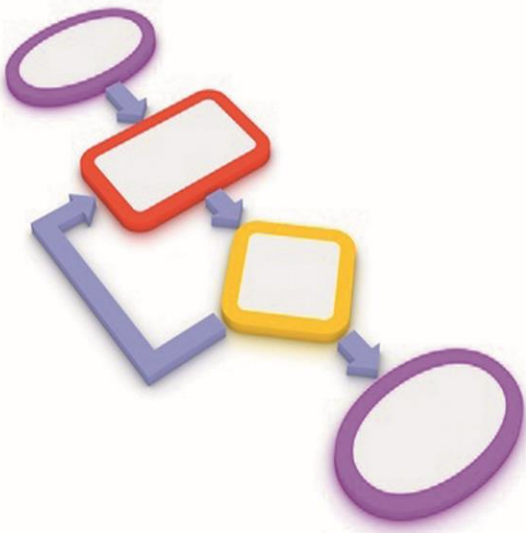


Pengantar **ALGORITMA** dengan Bahasa C



DAFTAR ISI

DAFTAR TABEL	7
BAB I PENDAHULUAN	8
1.1 MASALAH DAN SOLUSI	8
1.2 ALGORITMA.....	9
SEJARAH	9
Syarat algoritma	9
Kriteria algoritma yang baik	10
Jenis Proses Algoritma	10
Tahap Algoritma dan Penulisan Program	11
1.3 PENULISAN ALGORITMA	12
Pseudocode	13
FLOW CHART	15
SOAL LATIHAN	16
BAB II ELEMEN DAN STRUKTUR PROGRAM C	17
SEJARAH BAHASA C	17
KARAKTER PEMBENTUK PROGRAM	18
IDENTIFIER.....	18
Keyword	18
Standart Identifier	19
TIPE DATA (DATA TYPE).....	20
KONSTANTA	21
VARIABLE.....	22
OPERASI DAN OPERATOR.....	23
Operator Aritmatika	23
Operator Relasi	24
Operator Logika	24
Operator Bitwise	25
Operator Koma	25
Operator Bertingkat	25
Operator bersyarat	26
Operator Cast	26
Operator Penugasan (Assignment)	26
Presedensi (Precedence) dan Asosiasi (Associativity)	27
EKSPRESI (EXPRESSION)	28
STRUKTUR PROGRAM C	28

Komentar	29
SOAL LATIHAN	30
BAB III INPUT OUTPUT	31
INSTRUKSI KELUARAN	31
STANDARD I/O DAN REDIRECTION	36
PENEMPATAN KURSOR	37
PEMBERSIHAN LAYAR	37
SOAL LATIHAN	39
BAB IV STRUKTUR KENDALI PROSES	40
STRUKTUR KENDALI PEMILIHAN	40
Instruksi if.....	40
Klausa else.....	41
Instruksi switch	42
STRUKTUR KENDALI PENGULANGAN	44
Instruksi for	44
Instruksi while	45
Instruksi do while	46
Infinite Loop	47
break dan continue	47
MENGUJI KEBENARAN PROGRAM (PROGRAM TESTING)	48
Metode Black Box	48
Uji dengan Data yang Mudah Diperiksa.....	48
Uji dengan Data yang Wajar	48
Uji dengan Nilai Batas	48
Uji dengan Data Illegal	49
Metode White Box.....	49
Soal Latihan.....	50
BAB V FUNCTION	51
KATEGORI FUNCTION.....	51
1. Standard library function.	51
2. Programmer defined function.	52
STRUKTUR FUNCTION	52
STANDARD LIBRARY FUNCTION	53
PROGRAMMER-DEFINED FUNCTION	53
Deklarasi dan Definisi Function.....	53
Return Type dan return.....	55

Memanggil Function	55
Letak Function.....	56
JANGKAUAN IDENTIFIER DAN JENIS VARIABEL.....	57
Variabel Lokal.....	58
Variabel Eksternal	58
Variabel Statis	59
Variabel Register	60
Variabel Pointer	60
PENGIRIMAN PARAMETER (ARGUMEN)	60
Pengiriman Berupa Nilai (Call By Value)	61
Pengiriman Berupa Alamat Data (Call By Reference)	61
REKURSI (RECURSION)	62
Soal Latihan	63
BAB VI TIPE DATA TERSTRUKTUR.....	64
6.1 ARRAY.....	64
Array Satu Dimensi.....	64
Pointer dan Array	65
Array Multi Dimensi	67
6.2 STRING	67
6.3 STRUCT	69
Point to Structure.....	71
Structure Sebagai Parameter	71
6.4 UNION	72
SOAL LATIHAN	73
BAB VII FILE DATA	74
7.1 FILE DAN STREAM	74
TEXT FILE DAN BINARY FILE.....	75
INSTRUKSI I/O FILE DATA	75
7.2 FIELD, RECORD, DAN FILE.....	78
Sequential Access.....	80
7.3 STANDARD I/O STREAM.....	80
LATIHAN SOAL.....	82
BAB VIII PENGURUTAN.....	83
BUBBLE SORT	83
SELECTION SORT	84
INSERTION SORT	85

QUICK SORT.....	86
MERGE SORT	86
HEAP SORT	87
SHELL SORT	88
LATIHAN SOAL.....	89

DAFTAR PROGRAM

Program 1. clrscr() sebagai standard Identifier.....	19
Program 2.clrscr() didefinisikan kembali.....	20
Program 3.Tabel ASCII.....	29
Program 4.Pemakaian putch(), putchar(), dan puts()	33
Program 5.pemakaian printf() terhadap data karakter	33
Program 6.Masukan Data Karakter.....	34
Program 7.Nilai Akhir Mata Kuliah.....	35
Program 8.Luas Persegi Panjang	36
Program 9.Tabel Perkalian	36
Program 10.Pengaturan Cursor	37
Program 11.penghapusan layar	38
Program 12.Grade nilai	41
Program 13.Evaluasi Studi	42
Program 14.Uji Bilangan	42
Program 15.Kode Pos dan wilayah	43
Program 16.Case tanpa break.....	43
Program 17.Grafik sederhana	45
Program 18.Luas persegi panjang-persegi panjang.....	46
Program 19.Pilih Y atau T.....	47
Program 20.Deret bilangan ganjil	49
Program 21.Memanggil Function yang didefinisikan kemudian (di bawah)	56
Program 22.Memanggil Funtion yang didefinisikan sebelumnya (di atas).....	56
Program 23.Function memanggil function	57
Program 24.Perubahan Nilai variabel global	59
Program 25.Variabel Statis.....	59
Program 26.variabel pointer	60
Program 27.Jumlah Ganda.....	61
Program 28.pertukaran Setempat	61
Program 29.Menukar Isi.....	62
Program 30.Penjumlahan Deret Kuadrat.....	62

DAFTAR TABEL

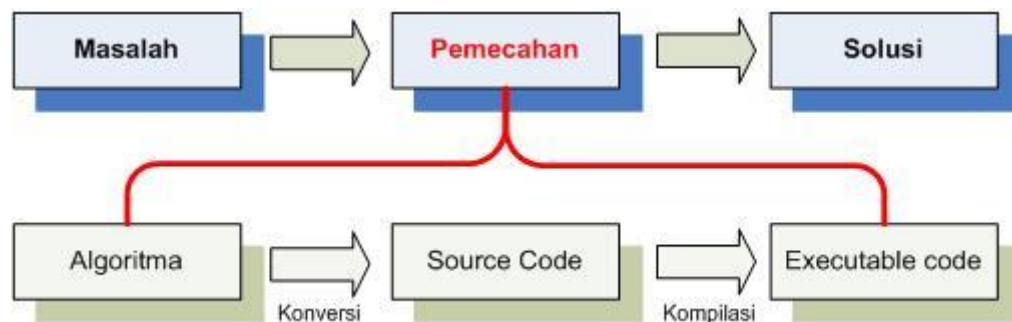
Tabel 1. Karakter Khusus Pembentuk Program C	18
Tabel 2. keyword bahasa C	19
Tabel 3. Tipe Data Dasar	20
Tabel 4. Tipe Data dan Modifier	20
Tabel 5. Escape Sequence	22
Tabel 6. Operator Aritmatika	23
Tabel 7. Operasi Relasi	24
Tabel 8. Operasi Logika	24
Tabel 9. Kebenaran Dua Variabel	24
Tabel 10. Operasi Bitwise	25
Tabel 11. Penyingkatan Penulisan Assignment	27
Tabel 12. Presedensi dan Asosiativitas operator	28
Tabel 13. Flag Characters	32
Tabel 14. Karakter Konversi Tipe Data	32
Tabel 15. Type Character pada instruksi scanf()	35
Tabel 16. Fungsi Standard Matematika	53

BAB I PENDAHULUAN

Dalam kehidupan nyata setiap manusia tidak akan terlepas dari sebuah masalah. Dari setiap masalah yang timbul diperlukan jalan keluar atau pemecahan masalah yang disebut dengan solusi. Setiap permasalahan yang dipecahkan dengan bantuan teknologi komputerisasi memerlukan algoritma. Secara umum algoritma dapat diartikan suatu tata cara untuk memecahkan suatu masalah dengan bantuan teknologi komputerisasi atau dapat disebut juga sebagai logika penulisan suatu program.

1.1 MASALAH DAN SOLUSI

Untuk setiap masalah perlu ditemukan solusi atau pemecahannya sehingga masalah tersebut dapat terselesaikan dengan benar atau yang dianggap paling benar. Dalam pemecahan masalah dengan menggunakan teknologi komputerisasi maka diperlukan suatu algoritma



gambar 1. Algoritma dalam pemecahan masalah

Secara umum algoritma adalah sejumlah tahapan komputasi yang mengubah masukan (*input*) menjadi keluaran (*output*) sehingga menghasilkan solusi yang benar. Algoritma bentuk awalnya berupa logika pemecahan masalah kemudian diubah menjadi program komputer (*source code*), selanjutnya *source code* tersebut perlu dilakukan proses kompilasi oleh penterjemah (*compiler, interpreter*) menjadi kode – kode yang dapat dimengerti dan dijalankan mesin komputer. Kode ini disebut juga dengan *executable code*. Pada saat *excuteable* code dijalankan, data dimasukkan melalui pengetikan pada keyboard atau membaca file data, kemudian diolah, dan hasilnya akan diinformasikan dalam bentuk tampilan dilayar monitor, cetakan printer atau dalam bentuk file data.

1.2 ALGORITMA

SEJARAH

Algoritma berasal dari seorang ilmuwan Persia yang bernama Abu Ja'far Mohammed Ibn Musa al-Khowarizmi, yang menulis buku berjudul *Kitab al Jabr wa'al-muqabala (rules of restoration and reduction)* pada sekitar tahun 825.

Terdapat beberapa kata yang mirip dengan algoritma seperti algorism dan algorithmus. Kata algorism digunakan untuk proses perhitungan aritmetika dengan menggunakan angka Arab. Kata algorithmus digunakan pada kamus matematika *Vollstandiges Mathematisches Lexicon* (Leipzig, 1747) untuk kombinasi dari empat jenis perhitungan matematika: penjumlahan, perkalian, pengurangan dan pembagian. Dalam Frasa bahasa latin algorithmus infinitesimalis saat ini digunakan untuk menyatakan cara berhitung dengan menggunakan bilangan kecil tidak terbatas seperti yang dikemukakan leibnitz, Hingga pada tahun 1950 istilah algorithm selalu diasosiasikan dengan Euclid's algorithm, yaitu suatu proses yang menjelaskan cara mencari bilangan pembagi terbesar untuk dua buah bilangan (*greatest common divisor*).

Istilah algorithm diartikan sebagai prosedur langkah demi langkah untuk memecahkan masalah atau menyelesaikan suatu tugas khusus dengan menggunakan bantuan komputer pada *Merriam-Webster's Collegiate Dictionary*. Pada kamus Besar Bahasa Indonesia (KBBI) mendefinisikan algoritma sebagai urutan logis pengambilan keputusan untuk pemecahan masalah.

Syarat algoritma

Suatu algoritma haruslah memenuhi beberapa persyaratan menurut Donald E.Kruth, antara lain:

1. Finiteness;

Algoritma harus mempunyai akhiran / berakhir (terminate) setelah melakukan sejumlah proses

2. Definitness;

Setiap langkah algoritma harus didefinisikan dengan tepat dan tidak menimbulkan makna ganda (ambiguous).

3. Input;

Setiap algoritma memerlukan data sebagai masukan untuk diolah. Algoritma yang tidak memerlukan masukan apapun sesungguhnya tidak begitu bermanfaat karena jumlah kasus yang dapat diselesaikan juga terbatas.

4. Output;

Setiap algoritma memberikan satu atau beberapa hasil keluaran

5. Effectiveness;

Langkah – langkah algoritma dikerjakan dalam waktu yang wajar.

Kriteria algoritma yang baik

1. Mempunyai logika yang tepat untuk memecahkan masalah
2. Menghasilkan output yang benar dalam waktu singkat
3. Ditulis dengan bahasa baku terstruktur sehingga dengan format baru sehingga mudah diimplementasikan kedalam bahasa pemrograman
4. Semua operasi didefinisikan dengan jelas dan berakhir sesudah sejumlah.

Jenis Proses Algoritma

Langkah yang membentuk suatu algoritma dapat dibagi menjadi 3 kelompok proses, antara lain:

1. *Sequences Process*

Instruksi kedua dikerjakan setelah instruksi pertama dikerjakan. Setelah itu instruksi ketiga baru dikerjakan. Instruksi dikerjakan satu demi satu mulai dari instruksi pertama sampai dengan instruksi terakhir. *Sequence* merupakan urutan pengerjaan dari perintah / *statement* pertama sampai dengan perintah / *statement* terakhir. Umumnya bahasa pemrograman mempunyai *sequence* (urutan pengerjaan dari perintah / *statement*) mulai dari atas ke bawah dan dari kanan ke kiri. Instruksi dikerjakan satu demi satu mulai dari instruksi pertama sampai dengan instruksi terakhir

2. *Selection Process*

Suatu instruksi dimana baru boleh dikerjakan apabila memenuhi persyaratan tertentu.

3. *Iteration / Repetition Process*

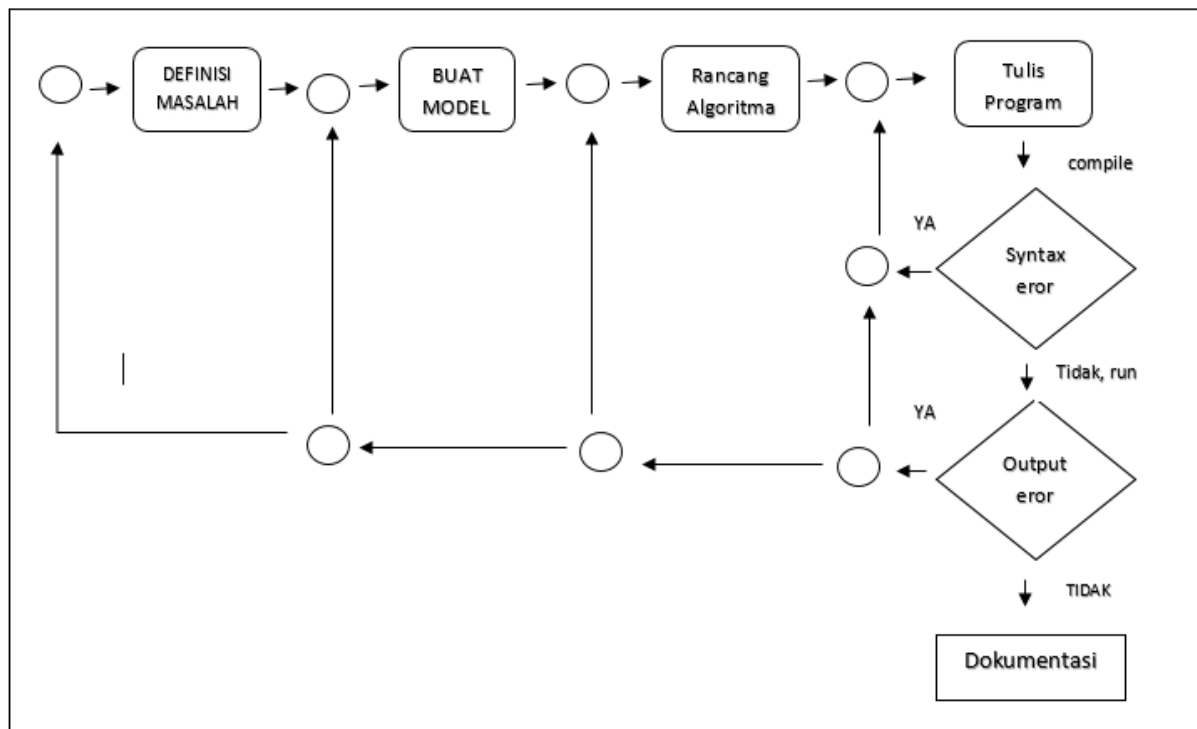
Suatu instruksi dimana dikerjakan berulang – ulang selama sekian kali atau selama suatu kondisi masih terpenuhi.

Struktur Kontrol Selection adalah penggambaran sebuah kondisi dan pilihan diantara dua aksi. Statement Pertama akan dikerjakan jika kondisi bernilai benar, jika tidak maka akan mengerjakan perintah setelah *keyword “else”* (jika ada). Suatu instruksi baru boleh dikerjakan apabila memenuhi persyaratan tertentu

Tahap Algoritma dan Penulisan Program

Proses pemecahan masalah dengan algoritma tertentu hingga menjadi program dapat dibagi menjadi sembilan tahapan.

1. Mendefinisikan Masalah;
Masalah yang akan dipecahkan harus jelas lingkupannya
2. Membuat model;
Model yang dimaksud adalah model / bentuk matematis yang dapat digunakan untuk memecahkan masalah, misalnya apakah harus dilakukan pengurutan atau melakukan perhitungan dan sebagainya
3. Merancang algoritma;
Membuat alur bagaimana rincian prosesnya, apa keluarannya
4. Menulis program
Merubah algoritma menjadi program (source code) sesuai dengan bahasa pemograman tertentu.
5. Mengubah source code menjadi executable code (compiling)
6. Memeriksa hasil compiling, jika terdapat kesalahan maka kembali ke tahap empat
7. Menjalankan program(run) untuk diuji kebenarannya dengan menggunakan berbagi data.
8. Memperbaiki kesalahan
Apabila hasilnya salah, kemungkinan terdapat kesalahan pada saat konversi rancangan algoritma menjadi program, atau salah rancangan algoritma, maka ulangi langkah yang sesuai.
9. Mendokumentasi program bila sudah benar



Gambar 1.2 Bagan tahapan algoritma hingga program

1.3 PENULISAN ALGORITMA

Algoritma bersifat programming language independent. Sebuah algoritma dapat diimplementasikan dengan berbagai bahasa pemrograman tetapi penulisannya tidak bergantung kepada bahasa pemrograman tertentu. Algoritma dapat disajikan menggunakan dua teknik, yaitu: tulisan dan gambar. Penyajian algoritma dalam bentuk tulisan biasanya menggunakan metode seperti: Bahasa Indonesia Terstruktur (BIT), pseudocode, spark, Structured English, Penyajian algoritma dengan menggunakan teknik gambar biasanya menggunakan metode seperti : *Flow chart*, *hierarchy plus input-process-output (HIPO) chart*, *structured chart*, dan *massi-schneiderman chart*.

BIT menggunakan beberapa istilah seperti:

1. untuk mendapatkan data masukkan: isi, baca, masukkan, baca file, ketik
2. untuk menampilkan keluaran hasil proses: tulis, cetak, tampilkan, rekam
3. untuk menyatakan proses pemilihan: jika..., selain itu..., akhir jika
4. untuk menyatakan proses pengulangan: ulangi,...sampai,...selama, akhir selama,ulangi selama,...akhir pengulangan

Pseudocode menggunakan beberapa istilah berikut:

1. untuk mendapatkan data masukkan: *input, read, get, key-in*
2. untuk menampilkan keluaran hasil proses: *print, write, display*
3. untuk menyatakan proses pemilihan: *if....else...endif*
4. untuk menyatakan proses pengulangan: *repeat....until....,while..... do while. End while. For.....do.....endfor*

Pseudecode

Dapat dikatakan sebuah outline dari sebuah program komputer yang ditulis dalam bahasa inggris atau indonesia sederhana, kata kunci / keyword digunakan untuk menjelaskan struktur kendalinya. Pseudecode biasanya mewakili enam operasi dasar komputer:

1. Menerima informasi (Input);
Contoh : Read bilangan, Get kode_pajak, Baca nama_mahasiswa
2. Menampilkan Informasi (output);
Contoh : Print "Unsada", cetak "Darma Persada"
3. Melakukan perhitungan aritmatike (Compute);
Contoh : + untuk penjumlahan (add), - untuk pengurangan (subtract) dst
4. Memberikan nilai ke suatu identifier (Store);
Contoh : memberi nilai awal "set", memberikan nilai hasil "=", menyimpan suatu nilai "store", Set mhs to 0, total = harga * jumlah
5. Membandingkan dan memilih (Compare);
Contoh : IF pilih = 1 thenelse...
6. Melakukan pengulangan (Loop);
Contoh : Dowhile bil < 10

Contoh algoritma dengan BIT

Contoh : Algoritma Berangkat Kuliah

Mulai

Bangun dari tempat tidur

Mandi Pagi

Sarapan Pagi

Pergi Ke Kampus

Cari Ruang Kuliah

Masuk kelas untuk Kuliah

Selesai

Contoh algoritma dengan peseudocode

Contoh : Algoritma Menggunakan Kalkulator

Mulai

Nyalakan kalkulator

Kosongkan Kalkulator

Ulangi

 Input harga

 Tekan tombol Plus (+)

Sampai semua harga diinput

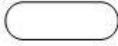








Tampilkan total harga

Matikan kalkulator

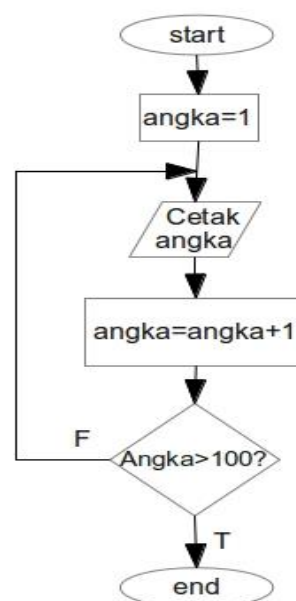
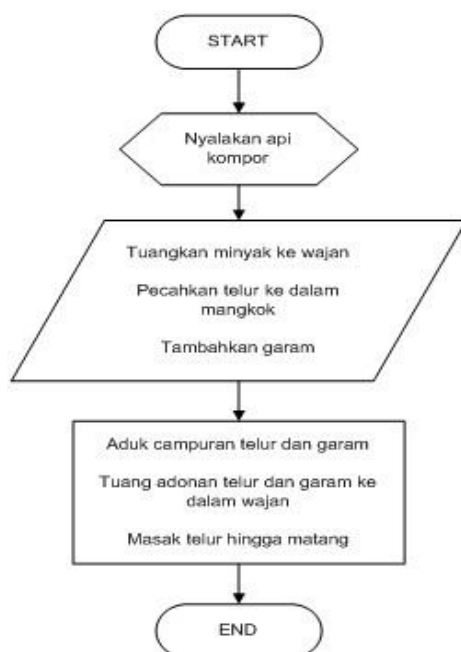
Selesai

FLOW CHART

Merupakan salah satu metode untuk menuliskan algoritma dalam bentuk gambar, flowchart dapat ditampilkan dalam beberapa bentuk bilangan datar, antara lain:

SIMBOL	NAMA	FUNGSI
	TERMINATOR	Permulaan/akhir program
	GARIS ALIR (FLOW LINE)	Arah aliran program
	PREPARATION	Proses inialisasi/pemberian harga awal
	PROCESS	Proses perhitungan/proses pengolahan data
	INPUT/OUTPUT DATA	Proses input/output data, parameter, informasi
	PREDEFINED PROCESS (SUB PROGRAM)	Permulaan sub program/proses menjalankan sub program
	DECISION	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	ON PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	OFF PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Contoh Algoritma dengan Flow Chart,



SOAL LATIHAN

1. Apa yang dimaksud dengan algoritma?
2. Apa gunanya bahasas pemograman?
3. Apa yang dimaksud dengan Source code?
4. Apa yang dimaksud dengan exceute Code?
5. Apakah suatu algoritma dapat diimplementasikan dengan dua bahasa pemograman yang berbeda?
6. Jenis Software apa yang diperlukan untuk menterjemahkan source code menjadi execute code?
7. Sebutkan syarat suatu algoritma?
8. Sebutkan tiga kelompok proses algoritma?
9. Jika executable code telah terbentuk apakah masih diperlukan compiler?
10. Sebutkan tahapan algoritma?
11. Apa beda compiler dan intepreter?
12. Jelaskan yang dimaksud dengan pseudecode?
13. Jelaskan yang dimaksud dengan flowchart?
14. Buatlah algoritma menggunakan pseudecode untuk menghitung luar persegi?
15. Buatlah algoritma menggunakan flowchart untuk proses kehidupan?

BAB II ELEMEN DAN STRUKTUR PROGRAM C

Program adalah kumpulan instruksi yang akan melakukan proses komputasi terhadap data menjadi informasi. Pada bab ini dijelaskan karakter pembentuk komponen program, berbagai jenis identifier, konstanta, tipe data beserta operasi yang dapat dilakukan terhadapnya, operator yang digunakan sebagai simbol operasi, dan variabel yang digunakan sebagai tempat penampungan data.

SEJARAH BAHASA C

Bahasa C dirancang oleh Dennis M. Ritchie di Bell Laboratories pada tahun 1972. Bahasa C dikembangkan dari bahasa BCPL atau bahasa B. Bahasa BCPL dikembangkan Martin Richards pada tahun 1967 sebagai bahasa untuk menulis sistem operasi dan compiler. Pada tahun 1970 Ken Thompson merancang bahasa B dengan memasukkan feature BCPL. Bahasa B dirancang dengan tujuan untuk digunakan membuat sistem operasi UNIX untuk komputer DEC PDP-7 pada Bell Laboratories.

Pada tahun 1978 Dennis M. Ritchie dan Brian W. Kernighan mempublikasi bahasa C melalui buku *The C Programming Language*. Pada tahun 1983 dibentuk komite teknis X3J11 dibawah American National Standards Committee on Computers and Information Processing (X3) untuk menyusun standar bahasa C. Standar ini terbentuk pada tahun 1989. Pembakuan ini diperbarui pada tahun 1999 dan tercantum pada dokumen ISO/IEC 9899:1999.

Bahasa pemrograman C merupakan bahasa yang bersifat umum (*general purpose language*), tidak dikhususkan untuk bidang aplikasi tertentu. Bahasa pemrograman ini digolongkan sebagai bahasa pemrograman tingkat menengah (*medium level language*). Hal ini dikarenakan bahasa C memiliki kemampuan dalam mengakses mesin komputer yang mendekati kemampuan bahasa rakitan, tetapi mudah dipelajari dan digunakan seperti halnya bahasa pemrograman tingkat tinggi. Selain itu bahasa pemrograman C memiliki karakteristik lain seperti hemat ekspresi, alur kontrol, menggunakan struktur data modern, dan kaya dengan operator.

Sifatnya yang umum dan longgarnya batasan – batasan yang diberikan membuat bahasa C lebih mudah digunakan dan lebih efektif dalam menyelesaikan berbagai permasalahan dibandingkan dengan bahasa – bahasa pemrograman lain. Dalam menyusun sebuah program C, seorang pemrogram diberikan keleluasaan penuh dalam mengimplementasikan konsep bahasa pemrograman inti ke dalam bentuk instruksi – instruksi program. Selain itu, seorang pemrogram juga diberikan tanggung jawab dan kendali penuh terhadap program yang dibuatnya. Hematnya ekspresi yang digunakan dalam program C juga merupakan kelebihan bahasa pemrograman ini, karena akan mengurangi jumlah baris perintah program. Namun di lain pihak kehematan ekspresi ini dapat pula membuat program yang disusun menjadi lebih sulit dibaca.

Selain itu bahasa C merupakan bahasa pemrograman yang banyak digunakan mulai dari komputer mikro hingga ke supercomputer. Sampai saat ini bahasa C merupakan bahasa pemrograman yang paling banyak digunakan untuk membuat sistem operasi dan sistem software lainnya.

KARAKTER PEMBENTUK PROGRAM

Program C ditulis dengan menggunakan sebagian dari karakter ASCII, yaitu:

1. Huruf besar → A B C sampai Z
2. Huruf kecil → a b c sampai z
3. Angka → 0 1 2 sampai 9
4. Karakter khusus

!	*	+	\	"	<
#	(=		{	>
%)	~	;	}	/
^	-	[:	,	?
&	_]	'	.	Spasi

Tabel 1. Karakter Khusus Pembentuk Program C

IDENTIFIER

Identifier adalah nama berbagai elemen program, nama variabel, nama function, nama tipe data, dan lain lain. Sebuah identifier harus diawali huruf atau garis bawah dan diikuti huruf atau garis bawah atau angka. Huruf besar dianggap berbeda dengan huruf kecil. Panjang sebuah identifier tidak dibatasi tetapi hanya 32 karakter pertama yang dikenal.

Keyword

Keyword adalah identifier yang telah didefinisikan oleh bahasa C. Keyword mempunyai arti dan pemakaian tertentu. Keyword pada bahasa C adalah reserved word artinya dicadangkan dan tidak boleh digunakan untuk keperluan lain. Semua keyword bahasa C ditulis dalam format huruf kecil.

- ❖ Contoh identifier yang benar:

x luas temperatur
unsada _jumlah tarif

- ❖ Contoh identifier yang salah:

3m diawali angka
luas" ada karakter "
total-biaya ada karakter -
indeks prestasi terdapat spasi

auto	extern	sizeof
break	float	static
case	for	struct
char	goto	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	register	void
double	return	volatile
else	short	while
enum	signed	

Tabel 2. keyword bahasa C

Standart Identifier

Standart identifier adalah identifier yang telah diberi makna tertentu oleh compiler bahasa C tetapi identifier ini tidak bersifat reserved (cadangan) sehingga masih bisa didefinisikan kembali oleh pemogram. Dengan definisi ulang sebuah standard identifier kehilangan kegunaan aslinya di dalam program tersebut.

Sebagai contoh clrscr adalah sebuah standard identifier (standard function) yang berguna untuk membersihkan layar tampilan. clrscr dapat didefinisikan ulang, misalnya menjadi sebuah tempat penampungan data bilangan bulat. Maka mulai dari perubahan itu clrscr berarti tempat menampung data, bukan instruksi untuk menghapus layar.

Program 1. clrscr () sebagai standard Identifier.

```
#include <stdio.h>
#include <conio.>
void main ( ) {
    clrscr ( );
    printf("Halo Bahasa \n ");
}
```

Program 2.clrscr () didefinisikan kembali

```
#include <stdio.h>
#include <conio.h>
void main ( ) {
    int clrscr = 5;
    clrscr ( ); // instruksi ini menimbulkan kesalahan
    printf("Nilai clrscr adalah %d", clrscr);
}
```

Sebaiknya tidak mendefinisikan kembali standar identifier kecuali untuk kegunaan yang sama namun dengan kemampuan yang lebih.

TIPE DATA (DATA TYPE)

Komputer digunakan untuk mengolah data elektronik. Data yang ingin diolah program C harus memiliki jenis (tipe) yang jelas. Tipe data dasar yang dikenal bahasa C sebagai berikut

Type Data Dasar	Contoh Data	Kata Kunci
Integer	-5 0 1	int
Floating Point	3.14 12.90	float
Double-precision FP	12345678900000000000	double
Character	'a' '1' 'H'	char
Void		void

Tabel 3. Tipe Data Dasar

Integer menyatakan bilangan bulat positif dan negatif termasuk nol. Floating point menyatakan bilangan pecahan. Double precision Floating point menyatakan bilangan pecahan yang sangat besar atau sangat kecil. Character menyatakan satu karakter ASCII

Jenis	Penulisan	Rentang Nilai
Charcter	unsigned char	0 s/d 255
	char	-128 s/d 127
Integer	unsigned int	0 s/d 65535
	int	-32768 s/d 32767
	Short int	-128 s/d 127
	unsigned long	0 s/d 4294967295
	long	-2147483648 s/d 2147483648
Floating Point	float	3.4E-38 s/d 3.4E+38
	double	1.7E-308 s/d 1.7E+308
	long double	3.4E-4932 s/d 1.1E+4932

Tabel 4. Tipe Data dan Modifier

KONSTANTA

Konstanta ialah suatu nilai konstan, yaitu nilai yang tidak berubah dalam sebuah program. Konstanta yang dikenal bahasa C berupa:

1. integer constant → -5 0 125 010 0xF1
2. floating-point constant → 3.14
3. character constant → 'C' '1' 'S'
4. string constant → 'UnSada'
5. escape sequence → \n \t \"

Integer constant adalah sebuah bilangan bulat. Integer constant dapat berupa bilangan basis 10 (desimal), basis 8 (oktal), atau basis 16 (heksadesimal). Konstanta oktal diawali dengan bilangan nol. Konstanta heksadesimal diawali dengan 0x atau 0X.

Floating point constant adalah sebuah bilangan pecah. Tanda desimal dalam bahasa C adalah titik. Antara angka ratusan dan ribuan atau antara angka ratus ribuan dengan angka jutaan tidak boleh diberi tanda pemisah.

Character constant adalah sebuah karakter ASCII. Karakter ini diapit tanda petik tunggal. String constant adalah kumpulan karakter ASCII. Kumpulan karakter ini diapit tanda petik ganda.

Escape sequence adalah karakter yang diawali dengan backslash (garis miring terbalik). Setiap escape sequence mempunyai makna tertentu.

Escape Sequence	Nilai ASCII	Karakter	Arti
\0	0x00	NULL	Karakter dengan nilai ASCII = 0
\a	0x07	BEL	Bunyi speaker
\b	0x08	BS	Backspace (mundur kekiri)
\f	0x0C	FF	FormFeed (ganti halaman)
\n	0x0A	LF	lineFeed (ganti baris)
\r	0x0D	CR	Carriage return (ke awal baris)
\t	0x09	HT	Tabulator horizontal

\v	0x0B	VT	Tabulator vertikal
\\	0x5C	\	Karakter miring kiri
\'	0x27	'	Karakter petik tunggal
\"	0x22	"	Karakter petik ganda
\?	0x3F	?	Karakter tanda tanya

Tabel 5.Escape Sequence

Sebuah karakter pada tabel ASCII dapat digunakan dalam character constant, string constant atau integer constant.

❖ Contoh berbagai constant:

'H' adalah sebuah character constant

"H" adalah sebuah string constant

1 adalah sebuah integer constant

'1' adalah sebuah character constant

Const modifier adalah pemberian nama kepada suatu nilai konstanta, hal ini digunakan untuk mencegah kesalahan ketikan yang berulang.

❖ Contoh deklarasi const modifier:

Const float pi = 3.1415926

VARIABLE

Variabel adalah tempat menampung data. Jenis data yang dapat ditampung suatu variabel harus ditentukan (harus disertakan tipe datanya). Penamaan variabel mengikuti aturan penamaan identifier.

❖ Contoh mendefinisikan variabel

int i, j = 0, k;

float f1 = 10.0;

i adalah tempat menampung sebuah integer, j adalah tempat menampung sebuah integer dan diberi nilai awal nol, f1 adalah tempat penampung sebuah bilangan pecah dan diberi nilai awal sepuluh.

Pendefinisian variabel yang setipe dapat digabung atau dipisah.

Int i, j = 0, k;

Sama dengan

Int i;

Int = 0;

Int k;

OPERASI DAN OPERATOR

Data sejenis dapat dilakukan pengolahan, misalnya sebuah integer dapat ditambahkan dengan sebuah integer, sebuah float dapat dibagi oleh sebuah float, pengolahan ini disebut **operasi**. Setiap jenis tipe data mempunyai jenis – jenis operasinya masing – masing. Simbol untuk jenis operasi disebut **operator** sedangkan data (variabel atau konstanta) yang dioperasikan disebut **operan**.

Berdasarkan jumlah operan maka operator dibagi atas:

1. Unary operator, menggunakan satu operan
2. Binary operator, menggunakan dua operan
3. Ternary operator, menggunakan tiga operan

Dalam bahasa C memiliki banyak sekali operator, hal ini merupakan salah satu kekuatan bahasa pemrograman ini untuk menyelesaikan berbagai persoalan. Berdasarkan jenis operasinya, operator dalam bahasa C dapat dikelompokkan sebagai berikut:

1. Operasi aritmatika
2. Operasi relasi
3. Operasi logika
4. Operasi bitwise

Operator Aritmatika

Operator aritmatika adalah operator yang digunakan untuk pengolahan aritmatika, seperti penjumlahan dua bilangan (bulat atau pecahan), pengurangan dua bilangan (bulat atau pecahan), perkalian dua bilangan (bulat atau pecahan), pembagian dua bilangan (bulat atau pecahan), sisa bagi dua bilangan bulat, penambahan variabel dengan satu, dan pengurangan nilai variabel dengan satu.

Simbol	Fungsi	Contoh
+	Penjumlahan	$Y + 6$
-	Pengurangan	2017 – tahunlahir
*	Perkalian	$3.14 * \text{diameter}$
/	Pembagian	Jumlahdetik / 3600
%	Modulo	$n \% 16$
++	Increment	$X++ \text{ } ++X$
--	Decrement	$Z-- \text{ } --Z$

Tabel 6.Operator Aritmatika

X++ dan ++X menyatakan dua operasi yang berbeda. Pada X++ variable X diproses (misalnya nilainya disalin pada variabel lain) setelah itu nilai variabel X ditambah satu. Pada ++X, nilai X ditambah satu sebelum X diproses selanjutnya.

Operator Relasi

Operator Relasi adalah operator yang digunakan untuk membandingkan dua nilai sejenis. Kedua nilai tersebut dapat berupa konstanta atau variabel. Jika hasil pembandingan benar maka akan dikembalikan angka 1, jika salah akan dikembalikan nilai nol.

Operator Logika

Operator logika adalah operator yang berkaitan dengan operasi logika, seperti: negasi (ingkaran), konjungsi (dan), dan disjungsi (atau)

Simbol	Fungsi	Contoh
==	Sama Dengan	grade== 'a'
!=	Tidak Sama Dengan	pilih !='x'
<	Lebih kecil dari	nilai < 70
>	Lebih besar dari	nilai > 80
<=	Lebih kecil atau sama dengan	Usia <= 17
>=	Lebih besar atau sama dengan	Ip >= 2.0

Tabel 7. Operasi Relasi

Simbol	Fungsi	Contoh
!	NOT	!b
&&	AND	(a>70) && (a<100)
	OR	(n<0) (n>100)

Tabel 8. Operasi Logika

A	B	!A	A&&B	A B
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Tabel 9. Kebenaran Dua Variabel

Operator Bitwise

Berbeda dengan operator logika yang memperlakukan operan – operannya sebagai sebuah nilai tunggal, operator bitwise memperlakukan operan – operannya sebagai sebuah kuantitas yang terdiri dari bit-bit

Simbol	Fungsi	Contoh	Penjelasan
&	AND	A & B	AND bitwise dari A dan B
	OR	A B	OR bitwise dari A dan B
^	XOR	A ^ B	Bernilai 1 jika bit – bit A dan B berbeda
~	Complement 1	~B	Mengubah bit 1 menjadi 0 dan sebaliknya
>>	Shift Right	A >> 3	A digeser ke kanan sebanyak 3 posisi bit
<<	Shift left	B << 2	B digeser ke kiri sebanyak 2 posisi bit

Tabel 10. Operasi Bitwise

Operator Koma

Operator koma digunakan untuk memisahkan sederetan nama variabel dalam sebuah deklarasi, memisahkan argumen fungsi, menyatukan dua ekspresi menjadi sebuah pernyataan, dan memungkinkan pemberian lebih dari satu ekspresi pada inisialisasi nilai awal dan langkah penaikan nilai suatu struktur pengulangan.

Operator Bertingkat

Operator bertingkat (nested operator) merupakan beberapa operator yang dikenakan di dalam sebuah ekspresi. Masing – masing operator di dalam ekspresi tersebut memberikan sebuah nilai yang kemudian dapat digunakan oleh ekspresi yang lebih besar. Sebagai contoh:

❖ While { (c = getchar ()) != '\027' }

Operator penugasan pada c = getchar () diatas berada di dalam ekspresi lain yang memuat operator relasi !=. Urutan pelaksanaan ekspresi di atas adalah sebagai berikut:

- 1) Melakukan pemanggilan terhadap fungsi standart getchar (), return value-nya adalah nilai karakter
- 2) Menyimpan hasilnya pada variabel c, dan
- 3) Membandingkan isi variabel c dan '\027', yang akan memberikan hasil berupa niali true atau false

Dalam menggunakan operator bertingkat, sebaiknya benar-benar memperhatikan penggunaan tanda kurung, karena operator penugasan memiliki prsedensi yang lebih rendah dibandingkan dengan operator relasi. Contoh penggunaan operator bertingkat lainnya adalah:

a = b = c = 0;

dalam contoh ini, bahasa pemrograman C secara otomatis akan menerjemahkan dari kanan ke kiri.

Operator bersyarat

Operator bersyarat merupakan satu-satunya operator yang bersifat triadic (ternary operator) yang membutuhkan tiga buah operator dan dikenal juga sebagai ekspresi bersyarat (conditional expression). Ketiga buah operan yang diperlukan terdiri dari satu ekspresi yang akan diuji dan dua ekspresi pilihan. Bentuk umumnya adalah sebagai berikut;

Operator bersyarat akan menghasilkan satu dari dua buah pilihan. Misalnya: $a = c ? x : y$; Pertama-tama c dievaluasi nilainya, jika c tidak sama dengan nol, maka x akan dievaluasi dan hasilnya akan diberikan kepada a sesuai aturan perubahan tipe data yang berlaku dengan aturan perubahan tipe data yang berlaku.

❖ Contoh pemakaian ekspresi bersyarat

Minimal $= x < y ? x : y;$

Maksimal $= x > y ? x : y;$

absX $= x < 0 ? -x : x;$

Keterangan:

Apabila nilai variabel x lebih kecil dari pada nilai variabel y maka variabel minimal bernilai sebesar nilai variabel x . Sebaliknya apabila nilai variabel y lebih kecil daripada nilai variabel x maka variabel minimal akan bernilai sebesar nilai variabel y . Apabila variabel x bernilai lebih kecil daripada nol (x bernilai negatif) maka nilai ini dikali dengan minus satu sehingga menjadi positif dan disalin ke variabel $absx$.

Operator Cast

Dalam sebuah program, pengkonversian tipe data secara otomatis (implicit type conversion) sedapat mungkin hendaklah dihindarkan, karena dapat menimbulkan hal-hal yang tidak diinginkan, misalnya terjadi pemotongan nilai (truncation). Metode untuk melakukan pengkonversian tipe data, secara eksplisit disebut dengan istilah type casting, menggunakan operator berupa sepasang tanda kurung biasa.

❖ Contoh tanpa dan dengan type casting

Usia = 32.3 + 1.5

Usia = (int) 32.3 + (int)1.5;

Operator Penugasan (Assignment)

Assignment adalah pemberian suatu nilai kepada variabel. Assignment menggunakan simbol sama dengan (=). Operan di sebelah kiri operator harus berupa variabel. Operan di sebelah kanan harus berupa ekspresi.

Luas = 3.14 * radius * radius;

Sebelum dikerjakan, luas = 25 , radius = 10

Setelah dikerjakan, luas = 314, radius = 10

Jika operan kiri sama dengan operan kiri pada ekspresi sebelah kanan maka penulisan assignment dapat disingkat.

Assigment	Dapat diganti dengan
a = a+ b;	a += b;
a = a – b;	a -= b;
a = a* b;	a*= b;
a = a / b;	a /= b;
a = a % b;	a %=b;
a = a << b;	a <<= b;
a = >> b;	a >>=b;
a = a & b;	a &=b;
a = a b;	a =b;
a = a ^ b;	a ^=b;

Tabel 11.Penyingkatan Penulisan Assigment

Presedensi (Precedence) dan Asosiativitas (Associativity)

Presedensi operator menunjukkan tingkat atau level operator, misalnya: operator * memiliki presedensi yang lebih tinggi dibandingkan dengan operator +. Operator dengan presedensi lebih tinggi akan dikerjakan terlebih dahulu. Asosiativitas menjelaskan cara pengerjaan sebuah ekspresi jika terdapat operator dengan tingkat/level yang sama, misalnya pengerjaan dilakukan dari kiri ke kanan atau dari kanan ke kiri.

Operator	Asosiativitas	Presedensi
() [] -> . “	Kiri ke Kanan	Tertinggi
! ~ ++ -- - & * (cast) sizeof	Kanan ke Kiri	
*/ %	Kiri ke kanan	
+ -	Kiri ke Kanan	
<< >>	Kiri ke Kanan	
< <= > >=	Kiri ke Kanan	
== !=	Kiri ke Kanan	
&	Kiri ke Kanan	
^	Kiri Ke kanan	
	Kiri ke kanan	
&&	Kiri ke kanan	
	Kiri ke kanan	
?:	Kiri ke Kanan	

= += -= *= /=	Kanan ke kiri	
,	Kiri ke kanan	Terendah

Tabel 12. Presedensi dan Asosiativitas operator

❖ Contoh Presedensi dan asosiativitas

a = b = c;
 /* b = c dikerjakan lebih dahulu , baru kemudian a = b */
 x = 2 * 6 * 9;
 /* 2 * 6 dikerjakan terlebih dahulu, hasilnya dikalikan dengan 9, karena prioritas '=' lebih rendah dari '*', maka operasi penugasan dilakukan terakhir */

EKSPRESI (EXPRESSION)

Ekspresi adalah segala sesuatu yang bila dievaluasi akan menghasilkan suatu nilai.

1. Sebuah konstanta adalah sebuah ekspresi
2. Sebuah variabel adalah sebuah ekspresi
3. Sebuah ekspresi yang dioperasikan dengan ekspresi lain adalah sebuah ekspresi.
4. Pemanggilan terhadap sebuah function adalah sebuah ekspresi

❖ Contoh ekspresi;

3.14 diameter
 3.15 * diameter
 gotoxy (10,5)
 (int) 32.3 + (int)1.5;

STRUKTUR PROGRAM C

Program C ditulis dalam modul-modul. Sebuah modul adalah sekumpulan instruksi (statement) untuk melaksanakan suatu tugas tertentu. Jenis modul yang terdapat dalam bahasa C adalah function. Sebuah program C paling sedikit harus mempunyai sebuah function yaitu bernama main(). Proses eksekusi program selalu dimulai dari function main().

❖ Contoh:

```
return_type nama_function (daftar_parameter) {
    deklarasi_variabel_lokal;
    instruksi_1;
    instruksi_2;
    ....
    instruksi_n;
    return (value);
}
```

Return type menyatakan tipe data yang akan dikembalikan function, Jika function tidak mengembalikan nilai maka return type ditulis void. Pada function yang return type-nya void tidak perlu instruksi return (value) karena tidak ada nilai yang akan dikembalikan kepada

modul pemanggil. Daftar parameter digunakan untuk menampung data masukan (yang dikirim ke function ini) untuk diolah. Jika sebuah function tidak memerlukan data masukan maka daftar parameter tidak perlu dicantumkan atau ditulis void tetapi kedua kurung tetap diperlukan.

❖ Contoh program berfunction main()

```
#include <stdio.h>
void main( ) {
    int n, i;
    printf("Bilangan ? ");
    scanf("%d", &n);
    for(i = 0; i < n; i++)
        printf("*");
}
```

Komentar

Pemberian komentar merupakan salah satu cara untuk memberikan dokumentasi pada program yang sangat berguna untuk membantu memperjelas alur logika penyusunan program. Dengan pemberian komentar secukupnya, program yang disusun akan lebih mudah dipahami sendiri atau oleh orang lain. Karena tujuannya hanya sebagai dokumentasi, komentar-komentar yang dituliskan pada program tidak diproses oleh compiler. Baris-baris karakter yang dimaksudkan sebagai komentar akan diabaikan oleh compiler pada waktu program dikompilasi.

Komentar diawali dengan simbol dua karakter yang terdiri dari garis miring dan asterik (/*) dan diakhiri dengan asterik dan garis miring (*). Karakter-karakter penunjuk komentar ini dapat diletakkan di mana saja di dalam program, dengan syarat setiap /* harus ditutup dengan */. Pada awal program, komentar diberikan dengan tujuan untuk menjelaskan apa tujuan program. Sedangkan pada bagian-bagian program komentar digunakan untuk memperjelas logika program.

Untuk komentar yang panjangnya satu baris ditandai dengan //. Semua karakter di sebelah kanan // diperlakukan sedang komentar.

Program 3. Tabel ASCII

```
#include <stdio.h>
#include <conio.h>
void main (void ) {
    int i, brs = 3, klm = 1, //cetak mulai dari baris 3 kolom 1
    clrscr ( );
    printf("Tabel ASCII");
    /* Karakter ASCII dari 169 sampai dengan 222 */
    for (i = 169; i <= 222; i++, brs++) {
        gotoxy(klm,brs);
        printf("%d = %c", i ,i);
        /* cetak ke bawah 12 baris, pindah 10 kolom ke kanan */
        brs = 2; klm += 10;
    }
}
```

SOAL LATIHAN

1. Apa yang dimaksud dengan identifier?
2. Bagaimana penulisan suatu identifier?
3. Apakah huruf besar dianggap sama dengan huruf kecil pada identifier?
4. Berapakah panjang suatu identifier?
5. Apa yang dimaksud dengan keyword?
6. Sebutkan 5 keyword yang digunakan oleh bahasa C?
7. Apa yang dimaksud dengan standart identifier?
8. Apa yang dimaksud dengan tipe data?
9. Sebutkan jenis-jenis tipe data?
10. Apa yang dimaksud dengan konstanta?
11. Apa yang dimaksud dengan escape sequence?
12. Apa perbedaan antara character konstant dan string konstant?
13. Apa yang dimaksud dengan variabel?
14. Jelaskan perbedaan antara Operasi, Operator dan Operan?
15. Sebutkan jenis-jenis Operator?

BAB III INPUT OUTPUT

Program yang ditulis akan mengolah data yang di input menjadi keluaran informasi yang di output. Untuk itu maka setiap bahasa pemrograman menyediakan instruksi untuk mendapatkan masukan baik melalui pengetikan keyboard atau dari file dan instruksi untuk menampilkan hasil proses ke layar monitor, tulis ke file, atau keluaran tercetak ke printer.

INSTRUKSI KELUARAN

Instruksi keluaran digunakan untuk menampilkan hasil proses, dapat berupa keluaran ke layar monitor, dicetak pada printer, atau ditulis ke file. Instruksi keluaran ke file akan dibahas pada bab file data.

❖ **int putchar(int c);**

Function ini menampilkan karakter ASCII untuk c ke layar monitor tanpa memindahkan cursor ke baris berikutnya.

❖ **int puts(const char *s);**

Instruksi ini merupakan macro yang didefinisikan untuk menjalankan function putchar().

❖ **int printf(const char *format,....);**

Function ini menampilkan string s ke standart output stream dan menambahkan karakter newline (memindahkan cursor ke baris berikutnya).

❖ **int scanf(const char *format,argument,...);**

Function ini menampilkan sejumlah keluaran dengan format tertentu, setiap argument sesuai format masing-masing.

Bentuk format

%[flags] [width] [.prec] [F|N|h|1|L type

flags kumpulan flag characters

width panjang tampilan

.prec presisi (jumlah angka setelah titik)

F|N|h|1|L input-size modifier

type karakter konversi tipe data

Tabel 13.Flag Characters

Karakter	Kegunaan
-	Rata kiri
+	Sertakan tanda + atau – untuk data numerik
blank	Jika data bernilai positif maka keluaran diawali spasi. Jika data bernilai negatif maka keluaran diawali -

Tabel 14.Karakter Konversi Tipe Data

Karakter	Jenis Keluaran
d	Signed decimal integer
i	Signed decimal integer
o	Unsigned octal integer
u	Unsigned decimal integer
x	Unsigned hexadecimal integer dengan a, b, c, d, e, f
X	Unsigned hexadecimal integer dengan A, B, C, D, E, F
f	Signed floating-point number dalam bentuk [-]dddd.dddd
e	Signed floating-point number dalam bentuk [-]d.dddd atau e[+ -]ddd
E	Seperti e tetapi dengan huruf E
c	Karakter
s	Sejumlah karakter

Program 4. Pemakaian `putch()`, `putchar()`, dan `puts()`

```
#include <stdio.h>
#include <conio.h>
void main ( ) {
    int bilangan;
    char huruf;
    char kalimat[50];

    clrscr( );
    bilangan = 65;
    huruf = 'B';
    putch (bilangan);
    putch (huruf);
    putchar (67);
    putchar ('D');
    strcpy (kalimat, "algoritma dan pemograman");
    puts (kalimat);
    puts ("*****");
}
```

Program 5. pemakaian `printf()` terhadap data karakter

```
#include <stdio.h>
#include <conio.h>
void main ( ) {
    char klm1[50] = "algoritma dan pemograman";
    char klm2[50] = "algo";
    char klm3[50] = "program";
    char huruf = 'h';

    clrscr( );
    printf("      1      2      3 \n");
    printf("12345678901234567890 \n");
    printf("%s \n", klm1);
    printf("%30s \n", klm1);
    printf("%20s \n", klm1);
    printf("%c \n", huruf);
    printf("\n");
    printf("%10s%10s \n", klm2, klm3);
    printf("%10s%10s \n", "web", "internet");
}
```

Statement input (instruksi masukan) adalah function atau macro yang digunakan untuk membaca data. Instruksi masukan yang berupa pembacaan dari file akan dibahas pada bab file data.

❖ **int `getchar`(void);**

Function ini mengembalikan karakter berikutnya dari standard input, berupa nilai ASCII-nya. Function ini tidak memerlukan input parameter karena bernilai void.

❖ **int `getch`(void);**

Function ini membaca satu karakter dari keyboard tetapi tidak menampilkan di layar

❖ **int `getche`(void);**

Function ini membaca satu karakter dari keyboard dan menampilkannya ke layar.

❖ **char *`gets`(char *s);**

Function ini mengembalikan sekumpulan karakter (yang diakhiri dengan karakter newline) dari standard input dan menyimpannya di variabel s. Karakter newline diganti dengan karakter null.

Program 6.Masukan Data Karakter

```
#include <stdio.h>
#include <conio.h>
void main ( ) {
    int a, b, c;
    char nama[50];
    clrscr( );
    printf("Nama anda ?"); gets(nama);
    printf("\nKetik satu huruf   : "); a = getche( );
    printf("\nKetik huruf kedua  : "); b = getch( );
    printf("\nKetik huruf ketiga : "); c = getchar( );
    printf("\nHalo %s ", nama);
    printf("\nKetikan = %c %c %c ", a, b, c,);
    printf("\nNilai ASCII-nya = %d %d %d", a, b, c);
}
```

❖ **int `scanf`(const char *format,(address,...));**

Function ini membaca sejumlah masukan berformat dari standard input stream. Data yang dimasukkan disesuaikan dengan formatnya dan disimpan di alamat variabel. Jumlah format harus sama dengan jumlah alamat.

Bentuk format

%[*] [width] [F|N] [h|l|L] type_character

***** assignment-suppression character

width panjang maksimum

F|N pointer size modifier

h|l|L argument-type modifier

Karakter	Jenis Masukan
d atau D	Decimal integer
o atau O	Octal integer
i atau I	Decimal/octal/hexadecimal integer
u atau U	Unsigned decimal integer
x atau X	Hexadecimal integer
e atau E	Floating point
f	Floating point
g atau G	Floating point
s	Character string
c	Character

Tabel 15.Type Character pada instruksi scanf()

Program 7.Nilai Akhir Mata Kuliah

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

void main ( ) {
    int nT, nUTS, nUAS;
    float nA;
    clrscr( );
    printf("Nilai Tugas, UTS dan, UAS? ");
    scanf("%d %d %d", &nT, &nUTS, &nUAS);
    nA = nT * 0.2 + nUTS * 0.3 + nUAS * 0.5;
    nA = floor(nA + 0.5); // Pembulatan keatas
    printf("Nilai akhir = %.0f", nA);
}
```

Pada saat instruksi `scanf("%d %d %d",&nT, &nUTS, &nUAS);` dikerjakan maka komputer menunggu data masukan berupa tiga buah integer. Bila hanya diberikan satu integer langsung di-enter maka program akan berhenti menunggu data masukan integer kedua dan ketiga untuk variabel `nUTS` dan `nUAS`.

Pada pemakaian `scanf()` untuk beberapa input sekaligus tidak harus bertipe sama. Instruksi `scanf("%d %d %d",&nT, &nUTS, &nUAS);` dapat dipecah menjadi tiga instruksi terpisah

```
scanf("%d", &nT);           //input nilai tugas
scanf("(%d", &nUTS);        //input nilai ujian tengah semester
scanf("(%d", &nUAS);        //input nilai ujian akhir semester
```

Hal penting yang harus diperhatikan adalah bahwa penampung masukan harus berupa address variabel, jika tanda & lupa disertakan maka data yang diketikkan tidak disimpan di alamat variabel melainkan di tempat lain.

Program 8. Luas Persegi Panjang

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

void main ( ) {
    int panjang, lebar, luas;

    clrscr( );
    printf("Panjang ? "); scanf("%d", &panjang);
    printf("Lebar ? "); scanf("%d", &lebar);
    luas = panjang * lebar;
    printf("Luas = %d", luas);
}
```

Pada nilai panjang dan lebar yang diketikkan tidak disimpan di alamat kedua variabel sehingga nilai luas yang ditampilkan bukan nilai luas hasil perkalian panjang dan lebar.

Instruksi scanf() dapat digunakan untuk meminta masukan string demikian juga dengan gets(). Penandaan ujung string pada scanf() adalah white space (spasi, tab, newline) sedangkan penandaan ujung string pada gets() adalah newline, karena itu jika masukan yang diinginkan lebih dari satu kata maka gunakan gets().

STANDARD I/O DAN REDIRECTION

Pada pembahasan sebelumnya terdapat istilah standard input dan standard output. Semua instruksi masukan yang menerima masukan melalui standard input yaitu getchar(), gets(), dan scanf() akan membaca masukan dari media input yang digunakan, Demikian juga dengan semua instruksi keluaran yang menampilkan hasil ke standard input output yaitu puts() dan printf() akan menampilkan hasil ke media output yang digunakan.

Program 9. Tabel Perkalian

```
#include <stdio.h>

void main ( ) {
    int bil, mulai, sampai, i;

    scanf("%d %d %d", &bil, &mulai, &sampai);
    for (i = mulai; i <= sampai; i++)
        printf("%d * %d = %d\n", bil, i, bil * i);
}
```

Program diatas menampilkan tabel perkalian suatu bilangan. Program ini menggunakan instruksi masukan dan keluaran standard I/O sehingga bisa kita atur masukan dan keluarannya baik melalui pengetikan keyboard dan penampilan ke layar monitor atau dengan redirection (pengarahan). Dengan redirection kita bisa mengarahkan data masukkan dari file dan mengarahkan hasil proses ke file.

PENEMPATAN KURSOR

Layar monitor komputer dalam modus teks berukuran 25 baris dan 80 kolom. Posisi baris pertama kolom berada pada sudut kiri atas. Beberapa compiler menyediakan perpustakaan fungsi untuk mengatur cursor dan layar. Pada compiler Borland misalnya function-function yang berhubungan dengan pengaturan cursor dan penghapusan layar didefinisikan pada header file conio.h sehingga untuk menggunakan function tersebut maka header file conio.h harus di-include.

❖ void **gotoxy** (int x, int y);

Function ini memindahkan cursor ke kolom x baris y.

❖ int **wherex** (void);

Function ini mengembalikan posisi kolom cursor.

❖ int **wherey** (void);

Function ini mengembalikan posisi baris cursor.

❖ void **window** (int left, int top, int right, int bottom);

Function ini mendefinisikan sebuah window berdasarkan koordinat kiri atas dan kanan bawah.

Program 10. Pengaturan Cursor

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

void main ( ) {
    char klm[50];
    int i;

    clrscr( );
    printf("kalimat? "); gets(klm);
    gotoxy(30,3); printf("%s", klm);
    gotoxy(wherex( )-15, wherey()+1); printf("%s", klm);
    gotoxy(wherex( )-15, wherey()+1); printf("%s", klm);
    gotoxy(wherex( )-15, wherey()+1); printf("%s", klm);
}
```

PEMBERSIHAN LAYAR

Pada penulisan program adakalanya layar perlu dihapus, baik sebagian atau seluruhnya

❖ void **clrscr** (void);

Function ini membersihkan seluruh layar window dan memindahkan cursor ke baris 1 kolom 1 (kiri atas layar).

❖ void **clreol** (void);

Function ini membersihkan layar mulai dari posisi cursor hingga kolom terakhir, posisi cursor tidak berubah.

Program 11.penghapusan layer

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

void main ( ) {
    clrscr( );
    puts("*****");
    puts("*****");
    puts("*****");
    puts("*****");
    getch();
    gotoxy(5,1);clreol();
    gotoxy(10,2);clreol();
    gotoxy(15,3);clreol();
    getch(); clrscr();
}
```

SOAL LATIHAN

1. Sebutkan instruksi untuk keluaran / output?
2. Sebutkan instruksi untuk masukkan / input ?
3. Apa fungsi dari putch?
4. Apa fungsi dari putchar?
5. Apa fungsi dari puts?
6. Apa fungsi dari printf?
7. Apa perbedaan antara putch dan putchar?
8. Apa fungsi dari scanf?
9. Apa fungsi dari getche?
10. Apa fungsi dari getch?
11. Apa fungsi dari getchar?
12. Apa fungsi dari gets?
13. Apa fungsi dari gotoxy?
14. Apa fungsi dari wherex?
15. Apa fungsi dari wherey?

BAB IV STRUKTUR KENDALI PROSES

Dalam proses pengolahan data adakalanya perlu memilih satu dari sekumpulan alternatif proses berdasarkan kriteria tertentu. Untuk itu disediakan instruksi if dan switch untuk melakukan proses seleksi (selection). Hal lain yang sering dilakukan adalah proses pengulangan. Untuk itu disediakan juga instruksi for, while, dan do – while, untuk mengolah proses pengulangan (repetition, iteration). Dalam komputer pengulangan disebut juga looping.

STRUKTUR KENDALI PEMILIHAN

Struktur kendali pemilihan digunakan untuk memilih instruksi-instruksi yang akan dikerjakan berdasarkan hasil pemeriksaan suatu kondisi. Bahasa C menyediakan dua instruksi pemilihan if dan switch.

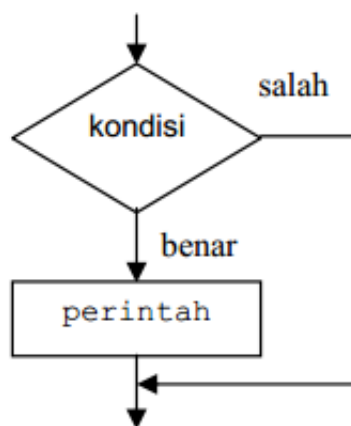
Instruksi if

Instruksi if digunakan untuk memilih jalur proses; melakukan atau tidak melakukan suatu proses (if), memilih satu dari dua proses (if else).

❖ If (expression) statement;

Expression dievaluasi. Jika hasilnya true maka statement dikerjakan. Jika hasilnya false maka statement tidak dikerjakan. Yang dimaksud dengan statement di sini dapat berupa satu instruksi tunggal atau sejumlah instruksi (compound statement) yang dalam bahasa C ditandai dengan { dan }.

Dengan flowchart instruksi if digambarkan sebagai:



gambar 2. Alur logika if

Program 12. Grade nilai

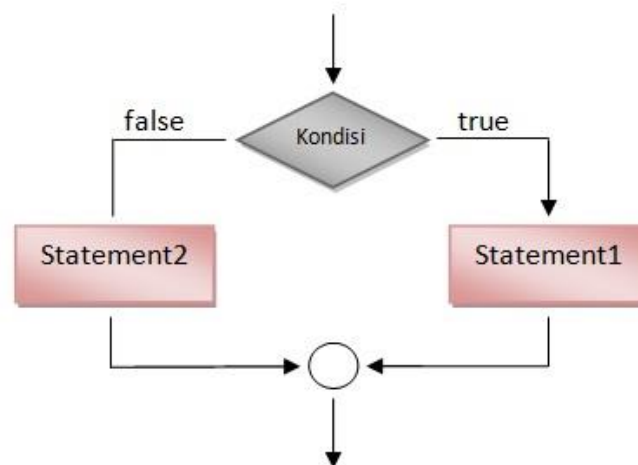
```
#include<stdio.h>
#include<conio.h>
void main( ) {
    Int nA;
    Char grade;
    Clrscr( );
    Printf("Nilai Akhir ? "); scanf("%d", &nA);
    If (nA > 84) grade = 'A';
    If (nA >74 && nA < 84) grade = 'B';
    If (nA > 64 && nA < 75) grade = 'C';
    If (nA > 49 && nA < 65) grade = 'D';
    If (nA < 50) grade = 'E';
    Printf ("grade = %c", grade);
}
```

Klausa else

❖ If (expression) statement1; else statement2;

Expression dievaluasi. Jika hasilnya true maka statement1 dikerjakan. Jika false maka statement2 dikerjakan. Statement1 dan statement2 dapat berupa instruksi tunggal atau beberapa instruksi yang diungkap { dan }. Expression dapat berupa ekspresi tunggal atau lebih.

Dengan flowchart instruksi if else digambarkan sebagai:



gambar 3. Alur logika if else

Statement yang dikerjakan saat true atau false dapat juga berupa instruksi if lainnya. Susunan seperti ini membentuk nested if.

Program 13.Evaluasi Studi

```
#include<stdio.h>
#include<conio.h>
void main( ) {
    int sksk;
    float ipk;
    clrscr( );
    printf("sks kumulatif ?"); scanf("%d", &sksk);
    printf("IP kumulatif ? "); scanf("%f", &ipk);
    if ((sksk >=20) && (ipk >= 2.00)) printf("calon sarjana");
    else printf("calon drop out");
}
```

Program 14.Uji Bilangan

```
#include<stdio.h>
#include<conio.h>
void main( ) {
    int bil;
    clrscr ( );
    printf("Bilangan ? ");
    scanf("%d", &bil);
    if (bil == 0) printf("Bilangan nol");
    else {
        if bil %2 == 0 ) printf("Bilangan genap");
        else printf ("Bilangan ganjil ");
        if (bil > 0) printf ("positif");
        else printf ("negatif");
    }
}
```

Instruksi switch

Instruksi ini digunakan untuk memilih satu dari sejumlah alternatif

❖ **Switch** (expression) {
 case constant_1:statement_1;
 case constant_2:statement_2;

 case constant_n:statement_n;
 (default:statement_x;)
}

Expression dievaluasi. Jika hasilnya constant_1 maka statement_1 dikerjakan, jika hasilnya constant_2 maka statement_2 dikerjakan. Jika hasil evaluasi bukan constant_1 sampai constant_n maka statement_x dikerjakan. Klausula default bersifat opsional.

Program 15. Kode Pos dan wilayah

```
#include<stdio.h>
#include<conio.h>
void main( ) {
    long kdpos;
    clrscr( );
    printf("kode pos ? "); scanf("%ld", &kdpos);
    printf("wilayah = ");
    if (kdpos >= 1000 && kdpos <= 14999) printf("Jakarta");
    switch (kdpos / 1000) {
        case 10: printf("Pusat"); break;
        case 11: printf("Barat"); break;
        case 12: printf("Selatan"); break;
        case 13: printf("Timur"); break;
        case 14: printf("Utara"); break;
        case 15: printf("Tangerang"); break;
        case 16: printf("Bogor"); break;
        case 17: printf("Bekasi"); break;
        default: printf("Luar jabodetabek");
    }
}
```

Instruksi break digunakan untuk memindahkan proses ke akhir switch. Apabila statement break tidak disertakan maka statement-statement case lainnya akan dikerjakan tanpa memeriksa kebenaran case tersebut.

Program 16. Case tanpa break

```
#include<stdio.h>
#include<conio.h>
void main( ) {
    char simbol;
    clrscr( );
    printf("ketik operator tunggal: ");
    printf("wilayah = ");
    switch (simbol) {
        case '+': printf("Operator Penjumlahan \n");
        case '-': printf("Operator Pengurangan \n");
        case '*': printf("Operator Perkalian \n");
        case '/': printf("Operator Pembagian \n");
        case '%': printf("Operator modulus \n");
        case '>': printf("Operator Lebih besar \n");
        case '<': printf("Operator lebih kecil \n");
    }
}
```

Berdasarkan sifat pengabaian pemeriksaan case berikutnya setelah didapatkan case yang memberikan nilai true, maka beberapa case dapat digabung.

STRUKTUR KENDALI PENGULANGAN

Struktur kendali pengulangan digunakan untuk melaksanakan satu atau beberapa instruksi secara berulang-ulang. Frekuensi pengulangan dapat ditentukan dalam program atau ditentukan saat program dijalankan. Meskipun dalam pengulangan ini instruksi yang sama dikerjakan berulang-ulang, namun nilai variabel pengendalian kondisi pengulangan harus berubah pada setiap iterasi pengulangan agar pengulangan bisa berakhir, Bahasa C menyediakan tiga instruksi pengulangan: for, while, dan do, while.

Instruksi for

Instruksi for akan melakukan sejumlah iterasi yang telah ditetapkan sebelumnya. Pada bentuk pengulangan ini terdapat tiga ekspresi yang digunakan untuk mengendalikan proses pengulangan. Ketiga ekspresi ini bersifat opsional.

❖ For ((expr1);(expr2);(expr3))statement;

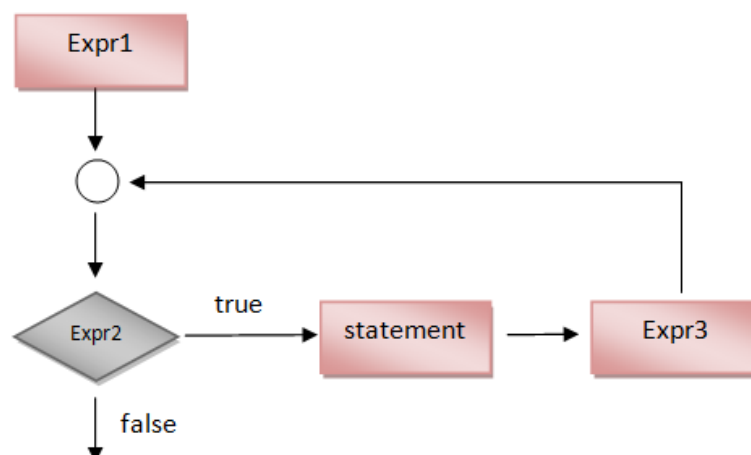
expr1 untuk inialisasi, berupa satu atau beberapa instruksi yang dipisahkan koma

expr2 berupa ekspresi kondisional untuk menentukan kondisi pengulangan. Selama expr2 menghasilkan nilai true maka statement dikerjakan.

expr3 untuk menaikkan atau menurunkan nilai pengendalian (control variable)

Urutan proses intruksi for adalah sebagai berikut: expr1 dikerjakan, expr2 dievaluasi, jika true maka kerjakan statement, expr3 dikerjakan, expr2 dievaluasi, jika true maka kerjakan statement, expr3 dikerjakan dan seterusnya.

Dengan flowchart instruksi for dapat digambarkan sebagai berikut:



gambar 4. Alur Logika for

Program 17. Grafik sederhana

```
#include <stdio.h>
#include <conio.h>
void main( ){
    int i, j, tripel;
    clrscr( );
    for(i = 1; j <= 6; i++) {
        tripel = 3 * i;
        printf("3 * %2d = %3d", i , tripel);
        for(j= 0 ; j < tripel; j++) putchar('*');
        printf("\n");
    }
}
```

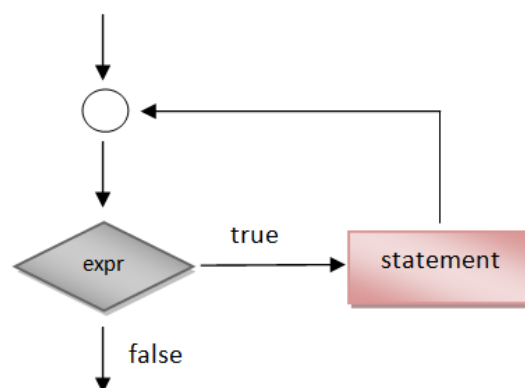
Instruksi while

Instruksi while melakukan proses pengulangan selama kondisi pengulangan masih benar.

❖ **while** (expression) statement;

Expression berupa ekspresi logika (boolean). Pengulangan terjadi selama hasil evaluasi ekspresi ini memberi nilai true. Mula-mula ekspresi dievaluasi. Jika hasilnya false maka instruksi while selesai. Jika true maka statement dikerjakan lalu ekspresi dievaluasi lagi. Jika true maka statement dikerjakan dan seterusnya.

Urutan proses instruksi while dapat digambarkan dengan flowchart berikut:



gambar 5.alur logika while

Program 18. Luas persegi panjang-persegi panjang

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

void main( ){
    float pj, lb, ls;
    char ulang = 'Y';

    while (toupper(ulang) == 'Y') {
        clrscr( );
        printf("PERSEGI PANJANG \n\n");
        printf("Panjang : "); scanf("%f", &pj);
        printf("Lebar : "); scanf("%f",&lb);
        ls = pj * lb;
        printf("*Luas = %.2f\n\n", ls);
        printf("Ulangi (Y/T)? ");
        fflush( );
        scanf("%c", &ulang);
    }
}
```

Instruksi do while

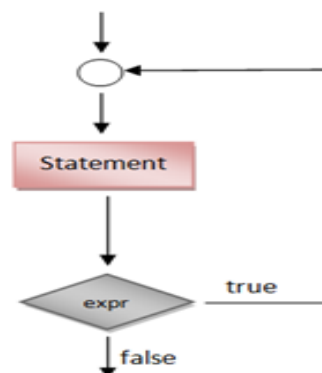
Instruksi do while akan melakukan pengujian terhadap ekspresi setelah melaksanakan instruksi yang terdapat dalam tubuh pengulangan.

❖ **do** Statement **while** (expression);

Statement dikerjakan. Expression dievaluasi, jika bernilai true maka statement dikerjakan. Expression dievaluasi, jika bernilai true maka statement dikerjakan, dan seterusnya.

Pada instruksi while statement mungkin tidak pernah dilaksanakan bila sejak awal evaluasi ekspresi menghasilkan nilai false. Hal ini terjadi karena evaluasi ekspresi dilakukan sebelum statement dilaksanakan. Pada instruksi do while, statement dikerjakan terlebih dahulu satu kali baru expression dievaluasi sehingga paling sedikit statement dikerjakan satu kali.

Dengan flowchart urutan proses instruksi do-while dapat digambarkan sebagai berikut:



gambar 6. Alur logika do while

Program 19. Pilih Y atau T

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

void main( ){
    char pilih;
    do {
        gotoxy (5,1);
        pilih = toupper (getch());
        putchar (pilih);
    }while (pilih !='Y' && pilih !='T');

    ....
}
```

Infinite Loop

Infinite loop adalah pengulangan tanpa akhir. Pengendalian berulang tidaknya instruksi for berada pada exp_2 dan exp-2 ini bersifat opsional. Sehingga jika ingin membentuk infinite loop dengan for maka kosongkan exp_2. Ekspresi yang bernilai bukan nol (positif) atau negatif () diinterpretasikan sebagai true, sehingga dapat digunakan sebagai kondisi pada instruksi while dan do while dalam membentuk infinite loop, juga untuk for.

❖

```
for ( ... ; ; ..... ) { .... }
for ( ..... ; 1 ; ..... ) { .... }
do { ..... } while (1);
while (1) { ..... }
```

break dan continue

break menyebabkan proses ke luar dari instruksi for, while, do, switch, continue menyebabkan proses mengabaikan instruksi selanjutnya dan melanjutkan proses iterasi berikutnya pada instruksi for, while, do while.

Pada proses yang berulang sekurangnya terdapat tiga cara untuk mengakhiri pengulangan:

1. Menanyai pemakai apakah masih mau mengulangi proses.
2. Meminta pemakai memasukkan nilai tertentu.
3. Meminta pemakai menekan tombol tertentu misalnya ESC.

Program kecuali kelipatan tiga

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

void main( ){
    int i, j;
    clrscr( );
    for(i=1; i<= 10; i++) {
        if (i % 3 == 0) continue;
        printf("%2d ", i);
        for (j = 0; j<i; j++) putchar('*');
        printf("\n");
    }
}
```

MENGUJI KEBENARAN PROGRAM (PROGRAM TESTING)

Program yang selesai dibuat perlu diuji kebenarannya. Pengujian program disebut program testing. Sebuah program disebut benar jika selalu menghasilkan keluaran sebagaimana mestinya. Kesalahan dapat terjadi karena algoritma program salah atau data yang dimasukkan tidak semestinya. Program yang baik mestinya dapat mengatasi kedua hal ini. Sebuah program yang baik tidak boleh menimbulkan run-time error, yaitu kesalahan yang timbul pada saat program dijalankan sehingga eksekusi program terhenti tiba-tiba dan kontrol dikembalikan kepada sistem operasi. Apabila terjadi kesalahan mestinya program berhenti secara mulus (gracefully ending). Ada berbagai metode pengujian program, di antaranya black box testing dan white box testing (ada yang menyebutnya glass box testing).

Pengujian terhadap program yang terdiri dari function-function dilakukan secara bertahap saat proses penulisan program sedang berlangsung (incremental coding), Jangan tunggu sampai seluruh program selesai baru diuji.

Metode Black Box

Pengujian black box ialah pengujian hasil keluaran tanpa menganalisis isi program. Pada metode ini program diuji dengan menggunakan:

1. Data yang mudah diperiksa,
2. Data yang bernilai wajar,
3. Data nilai batas, dan
4. Data ilegal.

Uji dengan Data yang Mudah Diperiksa

Pengujian kebenaran program dilakukan dengan membandingkan keluaran program dan hasil penghitungan secara manual. Jika data uji yang digunakan rumit tentunya akan mempersulit kerja.

Uji dengan Data yang Wajar

Yang dimaksud dengan data yang wajar adalah data yang diharapkan. Data masukkan yang wajar untuk program ini adalah dua buah bilangan bulat dengan bilangan pertama lebih kecil daripada bilangan kedua. Uji dengan dua bilangan ganjil, dua bilangan genap, bilangan ganjil dan bilangan genap, serta bilangan genap dan bilangan ganjil.

Uji dengan Nilai Batas

Selain melakukan pengujian dengan menggunakan nilai yang wajar, pengujian yang lengkap juga harus mencakup pengujian dengan menggunakan nilai ekstrim misalnya dengan nilai batas. Program di atas memberikan hasil yang kedua data masukan adalah bilangan ganjil yang sama tetapi program akan memberikan keluaran yang salah jika kedua data adalah bilangan genap yang sama.

Uji dengan Data Ilegal

Data ilegal ialah data yang sama sekali tidak bisa diterima, misalnya beda tipe. Pada program deret bilangan ganjil di atas, masukan yang diharapkan adalah dua buah bilangan bulat. Apabila data yang dimasukkan adalah huruf maka program akan memberikan respons lain.

Metode White Box

Pada metode white box alur logika program dipelajari untuk menentukan alternatif proses yang dapat diuji. Semua alternatif (jalur) proses diuji.

Program 20. Deret bilangan ganjil

```
#include <stdio.h>
#include <conio.h>
void main( ){
    int mulai, sampai, i;
    long jml = 0, frek = 0;
    clrscr( );
    printf("DERET BILANGAN GANJIL \n\n");
    printf("Mulai dari ? "); scanf("%d", &mulai);
    printf("Sampai dengan ? ") scanf("%d", &sampai);
    printf("\n");
    i = (mulai % 2 == 0)? mulai + 1; mulai;
    do {
        printf("%d ",i);
        jml += i;
        frek ++;
        i += 2;
    } while (i <= sampai);
        printf("\nJumlah nilai = %ld", jml);
        printf("\nFrekuensi nilai = %d", frek);
}
```

Sebagai contoh kasus akan dilakukan white box testing terhadap program deret bilangan ganjil di atas. Bilangan yang boleh dicetak adalah bilangan-bilangan ganjil antara kedua bilangan masukan, termasuk kedua bilangan tersebut apabila ganjil.

Pada program diatas terdapat satu alternatif proses, yaitu secara urut dari awal sampai pada instruksi pengulangan do while, proses pengulangan (satu kali atau beberapa kali) dan proses akhir yang dilakukan instruksi baris ke-20 dan ke-21.

Soal Latihan

1. Instruksi apa yang digunakan untuk mengolah proses pemilihan?
2. Instruksi apa yang digunakan untuk mengolah proses perulangan?
3. Apakah boleh menggunakan klausa else tanpa if?
4. Gambarkan flowchart proses pemilihan?
5. Gambarkan flowchart proses perulangan?
6. Apa kegunaan instruksi break?
7. Apa kegunaan instruksi case?
8. Bagaimana cara membuat infinite loop dengan instruksi for, while, dan do while?
9. Apakah instruksi while dapat digantikan do while? Jelaskan
10. Apakah instruksi while dapat digantikan for? Jelaskan

BAB V FUNCTION

Salah satu metode perancangan program yang baik adalah menggunakan konsep modular atau sering disebut dengan pemograman modular. Dengan metode ini suatu masalah dipecah menjadi beberapa masalah yang lebih kecil (ke dalam modul-modul). Dengan membagi masalah menjadi beberapa modul, maka masalah tersebut akan menjadi lebih sederhana, sehingga program dapat lebih mudah disusun dan dipahami. Dalam bahasa C modul direalisasi dengan Function.

Manfaat lain dari pemograman modular adalah software reusability. Function yang dibuat satu kali diharapkan dapat digunakan oleh program lain.

Keuntungan menggunakan modul:

1. Rancangan top-down dengan pendekatan divide-and-conquer, program besar dapat dibagi menjadi modul-modul yang lebih kecil;
2. Dapat dikerjakan oleh beberapa orang dengan koordinasi yang lebih mudah;
3. Mencari kesalahan relatif lebih mudah karena alur logika lebih jelas, dan kesalahan dapat dialokasikan dalam satu modul;
4. Modifikasi dapat dilakukan tanpa mengganggu program secara keseluruhan;
5. Mempermudah dokumentasi;

Sifat-sifat modul yang baik adalah;

- Fan-in yang tinggi; makin sering suatu modul dipanggil oleh pengguna makin tinggi fan-in.
- Fan-out yang rendah; makin sedikit (spesifik) tugas yang dilakukan oleh suatu modul, makin rendah nilai fan-out.
- Self-contained; kemampuan memenuhi kebutuhan sendiri.

KATEGORI FUNCTION

Function pada bahasa C terbagi dua:

1. Standard library function.

Standard library function adalah fungsi-fungsi yang telah disediakan compiler C, tinggal digunakan. Untuk menggunakan standard library function maka header file tempat function tersebut didefinisikan harus dipanggil. Misalnya instruksi-instruksi yang berhubungan dengan posisi cursor gotoxy(), wherex(), wherey() dan pembersihan layar clrscr(), clrscr() adalah standard function yang function prototype-nya didefinisikan di header file conio.h. Untuk menggunakan gotoxy() misalnya maka harus disertakan preprocessor directive #include <conio.h> pada awal program agar compiler tahu bagaimana melaksanakan gotoxy(). Yang

disebut dengan function prototype adalah judul function beserta deklarasi jumlah dan tipe data formal parameter (data apa yang harus diberikan kepada function ini untuk diolah).

2. Programmer defined function.

Programmer-defined function adalah fungsi-fungsi yang dibuat pemrogram untuk diunakan dalam program. Function yang dibuat dapat digunakan dalam satu program saja atau dijadikan satu library yang berisi kumpulan fungsi-fungsi sejenis.

STRUKTUR FUNCTION

Dalam merancang function harus ditentukan:

1. Data apa yang diperlukan function sebagai masukan untuk diolah,
2. Informasi apa yang dihasilkan (dikembalikan) function kepada pemanggil.
3. Algoritma yang digunakan untuk mengolah data masukan menjadi informasi keluaran beserta variabel lokal.

Penulisan sebuah function dibagian atas function prototype (pendeklarasian fungsi) dan function definition (pedefinisian fungsi)

❖ `return_type function_name (type1 (type2, ...));`

❖ Function prototype adalah kepala (judul) fungsi.

```
return_type
function_name(parameter_list) {
    deklarasi_variabel_lokal;
    instruksi_1;
    instruksi_2;
    ...
    Instruksi_n;
    return(value);
}
```

Deklarasi ini digunakan compiler dan pemrogram. Oleh compiler digunakan untuk memeriksa apakah pemanggilan terhadap suatu function sudah sesuai baik masukannya (jumlah dan tipe data) maupun keluarannya. Oleh pemrogram digunakan untuk mengetahui argumen (actual parameter) apa yang perlu dikirim kepada function dan jenis data hasil proses.

Function definition mendefinisikan function secara lengkap. Return type menyatakan jenis data apa yang akan dikembalikan function sebagai hasil proses. Nama function harus mengikuti ketentuan penulisan identifier. Daftar parameter terdiri dari nol, satu, atau beberapa parameter. Parameter (disebut juga formal parameter) merupakan data yang harus dikirim kepada function untuk diolah. Instruksi return digunakan untuk mengembalikan hasil proses kepada pemanggil.

STANDARD LIBRARY FUNCTION

Bahasa C menyediakan cukup banyak perpustakaan fungsi. Fungsi-fungsi tersebut dikelompokkan dan deklarasi prototype-nya disimpan dalam beberapa header file.

Beberapa standard function yang berhubungan dengan perhitungan matematika dapat dilihat dibawah ini. Waktu menggunakan function tersebut harus `#include <math.h>` Function lainnya akan diperkenalkan secara bertahap saat membahas topik tertentu.

Tabel 16.Fungsi Standard Matematika

Function	Deskripsi	Contoh
int abs(int X)	Absolut bilangan positif	Abs (-5) → 5
double sqrt (double X)	Akar positif pangkat dua	sqrt (900.0) → 30.0
double exp (double X)	e	exp(1.0) → 2.718282
doubel log (double X)	e log X	log (2.718282) → 1.0
double log10 (double X)	10 log X	log 10 (100.0) → 2.0
double fabs (double X)	Absolut bilangan pecah	Fabs (-5.6) → 5.6
double ceil (double X)	Membulatkan X ke bilangan bulat terkecil yang tidak kurang dari X	Ceil (9.2) → 10.0 Ceil (-9.8) → -9.0

PROGRAMMER-DEFINED FUNCTION

Deklarasi dan Definisi Function

Misalnya akan dibuat sebuah function untuk menentukan dan mengembalikan bilangan terbesar dari antara dua bilangan pecah yang berbeda. Masukan yang harus diberikan kepada function ini adalah kedua bilangan pecah yang akan diuji. Keluaran yang dihasilkan adalah bilangan pecah terbesar di antara kedua bilangan tersebut. Algoritmanya adalah dengan membandingkan kedua bilangan dengan menggunakan instruksi if.

- Mencari bilangan terbesar di antara dua bilangan (function prototype)

- float lebihbesar (float, float);
- Mencari bilangan terbesar di antara dua bilangan berbeda (function definition)

```
float lebihbesar (float f1, float f2) {
    If (f1 > f2 ) return f1; else return f2;
}
```

Function untuk menentukan grade berdasarkan nilai. Masukkan: nilai berupa bilangan bulat. Keluaran : grade berupa satu huruf 'A'. Atau 'B', atau 'C' atau 'D', atau 'E', atau 'F', atau spasi jika data tidak valid. Algoritma: jika nilai > 84 maka kembalikan 'A' dan seterusnya. (Periksa dengan if).

❖ Grade berdasarkan nilai (function prototype)

```
char marktograd (int);
```

❖ Grade berdasarkan nilai (function definition)

```
char marktograd (int mark) {
    char grade;
    if (mark > 100 || mark < 0 ) grade = ' ' else
    if (mark > 84) grade = 'A'; else
    if (mark > 74) grade = 'B'; else
    if (mark > 64) grade = 'C'; else
    if (mark > 49) grade = 'E'; else
    grade = 'F';
    return grade;
}
```

Function untuk menentukan bobot grade berdasarkan grade. Masukkan huruf grade 'A' atau 'B' atau 'C' atau 'D' atau 'E' atau 'F'. Keluaran: jika 'A' maka bobotnya 4, jika 'B' maka bobotnya 3, dan seterusnya. Seleksi dengan instruks switch.

❖ Bobot grade berdasarkan grade (function prototype)

```
int gradetoweight (char);
```

❖ Bobot grade berdasarkan grade (function definition)

```
int gradetoweight (char grade) {
    int bobot;
    switch (grade) {
        case 'a' :
        case 'A' : bobot = 4; break;
        case 'b' :
        case 'B' : bobot = 3; break;
        case 'c' :
        case 'C' : bobot = 2; break;
        case 'd' :
        case 'D' : bobot = 1; break;
        default : bobot = 0;
    }
    Return bobot;
}
```

Function boleh tanpa parameter dengan mengosongkan daftar formal parameter formal parameter ditulis vpid.

- Function tanpa parameter

```
Int judul ( ) {  
    printf("algoritma dan pemrograman");  
    return 1;  
}  
Atau  
Int judul (void) {  
    printf("Algoritma dan pemograman");  
    return1;  
}
```

Return Type dan return

Sebuah function boleh tanpa instruksi return (tidak mengembalikan nilai hasil proses), mempunyai sebuah instruksi return, atau mempunyai beberapa instruksi return. Untuk fungsion yang tidak mengembalikan nilai hasil proses maka return type-nya ditulis void.

- Function dengan return type void

```
void cetaksebarisbinrang (int len) {  
    for (int i = 0; i < len; i++) printf("*");  
}
```

Letak instruksi return disesuaikan dengan kebutuhan. Saat diproses instruksi return memindahkan kontrol proses dari function yang sedang aktif, kembali kepada modul pemanggil. Karena itu bila setelah instruksi return instruksi lagi maka instruksi tersebut tidak akan dikerjakan.

- Dua return tanpa kondisi

```
Float luasdankeliling(float panjang, float lebar) {  
    float luas, keliling;  
    luas = panjang * lebar;  
    keliling = 2 * (panjang + lebar);  
    return luas;  
    return keliling; //instruksi ini tidak akan dikerjakan.  
}
```

Memanggil Function

Pemanggilan terhadap sebuah function adalah suatu ekspresi. Argumen (actual parameter) yang dikirim ke function harus sama banyak dan sesuai tipenya dengan formal parameter, merupakan ekspresi (variabel atau data konstanta). Variabel digunakan untuk menampung hasil pemanggilan function harus setipe dengan return type dari function.

Tabel Function Prototype dan Pemanggilan Function

Function Prototype	Instruksi Pemanggil
float lebihbesar (float, float);	float bil1, bil2, besar; besar = lebihbesar (bil1, bil2);
char marktograd(int);	int nilai; char grade; grade = marktograd(nilai);
int gradetoweight (char);	char grade; int bobot; bobot = gradetoweight(grade);
char* nameofday(int);	int harike; char *namahari; Namahari = nameofday(harike);

Letak Function

Pada penulisan function dalam program yang sama misalnya function 1 dan function 2, dengan function 1 akan memanggil function 2, posisi function 2 dapat ditulis di atas (sebelum) ataupun dibawah (setelah) function 1. Jika function 2 diletakkan di atas function 1 maka function prototype untuk function 2 boleh tidak dituliskan

Program 21. Memanggil Function yang didefinisikan kemudian (di bawah)

```
#include <stdio.h>
float lebihbesar (float, float);
void main() {
    float n1, n2, besar;
    printf("N1 = "); scanf("%f",&n1);
    printf("N2 = "); scanf("%f", &n2);
    besar = lebihbesar (n1, n2);
    printf ("Yang lebih besar: %f", besar);
}
float lebihbesar (float f1, float f2) {
    If (f1>f2) return f1; else return f2;
}
```

Program 22. Memanggil Funtion yang didefinisikan sebelumnya (di atas)

```
#include <stdio.h>
float lebihbesar(float f1, float f2) {
    If (f1 > f2) return f1; else return f2;
}
void main( ) {
    float n1, n2, besar;

    printf("N1= "); scanf("%f", &n1);
    printf("N2= "); scanf("%f", &n2);
    besar = lebihbesar(n1, n2);
    printf("Yang lebih besar:%f", besar);
}
```


Suatu function yang dipanggil function main () dapat memanggil function lainnya.

Program 23.Function memanggil function

```
#include <stdio.h>

long faktorial (int n) {
    int i;
    long f = 1;

    for(i = 2; i <= n; i++) f *= i;
    return f;
}

long kombinasi (int n, int m) {
    long nf, mf, nmf, hasil;

    nf = faktorial (n);
    mf = faktorial (m);
    nmf = faktorial (n-m);
    hasil = nf / (mf*nmf);
    return hasil;
}

void main(void) {
    int bil_n, bil_m;
    long komb;
    printf("menghitung jumlah kombinasi \n\n");
    printf("N ? "); scanf("%d", &bil_n);
    printf("M ?"); scanf("%d", &bil_m);
    if (bil_n >= bil_m && bil_n > 0 && bil_m > 0) {
        komb = kombinasi (bil_n, bil_m);
        printf("Jumlah kombinasi = %ld ", komb);
    }
}
```

JANGKAUAN IDENTIFIER DAN JENIS VARIABEL

Berdasarkan jangkauannya identifier terbagi dua, yaitu global identifier dan local identifier.

Global Identifier

- Identifier yang dideklarasikan di luar function dan ditempatkan di atas semua function dalam suatu program.
- Jangkauan meliputi seluruh program.
- Identifier yang dideklarasikan secara global, dapat dideklarasikan kembali (redeclared) di dalam function.

Local Identifier

- Identifier yang dideklarasikan di dalam function, termasuk daftar parameter.
- Jangkauan terbatas pada function itu sendiri.

Nama variabel adalah identifier sehingga variabel dapat dideklarasikan sebagai variabel lokal atau variabel global.

Kerugian menggunakan variabel global:

- Jika program semakin besar, maka semakin besar pula kecenderungan terjadinya error.
- Sulit melacak kesalahan.
- Data tidak terjaga dengan baik, setiap function dapat mengubah isi variabel tanpa sepengetahuan function lainnya.

Keuntungan menggunakan global identifier adalah transfer data antar-function menjadi mudah. Bahasa C mengelompokkan variabel berdasarkan jangkauannya menjadi:

- 1) Variabel lokal.
- 2) Variabel eksternal.
- 3) Variabel statis
- 4) Variabel register

Variabel Lokal

Variabel lokal adalah variabel yang dideklarasikan di dalam suatu function. Variabel lokal memiliki beberapa sifat:

- Akan dibentuk ketika function dipanggil dan akan hilang ketika eksekusi terhadap function berakhir;
- Hanya dikenal oleh function tempat variabel dideklarasikan;
- Tidak ada inisialisasi secara otomatis (saat dibentuk nilainya bergantung pada sampah memory).

Variabel lokal dapat dideklarasikan dengan memberikan kata kunci `auto` di depan type data. Variabel yang dideklarasikan di dalam function bila tidak diberi kata kunci `auto` akan dianggap sebagai variabel lokal.

❖ Variabel lokal

```
void function1(void) {
    int lokal1;
    auto int lokal2;
    .....
}
```

Variabel Eksternal

Variabel eksternal atau sering juga disebut dengan variabel global merupakan variabel yang dideklarasikan di luar function. Sifat variabel eksternal:

- Dapat akses oleh semua function,
- Apabila tidak diberi nilai awal maka akan diinisialisasi dengan nilai nol.

Program 24. Perubahan Nilai variabel global

```
#include <stdio.h>
int gNilai=98; /*variabel Eksternal */
void inkremen (void) {
    gNilai = gNilai * 1.05;
}

void main( )    {
    printf("Nilai Awal : %d\n", gNilai);
    gNilai = gNilai + 2;
    printf("Nilai Sekarang : %d\n", gNilai);
    inkremen( );
    printf("Nilai Sekarang : %d\n", gNilai);
}
```

Variabel global juga diletakkan di antara function sehingga menjadi semi global, Jangkauan variabel seperti ini adalah mulai dari tempat deklarasi sampai ujung program. Apabila variabel global mempunyai nama yang sama dengan variabel lokal maka yang dikenal pada lokasi setempat adalah variabel lokal.

- ❖ Biasakan tidak menggunakan variabel eksternal (global), melainkan dengan pengiriman data antar function menggunakan parameter.

Variabel Statis

Variable statis dapat berupa variabel internal (variabel lokal) maupun variabel eksternal. Sifat variabel statis:

- Jika variabel statis bersifat lokal, maka variabel hanya dikenal oleh function tempat variabel dideklarasikan.
- Jika variabel statis bersifat eksternal, maka variabel dapat digunakan oleh semua function pada program tersebut.
- Perbedaan dengan variabel lokal adalah variabel statis tidak akan hilang setelah keluar dari pemanggilan function, (nilai, variabel akan tetap diingat).
- Inisialisasi hanya akan dilakukan sekali, yaitu pada saat function dipanggil pertama kali. Jika tidak terdapat inisialisasi maka variabel statis tersebut akan diberi nilai awal nol.

Program 25. Variabel Statis

```
#include <stdio.h>

void naik( ) {
    static int statX;
    statX ++;
    printf("Nilai statX dalam naik( ) : %d\n", statX);
}

void main( ) {
    int statX = 100;
    naik( );
    naik( );
    printf("Nilai statX dalam main( ) : %d\n", statX);
}
```

Variabel Register

Variabel register adalah variabel yang nilainya disimpan dalam register, bukan dalam RAM (Random Access Memory). Variabel seperti ini hanya dapat diterapkan pada variabel yang bersifat lokal atau parameter formal yang bertipe char atau int. Variabel register biasanya digunakan untuk variabel yang berfungsi sebagai pengendali loop. Tujuannya adalah untuk mempercepat proses dalam loop, sebab variabel yang dioperasikan pada register mempunyai kecepatan yang jauh lebih tinggi dari variabel yang diletakkan di RAM.

Variabel Pointer

Variabel pointer adalah variabel yang berisi alamat variabel lain. Variabel dideklarasikan dengan menambahkan tanda bintang sebelum nama variabel.

❖ `data_type *pointer_name;`

Operator yang digunakan dalam mengakses variabel pointer adalah:

Content of → *

Address of → &

Program 26.variabel pointer

```
#include <stdio.h>

void main ( ) {
    int a, b;
    int *p;

    a = 10; b = 20;
    p = &a;
    printf("%x %d \n", p, *p);
    *p = b + 15;
    printf("%d \n", a);
}
```

Dengan menggunakan variabel pointer kita dapat mengubah isi variabel lain dengan mengacukan pointer ini ke alamat variabel tersebut. Variabel pointer juga digunakan untuk pengiriman argumen.

PENGIRIMAN PARAMETER (ARGUMEN)

Komunikasi yang terjadi antara satu function dengan function lain dilakukan dengan cara saling bertukar data. Jika sebuah function memanggil function yang lain untuk melakukan tugas tertentu, kepada function yang dipanggil (called function) diberikan semua data yang dibutuhkannya untuk melaksanakan tugasnya. Setelah melaksanakan tugas tersebut, function yang dipanggil mengembalikan hasil proses kepada function pemanggil (calling function). Pengirim data tersebut melalui parameter yang menjadi interface (antar muka penghubung) suatu modul dengan modul yang lain. Pengirim parameter di dalam bahasa C ada 2 jenis: call by value dan call by reference.

Pengiriman Berupa Nilai (Call By Value)

Yang dimaksud dengan call by value adalah pengiriman ke function lain berupa nilai (r-value). Nilai parameter aktual akan disalin ke parameter formal. Dengan cara ini nilai parameter aktual tidak berubah sekalipun terjadi perubahan nilai parameter formal.

Program 27. Jumlah Ganda

```
#include <stdio.h>
int jumlahganda(int n1, int n2) {
    int n3;
    n3 = 2 * (n1 + n2);
    return n3;
}
void main ( ) {
    int bil1, bil2, bil3;
    printf("Bil 1? "); scanf("%d", &bil1);
    printf("Bil 2? "); scanf("%d", &bil2);
    bil3 = jumlahganda(bil1, bil2);
    printf("jumlah ganda = %d ", bil3);
}
```

Program 28. pertukaran Setempat

```
#include <stdio.h>
void tukar(int x, int y) {
    int z; /* variabel sementara */
    z = x; x = y; y = z;
    printf("Nilai di dalam fungsi tukar \n");
    printf("x = %d y = %d\n\n", x, y);
}
void main ( ) {
    int a = 100, b = 200;
    printf("Nilai sebelum pemanggilan fungsi \n");
    tukar (a,b);
    printf("Nilai di akhir fungsi tukar \n");
    printf("a = %d b = %d\n\n", a,b);
}
```

Pengiriman Berupa Alamat Data (Call By Reference)

Pemanggilan dengan alamat dilakukan dengan cara mengirimkan alamat-alamat variabel melalui parameter aktualnya ke function yang dipanggil. Function yang dipanggil melalui parameter menampung alamat-alamat ini pada variabel-variabel pointer yang terdapat dalam daftar parameter formal. Dengan cara pemanggilan ini, jika terdapat perubahan terhadap data yang dikirimkan, perubahan yang terjadi tidak hanya terjadi secara lokal. Semua perubahan data yang dilakukan pada function yang dipanggil akan otomatis mengubah data pada function yang memanggil. Hal ini disebabkan informasi yang dikirim ke function yang dipanggil adalah alamat data tersebut, sehingga perubahan terhadap data dilakukan langsung di tempat data itu tersimpan. Call by reference disebut juga call by location.

Program 29. Menukar Isi

```
#include <stdio.h>
void tukar(int *n1, int *n2) {
    int n3;
    n3 = *n1;
    *n1 = *n2;
    *n2 = n3;
}
void main ( ) {
    int bil1, bil2;

    printf("Bil-1 ? "); scanf("%d", &bil1);
    printf("Bil-2 ?"); scanf("%d", &bil2);
    tukar(&bil1, &bil2);
    printf("Tukar isinya : \n");
    printf("Bil-1 = %d\n", bil1);
    printf("Bil-2 = %d", bil2);
}
```

REKURSI (RECURSION)

Recursion adalah suatu proses yang memanggil dirinya sendiri kembali. Di dalam C recursion dilakukan oleh function. Function seperti itu disebut recursive function.

```
❖ Return_type func_name (parameter_list){
....
Func_name (...); //recursion
--- }
```

Pemecahan masalah dengan pendekatan recursive dapat dilakukan jika masalah tersebut dapat didefinisikan secara recursive, yaitu masalah semula dapat diuraikan menjadi masalah sejenis yang lebih sederhana.

- Menghitung faktorial secara rekursif

```
long faktorial (int n) {
    if (n == 0 ) return 1;
    else return n * faktorial(n-1); }
```

Program 30. Penjumlahan Deret Kuadrat

```
#include <stdio.h>
int senyum (int n) {
    int hasil = 0;
    if (n == 1) return 1;
    else {
        hasil = (n*n) + senyum(n-1);
        return hasil;
    }
}
void main( ) {
    int i, j;
    printf("n = "); scanf("%d", &i);
    j = senyum(i);
    printf("%d senyum = %d ", i, j);
}
```

Soal Latihan

1. Apa yang dimaksud dengan pemograman modular?
2. Apa bentuk modul pada bahasa C?
3. Apa yang dimaksud dengan software reusablilty?
4. Apa keuntungan menggunakan function?
5. Apa sifat modul yang baik?
6. Apa maksud dan kriteria fan-in yang baik?
7. Apa maksud dan kriteria Fan-out yang baik?
8. Apa yang dimaksud dengan standard library function?
9. Apa standard function boleh didefinisi ulang?
10. Apa yang dimaksud programmer-defined function?
11. Apa kegunaan header file?
12. Apakah pemgoraman juga bisa membuat perpustakaan fungsi?
13. Bagaimana cara membuat perpustakaan fungsi?
14. Apa yang dimaksud function prototype?
15. Apakah function yang dibuat program harus selalu mempunyai function prototype?
16. Apakah sebuah funtion boleh tanpa return type?
17. Apa beda judul function pada function definition dengan function prototype?
18. Apa kegunaan function prototype bagi compiler?
19. Apa kegunaan function prototyep bagi program?
20. Apa function prototype yang menampung printf() dan scanf()?

BAB VI TIPE DATA TERSTRUKTUR

6.1 ARRAY

Array (larik) ialah penampung sejumlah data sejenis (homogen) yang menggunakan satu identifier. Masing-masing elemen array diakses dengan menggunakan indeks (subscript) dari nol sampai $n-1$ (n menyatakan jumlah elemen array). Pengolahan data array harus per elemen. Elemen array dapat diakses secara langsung (random), maksudnya untuk memanipulasi elemen keempat tidak harus melalui elemen pertama, kedua, dan ketiga. Berdasarkan banyaknya indeks array dibagi menjadi satu dimensi dan multi dimensi (dua dimensi, tiga dimensi).

Array Satu Dimensi

Array satu dimensi menggunakan satu buah indeks. Array dideklarasikan dengan:

❖ `data_type array_indeks (size);`

Elemen array dapat juga langsung diberi nilai awal waktu array dideklarasikan. Dalam hal ini ukuran array boleh dituliskan atau dikosongkan.

❖ `data_type array_name() = (constant_1, constant_2, ..., constant_n)`

`data type` menyatakan tipe data array, tidak boleh jenis void.

`array_name` adalah nama array, harus memenuhi ketentuan identifier.

`size` menyatakan jumlah elemen array, normalnya lebih besar dari satu.

`constant_1,, constant_n` adalah nilai awal elemen array dan harus setipe.

❖ Contoh deklarasi array

`Int nilai{4};`

`Char huruf[] = {'a', 'b', 'c'};`

?	?	?	?
---	---	---	---

`nilai[0] nilai[1] nilai[2] nilai[3]`

'a'	'b'	'c'
-----	-----	-----

`huruf[0] huruf[1] huruf[2]`

Perhatikan bahwa elemen array tidak langsung diinisialisasi setiap kali dipesan. Array bilangan tidak dinolkan. array karakter tidak dijadikan spasi melainkan terisi dengan sampah memory, yaitu isi memory apa adanya. Array bilangan yang tidak dinolkan terlebih dahulu langsung digunakan sebagai akumulator akan memberikan hasil yang salah.

Untuk mengakses array tuliskan nama array beserta indeksnya mulai dari 0 sampai `size-1`, berupa konstanta atau ekspresi yang menghasilkan bilangan bulat.

Program 31.Manipulasi Array

```
#include <stdio.h>
#include <conio.h>
void main( ) {
    int bil[7], i;
    clrscr( );
    printf("element pertama? "); scanf("%d", &bil[0]);
    bil[1] = 5;
    bil[2] = bil[1] + 20;
    for (i = 1; i < 7; i++) bil[i] = i * 10;
    bil[3] = bil[bil[1]];
    for(i = 0; i < 7; i++)
        printf("bil[%d] = %d \n", i, bil[i]);
}
```

Pointer dan Array

Array berhubungan erat dengan pointer dan adakalanya dapat saling dipertukarkan. Secara internal array diterjemahkan ke dalam bentuk pointer yaitu array adalah pointer constant ke elemen pertama array, sehingga untuk manipulasi elemen array dapat juga dilakukan dengan pendekatan pointer. Hal ini dimungkinkan karena secara fisik elemen-elemen array diletakkan berurutan dalam memory.

❖ *(array_name + espression)

Array_name tanpa indeks ekuivalen dengan alamat elemen pertama array, maksudnya arr = &arr(0);

- Akses array dengan pointer

```
int arr[10];
int *ptr_arr;
ptr_arr = arr;
```

Untuk mengakses element ke-i dapat dilakukan dengan cara berikut:

```
ptr_arr [i];
arr[i];
*(ptr_arr + i);
*(arr + i);
ptr_arr = ptr_arr + i; *ptr_arr;
```

Program 32.Akses Array dengan pointer

```
#include <stdio.h>
#include <conio.h>
void main( ) {
    int tgl_lahir[ ] = {17, 8, 1945};
    int *ptgl;
    clrscr( );
    /*ptgl berisi alamat awal array */
    ptgl = &tgl_lahir[0];
    printf("REPUBLIK INDONESIA \n\n");
    printf("berdiri pada tgl : %d-%d-%d", *ptgl, ptgl[1], *(ptgl + 2));
}
```

Array Satu Dimensi Sebagai Parameter

Jika array digunakan sebagai parameter function, maka untuk mengitimi array tersebut harus menggunakan alamat array, yaitu passing parameter by reference.

Program 33. Pengiriman Array

```
#include <stdio.h>
#include <conio.h>
#define MAX 1000
void isiarray(int *arr, int n ) {
    int i;
    printf("data awal : ");
    for (i = 0; i < n; i++) scanf("%d", &arr[i]);
}
void gandaarray (int *arr, int n) {
    int i;
    for (i = 0 ; i < n; i++) arr[i] *=2;
}
void cetakarray (int *arr, int n){
    int i;
    printf("hasil ganda: ");
    for(i = 0; i < n; i++) printf("%d ", arr[i]);
}
void main( ){
    int bil[MAX], jmldata;
    clrscr( );
    printf("Jumlah elemen ?");
    scanf("%d", & jmldata);
    isiarray(bil, jmldata);
    gandaarray(bil, jmldata);
    cetakarray(bil, jmldata);
}
```

Pada program diatas mengisi data array dan menggandakannya. Instruksi awal mendefinisikan sebuah konstanta bernama MAX yang bernilai 1000. Nilai ini digunakan sebagai ukuran maksimum array. Apabila ukuran array akan diubah maka cukup ubah nilai ini. Setiap kali compiler menemukan MAX maka akan menggantikannya dengan 1000.

Modul Isianarray() mempunyai formal parameter berupa sebuah pointer to integer dan sebuah integer. Parameter yang pertama menyatakan bahwa bila modul ini dipanggil maka harus kirimkan actual parameter berupa sebuah alamat yang mengacu pada sebuah integer. Parameter kedua digunakan untuk menyatakan jumlah element array yang akan diisi. Modul ini dipanggil dengan actual parameter bil dan jmldata. bil yang dimaksud disini adalah alamat bil[0], yaitu awal array bil.

Karena pointer dan array berkaitan erat maka pengaksesan elemen array dapat dilakukan dengan menggunakan indeks array, walaupun pengiriman parameter dalam bentuk pointer.

Array Multi Dimensi

Bahasa C tidak membatasi jumlah dimensi array yang bisa digunakan. Hal ini semata-mata dibatasi jumlah memory yang tersedia.

Array dua dimensi dapat dibayangkan sebagai tabel. Array tiga dimensi dapat diumpamakan sebagai sebuah balok dengan balok-balok satuan sebagai elemen array.

❖ `data_type array_name (size1)(size2).....(sizen);`

size1, jumlah elemen dimensi pertama

size2, jumlah elemen dimensi kedua

sizen, jumlah elemen dimensi ke-n

Array dua dimensi pada dasarnya adalah array of array, yaitu sebuah array dimensi satu yang elemen-elemennya adalah array yang lain. Bentuk array dua dimensi dapat dipandang sebagai sebuah matriks atau tabel dengan baris dan kolom, dan diacu melalui dua buah indeks (subscript). Indeks pertama menyatakan posisi baris, indeks kedua menyatakan posisi kolom. Banyak elemen data atau ukuran array dapat dihitung dengan mengalikan banyaknya baris dan banyaknya kolom.

6.2 STRING

String adalah kumpulan karakter ASCII. Dalam bahasa C terdapat dua cara mendeklarasikan variabel string. Satu sebagai pointer to character dan yang lain sebagai array of character. Penandaan ujung string dilakukan dengan null character ('\0').

❖ `char *string_id;`

mendeklarasikan sebuah pointer yang akan berisi alamat awal sebuah string

❖ `char string_id(constant);`

mendeklarasikan sebuah array of character

variabel string yang dideklarasikan dapat langsung diinisialisasi dengan string constant (menggunakan tanda petik ganda) atau kumpulan karakter (menggunakan tanda petik tunggal).

❖ Contoh deklarasi variabel string

```
char *s1 = "array";
char s2[6] = "NUSA";
char s3[10] = {'P','R','O','G','R','A','M'};
char s4[] = "jakarta";
```

a	r	r	a	y	\0				
S1[0]	s1[1]	s1[2]	s1[3]	s1[4]	s1[5]				
N	U	S	A	\0					
S2[0]	s2[1]	s2[3]	s2[4]	s2[5]	s2[6]				
P	R	O	G	R	A	M	\0		
S3[0]	s3[1]	s3[2]	s3[3]	s3[4]	s3[5]	s3[6]	s3[7]	s3[8]	s3[9]

j	a	k	a	r	t	a	\0
s4[0]	s4[1]	s4[2]	s4[3]	s4[4]	s4[5]	s4[6]	s4[7]

Karena string adalah array of character maka sebuah string dapat dimanipulasi layaknya sebuah array. Selain dengan cara memanipulasi elemen string secara karakter per karakter, bahasa C menyediakan standard function yang cukup banyak untuk memanipulasi string. Function prototype-nya disimpan pada header file string.h

❖ char strcpy (char*dest, const char*scr)

Function ini menyalin string dari scr ke dest, scr dapat berupa variabel atau string constant.

❖ size_t strlen (const char*s)

function ini mengembalikan angka yang menyatakan panjang string s tanpa menghitung null character pada ujung string.

❖ int strcmp (const char*s1, const char*s2);

Function ini membandingkan string s1 dan s2, bila secara alfabetis;

S1 < S2 → dikembalikan angka negatif

S1 = S2 → dikembalikan angka nol

S1 > S2 → dikembalikan angka positif

❖ int strcmpi (const char *s1, const char *s2);

Function ini membandingkan string s1 dan s2 tanpa membedakan huruf besar dan kecil.

❖ char *strcat(char * dest, const char*src);

Function ini menambahkan string src ke string dest

Program 34. Dua jenis string

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( ) {
    char a[10], *b;
    clrscr( );
    strcpy(a, "bina");
    b = a; // sama dengan b = &a[0];
    printf("1 = %s \n", a);
    printf("2 = %s \n", b);
    printf("3 = %c \n\n", *b);
    strcpy(a, "nusantara");
    printf("4 = %s \n", a);
    printf("5 = %s \n\n", b);
    b = "jakarta";
    printf("6 = %s \n", a);
    printf("7 = %s \n\n", b);
}
```

6.3 STRUCT

Struct adalah suatu struktur data yang menggabungkan beberapa data yang berbeda tipe (heterogen) tetapi berkaitan. Misalnya data mengenai NIM, nama, dan IPK seorang mahasiswa. Ketiga data ini mempunyai tipe data yang berbeda tetapi masih berhubungan yaitu data akademik seorang mahasiswa. Dengan menggunakan struct maka data ini bisa diolah per elemen (per field) atau secara keseluruhan (per struct, per record).

❖ Struct (type_name) {

```
Type mvar_name (,mvar_name,...);  
(type mvar_name (,mvar_name,...);  
} (svar_name);
```

type_name adalah nama tipe struct

mvar_name adalah nama member atau field

svar_name adalah nama variabel struct

Deklarasi variabel struct dapat dilakukan pada saat deklarasi tipe struct atau secara terpisah.

Deklarasi variabel struct secara terpisah adalah:

- ❖ struct type_name svar_name (,Svar_name,...);
- ❖ struct type_name svar_name = {value1, value2,.....,value_n};

- Deklarasi variabel struct

```
struct {  
    char nim[11];  
    char name[26];  
    float_ipk;  
} akad_mhs;
```

- Deklarasi tipe struct dan variabel struct bersamaan, gunakan cara ini apabila struct t_akad akan diacu oleh bagian lain

```
struct t_akad {  
    char nim[11];  
    char nama[26];  
    float ipk;  
} akad_mhs;
```

- Deklarasi tipe struct dan variabel struct secara terpisah

```
struct t_akad {  
    char nim [11];  
    char nama [26];  
    float ipk;  
};  
struct t_akad akad_mhs;
```

Pada sebagian compiler C, kata kunci struct pada deklarasi variabel struct secara terpisah boleh dihilangkan. Seperti halnya tipe data lain, sebuah variabel struct dapat diberi harga awal pada saat pendeklarasian.

- Inisialisasi variabel struct

```
struct t_akad {
    char nim[11];
    char nama[26];
    float ipk;
};
void main ( ) {
    struct t_akad akad_mhs1;
    struct t_akad akad_mhs2=("2017123456","wahyuant0",3.14);
    ----
}
```

Program 35. Jarak Dua Titik

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

Struct t_koord { //global variabel
    int x, y;
};
void main( ) {
    struct t_koord tA, tB;
    float dist;
    clrscr( );
    printf("Titik A : \n ");
    printf("posisi X dan Y ? "); scanf("%d %d", &tA.x, &tA.y);
    printf("\n Titik B : \n");
    printf("posisi X dan Y ? "); scanf("%d %d", &tB.x, &tB.y);
    dist = sqrt (pow((tA.x - tB.x), 2) + pow ((tA.y - tB.y),2));
    printf("\n Jarak A dan B = %.2f unit", dist);
}
```

Menghitung jarak dua titik pada koordinat Cartesius. Tipe struct t_koord digunakan untuk menampung koordinat X dan y sebuah titik. Instruksi scanf() membaca dua nilai sebagai koordinat x titik A (tA.x) dan koordinat y titik A (tA.y). Kemudian membaca data untuk titik B (tB.x dan tB.y). Kemudian menghitung jarak antara kedua titik tersebut dengan pendekatan seperti menghitung sisi miring segitiga siku siku.

Selain deklarasi sebagai tipe data global, struct juga dapat dideklarasikan sebagai tipe data lokal dalam sebuah function.

Program 36.Usia

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

struct t_tgl {
int dd, mm, yy;
};
struct t_teman {
char nama[26];
char kelamin;
struct t_tgl tglhr;
};
void main ( ) {
    struct t_teman teman;
    struct date hari;
    clrscr( );
    printf("Nama ? "); gets (teman.nama);
    printf("Kelamin ? "); teman.kelamin = getchar ( );
    printf("Tgl lahir ? ");
    scanf("%d", &teman.tglhr.dd);
    gotoxy(13, 3); printf("%2d-", teman.tglhr.dd);
    scanf("%d", &teman.tglhr.mm);
    gotoxy(16,6);printf("2%d-", teman.tglhr.mm);
    gotoxy(19,3);scanf("%d", &teman.tglhr.yy);
    getdate(&hari);
    printf("\nUsia = %d", hari.da_year – teman.tglhr.yy);
}
```

Point to Structure

Sebagaimana pointer dapat mengacu ke alamat data tunggal (int, char, float) dan jamak (array), pointer juga dapat mengacu ke struct. Pengaksesan field suatu struct melalui pointer yang mengacu ke struktur lain dapat dilakukan dengan menggunakan indirection operator * dan indirect membership operator →

Structure Sebagai Parameter

Struct dapat dikirim ke function sebagai parameter. Bila struct ini hanya merupakan data masukan maka dapat dikirim dengan pendekatan call by value. Apabila struct yang dikirim ke function akan mengalami perubahan nilai maka pengiriman parameter harus dengan call by reference dengan mengirimkan pointer ke struct. Struct juga dapat menjadi return type sebuah function.

6.4 UNION

Union digunakan untuk berbagi memory. Dengan menggunakan union, suatu lokasi memory dapat ditempati oleh dua atau beberapa variabel dengan masing-masing tipe data yang berbeda.

```
❖ union (type_name){  
    type mvar_name;  
    type mvar_name;  
    (type mvar_name;.....)  
} [uvar_name];
```

type_name adalah nama tipe union

mvar_name adalah nama-nama variabel yang berbagi memory, minimal 2 variabel

uvar_name adalah nama variabel union

Jumlah memory yang diperlukan untuk menampung sebuah variabel union ditentukan oleh field terbesar. Jika elemen-elemen sebuah union terdiri dari data bertipe char dan int maka memory yang dibutuhkan adalah sebesar int. Jika elemen-elemen sebuah union berupa int, float, dan char maka memory yang dibutuhkan adalah sebesar float.

Program 37.Union

```
#include <stdio.h>  
#include <conio.h>  
  
void main( ) {  
    union tbil{  
        unsigned int di;  
        unsigned char dc[2];  
    } bil_x; /* variabel union */  
    Clrscr( );  
    Bil_x.di = 321;  
    Printf(" di    = %d \n", bil_x.di);  
    Printf("dc[0] = %d \n, bil_x.dc[0]);  
    Printf("dc[1] = %d\n", bil_x.dc[1]);  
}
```


SOAL LATIHAN

1. Apa yang dimaksud struktur data?
2. Apa beda array dari variabel biasa?
3. Apa karakteristik array?
4. Bagaimana mengetahui jumlah dimensi array?
5. Bagaimana mengidentifikasi elemen tertentu pada array?
6. Apa ketentuan penulisan indeks suatu array?
7. Apa batasan nilai indeks array?
8. Bagaimana cara mengirim array satu dimensi ke function sebagai parameter?
9. Apa beda struct dan array?
10. Apakah kegunaan union?

BAB VII FILE DATA

Untuk memecahkan masalah dengan bantuan komputer kita menulis program. Program tersebut diketik ke dalam komputer, di-compile, lalu dijalankan. Secara fisik program ini disimpan di random access memory (RAM), demikian juga dengan variabel yang dideklarasikan pada program tersebut. Setelah program selesai dijalankan maka memori yang digunakan variabel dikembalikan ke RAM sehingga semua data tersebut hilang. Demikian juga dengan program akan dihapus dari RAM, RAM bersifat volatile sehingga seluruh data dan program yang ditempatkan pada RAM akan hilang bila komputer dimatikan.

Pada waktu lain program tersebut mungkin diperlukan kembali. Tentunya kita tidak berharap untuk mengetik kembali program tersebut setiap kali ingin dijalankan. Karena itu media penyimpanan data yang bersifat permanen diperlukan. Media sejenis ini disebut media penyimpanan data yang bersifat permanen diperlukan. Media sejenis ini disebut media penyimpanan sekunder (secondary storage device). Program yang diketik tadi bisa disimpan ke media penyimpanan sekunder sebelum komputer dimatikan sehingga lain waktu bila diperlukan program tersebut dapat dipanggil kembali (di-retrieve) tanpa harus diketik ulang. Program disimpan dalam media penyimpanan sekunder dalam bentuk file (tepatnya disebut program file). Compiler yang digunakan untuk menterjemahkan program menjadi bentuk bahasa mesin juga adalah program file. Media penyimpanan sekunder yang umum digunakan adalah floppy disk dan harddisk.

Pada tingkat pemanfaatan yang paling sederhana komputer digunakan untuk mengolah data sehingga disebut electronic data processing. Data yang diolah diharapkan dapat dipanggil kembali pada waktu mendatang, Karena itu data perlu disimpan dalam media yang bersifat permanen dalam bentuk data file.

7.1 FILE DAN STREAM

Stream ialah kumpulan karakter yang disusun dalam baris-baris yang berpindah dari satu media ke media lain pada sistem komputer. Semua data masukan dan keluaran berupa stream. Bahasa C memperlakukan file sebagai stream. Saat program dijalankan maka tiga stream baku otomatis diaktifkan. Masing-masing stream berasosiasi dengan sebuah file. Ketiga stream itu adalah:

1. standard input stream, yang mengatur aliran data masukan melalui ketikan keyboard;
2. standard output stream, yang mengatur aliran keluaran ke layar monitor;
3. standard error stream, yang mengatur tampilan pesan kesalahan ke layar monitor;

Operasi terhadap file dapat dikelompokkan atas operasi yang mengolah isi file (misalnya membaca isi file atau menulis ke dalam file) dan operasi yang tidak mengolah isi file (misalnya mengganti nama file, menghapus file). Operasi yang mengolah isi file memerlukan buffer untuk menampung informasi yang berkenaan dengan file tersebut.

Operasi membuka (mengaktifkan) sebuah file mengakibatkan sebuah pointer dikembalikan kepada instruksi pemanggil. Pointer ini menunjukkan ke sebuah struktur bertipe FILE yang sudah didefinisikan di stdio.h Struktur ini berisi informasi tentang file yang dibuka. Untuk ketiga file baku secara otomatis. Pointer file-nya adalah stdin, stdout, dan stderr.

```
❖ typedef struct{
int          level;          // fill /empty level of buffer
unsigned     flags;          // File status flags
char         fd;             // File descriptor
unsigned     char   hold;    // Ungetc char if no buffer.
int          bsize;          // Buffer size
unsigned     char   *buffer; // Data transfer buffer
unsigned     char   *curp;   // Current active pointer
short        token;          // Temporary file indicator
} FILE;                      // Used for validity checking
```

```
❖ FILE *ptvar;
```

Menyimpan buffer area dan di-assign ke ptvar

TEXT FILE DAN BINARY FILE

Text file ialah berkas yang disimpan dalam format teks. Yang dimaksud dengan format teks adalah format yang sama seperti tampilan di layar. Sebuah bilangan bulat bertipe integer memerlukan memory penyimpanan di RAM sebanyak dua byte. Dengan format teks maka besarnya tempat penyimpanan disesuaikan dengan jumlah angka bilangan, misalnya bilangan 1000 yang berupa integer, apabila disimpan ke dalam file berformat teks memerlukan tempat penyimpanan sebesar empat byte. Karakteristik lain dari file teks adalah isinya dapat dibaca dengan menggunakan editor teks (misalnya editor IDE Borland C++ atau NOTEPAD) atau ditampilkan dengan instruksi sistem operasi TYPE. Binary file sebaliknya menyimpan data numerik dalam format yang tetap sesuai ketentuan micro-processor (misalnya dengan format sign-magnitude 2's complement). Tidak bergantung pada jumlah digit bilangan.

INSTRUKSI I/O FILE DATA

Manipulasi terhadap sebuah file harus diawali dengan pengaktifan file tersebut melalui function fopen(). Berbagai function untuk membaca isi file dan menulis data ke dalam file dikerjakan selama program berjalan. Setelah selesai proses maka file harus ditutup dengan memanggil function fclose () atau fcloseall().

```
❖ FILE *fopen(const char *filename, const char * mode);
```

Function ini mengasosiasikan nama file fisik (physical file) dengan buffer mengembalikan pointer ke awal buffer area, jika gagal akan mengembalikan NULL>

Mode ialah modus pengaktifan file, apakah untuk membaca atau ditulis.

Mode	Kegunaan
"r"	Buka file yang telah ada untuk dibaca
"w"	Buka file baru untuk ditulis
"a"	Buka file yang telah ada untuk ditambah record
"r+"	Buka file yang telah ada untuk dibaca atau ditulis
"w+"	Buka file baru untuk ditulis dan dibaca
"a+"	Buka file yang telah ada dibaca dan ditambah record
"...file baru"	Jika file telah ada maka isi file akan dihapus bersih terlebih dahulu

- Deklarasi file dan membuka file data bernama mhs.dat yang telah ada untuk dibaca
FILE *berkas;

berkas = fopen ("mhs.dat" , "r");

Library function C menyediakan modul yang cukup banyak untuk memanipulasi file.

- int fclose(FILE *stream);

Function ini menutup file stream tertentu. Apabila proses berhasil maka dikembalikan 0. Apabila terjadi kesalahan saat menutup file maka dikembalikan EOF.

- int fcloseall(void);

Function ini menutup seluruh stream yang aktif kecuali stdin, stdout, stderr, dan stderr. Apabila proses berhasil maka dikembalikan bilangan yang menyatakan jumlah stream yang berhasil ditutup. Apabila terjadi kesalahan maka dikembalikan EOF.

- int putc(int c, FILE *stream);

Function ini menulis karakter c ke file stream. Apabila proses penulisan berhasil maka dikembalikan karakter c. Apabila proses penulisan tidak berhasil maka dikembalikan EOF. Function ini memerlukan masukan berupa bilangan tetapi ditulis ke dalam file dalam bentuk karakter ASCII bilangan tersebut.

- int getc(FILE *stream);

Function ini membaca satu karakter dari file stream dan memindahkan penunjuk posisi file stream ke karakter berikutnya. Apabila proses berhasil maka akan dikembalikan karakter yang terbaca. Apabila terjadi kesalahan karena telah end-of-file maka akan dikembalikan EOF.

- int fputc(int c, FILE *stream);

Function ini menulis karakter c ke file stream. Apabila proses penulisan berhasil maka dikembalikan karakter c. Apabila proses penulisan tidak berhasil maka dikembalikan EOF.

- `int fgetc(FILE * stream);`

Function ini membaca karakter dari file stream. Apabila proses pembacaan berhasil maka dikembalikan karakter yang terbaca. Apabila end-of-file maka dikembalikan EOF.

- `int putw(int w, FILE *stream);`

Function ini menulis integer w ke file stream. Apabila proses penulisan berhasil maka function ini mengembalikan nilai w. Apabila proses penulisan tidak berhasil maka function ini mengembalikan EOF.

- `int getw(int w, FILE *Stream);`

Function ini membaca bilangan bulat berikutnya dari file stream ke dalam variabel w. Apabila terjadi kesalahan maka akan dikembalikan EOF.

- `int fputs(const char *s, FILE *stream);`

Function ini menulis string s yang diakhiri NULL ke file stream, tetapi karakter NULL tersebut tidak ikut ditulis ke file, juga tidak ditambahkan karakter newline. Apabila proses penulisan berhasil maka function ini mengembalikan sebuah bilangan positif, sebaliknya mengembalikan EOF.

- `Char *fgets(char *s, int n, FILE *stream);`

Function ini berusaha membaca n karakter dari file stream ke variabel s. Function ini berhenti membaca setelah terbaca n karakter atau terbaca karakter newline. Karakter newline yang terbaca akan ditambahkan pada akhir string. Apabila proses berhasil maka function ini mengembalikan string tersebut (ditunjukkan oleh pointer s), selain itu akan dikembalikan NULL.

- `int fprintf(FILE *stream, const char *format(, argument,...));`

Function ini menuliskan data berformat ke file stream sesuai dengan format masing-masing argumen. Apabila proses berhasil maka function ini mengembalikan bilangan yang menyatakan jumlah byte yang dituliskan, selain itu mengembalikan EOF.

- `int fscanf(FILE *stream, const char *format (, address,...));`

Function ini membaca data dari file stream ke dalam field-field sesuai format. Apabila proses berhasil maka function ini mengembalikan bilangan yang menyatakan jumlah field yang berhasil dibaca. Apabila tidak ada field yang berhasil dibentuk maka akan dikembalikan angka nol. Apabila end-of-file maka akan dikembalikan EOF.

- `Int feof(FILE *stream);`

Function ini mendeteksi akhir sebuah stream. Apabila kondisi end-of-file terpenuhi maka function ini akan mengembalikan bilangan selain nol. Pada standard input stream (stdin) kondisi end-of-file terpenuhi bila pemakai menekan tombol ctrl Z.

- `int fsetpos(FILE *stream, const fpos_t *pos);`

Function ini memindahkan petunjuk posisi file stream ke posisi lain. Posisi lain tersebut didapat dari hasil pemanggilan function `fgetpos()`.

Program 38. Pembentukan File Karakter

```
#include <stdio.h>
void main( ){
    FILE *fout;
    int n, i, j;
    fout = fopen("segitiga.dat", "w");
    printf("N ? "); scanf("%d", &n);
    while (!feof(stdin)) {
        for(i = 0; i < n; i++) {
            for (j = 0; j <= i; j++) putc(65+j, fout);
            putc('\n', fout);
        }
        putc('\n', fout);
        printf("N ? ");
        scanf("%d", &n);
    }
    fclose(fout);
}
```

Pada program diatas bilangan yang ditimbulkan pencacah j bernilai 0,1,2 dan seterusnya yang digunakan membentuk huruf A,B,C dan seterusnya. Karena fungsi yang digunakan adalah `putc()` maka argumen untuk fungsi ini perlu ditambah 65 (nilai ASCII untuk karakter A adalah 65). Program di atas akan memberikan hasil yang sama bila `putc()` diganti `fputc()`.

7.2 FIELD, RECORD, DAN FILE

Field ialah kumpulan karakter yang memiliki makna. Field-field yang berhubungan membentuk sebuah record. Kumpulan record sejenis membentuk field. Data mahasiswa terdiri dari elemen NIM, nama_jenis kelamin, tanggal lahir, tempat lahir, dan lain-lain. Masing-masing elemen data tersebut adalah sebuah field. Field-field untuk seorang mahasiswa membentuk sebuah record mahasiswa. Record-record mahasiswa ini membentuk file mahasiswa.

- Objek data mahasiswa

Field : NIM, nama, jeniskelamin, tanggalahir, tempatlahir.

Record : record mahasiswa

File : file mahasiswa

- Objek prestasi akademik mahasiswa

Field : NIM, nama, ipk

Record : record prestasiAkadMhs

File : file prestasiAkademikMahasiswa

File Prestasi Akademik Mahasiswa		
0600400031	Lidia Karlina	2.93
0600400032	Fadli Sanyoto	2.87
0600400035	Cecep Gorbacep	3.56
0600400038	Kwik Kian liong	3.14

gambar 7.komposisi File

Berdasarkan cara pemanggilan kembali terhadap record, suatu file dapat dibedakan atas sequential-access file dan random-access file. Pada sequential-access file, record harus dibaca mulai dari record pertama. Pada random-access file dapat dilakukan pembacaan terhadap record tertentu secara langsung. Untuk menulis dan membaca random-access file maka posisi record harus ditentukan terlebih dahulu dengan bantuan function `fseek()` lalu menggunakan function `fwrite()` dan `fread()`. Penanganan sequential-access file dapat dilakukan dengan function `fscanf()` dan `fprintf()` maupun function `fwrite()` dan `fread()` function `fwrite()` akan menyimpan data numerik secara biner.

- `size_t fwrite(const void*ptr, size_t size, size_t n, FILE *stream);`

Function ini menambah n item data masing-masing berukuran size ke file stream. Data yang akan ditulis ditunjuk pointer ptr. Apabila proses berhasil maka function ini mengembalikan jumlah item data yang tertulis. Apabila terjadi kesalahan maka akan dikembalikan angka lain.

- `size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`

Function ini membaca n elemen data, masing-masing berukuran size dari file stream ke dalam suatu blok yang ditunjuk pointer ptr. Apabila proses berhasil maka function ini mengembalikan jumlah elemen data yang terbaca. Apabila tidak ada yang terbaca maka function mengembalikan 0.

- `int fseek(FILE *stream, long offset, int whence);`

Function ini memindahkan penunjukkan posisi file stream ke n byte dari lokasi file relatif terhadap whence. Angka n dinyatakan dengan offset. Untuk stream yang berupa text mode offset harus bernilai 0 atau nilai lain yang dikembalikan function `ftell()`, whence harus berupa nilai 0,1, atau 2 yang mewakili tiga symbolic constant.

Tabel 17.symbolic Constant pengatur posisi record

whence	Lokasi File
SEEK_SET atau 0	Awal file
SEEK_CUR atau 1	Posisi pointer saat itu
SEEK_END atau 2	End-of-file

Sequential Access

Pada program berikut ini pembentukan file menggunakan function `fprintf()` dan diberi `\n` pada ujung record, akibatnya file bermodus teks dengan field-field disusun ke samping antara record yang satu dengan yang berikutnya dipisahkan karakter carriage return dan line field sehingga jika ditampilkan dengan editor teks maka record tersusun ke bawah baris demi sebaris. Struktur file seperti ini disebut line sequential. Struktur file line sequential tidak mengenal field. Pemrogramlah yang harus menentukan dan mengingat posisi setiap field.

Program 39.Append File Line Sequential

```
#include <stdio.h>
#include <string.h>
struct t_akad {
    char nim[11];
    char nama[21];
    float ipk;
};
Void main( ) {
    FILE *f_akad;
    struct t_akad r_akad;

    f_akad = fopen("akad.data", "a");
    do {
        fflush(stdin);
        printf("NIM ? "); gets(r_akad.nim);
        if (strlen(r_akad.nim) == 0) break;
        printf("nama ? "); gets(r_akad.nama);
        printf(" IPK ? "); scanf("%f",&r_akad.ipk);
        fprintf(f_akad, "%-10s %-20s %4.2f\n",
            r_akad.nim, r_akad.nama, r_akad.ipk);
    }while (1);
    Fclose(f_akad);
}
```

7.3 STANDARD I/O STREAM

Function yang berhubungan dengan file stream berlaku untuk berbagai stream yang sejenis (masukan atau keluaran) artinya `fscanf()` dapat digunakan untuk membaca masukan melalui standard input stream (`stdin`, masukan melalui ketika keyboard). Demikian juga `fprintf()` dapat digunakan untuk mengirim keluaran ke layar monitor (`stdout`) di samping menulis ke file.

Program 40. Standard I/O Stream

```
#include <stdio.h>
void main( ){
    int umur;
    char *nama;
    fprintf(stdout, "%s", "Nama ?");
    fgets(nama, 25, stdin);
    fprintf(stdout, "%s", "Umur ?"); fscanf(stdin, "%d", &umur ?");
    fprintf(stdout, "\n");
    fputs (nama, stdout); fprintf(stdout, "umur %d tahun", umur); }
```

LATIHAN SOAL

1. Apa yang dimaksud program file dan data file?
2. Apa perbedaan sifat media penyimpanan sekunder dan RAM?
3. Apa yang dimaksud dengan stream?
4. Bagaimana cara mendeklarasikan standard input stream?
5. Apakah stdin itu?
6. Apa yang dimaksud dengan text file?
7. Apa yang dimaksud dengan binary file?
8. Apa hubungan field, record, dan file?
9. Apa perbedaan file dan database?
10. Apa yang dimaksud dengan sequential access?

BAB VIII PENGURUTAN

Dalam pengelompokan data sering data perlu diurutkan (disusun, ditata) sedemikian rupa. Proses pengurutan data disebut sorting. Salah satu tujuan sorting adalah mempercepat pencarian data (searching, retrieving)

Pengurutan data dilakukan berdasarkan nilai kunci (key), Misalnya sejumlah data (record) prestasi akademik mahasiswa yang masing-masing terdiri dari NIM, Nama, dan IPK maka kunci pengurut dapat berupa NIM (Jika ingin diurutkan berdasarkan NIM), atau berupa IPK dan NIM bila ingin didapatkan susunan data dari IPK tertinggi sampai IPK terendah (atau sebaliknya).

Berdasarkan perbandingan nilai data maka sorting dapat dilakukan secara menaik (ascending, atau non-decreasing order) dan menurun (descending atau non-increasing order). Berdasarkan lokasi data saat dilakukan pengurutan maka sorting dibedakan atas internal sorting dan external sorting. Internal sorting adalah proses pengurutan data dengan seluruh data yang akan diolah berada pada Random Access Memory (RAM). Pendekatan ini dilakukan jika data tidak terlalu banyak sehingga dapat seluruhnya dimuat kedalam RAM. External Sorting adalah proses pengurutan data dengan sejumlah data berada pada RAM dan selebihnya berada pada secondary storage device. Pendekatan ini dilakukan jika RAM tidak dapat sekaligus menampung seluruh data yang akan diproses.

BUBBLE SORT

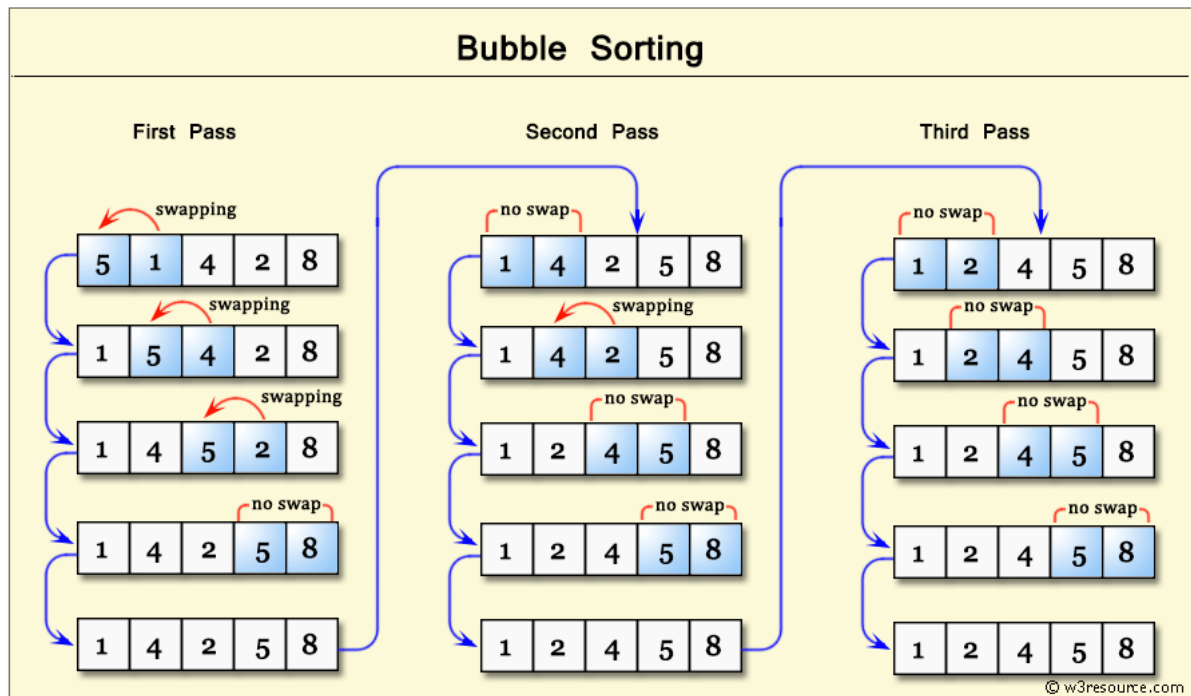
Algoritma ini diberi nama bubble sort karena mengikuti prinsip bubble (gelembung udara). Karena gelembung udara lebih ringan daripada air maka gelembung udara di dalam air akan naik ke atas. Ada penulis yang menamakan algoritma ini exchange sort karena pada proses pengurutan selalu dilakukan pertukaran (exchange) dua data yang berdekatan.

Pengurutan dalam algoritma ini dibagi dalam beberapa putaran (ronde, round). Pada putaran pertama dicari data dengan nilai terkecil (jika pengurutan secara ascending) dan meletkannya pada posisi indeks terkecil (0). Putaran kedua bertujuan mencari data nomor dua kecil dan meletakkannya pada (1) dan seterusnya.

Pada setiap putaran, proses pencarian data terkecil dilakukan dengan membandingkan data terakhir (data(n-1)) dengan data sebelumnya (data(n-2)). Jika ternyata data(n-1) lebih kecil daripada data(n-2) maka tukarkan data tersebut. Selanjutnya perbandingan dilakukan antara data(n-2) dengan data(n-3), dan seterusnya. Banyaknya putaran pengurutan adalah sejumlah data dikurangi satu.

❖ Bubble sort

```
Void bubblesort (int *arr, int n) {  
    Int i, j;  
    For (i = 1, i < n, i++)  
        For(j = n-1; j >= i; j--)  
            If(arr[j-1] > arr[j] tukar (&arr[j], &arr[j-1]);  
}
```



gambar 8.Bubble Sort

SELECTION SORT

Algoritma ini membagi proses pengurutan menjadi putaran-putaran. Pada putaran pertama diseleksi data dengan nilai terkecil dan data ini ditempatkan pada posisi indeks terkecil (data(0)). Pada putaran kedua diseleksi data dengan nilai nomor dua kecil untuk ditempatkan pada posisi kedua (data(1)). Proses seleksi dalam satu putaran dilakukan dengan membandingkan nilai elemen pada dua indeks pembanding. Selama proses pembandingan berlangsung perubahan hanya dilakukan terhadap indeks pembanding. Pertukaran data secara fisik dilakukan pada akhir setiap putaran. Jadi berbeda dengan bubble sort yang pada setiap pembandingan dilakukan pertukaran data bila diharuskan, selection sort melakukan pertukaran data hanya satu kali untuk setiap putarannya yaitu saat putaran tersebut selesai.

❖ Selection sort

```
void selecionsort (int * arr, int n) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        temp = i;
        for (j = i+1; j < n; j++)
            If (arr[j] < arr[temp]) temp = j;
        If (temp != i )tukar(&arr[temp],&arr[i]);
    }
}
```

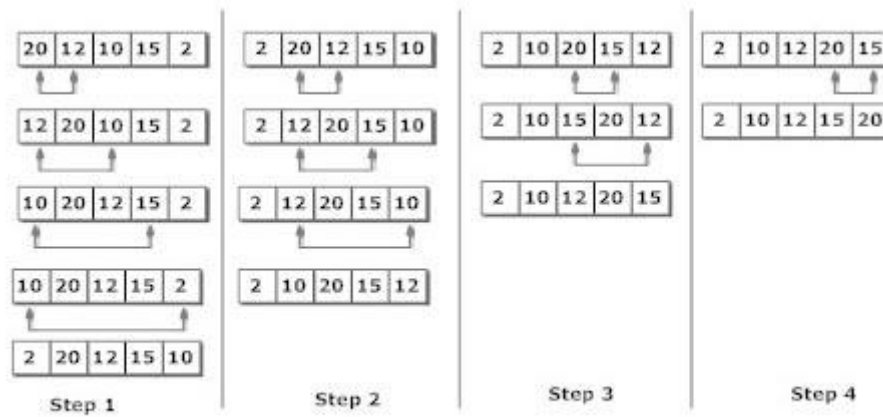


Figure: Selection Sort

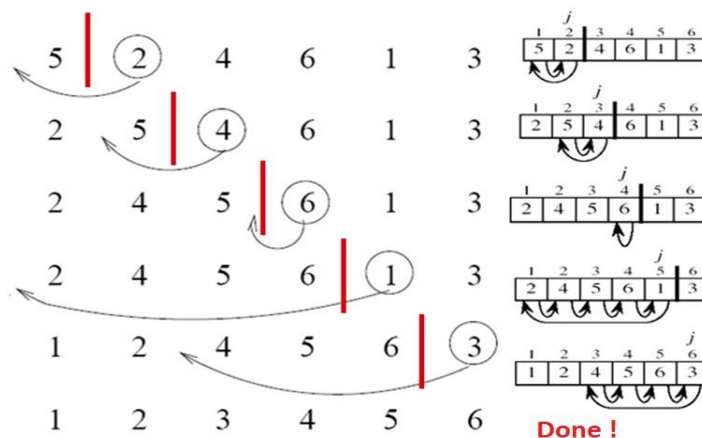
gambar 9.Selection Sort

INSERTION SORT

Cara kerja insertion sort mirip seorang yang mengurutkan kartu. Selembar demi selembar kartu diambil dari kumpulan kartu dan disisipkan pada posisinya yang tepat. Pada putaran pertama urutkan 2 data pertama. Pengurutan ini bersifat relatif, artinya kedua data ini belum tentu 2 data terkecil dari seluruh data melainkan bahwa kedua data ini telah terurut dalam lingkup dua kartu. Pada putaran kedua dicarikan tempat sisip yang tepat bagi data(2).

❖ Insertion Sort

```
void insertionsort (int *arr, int n) {
    int i, j, temp;
    for (i = 1; i <= n - 1; i++) {
        temp = arr[i];
        for (j = i - 1; j >= 0 && arr[j] > temp; j--)
            arr[j+1] = arr[j];
        arr[j+1] = temp;
    }
}
```



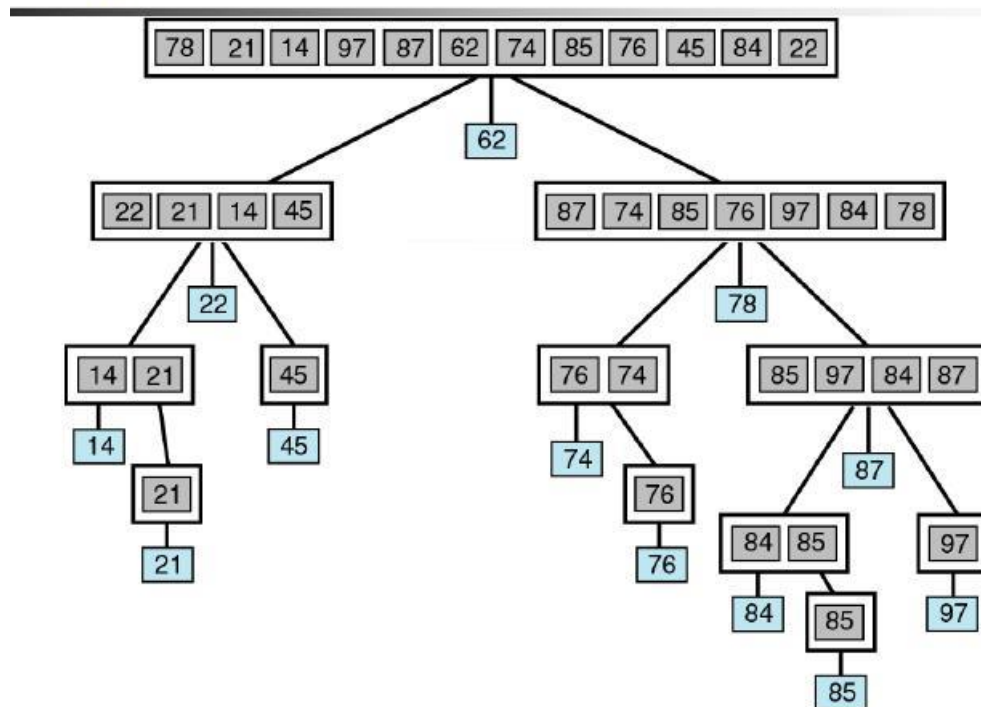
gambar 10.Insertion Sort

QUICK SORT

Quicksort bekerja dengan memartisi sekumpulan data menjadi dua bagian sedemikian rupa sehingga elemen tertentu (elemen ke-i) berada tepat pada posisinya, semua elemen yang nilainya lebih kecil dari elemen ke-i berada disebelah kiri elemen ke-i dan semua elemen yang nilainya lebih besar dari element ke-i berada disebelah kanan element ke-i

Selanjutnya elemen-elemen pada partisi di sebelah kiri elemen ke-i di-sort lagi dengan cara yang sama, demikian pula dengan elemen-elemen pada partisi di sebelah kanan elemen ke-i.

QUICK SORT OPERATION



gambar 11.QuickSort

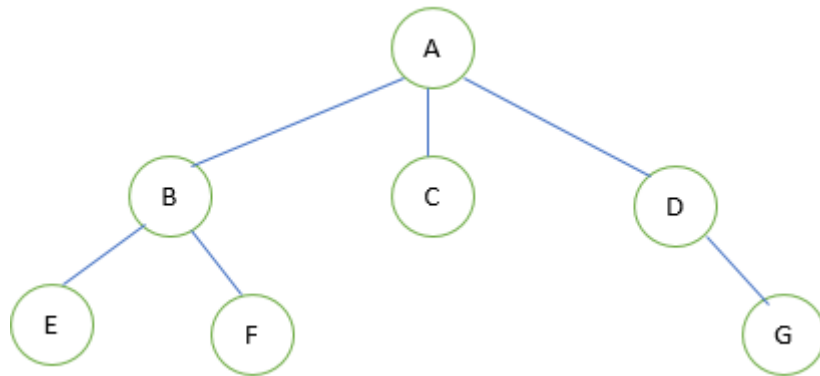
MERGE SORT

Mergesort adalah pengurutan dengan cara penggabungan. Dua kumpulan data yang masing-masing telah diurutkan digabung menjadi satu. Penggabungan dimulai dengan menggabungkan kelompok data dengan jumlah elemen terkecil, yaitu kelompok satu data digabung dengan kelompok satu data. Algoritma mergesort mempunyai kompleksitas $O(n \log n)$ untuk worst case tetapi memerlukan memori yang lebih banyak.

Algoritma pengurutan ini berbeda dengan empat algoritma pengurutan sebelumnya yang melakukan proses pengurutan “di tempat”. Yang dimaksud dengan pengurutan “di tempat” adalah proses pertukaran data langsung dilakukan pada array penampung data, tidak menggunakan array pembantu.

HEAP SORT

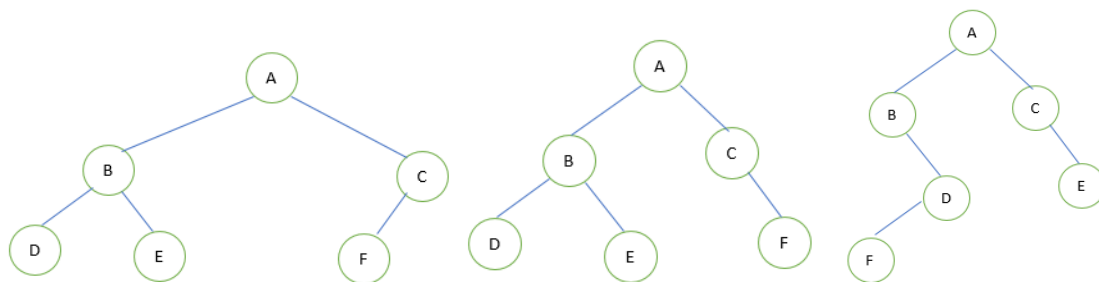
Tree (pohon) adalah struktur data hierarkis (bertingkat) yang menghubungkan satu elemen data dengan beberapa elemen data lainnya. Masing-masing elemen data pembentuk tree disebut node. Tree digambarkan seperti pohon yang terbalik. Tinggi suatu tree diistilahkan dengan height. Masing-masing tingkat disebut level, dihitung dari posisi paling atas.



gambar 12.HeapSort

Node A disebut root. Node B, C, dan D disebut children dari node A. Node A disebut parent dari node B. Node A juga merupakan parent dari node D. Node yang mempunyai children disebut interior node (contoh node A, B, D) Node yang tidak mempunyai children disebut leaf (contoh node E, F, C, G). Tree di atas mempunyai height 3.

Binary tree adalah tree yang mempunyai karakteristik setiap parent mempunyai maksimum dua children. Node anak sebelah kiri disebut leftchild dan node anak sebelah kanan disebut rightchild. Seluruh node yang berinduk pada sebuah leftchild membentuk left subtree. Pada binary tree 2 dibawah ini B adalah leftchild dari A, C adalah rightchild dari A, D adalah leftchild dari B, dan E adalah rightchild dari B. Tree yang dibentuk node B,D, dan E adalah left subtree dari A



gambar 13.Binary Tree

Complete binary tree ialah binary dengan height minimum dan node-node pada level terendah tersusun dari kiri ke kanan. Pada gambar di atas hanya binary tree1 yang merupakan complete binary tree.

Heap adalah complete binary yang nilai setiap parent lebih kecil atau lebih besar daripada nilai kedua anaknya. Untuk keperluan heapsort, head disusun sehingga nilai setiap parent

lebih besar daripada nilai kedua anaknya. Heap diimplementasi dengan menggunakan array dengan elemen [0] dikosongkan. Sehingga $\text{data}[i] > \text{data}[2*i+1]$ dan $\text{data}[i] > \text{data}[2*i+2]$.

Algoritma heapsort mempunyai kompleksitas $O(n \log n)$. Proses heapsort terhadap data yang ditampung dalam array dibagi atas dua tahap.

1. Heapify terhadap data awal sehingga memenuhi kondisi heap, yaitu dilakukan pertukaran sehingga $\text{data}[i] > \text{data}[2*i+1]$ dan $\text{data}[i] > \text{data}[2*i+2]$
2. Tukar $\text{data}[1]$ dan $\text{data}[j]$ untuk $j = n, n-1, \dots, 2$ setiap pertukaran data menyebabkan kondisi heap menjadi rusak. Untuk menanggulangi hal ini lakukan heapify terhadap $\text{data}[1]$.

SHELL SORT

Cara kerja algoritma shellsort mirip dengan algoritma insertion sort. Insertion sort membandingkan elemen-elemen data yang berdekatan (berjarak satu posisi). Shellsort mengurutkan elemen berjarak tertentu (diminishing distance) misalnya yang berjarak 5 posisi, lalu berjarak 3 posisi, dan terakhir berjarak 1 posisi. Diminishing distance harus dicari sedemikian rupa sehingga tidak mengulangi atau merusak hasil sorting sebelumnya. Beberapa usulan angka diminishing distance adalah:

Shell's increment: 1,2,4,8

Hibbard's increment : 1,3,7,15,.....,

Sedgewick's increment : 1,5,19,41,109

Pada proses sorting jarak diambil secara menurun. Sorting elemen secara insertion. Pada pengurutan elemen-elemen jarak lima

LATIHAN SOAL

1. Apa yang dimaksud dengan sorting?
2. Apa kegunaan dari sorting?
3. Apa yang dibandingkan dalam proses sorting?
4. Apa yang dimaksud dengan sorting secara descending?
5. Apa yang dimaksud dengan sorting secara ascending?
6. Jelaskan cara kerja bubble sort?
7. Apa perbedaan antara bubble sort dengan exchange sort?
8. Sebutkan jenis-jenis algoritma sorting?
9. Secara umum proses sorting mana yang lebih cepat?
10. Jelaskan cara kerja insertion sort?