Modul Struktur Data C++

STMIK AMIKOM YOGYAKARTA

YULI ASTUTI, S.KOM

MATERI I
PENGENALAN

Bagaimana cara mengatasi masalah implementasi program dengan komputer?

- Pemahaman masalah secara menyeluruh dan persiapan data
- Keputusan operasi-operasi yang dilakukan terhadap data
- Penyimpanan data-data pada memori sehingga tersimpan dan terstruktur secara logis, operasinya efisien
- Pengambilan keputusan terhadap bahasa pemrograman mana yang paling cocok untuk jenis data yang ada

Perbedaan Tipe Data, Obyek Data & Struktur Data

Tipe data adalah jenis data yang mampu ditangani oleh suatu bahasa pemrograman pada komputer, tiap-tiap bahasa pemrograman memiliki tipe data.

Obyek Data adalah kumpulan elemen yang mungkin untuk suatu tipe data tertentu. Mis: integer mengacu pada obyek data -32768 s/d 32767, byte 0 s/d 255, string adalah kumpulan karakter maks 255 huruf

Struktur Data adalah cara penyimpanan dan pengorganisasian data-data pada memori komputer maupun file secara efektif sehingga dapat digunakan secara efisien, termasuk operasi-operasi di dalamnya.

Ciri algoritma yang baik menurut Donald E.Knuth:

- Input : ada minimal 0 input atau lebih
- Ouput : ada minimal 1 output atau lebih
- Definite : ada kejelasan apa yang dilakukan
- Efective : langkah yang dikerjakan harus efektif
- Terminate : langkah harus dapat berhenti (stop) secara jelas

MATERI II

ARRAY

1. Array 1 Dimensi

Array atau larik adalah kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data pada suatu larik disebut dengan elekmen-elemen larik. Letak urutan dari suatu larik ditunjukkan oleh suatu subscript atau index.

Deklarasi array (larik):

```
tipe_data nama_var_array [ukuran];
```

Keterangan :
tipe_data : menyatakan jenis tipe data elemen larik (int, char, float, dll)
nama_var_array : menyatakan nama variabel yang dipakai.
ukuran : menunjukkan jumlah maksimal elemen larik.

Contoh :

```
Int nilai[6];
```

Inisialisasi array :

Menginisialisasi array sama dengan memberikan nilai awal array pada saat didefinisikan.

```
int nilai[6] = {8,7,5,6,4,3};
```

bisa disederhanakan sehingga menjadi :

```
int nilai[] = {8,7,5,6,4,3};
```

Keterangan :
Contoh diatas berarti berarti anda memesan tempat di memori komputer sebanyak 6 tempat dengan indeks dari 0-5, dimana indeks ke-0 bernilai 8, ke-1 bernilai 7, dst, dan semua elemennya bertipe data integer.

Catatan: Untuk memberikan niai 0 terhadap seluruh elemen array pada saat didefinisikan, Anda dapat memberikan nilai awal 0 pada elemen pertama. Sebagai contoh:

```
Int temp[100] = {0};
```

Akan memberikan hasil pemberian nilai nol dari subscript bernilai 0 hingga 99.

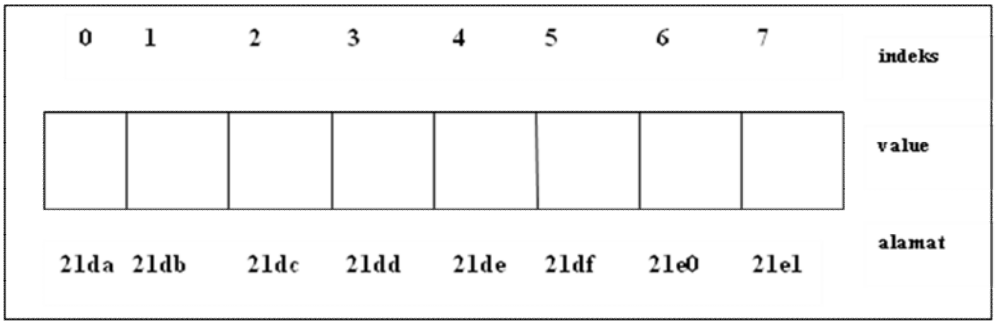
Mengakses elemen array :

```
nama_var_array [indeks];
```

Keterangan :
Pengisian dan pengambilan nilai pada indeks tertentu dapat dilakukan dengan mengeset nilai atau menampilkan nilai pada indeks yang dimaksud. Pengaksesan elemen array dapat dilakukan berurutan atau random berdasarkan indeks tertentu secara langsung.

Contoh : `nilai[2];`

Ilustrasi Array 1



Contoh 1 mengisi elemen array dan menampilkan elemen ke n:

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

void main ()
{ int y [] = {1, 2, 7, 4, 5};
  int n, r=0;
  for ( n=0 ; n<5 ; n++ )
  {
    r += y[n];
  }
  cout<<" "<<r;
  getch(); }
```

Latihan :

1. Cobalah contoh program 1
2. Siapkan 1 lembar kertas dan tulis Nama, NIM dan Kelas anda
3. Tulis alur logikanya
4. Tulis outputnya
5. Jika `r += y[n];` di ganti dengan perintah `r *= y[n];` apa hasilnya?

Contoh 2 mengedit elemen array:

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

void main ()
{ int A [5]={20,9,1986,200,13};
  Int n;
  clrscr();
  cout<<"Data yang lama\n";
  for (n=0;n<5;n++)
  {
    cout<<" "<<A[n];
  }
  cout<<"\nData yang baru : \n";
  A[0]=4;
  A[1]=2;
  A[2]=1;
  A[3]=3;
  A[4]=5;
  for (n=0;n<5;n++)
  {
    cout<<" "<<A[n];
  }
  getch();
}
```

Contoh 3 menghapus elemen array:

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

void main ()
{ int A [5]={20,9,1986,200,13};
  int n,hapus;
  clrscr();
  cout<<"Data yang lama\n";
  for (n=0;n<5;n++)
  {
    cout<<" "<<A[n];
  }
  cout<<" data yang ingin dihapus : ";
  cin>>hapus;
  cout<<"\nData yang baru : \n";
  for (n=hapus-1;n<5-1;n++)
  {
    A[n]=A[n+1];
  }
  for (n=0;n<4;n++)
  {
    cout<<" "<<A[n];
  }
  getch()}
```

Contoh 4 pencarian elemen array

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

int main(){

//deklarasi array
int A[10]={12,24,13,25,10,11,21,20,15,18};
int bil;

//menampilkan elemen array
for (int i=0;i<10;i++)
 { cout<<A[i]<<endl;}

cout<<endl;

//memasukkan nilai yang akan dicari
cout<<"Masukkan nilai yang akan dicari : ";
cin>>bil;

//pencarian data
for (int c=0;c<10;c++)
 {
   if (A[c]==bil)
     { cout<<"Nilai yang anda cari terdapat pada indek ke- "<<c;
      break;
     }
 } getch(); }
```

Langkah tugas :

1. Tulis alur logikanya
2. Tulis outputnya
3. Modifikasikan contoh program 4, sehingga kita bisa menginputkan elemen array secara manual.

Contoh 5, program fibonanci dengan array

```
#include <conio.h>
#include <iostream.h>
#define max 10

int fibo[max];

main(){
 int i;
 fibo[1]=1;
 fibo[2]=1;

 for (i=3;i<max;i++)
  { fibo [i]=fibo[i-2]+fibo[i-1]; }
 cout<<max<<" Bilangan Fibonanci Pertama ada di : \n";

 for (i=1;i<max;i++)
   { cout<<" "<<fibo[i];}
getch();}
```

Array multidimensi yaitu array yang terdiri dari beberapa subskrip atau index array. Contoh, array 2 dimensi adalah array yang mempunyai 2 index, array 3 dimensi adalah array yang mempunyai 3 index. Array seperti ini sering digunakan untuk pemrosesan matrik.

Keunggulan array :

- Array sangat cocok untuk pengaksesan acak. Sembarang elemen di array dapat diacu secara langsung tanpa melalui elemen-elemen lain.
- Jika berada di suatu lokasi elemen, maka sangat mudah menelusuri ke elemenelemen tetangga, baik elemen pendahulu atau elemen penerus 3
- Jika elemen-elemen array adalah nilai-nilai independen dan seluruhnya harus terjaga, maka penggunaan penyimpanannya sangat efisien

Kelemahan array :

- Array harus bertipe homogen. Kita tidak dapat mempunyai array dimana satu elemen adalah karakter, elemen lain bilangan, dan elemen lain adalah tipe-tipe lain
- Kebanyakan bahasa pemrograman mengimplementasikan array statik yang sulit diubah ukurannya di waktu eksekusi. Bila penambahan dan pengurangan terjadi terus-menerus, maka representasi statis

- Tidak efisien dalam penggunaan memori
- Menyiakan banyak waktu komputasi
- Pada suatu aplikasi, representasi statis tidak dimungkinkan

2. Array 2 Dimensi

Pendeklarasian array 2 dimensi :

```
tipe_data nama_var_array[batas_baris][batas_kolom];
```

- int matriks[3][4];
- int matriks2[3][4]={{5,2,1,18}, {4,7,6,-9}, {9,0,4,43}};

Contoh :

Jurusan	1992	1993	1994	1995
1. Teknik Informatika	35	45	80	120
2. Manajemen Informatika	100	110	70	101
3. Teknik Komputer	10	15	20	17

Bentuk seperti pada tabel diatas dapat dituangkan pada array berdimensi dua.

Pendefinisiannya : int data_lulus [3] [4];

Pada pendefinisian diatas : - 3 menyatakan jumlah baris (mewakili jurusan)
- 4 menyatakan jumlah kolom (mewakili tahun kelulusan).

Array hasil pendefinisian diatas dapat dinyatakan seperti di bawah ini :

TI → 0	35	45	80	120
MI→ 1	100	110	70	101
TK → 2	10	15	20	17
	0	1	2	3
	1992	1993	1994	1995

Listing program :

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int data_lulus[3][4];// array berdimensi 2
 int tahun, jurusan;
 data_lulus[0][0]=35;
 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=TI, 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=tahun-1992; //konversi ke 0,1,2,3
        break;
     }
 }
 cout<<"Jumlah yang lulus= "<<data_lulus[jurusan][tahun];
 getch();}
```

Contoh :

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \\ 10 & 12 \end{pmatrix}$$

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

int main(){

//definisi array 2dimensi
typedef int matrik32[3][2];

//deklarasi array A,B,C
matrik32 A,B,C;
int j,k;

//mengisi elemen array A
for(j=0;j<3;j++)
{ for(k=0;k<2;k++)
  {
    cout<<"A["<<j<<"]["<<k<<"] = ";
    cin>>A[j][k];
  }
}
cout<<endl;

//mengisi elemen array B
for(j=0;j<3;j++)
{ for(k=0;k<2;k++)
  {
    cout<<"B["<<j<<"]["<<k<<"] = ";
    cin>>B[j][k];
  }
}
cout<<endl;

//menjumlahkan array A dan B
for(j=0;j<3;j++)
{ for(k=0;k<2;k++)
  {
    C[j][k]=A[j][k] + B[j][k];
  }
}
//menampilkan hasil penjumlahan array
for(j=0;j<3;j++)
{ for(k=0;k<2;k++)
  {
    cout<<"C["<<j<<"]["<<k<<"]= "<<C[j][k]<<endl;;
  }
}
getch();}
```

Latihan !!!

1. Modifikasikan contoh program 6 sehingga menjadi 3 kolom dan 3 baris

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix}$$

2. Buatlah program menggunakan array 2 dimensi dari tabel berikut :

5	*	2	=	10
7	*	3	=	21
8	*	4	=	32

3. Buatlah program menggunakan array 2 dimensi unruk Perkalian Matrik (2 x 2)
4. Buatlah program menggunakan array 2 dimensi unruk Penjumlahan Matrik (3 + 3)
5. Tulis algoritma dari program di bawah ini :

```
#include <iostream.h>
#include <conio.h>
void main(){
  int i, x, ketemu=0;
  int data[]={1,2,3,4,5,6,7,8,9,10};
  cout<<"Data yang anda cari :";cin>>x;
  for(i=0; i<=9;i++)
  {   if (data[i]==x)
      {   ketemu=!ketemu;
          break; }
  }
  if (ketemu)
     cout<<"Data tersebut ada pada posisi ke: "<<i+1;
  else
     cout<<"Data tersebut tidak ada";
 getch(); }
```

MATERI IV
STRUCTURE

Struktur merupakan kumpulan elemen data yang digabungkan menjadi satu kesatuan data. Masing-masing elemen data tersebut dinamakan field atau elemen struktur. Field tersebut bisa memiliki tipe data yang ataupun berbeda, meskipun field tersebut dalam satu kesatuan tetapi tetap bisa siakses secara individu.

Struktur dan array mempunyai kesamaan dan perbedaan, kesamaannya yaitu alokasi memori untuk elemen-elemennya sudah ditentukan sebelum program dijalankan. **Perbedaannya, array adalah struktur data yang tipe data dari elemen-elemennya harus sama** dan elemen tersebut diakses melalui indeks sedangkan **struktur adalah struktur data yang tipe data dari elemen-elemennya tidak harus sama dan elemen tersebut diakses melalui identifier atau nama variabel.**

Deklarasi struktur

```
struct nama_struktur // nama struktur, kata struct harus
ada
 {
  type1 element1;  ⎫
  type2 element2;  ⎬ anggota / elemen dari struktur
  type3 element3;  ⎭
  .
  .
} nama_object;   // identifier yang digunakan untuk
                    pemanggilan struktur


------------atau--------------

struct nama_struktur
{
  type1 element1;
  type2 element2;
  type3 element3;
  .
  .
} ;
struct nama_struktur nama_object;
```

Contoh struktur dengan tipe data berbeda :

```
struct  mahasiswa
{
   char nim[10];
   char nama[25];
   char jurusan[2];
   float ipk;
}mhs;
```

Contoh struktur dengan tipe data sama :

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

---atau bisa ditulis ---

struct  tanggal
{
  int tanggal, bulan, tahun;
}tgl;
```

Contoh program 1 :

```
//program data stok
#include <iostream.h>
#include <conio.h>

struct stok {
   char nama [50];
   int jml;
}stoks ;

main ()
{
   cout<<"Masukkan nama barang   = ";cin>>stoks.nama;
   cout<<"Masukkan jumlah barang = ";cin>>stoks.jml;
   cout<<"";
   cout<<"-------------------------\n";
   cout<<"";
   cout<<"Output "<<stoks.nama<<" = ";
   cout<<stoks.jml;
   getch();}
```

Bisa juga di tulis :

```
//program data stok
#include <iostream.h>
#include <conio.h>
struct stok {
  char nama [50];
  int jml;
};
struct stok stoks;
main ()
{
  cout<<"Masukkan nama barang   = ";cin>>stoks.nama;
  cout<<"Masukkan jumlah barang = ";cin>>stoks.jml;
  cout<<"";
  cout<<"-------------------------\n";
  cout<<"";
  cout<<"Output "<<stoks.nama<<" = ";
  cout<<stoks.jml;
  getch();}
```

Contoh program 2 :

```
//program mengakses elemen struktur

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

struct data_tgl
{  int tgl,bln,thn; };

struct teman
{ char nama[20];
  char j_kel[1];
  struct data_tgl tgl; };

struct teman info;
```

```
main ()
{
  //input data
  cout<<"Masukkan nama anda = ";cin>>info.nama;
  cout<<"Jenis kelamin anda = ";cin>>info.j_kel;
  cout<<"Tanggal lahir anda = ";cin>>info.tgl.tgl;
  cout<<"Bulan lahir anda   = ";cin>>info.tgl.bln;
  cout<<"Tahun lahir anda   = ";cin>>info.tgl.thn;
  cout<<"";
  cout<<"-------------------------\n";
  cout<<"";
  //output data
  cout<<"Nama             : "<<info.nama;
  cout<<"\nKelamin        : "<<info.j_kel;
  cout<<"\nTanggal lahir : "<<info.tgl.tgl<<"-
"<<info.tgl.bln
    <<"-"<<info.tgl.thn;
  getch();}
```

Enumerasi

Enumerasi adalah tipe data yang mempunyai elemen-elemen bertipe konstnta dengan urutan yang sudah dtentukan. Nilai-nilai dari konstanta ini berupa nilai-nilai integer yang diwakili oleh pengenal yang ditulis di antara tanda kurung kurawal "{ " dan "}". Tipe ini dideklarasikan dengan kata kunci **enum.**

Deklarasi Enumerasi :

Enum nama_enumerasi{nilai1,nilai2,…}

Contoh program 3 :

```
//program mengakses elemen struktur
#include <iostream.h>
#include <conio.h>

enum j_kel{pria,wanita} ;
int main() {

struct siswa
{ int nis;
  char nama[20];
  j_kel kelamin;
}A;
  //input data
  A.nis=123;
  A.nama=="Yuli";
  A.kelamin=wanita;
  //output data
  cout<<"\nNis       : "<<A.nis<<endl;
  cout<<"Nama      : "<<A.nama;
  cout<<"\nKelamin   : "<<A.kelamin;
  getch();}
```

Latihan :

Buatlah program untuk menghitung spp mahasiswa menggunakan struktur, diketahui :

a. D3

- spp tetap Rp 500.000
- spp var Rp 25.000/sks

b. S1

- spp tetap Rp 750.000
- spp var Rp 50.000/sks

Jawaban :

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

struct mhs
{ char nama[20],nim[10],jurusan[2];
  int sks,program; };
  struct mhs bayar;
main ()
{
  int bts,var,tetap;

    //input data
    cout<<"\nNama mhs        = ";cin>>bayar.nama;
    cout<<"NIM              = ";cin>>bayar.nim;
    cout<<"Jurusan[TI,MI,SI] = ";
    cin>>bayar.jurusan;
    input:
    cout<<"Program[1=D3,2=S1]= ";
    cin>>bayar.program;
    if (bayar.program < 0 || bayar.program > 2)
       {cout<<"Program tidak sesuai\n";
        goto input;}
    cout<<"Jumlah sks        = ";cin>>bayar.sks;

    if (bayar.program==1)
       {tetap=500000;
        var=bayar.sks*25000;}
    else if (bayar.program==2)
       {tetap=750000;
        var=bayar.sks*50000;}
    cout<<"";

  //output data
    cout<<"\n\n-----------------------\n";
    cout<<"       Output  ";
    cout<<"\n-----------------------\n";
    cout<<"\nNama mhs     = "<<bayar.nama;
    cout<<"\nNIM          = "<<bayar.nim;
    cout<<"\nJurusan      = "<<bayar.jurusan;
    cout<<"\nProgram      = "<<bayar.program;
    cout<<"\nJumlah sks   = "<<bayar.sks;
    cout<<"\nSpp tetap    = "<<tetap;
    cout<<"\nSpp variabel = "<<var;
    cout<<endl;
  getch();}
```

Kuis 1 :

Buat program untuk menghitung jumlah nilai akhir mahasiswa menggunakan structure dengan ketentuan :

Nilai akhir = (10%*tugas) + (20%*kuis) + (30%*mid) + (40%*uas)

Nilai Huruf : Nilai akhir >85 : A

85 >= nilai akhir > 70 : B

70 >= nilai akhir > 55 : C

55 >= nilai akhir > 40 : D

Nilai akhir <=40 : E

Kuis 2 :

Buat sebuah program untuk menghitung gaji harian pegawai, bila diketahui ketentuannya sebagai berikut :

Gaji per jam = 500

Bila jumlah jam kerja hari itu > 7 jam, maka kelebihannya dihitung lembur yang besarnya 15 x gaji per jam.

Input : jumlah jam kerja

Output : gaji harian pegawai

Tentukan sendiri variabel-variabel yang dibutuhkan pada program ini.

MATERI V

STRUCT OF ARRAY

Apabila hendak menggunakan 1 struct untuk beberapa kali, ada 2 cara :

1. Deklarasi manual

```
#include <stdio.h>

typedef struct Mahasiswa {
      char NIM[8];
      char nama[50];
      float ipk; };
void main()
   { Mahasiswa a,b,c;
     ……
     …… }
```

   artinya struct mahasiswa digunakan untuk 3 variabel, yaitu a,b,c

2. Struct of array

```
#include <stdio.h>

typedef struct Mahasiswa {
      char NIM[8];
      char nama[50];
      float ipk; };
void main()
   { Mahasiswa mhs[3];
     ……
     …… }
```

   artinya struct mahasiswa digunakan untuk mhs[0], mhs[1], dan mhs[2]

Contoh :

```
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
typedef struct orang
{
 char nama[30];
 short umur;
 }org;
main()
{
 org saya[5];
 int i,x;
 for(i=0;i<=4;i++)
 {
  cout<<"Nama  : ";cin>>saya[i].nama;
  cout<<"Umur  : ";cin>>saya[i].umur;
  cout<<endl;
 }
  for(x=0;x<=4;x++)
 {
  cout<<"Data ke ["<<x<<"] "<<"bernama "
      <<saya[x].nama<<" dan berumur "
      <<saya[x].umur<<" tahun";
  cout<<endl;
 }
 getch(); }
```

Latihan :

Buatlah program untuk menghitung spp mahasiswa menggunakan struktur, diketahui :

c. D3

- spp tetap Rp 500.000
- spp var Rp 25.000/sks

d. S1

- spp tetap Rp 750.000
- spp var Rp 50.000/sks

Jawaban :

```
#include <iostream.h>
#include <conio.h>
struct mhs
{ char nama[20],nim[10],jurusan[2];
  int sks,program; };
struct mhs bayar[2];
main ()
{
  int bts,var,tetap;
  for(int i=0;i<2;i++)
  {
    //input data
    cout<<"\nNama mhs        = ";cin>>bayar[i].nama;
    cout<<"NIM              = ";cin>>bayar[i].nim;
    cout<<"Jurusan[TI,MI,SI] = ";
    cin>>bayar[i].jurusan;
    input:
    cout<<"Program[1=D3,2=S1]= ";
    cin>>bayar[i].program;
if (bayar[i].program < 0 || bayar[i].program > 2)
       {cout<<"Program tidak sesuai\n";
        goto input;}
    cout<<"Jumlah sks        = ";cin>>bayar[i].sks;
    if (bayar[i].program==1)
       {tetap=500000;
        var=bayar[i].sks*25000;}
    else if (bayar[i].program==2)
       {tetap=750000;
        var=bayar[i].sks*50000;}
    cout<<"“;
  //output data
    cout<<"\n\n-----------------------\n";
    cout<<"       Output  ";
    cout<<"\n-----------------------\n";
    cout<<"\nNama mhs     = "<<bayar[i].nama;
    cout<<"\nNIM          = "<<bayar[i].nim;
    cout<<"\nJurusan      = "<<bayar[i].jurusan;
    cout<<"\nProgram      = "<<bayar[i].program;
    cout<<"\nJumlah sks   = "<<bayar[i].sks;
    cout<<"\nSpp tetap    = "<<tetap;
    cout<<"\nSpp variabel = "<<var;
    cout<<endl<<endl;
  }  getch();}
```

MATERI VI
POINTER (VAR. PENUNJUK)

Pointer adalah suatu variabel penunjuk yang menunjuk pada suatu alamat memori komputer tertentu. Pointer merupakan variabel level rendah yang dapat digunakan untuk menunjuk nilai integer, character, float, double, atau single, dan bahkan tipe-tipe data lain yang didukung oleh bahasa C. Variabel biasa, sifatnya statis dan sudah pasti, sedangkan pada pointer sifatnya dinamis dan dapat lebih fleksibel. Variabel pointer yang tidak menunjuk pada nilai apapun berarti memiliki nilai NULL, dan disebut sebagai **dangling pointer** karena nilainya tidak diinisialisasi dan tidak dapat diprediksi.

1. Operator Alamat / Dereference Operator(&)

Setiap variabel yang dideklarasikan, disimpan dalam sebuah lokasi memori dan pengguna biasanya tidak mengetahui di alamat mana data tersebut disimpan.

Dalam C++, untuk mengetahui alamat tempat penyimpanan data, dapat digunakan tanda ampersand(&) yang dapat diartikan “alamat”.

Contoh :

Bil1 = &Bil2;

dibaca: isi variabel bil1 sama dengan alamat bil2

2. Operator Reference (*)

Penggunaan operator ini, berarti mengakses nilai sebuah alamat yang ditunjuk oleh variabel pointer.

Contoh :

Bil1=*Bil2;

dibaca: bil1 sama dengan nilai yang ditunjuk oleh bil2.

Deklarasi variabel pointer

```
tipe * nama_pointer;
```

Contoh :

```
#include <iostream.h>
main(){
  int nil1 = 5, nil2 = 15;
  int *ptr;
  ptr = &nil1;
  *ptr = 10;
  ptr=&nil2;
  *ptr=20
  cout<<"Nilai 1 = "<<nil1<<"dan nilai 2 = "<<nil2;
  return 0;}
```

Hasil : Nilai 1 = 10 dan nilai 2 = 20

Beberapa hal tentang pointer :

Operasi variabel pointer dapat dikerjakan oleh variabel pointer yang lain, contoh :

X = 10;

Ptr1 = &X;

Ptr2 = Ptr1;

Ptr1 dan Ptr2 menghasilkan alamat variabel X yang sama.

Contoh 1 :

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

int main(){
    int nilai1 = 4;
    int nilai2 = 5;
    float nilai3 = 3.5;
    char nama[11] = "abcdefghij";

    int *nilai_p1 = &nilai1;
    int *nilai_p2 = &nilai2;
    char *nilai_p4 = nama;
    float *nilai_p3= &nilai3;

    cout<<"nilai 1 = "<<*nilai_p1<<", alamat1 = "
                        <<&nilai_p1;
    cout<<"\nnilai 2 = "<<*nilai_p2<<", alamat2 = "
                          <<&nilai_p2;
    cout<<"\nnilai 3 = "<<*nilai_p3<<", alamat3 = "
                          <<&nilai_p3;
    cout<<"\nnilai 4 = "<<*nilai_p4<<", alamat4 = "
                         <<&nilai_p4;
    getch();}
```

OPERASI POINTER

1. **Operasi penugasan**

   Nilai dari suatu variabel pointer dapat disalin ke variabel pointer yang lain.

   contoh: y = 35;

   x1 = &y;

   x2 = x1;

Contoh 2:

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

int main(){

float nilai,*p1,*p2;
nilai = 14.54;

cout<<"nilai = "<<nilai<<", alamatnya "<<&nilai;
p1 = &nilai;
p2 = p1;  //operasi pemberian nilai, berarti alamat
          x2 sama dengan x1
cout<<"\nnilai p1 = "<<*p2<<", p1 menunjuk alamat "
     <<p1;
  //pada awalnya p2 masih dangling pointer
cout<<"\nmula-mula nilai p2 = "<<*p2
     <<", p2 menunjuk alamat" <<p2;
cout<<"\nsekarang nilai p2 = "<<*p2
     <<", p2 menunjuk alamat "<<p2;
getch();}
```

2. **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 (index selanjutnya) dalam memori.

Begitu juga operasi pengurangan.

Contoh 3 :

```
#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     memori "
      <<penunjuk;
  cout<<"\nNilai "<<*(penunjuk+1)
      <<" ada di alamat memori"<<penunjuk+1;
  cout<<"\nNilai "<<*(penunjuk+2)
      <<" ada di alamat memori"<<penunjuk+2;
getch(); }
```

3. **Operasi Logika**

Contoh 4 :

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

void main()
{ int *pa, *pb, a = 100, b = 10;
  clrscr();
  pa = &a;
  pb = &b;
  if (*pa < *pb)
     {cout<<"pa menunjuk ke memori lebih RENDAH
     dari pb\n";}
  else if(*pa == *pb)
     {cout<<"pa menunjuk ke memori yang SAMA dengan
     pb\n";}
  else if(*pa > *pb)
     {cout<<"pa menunjuk ke memori lebih TINGGI
     dari pb\n";}
getch(); }
```

ARRAY DAN POINTER

Contoh 5 :

```
// more pointers
#include <iostream.h>
#include <conio.h>

int main ()
{
  char array[5];
  char * p;
  p = array;  *p = 'a';
  p++;  *p = 'b';
  p = &array[2];  *p = 'c';
  p = array + 3;  *p = 'd';
  p = array;  *(p+4) = 'e';
  for (int n=0; n<5; n++)
    cout << array[n] << ", ";
  getch();}
```

Contoh 6

```
#include <iostream>
#include <conio>

const int array = 5;
int main ()
{
   int A [array];
   const int *pInt = A;
   for(int i=0;i<array;i++)
      {cout<<"Input array : ";cin>>A[i];}
       for (int n=0; n<array; n++)
         { cout << "Element [" << n << "] = "
                <<*(pInt) << endl;
       pInt++;}
getch(); }
```

:

POINTER DGN ARRAY

Contoh 7 :

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

void main(){
 int tgl_lahir[] = { 13,9,1982 };
 int *ptgl;
 ptgl = tgl_lahir; /* ptgl berisi alamat array */
 cout<<"Diakses dengan pointer\n";
 cout<<"Tanggal = "<< *ptgl;
 cout<<"\nBulan = "<< *(ptgl + 1);
 cout<<"\nTahun = "<< *(ptgl + 2);
 cout<<"\nDiakses dengan array biasa\n";
 cout<<"Tanggal = "<< tgl_lahir[0];
 cout<<"\nBulan   = "<< tgl_lahir[1];
 cout<<"\nTahun   = "<< tgl_lahir[2];
getch();}
```

PEMBERIAN NILAI ARRAY DGN POINTER

Contoh 8 :

```
//Contoh Program7
#include <iostream.h>
#include <conio.h>

void main(){
 int x[5], *p, k;
 clrscr();
 p = x;
 x[0] = 5;
 x[1] = x[0]; /*x[1]diisi dengan x[0] sehingga
               x[1] = 5 */
 x[2] = *p + 2; /*x[2] diisi dengan x[0] + 2
                  sehingga x[2] = 7 */
 x[3] = *(p+1)-3; /*x[3] diisi dengan x[1] - 3
                  sehingga x[3] = 2 */
//x[4] = *(x + 2); /*x[4] diisi dengan x[2]
                   sehingga x[4] = 7 */
 x[4]=*(p+2)-1;
 for(k=0; k<5; k++)
    cout<<"\nx["<<k<<"] = "<< x[k];
getch(); }
```

Contoh struct dengan pointer

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

struct orang
{  char nama[30],alamat[30];
   short umur; };
   main(){
   struct orang *saya ;
   int i,n;
   cout<<"Jumlah data : ";cin>>n;
   for(i=1;i<=n;i++)
   {  cout<<"Nama  : ";cin>>saya->nama;
      cout<<"Umur  : ";cin>>saya->umur;
      cout<<endl; }
       for(i=1;i<=n;i++)
          {  cout<<"Data ke ["<<i<<"] "<<"bernama "
                <<saya->nama<<" dan berumur "
                 <<saya->umur<<" tahun";
             cout<<endl;  }
getch(); }
```

Latihan buat inputan scr dinamis dari program di bawah ini :

```
// more pointers
#include <iostream.h>
#include <conio.h>

int main ()
{
  char array[5];
  char * p;
  p = array;  *p = 'a';
  p++;  *p = 'b';
  p = &array[2];  *p = 'c';
  p = array + 3;  *p = 'd';
  p = array;  *(p+4) = 'e';
  for (int n=0; n<5; n++)
    cout << array[n] << ", ";
  getch();
```

Jawaban :

```
// more pointers
#include <iostream.h>
#include <conio.h>

int main ()
{
  char array[5];
  char * p;
  p = array;
  for (int i=0;i<=4;i++)
  { cout<<"\nInput array["<<i<<"]=";cin>>*p;
    p++;  }
    for (int n=0; n<5; n++)
        cout << array[n] << ", ";
getch();}
```

MATERI VII
SORTING (PENGURUTAN)

Sorting adalah proses mengatur sekumpulan objek menurut aturan atau susunan tertentu. Urutan objek tersebut dapat menaik (ascending = dari data kecil ke data lebih besar) atau menurun (descending = dari data besar ke data lebih kecil).

Banyak sekali algoritma pengurutan yang ada, tetapi disini akan kita pelajari 3 metode yaitu:

- Bubble sort (gelembung)
- Selection sort (maksimum/minimun)
- Insertion sort (sisip)

1. PENGURUTAN GELEMBUNG

Metode pengurutan gelembung (bubble sort) diinspirasi oleh gelembung sabun yang ada di permukaan air. Karena berat jenis gelembung sabun lebih ringan daripada berat jenis air maka gelembung sabun akan selalu mengapung.

Prinsip pengapungan ini juga dipakai pada pengurutan gelembung. Elemen yang berharga paling kecil “diapungkan”, artinya diangkat ke atas (atau ke ujung paling kiri) melalui pertukaran. Proses pengapungan ini dilakukan N kali langkah. Pada langkah ke-I, Larik[1..N] akan terdiri dari 2 bagian yaitu:

- Bagian yang sudah terurut yaitu L[1]..L[i].
- Bagian yang belum terurut L[I+1]..L[n].

ALGORITMA PENGURUTAN GELEMBUNG

Untuk mendapatkan larik yan terurut **menaik**, proses yang harus dilakukan pada setiap langkah sebagai berikut:

Langkah 1: Mulai dari elemen K=N,N-1,N-2,..2 bandingkan L[K] jika L[K] < L[K-1], pertukarkan L[K] dengan L[K-1].

Pada akhir langkah 1, elemen L[1] berisi harga minimum pertama.

Langkah 2: Mulai dari elemen K=N,N-1,N-2,..3 bandingkan L[K] jika L[K] < L[K-1], pertukarkan L[K] dengan L[K-1].

Pada akhir langkah 2, elemen L[2] berisi harga minimum kedua dan L[1]..L[2] terurut..

Langkah 3: Mulai dari elemen K=N,N-1,N-2,..4 bandingkan L[K] jika L[K] < L[K-1], pertukarkan L[K] dengan L[K-1].

Pada akhir langkah 3, elemen L[3] berisi harga minimum ketiga dan L[1]..L[3] terurut..

Langkah N-1: Mulai dari elemen K=N,N-1,N-2,..4 bandingkan L[K] jika L[K] < L[K-1], pertukarkan L[K] dengan L[K-1].

Contoh: Larik dengan N=6 buah elemen dibawah ini. Larik ini akan diurutkan menaik.

25	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:

K=N=6					21	76
K=5				8	21	76
K=4			8	10	21	76
K=3		8	27	10	21	76
K=2	8	25	27	10	21	76

Hasil Akhir dari langkah 1 :

8	25	27	10	21	76
1	2	3	4	5	6

Langkah 2 : Berdasarkan hasil akhir langkah 1

K=N=6					21	76
K=5				10	21	76
K=4			10	27	21	76
K=3		10	25	27	21	76

MATERI VIII

SEARCHING (PENCARIAN)

Ada beberapa pencarian yang akan kita uraikan disini:

- Pencarian Beruntun (Sekuensial Search)
- Pencarian Beruntun dengan sentinel
- Pencarian Bagi dua (Binary Search)

Pencarian Beruntun (Sekuensial Search)

Konsep : membandingkan setiap setiap elemen larik satu per satu secara urut (beruntun), mulai dari elemen pertama sampai dengan elemen yang terakhir. Ada 2 macam pencarian beruntun,yaitu pencarian pada array yang **sudah** terurut, dan pencarian pada array yang **belum** terurut.

Contoh 1:

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

main()
{
int larik[9]={1,12,3,4,10,6,7,11,9}, i,n, x, posisi;

cout<<"data yang ingin dicari ? ";cin>>x;
i=0;
posisi=0;

while(i<8 && larik[i]!=x)
{

i++;
}
if (larik[i]!=x)
cout<<"maaf data yang dicari tidak ada";
else if (larik[i]==x)
{
 posisi=i+1;


 cout<<"pada posisi ke "<<posisi;
}
getch();
}
```

Contoh 2:

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

 int main (void) {
     int i, n, dt, posisi, ketemu = 0;
     int data[100];

     printf ("berapa banyak data di dalam array: ");
     scanf ("%i", &n);
     for (i = 0; i < n; i++) {
          printf("masukkan data ke-%i = ", (i+1));
          scanf ("%i", &(*(data+i)));
     }
     printf("data yang dicari: ");
     scanf ("%i", &dt);
     ketemu = 0;


     for (i = 0; i < n; i++) {
          if (*(data+i) == dt) {
               ketemu = 1;
               posisi = i;
               printf  ("Data:  \"%i\"  ditemukan  di
posisi= %i\n",
               *(data+i), (posisi+1));
          }
     }
     if   (ketemu   ==   0)   printf("data   tidak
ditemukan!\n");
getch(); }
```

Pencarian Beruntun Dengan Sentinel

Algoritma ini adalah pengembangan dari algoritma pencarian beruntun. Yang dimaksud dengan sentinel adalah elemen fiktif yang sengaja ditambahkan sesudah elemen terakhir dari larik tersebut. Jadi jika elemen terakhir dari larik adalah L[N], maka sentinel diletakkan pada elemen L[N+1]. Akibatnya proses pencarian akan selalu menemukan data yang dicari, akan tetapi harus selalu diperiksa letak data yang ditemukan, apakah:

1. Di antara elemen-elemen larik yang sesungguhnya(antara L[1] sampai dengan L[N])
2. Pada elemen fiktif [L[N+1]]

Contoh 3:

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

void main()
{
int array[]={1,2,3,4,5}, i=0,x;
cout<<"masukkan data yang akan dicari "; cin>>x;
array[5]=x;
 i=0;
 while(array[i]!=x)
 i++;

 if (i<5)
  {
  cout<<"Data Ketemu";
  }
 else
 cout<<"data tidak ditemukan";
getch();
}
```

Pencarian Bagi Dua (Binary Search)

Syarat : Data harus terurut

Salah satu keuntungan data yang terurut adalah memudahkan pencarian, yang dalam hal ini adalah pencarian bagi dua. Sebenarnya dalam kehidupan sehari-hari kita sering menerapkan algoritma ini. Untuk mencari kata tertentu dalam kamus (misalnya kamus bahasa Inggris), kita tidak membuka kamus tersebut dari halaman awal sampai halaman akhir satu persatu, namun kita mencarinya dengan cara membelah atau membagi halaman-halaman buku tersebut. Begitu seterusnya sampai kita menemukan kata yang dicari.

Prinsip Pencarian:

Kita asumsikan data sudah terurut, misalkan terurut menurun. Kita menyebut indeks terkecil sebagai indeks ujung paling kiri, dan indeks terbesar sebagai indeks ujung paling kanan.

Misalkan indeks kiri Ia dan indeks kanan adalah Ib. Pada mulanya Ia adalah 0 dan Ib adalah N.

Langkah 1:

Bagi 2 elemen larik pada elemen tengah. Elemen tengah adalah elemen dengan indeks k=(Ia+Ib) div 2.

(Elemen tengah, L[k], membagi larik menjadi 2 bagian L[Ia…k-1] dan bagian kanan L[k+1…Ib]).

Langkah 2:

Periksa apakah L[k]=X. Jika L[k]=X, pencarian dihentikan sebab X sudah ditemukan, tetapi jika tidak , harus ditentukan apakah pencarian pada larik bagian kiri atau larik bagian kanan. Jika L[k] < X maka pencarian dilakukan pada larik kiri. Sebaliknya jika L[k] >X maka pencarian dilakukan pada larik bagian kanan.

Langkah 3:

Ulangi langkah 1 sampai X atau Ia>Ib.

Contoh 4:

```
#include <iostream>
    #include <conio>

    int main() {
      const int arraySize = 5;
      int target;

      // Array size and values already known.

      int array[arraySize] = {1, 2, 3, 4, 5};
      int first, mid, last;

      cout << "Enter a target to be found: ";
      cin >> target;

      // Initialize first and last variables.
      first = 0;
      last = 2;

      while(first <= last)
      { mid = (first + last)/2;
        if(target > array[mid])
        { first = mid + 1; }
        else if(target < array[mid])
        { last = mid + 1;  }
        else
        { first = last + 1; }
      }
      // When dividing can no longer proceed you are
left with the
      // middle term. If the target equal that term
you're are
      // successful.

      if(target == array[mid])
      { cout << "Target found." << endl; }
      else
      { cout << "Target not found." << endl;  }
      getch();  }
```

NB : UNTUK MODUL MATERI STACK, QUEUE, LINK LIST, TREE DAN GRAPH MENYUSUL