DATA STRUKTUR : INTRODUCTION

STRUKTUR DATA : STRUKTUR dan DATA

Struktur dapat diartikan sebagai suatu susunan, bentuk, pola atau bangunan.

Data dapat diartikan sebagai suatu fakta, segala sesuatu yang dapat dikodekan atau disimbolkan dengan kode-kode atau lambang-lambang yang telah disediakan di setiap komputer. Data yang disediakan oleh komputer sendiri terdiri dari berbagai jenis atau TYPE.

Pada garis besarnya, data dapat dikategorikan menjadi :

Type Data terdiri dari :

- Data Tunggal : **Integer, Real, Boolean dan Karakter.**
- Data Majemuk : **String**

Definisi :

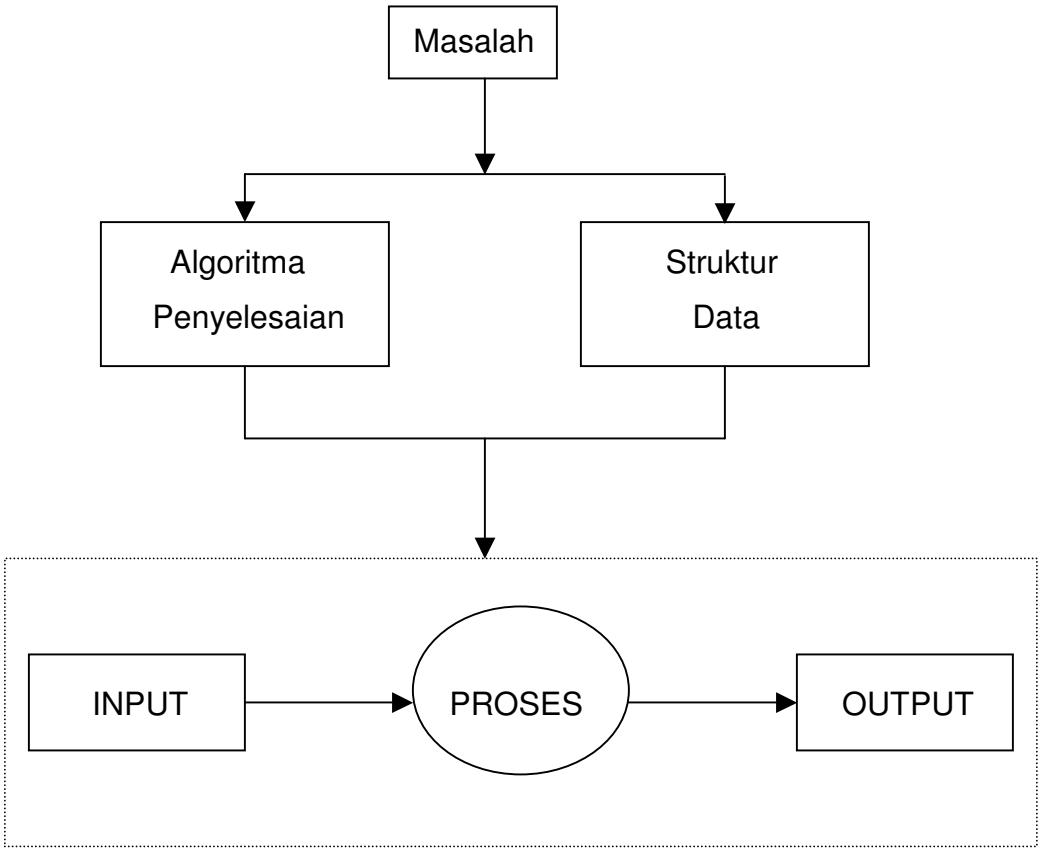
Struktur Data adalah suatu koleksi atau kelompok data (susunan simbol-simbol) yang dapat dikarakterisasikan oleh organisasi serta dapat dioperasikan sesuai dengan definisi yang diberikan terhadapnya dikomputer. Struktur Data adalah cara penyimpanan dan pengorganisasian data-data pada memori komputer maupun file pada media penyimpanan secara efektif sehingga dapat digunakan secara efisien, termasuk operasi-operasi di dalamnya.

Struktur Data, meliputi:

- Struktur data sederhana : **Array dan Record**
- Struktur data majemuk :
  - Linier : **Stack, Queue, Linier Link List**
  - Nonlinier : **Tree, Binari Tree, Binary Search Tree, Graph.**

Mempelajari struktur data berarti mempelajari bagaimana data disusun/terstruktur di memori utama komputer seara logis agar penggunaan *space* di memori dapat dilakukan secara optimal, cepat dalam pencarian dan pengambilannya kembali, dan dapat diolah/dioperasikan sesuai dengan tujuannya.
Selain itu hal yang terpenting dalam mempelajari struktur data adalah erat kaitannya dengan pemilihan struktur data yang tepat membuat suatu algoritma yang digunakan untuk memecahkan suatu masalahmenjadi efisien, yang akan membantu logika kita dalam membuat program yang rumit, sehingga operasi-operasi penting dapat dieksekusi dengan sumber daya yang lebih kecil, memori lebih kecil, dan waktu eksekusi yang lebih cepat dan outputnya sesuai dengan yang diharapkan.

Gambar di bawah ini menjelaskan posisi algoritma dan struktur data dalam membuat program.



Cara memecahkan masalah (membuat algoritma penyelesaiannya) dapat dilakukan dengan berbagai cara, meskipun hasil akhirnya akan sama. Untuk mengecek mana algoritma yang terbaik dan benar, maka dilakukan uji kompleksitas suatu algoritma.

Pemakaian struktur data yang tepat didalam proses pemrograman, akan menghasilkan algoritma yang lebih jelas dan tepat, sehingga menjadikan program secara keseluruhan lebih sederhana.

* Algoritma adalah suatu strategi yang mengandalkan kemampuan berpikir secara logis untuk memecahkan suatu masalah.

TIPE DATA

Disetiap bahasa pemrograman, disediakan berbagai jenis tipe data. Penentuan tipe data yang tepat (sesuai dengan karakterisitik data yang akan diolah) akan menjadikan sebuah program dapat dieksekusi secara efektif.

Jenis-Jenis Tipe Data :

1. Integer

Interger adalah data numerik yang tidak mengandung pecahan, dan disajikan dalam memori komputer sebagai angka bulat. Mengacu pada obyek data dengan range -32768 s/d 32767.

Operasi yang dapat dilaksanakan :

- Penambahan (+)
- Pengurangan (—)
- Perkalian (*)
- Pembagian Integer (/)
- Pemangkatan (^)

Operasi tersebut diatas disebut dengan opersi **Binar** atau arimatic operator yaitu operasi yang bekerja terhadap 2 Integer (operand). Sedangkan operator yang mempunyai satu operand disebut **Unar** (**Negasi** = Not).
Selain itu ada juga operasi tambahan yang disediakan oleh bahasa pemrograman tertentu, yaitu :

- MOD : sisa hasil pembagian bilangan
- DIV : hasil pembagi bilangan
- ABS : Mempositifkan bilangan negatif
- SQR : menghitung nilai akar dari bilangan

Penulisan di dalam bahasa pemrograman Pascal : *var a : integer*

2. Real

Data numerik yang mengandung pecahan digolongkan dalam jenis data Real (floating point). Operasi yang berlaku pada bilangan integer juga berlaku pada bilangan real. Selain itu ada operasi lainnya seperti :

INT : membulatkan bilangan real , misal INT(34.67) = 34

3. Boolean

Type ini dikenal pula sebagai " Logical Data Types", digunakan untuk melakukan pengecekan suatu kondisi dalam suatu program.
Elemen datanya hanya ada 2 yaitu **True** dan **False**, biasanya dinyatakan pula sebagai **1** dan **0**.
Operatornya terdiri dari : AND, OR, NOT
Binar Unar
Dalam urutan operasi, Not mendapat prioritas pertama, kemudian baru AND dan OR kecuali bila diberi tanda kurung.

Masih ingatkah anda dengan table logika ?

Nilai true dan false dapat juga dihasilkan oleh operator Relational.

Operator tersebut : < , > , <= , >= , = , <> , =

Ex. 6 < 12 : True ,

A <>A : False.

Contoh lainnya:

PROGRAM	OUTPUT

a = T
b = F
c = F
IF a AND NOT b
hasil1 = a
ELSE
hasil1 = b
ENDIF
	hasil1 = TRUE (a)

IF a AND b OR c
hasil2 = a
ELSE
hasil2 = b
ENDIF
	hasil2 = FALSE (b)

IF a OR (b OR c)
hasil3 = a
ELSE
hasil3 = b
ENDIF
	hasil3 = TRUE (a)

4. Karakter dan String

Type karater mempunyai elemen sebagai berikut :
(0,1,2,3,…,9,A,B,C,…,X,Y,Z,?,*,/,…)

Data type majemuk yang dibentuk dari karakter disebut STRING.
Suatu string adalah barisan hingga simbol yang diambil dari himpunan karakter yang digunakan untuk membentuk string dinamakan Alfabet.

Contoh : Himpunan string {A,A,1} dapat berisi antara lain :

(AB1), (A1B), (1AB),…dst.

Termasuk string Null (empty / hampa / kosong) = { }

Secara umum suatu string **S** dinyatakan :

S : a1, a2, a3,… an,

Panjang dari string dilambangkan | S | =N atau Length (S) = N dimana N adalah banyaknya karakter pembentuk string.
Untuk string Null | | = 0, untuk blank (spasi)=1.

Operasi yang berlaku terhadap string :

a. LENGTH(S) berfungsi untuk menghitung panjang suatu string.

Contoh : S1 adalah string = a1,a2,a3…an

S2 adalah string = b1, b2, b3,…bk

Len(s2) = n , Len(s1) = k

b. CONCAT (S1 , S2)

yaitu concatenation (Penyambungan 2 buah string atau lebih.
Penggabungan juga dapat dilakukan terhadap dirinya sendiri.
Contoh : concat(s1,s2) = a1, . . . , an , b1 , . . . , bk
atau dalam bentuk lain : s1 // s2 , s1 + s2

c. SUBSTR (s, i, j) yaitu operasi pengambilan beberapa karakter dari string untuk membentuk string baru.

s : adalah string

i : adalah posisi karakter awal yang diambil

j : adalah banyaknya karakter yang diambil

dimana **i** dan **j** ber-type Integer

S1(a1,a2,a3,…an)

SUBSTR (s, 3, 2) = a3, a4 ⟶ S1(a3, a4)

Selain dari itu terdapat juga operasi pemenggalan lainnya yaitu :

RIGHT(S1, j) dan LEFT(S1, j)

d. INSERT (S1, S2, j)

Operasi ini membutuhkan dua operand string dan sebuah operand Integer.

Contoh : Insert (S1, S2, 3) = a1, a2, b1, b2,…, bk, a3,…, an

Menyisipkan S2 didalam S1 mulai posisi ke 3 dari S1.

Bila tidak ada statemen INSERT dalam bahasa pemrograman maka dapat dilakukan dengan cara lain, misal :

LEFT(S1, j) // S2 // RIGHT(S1, j)

e. DELETE (S, i, j)

Operasi ini membutuhkan sebuah string dan dua operand integer.

Contoh : S1 : a1, a2, a3,…, an.

DELETE (S1 , 4, 3) = a1, a2, a3, a7, a8,…,an.

Menghapus string pada posisi awal 4, sebanyak 3.

Bila tidak ada statemen DELETE dalam bahasa pemrograman maka dapat dilakukan dengan cara lain, misal :

LEFT(S1, j) // RIGHT(S1, j)

f. INDEX(S1,'substring')

Mencari posisi awal (karakter ke berapa) suatu substring pada suatu string.

Contoh : INDEX(S1,'a3,a4,5') = 3

Dalam bahasa pemrograman untuk membedakan sebuah string atau integer menggunakan tanda kutip. Integer : 34 string : '34'.

Pemetaan (MAPPING) Type Data ke Storage

Komputer merepresentasikan data dalam bentuk biner, karena setiap bit data dalam komputer hanya dapat menyimpan dua macam keadaan, yaitu voltase tinggi dan voltase rendah. Perbedaan voltase tersebut mewakili nilai TRUE dan FALSE, atau bit '1' dan '0'

Representasi Karakter dan String

Ada beberapa aturan yang digunakan untuk menyatakan karakter dalam storage. Diantaranya adalah :

1. EBCDIC (Extended Binary Coded Decimal Interchange Code)
   EBCDIC adalah suatu sistem peng-kode-an (mapping) yang menggunakan 8 binary digit (bit) untuk menyatakan suatu karakter dalam alfabet.
   (1 karakter = 8 bit)
   Dalam 8 bit terdapat 2^8 **(**256) kemungkinan karakter yang dapat dibentuk.

2. ASCII (American Standard Code For Information Interchange)
   ASCII adalah cara peng-kode-an yang menggunakan 7 bit untuk menyatakan suatu karakter dalam alfabet.
   (1 karakter = 7 bit). Dalam 7 bit terdapat 2^7 (128) kemungkinan karakter yang dapat dibentuk, separuh dari yang dimiliki EBCDIC.

3. BCD (Binary Coded Decimal)
   BCD ini menggunakan 4 bit untuk setiap karakternya.

4. PACKED DECIMAL
   Packed Decimal umumnya digunakan untuk karakter berjenis data numerik dengan cara penyimpanannya menggunakan 2 digit setiap 8 bit. Pada 8 bit

terakhir disimpan selain digit derajat terendah, juga tanda dari bilangan tersebut (positif atau negatif).

Berikut ini perbandingan kode EBCDIC, ASCII dan PACKED-DECIMAL untuk menyatakan +903.

	9	0	3	+
EBCDIC	: 11111001	11110000	11110011	01001110
ASCII	: 0111001	0110000	0110011	0101011
PACKED DECIMAL	: 10010000	00111100		

5. Unicode

Unicode menggunakan 16 bit untuk merepresentasikan karakter. Dengan demikian, banyaknya karakter yang dapat direpresentasikan adalah 216 atau 65.536 karakter.
Keunggulan Unicode dari ASCII adalah kemampuannya untuk menyimpan simbol / karakter yang jauh lebih besar. Himpunan 256 karakter pertama dari Unicode merupakan pemetaan karakter ASCII 8 bit, sehingga Unicode tetap kompatibel dengan ASCII. Selain merepresentasikan seluruh karakter ASCII, Unicode dapat merepresentasikan juga berbagai macam simbol diluar ASCII, seperti huruf Arab, Kanji, Hiragana, Katakana, dan lain-lain.

Representasi Bilangan Bulat / Integer

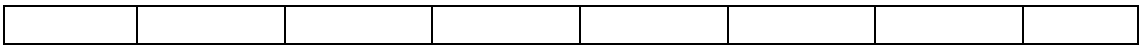
Bilangan Bulat *Tak Bertanda* dapat direpresentasikan dengan

- bilangan biner – oktal - heksadesimal
- gray code
- BCD (*binary coded decimal*)

Bilangan bulat ***Bertanda*** (positif atau negatif) dapat direpresentasikan dengan

- Sign/Magnitude (S/M)
- 1's complement
- 2's complement

Untuk bilangan bulat positif, tidak ada perbedaan dalam ketiga macam representasi bilangan di atas. Terdapat persamaan dalam ketiga representasi tersebut berupa digunakannya MSB (*most significant bit*) sebagai penanda. MSB bernilai '0' untuk bilangan positif dan '1' untuk bilangan negatif



7 6 5 4 3 2 1 0
MSB LSB

SIGN / MAGNITUDE

Salah satu storage mapping yang dapat dilakukan terhadap integer adalah apa yang disebut bentuk *sign-and-magnitude*, yaitu digit untuk tanda integer positif atau negatif dan sebarisan digit untuk menyatakan magnitude/besarnya.
Contoh : -7 = -111 dan +7 = +111

Bagi kita mudah bekerja terhadap bilangan dalam bentuk sign-and-magnitude, namun apabila dilakukan penjumlahan dengan kedua operand berbeda tanda, penjumlahan akan beralih menjadi pengurangan yang kadang-kadang menimbulkan kesukaran. Untuk itu, digunakan apa yang disebut sebagai

COMPLEMENT (merubah tanda negatif pada bilangan pengurangan menjadi tanda positif)

X' adalah complement dari X terhadap R (R 's complement dari X) bila
X + X' = R.
X' = R – X menyatakan integer negatif -X.

Representasi negatif dari suatu bilangan diperoleh dari bentuk positifnya dengan mengubah bit pada MSB menjadi bernilai 1. Jika dipergunakan N bit untuk representasi data, maka rentang nilai yang dapat direpresentasikan adalah

$-2^{N-1}-1$ s.d $2^{N-1}-1$

Contoh : jika dipergunakan 5 bit untuk representasi bilangan
+3 = 00011
-3 = 10011

Terdapat dua jenis Complement :

ONE'S COMPLEMENT

Representasi negatif dari suatu bilangan diperoleh dengan mengkomplemenkan seluruh bit dari nilai positifnya. Jika dipergunakan N bit untuk representasi data, maka rentang nilai yang dapat direpresentasikan adalah $-2^{N-1}-1$ s.d $2^{N-1}-1$

1's complement menggunakan mapping dengan **$R = 2^N - 1$**
N adalah jumlah bit integer yang dapat disajikan.

Contoh : jika dipergunakan 5 bit untuk representasi bilangan
+3 = 00011
-3 = 11100

TWO'S COMPLEMENT

Representasi negatif dari suatu bilangan diperoleh dengan mengurangkan 2n dengan nilai positifnya. Jika dipergunakan N bit untuk representasi data, maka rentang nilai yang dapat direpresentasikan adalah -2^{N-1} s.d $2^{N-1}-1$

Two's Complement menggunakan mapping dengan **$R = 2^N$**

Contoh : jika dipergunakan 5 bit untuk representasi bilangan

$2^n = 2^5 = 100000$

+3 = 00011

- 3 = 100000−00011

```
100000
 00011 −
 11101
```

→ − 3 = 11101

PERBANDINGAN

Berikut tabel perbandingan ketiga cara representasi bilangan bulat bertanda.

B	Nilai yang direpresentasikan		
$b_3b_2b_1b_0$	Sign/Magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	−0	−7	−8
1001	−1	−6	−7
1010	−2	−5	−6
1011	−3	−4	−5
1100	−4	−3	−4
1101	−5	−2	−3
1110	−6	−1	−2
1111	−7	−0	−1

Representasi Bilangan Pecahan / Floating Point

Bilangan pecahan dapat direpresentasikan dalam bentuk pecahan biasa atau dalam bentuk *scientific*.

- Bentuk Pecahan Biasa

Dalam bentuk pecahan biasa, bilangan direpresentasikan langsung kedalam bentuk binernya. Contoh : $27.625 = 11011.101_2$

- Bentuk S C I E N T I F I C

Dalam notasi scientific, bilangan pecahan dinyatakan sebagai $X = \pm M \cdot B^{\pm E}$.

M = mantissa
B = basis
E = eksponen

Contoh : $5.700.000 = 57.10^5 \longrightarrow$ M=57, B=10, E=5

Masalah : terdapat tak berhingga banyaknya representasi yang dapat dibuat. Dalam contoh sebelumnya, $5.700.000 = 57.10^5 = 570.10^4 = 5{,}7.10^6 = 0{,}57.10^7 = 0{,}057.10^8$ dst. Untuk mengatasinya, ditentukan adanya bentuk normal, dengan syarat

$1/B = |M| < 1$

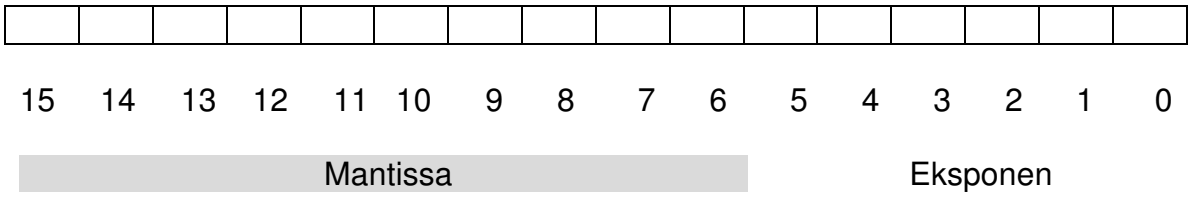
Dengan demikian, bentuk scientific yang normal (memenuhi persyaratan) dari 5.700.000 adalah $0{,}57.10^7$

Dalam bentuk normal tersebut, selalu diperoleh mantissa berbentuk '0,…' sehingga dalam representasinya kedalam bit data, fraksi '0,' tersebut dapat dihilangkan.

Mantissa dan eksponen tersebut dapat direpresentasikan menggunakan salah satu cara representasi bilangan bulat bertanda yang telah dibahas di atas. Representasi yang dipilih dapat saja berbeda antara mantissa dengan eksponennya.

Contoh :
- Digunakan untaian 16 bit untuk representasi bilangan pecahan
- 10 bit pertama digunakan untuk menyimpan mantissa dalam bentuk S/M
- 6 bit sisanya digunakan untuk menyimpan mantissa dalam bentuk 1's complement

Contoh akan direpresentasikan bilangan 0,00000075



0,00000075 = 0,75 . 10-6 → M = 0,75; E = -6

Representasi Mantissa :

0,75 = $0,11_2$. Karena sudah dalam bentuk normal '0,' dapat dihilangkan.
S/M → MSB sebagai penanda. Dengan demikian, mantissa = 0110000000

Representasi Eksponen : 6=1102 . Karena digunakan 6 bit, 1102 = 000110.
1's complement → -6 = 111001

Representasi :