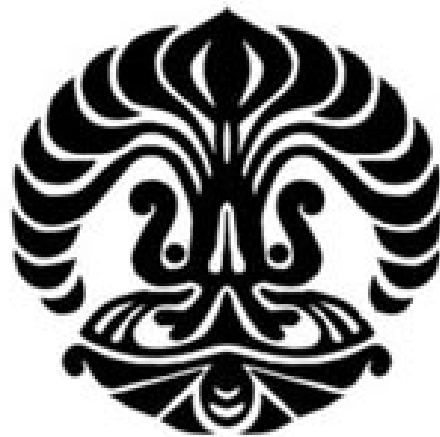


**TUGAS SUPPLEMEN
PENGANTAR KONSEP PEMROGRAMAN DAN
STRUKTUR DATA**

DISUSUN OLEH :

JENNY SARI TARIGAN (NPM : 7205000911)

YULIE ASTANTO (NPM:7205001039)



**MAGISTER TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA
2005**

Lisensi Dokumen:

Copyright © 2005 by Jenny Sari Tarigan dan Yulie Astanto

Seluruh dokumen ini dapat digunakan, dimodifikasi dan disebarakan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen.

Bab 3. DATA REPRESENTATION

Data Representation and Processing

Automated Data Processing

Binary Data Representation

Hexadecimal Notation

Octal Notation

Goals of Computer Data Representation

Compactness

Accuracy

Ease of Manipulation

Standardization

CPU Data Types

Integers

Real Numbers

Character Data

Beberapa aplikasi menggunakan data yang bukan hanya bilangan tetapi juga huruf dari alfabet dan karakter khusus lainnya. Data semacam ini disebut dengan data alfanumerik dan mungkin dapat ditunjukkan dengan kode numerik. Jika bilangan-bilangan dimasukkan dalam data, maka bilangan-bilangan tersebut juga dapat ditunjukkan dengan kode khusus.

Set karakter alfanumerik secara khusus mencakup 26 huruf alfabet (termasuk huruf besar dan huruf kecil), angka dalam digit sepuluh desimal, dan sejumlah simbol seperti +, =, *, \$, ..., dan !. Dua kode alfabet yang paling umum dipakai adalah ASCII (American Standard Code for Information Interchange) dan EBCDIC (Extended Binary Coded Decimal Interchange Code). ASCII merupakan kode 7-bit dan EBCDIC berupa kode 8-bit. Jika suatu komputer menangani 8-bit (1-byte) kode lebih efisien, versi 8-bit, disebut dengan ASCII-8 juga telah dikembangkan.

Selain itu ada juga beberapa kode spesial didalam penambahan set karakter alfanumerik. Kode simpanan ini digunakan sebagai signal komunikasi dalam aplikasi dimana data transfer terjadi antara komputer yang dihubungkan melalui baris komunikasi. Misalnya,

LF (line feed) dan CR (carriage return) dihubungkan dengan printer, BEL digunakan untuk mengaktifitaskan bell; ACK (acknowledge), NAK (negative acknowledge), dan DLE (data link escape) berupa signal yang dapat diubah dalam baris komunikasi.

Bagi yang sudah cukup lama berkecimpung di dunia komputer, pasti pernah bekerja dengan 'kode ASCII'. Dan bagi yang bekerja dengan mesin-mesin mainframe IBM, pasti pernah menjumpai 'kode EBCDIC' (dibaca: *eb-si-dik*). Di luar ASCII dan EBCDIC, besar kemungkinan anda paling tidak pernah mendengar istilah-istilah lain seperti berikut ini: ISO-8859-1, UCS-2, UTF-8, UTF-16, atau windows-1252. Kode-kode apakah itu? ASCII, EBCDIC, ISO-8859-*x*, UCS-2, UTF-*x*, dan windows-*x* merupakan sebagian dari kumpulan *character set* (set karakter) yang ada di dunia komputer.

Sistim Binary Coded Decimal (BCD):

Sebelum ASCII dan EBCDIC berkembang terlebih dahulu dikembangkan Binary Coded Decimal (BCD). Metode ini awalnya digunakan pada komputer mainframe IBM. Pada grup ini karakter diwakili oleh $64 - (2^6)$ lambang. Dengan kode ini, setiap huruf/angka diberikan kode yang terdiri dari enam bit, dua untuk zone dan empat untuk angka. Huruf A sampai dengan I diberikan tanda 11 pada tempat zone. Karena A adalah huruf pertama dalam kelompok ini, maka kodenya adalah: 0001, B sebagai huruf kedua dengan kode: 0010, C adalah 0011 dan seterusnya. Dengan perkataan lain, zone bit yang mempunyai formasi 11 harus juga disertakan pada kode lengkap masing-masing pada grup ini. Grup alfabetik kedua adalah J hingga R, ditetapkan kode awalnya 10, yang juga posisi masing-masing huruf ditentukan oleh angkanya masing-masing. Huruf S hingga Z dibentuk dengan menambahkan angka bit 0010 hingga 1001 berurutan pada kode 01 dimana pada grup ini hanya ada delapan huruf. Angka-angka 0 hingga sembilan diberikan kode 00 di depannya diikuti oleh angka itu sendiri dalam sistim binary. Angka 0 (nol) harus dibedakan dengan tanda kosong (spasi) guna mempermudah cara penggunaan kode.

Sistim Extended Binary Coded Decimal Interchange Code (EBCDIC):

EBCDIC merupakan set karakter yang merupakan ciptaan dari IBM. Salah satu penyebab IBM menggunakan set karakter di luar ASCII sebagai standar pada komputer ciptaan IBM adalah karena EBCDIC lebih mudah dikodekan pada *punch card* yang pada tahun 1960-an masih jamak digunakan. Penggunaan EBCDIC pada mainframe IBM masih terbawa hingga saat ini, walaupun *punch card* sudah tidak digunakan lagi. Seperti halnya ASCII, EBCDIC juga terdiri dari 128 karakter yang masing-masing berukuran 7-bit. Bila menggunakan ukuran 8-bit maka karakternya menjadi $256 - (2^8)$. Hampir semua karakter pada ASCII juga terdapat pada set karakter EBCDIC.

Sistim American Standard Code for Information Interchange (ASCII):

ASCII dan EBCDIC merupakan cikal bakal dari set karakter lainnya. ASCII merupakan set karakter yang paling umum digunakan hingga sekarang. Set karakter ASCII terdiri dari $128 - (2^7)$ buah karakter yang masing-masing memiliki lebar 7-bit atau gabungan tujuh angka 0 dan 1, dari 0000000 sampai dengan 1111111. Mengapa 7-bit? Karena komputer pada awalnya memiliki ukuran memori yang sangat terbatas, dan 128 karakter dianggap memadai untuk menampung semua huruf Latin dengan tanda bacanya, dan beberapa karakter kontrol. ASCII telah dibakukan oleh ANSI (American National Standards Institute) menjadi standar ANSI X3.4-1986.

Adapun kode-kode pada sistim-sistim tersebut di atas dapat dilihat pada tabel di bawah ini:

Tabel Kode ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	:	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.asciitable.com

Tabel Sebagian dari ASCII, EBCDIC dan BCD

Symbol	ASCII	EBCDIC	BCD
0	0110000	11110000	000000
1	0110001	11110001	000001
2	0110010	11110010	000010
3	0110011	11110011	000011
4	0110100	11110100	000100
5	0110101	11110101	000101
6	0110110	11110110	000110
7	0110111	11110111	000111
8	0111000	11111000	001000
9	0111001	11111001	001001
A	1000001	11000001	010001
B	1000010	11000010	010010

Unicode

Orang-orang di negara-negara yang berbeda menggunakan karakter berbeda untuk menuliskan kata-kata dalam bahasa pribumi mereka. Sekarang ini kebanyakan aplikasi, mencakup sistim email dan web browsers, menggunakan sistim 8-bit yaitu mereka dapat beroperasi dengan teks yang tepat sesuai dengan ketentuan, seperti ISO-8859-1. Kalau ASCII dan EBCDIC sudah mampu mengkodekan 128 karakter, lalu mengapa masih dianggap perlu untuk menciptakan set karakter baru? Lebar set karakter yang cuma 128 karakter tidak memungkinkan penulisan karakter-karakter di luar huruf Latin (*basic Latin*), seperti misalnya huruf *ü* atau simbol-simbol matematika dalam huruf Yunani.

Set karakter Unicode mampu menampung lebih dari satu juta karakter ($2^{20} = 1.048.576$). Akan tetapi, saat ini hanya 65.535 karakter yang pertama yang mampu direpresentasikan pada komputer (65.535 karakter pertama dari Unicode sering disebut dengan istilah UCS-2 atau Universal Character Set-2). Karakter 0 sampai dengan 65.535 menampung karakter-karakter dari alfabet-alfabet yang belum punah (Latin, Kanji, Devanagari, dan lain sebagainya) sedangkan karakter 65.536 sampai dengan 1.048.575 menampung karakter-karakter dari alfabet-alfabet yang sudah punah (misalnya *hieroglyph* dan beberapa karakter Cina yang sangat jarang digunakan). Terdapat lebih dari 256 karakter di dunia seperti kode cyrillic, hebrew, arabic, chinese, japanese, korean dan thailand, dan karakter baru kadang-kadang ditemukan. Dengan menggunakan kode 16-bit ($2^{16} = 65.535$) diharapkan karakter-karakter tersebut dapat terwakili.

Set karakter Unicode dialokasikan untuk lebih dari satu alfabet. Bahkan, Unicode Consortium menargetkan untuk mengkodekan seluruh alfabet yang ada di dunia. Set karakter Unicode ini diharapkan dapat menjadi standar set karakter pada semua komputer di masa depan. Karena Unicode mampu merepresentasikan semua alfabet yang ada di dunia ini, maka secara teori seluruh set karakter lainnya tidak diperlukan lagi. Unicode mampu mengkodekan berbagai karakteristik alfabet. Mulai dari alfabet Latin yang sederhana, alfabet Arab yang ditulis sambung-menyambung (*cursive*) dari kanan ke kiri, alfabet Cina yang ditulis dari atas ke bawah, dan alfabet India yang memiliki huruf vokal yang letaknya di atas-bawah-depan-belakang dari konsonan.

Oleh karena ke-superior-an dari Unicode, maka Unicode menjadi set karakter standar yang digunakan pada komunikasi data melalui Internet. Unicode memiliki lebar per

karakter sebesar 20 bit. Akan menjadi sangat boros jika kita mengirim data Unicode yang berisikan teks huruf Latin saja menggunakan 20-bit per karakter. Oleh karena itu, maka Unicode perlu ditransformasikan terlebih dahulu menjadi UTF-8 atau UTF-16 (Unicode Transformation Format). Dengan UTF-8, maka karakter-karakter pada U+0000 (Notasi U+abcd digunakan untuk mengacu pada karakter bernomor *abcd* pada tabel Unicode. Sebagai contoh, U+0053 adalah Latin Capital Letter S dan U+0584 adalah Armenian Small Letter KEH) sampai dengan U+007F (128 karakter pertama) dapat dikodekan menjadi satu byte saja; sedangkan karakter-karakter lainnya dikodekan dengan menggunakan antara 2 sampai 4 byte per karakter. UTF-8 dan UTF-16 digunakan hanya pada saat komunikasi data saja sedangkan Unicode dalam bentuk normal (20 bit) tetap digunakan pada representasi karakter di memori komputer.

Pada dasarnya ada empat cara untuk mengkode karakter Unicode yaitu:

UTF-8

128 karakter digunakan mengkode 1 byte (karakter ASCII). 1.920 karakter digunakan mengkode 2 byte (untuk karakter-karakter Roma, Yunani, Cyrillic, Coptic, Armenian, Ibrani, dan Arab). 63.488 karakter digunakan mengkode 3 byte (Cina dan Jepang). 2.47.418.112 karakter yanglainnya, yang tidak belum digunakan dapat digunakan mengkode 4, 5 atau 6 karakter.

UCS-2

Tiap-tiap karakter direpresentasikan oleh dua byte. Pengkodean ini digunakan untuk merepresentasikan 65.536 karakter Unicode yang pertama.

UTF-16

Ini adalah perluasan UCS-2 dimana dapat direpresentasikan 1.112.064 karakter Unicode. 65.536 karakter Unicode yang pertama diwakili dua byte, yang yang lainnya empat byte.

UCS-4

Tiap-tiap karakter direpresentasikan oleh empat byte.

Boolean Data

Konsep boolean ini ditemukan oleh seorang ahli matematika abad kesembilan belas yang bernama George Boole. Tipe data boolean digunakan untuk membuat statemen logika atau menguji suatu ungkapan yang akan menghasilkan nilai benar (*true*) atau salah (*false*). Variabel bertipe bool hanya mempunyai dua nilai yaitu *true* atau *false*, jadi anda hanya dapat mengisinya dengan nilai ini. Perlu diperhatikan bahwa variabel bertipe bool dianggap bernilai false jika nilainya 0 , dan benar jika nilainya ≥ 1 .

Operator adalah karakter atau kumpulan karakter yang digunakan untuk memanipulasi variabel. Karakter atau kumpulan karakter ini dikenali secara spesifik oleh compiler sehingga variabel yang diubah-ubah nilainya (*operand*) akan dikenai operasi sesuai dengan definisi operasi yang dimiliki oleh operator tadi.

Ada tiga golongan besar operator, yaitu:

§ **unary operator**, operator ini hanya bekerja pada satu operand. Unary operator antara lain:

Operator Unary	Kegunaan
- ~ !	Operator ini melakukan operasi negasi, yaitu

	merubah nilai operand menjadi nilai yang berlawanan tanda. Disebut juga Negation atau complement operator
++	Operator ini menaikkan nilai operand 1 satuan, disebut juga increment operator.
--	Operator ini menurunkan nilai operand 1 satuan, disebut juga decrement operator.
*	Operator ini memberikan nilai yang tersimpan pada alamat memory yang ditunjuk oleh operand, disebut juga indirection operator. Biasanya digunakan dalam operasi dengan pointer (akan dijelaskan lebih lanjut)
&	Operator ini memberikan alamat memory operand. Biasanya digunakan dalam operasi dengan pointer (akan dijelaskan lebih lanjut). Disebut juga address of operator.
sizeof (operand)	Operator ini memberikan ukuran memory yang digunakan operand dalam byte. operand adalah operand yang ukurannya akan dicari.

§ **binary operator**, operator ini bekerja pada dua operand sekaligus. Operator binary antara lain:

Operator Binary	Kegunaan
<i>Operator Matematis</i>	
+	Operator ini menjumlahkan nilai operand di sebelah kiri dengan nilai operand di sebelah kanan
-	Operator ini mengurangi nilai operand di sebelah kiri dengan nilai operand di sebelah kanan
*	Operator ini mengalikan nilai operand sebelah kiri dengan nilai operand di sebelah kanan
/	Operator ini membagi nilai operand sebelah kiri dengan nilai operand di sebelah kanan
%	Operator ini disebut juga modulo operator. Nilai Operand di sebelah kiri akan dibagi dengan nilai operand di sebelah kanan, kemudian hasil operasinya adalah sisa dari pembagian tersebut. Misalnya: 3%2 akan menghasilkan 1.
<i>Operator Logic (Logical Operator)</i>	
&&	Operator ini disebut operator logical and. Cara kerjanya adalah mengetes ekspresi disebelah kanan dan kirinya, jika keduanya benar maka nilai yang dihasilkan 1 (true), jika tidak maka

	nilai yang dihasilkan adalah 0 (false)
	Operator ini disebut operator logical or. Cara kerjanya adalah mengetes ekspresi disebelah kanan dan kirinya, jika salah satunya benar maka nilai yang dihasilkan 1, jika tidak maka nilai yang dihasilkan adalah 0
<i>Operator Logic (Logical Operator)</i>	
&&	Operator ini disebut operator logical and. Cara kerjanya adalah mengetes ekspresi disebelah kanan dan kirinya, jika keduanya benar maka nilai yang dihasilkan 1 (true), jika tidak maka nilai yang dihasilkan adalah 0 (false)
	Operator ini disebut operator logical or. Cara kerjanya adalah mengetes ekspresi disebelah kanan dan kirinya, jika salah satunya benar maka nilai yang dihasilkan 1, jika tidak maka nilai yang dihasilkan adalah 0
<i>Operator Operasi Bit (Bitwise Operator)</i>	
&	Operator disebut juga operator bitwise and. Cara kerjanya adalah melakukan operasi AND (&) pada setiap bit kedua operand. misalnya: 0xA & 0xC akan menghasilkan 0x8, perhatikan bahwa 0xA = 1010 _{biner} dan 0xC = 1100 _{biner} . Sehingga 1010 & 1100 = 1000 _{biner} atau 0x8 _{heksadesimal} sebab 0&1 = 0, 1&0 = 0, 0&0 = 0, 1&1 = 1.
	Operator ini disebut juga operator bitwise or. Cara kerjanya adalah melakukan operasi OR () pada setiap bit kedua operand. misalnya: 0xA 0xC akan menghasilkan 0xE, perhatikan bahwa 0xA = 1010 _{biner} dan 0xC = 1100 _{biner} . Sehingga 1010 1100 = 1110 _{biner} atau 0xE _{heksadesimal} sebab 0 1 = 1, 1 0 = 1, 0 0 = 0, 1 1 = 1.
^	Operator ini disebut juga operator bitwise xor. Cara kerjanya adalah melakukan operasi XOR (^) pada setiap bit kedua operand. misalnya: 0xA ^ 0xC akan menghasilkan 0x6, perhatikan bahwa 0xA = 1010 _{biner} dan 0xC = 1100 _{biner} . Sehingga 1010 ^ 1100 = 0110 _{biner} atau 0x6 _{heksadesimal} sebab 0^1 = 1, 1^0 = 1, 0^0 = 0, 1^1 = 0.

§ **ternary operator**, operator ini bekerja pada tiga operand sekaligus.

Operator	Kegunaan
Ekspresi1 ? Ekspresi2 : Ekspresi3	Cara kerja dari operator ini adalah mengecek nilai ekspresi1, jika nilainya benar, maka

	<p>yang akan dievaluasi adalah ekspresi2, jika salah maka yang akan dievaluasi adalah ekspresi3. Berikut ini contoh untuk memperjelas:</p> <pre>int z; int x = 2; int y = 3; z = (x > y) ? x : y ;</pre> <p>pada source code di atas, nilai z akan sama dengan y, sebab ekspresi1 nilainya salah, sehingga yang akan dievaluasi adalah ekspresi3.</p>
--	--

Beberapa operator yang sering digunakan untuk membandingkan/menguji suatu kondisi adalah:

Tabel Operator logika

Operator Logika	Nama Operasi	Deskripsi
!	NOT	Negasi
&	AND	Konjungsi
	OR	Disjungsi
^	Exclusive OR	Eksklusif

Tabel Kebenaran NOT (!)

Operand	!Operand
true	false
false	true

Tabel Kebenaran Conditional AND (&&) dan Unconditional AND (&)

Operand1	Operand2	Operand1 && Operand2
false	false	false
false	true	false
true	false	false
true	true	true

Tabel Kebenaran Conditional OR (||) dan Unconditional OR (|)

Operand1	Operand2	Operand1 && Operand2
false	false	false
false	true	true
true	false	true
true	true	true

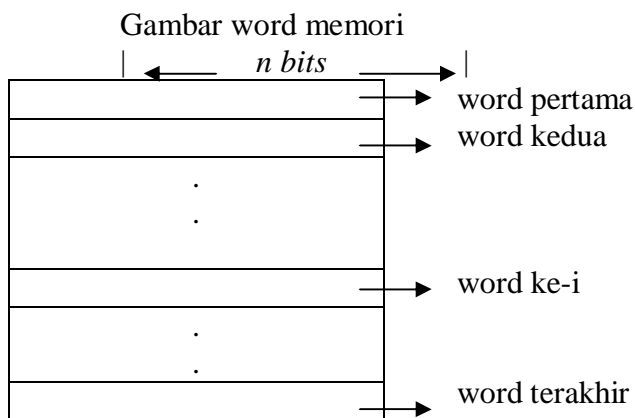
Tabel Kebenaran Exclusive OR (^)

Operand1	Operand2	Operand1 && Operand2
false	false	false
false	true	true

true	false	true
true	true	false

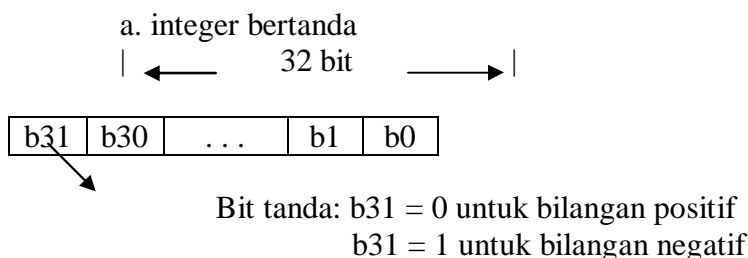
Memory Addresses

Operand bilangan dan karakter, seperti halnya instruksi, disimpan dalam memori komputer. Memori terdiri atas jutaan sel penyimpanan, dimana tiap sel tersebut menyimpan suatu bit informasi yang berupa nilai 0 atau 1. Karena bit tunggal mewakili informasi yang sangat sedikit, maka bit jarang ditangani secara individu. Pendekatan yang umum adalah menanganinya dalam kelompok dengan ukuran tertentu. Untuk tujuan ini, memori tersebut diatur sehingga kelompok n bit dapat disimpan dan diambil dalam satu operasi dasar tunggal. Tiap kelompok n bit disebut sebagai word informasi, dan n disebut word length. Memori suatu computer dapat direpresentasikan secara skematis sebagai kumpulan word sebagaimana ditunjukkan dalam gambar berikut:

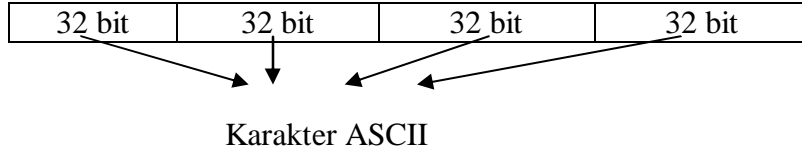


Komputer modern memiliki *word length* yang biasanya berkisar dari 16 hingga 64 bit. Suatu unit 8 bit disebut byte. Instruksi mesin mungkin memerlukan satu atau lebih word untuk representasinya. Jika *word length* suatu komputer adalah 32 bit, maka word tunggal dapat menyimpan 32-bit bilangan *2's-complement* atau empat karakter ASCII, masing-masing memiliki 8 bit, sebagaimana ditunjukkan gambar berikut:

Gambar contoh informasi dalam 32-bit word



b. empat karakter



Mengakses memori untuk menyimpan atau mengambil suatu item informasi, baik berupa word atau byte, memerlukan nama yang berbeda atau alamat untuk tiap lokasi item. Merupakan hal yang biasa menggunakan bilangan 0 hingga $2^k - 1$, untuk beberapa nilai k yang sesuai, sebagai alamat dari lokasi yang berurutan dalam memori. Alamat 2^k meliputi ruang alamat komputer tersebut, dan memori tersebut dapat memiliki lokasi *addressable* hingga 2^k . Misalnya, alamat 24-bit menghasilkan ruang alamat 2^{24} (16.777.216) lokasi. Bilangan ini biasanya ditulis 16M (16 mega), dimana 1M adalah bilangan 2^{20} (1.048.576). Alamat 32-bit menghasilkan ruang alamat 2^{32} atau 4G (4 giga) lokasi, dimana 1G adalah 2^{30} . Konvensi yang biasanya digunakan adalah K (kilo) untuk bilangan 2^{10} (1.024), dan T (tera) untuk bilangan 2^{40} .

Data Structures

CPU secara langsung dapat memanipulasi data integer, real number, karakter, boolean dan memory adres. Jenis data yang secara langsung didukung oleh CPU disebut jenis data primitif, atau jenis data mesin. Program akan sulit dikembangkan bila hanya jenis data primitif yang tersedia. Kebanyakan program aplikasi harus menggabungkan item data primitif untuk memperoleh hasil yang bermanfaat. Contoh yang umum adalah satu karakter atau string. Umumnya program aplikasi menggunakan karakter string, tetapi sebagian CPU menyediakan instruksi yang langsung memanipulasinya. Pengembangan aplikasi sederhana jika karakter string dapat digambarkan dan dimanipulasi (seperti, membaca, menulis, dan membandingkan) sebagai unit tunggal disamping sebagai satu karakter pada satu saat.

Struktur data termasuk kelompok elemen data primitif yang diorganisir untuk beberapa bentuk pengolahan umum. Struktur data digambarkan dan dimanipulasi dalam perangkat lunak. Perangkat keras komputer tidak bisa memanipulasi struktur data secara langsung, tetapi harus memperlakukannya berkaitan dengan komponen primitif seperti bilangan integer, bilangan floating point, karakter tunggal, dan seterusnya. Perangkat lunak harus menterjemahkan operasi struktur data ke dalam seperangkat instruksi mesin yang beroperasi pada elemen data primitif individu.

Kompleksitas struktur data terbatas pada keterampilan dan imajinasi para programmer. Praktisnya, struktur data tertentu bermanfaat dalam situasi yang bervariasi. Sebagai contoh, karakter string atau array, record, dan file. Perangkat lunak sistem sering menyediakan jasa aplikasi untuk manipulasi struktur data. Sebagai contoh, suatu sistem operasi umumnya menyediakan jasa untuk membaca dan menulis ke dan dari file. Struktur data lain lebih sedikit didukung oleh perangkat lunak sistem. Contohnya meliputi array numerik, file indeks, dan struktur database kompleks. File indeks didukung oleh beberapa, tetapi tidak semua, sistem operasi. Array numerik umumnya mendukung

bahasa pemrograman tetapi bukan di dalam sistem operasi. Struktur database secara normal didukung oleh suatu database manajemen sistem. Kebanyakan bahasa program mendukung manipulasi langsung dari karakter string.

Struktur data mempunyai peran penting dalam pengembangan sistem software. Sebagai contoh, linked list biasanya digunakan oleh sistem operasi untuk menjejaki blok memori yang dialokasikan ke program dan blok disk yang dialokasikan ke file dan direktori. Indeks digunakan dalam database manajemen sistem untuk mempercepat pencarian dan memanggil kembali operasi.

Pointers and Addresses

Pointer adalah variabel khusus yang berisi suatu alamat (address) memori variabel lain dan secara tidak langsung menunjuk ke variabel tersebut, bias juga dikatakan register atau lokasi memori yang berisi alamat operand. Suatu variabel yang menunjuk (points) ke sesuatu sehingga disebut pointer. Ada dua macam pointer:

- typed (tertentu): merupakan pointer yang menunjuk pada tipe data tertentu pada variabel.
- generic (umum): merupakan pointer yang tidak menunjuk pada tipe data tertentu pada variabel.

Analoginya – sebagai contoh – Andi berteman dengan Budi, lalu anda ingin mengetahui jumlah keluarga Budi untuk keperluan sensus penduduk. Anda tidak mengetahui alamat Budi, tetapi anda mengenal Andi. Untuk mencari jumlah keluarga Budi, maka pertamanya anda pergi kerumah Andi, misalnya dirumah no 8321. Sesampai di Andi, Andi memberitahukan kepada anda bahwa alamat Budi ada pada alamat 9821. Kemudian anda pergi ke rumah Budi lalu mencatat jumlah keluarga yang dimiliki Budi yaitu lima orang (misalkan). Jadi dapat dilihat bahwa lokasi alamat dapat diubah, tetapi instruksi untuk pencarian alamat tetap sama. Dalam contoh di atas, Andi bertindak sebagai pointer. Andi tidak memberitahukan jumlah keluarga Budi, tetapi Andi memberitahu alamat Budi, di alamat 9821 (alamat Budi) itulah anda mengetahui jumlah keluarga Budi.

Pointer adalah salah satu konsep yang paling sulit dipahami oleh programmer pemula. Untuk memahami dengan baik bagaimana pointer bekerja, anda harus bisa membayangkan pergerakan variabel-variabel antara *microprocessor* dan memori utama, dan akan lebih baik jika anda tahu bagaimana sistem operasi semacam windows mengorganisasikan memori.

Pointer pada hakikatnya adalah variabel yang menyimpan alamat variabel lain yang tipe data-nya sama dengan tipe data pointer tersebut. Untuk mempermudah memahami pointer, terlebih dahulu akan dijelaskan bagaimana pengorganisasian memori dalam sebuah komputer oleh sistem operasi yang berjalan pada komputer tersebut. Sistem operasi modern umumnya menggunakan apa yang dikatakan *swap file* atau *page file*. *Page file* adalah bagian dari media penyimpanan eksternal (diluar *motherboard*) yang diperlakukan seperti RAM. Dalam sistem operasi seperti ini setiap program mempunyai wilayah memori sendiri (*memory space*) yang hanya dapat digunakan oleh program itu sendiri, tidak dapat digunakan oleh program lain. *Memory space* ini adalah bagian dari RAM namun bisa juga sebagian berada pada *swap file*, hal ini diatur oleh bagian

manajemen memori dari sistem operasi. Wilayah memori ini dibagi ke dalam beberapa komponen lagi, yaitu:

- *Global namespace*, bagian ini adalah bagian memory space yang dapat diakses oleh program yang kita miliki dengan bebas.
- *Code Space*, bagian ini menyimpan kode program yang akan dieksekusi (*machine code*) yang kita hasilkan dari source code program yang sudah di-compile.. Bagian ini adalah bagian yang read only, sebab kita hanya dapat mengeksekusi sebuah program, bukan merubah kode program itu sendiri saat dia dijalankan.
- *Stack*, yaitu bagian dari memory yang berfungsi untuk menyimpan data sementara, saat sebuah fungsi sedang dipanggil dan diproses.
- *Register*, yaitu bagian dari Microprocessor yang kita miliki yang dapat menyimpan data sementara ketika program sedang dijalankan. Register-register inilah yang memelihara keadaan sistem kita agar tetap berjalan sebagaimana mestinya. Tentang register akan dijelaskan lebih lanjut pada pemrograman assembly. Yang perlu kita ketahui saat ini adalah beberapa register yang bertugas menyimpan alamat kode program yang sedang dieksekusi saat ini, register ini disebut sebagai *instruction pointer*, selain itu ada register yang berfungsi menunjukkan alamat stack yang digunakan saat ini, register ini disebut *stack pointer* dan ada pula register yang bertugas menyimpan sementara alamat yang dimiliki oleh stack pointer, register ini disebut *stack frame*.
- *Free store*, adalah bagian memory di luar bagian memory yang telah di sebut di atas.

Sebelum melangkah lebih jauh, akan dijelaskan apa yang terjadi ketika sebuah fungsi dipanggil. Berikut ini adalah proses yang terjadi :

1. Pertama, alamat kode instruksi yang ditunjuk oleh instruction pointer dinaikkan satu instruksi, sehingga instruction pointer menunjuk ke alamat kode instruksi yang akan dieksekusi setelah fungsi yang dimaksud selesai dieksekusi. Alamat ini kemudian di simpan pada stack.
2. Kemudian, nilai stack pointer dinaikkan untuk menyediakan ruang bagi return value dari fungsi yang akan kita eksekusi.
3. Alamat kode instruksi pertama dari fungsi yang ingin kita eksekusi kemudian di pindahkan ke instruction pointer, sehingga fungsi tersebut adalah perintah yang akan dieksekusi selanjutnya.
4. Alamat stack saat ini yang ada pada register stack pointer kemudian di copy ke register stack frame pointer, mulai saat ini, data apapun yang dimasukkan ke stack dianggap lokal terhadap fungsi yang kita panggil.
5. Seluruh parameter fungsi kemudian disimpan atau tepatnya di copy ke stack
6. Instruction pointer kemudian mengeksekusi kode instruksi pertama yang telah ditunjuk pada langkah 3, sehingga fungsi dieksekusi.
7. Variabel lokal yang didefinisikan di dalam fungsi di simpan di stack, misalnya variabel c pada fungsi kali pada source code sebelumnya.
8. Fungsi telah selesai dieksekusi, hasil perhitungan dari fungsi tersebut kemudian diletakkan pada bagian stack yang telah disediakan pada langkah 2.

9. Alamat yang disimpan oleh register stack frame pointer kemudian di restore kembali ke register stack pointer, sehingga stack efektif saat ini tidak mengandung parameter fungsi dan variabel yang lokal terhadap fungsi.
10. Return value dari fungsi di pindahkan ke variabel yang menerima hasil fungsi tersebut.
11. Isi instruction pointer yang di simpan di stak pada langkah 1 kemudian dipindahkan ke instruction pointer , sehingga program kembali dilanjutkan.

Setelah membaca uraian di atas kita tahu bahwa ada beberapa alasan mengapa pointer diperlukan, yaitu:

1. Dalam kasus tertentu kita perlu mengubah isi dari sebuah variabel yang sebenarnya, kita tidak bisa melakukan hal ini dengan fungsi seperti yang telah kita pelajari sebab fungsi tersebut hanya memanipulasi "kopian" dari variabel yang sebenarnya ingin kita manipulasi (yang di simpan di stack ketika fungsi tersebut diproses)
2. Jika kita bisa merubah secara langsung variabel yang ingin kita manipulasi maka hal itu adalah lebih efisien dibandingkan mengkopi dan memanipulasinya, kemudian mengkopi kembali hasil manipulasi tersebut dari stack ke variabel aslinya.
3. Dalam kasus sebuah fungsi yang harus mengembalikan 2 atau lebih return value. Jelas hal ini tidak mungkin dilakukan dengan metode yang telah kita ketahui, sebab sebuah fungsi hanya mungkin mempunyai 1 return value (baca kembali bagian fungsi).

Sebuah pointer harus menunjuk ke alamat memori tertentu, jika tidak maka pointer tersebut bisa menimbulkan bug, pointer seperti ini disebut *stray pointer*. Dengan demikian jika anda belum mengisi sebuah pointer dengan alamat memori tertentu maka inisialisasi pointer tersebut dengan nilai 0 sehingga pointer tersebut menjadi *null pointer*. Null pointer adalah pointer yang tidak berbahaya, sehingga tidak akan menimbulkan bug seperti stray pointer.

Nama sebuah array sebenarnya merupakan pointer ke elemen pertama dari array tersebut, sehingga anda dapat mengakses elemen array tersebut dengan sebuah trik yang disebut dengan pointer arithmetic. Trik ini memanfaatkan nama array yang diperlakukan sebagai pointer.

Arrays and Lists

Array adalah suatu entitas (kesatuan) yang beranggotakan elemen-elemen (variabel) bertipe data sama dan dapat diakses dengan memanggil nama array beserta indeks elemennya. Dengan menggunakan array, sejumlah variabel dapat memakai nama yang sama. Antara satu variabel dengan variabel yang lain di dalam array dibedakan berdasarkan *subscript*. Sebuah *subscript* berupa bilangan didalam tanda kurung siku. Melalui *subscript* inilah masing-masing elemen array dapat diakses. Pada saat terbentuk pertama kali, elemen-elemennya akan memakai nilai default 0 untuk tipe numerik, kosong (blank) untuk karakter, dan salah (false) untuk boolean.

Indeks adalah bilangan integer yang menunjukkan letak urutan elemen itu dalam array. Indeks array berukuran N dimulai dari 0 hingga N-1. Perlu diketahui bahwa setiap komponen array dapat diakses dengan sebuah nilai yang disebut *indeks*. Indeks menunjukkan letak komponen tersebut di dalam array. Indeks paling kecil adalah 0, yaitu indeks yang menunjukkan komponen pertama dari array. Indeks 1 menunjukkan komponen ke-2, dan seterusnya. Satu hal yang perlu diwaspadai saat menggunakan array, yaitu pastikan bahwa anda tidak mengakses indeks array yang melebihi jumlah komponen array tersebut, sebab hal ini akan menjadi bug pada program anda. Misalnya anda mendeklarasikan sebuah array mempunyai 9 komponen, maka indeks maksimum yang dapat anda gunakan adalah `nama_array[8]`, sebab indeks dimulai dari 0. Selain berupa deretan variabel satu dimensi, dapat juga dibuat array yang berukuran lebih dari satu dimensi atau disebut juga array multidimensi. Array dua dimensi ini dikenal dengan matrix dua dimensi berukuran $m \times n$. Misalnya: e_{xy} , artinya elemen pada baris ke- x dan kolom ke- y .

Representasi Array

Misalkan kita memiliki sekumpulan data *ujian* seorang siswa, *ujian* pertama bernilai 90, kemudian 95,78,85. Sekarang kita ingin menyusunnya sebagai suatu data kumpulan *ujian* seorang siswa. Dalam array kita menyusunnya sebagai berikut

```
ujian[0] = 90;
ujian[1] = 95;
ujian[2] = 78;
ujian[3] = 85;
```

Perhatikan :

- Tanda kurung [] digunakan untuk menunjukkan elemen array
- Perhitungan elemen array dimulai dari 0, bukan 1

Empat pernyataan di atas memberikan nilai kepada array *ujian*. Tetapi sebelum kita memberikan nilai kepada array, kita harus mendeklarasikannya terlebih dahulu, yaitu:

```
int ujian[4];
```

Perhatikan bahwa nilai 4 yang berada didalam tanda kurung menunjukkan jumlah elemen array, bukan menunjukkan elemen array yang ke-4. Jadi elemen array *ujian* dimulai dari angka 0 sampai 3.

Pemrogram juga dapat menginisialisasi array sekaligus mendeklarasikannya, sebagai contoh :

```
int ujian[4] = {90,95,78,85};
```

Elemen terakhir dari array diisi dengan karakter '\0'. Karakter ini memberitahu kompiler bahwa akhir dari elemen array telah dicapai. Walaupun pemrogram tidak dapat melihat karakter ini secara eksplisit, namun kompiler mengetahui dan membutuhkannya.

Sekarang akan dibuat daftar beberapa nama pahlawan di Indonesia:

```
char pahlawan[3][15];
```

```
char pahlawan[0][15] = "Soekarno";
```

```
char pahlawan[1][15] = "Diponegoro";  
char pahlawan[2][15] = "Soedirman";
```

Array di atas terlihat berbeda dengan contoh array pertama kita. Perhatikan bahwa pada array *pahlawan* memiliki dua buah tanda kurung [][]. Array seperti itu disebut array dua dimensi. Tanda kurung pertama menyatakan total elemen yang dapat dimiliki oleh array pahlawan dan tanda kurung kedua menyatakan total elemen yang dapat dimiliki setiap elemen array pahlawan. Dalam contoh di atas, tanda kurung kedua menyatakan karakter yang menyatakan nama pahlawan.

Records and Files

Suatu record adalah suatu struktur data yang terdiri dari struktur data yang lain atau elemen data primitif. Record umumnya digunakan sebagai suatu unit keluaran dan masukan ke database atau file. Untuk mempercepat input dan output, record biasanya disimpan dalam lokasi penyimpanan contiguous.

Gambar Struktur Record Data

Account- Number	Last- Name	First- Name	Middle- Initial	Street- Address	City	State	Zip- Code
--------------------	---------------	----------------	--------------------	--------------------	------	-------	--------------

Suatu urutan record yang terdapat dalam penyimpanan sekunder disebut suatu file. Suatu urutan record yang disimpan di dalam memori utama disebut suatu tabel, meskipun strukturnya sama dengan file. File dapat diorganisasikan dalam banyak cara, tapi umumnya secara berurutan (sequential) dan diindeks. Dalam file yang berurutan record biasanya disimpan di lokasi penyimpanan contiguous.

Sequential file memiliki masalah yang sama dengan array ketika record harus disisipkan maupun dihapus. Prosedur kurang efisien untuk file dibandingkan array terlebih bila filenya relatif berukuran besar. Untuk mengatasi masalah ini digunakanlah linked list. Metode lain yang digunakan untuk mengorganisir file adalah dengan indeks. Indeks adalah suatu array dari pointer menjadi record. Pointer bisa diminta melakukan pengurutan sesuai keinginan pengguna. Keuntungan lain menggunakan indeks adalah efisien dalam menyisip, menghapus, dan memanggil record.

Classes and Objects

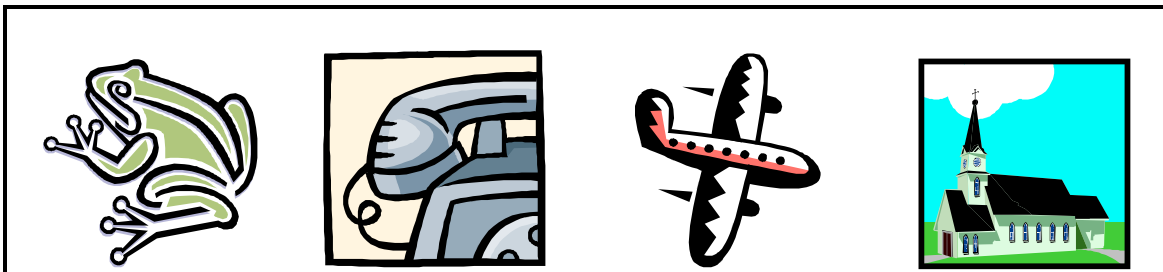
- **Objek**

Objek merepresentasikan sebuah entitas, baik secara fisik, konsep ataupun secara perangkat lunak. Definisi yang formal dari objek adalah sebuah konsep, abstraksi atau sesuatu yang diberi batasan jelas dan dimaksudkan untuk sebuah aplikasi.

Sebuah objek adalah sesuatu yang mempunyai keadaan, kelakuan dan identitas. Keadaan dari objek adalah satu dari kondisi yang memungkinkan dimana objek dapat muncul, dan dapat secara normal berubah berdasarkan waktu. Keadaan dari objek biasanya

diimplementasikan dengan kelompok propertinya (disebut atribut), berisi nilai dari properti tersebut, ditambah keterhubungan objek yang mungkin dengan objek lainnya. Kelakuan menentukan bagaimana sebuah objek beraksi dan bereaksi terhadap permintaan dari objek lainnya. Direpresentasikan dengan kelompok pesan yang direspon oleh objek (operasi yang dilakukan oleh objek). Kelakuan dari objek mendeskripsikan segala sesuatu yang dapat kita lakukan terhadap objek tersebut dan segala sesuatu yang dapat dilakukan oleh objek untuk kita.

Setiap objek mempunyai identitas yang unik. Identitas yang unik ini membuat kita dapat membedakan dua objek yang berbeda, walaupun kedua objek tersebut mempunyai keadaan dan nilai yang sama pada atributnya.



Gambar contoh objek

- **Kelas**

Kelas adalah deskripsi dari kelompok objek dengan properti yang sama (atribut), kelakuan yang sama (operasi), serta *relationship* dan semantik yang sama. Dimana telah dinyatakan, bahwa sebuah objek adalah instansiasi dari kelas. Sebuah kelas adalah sebuah hasil abstraksi dari sesuatu dengan mengelompokkan karakteristik yang sejenis dengan mengabaikan karakteristik lainnya. Contoh penerapan sebuah kelas adalah sebagai berikut: kita bisa mempunyai kelas dengan diberi nama kursus, maka dapat mempunyai properti atau atribut berupa lokasi, hari yang ditawarkan, waktu SKS, waktu mulai dan waktu berakhirnya, juga bisa mempunyai kelakuan (operasi) seperti tambah siswa, kurangi siswa, dapatkan jadwal kursus maupun penentuan kelas yang penuh. Karena kelas merupakan wadah yang akan digunakan untuk menciptakan objek, maka jelaslah bahwa harus dibuat kelas terlebih dahulu sebelum membuat objek.

Ilustrasi contoh perbedaan kelas dengan objek adalah sebagai berikut:

Jika kita mempunyai benda berupa burung elang, ikan hiu, gajah, telepon, televisi dan kamera, maka jumlah objek yang dimiliki sebanyak jumlah benda yang ada yaitu enam buah objek, namun kita dapat menentukan kelas dari objek tersebut yaitu berdasarkan sifat yang terdiri dari dua kelas, yaitu benda mati dan benda hidup. Akan tetapi jika atribut dan operasi diperinci, akan diperoleh lebih banyak jenis kelas yang memiliki batasan lebih lengkap seperti kelas mamalia, burung, audio, visual dan audio visual. Kelas adalah definisi abstrak dari sebuah objek, dimana dijelaskan bahwa struktur dan kelakuan dari tiap objek yang tergabung dalam suatu kelas. Dari kelas dapat disediakan fasilitas untuk membuat suatu objek dan beberapa objek yang dikelompokkan ke dalam satu kelas. Sebelumnya telah dinyatakan bahwa kelas adalah deskripsi dari kumpulan objek yang mempunyai tanggung jawab yang sama, keterhubungan yang sama, operasi, atribut dan semantik yang sama pula. Dapat dinyatakan bahwa sebuah objek dijelaskan di

sebuah kelas, kelas menjelaskannya dengan bentuk struktur dan kelakuan dari semua objeknya. Sebuah objek yang diciptakan dari sebuah kelas disebut juga instansiasi dari kelas, dengan kata lain kelas adalah deskripsi statik dan objek adalah instansiasi dinamis dari kelas.

DAFTAR PUSTAKA

Burd, Stephen D., **Systems Architecture**, Fourth Edition, Thomson Course Technology, New Mexico, 2003

Davis, William S., Alih Bahasa John B. Pasaribu, **Sistem Pengolahan Informasi**, Edisi Kedua, Penerbit Erlangga, Jakarta, 1983

Hamacher, Carl., Vranesic, Zvonko., Zaky, Safwat., **Organisasi Komputer**, Edisi Kelima, Penerbit Andi, Yogyakarta, 2004

Hariyanto, Bambang, **Esensi-esensi Bahasa Pemrograman JAVA**, Penerbit Informatika, Bandung, 2003

Hermawan, Benny, **Menguasai Java2 & Object Oriented programming**, Penerbit Andi, Yogyakarta, 2004

Malvino, Albert Paul, **Elektronika Komputer Digital Pengantar Mikrokomputer**, Edisi Kedua, Penerbit Erlangga, Jakarta, 1989

Soeparlan, Soepono, **Pengantar Organisasi Sistem Komputer**, Gunadarma, Jakarta, 1995

Stallings, William, Alih Bahasa Gurnita Priatna, **Organisasi dan Arsitektur Komputer Perancangan Kinerja**, Jilid 1, Edisi Bahasa Indonesia, PT Indeks, kelompok Gramedia, Jakarta, 2003

<http://www.cs.ui.ac.id/kuliah/IKI10100/slides/10-11-Stacks-Queue-4pp.pdf#struktur> data & algoritme

http://overclockerindo.com/modules.php?name=Reviews&rop=showcontent&id=22#Pointer_Array

<http://www.mti.gadjahmada.edu/~rudy/VII%20%20%20STRUKTUR%20ARRAY.pdf>

Didapatkan dari halaman web "http://id.wikipedia.org/wiki/Struktur_data"

<http://www.ilmukomputer.com/umum/chendra-pemrograman.php>

http://svl.petra.ac.id/files/processingdc_manual.pdf

<http://www.indodigest.com/indonesia-special-article-1.html>

<http://ikc.dephan.go.id/umum/taufik-visualisasi.php>

<http://ikc.dephan.go.id/pengantar/abepoetra-matbiner.php>

<http://ikc.dephan.go.id/umum/ibam-os.php>

<http://ikc.dephan.go.id/umum/nursapta-javafundamental.php>

<http://ikc.dephan.go.id/berseri/eko-java/index.php>

<http://ikc.dephan.go.id/populer/sudewa-set.php>

<http://ikc.dephan.go.id/berseri/eko-tipsjava/index.php>

Universitas Kristen Duta Wacana Students Community tipe data.htm

Universitas Kristen Duta Wacana Students Community pemrograman.htm

<http://www.ilmukomputer.com/pengantar/abepoetra-matbiner.php>

<http://www.ilmukomputer.com/umum/ibam-os.php>

<http://www.ilmukomputer.com/populer/dindin-sti.php>

<http://www.ilmukomputer.com/berseri/teguh-cbis/index.php>

<http://www.ilmukomputer.com/berseri/ginajar-java/index.php>

<http://www.ilmukomputer.com/populer/sudewa-set.php>

<http://www.ilmukomputer.com/pengantar/noor-informatika.php>

<http://www.ilmukomputer.com/tipstrik/pemrograman/index.php>

<http://lecturer.eepis-its.edu/~hero/kuliah/Ketrampilan%20Komputer/01%20-%20Komputer%20dan%20Perangkat%20Lunak.pdf>

<http://www2.ukdw.ac.id/tutorial/php/page03.htm#integer> tipe data php

