ALGORITMA, STRUKTUR DATA DAN PEMROGRAMAN

Akmal, S.Si, MT

Mata Kuliah : Struktur Data

Tujuan

- mahasiswa dapat membuat program yang terstruktur dan modular menggunakan prosedur dan fungsi dengan tipe data dasar / sederhana menggunakan bahasa C++ dengan benar

Pokok Bahasan

- Tipe Data dan *Abstract Data Type (ADT)*
- Algoritma dan bahasa C++
- Pemrograman Terstruktur (Fungsi / Prosedur)

Algoritma dan Pemrograman

- Algoritma adalah rangkaian terurut langkah-langkah yang logis dan sistematis yang disusun untuk menyelesaikan suatu masalah
- Komputer digunakan sbg alat bantu penyelesaian suatu persoalan

  Problematika --> komputer --> penyelesaian (?)

  ==> harus ditanamkan dalam bentuk program
- Program secara umum adalah : kumpulan instruksi atau perintah yang disusun sedemikian rupa sehingga mempunyai urutan nalar yang logis untuk menyelesaikan suatu persoalan yang dimengerti oleh komputer.

3 Skema Dasar Algoritma

1. Runtunan
2. Pemilihan / Seleksi
3. Pengulangan

Penulisan algoritma :

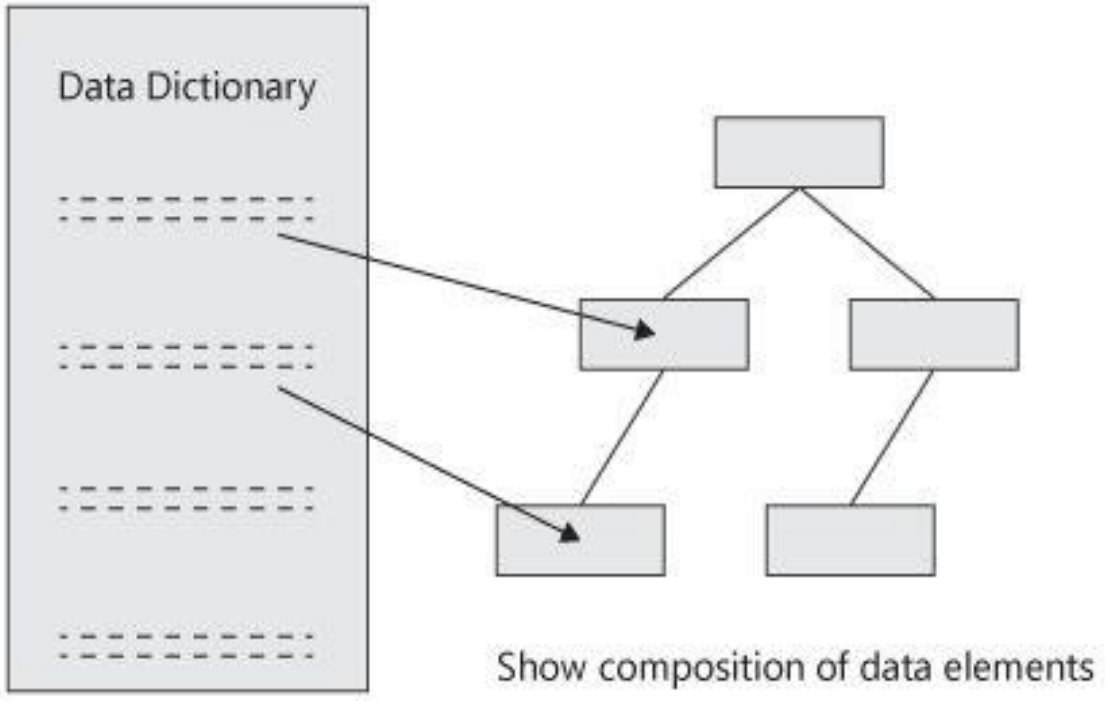
a. Diagram Alir / Flow Chart
b. Kode Semu / Pseudo Code

Algoritma yang baik

1. **Benar**, di mana algoritma menyelesaikan masalah dengan tepat, sesuai dengan definisi masukan / keluaran algoritma yang diberikan.
2. **Efisien**, berarti algoritma menyelesaikan masalah tanpa memberatkan bagian lain dari apliikasi. Sebuah algoritma yang tidak efisien akan menggunakan sumber daya (memori, CPU) yang besar dan memberatkan aplikasi yang mengimplementasikan algoritma tersebut.
3. **Mudah diimplementasikan**, artinya sebuah algoritma yang baik harus dapat dimengerti dengan mudah sehingga implementasi algoritma dapat dilakukan siapapun dengan pendidikan yang tepat, dalam waktu yang masuk akal.

Struktur Data

- Struktur Data adalah studi atau teknik yang digunakan untuk memahami bagaimana menyimpan sekelompok data secara terorganisir, sehingga dapat digunakan dengan cara yang sangat canggih untuk merancang program dan algoritma.
- Struktur data biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah kesatuan.



Type structure data

- *Arrays : Struktur data paling sederhana yang menunjukkan linearitas sebagai properti. Unsur-unsur tersebut disusun secara linier dan memiliki indeks.*
- *Pointer : digunakan ketika dibutuhkan oleh elemen dalam satu list untuk menunjuk ke elemen lain dalam list*
- *Linked list : Dalam jenis struktur data ini, sulit untuk mengambil data karena mungkin datanya masif. Dalam hal ini digunakan alat seperti pointer dan link. Elemen data terdiri atas informasi dan pengait ke elemen yang lain.*
- *Stack: (LIFO) → terakhir masuk pertama keluar. Pada tipe struktur data ini data disisipkan dan dihapus dari satu titik.*
- *Antrian: (FIFO) → yang pertama masuk yang pertama keluar. Pada data ini disisipkan dari satu sisi yang akan menjadi bagian depan dan didorong ke sisi yang lain hingga mencapai ujung yang merupakan sisi belakang. Ini memiliki dua cara, satu jalan masuk dan satu lagi jalan keluar.*
- *Pohon: Tipe ini mewakili seluruh data yang ada dalam bentuk hierarkis berdasarkan hubungan antara elemen-elemen dalam kelompok data tersebut.*
- *Graf: Ini menggambarkan hubungan antara pasangan elemen.*

Abstract Data Type (ADT)

- *Abstract Data Type* (ADT) adalah sekumpulan objek dengan sekumpulan operasi.
- ADT adalah abstraksi matematis , tidak ada penjelasan bagaimana sekumpulan operasi diimplementasikan dalam definisi ADT.
- ADT digunakan untuk memodelkan **(abstraksi)** sekumpulan data yang ditemukan dalam sebuah permasalahan.

Abstract Data Type (ADT)

- ADT adalah definisi dari TYPE dan sekumpulan operasi dasar (PRIMITIF) dari TYPE tersebut
- Definisi TYPE dari sebuah ADT dapat mengandung definisi ADT lainnya.
- Contoh :
  - ADT WAKTU terdiri atas ADT JAM dan ADT DATE
  - ADT GARIS memiliki dua buah TITIK
- TYPE diterjemahkan menjadi data type yang terdefinisi sesuai bahasa pemrograman, misalnya struct dalam C, record dalam Pascal, class dalam C++/ Java
- Contoh tipe data abstrak adalah stack, queue, list, tree, graph, dll.

Beda struktur data dan tipe data abstrak.

- Struktur data hanya memperlihatkan bagaimana data-data di organisir,
- Sedangkan tipe data abstrak (ADT) mengemas struktur data tertentu sekaligus dengan operasi-operasi yang dapat dilakukan pada struktur data tersebut.

Penulisan program dengan C++

```
/* Nama program   :
   Nama           : Akmal
   NPM            :
   Tanggal buat   :
   Deskripsi      :
**************************************************/
// deklarasi header file / Preprocessor directive
// deklarasi fungsi / void
main() {
  /* KAMUS Data*/
  /* Deskripsi Algoritma*/
}
//atau
int main() {
  /* KAMUS Data*/
  /* Deskripsi Algoritma*/
  return 0;
}
```

Program HelloWorld.cpp

```
/* Nama program   :
   Nama           : Akmal
   NPM            :
   Tanggal buat   :
   Deskripsi      :
**************************************************/
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";          //std::cout << "Hello World!";
  return 0;
}
```

Tipe primitive (tipe data dasar)

Tipe data	Byte	Batasan
char	1	Bilangan bulat / ASCII antara -128 s.d. 127
unsigned char	1	Bilangan bulat antara 0 s.d. 255
short	2	Bilangan bulat antara -32.768 s.d. 32.767 (-2^{15} s.d $2^{15}-1$)
unsigned short	2	Bilangan bulat antara 0 s.d. 65.535 (0 s.d $2^{16}-1$)
int	4	Bilangan bulat antara -2.147.483.648 s.d. 2.147.483.647 (-2^{31} s.d $2^{31}-1$)
unsigned int / unsigned	4	Bilangan bulat antara 0 s.d $2^{32}-1$
long int	4	Bilangan bulat antara -2^{31} s.d $2^{31}-1$
unsigned long int	4	Bilangan bulat antara 0 s.d $2^{32}-1$
float	4	Bilangan real antara - 3.4 E+38 s.d. 3.4E+38 (7 digit presisi)
double	8	Bilangan real antara -1.7E+308 s.d. 1.7E+308 (15 digit presisi)
bool	1	true / false ➔ Tidak semua compiler yang support

```
#include   <limits.h>
cout << ”Bilangan minimum char: ”<<CHAR_MIN<<endl;          //??
cout << ”Bilangan minimum char: ”<<UCHAR_MAX<<endl;         //??
cout << ”Bilangan maximum int : ”<<INT_MAX;                 //??
```

Input (Masukan) dan Output (Keluaran)

- Contoh : Menggunakan style bahasa C (**dengan scanf dan printf**)

```
/* Nama program          : tulisInt.c
   Nama                  : Akmal
   NPM                   : 007
   Tanggal buat          :
   Deskripsi             : Contoh membaca dan menulis  nilai
                           integer  dgn bahasa C
*************************************************/
# include <stdio.h>
main() {
 /* KAMUS */
      int a;
 /* ALGORITMA */
      printf("Contoh baca dan tulis, ketik integer :");
      scanf("%d",&a);
      printf("Nilai yang dibaca : %d \n",a);
      return 0;
}
```

Contoh : Menggunakan C++ (**dengan cin dan cout**)

```
//    Nama program      : baca.cpp
//    Nama              : Akmal
//    NPM               : 007
//    Tanggal buat      :
//    Deskripsi    : Contoh membaca dan menulis  nilai integer
//                 dengan bahasa C++
//*****************************************
# include <iostream>
main() {
 // KAMUS
     int a;
 // ALGORITMA
     cout << "Contoh baca dan tulis, ketik integer :";
     cin >>a;
     cout << "Nilai yang dibaca : " << a;
}
```

Contoh : Input string dengan getline() → type char dan string

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

main() {
 // KAMUS
     string  nama1;
     char nama2[50], alamat[50];

 // ALGORITMA
     cout << "Contoh baca dan tulis" << endl;
     cout << "Ketik Nama anda: "; getline (cin, nama1);
     cout << "Nama anda adalah : " << nama1 << endl;

     cout << "Ketik Alamat anda: "; cin.getline (alamat,50);
     cout << "Nama anda adalah : " << alamat << endl


     cout << "Ketik Nama anda lagi : "; cin >> nama2;
     cout << "Nama anda adalah : " << nama2 << endl;
}
```

Operator Aritmatika & Assignment

Simbol	Fungsi	Contoh Penggunaan
-	Pengurangan	x = x - 10;
+	Penjumlahan	x = x + 10;
/	Pembagian	x = x / 10;
*	Perkalian	x = x * 10;
%	Modulo	x = 11 % 2;
++	Increment	x++
--	Decrement	x--

```
cout << "13 % 5 = "<<(13%5);
```

Simbol	Keterangan
-=	x = x - y dapat ditulis x -= y
+=	x = x + y dapat ditulis x += y
/=	x = x / y dapat ditulis x /= y
*=	x = x * y dapat ditulis x *= y
%=	x = x % y dapat ditulis x %= y

```
int nilai=50;
nilai +=20;
cout<<"nilai = "<<nilai;
      // hasil  nilai=70
```

Operator Relasi dan Logika

Simbol	Keterangan
==	Equal (sama dengan)
!=	Not Equal (tidak sama dengan)
<	Less than (lebih kecil)
<=	Less than or equal (lebih kecil atau sama
>	dengan)
>=	Greater than (lebih besar)
	Greater than or equal (lebih besar atau sama)

```
int nilai = 100;
if (nilai==100)
     cout<<"nilai sempurna ";
```

Simbol	Keterangan
&&	AND
||	OR
!	Not

```
int nilai = 70;
if (nilai>=60 && nilai <=100)
    cout<<"Lulus ";
```

Operator Bitwise dan Shift

Simbol	Keterangan
&	AND
|	OR
^	XOR
~	Complement
>>	Shift Right
<<	Shift Left

```
int a = 13;                          // biner :  1101
int b = 6;                           // biner :  0110
cout<<"a & b ="<<(a&b);     // biner :  0100  =  4
```

Bentuk umum dari operator shift :
Shift Right (Geser Kanan) :
variabel >> nomor posisi bit

Shift Left (Geser Kiri):
variable << nomor posisi bit

```
int a = 13;                                  // biner :   1101
cout<<"a << 1 ="<<(a<<1);          // biner :  11010  =  26
```

Sequence (input → proses → output)

- Buat program mencari luas persegi Panjang
  - Luas = Panjang x lebar

Pemilihan / Seleksi

Analisa Kasus Tunggal (If)

- If digunakan untuk melakukan proses penyeleksian.
- Jika nilai if yang diseleksi bernilai **true** maka semua statement yang ada di dalam blok if akan dijalankan.

```
main() {
        int i = 1;
        if   (i==1){
              cout<<"i = 1”<<endl;
        }
}
```

Jika dijalankan maka akan didapatkan output "i = 1". Karena seleksi pada kurung if() menghasilkan nilai true (!= 0), yaitu nilai i adalah benar satu(1).

Analisa 2 kasus komplementer (If Else)

- Pernyataan Else digunakan sebagai alternatif apabila proses seleksi **if** menghasilkan nilai false (==0). Maka semua block yang ada pada statement **else** akan dijalankan

```
main() {
      int i = 2;
      if(i==1){
         cout<<"i = 1”<<endl;
      }
      else{
         cout<<"i is not 1”<<endl;
      }
}
```

- Dapat dilihat bahwa di dalam seleksi **if(i==1)** menghasilkan nilai false karena nilai i adalah 2. Maka blok yang dijalankan adalah blok else, yaitu mencetak **i is not 1**

Analisa Banyak Kasus (If-Else IF)

- Alternatif else if digunakan sebagai pilihan jika proses seleksi ada banyak. Misalkan kita harus menyeleksi variable untuk beberapa pilihan

```
main() {
    int i = 0;
    if(i==0){
        cout<<"i = 0”<<endl;
    }else if(i==1){
        cout<<"i = 1”<<endl;
    }else if(i==2){
        cout<<"i = 2”<<endl;
    }else{
        cout<<"i is not 0, 1, or 2”<<endl;
    }
}
```

- Jalankan program dan akan dicetak **i = 0**. Karena seleksi pertama yang memenuhi persyaratan. Yaitu **if(i==0)**. Seleksi yang lain yang tidak memenuhi persyaratan tidak akan dijalankan

Analisa Banyak Kasus (switch)

- Switch mengkonstruksikan cabang untuk beberapa kondisi dari nilai

```
main() {
    int gol = 2;
    switch (gol) {
       case 1  :  cout<<"Gaji = 100");
                  break;
       case 2  :  cout<<"Gaji = 200 ");
                  break;
       case 3  :  cout<<"Gaji = 300");
                          break;
       default :  cout<<"Golongan Salah");
                          break;
    }
}
```

PENGULANGAN / ITERASI

Pengulangan dengan Statement for

Syntaxnya adalah sbb :

```
for (inisialisasi ; kondisi kontinu; update)  {
  aksi ;
}
```

Ini merupakan bentuk yang sederhana untuk pengontrolan sebuah loop yang terdiri atas tiga bagian terpisah. Ketiganya ini dimungkinkan untuk kosong.

contoh:

```
// Menghitung Sigma(i) = 1 + 2 + 3 + ... +  n  dengan for
main() {
     int  n, sigma = 0;
     cout << "Masukkan bilangan integer positif"; cin >> n;
     for (int i = 1; i <= n; i++) {
             sigma += i;
     }
     cout << "Jumlahnya dari << n <<" bil. pertama adalah : " << sigma << endl;
}
```

Pengulangan dengan Statement while

syntaxnya adalah sbb :

```
while (kondisi) {
  aksi;
}
```

Pertama sekali kondisi diperiksa dan jika kondisi tidak nol (contoh true) maka statement akan dieksekusi kemudian kondisi dievaluasi lagi sampai kondisi adalah nol (contoh false).

contoh :

```
// Program menghitung Sigma(i) = 1 + 2 + 3 + ... +  n
// dengan while
main() {
      int i=1, n, sigma = 0;
      cout << "Masukkan bilangan integer positif"; cin >> n;
       while (i<=n) {
          sigma +=i;
          i++;
       }
      cout << "Jumlah dari << n <<" bil. pertama adalah : " << sigma <<
     endl;
}
```

Pengulangan dengan Statement do ... while

Syntaxnya adalah sbb:

do {
aksi
} while (kondisi);

Statement ini akan mengerjakan statement minimal satu kali (pertama yang dilakukan) dan kemudian baru dievaluasi kondisinya. Akan terus diulang selama kondisi bukan nol. Berhenti sampai kondisi bernilai nol (contoh false).

contoh :

```
main() {
    int n, f = 1;
    cout << "Masukkan bil integer positif : "; cin >> n;
    cout << n << " faktorial adalah : ";
    do {
        f *= n;
        n--;
    } while (n > 1);
    cout << f << endl;
}
```

Latihan dan Tugas

Apa output program berikut jika nilai n=5 dan nilai n=6 (soal a,b). Perlihatkan supaya lebih jelas.

a. if (n>2) { if (n<6) cout << "Jelek" ; } else cout << "OK";

b. if (n>2) { if (n<6) cout << "Jelek" ; else cout << "OK";

c.
```
int i=0;
while (i < 7) {
      if (i <2) {
            i += 2;
       }
       else  cout <<  (++i)  << endl;
       cout << "Bottom of loop"<<endl ;
}
```

2. Buat program untuk menampilkan output sbb (dicetak angka dan asterik):

Banyak baris : 4 (diinputkan dari keyboard)

1. *
2. * *
3. * * *
4. * * * *

1. * * * *
2. * * *
3. * *
4. *

Gunakan skema nested loop dengan

a. 2 buah for
b. 2 buah while