengan elemen ke-i
    c. Kurangin n sayu (n = n -1)

Rincian tiap-tiap pas adalah sebagai berikut :

- pass 0
  Cari elemen maksimum di dalam L[0 … (n-1)].
  Pertukarkan elemen maksimum dengan elemen L[n-1]
- pass 1
  Cari elemen maksimum di dalam L[0 .. (n-2)]
  Pertukarkan elemen maksimum dengan elemen L[n-2]
- pass 2
  Cari elemen maksimum di dalam L[0 .. (n-3)]
  Pertukarkan elemen maksimum dengan elemen L[n-3]

.

.

.

- pass 3
  Cari elemen maksimum di dalam L[0 .. 1]
  Pertukarkan elemen maksimum dengan elemen L[1]

Contoh program :

```
//Program pengurutan sisipan
//Nama File select.cpp

#include <iostream.h>
#include <conio.h>

int main()
{
      int i,j;
      int Imaks, maks, temp;
      int L[10] = {20,15,22,14,12,10,24,19,18,16};

      for(i=9;i>=1;i--)
      {
            Imaks = 0;
            maks = L[0];
            for(j=1;j<=i;j++)
            {
                  if(L[j]>maks)
                  {
                        Imaks =j;
                        maks = L[j];
                  }
            }
            temp=L[i];
            L[i]=maks;
            L[Imaks]=temp;
      }
      for(i=0;i<=9;i++)
      {
            cout<<L[i]<<endl;
      }
      getch();
      return 0;
}
```

Secara garis besar, metode ini relatif lebih cepat dibandingkan dengan metode gelembung, karena pertukaran hanya sekali dilakukan pada setiap pass, sehingga waktu pengurutan dapat dikurangi.

Metode Pengurutan Sisip (Insertion Sort)

Metode ini dilakukan dengan cara menyisipkan elemen larik pada posisi tang tetap. Pencarian posisi yang tepat dilakukan dengan melakukan pencarian beruntun di dalam larik.

Perhatikan tahap-tahap di bawah ini :

Misalkan ada sekumpulan data acak berjumlah n elemen yang disimpan di dalam larik L, akan diurutkan menaik, maka langkah-langkah yang harus dilakukan adalah :
Untuk setiap pass ke-I = 1,2,..,n-1 lakukan :

a. X = L[i]
b. Sisiipkan X pada tempat yang sesuai antara L[0] .. L[i]

Rincian tiap-tiap pass adalah sebagai berikut :
Dianggap pass 0 : L[0] dianggap sudah pada tempatnya

- pass 1

  x = L[1] harus dicari tempatnya yang tepat pada L[0 .. 1] dengan cara mengeser elemen L[0 … 0] ke kanan bila L[0 … 0] lebih besar daripada L[1]. Misalkan posisi yang tepat adalah k, sisipkan L[1] pada L[k].

- pass 2

  X = L[2] harus dicari tempatnya yang tepat pada L[0 … 2] dengan cara menggeser elemen L[0 … 1] ke kanan bila L[0 … 1] lebih besar daripada L[2].
  Misalkan posisi yang tepat adalah k, sisipkan L[2] pada l[k].

.
.
.

- pass n-1

  x = L[n-1] harus dicari tempatnya yang tepat pada L[0 … (n-1)] dengan cara menggeser elemen L[0 … (n-2)] ke kanan bila L[0 … (n-2)] lebih besar daripada L[n-1]. Misalkan posisi yang tepat adalah k, sisipkan L[n-1] pada L[k].

Kelemahan metode in terletak oada banyaknya operasi pergeseran yang diperlukan dalam mencari posisi yang tepat untuk elemen larik Pada setiap pass ke-1, operasi pergeseran yang diperlukan maksimum i-1 kali. Untuk larik dengan n yang besar, jumlah operasi pergeseran meningkat secara kuadratik, sehingga pengurutan sisip tidak praktis untuk volume data yang besar.

Daftar Pustaka

i. Abdul Kadir, *Pemrogramann C++*, Penerbit Andi, Yogyakarta, 2001.

ii. Andri Kristanto, *Struktur Data C++*, Penerbit Graha Ilmu, Yogyakarta, 2003.

iii. M Fachrurrozi, *Modul Algoritma dan Pemrograman II*, Program Diploma Komputer Unsri, 2004.

iv. Rinaldi Munir, *Algoritma dan Pemrograman Buku 1 dan 2*, Penerbit Informatika, Bandung, 1998