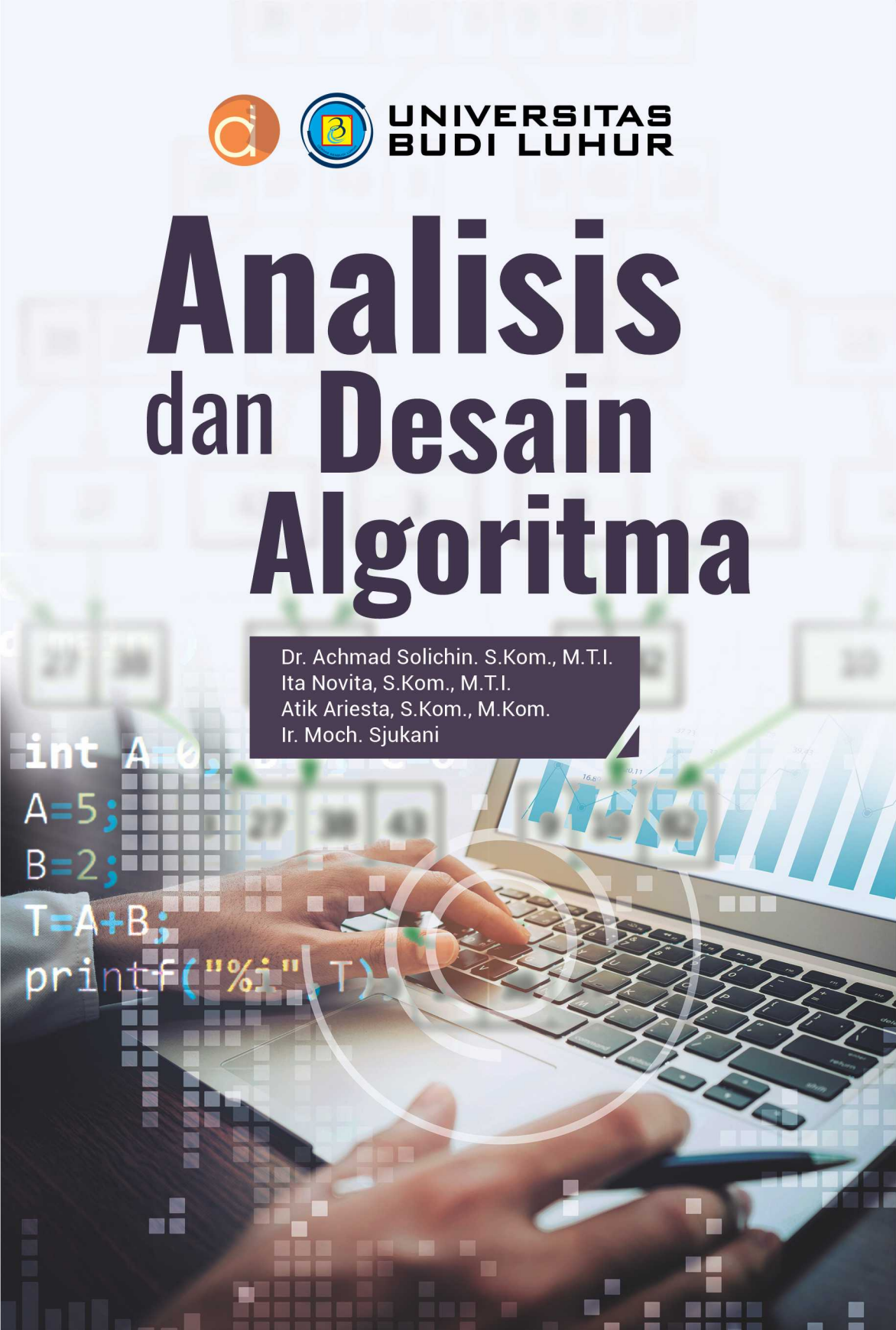
UNIVERSITAS BUDI LUHUR

Analisis dan Desain Algoritma

Dr. Achmad Solichin. S.Kom., M.T.I.
Ita Novita, S.Kom., M.T.I.
Atik Ariesta, S.Kom., M.Kom.
Ir. Moch. Sjukani



Analisis
dan Desain Algoritma

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

Analisis dan Desain Algoritma

Dr. Achmad Solichin, S.Kom., M.T.I.
Ita Novita, S.Kom., M.T.I.
Atik Ariesta, S.Kom., M.Kom.
Ir. Moch. Sjukani

deepublish
Cerdas, Bahagia, Mulia, Lintas Generasi.
Digital

Analisis dan Desain Algoritma

Nama Penulis
Dr. Achmad Solichin, S.Kom., M.T.I.
Ita Novita, S.Kom., M.T.I.
Atik Ariesta, S.Kom., M.Kom.
Ir. Moch. Sjukani

Desain Cover :
Syaiful Anwar

Sumber :
www.shutterstock.com (HAKINMHAN)

Tata Letak :
Hifzillah Fahmi

Proofreader :
Tiara Nabilah Azalia

Ukuran :
x, 216 hlm, Uk: 15.5x23 cm

ISBN :
978-634-200-004-5 (PDF)

Tahun Terbit Digital :
2024

PENERBIT DEEPUBLISH DIGITAL
Jl.Rajawali, G. Elang 6, No 3, Drono, Sardonoharjo, Ngaglik, Sleman
Jl.Kaliurang Km.9,3 – Yogyakarta 55581
Telp: +6281362311132
Website: www.deepublish.co.id
www.deepublishdigitalstore.com
E-mail: digital@deepublish.co.id

Penerbitan buku ini sudah bekerjasama dengan Universitas Budi Luhur

KATA PENGANTAR

Puji syukur dan hormat, kami haturkan ke hadirat Allah Swt., karena atas perkenan-Nya, sehingga kami dapat menyelesaikan buku *Analisis dan Desain Algoritma* ini.

Penghargaan tertinggi dan ucapan terima kasih yang sebesar-besarnya kepada seluruh pihak yang telah memberikan bantuan serta dukungan dalam menyusun buku ini.

Tim penyusun telah menyusun buku ini semaksimal mungkin, namun kami menyadari bahwa penyusun tentunya tidak lepas dari salah dan khilaf semata. Tim penyusun sangat terbuka untuk berbagai masukan, ide dan saran dari berbagai pihak agar buku ini bisa lebih baik lagi.

Jakarta, Januari 2020

Tim Penyusun

KATA PENGANTAR PENERBIT

Segala puji kami haturkan kepada Tuhan Yang Maha Esa, atas limpahan segala anugerah dan karunia-Nya. Dalam rangka mencerdaskan dan memuliakan umat manusia dengan penyediaan serta pemanfaatan ilmu pengetahuan dan teknologi untuk menciptakan industri *processing* berbasis sumber daya alam (SDA) Indonesia, Penerbit Deepublish Digital dengan bangga menerbitkan buku dengan judul ***Analisis dan Desain Algoritma.***

Buku ini memuat 14 Bab yang mengulas tentang analisis dan desain dari algoritma. Algoritma adalah langkah-langkah yang diambil dalam menyelesaikan suatu pekerjaan. Suatu pekerjaan dapat diselesaikan dalam satu langkah, dua langkah atau banyak langkah. Langkah-langkah harus tersusun secara logis agar pekerjaan dapat diselesaikan dengan benar.

Terima kasih dan penghargaan terbesar kami sampaikan kepada penulis yang telah memberikan kepercayaan, perhatian, dan kontribusi penuh demi kesempurnaan buku ini. Semoga buku ini bermanfaat bagi semua pembaca, mampu berkontribusi dalam mencerdaskan dan memuliakan umat manusia, serta mengoptimalkan pemanfaatan ilmu pengetahuan dan teknologi di tanah air.

Hormat Kami,

Penerbit Deepublish Digital

DAFTAR ISI

BAB 1

PENGANTAR ALGORITMA

Capaian Pembelajaran	:	Mahasiswa dapat memahami konsep dasar algoritma, program, *pseudocode* dan *flowchart*.

Subpokok Bahasan	:	1.1. Pengertian Algoritma 1.2. Pengertian Program 1.3. Pengertian *Pseudocode* 1.4. Algoritma VS *Pseudocode* 1.5. Pengertian *Flowchart* 1.6. Simbol-Simbol *Flowchart* 1.7. Contoh Studi Kasus *Pseudocode* VS *Flowchart*

Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

PENGANTAR ALGORITMA

1.1. PENGERTIAN ALGORITMA

Algoritma adalah langkah-langkah yang diambil dalam menyelesaikan suatu pekerjaan.

Suatu pekerjaan dapat diselesaikan dalam satu langkah, dua langkah atau banyak langkah. Langkah-langkah harus tersusun secara logis agar pekerjaan dapat diselesaikan dengan benar.

Dalam pelajaran algoritma yang menyelesaikan pekerjaan, adalah komputer. Tugas kita adalah memberikan perintah kepada komputer, langkah per langkah yang akan dilaksanakan oleh komputer untuk menyelesaikan pekerjaan tersebut.

Algoritma merupakan gabungan seni dan teknik.

Seni, karena algoritma penuh dengan kreativitas dan imajinasi yang jenius.

Teknik, karena algoritma diterapkan di komputer yang penuh dengan *tool* dan metodologi.

KRITERIA ALGORITMA

Setiap algoritma harus memenuhi kriteria sebagai berikut:

1) Ada atau tidak ada data yang dimasukan dari luar.
2) Paling tidak ada satu buah keluaran.
3) Setiap instruksi jelas maksudnya dan hanya mempunyai satu arti.
4) Algoritma baik secara keseluruhan maupun sub algoritma bila ditelusuri harus ada titik hentinya.
5) Setiap instruksi selain jelas juga harus dapat dilaksanakan, dan juga efektif dalam arti hanya menghasilkan sesuatu. Sebagai contoh A=A+0 (A ditambah 0) atau A=A*1 (A dikali satu), adalah termasuk instruksi yang tidak efektif.

1.2. PENGERTIAN PROGRAM

Program adalah kumpulan instruksi-instruksi yang diberikan kepada komputer untuk menyelesaikan suatu tugas.

Instruksi-instruksi merupakan langkah-langkah dalam algoritma yang tersusun secara logis.
Program ditulis dalam suatu bahasa yang disebut dengan bahasa pemrograman (*programming language*).
Contoh bahasa pemrograman yaitu Cobol, Fortran, Pascal, Basic, Java, C dan sebagainya.
Pada modul ini akan digunakan Bahasa C untuk menerapkan logika di komputer.

1.3. PENGERTIAN *PSEUDOCODE*

Pseudocode adalah kode atau tanda atau ceritera yang menyerupai atau merupakan (pseudo) penjelasan cara menyelesaikan persoalan.
Kode atau tanda atau ceritera tersebut ditulis dalam suatu bahasa yang dimengerti oleh manusia.
contoh STUDI KASUS *PSEUDOCODE*
Bagaimana *login* ke Facebook?
Dalam bahasa sederhana:

1. Buka *website* www.facebook.com
2. Isi *Username*
3. Isi *Password*
4. Klik tombol *Login*

1.4. ALGORITMA VS *PSEUDOCODE*

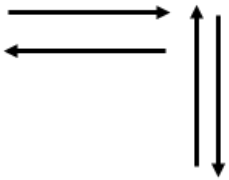
ALGORITMA	PESUDOCODE
A = A + 5	Nilai A ditambah 5
IF(A>5) THEN WRITE(A)	Cetak nilai A, bila nilai tersebut lebih besar dari 5
IF(A>B) THEN WRITE(A) **ELSE WRITE(B)**	Dari dua buah nilai A dan B cetak salah satu yang terbesar

While (A>0) do
A=A-2
End Do
	Kurangi dengan 2 nilai A terus menerus sampai nilainya lebih kecil atau sama dengan nol

1.5. PENGERTIAN *FLOWCHART*

Flowchart merupakan American National Standard Institute (ANSI) untuk menggambarkan algoritma dalam bentuk gambar dengan panah yang menunjuk alur suatu aktivitas.

1.6. SIMBOL-SIMBOL *FLOWCHART*

No.	Simbol	Nama Simbol	Keterangan Simbol
1		Terminal	Menggambarkan sebuah awal atau akhir program
2		Input/Output	Menggambarkan Input atau Output
3		Proses	Menggambarkan jenis operasi internal seperti inisialisasi atau perhitungan
4		*Decision*	Digunakan untuk menanyakan yang memiliki jawaban TRUE/FALSE (YES atau NO)
5		Konektor	Digunakan untuk menghubungkan *flowchart* yang terbelah/terpisah
6		*Control Flow*	Menunjukkan arah dari aktivitas

Aturan penggambaran *flowchart*

Dalam membuat *flowchart* harus mengikuti aturannya, yaitu:

1) *Flowchart* umumnya digambarkan dari atas ke bawah
2) Semua simbol *flowchart* harus terhubung dengan panah (simbol *control flow*)
3) *Flowchart* diawali dan diakhiri dengan simbol terminal
4) Khusus simbol *decision*, memiliki dua arah keluaran satu untuk True (yes) satu lagi untuk False (no)

1.7. CONTOH STUDI KASUS *PSEUDOCODE* VS *FLOWCHART*

SOAL-1

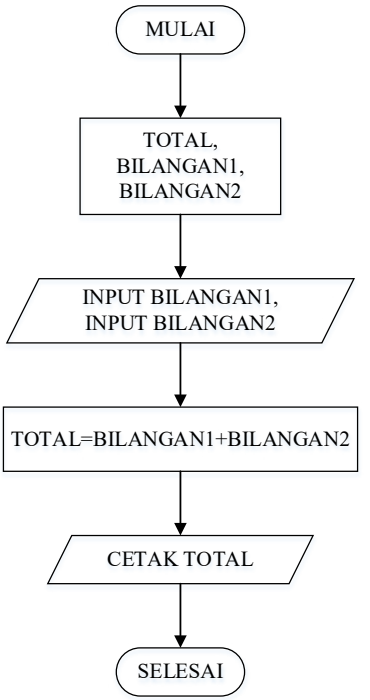
Hitung dan cetak TOTAL PENJUMLAHAN dari bilangan yang diinput yaitu 10 dan 20.

Jawab:

PSEUDOCODE

1. Inisialisasi TOTAL=0, BILANGAN1=0, BILANGAN2=0
2. Input BILANGAN1 dengan 10
3. Input BILANGAN2 dengan 20
4. Tambahkan BILANGAN1 dengan BILANGAN2 yang disimpan kedalam TOTAL
5. Tampilkan TOTAL

FLOWCHART



SOAL-2

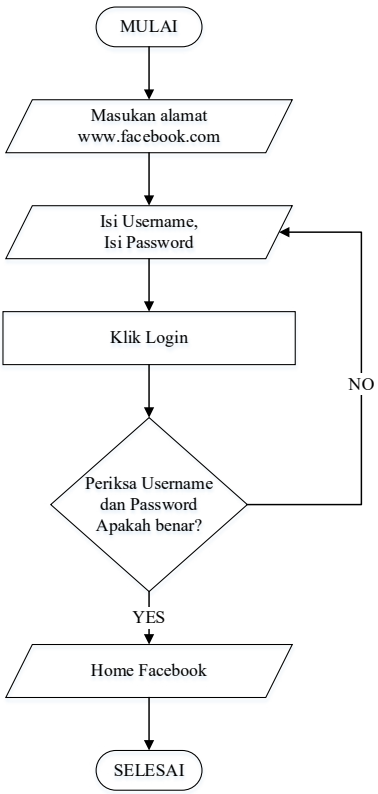
Bagaimana *Flowchart* untuk *login* ke Facebook?

Jawab:

PSEUDOCODE

1. Buka *website* www.facebook.com
2. Isi *Username*
3. Isi *Password*
4. Klik tombol *Login*
5. Jika *username* dan *password* sesuai, tampil halaman Home Facebook
   Jika *username* dan *password* tidak sesuai, ulangi isi *username* atau *password*

FLOWCHART



KESIMPULAN

Membuat algoritma yang akan dijalankan oleh komputer dapat menggunakan du acara yaitu:

1. Membuat *Pseudocode*
2. Membuat *Flowchart*

Setiap desain sebuah algoritma antara satu orang dan orang yang lain dapat berbeda, hal ini adalah wajar selama algoritma tersebut dapat memecahkan permasalahan yang diberikan dengan benar dan efisien.

SOAL LATIHAN

1. Bagaimana cara untuk mengirim pesan melalui WhatsApp? Buatlah *pseudocode*-nya dan *flowchart*-nya.
2. Bagaimana cara untuk memeriksa *e-mail* yang masuk? Buatlah *pseudocode*-nya dan *flowchart*-nya.
3. Hitung dan cetak LUAS PERSEGI dari panjang yang diinput 10 dengan lebar 10. Buatlah *pseudocode* dan *flowchart*-nya.
4. Hitung dan cetak KELILING PERSEGI dari sisi yang diinput 10. Buatlah *pseudocode* dan *flowchart*-nya.

BAB 2

FLOWCHART

Capaian Pembelajaran	:	Mahasiswa memahami simbol-simbol *flowchart* serta menggunakannya dalam menggambarkan penyelesaian persoalan sederhana.

Subpokok Bahasan	:	2.1. Pengertian Flowchart 2.2. Simbol-Simbol Flowchart 2.3. Aturan Pembuatan Flowchart 2.4. Modulus 2.5. Contoh Penyelesaian Kasus/Persoalan Sederhana

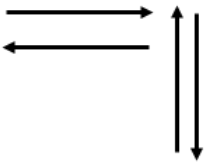
Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

FLOWCHART

2.1. PENGERTIAN *FLOWCHART*

Flowchart merupakan American National Standard Institute (ANSI) untuk menggambarkan algoritma dalam bentuk gambar dengan panah yang menunjuk alur suatu aktivitas.

2.2. SIMBOL-SIMBOL *FLOWCHART*

No.	Simbol	Nama Simbol	Keterangan Simbol
1		Terminal	Menggambarkan sebuah awal atau akhir program
2		Input/Output	Menggambarkan Input atau Output
3		Proses	Menggambarkan jenis operasi internal seperti inisialisasi atau perhitungan
4		*Decision*	Digunakan untuk menanyakan yang memiliki jawaban TRUE/FALSE (YES atau NO)
5		Konektor	Digunakan untuk menghubungkan *flowchart* yang terbelah/terpisah
6		*Control Flow*	Menunjukkan arah dari aktivitas

2.3. ATURAN PEMBUATAN *FLOWCHART*

Aturan penggambaran *flowchart*

1) *Flowchart* umumnya digambarkan dari atas ke bawah
2) Semua simbol *flowchart* harus terhubung dengan panah (simbol *control flow*)
3) *Flowchart* diawali dan diakhiri dengan simbol terminal
4) Khusus simbol *decision*, memiliki dua arah keluaran satu untuk True (yes) satu lagi untuk False (no)

2.4. MODULUS

Modulus adalah sisa pembagian bilangan. Modulus hanya berlaku untuk bilangan integer. Operator yang digunakan adalah %

BEBERAPA CONTOH HASIL PEMBAGIAN MODULUS

15 % 2 = 1	15 % -4 = 3
15 % 3 = 0	-15 % 4 = -3
15 % 4 = 3	-15 % -4 = -3
15 % 15 = 0	-15 % 7 = -1
15 % 17 = 15	15 % -7 = 1
7 % 15 = 7	5 % -7 = 5
0 % 2 = 0	-5 % -7 = -5
	-5 % 7 = -5

CONTOH MODULUS

Budi memiliki 20 buah kelereng. Dia ingin membagi seluruh kelereng ke 3 orang temannya secara merata.

a. Berapa kelereng akan didapat oleh masing-masing teman Budi?
b. Berapa sisa kelereng yang dimiliki oleh Budi setelah dibagikan?

Jawab:

a. Hasil Bagi: 20 / 3 = 6

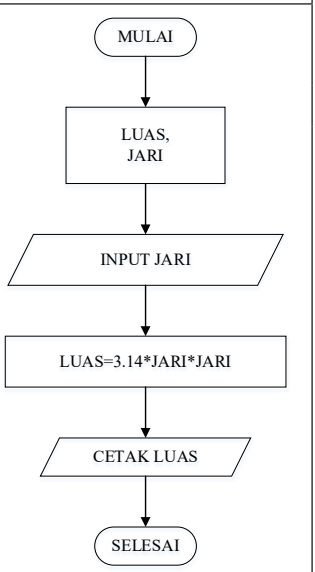
b. Sisa hasil bagi: 20 % 3 = 2

2.5. CONTOH PENYELESAIAN KASUS/PERSOALAN SEDERHANA

SOAL-1

Hitung dan tampilkan Luas Lingkaran yang jari-jarinya di-*entry* melalui *keyboard*. Buatlah *flowchart*-nya!

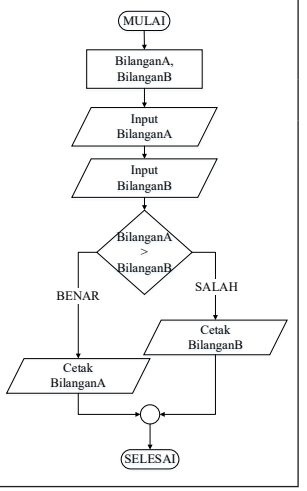
Jawab:

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisialisasi Luas dan Jari	Proses	
2.	Input Jari	Input/Output	
3.	Hitung Luas= 3.14 * Jari * Jari	Proses	
4.	Tampilkan Luas	Input/Output	

SOAL-2

Input dua buah bilangan bulat (menggunakan *keyboard*) dan tampilkan bilangan terbesar di antara kedua bilangan tersebut (dianggap kedua bilangan yang diinput memiliki nilai yang berbeda).

Jawab:

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisialisasi BilanganA, BilanganB	Proses	
2.	Input BilanganA	Input/Output	
3.	Input BilanganB	Input/Output	
4.	BilanganA>BilanganB, cetak BilanganA Jika tidak, Cetak BilanganB	*Decision* dan Input/ Output	

SOAL-3

Inputkan sebuah bilangan bulat melalui *keyboard* kemudian tampilkan perkataan GANJIL jika bilangan tersebut merupakan bilangan ganjil.

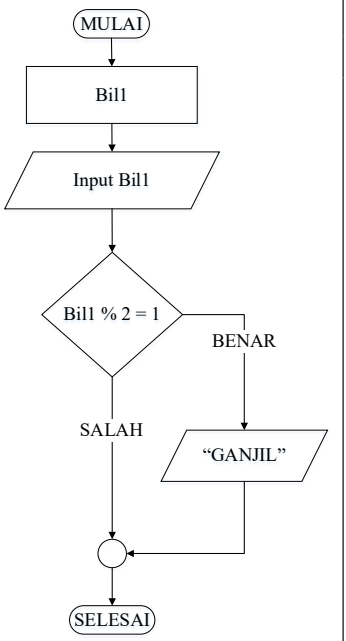
Jawab:

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi Bil1	Proses
2.	Input Bil1	Input/Output
3.	Jika N Ganjil, cetak "GANJIL"	*Decision*, Input/Output

Komputer tidak mengetahui secara langsung angka yang diinput melalui *keyboard* merupakan bilangan ganjil atau bukan.

Bagaimana membuat sebuah perintah agar komputer dapat mengetahui bilangan yang diinput merupakan bilangan ganjil atau bukan? Gunakan Modulus

Perbaikan *Pseudocode*

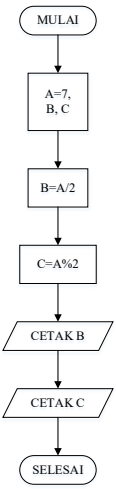
No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisialisasi Bil1	Proses	
2.	Input Bil1	Input/Output	
3.	**Jika Bil1 modulus 2 = 1, cetak "GANJIL"**	*Decision*, Input/ Output	

SOAL-4

Buatlah *flowchart* dari penggalan program di bawah ini!

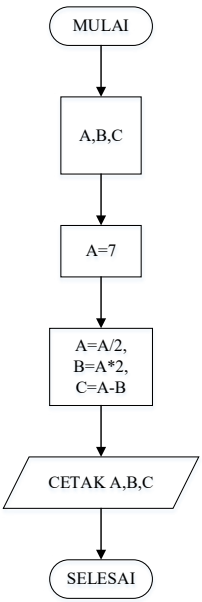
```
#include<stdio.h>
void main()
{
    int A=7,B,C;
    B=A/2;
    C=A%2;
    printf("\n%i",B);
    printf("\n%i",C);
}
```

Jawab:

No.	Penggalan Program	Simbol *Flowchart*	*Flowchart*
1.	int A=7,B,C;	Proses/Inisialisasi	
2.	B=A/2;	Proses	
3.	C=A%2;	Proses	
4.	printf(“\n%i”,B);	Input/Output	
5.	printf(“\n%i”,C);	Input/Output	

SOAL-5

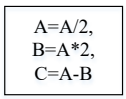
Tuliskan penggalan program dari *flowchart* di bawah ini!



Jawab:

No.	Simbol *Flowchart*	Penggalan Program Sesuai Simbol	Penggalan Program

1.		int A,B,C;	#include<stdio.h>
			void main()
			{
			int A,B,C;
			A=7;
			A=A/2;
			B=A*2;
			C=A-B;
			printf("\n%i",A);
			printf("\n%i",B);
			printf("\n%i",C);
			}

2.		A=7;	
3.		A=A/2; B=A*2; C=A-B;	
4.		printf(“\n%i”,A); printf(“\n%i”,B); printf(“\n%i”,C);	

KESIMPULAN

Flowchart dapat digunakan untuk menggambarkan algoritma dengan menggunakan simbol-simbol *flowchart* serta mengikuti aturan dalam pembuatan *flowchart*.

Ketika membuat algoritma, perhatikan apakah perintah yang akan diberikan nantinya bisa dipahami oleh komputer atau tidak. Jika tidak, maka harus mencari cara agar algoritma bisa dipahami oleh komputer.

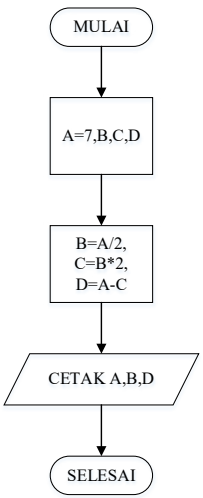
SOAL LATIHAN

1. Buatlah algoritma dan *flowchart* untuk menghitung dan menampilkan luas dari segitiga tersebut jika alas yang diinput adalah 8 cm dan tinggi yang diinput adalah 5 cm. Diketahui Rumus Luas Segitiga = ½ x alas x tinggi.
2. Buatlah algoritma dan *flowchart* untuk menghitung dan menampilkan isi (volume) dari bangun ruang bola tersebut jika diameter bola yang diinput adalah 15 cm. Diketahui Rumus Volume Bola = $4 \times \pi \times r^2$.

3. Buatlah algoritma dan *flowchart* untuk menghitung sisi miring segitiga tersebut dan menampilkannya di layar jika alas yang diinput adalah 4 cm dan tinggi yang diinput adalah 5 cm. Mencari sisi miring segitiga dapat menggunakan rumus phytagoras.
4. Buatlah algoritma dan *flowchart* untuk mencari bilangan TERKECIL dari dua bilangan bulat yang diinput (dianggap kedua bilangan nilainya berbeda)!
5. Buatlah algoritma dan *flowchart* untuk mencetak perkataan "SAMA KAKI" jika segitiga memiliki dua sisi yang sama, berdasarkan 3 buah bilangan bulat yang diinput mewakili sisi-sisi dari sebuah segitiga.
6. Buatlah algoritma dan *flowchart* untuk mencari dan mencetak bilangan TERBESAR di antara ketiganya bilangan yang diinput (dianggap ketiga bilangan nilainya berbeda)!
7. Buatlah *flowchart* dari penggalan program di bawah ini!

```
#include<stdio.h>
void main()
{
    int A=25,B,C;
    B=A/2;
    C=A-B*2;
    printf("\n%i",C);
}
```

8. Buatlah penggalan program dari *flowchart* di bawah ini!



BAB 3

DASAR PEMROGRAMAN

Capaian Pembelajaran	:	Mahasiswa memahami konsep dasar dalam bahasa pemrograman.

Subpokok Bahasan	:	3.1. Bahasa Pemrograman 3.2. Variabel 3.3. Konstanta 3.4. Tipe Data 3.5. Operator

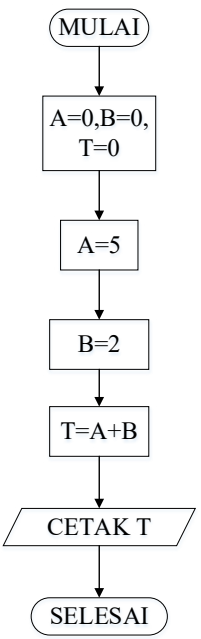
Daftar Pustaka	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

3.1. BAHASA PEMROGRAMAN

Program adalah kumpulan instruksi-instruksi yang diberikan kepada komputer untuk melaksanakan suatu tugas atau pekerjaan.
Program adalah algoritma yang ditulis dalam suatu bahasa yang dikenal oleh komputer yang disebut bahasa pemrograman (*programming language*).
Contoh bahasa pemrograman yang sering digunakan di Indonesia antara lain Java, C, PHP, Visual Basic, Phyton, C++, JavaScript, C#, Objective-C, ActionScript.

CONTOH IMPLEMENTASI *FLOWCHART* DALAM BEBERAPA BAHASA PEMROGRAMAN

Tuliskan program (penggalan program) untuk menyatakan Algoritma yang digambarkan pada *flowchart* berikut:



Jawab:

PrEogram 3.1 Contoh Program dalam Bahasa C

```c
#include "stdio.h"
void main()
{
    int A=0, B=0, C=0;
    A=5;
    B=2;
    T=A+B;
    printf("%i",T);
}
```

Program 3. 2 Contoh Program Java

```java
public class Hitung
{
    public static void main(String args[])
    {
        int A=0, B=0, T=0;
        A=5;
        B=2;
        T=A+B;
        System.out.println(T);
    }
}
```

BAHASA MANUSIA VS BAHASA PEMROGRAMAN

Komputer hanya mengerti bahasanya sendiri yaitu bahasa mesin. Sebuah *pseudocode* tidak dipahami oleh komputer oleh karena itu diperlukan sebuah bahasa pemrograman agar dapat dimengerti oleh komputer.

No.	Bahasa Manusia	Bahasa Pemrograman
1	Diajarkan oleh manusia ke manusia	Diajarkan oleh manusia ke komputer
2	Sebagai sarana komunikasi antara manusia	Sebagai sarana manusia "memerintah" komputer
3	Terdiri dari sekumpulan kalimat	Terdiri dari sekumpulan perintah
4	Kalimat diakhiri dengan simbol. (titik)	Perintah diakhiri dengan; (titik koma) atau simbol lain sesuai dengan bahasa pemrogramannya
5	Memiliki aturan tata bahasa (*grammar*)	Memiliki aturan tata bahasa program

JENIS-JENIS BAHASA PEMROGRAMAN

Terdapat 4 jenis bahasa pemrograman, yaitu:

1. Bahasa Mesin, yaitu bahasa yang memberikan perintah kepada komputer dengan memakai kode bahasa biner
2. Bahasa Tingkat Rendah, bahasa yang memberikan perintah kepada komputer dengan memakai instruksi-instruksi tingkat rendah. Contoh: Bahasa Rakitan (*Assembly*)
3. Bahasa Tingkat Menengah, bahasa komputer yang memakai campuran instruksi dalam kata-kata bahasa manusia dan instruksi yang bersifat simbolik. Contoh: Bahasa C
4. Bahasa Tingkat Tinggi, bahasa komputer yang memakai instruksi berasal dari unsur kata-kata bahasa manusia. Komputer dapat mengerti bahasa manusia tersebut dengan menggunakan compiler atau interpreter. Contoh: Java, C++, PHP, C#, Visual Basic, Dll.

3.2. VARIABEL

Variabel adalah suatu simbol atau lambang yang mempunyai nilai. Dalam pemrograman, variabel termasuk pengenal (*identifier*).

Secara teknis, variabel adalah area atau tempat di dalam *memory* komputer yang isinya dapat diubah-ubah.

Variabel harus diberi nama yang berbeda satu dengan lainnya, masing-masing variabel memiliki alamatnya sendiri dalam *memory*. Komputer akan dapat menemukan alamat variabel atau alamat data pada *memory* hanya dengan menyebutkan nama variabel pada bahasa pemrograman.

ATURAN PENAMAAN VARIABEL

Nama variabel ditentukan atau dikarang sendiri oleh pembuat program dengan syarat sebagai berikut:

1. Tidak boleh sama dengan nama atau kata yang sudah disiapkan oleh komputer (*reserved word*) seperti *keyword*, dan *function*.
2. Harus berbeda dengan nama label atau konstanta yang dibuat oleh pemrogram.
3. Setiap bahasa pemrograman memiliki maksimum panjang variabel yang berbeda-beda, contohnya Bahasa C memiliki nilai maksimum 32 karakter, bila lebih dari 32 karakter, maka karakter selebihnya tidak diperhatikan komputer.
4. Setiap bahasa pemrograman memiliki aturan khusus penamaan variabel.
   a) *Case Sensitive*, artinya huruf besar dan huruf kecil berbeda
   b) *Case Insensitive*, artinya huruf besar dan huruf kecil sama.
5. Karakter pertama harus huruf atau karakter garis bawah (*underscore*), dan karakter berikutnya boleh huruf, angka, atau karakter garis bawah.
6. Tidak boleh mengandung spasi atau *blank*.

Penamaan variabel yang benar dan salah

PENAMAAN VARIABEL YANG BENAR	PENAMAAN VARIABEL YANG SALAH
A	1A
A1	Nilai-1
Nilai	Harga Satuan
NILAI	Benar/Salah
nilai	Switch
HargaSatuan	Long
Harga_Satuan	Harga-Satuan
HS	
_Harga	
SWITCH	

Pada saat memberikan nama variabel disesuaikan dengan data yang akan disimpan pada variabel, contohnya variabel yang digunakan untuk menyimpan data Luas Segitiga maka bisa diberikan nama variabel Luas atau L.

3.3. KONSTANTA

Konstanta adalah variabel yang nilainya tetap dan tidak dapat diubah selama program berjalan.

Contoh: PI = 3.14

3.4. TIPE DATA

Bentuk data bermacam-macam, ada angka, huruf, dan simbol-simbol lain yang dikenal sebagai karakter. Ada lagi sederetan angka yang menyatakan suatu nilai tertentu seperti 12475 atau 254.75.
Dalam penggunaan komputer umumnya, pemrograman khususnya, data dibedakan dalam empat tipe yaitu:

1. Karakter, adalah tipe data yang disimpan dalam 1 karakter, biasanya data diapit dengan simbol ‘ ‘ (kutip satu). Contoh: char A=’A’, char A=127.

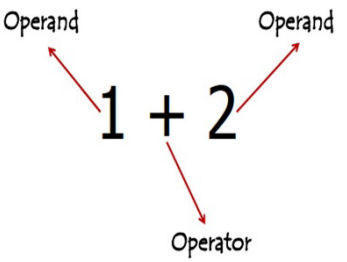
2. Numerik, tipe data numerik dibedakan menjadi dua macam yaitu:
   a) Integer (bilangan bulat), contoh: int A=127, int harga=3000
   b) real (floating-point) yang mengandung bilangan pecahan. Contoh: float Nilai=9.5, float suhu=30.5.
3. String, adalah tipe data yang disimpan terdiri dari banyak karakter yang biasanya diapit dengan simbol " " (kutip dua). Contoh: char A[7]="Jakarta", char B[10]="abcd".
4. Boolean, adalah tipe data yang disimpan dengan nilai True (1) atau False (1).

MEMILIH TIPE DATA

Ketika akan memilih tipe data untuk sebuah variabel harus memperhatikan hal-hal berikut:

1. Perhatikan ukuran dan jangkauan (*range*) tipe data. Sesuai dengan bahasa pemrograman yang digunakan.
2. Sesuaikan kebutuhan data yang akan disimpan.
3. Semakin besar ukuran tipe data, maka program semakin "gemuk".

3.5. OPERATOR



JENIS-JENIS OPERATOR:

1. Operator Penugasan
   Operator penugasan merupakan operator yang digunakan untuk mengisi sebuah variabel atau meng-*assign* suatu nilai kedalam

sebuah variabel. Simbol operator penugasan adalah = (sama dengan).
Contoh:
A = 5, Harga=3000, Grade='A', NilaiAkhir=Absen+Tugas+UTS+UAS

2. Operator Aritmetika

No.	Simbol	Keterangan
1	+	Penjumlahan
2	-	Pengurangan
3	*	Perkalian
4	/	Pembagian
5	%	Sisa hasil pembagian

Dalam sebuah bahasa pemrograman jika dalam suatu ekspresi aritmetika memiliki lebih dari satu operator yang berbeda maka ketika mengerjakan komputer akan dimulai dari tingkatan (hierarki) paling tinggi ke rendah. Adapun urutannya dimulai dari tingkat paling atas sebagai berikut.

No.	Tingkat
1	* (Perkalian) / (Pembagian) % (Modulus) Memiliki tingkat hierarki sederajat dan paling tinggi dibandingkan operator yang lain.
2	+ (Penjumlahan) - (Pengurangan) Memilki tingkat hierarki sederajat pada tingkatan kedua
3	( ) Jika terdapat tanda kurung adalah satu kesatuan

3. Operator Hubungan (Perbandingan)

No.	Operator	Arti	Contoh	Keterangan
1	<	Kurang dari	X < Y	Apakah X kurang dari Y
2	<=	Kurang dari sama dengan	X <= Y	Apakah X kurang dari sama dengan Y
3	>	Lebih dari	X > Y	Apakah X lebih dari Y
4	>=	Lebih dari sama dengan	X >= Y	Apakah X lebih besar sama dengan Y
5	==	Sama dengan	X == Y	Apakah X sama dengan Y
6	!=	Tidak sama dengan	X != Y	Apakah X tidak sama dengan Y

4. Operator Logika

No.	Simbol	Arti
1	&& atau AND	Logika AND (DAN)
2	|| atau OR	Logika OR (ATAU)
3	! atau NOT	Logika NOT (INGKARAN)

Tabel kebenaran Logika OR

No.	Kondisi 1	Kondisi 2	Kondisi 1 OR Kondisi 2	Hasil
1	True	True	True OR True	True
2	True	False	True OR False	True
3	False	True	False OR True	True
4	False	False	False OR False	False

Tabel kebenaran Logika AND

No.	Kondisi 1	Kondisi 2	Kondisi 1 AND Kondisi 2	Hasil
1	True	True	True AND True	True
2	True	False	True AND False	False
3	False	True	False AND True	False
4	False	False	False AND False	False

5. Operator Bitwise

No.	Operator	Arti
1	<<	Pergeseran bit ke kiri
2	>>	Pergeseran bit ke kanan
3	&	Bitwise AND
4	^	Bitwise XOR (Exclusive OR)
5	|	Bitwise OR
6	~	Bitwise NOT

6. Operator Unary

No.	Operator	Arti/Maksud	Letak	Contoh	Equivalen
1	-	Unary minus	Sebelum operator	A + -B * C	A + (-B) * C
2	++	Peningkatan dengan penambahan nilai 1	Sebelum dan sesudah	A++	A=A+1
3	--	Penurunan dengan pengurangan 1	Sebelum dan sesudah	A--	A=A=1
4	sizeof	Ukuran operand dalam byte	Sebelum	Sizeof(I)	-
5	!	Unary Not	Sebelum	!A	-

No.	Operator	Arti/Maksud	Letak	Contoh	Equivalen
6	~	Bitwise Not	Sebelum	~A	-
7	&	Menghasilkan alamat *memory* operand	Sebelum	&A	-
8	*	Menghasilkan nilai dari *pointer*	Sebelum	*A	-

KESIMPULAN

Dalam membuat perintah pada bahasa pemrograman maka perlu diperhatikan komponen-komponen yang terkait dalam bahasa pemrograman seperti:

1. Variabel
2. Tipe Data
3. Konstanta
4. Operator

SOAL LATIHAN

1. Buatlah algoritma/*flowchart* untuk menginputkan 3 buah bilangan bulat dan tampilkan bilangan TERBESAR di antara ketiganya (dianggap ketiga bilangan nilainya berbeda). **TIDAK BOLEH MENGGUNAKAN OPERATOR LOGIKA.**
2. Buatlah algoritma/*flowchart* untuk menginputkan 3 buah bilangan bulat dan tampilkan bilangan TERBESAR di antara ketiganya (dianggap ketiga bilangan nilainya berbeda). **BOLEH MENGGUNAKAN OPERATOR LOGIKA**
3. Apa yang tercetak bila program di bawah ini dijalankan?

a

```c
#include "stdio.h"
void main()
{
    int A=7, B, C;
    B=A/2;
    C=A%2;
    printf("%i",B);
    printf("%i",C);
}
```

b

```c
#include "stdio.h"
void main()
{
    int A=25, B, C;
    B=A/2;
    C=A-B*2;
    printf("%i",C);
    A=B;
    B=A/2;
    C=A-B*2;
    printf("%i",C);
    A=B;
    B=A/2;
    C=A-B*2;
    printf("%i",C);
}
```

BAB 4

STRUKTUR KONDISI

Capaian Pembelajaran	:	Mahasiswa memahami bentuk umum dan penggunaan struktur kontrol percabangan: IF-THEN dan IF-THEN-ELSE.

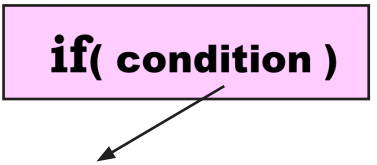
Subpokok Bahasan	:	4.1. Syntax Statement If 4.2. Bentuk Umum Statement IF-THEN 4.3. Contoh Penggunaan Statement IF-THEN 4.4. Bentuk Umum Statement IF-THEN-ELSE 4.5. Contoh Penggunaan Statement IF-THEN-ELSE 4.6. Contoh Penggunaan Struktur Kontrol Percabangan

Daftar Pustaka	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

STRUKTUR KONDISI

4.1. SYNTAX STATEMENT IF

Syntax merupakan tata bahasa. Penulisan instruksi-instruksi dalam sebuah program sama dengan menulis sebuah surat yaitu ditulis dengan tersusun dari atas ke bawah dan dari kiri ke kanan. Pelaksanaan instruksi-instruksinya tentu demikian pula dari atas ke bawah dan dari kiri ke kanan. Urutan proses demikian disebut dengan *sequence*. Tapi terkadang, pada suatu kondisi tertentu, diperlukan sebuah percabangan (*branching*), tidak mengerjakan suatu blok instruksi tertentu tetapi langsung mengerjakan instruksi yang ada di 'bawah' blok instruksi tersebut. Atau pada suatu kondisi tertentu dilakukan pemilihan (*selection*) yaitu memilih mengerjakan salah satu dari dua buah blok instruksi tertentu.



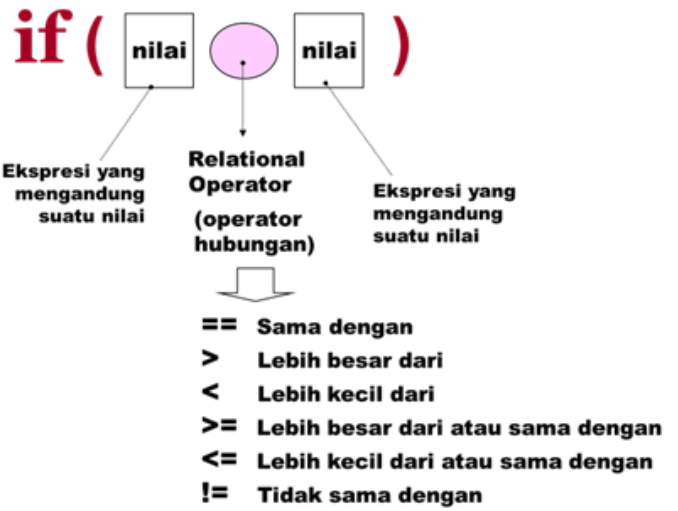
Condition: apapun yang ditulis pada *condition*, bila dimengerti oleh komputer, akan dianggap sebagai suatu kondisi. Kondisi adalah suatu pernyataan, atau ekspresi yang mengandung nilai benar (TRUE) atau salah (FALSE).

Contoh *Condition*:
if(A>B) dibaca A lebih besar dari B
if(A<=5) dibaca A lebih kecil sama dengan 5

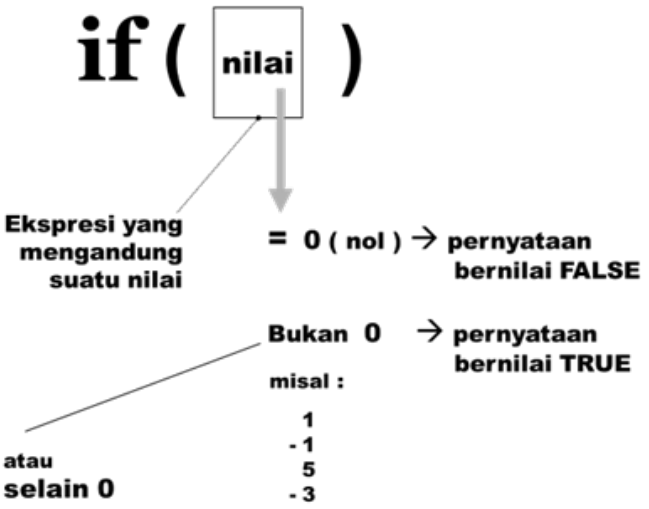
JENIS *SYNTAX*

Syntax percabangan terdapat dua bentuk yaitu:

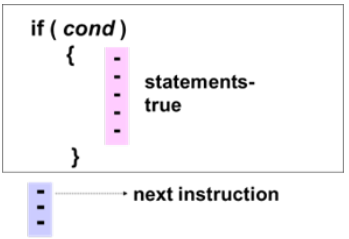
1. Bentuk Pertama, kondisi yang menganalisa hubungan dua buah nilai.



2. Bentuk Kedua, kondisi yang menghasilkan nilai 0 (nol) atau bukan nol.



4.2. BENTUK UMUM STATEMENT IF-THEN



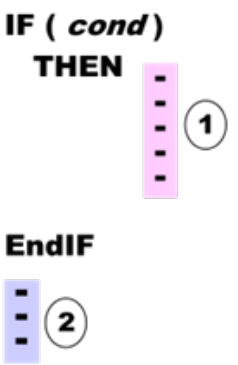
cond: *condition*, yang mempunyai nilai TRUE atau FALSE

Statement-true: terdiri dari satu atau lebih dari satu statement atau instruksi. Minimal satu statement. Instruksi-instruksi dibuat menjadi sebuah blok statement yang diapit oleh kurung { dan }

CARA KERJA

1. Memeriksa nilai cond
2. * Bila cond bernilai TRUE, maka kerjakan statement-statement yang berada dalam blok statement-true. Setelah selesai mengerjakan semua statement yang ada dalam blok statement-true, langsung melanjutkan mengerjakan next-instruction
   * Bila cond bernilai FALSE, maka langsung 'meloncat' mengerjakan next-instruction

PSEUDOCODE

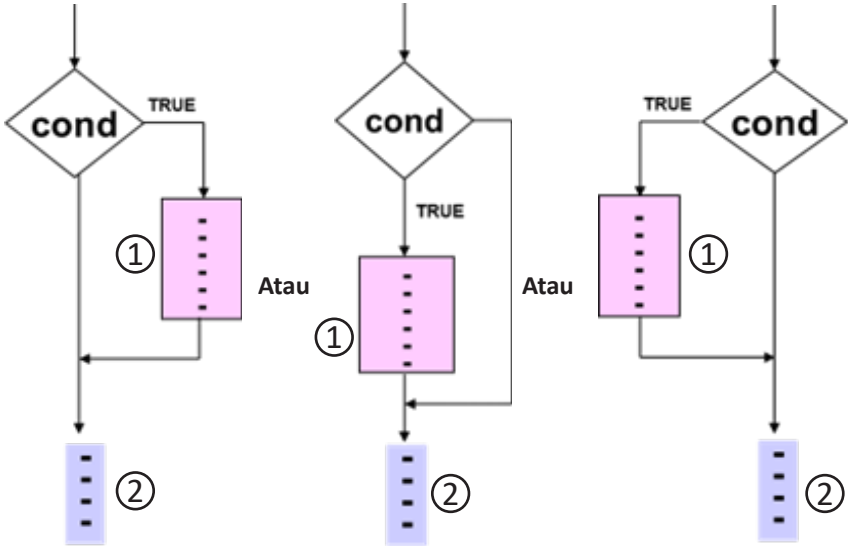


Keterangan

① Statement-True

② Next-Instruction

FLOWCHART IF-THEN



Yang umum di pakai adalah bentuk di atas

Keterangan:

① Statement-True

② Next-Instruction

4.3. CONTOH PENGGUNAAN STATEMENT IF-THEN

SOAL-1

Gambarkan *flowchart* dari penggalan program-1 sampai dengan program-7 di bawah ini:

1
```
int A=5, B=7;
if(A<B)
  {
    printf("Jakarta");
  }
printf("\nBandung");
```

2
```
int A=5, B=7;
if(A<B)
  { printf("Jakarta");
  }
printf("\nBandung");
```

3
```
int A=5, B=7;
if(A<B)
  { printf("Jakarta");}
printf("\nBandung");
```

4
```
int A=5, B=7;
if(A<B)
   printf("Jakarta");
printf("\nBandung");
```

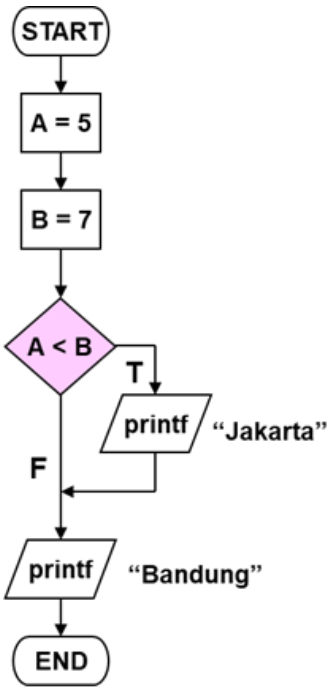
5
```
int A=5, B=7;
if(A<B)printf("Jakarta");
printf("\nBandung");
```

Nomor 6 dan 7 cara menulis yang tidak baik walaupun bagi komputer tidak ada bedanya dengan program nomor 1,2,3,4 dan 5

6
```
int A=5, B=7;
if(A<B)
      printf("Jakarta");
      printf("Bandung");
```

7
```
int A=5, B=7;
if(A<B)
printf("Jakarta");
printf("Bandung");
```

Jawab:
Semua program nomor 1 sampai dengan nomor 7 *flowchart*-nya sama:



SOAL-2

Sebuah toko memberikan potongan harga yang harus dibayar sebesar Rp10, bila nilai belanjaan lebih besar dari Rp100. Dari soal kasus tersebut maka dapat dibuatkan penggalan program sebagai berikut:

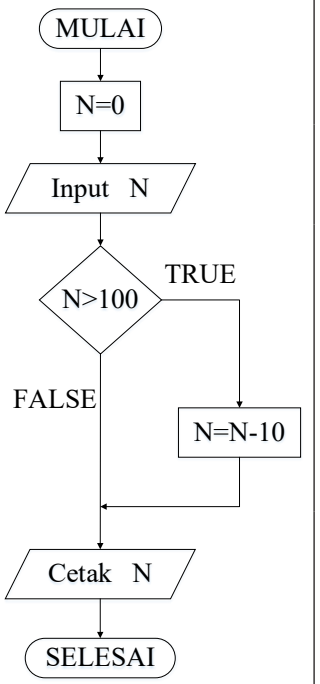
```
1  #include<stdio.h>
2  void main()
3  {
4      int N=0;
5      scanf("%i",&N);
6      if(N>100)
7          N=N-10;
8      printf("%i",N);
9  }
```

Berdasarkan penggalan program di samping, apa yang tercetak bila diinput untuk Nilai N:

a. 100
b. 105
c. 98

Gambarkan juga *flowchart* dari penggalan program di samping.

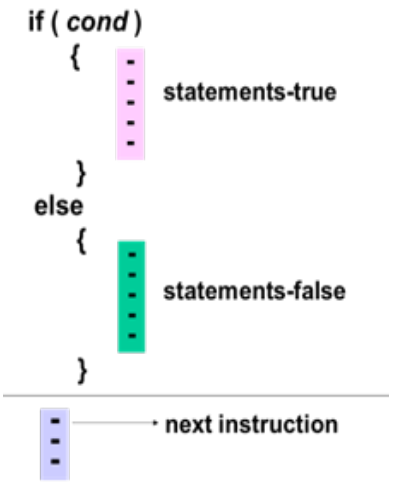
Jawab:

No.	Penggalan Program	Simbol *Flowchart*	*Flowchart*
1.	int N (inisialisasi N=0)	N=0	
2.	scanf("%i",&N);	Input N	
3.	if(N>100)	N>100	
4.	N=N-10	N=N-10	
5.	printf("%i",N)	Cetak N	

Yang tercetak sesuai dengan Nilai N:

No.	Nilai N	Tercetak
a.	100	100
b.	105	95
c.	98	98

4.4. BENTUK UMUM STATEMENT IF-THEN-ELSE



cond: *condition*, yang mempunyai nilai TRUE atau FALSE

Blok statement-true: terdiri dari satu atau lebih dari satu statement. Minimal satu statement. Instruksi-instruksi dibuat menjadi sebuah blok statement yang diapit oleh kurung { dan }

Blok statement-false: terdiri dari satu atau lebih dari satu statement. Minimal satu statement. Instruksi-instruksi dibuat menjadi sebuah blok statement yang diapit oleh kurng { dan }

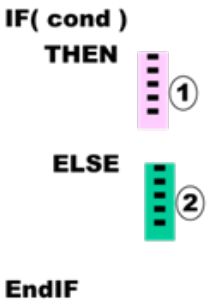
CARA KERJA:

a. Memeriksa nilai cond

b. * Bila cond bernilai TRUE, maka kerjakan statement-statement yang berada dalam blok statement-true. Setelah selesai mengerjakan semua statement yang ada dalam blok statement-true, maka langsung 'meloncat' mengerjakan next-instruction

* Bila cond bernilai FALSE, maka kerjakan statement-statement yang berada dalam blok statement-false. Setelah selesai mengerjakan statement dalam blok ini, maka langsung meloncat mengerjakan next-instruction

PSEUDOCODE

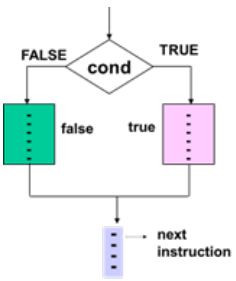


Keterangan

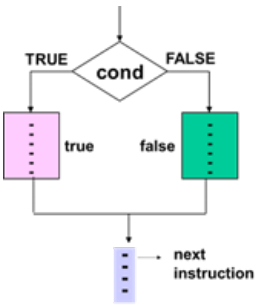
① Statement-True

② Statement-False

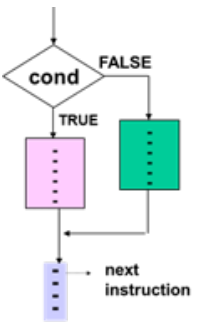
FLOWCHART



Atau



Atau



Yang umum di pakai adalah bentuk di atas

4.5. CONTOH PENGGUNAAN STATEMENT IF THEN ELSE

SOAL-1

Gambarkan *flowchart* dari penggalan program-1 sampai dengan program-7 di bawah ini:

1
```
int A=5, B=7;
if(A<B)
  {
    printf("Jakarta");
  }
else
   {
     printf("Bandung");
   }
printf("\nSurabaya");
```

2
```
int A=5, B=7;
if(A<B)
  { printf("Jakarta");
  }
else
   {printf("Bandung");
   }
printf("\nSurabaya");
```

3
```
int A=5, B=7;
if(A<B)
  { printf("Jakarta");}
else
  {printf("Bandung");}
printf("\nSurabaya");
```

4
```
int A=5, B=7;
if(A<B)
   printf("Jakarta");
else
   printf("Bandung");
printf("\nSurabaya");
```

5
```
int A=5, B=7;
if(A<B) { printf("Jakarta");}
else {printf("Bandung");}
printf("\nSurabaya");
```

6.
```
int A=5, B=7;
if(A<B) printf("Jakarta"); else printf("Bandung");
printf("\nSelesai");
```

7.
```
int A=5, B=7;
if(A<B) printf("Jakarta"); else printf("Bandung"); printf("\nSelesai");
```

8.
```
int A=5, B=7; if(A<B) printf("Jakarta"); else printf("Bandung"); printf("\nSelesai");
```

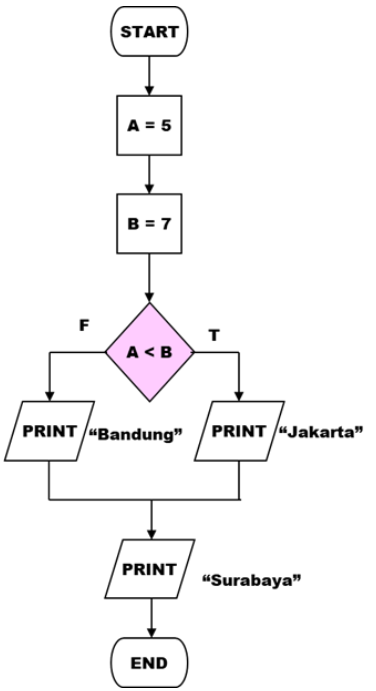
Cara menulis TIDAK BAIK walaupun bagi komputer tidak ada bedanya dengan cara yang lain

9
```
int A=5, B=7;
if(A<B)
  printf("Jakarta");
else
  printf("Bandung");
  printf("\nSurabaya");
```

10
```
int A=5, B=7;
if(A<B)
printf("Jakarta");
else
printf("Bandung");
printf("\nSurabaya");
```

Jawab:

Semua program nomor 1 – nomor 10 memiliki *flowchart* yang sama:

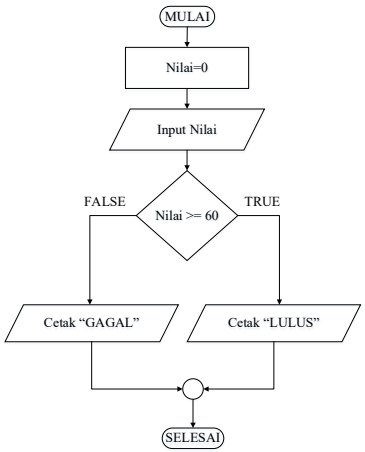


SOAL-2

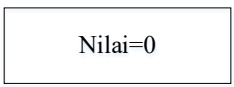
Susun penggalan program dan *flowchart* untuk menginput sebuah nilai integer (nilai ujian mahasiswa) kemudian cetak perkataan "LULUS" bila nilai tersebut >= 60 atau cetak perkataan "GAGAL" bila nilai tersebut < 60.

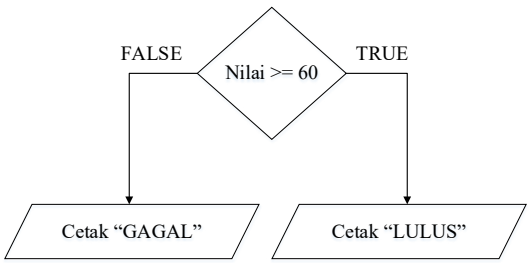
Jawab:

JAWABAN SOAL-2 CARA – 1

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial Nilai	Inisialisasi/ Proses	
2.	Input Nilai	Input/Output	
3.	Nilai >= 60 , cetak “LULUS” Jika tidak, cetak “GAGAL”	*Decision*, Input/Output	

Penggalan Program Cara – 1

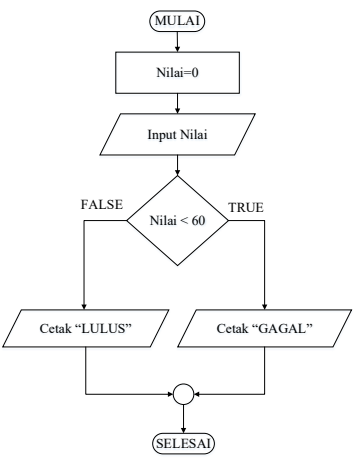
No.	Simbol *Flowchart*	Penggalan Program
1.		int Nilai=0
2.		scanf(“%i”,&Nilai)

3.		if(Nilai>=60)
		  printf(“LULUS”)
		else
		  printf(“GAGAL”)

Program 4.1 Implementasi Program Cara – 1

```
#include<stdio.h>
void main()
{
    int Nilai=0;
    scanf("%i",&Nilai);
    if(Nilai>=60)
        printf("LULUS");
    else
        printf("GAGAL");
}
```

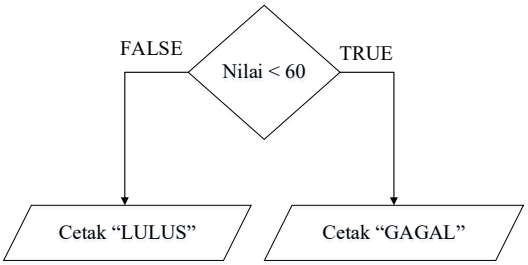
JAWABAN SOAL-3 CARA – 2

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial Nilai	Inisialisasi/ Proses	
2.	Input Nilai	Input/Output	
3.	Jika Nilai < 60 , cetak "GAGAL" Jika tidak, cetak "LULUS"	*Decision*, Input/Output	

Penggalan Program Cara – 2

No.	Simbol *Flowchart*	Penggalan Program
1.	Nilai=0	int Nilai=0

No.	Simbol *Flowchart*	Penggalan Program
2.		scanf("%i",&Nilai)

3.		if(Nilai<60)
		printf("GAGAL")
		else
		printf("LULUS")

Program 4.2 Implementasi Program Cara – 2

```
1  #include<stdio.h>
2  void main()
3  {
4      int Nilai=0;
5      scanf("%i",&Nilai);
6      if(Nilai<60)
7          printf("GAGAL");
8      else
9          printf("LULUS");
10 }
```

4.6. CONTOH PENGGUNAAN STRUKTUR KONTROL PERCABANGAN

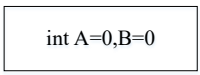
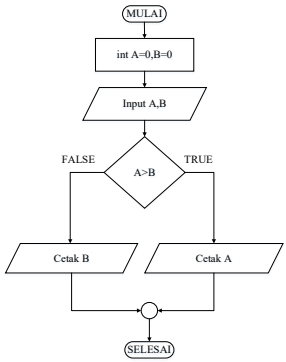
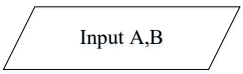
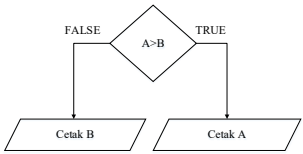
SOAL-1

Susun penggalan program dan *flowchart* untuk menginput dua buah bilangan bulat yang nilainya tidak sama kemudian mencetak salah satu bilangan yang nilainya terbesar.

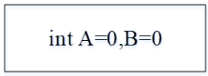
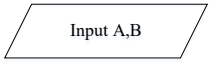
Jawab:

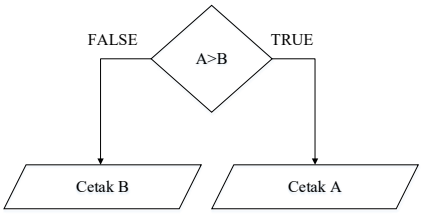
JAWABAN SOAL-1 CARA – 1

Cara pertama dengan menggunakan dua variabel untuk menyimpan bilangan, kemudian kedua variabel tersebut dibandingkan dengan menggunakan kondisi untuk mencari mana yang nilainya terbesar.

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial A,B		
2.	Input Nilai A, B		
3.	Jika A>B , cetak Nilai A Jika tidak, cetak Nilai B		

Penggalan Program Cara – 1

No.	Simbol *Flowchart*	Penggalan Program
1.		int A=0,B=0
2.		scanf(“%i”,&A) scanf(“%i”,&B)

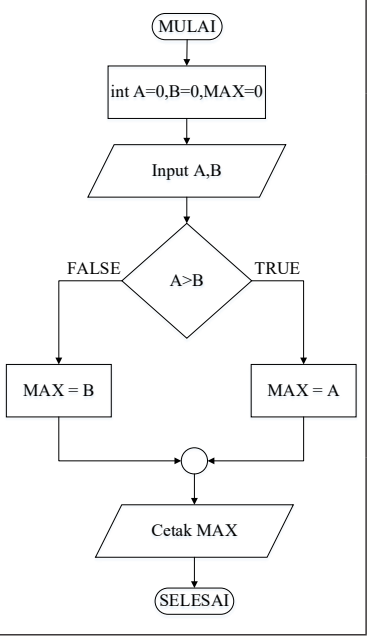
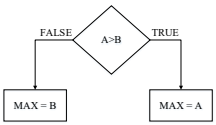
3.		if(A>B)
		printf(“%i”,A)
		else
		printf(“%i”,B)

Program 4.3 Implementasi Program Cara – 1

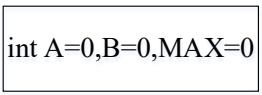
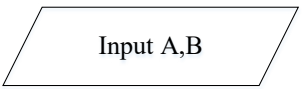
```
#include<stdio.h>
void main()
{
    int A=0,B=0;
    scanf("%i",&A);
    scanf("%i",&B);
    if(A>B)
        printf("%i",A);
    else
        printf("%i",B);
}
```

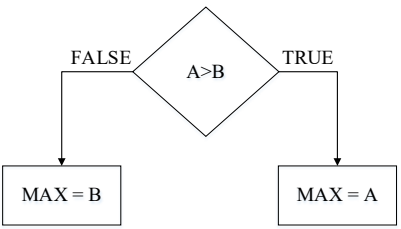
JAWABAN SOAL – 1 CARA – 2

Cara kedua dengan menggunakan tiga variabel, dua variabel untuk menyimpan bilangan yang diinput, satu variabel untuk menyimpan bilangan terbesar (MAX).

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial A,B, MAX	int A=0,B=0,MAX=0	
2.	Input A, B	Input A,B	
3.	Jika A>B , Isi MAX dengan Nilai A Jika tidak, Isi MAX dengan Nilai B		
4.	Cetak MAX	Cetak MAX	

Penggalan Program Cara – 2

No.	Simbol *Flowchart*	Penggalan Program
1.		int A=0,B=0, MAX=0
2.		scanf(“%i”,&A) scanf(“%i”,&B)

3.		if(A>B)
		MAX = A
		else
		MAX = B

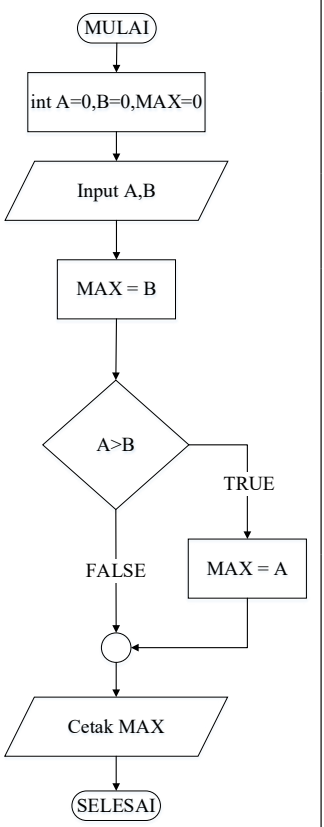
4.		printf(“%i”,MAX)

Program 4.4 Implementasi Program Cara – 2

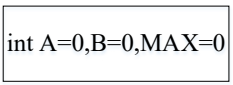
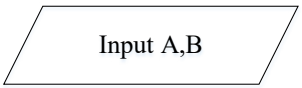
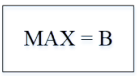
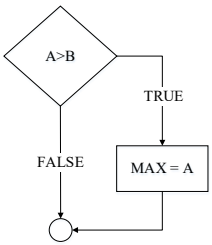
```
1  #include<stdio.h>
2  void main()
3  {
4      int A=0,B=0,MAX=0;
5      scanf("%i",&A);
6      scanf("%i",&B);
7      if(A>B)
8          MAX=A;
9      else
10         MAX=B;
11     printf("%i",MAX);
12 }
```

JAWABAN SOAL-1 CARA – 3

Cara ketika hampir mirip seperti cara yang kedua, yaitu dengan menggunakan dua variabel untuk menyimpan bilangan yang akan diinput, dan satu variabel untuk menyimpan bilangan terbesar (MAX). Perbedaan cara ketiga dengan cara kedua adalah sebelum dilakukan perbandingan antara kedua bilangan, salah satu bilangan disimpan terlebih dahulu pada MAX (yang disimpan adalah MAX=B).

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial A,B, MAX	int A=0,B=0,MAX=0	
2.	Input A, B	Input A,B	
3.	Isi MAX dengan Nilai B	MAX = B	
4.	Jika A>B , MAX dengan Nilai A	A>B, TRUE, FALSE, MAX = A	
5.	Cetak MAX	Cetak MAX	

Penggalan Program Cara – 3

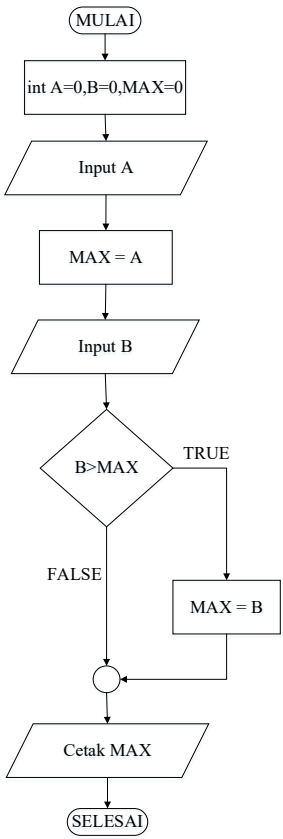
No.	Simbol *Flowchart*	Penggalan Program
1.		int A=0,B=0, MAX=0
2.		scanf("%i",&A) scanf("%i",&B)
3.		MAX=B
4.		if(A>B) MAX = A
5.		printf("%i",MAX)

Program 4.5 Implementasi Program Cara – 3

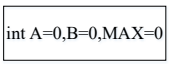
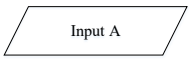
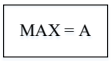
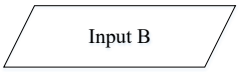
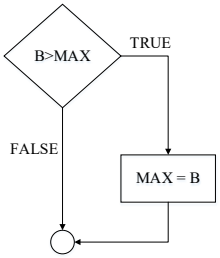
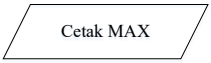
```
1  #include<stdio.h>
2  void main()
3  {
4      int A=0,B=0,MAX=0;
5      scanf("%i",&A);
6      scanf("%i",&B);
7      MAX=B;
8      if(A>B)
9          MAX=A;
10     printf("%i",MAX);
11 }
```

JAWABAN SOAL – 1 CARA – 4

Cara keempat hampir sama dengan cara ketiga. Tetap menggunakan tiga variabel yaitu dua variabel untuk menyimpan bilangan yang diinput dan satu variabel untuk menyimpan nilai Maksimal (MAX). Perbedaan cara ketiga dengan cara keempat adalah yang disimpan pada variabel MAX adalah nilai A (MAX=A).

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial A,B, MAX	int A=0,B=0,MAX=0	
2.	Input A	Input A	
3.	Isi nilai MAX dengan A	MAX = A	
4.	Input B	Input B	
5.	Jika B>MAX , Isi MAX dengan Nilai B	B>MAX, TRUE, FALSE, MAX = B	
6.	Cetak MAX	Cetak MAX	

Penggalan Program Cara – 4

No.	Simbol *Flowchart*	Penggalan Program
1.		int A,B, MAX
2.		scanf("%i",&A)
3.		MAX=A
4.		scanf("%i",&B)
5.		if(B>MAX) MAX = B
6.		printf("%i",MAX)

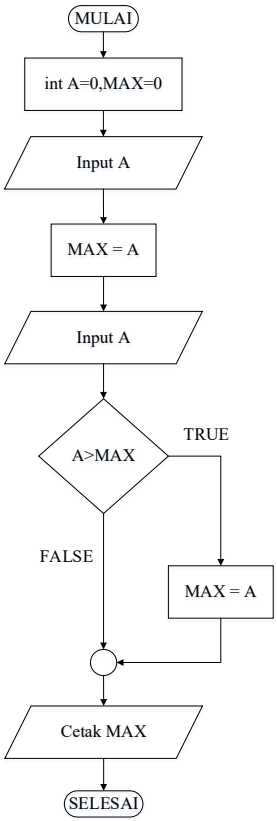
Program 4.6 Implementasi Program Cara – 4

```
1  #include<stdio.h>
2  void main()
3  {
4      int A=0,B=0,MAX=0;
5      scanf("%i",&A);
6      MAX=A;
7      scanf("%i",&B);
8      if(B>MAX)
9          MAX=B;
10     printf("%i",MAX);
11 }
```

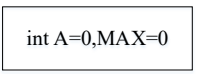
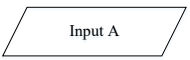
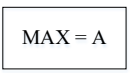
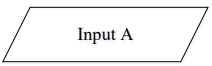
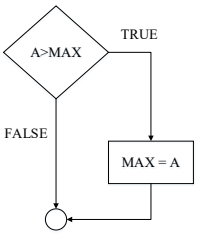
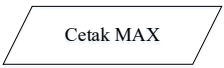
JAWABAN SOAL – 1 CARA – 5

Algoritma Cara-5 ini yang nanti akan menjadi dasar algoritma pencarian bilangan terbesar ataupun terkecil dari sekian buah bilangan yang ada.

Pada cara kelima ini menggunakan dua variabel, satu variabel untuk menyimpan bilangan yang diinput, dan satu variabel untuk menyimpan nilai Maksimal. Setiap bilangan diinput dan disimpan ke variabel A maka akan dilakukan pemeriksaan kondisi apakah bilangan yang baru diinput lebih besar dari nilai Maksimal. Karena jumlah bilangan yang diperiksa sebanyak 3 bilangan, maka input dan pengecekan dilakukan sebanyak 3 kali juga.

No.	*Pseudocode*	Simbol *Flowchart*	*Flowchart*
1.	Inisial A, MAX	int A=0,MAX=0	
2.	Input A	Input A	
3.	Isi MAX dengan Nilai A	MAX = A	
4.	Input A	Input A	
5.	Jika A>MAX , Isi MAX dengan Nilai A	A>MAX, TRUE, FALSE, MAX = A	
6.	Cetak MAX	Cetak MAX	

Penggalan Program Cara – 5

No.	Simbol *Flowchart*	Penggalan Program
1.		int A=0, MAX=0
2.		scanf(“%i”,&A)
3.		MAX=A
4.		scanf(“%i”,&A)
5.		if(A>MAX) MAX = A
6.		printf(“%i”,MAX)

Program 4.7 Implementasi Program Cara – 5

```
1  #include<stdio.h>
2  void main()
3  {
4      int A=0,MAX=0;
5      scanf("%i",&A);
6      MAX=A;
7      scanf("%i",&A);
8      if(A>MAX)
9          MAX=A;
10     printf("%i",MAX);
11 }
```

KESIMPULAN

Dalam mendesain sebuah Algoritma, di mana Algoritma tersebut terdapat pilihan maka dapat menggunakan Struktur Kontrol Percabangan, yang dibagi menjadi dua, yaitu:

1. IF-THEN
2. IF-THEN-ELSE

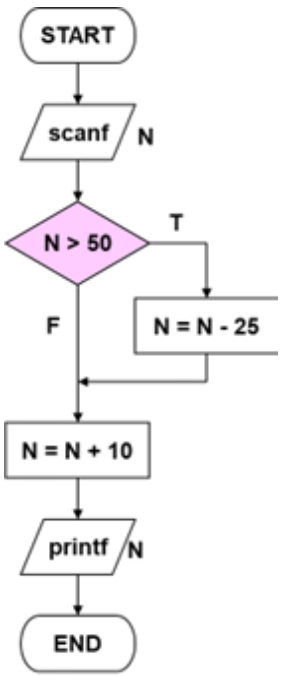
Setiap struktur kontrol IF-THEN serta IF-THEN-ELSE memiliki bentuk penulisan pada program, bentuk *pseudocode*, serta gambar *flowchart* yang berbeda antara satu dengan yang lainnya.

Salah satu contoh penggunaan Struktur Kontrol Percabangan yaitu mencari nilai terbesar/terkecil dari dua buah bilangan.

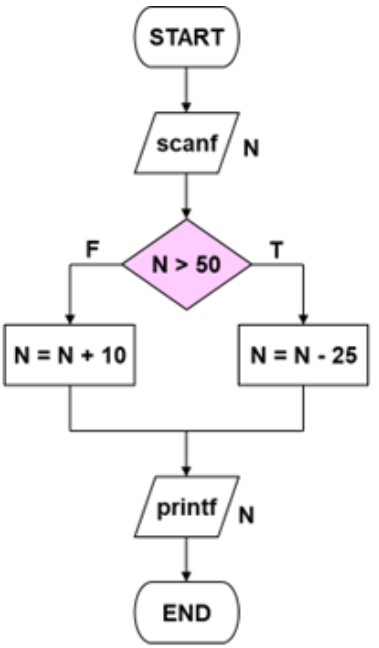
SOAL LATIHAN

1. Tuliskan Program (penggalan program) untuk menyatakan algoritma digambarkan pada *flowchart* berikut:

a



b



Pada *Flowchart* 1.a Apa yang tercetak bila Nilai N diinputkan nilai:

a. 30
b. 50
c. 65

2. Pada *Flowchart* 1.b Apa yang tercetak bila Nilai N diinputkan nilai:
   a. 30
   b. 50
   c. 65

3. Susun algoritma (program) untuk menginput 3 buah bilangan yang masing-masing menyatakan panjang sisi sebuah segitiga. Kemudian periksa ketiga buah garis (sisi) tersebut. Bila ketiga buah garis (sisi) tersebut panjangnya sama maka cetak perkataan "SAMA SISI". Bila hanya dua sisi yang sama maka cetak perkataan "SAMA KAKI ". Tapi bila ketiga-tiganya tidak sama maka cetak perkataan "SEMBARANG". Tidak boleh menggunakan *logical operator* AND dan OR

4. Susun program untuk menginput tiga buah bilangan yang menyatakan nilai ujian tiga buah mata kuliah.
   a. Cetak perkataan "TIGA" bila ketiga mata kuliah tersebut mendapat nilai lulus.
   b. Cetak perkataan "DUA", bila hanya dua dari ketiga mata kuliah tersebut yang mendapat nilai lulus.
   c. Cetak perkataan "SATU" bila hanya satu mata kuliah yang mendapat nilai lulus.
   d. Cetak perkataan "NOL" bila ketiga mata kuliah tersebut dinyatakan tidak lulus.

   Sebuah mata kuliah dinyatakan mendapat nilai lulus bila nilainya lebih besar atau sama dengan 60.

BAB 5

STRUKTUR KONDISI LANJUTAN

Capaian Pembelajaran	:	Mahasiswa memahami bentuk umum dan penggunaan struktur kondisi IF bertingkat dan Switch Case.

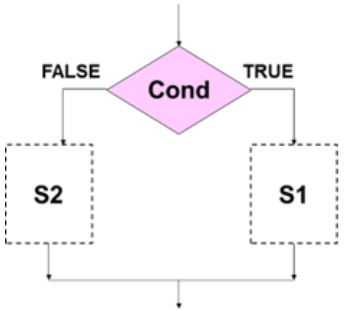
Subpokok Bahasan	:	5.1. Nested If 5.2. Bentuk Nested If *5.3. Multi Condition* dan *Logical Operator* 5.4. Jenis Operator Logika 5.5. Konversi *Multi Condition* Menjadi Nested-If 5.6. Contoh Program Sederhana Menggunakan Nested If dan *Multi Condition* 5.7. Seleksi Menggunakan Switch-Case 5.8. Swtich-Case Berjenjang

Daftar Pustaka	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

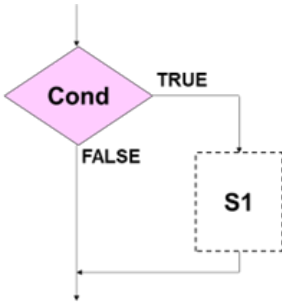
STRUKTUR KONDISI LANJUTAN

5.1. NESTED IF

Perhatikan kembali struktur IF-THEN dan IF-THEN-ELSE statement seperti di bawah ini:



Struktur
IF-THEN-ELSE



Struktur
IF-THEN

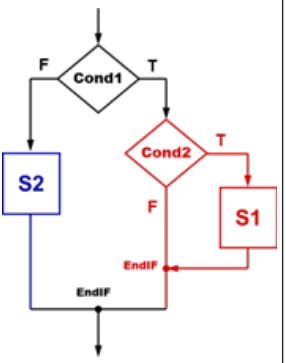
Dari ilustrasi di atas, terlihat S adalah satu atau sekelompok statement. Di dalam kelompok S mungkin terdapat statement IF sehingga terjadi IF secara berjenjang atau secara tersarang yang biasa disebut Nested If (*nest*=sarang)

5.2. BENTUK NESTED IF

Bentuk-bentuk NESTED-IF (If bersarang) dapat dilihat pada contoh-contoh di bawah ini. Contoh-contoh tersebut merupakan salah satu di antara beberapa cara yang bisa dilakukan dalam membuat Nested-If selama mengikuti aturan penggambaran *Flowchart* pada IF-THEN dan IF-THEN-ELSE.

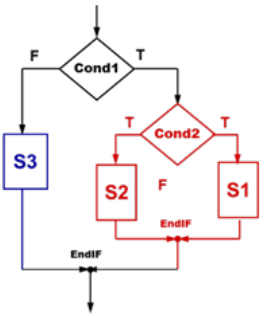
CONTOH 1

Penggalan Program	*Pseudocode*	*Flowchart*

if (cond1)
{ if (cond2)
{ -
- S1
-
}
}
else
{ -
- S2
-
}
	IF (cond1)
	THEN IF(cond2)
	THEN
	EndIF
	ELSE
	EndIF
		

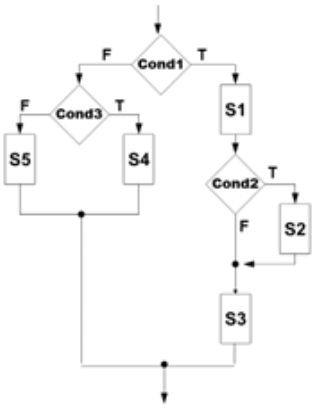
CONTOH 2

Penggalan Program	*Pseudocode*	*Flowchart*

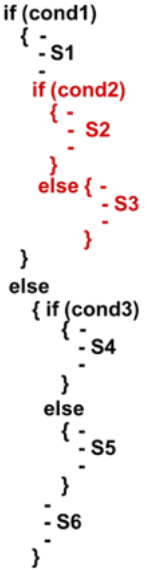
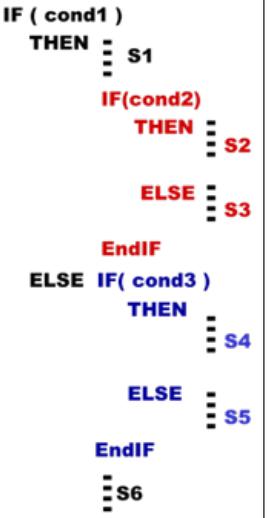
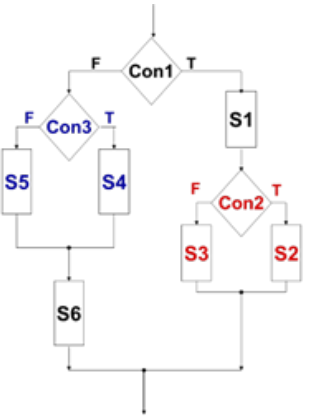
if (cond1)
{ if (cond2)
{ -
- S1
-
}
else { -
- S2
-
}
}
else { -
- S3
-
}
	IF (cond1)
	THEN IF(cond2)
	THEN
	ELSE
	EndIF
	ELSE
	EndIF
		

CONTOH 3

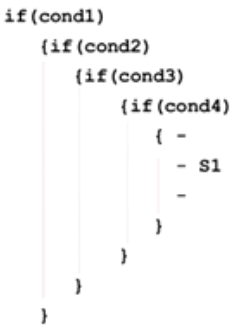
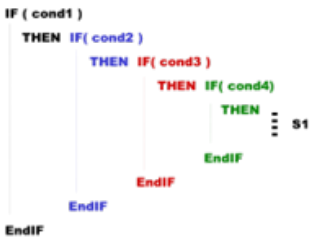
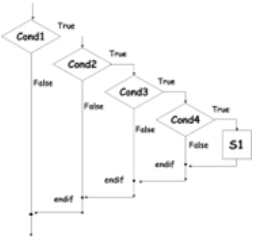
Penggalan Program	*Pseudocode*	*Flowchart*

if cond1
{ -
- S1
-
if cond2
{ -
- S2
-
}
-
- S3
-
}
else
{ if cond3
{ -
- S4
-
}
else
{ -
- S5
-
}
}
	IF (cond1)
	THEN
	S1
	IF(cond2)
	THEN
	S2
	EndIF
	S3
	ELSE IF(cond3)
	THEN
	S4
	ELSE
	S5
	EndIF
	EndIF
		
		Perhatikan posisi letak 'titik' Endif (akhir fungsi if) dalam *flowchart*.
		Posisi ini penting untuk menganalisa aliran terutama untuk nested IF yang kompleks atau untuk proses pengulangan yang bersifat rekursif.

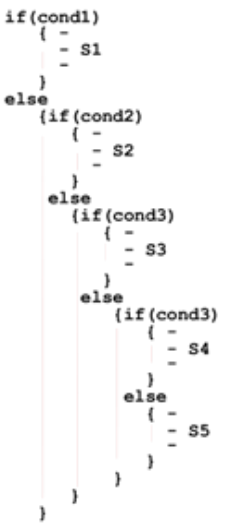
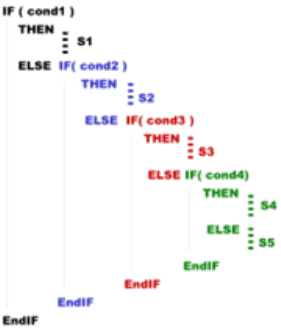
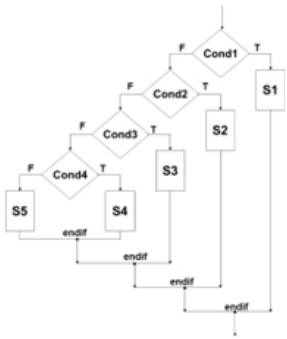
CONTOH 4

Penggalan Program	*Pseudocode*	*Flowchart*
		

CONTOH 5

Penggalan Program	*Pseudocode*	*Flowchart*
		 S1 dilaksanakan, hanya bila keempat kondisi nilainya TRUE

CONTOH 6

Penggalan Program	*Pseudocode*	*Flowchart*
		

5.3. *MULTI CONDITION* DAN *LOGICAL OPERATOR*

Kadang-kadang satu kondisi saja tidak cukup untuk menentukan suatu syarat, sehingga diperlukan dua atau lebih kondisi. Untuk menggabung kondisi-kondisi tersebut gunakan operator logika yang disebut *logical operator*.

5.4. JENIS OPERATOR LOGIKA

OPERATOR NOT

Operator NOT, bukan digunakan untuk menggabungkan dua buah kondisi tepi bekerja sebagai pembalik nilai logika TRUE menjadi FALSE, FALSE menjadi TRUE sehingga sering disebut Unary Operator.

No.	A	B	*Condition*	Nilai	Not (*Condition*)	Sama Maksudnya	Nilai
1.	5	2	A == B	False	! (A == B)	A != B	True
2.	5	2	A > B	True	! (A > B)	A <= B	False

No.	A	B	*Condition*	Nilai	Not (*Condition*)	Sama Maksudnya	Nilai
3.	5	2	A < B	False	! (A < B)	A >= B	True
4.	5	2	A >= B	True	! (A >= B)	A < B	False
5.	5	2	A <= B	False	! (A <= B)	A > B	True
6.	5	2	A != B	True	! (A != B)	A == B	False

OPERATOR AND

Operator AND menggabungkan dua buah kondisi menjadi satu nilai sedemikian rupa akan bernilai TRUE hanya bila kedua kondisi yang digabungkan bernilai TRUE. Dengan istilah lain, kedua syarat harus dipenuhi.

Syntax Operator AND pada kondisi: if (cond1 && cond2) atau if ((cond1) && (cond2))

Contoh: if (Kode==1 && Umur<=25) atau if(Nilai >=60 && Nilai<70)

Tabel kebenaran untuk operator AND

No.	Cond1	Cond2	Cond1 AND Cond2
1.	True	True	True
2.	True	False	False
3.	False	True	False
4.	False	False	False

Perhatikan tabel kebenaran No. 1 hanya bila kedua kondisi bernilai TRUE, yang akan menghasilkan nilai gabungan = TRUE

Contoh tabel kebenaran untuk melihat nilai gabungan dua buah kondisi yang digabung dengan operator AND

No.	A	B	C	D	Cond1	Cond2	Cond1 && Cond2	Nilai Akhir
1.	5	2	6	4	A>B (True)	C>D (True)	True && True	True

No.	A	B	C	D	Cond1	Cond2	Cond1 && Cond2	Nilai Akhir
2.	5	2	6	4	A>B (True)	B>D (False)	True && False	False
3.	5	2	6	4	A>C (False)	B>D (False)	False && False	False
4.	5	2	6	4	A>B (True)	!(B>D) (True)	True && True	True

OPERATOR OR

Operator OR menggabungkan dua buah kondisi menjadi satu nilai sedemikian rupa akan bernilai TRUE cukup bila salah satu saja dari kedua kondisi yang digabungkan bernilai TRUE. Hanya bila kedua kondisi bernilai FALSE, maka nilai gabungannya bernilai FALSE. Dengan perkataan lain cukup satu syarat saja yang harus dipenuhi.

Syntax Operator OR pada kondisi: if (cond1 || cond2) atau if ((cond1) || (cond2))

Contoh: if (Status==1 || Umur>=17) atau if(Nil1 >=60 || Nil2>=65)

Tabel kebenaran untuk operator OR

No.	Cond1	Cond2	Cond1 OR Cond2
1.	True	True	True
2.	True	False	True
3.	False	True	True
4.	False	False	False

Perhatikan tabel kebenaran No. 1, 2, dan 3 hanya salah satu kondisi bernilai TRUE, maka akan menghasilkan nilai gabungan = TRUE

Contoh tabel kebenaran untuk melihat nilai gabungan dua buah kondisi yang digabung dengan operator AND

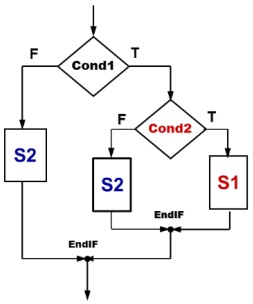
No.	A	B	C	D	Cond1	Cond2	Cond1 || Cond2	Nilai Akhir
1.	5	2	6	4	A>B (True)	C>D (True)	True || True	True
2.	5	2	6	4	A>B (True)	B>D (False)	True || False	True
3.	5	2	6	4	A>C (False)	B>D (False)	False || False	False
4.	5	2	6	4	!(A>B) (False)	!(C>D) (False)	False || False	False

5.5. KONVERSI *MULTI CONDITION* MENJADI NESTED IF

Multi Condition dapat dikonversikan menjadi bentuk Nested If. Contoh-contoh konversi di bawah ini bukan bermaksud menerangkan TRUE atau FALSE suatu kondisi tetapi contoh ini diperlukan jika harus menulis dalam bentuk Nested If.

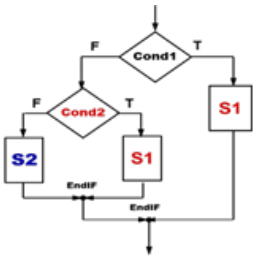
CONTOH 1

Multi Kondisi	Nested If	*Flowchart*

`if (cond1 && cond2)`
`  { S1 }`
`else`
`  { S2 }`
	`if (cond1)`
	`  { if (cond2)`
	`  { S1 }`
	`  else`
	`  { S2 }`
	`  }`
	`else`
	`  { S2 }`
	
	Penggalan perintah nested if di atas, merupakan salah satu cara (bukan satu-satunya)
		

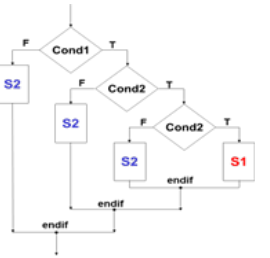
CONTOH 2

Multi Kondisi	Nested If	*Flowchart*

if (cond1 || cond2)
{ S1 }
else
{ S2 }
	if (cond1)
	{ S1 }
	else
	{ if(cond2)
	{ S1 }
	else
	{ S2 }
	}
		

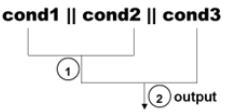
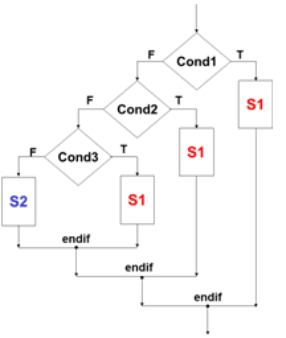
CONTOH 3

Multi Kondisi	Nested If	*Flowchart*

if (cond1 && cond2 && cond3)
{ S1 }
else
{ S2 }
 	
Yang diproses oleh komputer:
 	
 
	if (cond1)
	{ if (cond2)
	{ if(cond3)
	{ S1 }
	else
	{ S2 }
	}
	else
	{ S2 }
	}
	else
	{ S2 }
		

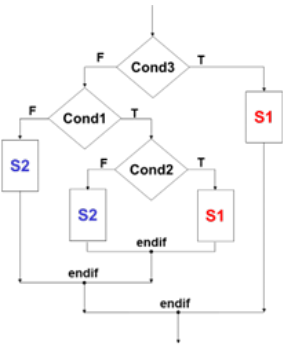
CONTOH 4

Multi Kondisi	Nested If	*Flowchart*

if (cond1 || cond2 || cond3)
{ S1 }
else
{ S2 }
 	
Yang diproses oleh komputer:
 	
 
	if (cond1)
	{ S1 }
	else
	{ if(cond2)
	{ S1 }
	else
	{ S2 }
	}
		

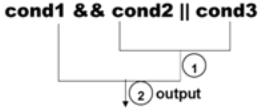
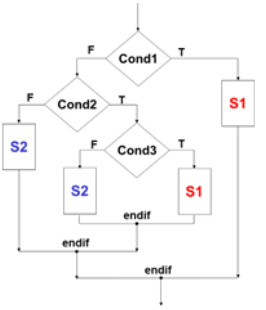
CONTOH 5

Multi Kondisi	Nested If	*Flowchart*

`if (cond1 && cond2 || cond3)`
`{ S1 }`
`else`
`{ S2 }`
 	
Yang diproses oleh komputer:
 	
`cond1 && cond2 || cond3`
 
 	
	`if (cond3)`
	`{ S1 }`
	`else`
	`{ if(cond1)`
	`{ if(cond2)`
	`{ S1 }`
	`else`
	`{ S2 }`
	`}`
	`else`
	`{ S2 }`
	`}`
		

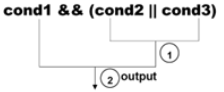
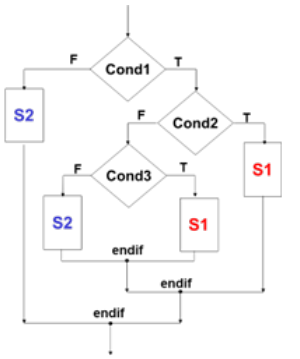
CONTOH 6

Multi Kondisi	Nested If	*Flowchart*

`if (cond1 || cond2 && cond3)`
`{ S1 }`
`else`
`{ S2 }`
 	
Yang diproses oleh komputer:
 	
`cond1 && cond2 || cond3`
 
 	
	`if (cond1)`
	`{ S1 }`
	`else`
	`{ if(cond2)`
	`{ if(cond3)`
	`{ S1 }`
	`else`
	`{ S2 }`
	`}`
	`else`
	`{ S2 }`
	`}`
		

CONTOH 7

Multi Kondisi	Nested If	*Flowchart*

`if (cond1 && (cond2 || cond3))`
`{ S1 }`
`else`
`{ S2 }`
 	
Yang diproses oleh komputer:
 	
 
 	
	`if (cond1)`
	`{ if(cond2)`
	`{ S1 }`
	`else`
	`{ if(cond3)`
	`{ S1 }`
	`else`
	`{ S2 }`
	`}`
	`}`
	`else`
	`{ S2 }`
		

5.6. CONTOH PROGRAM SEDERHANA MENGGUNAKAN NESTED IF DAN *MULTI CONDITION*

Susun Algoritma untuk menginput 3 (tiga) bilangan bulat (integer), di mana ketiga buah bilangan tersebut dianggap bernilai tidak sama, kemudian mencetak salah satu bilangan yang nilainya terbesar.

JAWAB:

CARA – 1

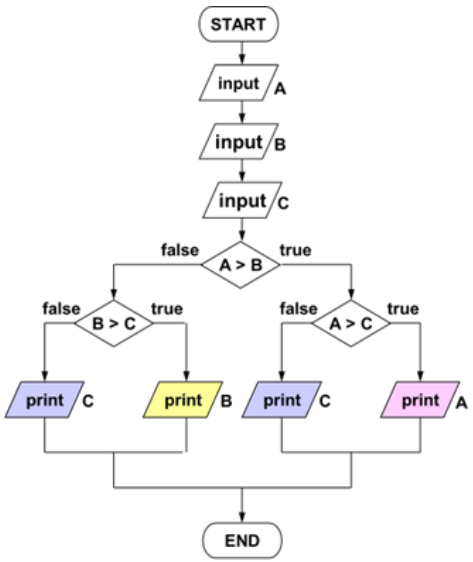
Cara satu dengan menggunakan tiga variabel untuk menyimpan Bilangan 1, Bilangan 2, dan Bilangan 3. Kemudian ketiga bilangan tersebut diperiksa satu persatu sehingga mendapatkan nilai yang terbesar.

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi A, B, C	Proses
2.	Input A, B, C	Input/Outpt

No.	*Pseudocode*	Simbol *Flowchart*

3.	IF (A>B)
	THEN IF(A>C)
	THEN Cetak A
	ELSE Cetak B
	ENDIF
	ELSE
	IF (B>C)
	THEN Cetak B
	ELSE Cetak C
	ENDIF
	ENDIF
		Kondisi, Input/Output

***Flowchart* Cara – 1**



CARA – 2

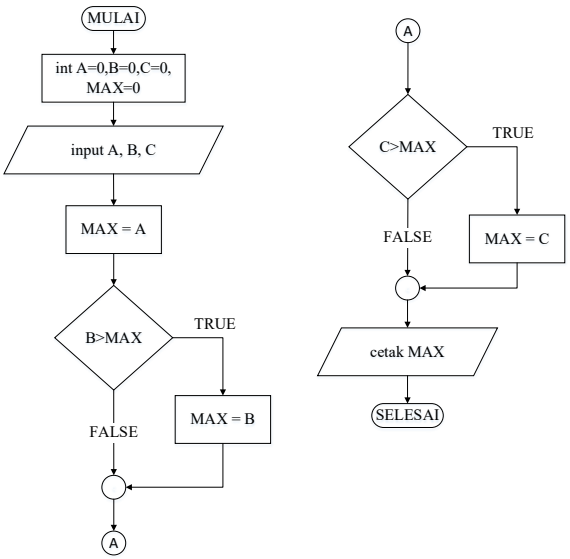
Cara kedua dengan menggunakan empat variabel yaitu untuk menyimpan Bilangan 1, Bilangan 2, Bilangan 3, dan variabel MAX untuk menyimpan nilai maksimal yang di dapat. Bilangan yang diinput akan diperiksa satu-persatu jika bilangan tersebut merupakan bilangan terbesar maka akan disimpan kedalam MAX

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi A, B, C, MAX	Proses
2.	Input A, B, C	Input/Outpt
3.	Isikan MAX dengan Nilai A	Proses

4.	IF (B>MAX)
	THEN MAX = B
	ENDIF
	IF (C>MAX)
	THEN MAX = C
	ENDIF
		Kondisi, Input/Output

5.	Cetak MAX	Input/Output

***Flowchart* Cara – 2**



CARA – 3

Cara ketiga hampir sama dengan Cara – 2 yaitu dengan menggunakan empat variabel yaitu untuk menyimpan Bilangan 1, Bilangan 2, Bilangan 3, dan variabel MAX untuk menyimpan nilai maksimal yang di dapat. Perbedaannya, setiap menginput Bilangan baik bilangan pertama, kedua, atau ketiga langsung dilakukan pengecekan, jika ternyata bilangan yang

diinput merupakan bilangan yang terbesar maka akan disimpan kedalam variabel MAX

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi A, B, C, MAX	Proses
2.	Input A	Input/Output
3.	Isikan MAX dengan Nilai A	Proses
4.	Input B	Input/Output

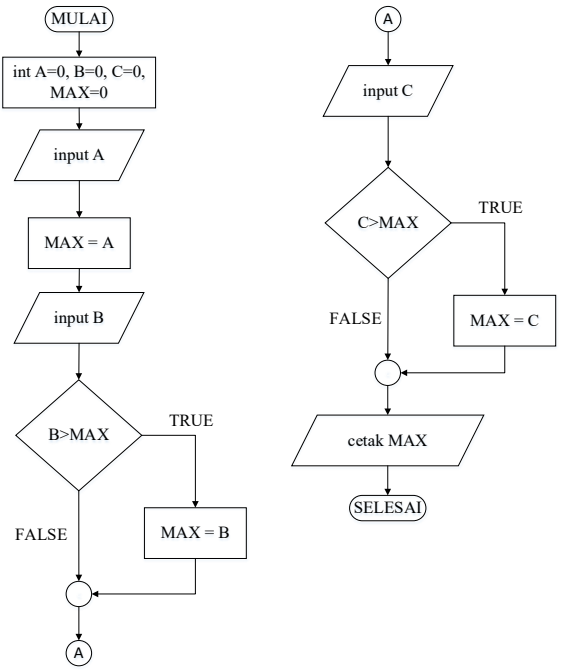
5.	IF (B>MAX)
	THEN MAX = B
	ENDIF
		Kondisi, Input/Output

6.	Input C	Input/Output

7.	IF (C>MAX)
	THEN MAX = C
	ENDIF	

8.	Cetak MAX	Input/Output

Flowchart Cara – 3



CARA – 4

Cara keempat hanya menggunakan dua variabel yaitu satu variabel yaitu A untuk menyimpan tiga bilangan yang diinput (secara bergantian), dan satu variabel lagi yaitu MAX untuk menyimpan nilai maksimal yang didapat. Setiap menginput bilangan akan di periksa yang jika ternyata nilainya lebih besar dari nilai sebelumnya, akan disimpan ke dalam variabel MAX.

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi A, MAX	Proses
2.	Input A	Input/Output
3.	Isikan MAX dengan Nilai A	Proses
4.	Input A	Input/Output

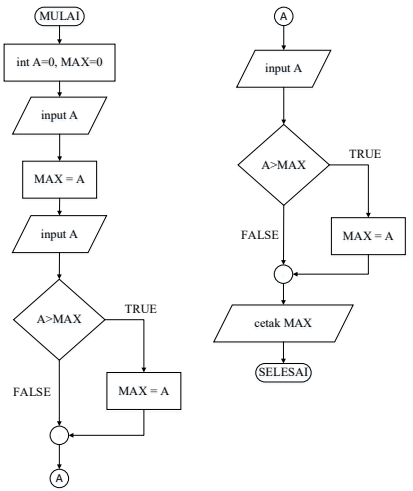
5.	IF (A>MAX)
	THEN MAX = A
	ENDIF
		Kondisi, Input/Output

6.	Input A	Input/Output

7.	IF (A>MAX)
	THEN MAX = A
	ENDIF	

8.	Cetak MAX	Input/Output

***Flowchart* Cara – 4**



Algoritma ini nanti yang akan menjadi dasar algoritma pencarian bilangan terbesar atau terkecil dari sejumlah bilangan yang ada atau bilangan yang diinput

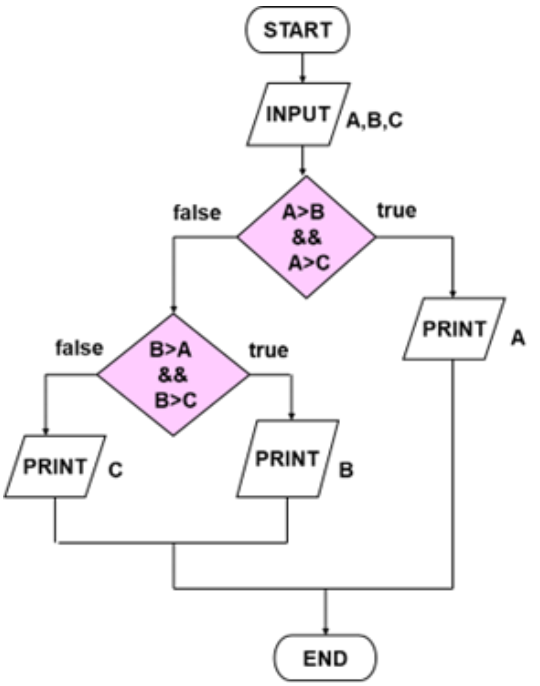
CARA – 5

Cara kelima menggunakan tiga variabel untuk menyimpan masing-masing bilangan, tetapi pemeriksaan kondisinya menggunakan *Multi Condition* dengan NESTED-IF

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi A,B,C	Proses
2.	Input A, B, C	Input/Output

3.	IF (A>B && A>C)
	THEN Cetak A
	ELSE
	IF (B>A && B>C)
	THEN Cetak B
	ELSE Cetak C
	ENDIF
	ENDIF
		Kondisi, Input/Output

***Flowchart* Cara – 5**



CARA – 6

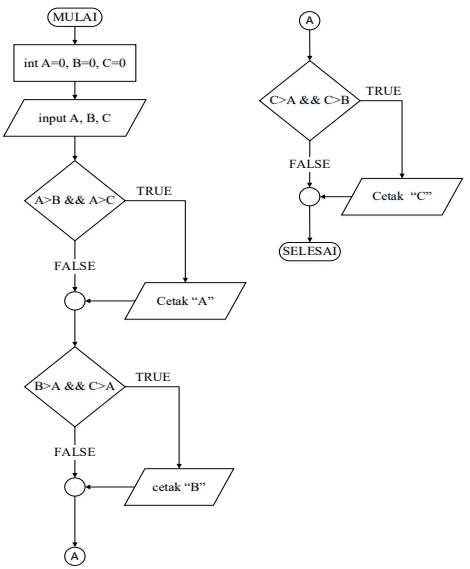
Cara keenam menggunakan 3 Variabel untuk menyimpan masing-masing bilangan, kemudian bilangan tersebut diperiksa dengan menggunakan IF-THEN di mana kondisinya menggunakan *Multi Condition*.

Cara ini paling mudah logikanya tetapi dengan cara seperti ini *computer-time* bisa lebih panjang karena komputer akan melaksanakan semua instruksi if.

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi A,B,C	Proses
2.	Input A, B, C	Input/Output

3.	IF (A>B && A>C)
	THEN Cetak "A"
	ENDIF
	IF (B>A && B>C)
	THEN Cetak "B"
	ENDIF
	IF (C>A && C>B)
	THEN Cetak "C"
	ENDIF
		Kondisi, Input/Output

***Flowchart* Cara – 6**



5.7. SELEKSI MENGGUNAKAN SWITCH-CASE

Switch-Case merupakan jenis seleksi yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif penyelesaian.

Kondisi Switch-Case terdiri dari dua bagian, yaitu Switch di mana terdapat variabel yang akan diperiksa. Serta satu atau lebih perintah Case masing-masing untuk setiap nilai yang ingin diperiksa.

Switch-Case memiliki batasan, yaitu:

1. Data yang diperiksa harus bertipe integer atau karakter
2. *Range* data yang diperiksa bernilai 0 s/d 255

DEFAULT

Blok default merupakan perintah yang akan dijalankan jika semua kondisi di Case tidak ada yang True.

BREAK

Break diberikan pada Blok Case, dengan tujuan untuk keluar dari Case yang bernilai true dan lanjut pada next-instruction sehingga tidak perlu memeriksa pada perintah case selanjutnya.

CONTOH SELEKSI MENGGUNAKAN SWITCH-CASE

Susun Algoritma atau Program untuk menginput sebuah nilai integer (misal N) kemudian cetak peringkat nilai sesuai dengan nilai N sebagai berikut:

Nilai N	Peringkat yang akan dicetak
1	Kurang Sekali
2	Kurang
3	Cukup

Nilai N	Peringkat yang akan dicetak
4	Baik
5	Baik Sekali

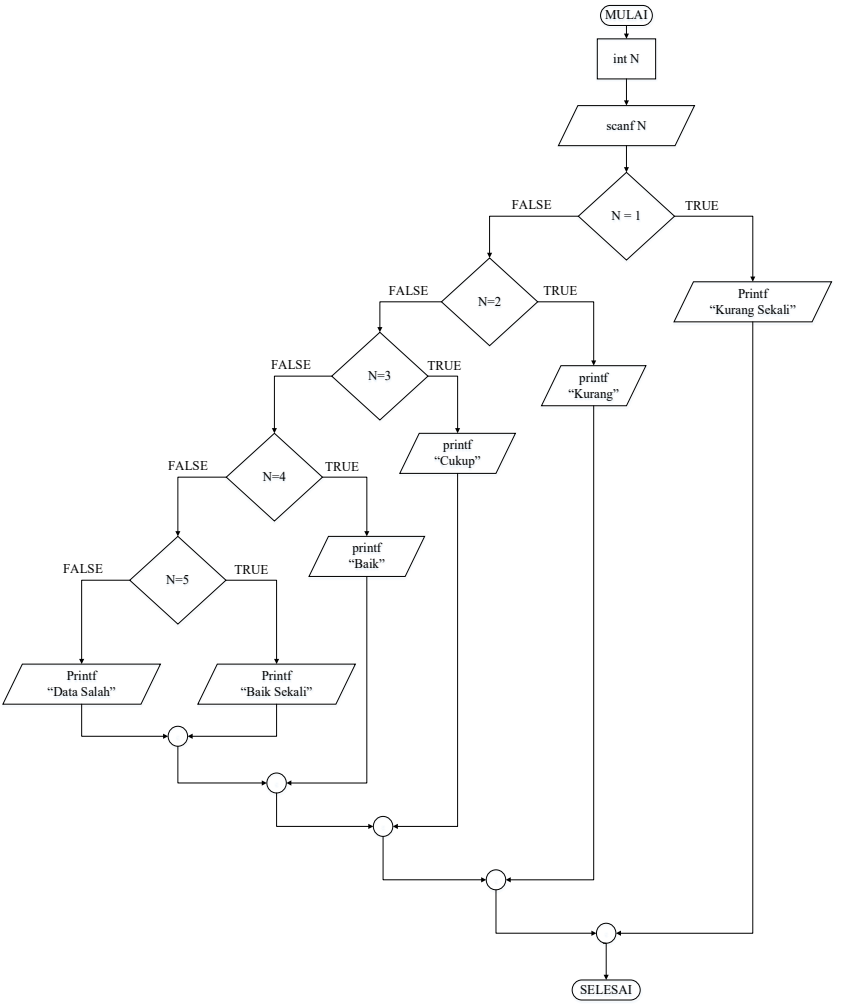
Jawab:

CARA – 1 (MENGGUNAKAN NESTED IF)

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi Nilai N	Proses
2.	Input Nilai N	Input /utput

3.	IF (N=1)
	  THEN Cetak “KURANG SEKALI”
	ELSE IF (N=2)
	  THEN Cetak “Kurang”
	ELSE IF (N=3)
	  THEN Cetak “Cukup”
	ELSE IF (N=4)
	  THEN Cetak “Baik”
	ELSE IF (N=5)
	  THEN Cetak “Baik Sekali”
	ELSE
	  Cetak “Data Salah”
	ENDIF
		Kondisi, Input/Output

Flowchart Cara – 1

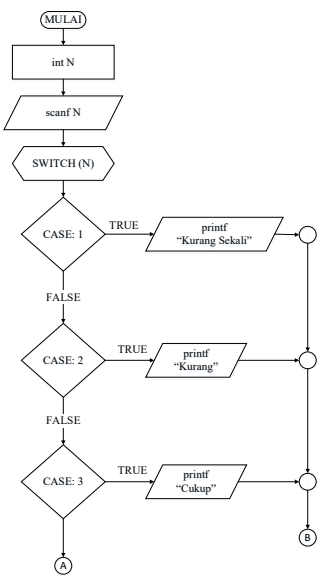


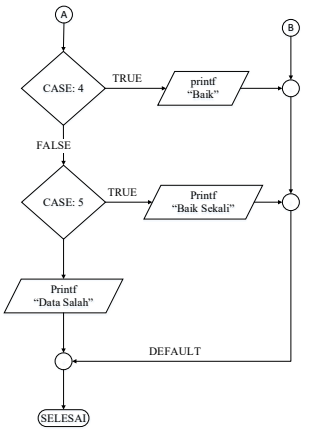
Cara – 2 (Menggunakan Switch-Case)

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi Nilai N	Proses
2.	Input Nilai N	Input/Output

No.	*Pseudocode*	Simbol *Flowchart*
3.	CASE 1: Cetak “Kurang Sekali” break CASE 2: Cetak “Kurang” break CASE 3: Cetak “Cukup” break CASE 4: Cetak “Baik” break CASE 5: Cetak “Baik Sekali” break Default: Cetak “Data Salah”	Kondisi, Input/Output

Flowchart Cara – 2





5.8. SWITCH-CASE BERJENJANG

Switch-Case berjenjang seperti Nested-IF (If bersarang).

CONTOH SWITCH-CASE BERJENJANG

Buatlah sebuah program atau algoritma yang digunakan untuk menginput inisial Pulau beserta dengan nama kota yang berada di pulau tersebut. Dengan data Pulau beserta dengan nama kota sebagai berikut:

Pulau		Kota	
Kode Pulau	**Nama Pulau**	**Kode Kota**	**Nama Kota**
J	Pulau Jawa	1	Jakarta
		2	Surabaya
		3	Bandung
		4	Semarang
		5	Yogyakarta
S	Pulau Sumatera	1	Medan
		2	Palembang
		3	Padang
K	Pulau Kalimantan	1	Banjarmasin
		2	Pontianak

Misalkan:
Bila diinput : J 2 <enter>
Maka tercetak : Pulau Jawa
Surabaya

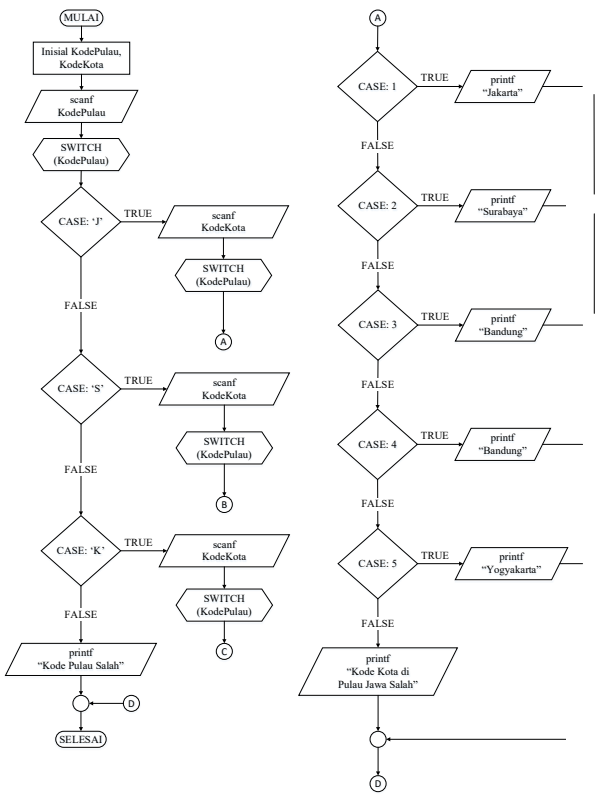
Bilai diinput : K 2 <enter>
Maka tercetak : Pulau Kalimantan
Pontianak

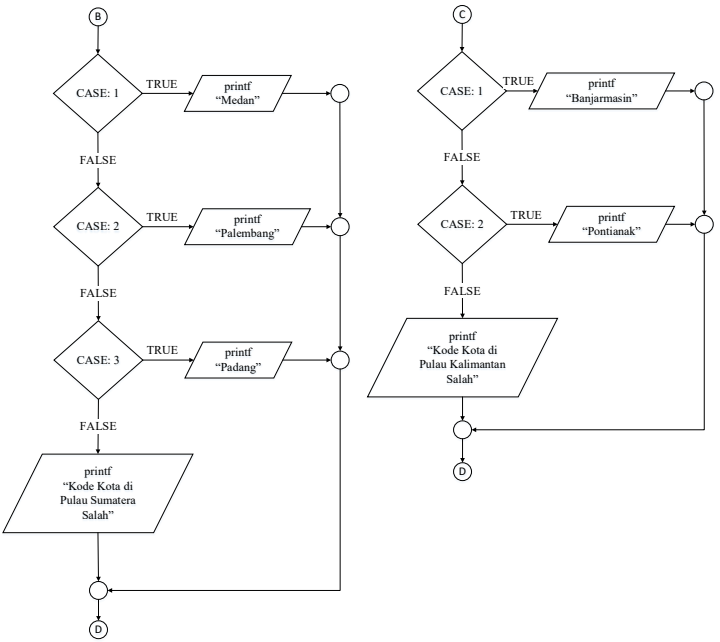
Jawab:

No.	*Pseudocode*	Simbol *Flowchart*
1.	Inisialisasi KodePulau, KodeKota	Proses
2.	Input KodePulau	Input /Output
3	CASE 'J': Cetak "Pulau Jawa" Input KodeKota CASE 1: Cetak "Jakarta" break; CASE 2: Cetak "Surabaya" break; CASE 3: Cetak "Semarang" break; CASE 4: Cetak "Yogyakarta" break; default: Cetak "Kode Kota di Pulau Jawa Salah" CASE 'S': Cetak "Pulau Sumatera" Input Kode Kota Case 1: Cetak "Medan" break; Case 2: Cetak "Palembang" break; Case 3: Cetak "Padang" break; default: Cetak "Kode Kota di Sumatera Salah"	

No.	*Pseudocode*	Simbol *Flowchart*
	Case 'K': Cetak "Pulau Kalimantan" Input Kode Kota Case 1: Cetak "Banjarmasin" break; Case 2: Cetak "Pontianak" break; default: Cetak "Kode Kota di Kalimantan Salah" default: Cetak "Kode Pulau Salah"	Kondisi, Input/Output

Flowchart





KESIMPULAN

Dalam membuat sebuah perintah algoritma dengan menggunakan struktur kontrol percabangan bisa saja kemungkinan digunakan Nested-If (If berjenjang) atau Switch-Case.

Pilihlah struktur yang sesuai dengan ketentuan algoritma benar dan efisien

SOAL LATIHAN

1. Susun algoritma (program) untuk menginput 3 buah bilangan yang masing-masing menyatakan panjang sisi sebuah segitiga. Kemudian periksa ketiga buah garis (sisi) tersebut.

   Bila ketiga buah garis (sisi) tersebut panjangnya sama maka cetak perkataan "SAMA SISI". Bila hanya dua sisi yang sama maka cetak perkataan "SAMA KAKI ". Tapi bila ketiga-tiganya tidak sama maka cetak perkataan "SEMBARANG".

   Tidak boleh menggunakan *logical operator* **AND dan OR.**

2. Susun program untuk menginput tiga buah bilangan yang menyatakan nilai ujian tiga buah mata kuliah.
   a. Cetak perkataan “TIGA” bila ketiga mata kuliah tersebut mendapat nilai lulus.
   b. Cetak perkataan “DUA”, bila hanya dua dari ketiga mata kuliah tersebut yang mendapat nilai lulus.
   c. Cetak perkataan “SATU” bila hanya satu mata kuliah yang mendapat nilai lulus.
   d. Cetak perkataan “NOL” bila ketiga mata kuliah tersebut dinyatakan tidak lulus
   e. Sebuah mata kuliah dinyatakan mendapat nilai lulus bila nilainya lebih besar atau sama dengan 60.

3. Tulis program untuk menentukan lama bekerja seorang pegawai, jika jam masuk dan jam pulang diinput. Catatan: jam berupa angka 1-12, dan seorang pegawai bekerja kurang dari 12 jam.
   Contoh keluaran:

Jam masuk	Jam keluar	Keluaran/tampilan
10	11	Lama bekerja 1 jam
10	2	Lama bekerja 4 jam
10	7	Lama bekerja 9 jam

4. Buatlah program dalam bahasa C untuk menyelesaikan masalah berikut:
   Program akan menerima masukan berupa kode, jenis dan harga, dengan jenis adalah “A”, “B”, dan “C”. Untuk setiap jenis, masing-masing akan diberikan diskon sebesar 10% untuk A, 15% untuk B, dan 20% untuk C. Program akan menghitung berapa harga setelah didiskon.
   Contoh masukan:
   Kode = 10
   Jenis = B
   Harga = 10000

Contoh keluaran:

Jenis barang B mendapat diskon = 15%, Harga setelah didiskon = 8500

5. Buatlah sebuah program untuk menginput data nilai matakuliah berupa NAMA MATAKULIAH, SKS dan NILAI. Selanjutnya tentukan dan tampilkan GRADE matakuliah berdasarkan NILAI yang diinput (gunakan aturan grade baru).

6.

Contoh masukan:

Matakuliah = LogikaMatematika
SKS = 3
Nilai = 84

Contoh keluaran:

Grade = A-

BAB 6

STRUKTUR PERULANGAN

Capaian Pembelajaran	:	Mahasiswa mampu memahami dasar penggunaan struktur perulangan For, While dan Do While.

Subpokok Bahasan	:	6.1. Struktur perulangan For, While dan Do While 6.2. Contoh algoritma untuk menginput 100 buah nilai integer dan mencetak salah satu nilai yang terbesar atau yang terkecil 6.3. Contoh algoritma mencetak deret atau menghitung dan mencetak total suatu deret 6.4. Contoh algoritma menghitung dan mencetak bunga berganda

Daftar Pustaka	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

6.1. STRUKTUR PERULANGAN FOR, WHILE dan Do.. WHile

PENGERTIAN PERULANGAN (*LOOP*)

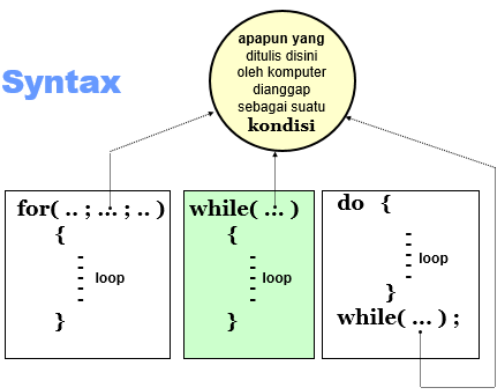
- Struktur perulangan (*loop*) digunakan untuk menyelesaikan persoalan yang melibatkan suatu proses yang dikerjakan beberapa kali sesuai pola tertentu.
- Dengan perulangan (*loop*) memungkinkan pemrograman untuk menjalankan satu atau beberapa perintah yang ada di dalam blok perulangan secara berulang sesuai dengan nilai yang ditentukan atau sampai mencapai sebuah batas tertentu.
- Perulangan adalah sebuah kelompok atau blok instruksi yang dapat dilaksanakan secara berulang-ulang.
- Perulangan adalah proses yang dilaksanakan secara berulang-ulang yang disebut *looping*.

3 MACAM INSTRUKSI PEMBENTUK PERULANGAN (*LOOP*)

Ada 3 macam instruksi pembentuk perulangan (*loop*) yaitu:

1. For
2. While
3. Do.. While

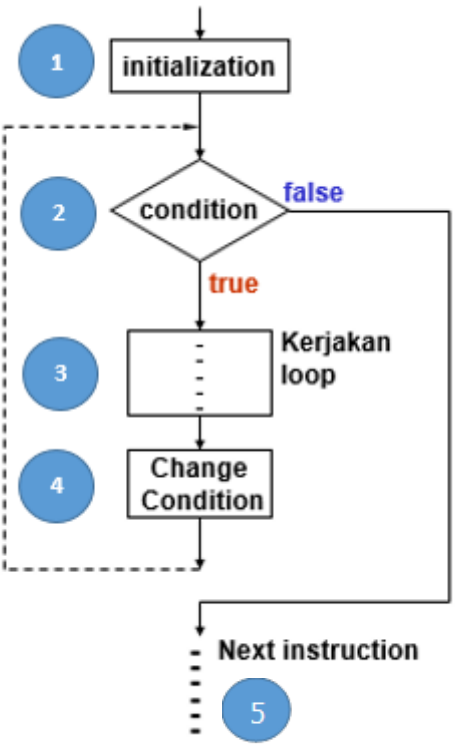
Berikut adalah *syntax* penulisannya:



BENTUK UMUM PERULANGAN FOR

Keterangan	**Init**	=	**Inisialisasi** Instruksi pemberian suatu nilai yang mempengaruhi nilai *condition*. Pada proses yang normal, pemberian nilai awal ini akan menyebabkan *condition* bernilai true. Instruksi ini hanya pernah satu kali dilaksanakan, yaitu hanya pada saat awal.
	Cond	=	***Condition*** Suatu pernyataan yang mengandung nilai BENAR (true) atau SALAH (false).
	Chng of cond	=	***Change of condition*** Suatu instruksi yang dapat mempengaruhi nilai *condition*. Pada proses yang normal, perubahan nilai di sini suatu saat akan membuat nilai *condition* = false.

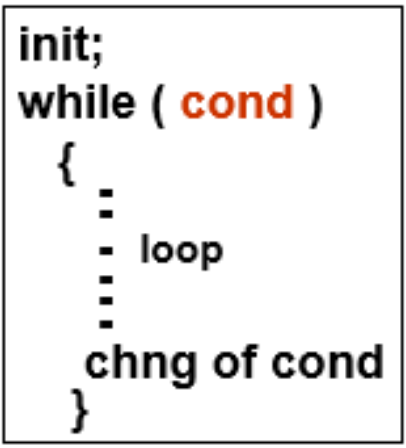
BAGAN DAN CARA KERJA PERULANGAN FOR



Keterangan:

1. Melakukan inisialisasi (*initialization*). Memberi nilai awal kepada sebuah variabel yang ada kaitannya dengan nilai *condition* (kondisi) yang nantinya akan diperiksa.
2. Memeriksa nilai kondisi (*condition*)
   a. Bila nilainya TRUE, maka laksanakan *loop* 3 (mengerjakan instruksi yang ada dalam *loop*)
      Setelah melaksanakan *loop*, lanjutkan ke no. 4, mengubah suatu nilai yang dapat mempengaruhi nilai kondisi.
      Kemudian, kembali ke no. 2, memeriksa kondisi.
   b. Bila kondisi nilainya FALSE, maka *loop* selesai, keluar dari *loop* dan langsung keluar ke no. 5, melaksanakan next instruction 5 (bila ada). Bila tidak ada next instruction maka selesai.

BENTUK UMUM PERULANGAN WHILE



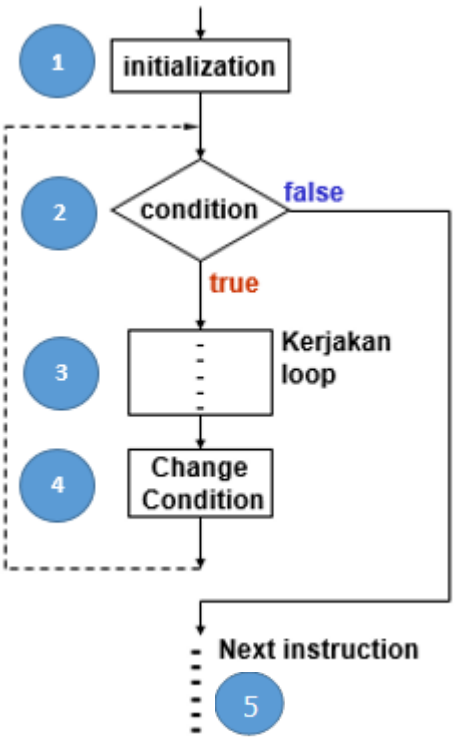
Keterangan

Init = **Inisialisasi**
Instruksi pemberian suatu nilai yang mempengaruhi nilai *condition*. Pada proses yang normal, pemberian nilai awal ini akan menyebabkan *condition* bernilai true. Instruksi ini hanya pernah satu kali dilaksanakan, yaitu hanya pada saat awal.

Cond = **Condition**
Suatu pernyataan yang mengandung nilai BENAR (true) atau SALAH (false).

Chng of cond = **Change of condition**
Suatu instruksi yang dapat mempengaruhi nilai *condition*. Pada proses yang normal, perubahan nilai di sini suatu saat akan membuat nilai *condition* = false.

BAGAN DAN CARA KERJA PERULANGAN WHILE

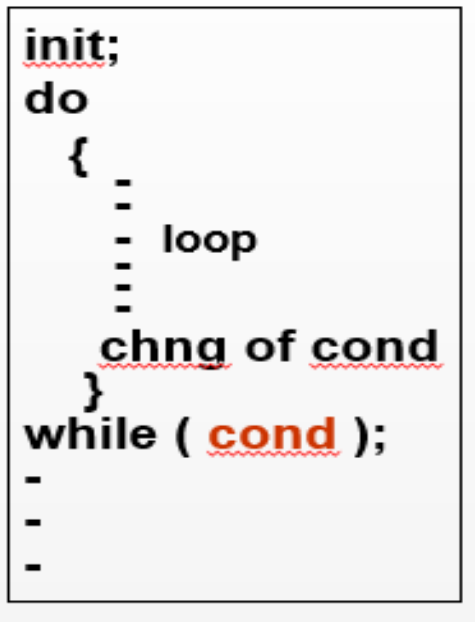


Bagan dan cara kerja perulangan While == For

Keterangan:

1. Melakukan inisialisasi (*initialization*). Memberi nilai awal kepada sebuah variabel yang ada kaitannya dengan nilai *condition* (kondisi) yang nantinya akan diperiksa.
2. Memeriksa nilai kondisi (*condition*)
   a. Bila nilainya TRUE, maka laksanakan *loop* 3 (mengerjakan instruksi yang ada dalam *loop*)
      Setelah melaksanakan *loop*, lanjutkan ke no. 4, mengubah suatu nilai yang dapat mempengaruhi nilai kondisi.
      Kemudian, kembali ke no. 2, memeriksa kondisi.
   b. Bila kondisi nilainya FALSE, maka *loop* selesai, keluar dari *loop* dan langsung keluar ke no. 5, melaksanakan next instruction 5 (bila ada). Bila tidak ada next instruction maka selesai.

BENTUK UMUM PERULANGAN DO.. WHILE



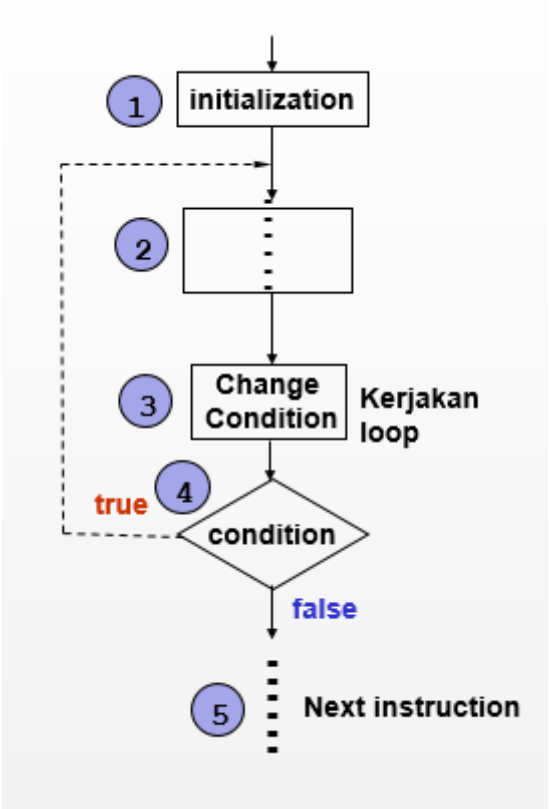
Keterangan

Init = **Inisialisasi**
Instruksi pemberian suatu nilai yang mempengaruhi nilai *condition*. Pada proses yang normal, pemberian nilai awal ini akan menyebabkam *condition* bernilai true. Instruksi ini hanya pernah satu kali dilaksanakan, yaitu hanya pada saat awal.

Cond = **Condition**
Suatu pernyataan yang mengandung nilai BENAR (true) atau SALAH (false).

Chng of cond = **Change of condition**
Suatu instruksi yang dapat mempengaruhi nilai *condition*. Pada proses yang normal, perubahan nilai di sini suatu saat akan membuat nilai *condition* = false.

BAGAN DAN CARA KERJA PERULANGAN DO.. WHILE



Keterangan:

1. Melakukan inisialisasi (*initialization*). Memberi nilai awal kepada sebuah variabel yang ada kaitannya dengan nilai *condition* (kondisi) yang nantinya akan diperiksa.
2. Kerjakan *loop* no. 2, yang dilanjutkan instruksi no. 3 yang dapat mempengaruhi nilai *condition*.
3. Setelah itu periksa *condition* 4 yang ada pada instruksi While.
   a. Bila nilainya TRUE, maka ulangi mengerjakan *loop* no. 2 dan seterusnya.
   b. Bila kondisi nilainya FALSE, maka *loop* selesai, keluar dari *loop* dan langsung mengerjakan next instruction 5.

CONTOH PENGGUNAAN PERULANGAN FOR SEDERHANA

Contoh 1:

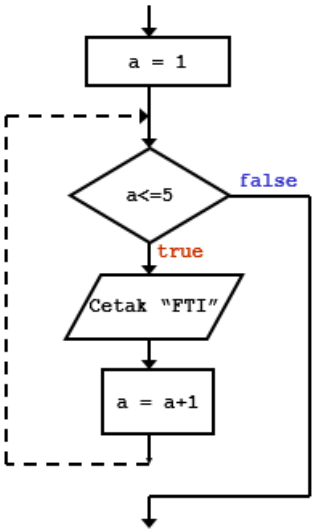
Buatlah algoritma dan *flowchart* untuk mencetak kata "FTI" ke layar sebanyak 5 kali!

Jawab:

Algoritma (For):

Inisialisasi variabel a = 1
Lakukan selama a <=5
    Cetak "FTI"
a=a+1

***Flowchart*:**



Program 6.1 Contoh Program Mencetak Kata dalam Bahasa C

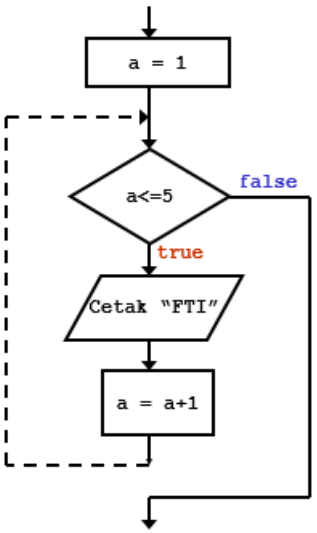
```
#include<stdio.h>
void main()
{ int a;
  for(a=1; a<=5; a=a+1)
   {
    printf("\n FTI");
   }
}
```

Hasil : FTI
FTI
FTI
FTI
FTI

Algoritma (While):

Inisialisasi variabel a = 1
Lakukan selama a <=5
    Cetak “FTI”
a=a+1

***Flowchart*:**



Program 6.2 Contoh Program Mencetak Kata Menggunakan While dalam Bahasa C

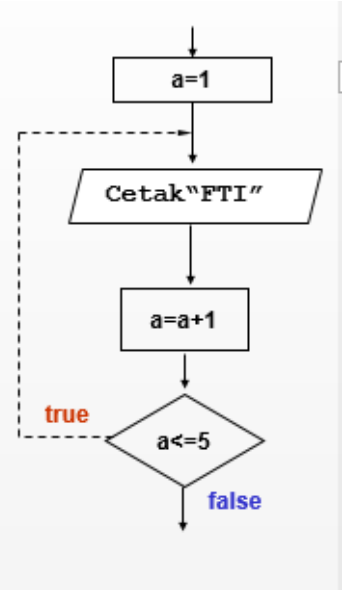
```
#include<stdio.h>
void main()
{ int a;
  a=1;
  while (a<=5) {
    printf(”\n FTI”);
    a=a+1;}
}
```

Hasil:
FTI
FTI
FTI
FTI
FTI

Algoritma (Do.. While):

Inisialisasi variabel a = 1
Cetak “FTI”
a=a+1
Lakukan selama a <=5

***Flowchart*:**



Program 6.3 Contoh Program Mencetak Kata Menggunakan Do.. While dalam Bahasa C

```
#include<stdio.h>
void main()
{ int a;
  a=1;
  do
   {
    printf("\n FTI");
    a=a+1;
   }
  while(a<=5);
}
```

Hasil: FTI
FTI
FTI
FTI
FTI

Contoh 2:

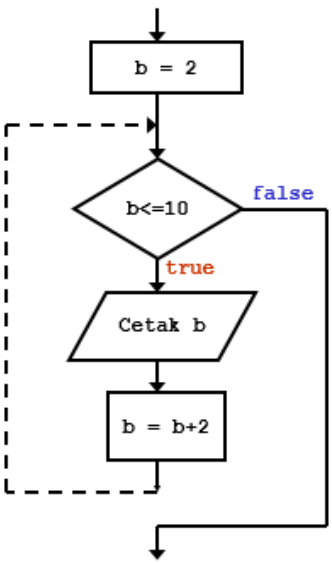
Buatlah algoritma dan *flowchart* untuk mencetak deret bilangan 2 4 6 8 10 ke layar!

Jawab:

Algoritma (For):

Inisialisasi variabel b = 2
Lakukan selama b<=10
    Cetak b
b=b+2

***Flowchart*:**



Program 6.4 Contoh Program Mencetak Deret Menggunakan For dalam Bahasa C

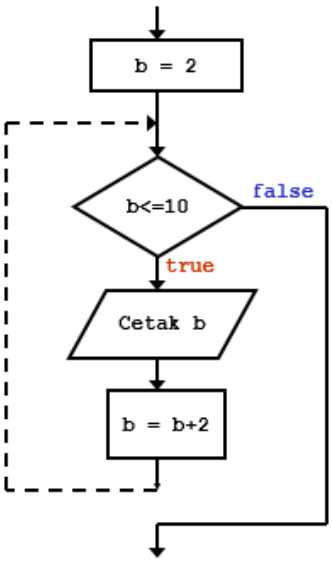
```
#include<stdio.h>
void main()
{ int b;
  for(b=2; b<=10; b=b+2)
   {
    printf("%3i", b);
   }
}
```

Hasil : 2 4 6 8 10

Algoritma (While):

Inisialisasi variabel b = 2
Lakukan selama b<=10
    Cetak b
b=b+2

***Flowchart*:**



Program 6.5 Contoh Program Mencetak Deret Menggunakan While dalam Bahasa C

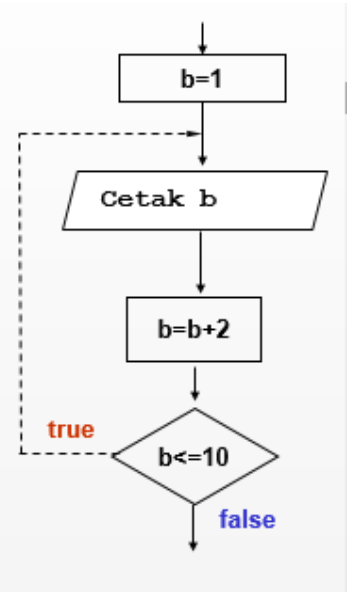
```
#include<stdio.h>
void main()
{ int b;
  b=2;
  while (b<=10) {
    printf ("%i", b);
    b=b+2;}
}
```

Hasil : 2 4 6 8 10

Algoritma (Do... While):

Inisialisasi variabel b = 2
    Cetak b
    b=b+2
Lakukan selama b<=10

***Flowchart*:**



Program 6.6 Contoh Program Mencetak Deret Menggunakan Do.. While dalam Bahasa C

```
#include<stdio.h>
void main()
{ int b;
  b=2;
  do{
    printf ("%i", b);
    b=b+2;}
  while (b<=10);
}
```

Hasil : 2 4 6 8 10

6.2. ALGORITMA UNTUK MENGINPUT SEJUMLAH BUAH NILAI INTEGER KEMUDIAN MENCETAK SALAH SATU NILAI TERBESAR ATAU TERKECIL DARI NILAI YANG DIINPUT

Contoh1:

Buatlah algoritma dan *flowchart* untuk menginputkan 3 buah bilangan bulat dan tampilkan bilangan TERBESAR di antara ketiganya. (dianggap ketiga bilangan nilainya berbeda).

Jawab:

Algoritma: (Menggunakan 1 variabel)

Deklarasi variabel A, MAX
Input nilai A
MAX = A
Input nilai A
Jik A>MAX
    Maka MAX = A
Input nilai A
Jika A>MAX
  Maka MAX = A
Cetak MAX

Berdasarkan algoritma di atas, ada bagian yang sama/diulang:

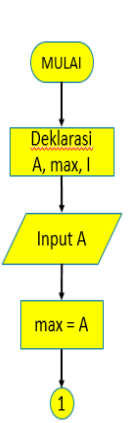
```
Deklarasi variabel A, MAX
Input nilai A
MAX = A
Input nilai A
Jik A>MAX
    Maka MAX = A
Input nilai A
Jika A>MAX
    Maka MAX = A
Cetak MAX
```

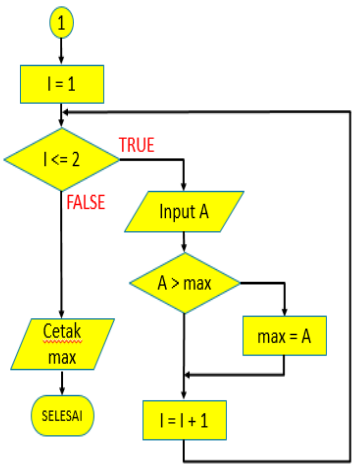
Ada bagian yang sama/ diulang. Bagian ini hanya ditulis sekali saja dengan menggunakan perulangan

Algoritma hasil menggunakan konsep perulangan:

```
Deklarasi variabel A, MAX, i

Input nilai A
MAX = A
i = 1
Lakukan selama 1<=2
    Input nilai A
    Jika A>MAX
        Maka MAX = A
i=i+1
Cetak MAX
```

Flowchart





Program 6.7 Contoh Program Mencetak Bilangan Terbesar Menggunakan For dalam Bahasa C

```
#include <stdio.h>
void main()
{ int A, MAX, i;
  scanf ("%i", &A);
  MAX = A;

  for (i=1; i<=2;i++)
  {  scanf ("%i", &A);
     if (A > MAX)
        { MAX = A;}
  }
  printf ("%i", MAX);
}
```

Program 6.8 Contoh Program Mencetak Bilangan Terbesar Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int A, MAX, i;
  scanf ("%i", &A);
  MAX = A;

  i=1;
  while (i<=2)
  {  scanf ("%i", &A);
     if (A > MAX)
         { MAX = A;}
     i++;
  }
  printf ("%i", MAX);
}
```

Contoh 2:

Buatlah algoritma dan *flowchart* untuk menginputkan 100 buah bilangan bulat dan tampilkan bilangan TERKECIL di antara 100 buah nilai yang diinput tersebut.

Jawab:

Algoritma:

Deklarasi variabel A, MIN, i

Input nilai A

MIN = A

i = 1

Lakukan selama i <=99

Input nilai A

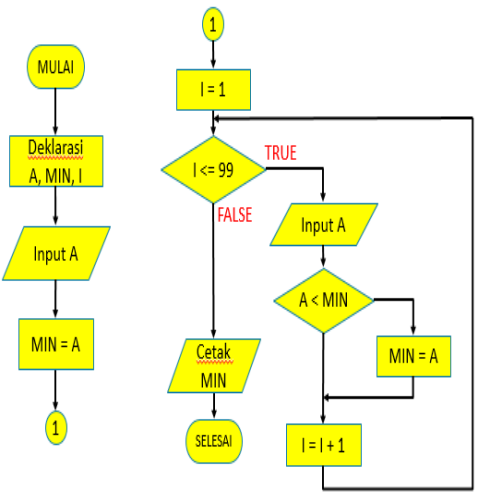
Jik A<MIN

    Maka MIN = A

I++

Cetak MIN

***Flowchart*:**



Program 6.9 Contoh Program Mencetak Bilangan Terkecil Menggunakan For dalam Bahasa C

```
#include <stdio.h>
void main()
{ int A, MIN, i
  scanf ("%i", &A)
  MIN = A

  for (i=1;i < = 99; i++)
  {scanf ("%i", &A)
   if A < MIN
      MIN = A
  }
  printf ("%i", MIN);
}
```

Program 6.10 Contoh Program Mencetak Bilangan Terkecil Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int A, MAX, i;
  scanf ("%i", &A);
  MAX = A;

  i=1;
  while (i<=100)
  {  scanf ("%i", &A);
     if (A > MAX)
         { MAX = A;}
     i++;
  }
  printf ("%i", MAX);
}
```

6.3. ALGORITMA UNTUK MENCETAK DERET ATAU MENGHITUNG DAN MENCETAK TOTAL SUATU DERET

Contoh 1:

Buatlah algoritma dan *flowchart* untuk mencetak deret bilangan berikut ini:

1 3 5 7 9 1 1 13 15 17 19

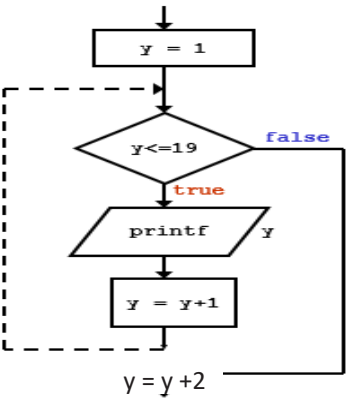
Jawab:

Cara 1:

Algoritma:

Deklarasi variabel y
y = 1
Lakukan selama y <=19
    Cetak y
y = y +2

***Flowchart*:**

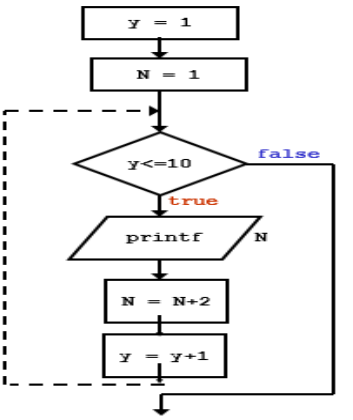


Cara 2:

Algoritma:

Deklarasi variabel y, N
N = 1
y = 1
Lakukan selama y <=10
    Cetak N
N = N +2
y = y +1

***lowchart*:**



Program 6.11 Contoh Program Menghitung dan Mencetak Deret Bilangan Ganjil Menggunakan For dalam Bahasa C

```
#include <stdio.h>
void main()
{ int y, N;

  N=1;
  for (y=1; y<=10; y++)
  {  printf ("%3i", N;
     N=N+2;
}
```

Program 6.12 Contoh Program Menghitung dan Mencetak Deret Bilangan Ganjil Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int y, N;

  N=1;
  y=1;
  while(y<=10)
  {  printf ("%3i", N);
     N=N+2;
     y=y+1;
  }
```

Contoh 2:

Buatlah algoritma dan *flowchart* untuk menghitung dan mencetak total 10 suku pertama deret berikut ini:

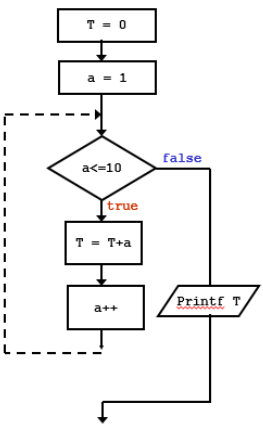
1 2 3 4 5 6 7 8 9 10

Jawab:

Algoritma:

Deklarasi variabel a, T
T = 0
a = 1
Lakukan selama a <=10
    T = T + a
a++
Cetak T

***Flowchart*:**



Program 6.13 Contoh Program Menghitung dan Mencetak Total Deret Menggunakan For dalam Bahasa C

```
#include <stdio.h>
void main()
{ int A, T;
  T = 0;
  for (A = 1; A<=10; A++)
  {  T= T + A; }
  printf ("%i", T);
}
```

Program 6.14 Contoh Program Menghitung dan Mencetak Total Deret Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int a, T;
  T = 0;
  a = 1;
  while (a<=10)
  {  T= T + a;
     a++;
  }
  printf ("%i", T);
}
```

6.4. ALGORITMA UNTUK MENGHITUNG DAN MENCETAK BUNGA BERGANDA

Seseorang menyimpan uang Rp1.000.000 di bank dengan bunga berbunga 2% per bulan. Jadi setelah satu bulan uangnya menjadi Rp1.020.000. Satu bulan berikutnya uang Rp1.020.000 ini mendapat bunga lagi 2%, yaitu Rp20.400 sehingga setelah 2 bulan uangnya menjadi Rp1.020.000 + Rp20.400 = Rp1.040.400. Demikian seterusnya (bunga bulan ini ditambahkan ke saldo uangnya dan mendapatkan bunga lagi pada bulan berikutnya). Susun algoritma dan *flowchart* untuk menghitung dan mencetak jumlah uangnya setelah 10 bulan.

Ilustrasi:

Bulan ke-	Jumlah uang		
	Pada awal bulan ke-I	Bunga 2%	Pada akhir bulan ke-I
I	U	B=0.02*U	U=U+B
1	1.000.000	20.000	1.020.000
2	1.020.000	20.400	1.040.400
3	1.040.400	20.808	1.061.208
4	1.061.208	xxxxxx	xxxxxxxxx
--	----------	------	----------
--	----------	------	----------
--	----------	------	----------
10	xxxxxxxxx	xxxxxx	xxxxxxxxx

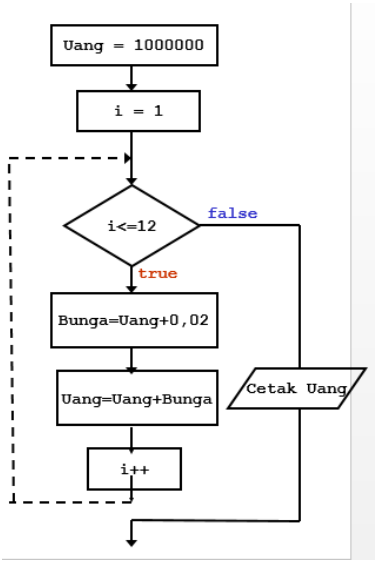
ini yang dicetak

Jawab:

Algoritma:

Deklarasi variabel Uang,
Bunga, i
Uang = 1000000
i = 1
Lakukan selama 1 < =12
    Bunga = Uang * 0,02
    Uang = Uang +
Bunga
i++
Cetak Uang

***Flowchart*:**



Program 6.15 Contoh Program Menghitung dan Mencetak Bunga Berganda Menggunakan For dalam Bahasa C

```
#include <stdio.h>
void main()
{float U, B;
 int I;
 U = 1000000.0;
 for(I=1; I<=10; I++)
   {
     B = U * 0.02;
     U = U + B;
   }
 printf(" %f ", U );
}
```

f

Program 6.16 Contoh Program Menghitung dan Mencetak Bunga Berganda Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{float U, B;
 int I;
 U = 1000000.0;
 I=1;
 while (I<=10)
   {
     B = U * 0.02;
     U = U + B;
     I++;
   }
 printf(" %f ", U );
}
```

KESIMPULAN

Struktur perulangan (*loop*) digunakan untuk menyelesaikan persoalan yang melibatkan suatu proses yang dikerjakan beberapa kali sesuai pola tertentu. Bentuk struktur perulangan ada 3 yaitu:

1. For
2. While
3. Do... While

Struktur perulangan For, While dan Do.. While memiliki bentuk penulisan pada program yang berbeda satu dengan yang lainnya. Namun untuk perulangan For dan While memiliki bentuk penulisan algoritma serta gambar *flowchart* yang sama. Sedangkan untuk perulangan Do.. While memiliki bentuk penulisan algoritma serta gambar *flowchart* yang berbeda.

SOAL LATIHAN

1. Susun algoritma untuk mencetak deret berikut ini:

   a. 1 2 4 8 16 32 64 128 256 512

   b. 5 8 11 14 17 20 23 26 29 32

   c. 100 95 90 85 80 75 70 65 60 55

2. Susun algoritma untuk menginput 50 buah bilangan yang merupakan nilai ujian mahasiswa serta mencetak nilai tertinggi yang dicapai mahasiswa tersebut

3. Seseorang menyimpan uang Rp1.000.000,- di bank dengan bunga 2% per bulan. Susun algoritma dan *flowchart* untuk menghitung dan mencetak pada bulan ke berapa uangnya mencapai atau sedikit melebihi Rp1.500.000,-.

4. Apa yang tercetak jika program berikut dijalankan:

   a

```
#include<stdio.h>
void main()
{ int I, N;
  N = 45;
  for( I=1; I <= 5; I++)
    { printf("\n%i",N);
      N = N + 5;
     }
}
```

   b.

```
int X, N;
  N = 35; X = 0
  while (N<=100)
    { N=N+X;
      printf("\n%i",N);
      X=X+5;
     }
```

BAB 7

STRUKTUR PERULANGAN BERTINGKAT (*NESTED LOOP*)

Capaian Pembelajaran	:	Mahasiswa mampu memahami dasar penggunaan struktur perulangan bertingkat serta penggunaan break dan continue.

Subpokok Bahasan	:	7.1. Penggunaan break dan continue 7.2. Struktur perulangan bertingkat (nested loop) 7.3. Contoh penggunaan nested loop

Capaian Pembelajaran	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

STRUKTUR PERULANGAN BERTINGKAT (*NESTED LOOP*)

7.1. PENGGUNAAN BREAK DAN CONTINUE PADA PERULANGAN

PENGGUNAAN BREAK UNTUK MEMAKSA KELUAR DARI *LOOP*

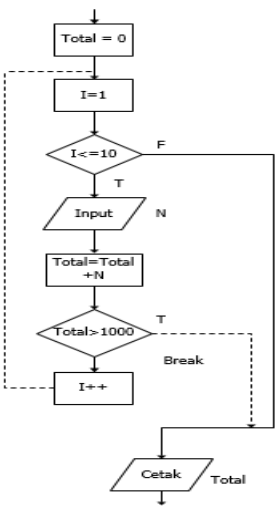
Perintah **break** jika digunakan di dalam perulangan berfungsi untuk 'menghentikan paksa' proses perulangan yang berlangsung. **Break** biasanya digunakan setelah **kondisi IF**, untuk menyelesaikan 'kapan' perulangan harus diberhentikan.

Contoh:

Dalam lembar dokumen ada 10 buah bilangan integer. Susun algoritma untuk menginput bilangan-bilangan tersebut dan menghitung total nilai bilangan-bilangan yang diinput. Bila totalnya sudah >1000, maka berhenti menginput dan mencetak totalnya serta proses selesai walaupun belum semua bilangan diinput. Bila totalnya belum > 1000 maka input dan hitung terus total terus-menerus sampai 10 buah bilangan diinput dan dihitung totalnya, kemudian cetak total dan proses selesai.

Jawab:

Algoritma:	***Flowchart***

Deklarasi variabel I, N, Total
Total = 0
I = 1
Lakukan selama I<=10
  Input nilai N
  Total = Total +N
  Jika Total > 1000
  Maka Berhenti (break)
I=I+1
Cetak Total
	

Program 7.1 Contoh Program Penggunaan Break Menggunakan For dalam Bahasa C

```
#include <stdio.h>
void main()
{ int I, N, Total;
  Total = 0;

  for(I=1;I<=10;I++) //berhenti input bila sudah 10x input
  {  scanf ("%i", &N);
     Total= Total + N;
     If (Total > 1000) break;//berhenti bila Total>1000
  }

  printf("%i", Total)
}
```

Program 7.2 Contoh Program Penggunaan Break Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int I, N, Total;
  Total = 0;

  I=1;
  while (I<=10) //berhenti input bila sudah 10x input
  {  scanf ("%i", &N);
     Total= Total + N;
     If (Total > 1000) break;//berhenti bila Total>1000
     I++;
  }

  printf("%i", Total);
}
```

Program 7.3 Contoh Program Penggunaan Break Menggunakan Do.. While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int I, N, Total;
  Total = 0;

  I=1;
  do
  {  scanf ("%i", &N);
     Total= Total + N;
     If (Total > 1000) break;//berhenti bila Total>1000
     I++;
  }
  while (I<=10); //berhenti input bila sudah 10x input
  printf("%i", Total;
}
```

Sebenarnya untuk menyelesaikan persoalan di atas, selain menggunakan break bisa memasukkan 2 syarat di atas kedalam kondisi.
2 syarat tersebut yaitu:

1. Data yang diinput paling banyak 10 buah
2. Berhenti menginput bila Total > 1000

Sehingga bila menggunakan While, logikanya dapat dibuat sebagai berikut:

Selama data yang diinput belum 10 buah dan

Totalnya belum >1000 (Total<=1000) maka lanjutkan input

Contoh Program 7.4 Penggunaan kondisi:

```
#include <stdio.h>
void main()
{ int I, N, Total;
  Total = 0; I=1;
  while (I<=10 && Total <=1000)
  {  scanf ("%i", &N);
     Total= Total + N;
     I++;
  }
  printf("%i", Total;
}
```

PENGGUNAAN CONTINUE UNTUK MEMAKSA MELANJUTKAN *LOOP*

Perintah **continue** digunakan untuk menghentikan perulangan yang saat ini terjadi (1 iterasi saja), dan kemudian melanjutkan perulangan iterasi berikutnya.
Perintah **continue** digunakan setelah **kondisi IF** yang digunakan untuk menyeleksi 'kapan' perulangan harus di-*skip* atau dilewati.

Contoh:
Dalam lembar dokumen ada banyak sekali bilangan integer yang merupakan nilai ujian mahasiswa. Susun algoritma untuk mengambil 5 nilai pertama yang nilainya >=60 dan mencetak nilai rata-rata. Dipastikan yang mendapat nilai >=60 lebih dari 5 orang.

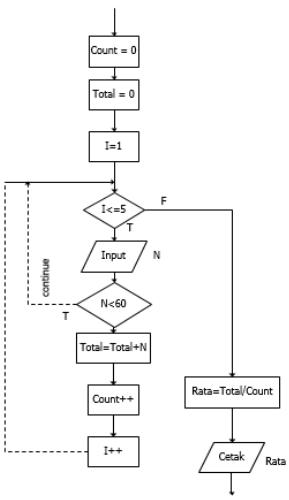
Ilustrasi:
Diinput satu persatu dari awal, tapi yang diambil hanya yang nilainya >=60. Bila nilai yang diinput <60, maka lanjutkan menginput nilai berikutnya. Bila telah dapat 5 orang yang nilainya >=60, maka proses input selesai, kemudian cetak nilai rata-rata.

Jawab:

Algoritma:

```
Deklarasi variabel I, N, Total,
Rata, Count
Count = 0
Total = 0
I = 1
Lakukan selama I<=5
  Input nilai N
  Jika N < 60 maka lanjutkan
(continue)
  Total=Total+N
  Count=Count+1
  I++
Rata=Total/Count
Cetak Rata
```

Flowchart



Program 7.5 Contoh Program Penggunaan Continue Menggunakan While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int I, N, Total, Count;
  Total = 0;
  Count = 0;
  I=1;
  while (I<=5)
  {  scanf ("%i", &N);
     if (N<60) continue;
     Total= Total + N;
     Count++;
     I++;
  }
  Rata= Total/Count;
  printf("%i", Rata);
}
```

Keterangan:

Misal data yang ada dalam dokumen sebagai berikut:

75, 50, 60, 55, 80, 70, 45, 52, 65, 70, 86, 50, ... dst.

Akan diinput: 75, 50, 60, 55, 80, 70, 45, 52, 65

Yang ditambahkan ke Total hanya 5 buah bilangan pertama yang >=60 yaitu: 75, 60, 80, 70, 65

Sehingga Total = 350

Dan Nilai rata-rata yang tercetak adalah 350/5 = 70

7.2. STRUKTUR PERULANGAN BERTINGKAT (*NESTED LOOP*)

PENGERTIAN *NESTED LOOP*

Nested loop atau perulangan bertingkat adalah struktur perulangan yang berada di dalam struktur perulangan lainnya, biasanya melibatkan 2 kali perulangan.

BENTUK UMUM PERULANGAN BERTINGKAT (*NESTED LOOP*)

Program A

```
#include<stdio.h>
void main()
{  int J;
   for ( J=1; J<=3; J++ )
     {
       -
       -
       -
       -
       }
}
```

Loop

Loop ini dikerjakan 3x

Program B

```
for ( I=1; I<=5; I++)
 {
 }
```

Loop ini dikerjakan 5x

Dalam program A ada *loop* yang dikerjakan sebanyak 3x, dan di penggalan program B ada *loop* yang dikerjakan sebanyak 5x.

Bila penggalan program B dimasukkan kedalam program A menggantikan kotak 1 maka berbentuk **suatu *Loop* di dalam *Loop*** yang biasa disebut ***Nested Loop* atau *Loop* Bertingkat**.

Hasil Penggabungan:

```
#include<stdio.h>
void main()
{ int I, J;
  for ( J=1; J<=3; J++)
   {
        for ( I=1; I<=5; I++)
         {
         }
   }
}
```

Outer Loop (Loop Bagian Luar)

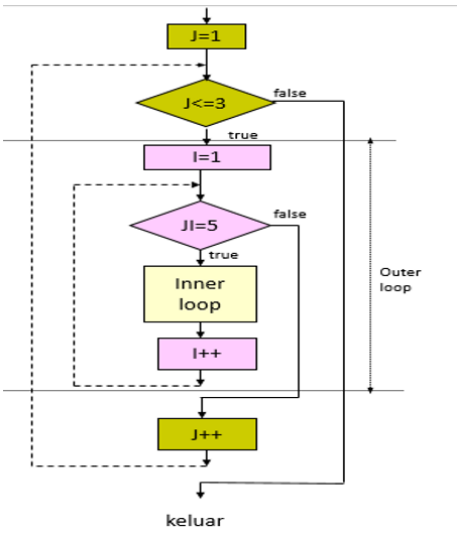
Inner Loop (Loop Bagian Dalam)

Instruksi yang ada di dalam *loop* ini akan dilaksanakan sebanyak 3x5 kali = 15 kali

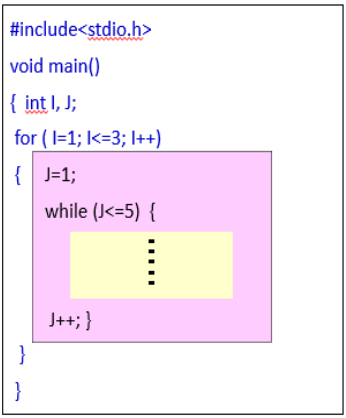
Untuk *loop* yang dibentuk dengan While, maka *nested loop*-nya sebagai berikut:

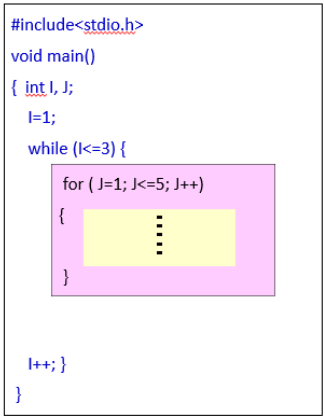
```
#include<stdio.h>
void main()
{  int I, J;
   I=1;
   while (I<=3) {
        J=1;
        while (J<=5)  {
                 ⋮
         J++; }

   I++; }
}
```

***Flowchart* (bentuk For maupun While sama):**



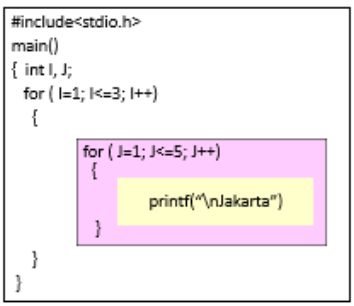
Nested Loop dapat merupakan gabungan antara for() dan while() seperti berikut ini:





7.3. CONTOH PENGGUNAAN *NESTED LOOP*

Contoh 1: (*Looping* 3 x 5)



Perhatikan tabel berikut untuk Contoh 1:

I	J	Cetak
1	1	**Jakarta**
	2	**Jakarta**
	3	**Jakarta**
	4	**Jakarta**
	5	**Jakarta**
2	1	**Jakarta**
	2	**Jakarta**
	3	**Jakarta**
	4	**Jakarta**
	5	**Jakarta**
3	1	**Jakarta**
	2	**Jakarta**
	3	**Jakarta**
	4	**Jakarta**
	5	**Jakarta**

Hasil:

Jakarta
Jakarta
-
-
Jakarta

} Mencetak Jakarta sebanyak 15 kali

Contoh 2: (*Looping* 3 x 5)

```
#include<stdio.h>
main()
{ int I, J;
  for ( I=1; I<=3; I++)
    {
          for (J=1; J<=5; J++)
          {


    }
}
```

Perhatikan tabel berikut untuk Contoh 2:

I	J	Cetak
1	1	**1**
	2	**2**
	3	**3**
	4	**4**
	5	**5**
2	1	**1**
	2	**2**
	3	**3**
	4	**4**
	5	**5**
3	1	**1**
	2	**2**
	3	**3**
	4	**4**
	5	**5**

Hasil:

1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

Contoh 3: (*Loop* 3 x 5)

```
#include<stdio.h>
main()
{ int I, J;
  for ( I=1; I<=3; I++)
    {
       for ( J=1; J<=5; J++)
        {
            printf("%i, J")
        }

    printf ("\n");
    }
}
```

Perhatikan tabel berikut untuk Contoh 3:

I	J	Cetak
1	1	**1**
	2	**2**
	3	**3**
	4	**4**
	5	**5**
2	1	**1**
	2	**2**
	3	**3**
	4	**4**
	5	**5**
3	1	**1**
	2	**2**
	3	**3**
	4	**4**
	5	**5**

Hasil:

1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

Setiap 5x cetak nilai J, turun 1 baris

Contoh 4: (*Loop* 3 x 5)

```
#include<stdio.h>
main()
{ int I, J;
  for ( I=1; I<=3; I++)
    {
       for (J=1; J<=5; J++)
       {

    printf ("\n");
    }
}
```

Perhatikan tabel berikut untuk Contoh 4:

I	J	Cetak
1	1	**1**
	2	**1**
	3	**1**
	4	**1**
	5	**1**
2	1	**2**
	2	**2**
	3	**2**
	4	**2**
	5	**2**
3	1	**3**
	2	**3**
	3	**3**
	4	**3**
	5	**3**

Hasil:

1 1 1 1 1
2 2 2 2 2
3 3 3 3 3

Contoh 5: (*Loop* 5 x 3)

```
#include<stdio.h>
main()
{ int I, J;
  for ( I=1; I<=5; I++)
   {
      for (J=1; J<=3; J++)
      {
        printf ("%3i", J);
        }
   printf ("\n");
   }
}
```

Hasil:

1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

Perhatikan tabel berikut untuk Contoh 5:

I	J	Cetak
1	1 2 3	**1** **2** **3**
2	1 2 3	**1** **2** **3**
3	1 2 3	**1** **2** **3**
4	1 2 3	**1** **2** **3**
5	1 2 3	**1** **2** **3**

Contoh 6:

```
#include<stdio.h>
main()
{ int I, J, N;
  N = 1;
  for ( I=1; I<=3; I++)
   {
      for ( J=1; J<=5; J++)
      {
          printf("%3i, N");
          N=N+1;
      }
   printf ("\n");
   }
}
```

Hasil:

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15

Perhatikan tabel berikut untuk Contoh 6:

I	J	N
1	1 2 3 4 5	**1** **2** **3** **4** **5**
2	1 2 3 4 5	**6** **7** **8** **9** **10**
3	1 2 3 4 5	11 12 13 14 15

Contoh 7:

```
#include<stdio.h>
main()
{ int I, J, N;
  N = 1;
  for ( I=1; I<=5; I++)
   {
      for ( J=1; J<=3; J++)
       {
          printf("%3i, N");
            N=N+1;
       }
   printf ("\n");
    }
}
```

Hasil:

1 2 3
4 5 6
7 8 9
10 11 12
13 14 15

Perhatikan tabel berikut untuk Contoh 7:

I	J	N
1	1 2 3	**1** **2** **3**
2	1 2 3	**4** **5** **6**
3	1 2 3	**7** **8** **9**
4	1 2 3	**10** **11** **12**
5	1 2 2	**13** **14** **15**

Contoh 8:

```
#include<stdio.h>
main()
{ int I, J;
  for ( I=1; I<=5; I++)
   {
      for ( J=I; J<=3; J++)
       {
          printf("%3i, J");
       }
   printf ("\n");
    }
}
```

Hasil:

1 2 3 4 5
2 3 4 5
3 4 5
4 5
5

Perhatikan tabel berikut untuk Contoh 8:

I	J
1	**1** **2** **3** **4** **5**
2	**2** **3** **4** **5**
3	**3** **4** **5**
4	**4** **5**
5	**5**

Contoh 9:

```
#include<stdio.h>
main()
{ int I, J, N=1;
  for ( I=1; I<=5; I++)
   {
      for ( J=I; J<=I; J++)
       {
          printf("%3i, N");
              N++;
       }
    printf ("\n");
    }
}
```

Perhatikan tabel berikut untuk Contoh 9:

I	J	N
1	1	**1**
2 2	1 2	**2** **3**
3 3 3	1 2 3	**4** **5** **6**
4 4 4 4	1 2 3 4	**7** **8** **9** **10**
5 5 5 5 5	1 2 3 4 5	**11** **12** **13** **14** **15**

Hasil:

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15

Contoh 10:

```
#include<stdio.h>
main()
{ int I, J; char C='A';
  for ( I=1; I<=5; I++)
   {
      for ( J=I; J<=3; J++)
       {
          printf("%3c", C);
              C++;
       }
    printf ("\n");
    }
}
```

Perhatikan tabel berikut untuk Contoh 10:

I	J	Char C
1	1 2 3	**A** **B** **C**
2	1 2 3	**D** **E** **F**
3	1 2 3	**G** **H** **I**
4	1 2 3	**J** **K** **L**
5	1 2 3	**M** **N** **O**

Hasil:

A B C
D E F
G H I
J K L
M N O

Contoh 11:
Buatlah algoritma dan *flowchart* untuk mencetak deret bilangan berikut ke layar!

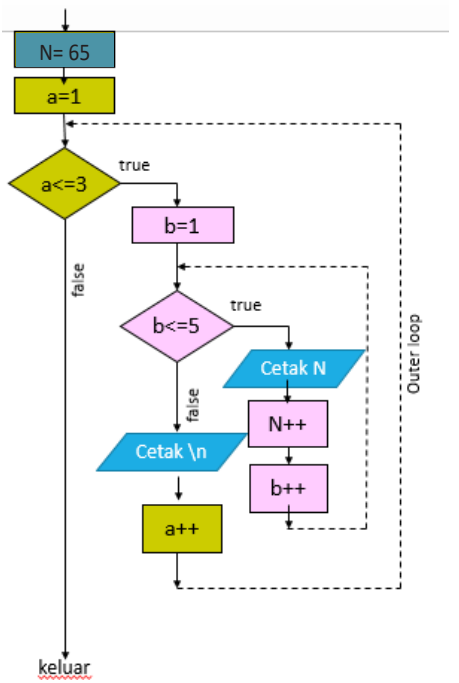
A B C D E
F G H I J
K L M N O

Jawab:

Algoritma:

Deklarasi variabel a,b,N
N=65
a = 1
Lakukan selama a<=3
    b = 1
    Lakukan selama b<=5
        Cetak N
        N=N+1
    b=b+1
Cetak untuk pindah baris
a=a+1

Flowchart



Program 7.6 Contoh Program Mencetak Deret Menggunakan Nested For dalam Bahasa C

```
#include <stdio.h>
void main()
{ int a,b,char N;
  N = 65;
  for (a=1; a<=3; a++)
  { for (b=1; b<=5; b++)
    { printf ("%3c", N);
      N++;
    }
  printf("\n");
  }
}
```

Program 7.7 Contoh Program Mencetak Deret Menggunakan Nested While dalam Bahasa C

```
#include <stdio.h>
void main()
{ int a,b,char N;
  N = 65;
  a=1;
  while (a<=3)
  { b=1;
    while (b<=5)
    { printf ("%3c", N);
      N++;
      b++; }
   printf("\n");
   a++;
  }
}
```

KESIMPULAN

Struktur perulangan bertingkat atau *nested loop* adalah perulangan di dalam perulangan. Umumnya, perulangan di dalam perulangan memiliki hubungan yang saling terkait dalam menyelesaikan sebuah kasus.

Jika perulangan luar (*outer loop*) tidak memiliki hubungan terkait dan tidak memiliki kepentingan dalam melakukan proses komputasi, sebaiknya hindari penggunaan *nested loop* karena akan menghabiskan waktu eksekusi yang sia-sia dan program tidak berjalan optimal.

SOAL LATIHAN

1. Dalam lembar dokumen banyak sekali nilai-nilai integer yang kalau nilainya di total akan lebih besar dari 1000. Susun algoritma untuk menginput bilangan-bilangan tersebut satu per satu dan menghitung totalnya. Input berhenti apabila total nilai yang diinput lebih besar dari 500. Kemudian mencetak total tersebut dan proses selesai.

1. Susun algoritma untuk mencetak deret berikut ini:

   a.
   ```
   5   5   5   5   5
   10  10  10  10  10
   15  15  15  15  15
   ```

   b.
   ```
   1  1  1  1  1
   2  2  2  2
   3  3  3
   4  4
   5
   ```

3. Apa yang tercetak jika program berikut dijalankan:

a.

```
#include<stdio.h>
void main()
{  int I, J, T;
   T = 0;
   for ( I=1; I<=5; I+=2)
   { for(J = I; J<=9; J+=3)
     { printf("%4i", J );
     }
        printf("\n);
   }
}
```

b.

```
#include<stdio.h>
void main()
{  int I, J, T;
   T = 0;
   for ( I=1; I<=3; I++)
        {  for(J = I; J<=5; J++)
                { T = T + J;
                  printf("%3i", T );
                }
          printf("\n);
        }
}
```

BAB 8

ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa mampu memahami penggunaan variabel struktur array satu dimensi.

Subpokok Bahasan	:	8.1. Konsep array satu dimensi 8.2. Contoh algoritma operasi dasar array 8.3. Contoh algoritma yang melibatkan array satu dimensi secara sederhana

Daftar Pustaka	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

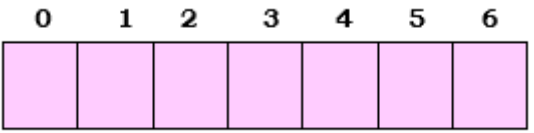
ARRAY SATU DIMENSI

8.1. KONSEP ARRAY SATU DIMENSI

PENGERTIAN ARRAY SATU DIMENSI

Array dapat diartikan sebagai sesuatu yang berbaris atau berderet-deret. Dalam bahasa pemrograman, array adalah variabel sejenis yang berderet-deret sedemikan rupa sehingga alamatnya saling bersambung atau bersebelahan/berdampingan (*contiguous*).

Karena variabel saling bersambung, maka array satu dimensi biasanya diilustrasikan dengan gambar sebagai berikut:



Dari ilustrasi di atas, terlihat sebuah array satu dimensi, yang digambarkan dengan 7 buah kotak. Yang disebut kotak di sini, dalam istilah array disebut dengan elemen, cell (sel), lokasi atau kolom. Jadi array di atas terdiri dari 7 elemen, atau 7 lokasi, atau 7 kolom. Untuk bahasa C/C++ dan Java, elemen pertama diberi nomor 0, yang dilanjutkan dengan nomor 1,2 dan seterusnya.

MENYIAPKAN ARRAY SATU DIMENSI

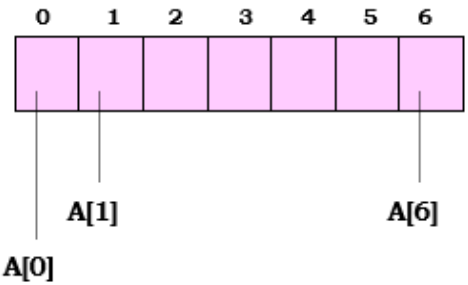
Menyiapkan array satu dimensi dilakukan dengan cara menyebutkan tipe, nama dan jumlah elemen.

Contoh:

C

```
#include<stdio.h>
void main()
{  int A[7];
```

Dengan instruksi int A[7], akan disiapkan sebuah array satu dimensi bertipe int dengan 7 elemen yang diberi nomor indeks (*index*) dari 0 sampai dengan 6 yang diilustrasikan dengan gambar sebagai berikut:



A =variabel array

Karena ada 7 elemen, maka setiap elemen diberi sebutan nama yang berbeda dengan memberikan nomor indeks (*index*), sehingga masing-masing menjadi: A[0], A[1], A[2], sampai dengan A[6], yang biasa disebut (dibaca) dengan:

A dengan indeks 0
A dengan indeks 1
Dan seterusnya

ALAMAT ELEMEN-ELEMEN ARRAY SATU DIMENSI

Alamat elemen suatu array saling bersambung/bersebelahan (*contiguous*). Contoh:

Contoh-1. dengan C

```
#include<stdio.h>
void main()
{ int A[5];
   printf("\n%X",  &A[0] );
   printf("\n%X",  &A[1] );
   printf("\n%X",  &A[2] );
   printf("\n%X",  &A[3] );
   printf("\n%X",  &A[4] );
}
```

Akan tercetak :

21E6
21E8
21EA
21EC
21EE

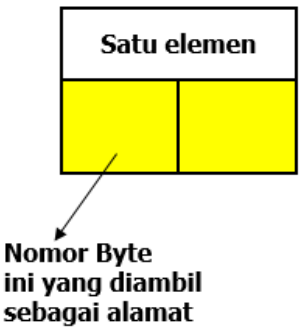
format : %X

integer : 2 Byte

Alamat dicetak dengan notasi bilangan Hexadecimal.
Alamat ini bisa berbeda bila dirun pada komputer(sistem) yang berbeda.

Tanda & artinya alamat, &A[0] artinya alamat elemen A[0]. Sedangkan yang dimaksud dengan alamat adalah nomor byte pertama (diilustrasikan dengan byte paling kiri) dari suatu elemen sebagai berikut:

Satu elemen
integer= 2 byte



8.2. CONTOH ALGORITMA OPERASI DASAR ARRAY

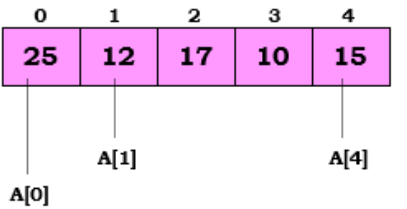
MENYIAPKAN ARRAY NUMERIK SATU DIMENSI LENGKAP DENGAN ISINYA

Contoh:

Algoritma:

Inisialisasi variabel Array A[5]={ 25, 12, 17, 10, 15}

Ilustrasi:



Penulisan dalam Bahasa C (Cara 1)

```
#include <stdio.h>
void main()
{
 int A[5] = { 25,12,17,10,15 };
   -
   -
}
```

Penulisan dalam Bahasa C (Cara 2)

```
#include <stdio.h>
void main()
{
 int A[5] = { 25,12,17,
              10,15 };
   -
   -
}
```

Penulisan dalam Bahasa C (Cara 3)

```
#include <stdio.h>
void main()
{
 int A[ ] = { 25,12,17,10,15 };
   -
   -
}
```

Penulisan dalam Bahasa C (Cara 4)

```
#include <stdio.h>
#define n 5
void main()
{
 int A[n] = { 25,12,17,10,15 };
   -
   -
}
```

Catatan:

1. Jika jumlah data yang akan diisi lebih banyak dari jumlah elemen yang disediakan, maka akan terjadi *error* sewaktu program di-*compile*.

Contoh:

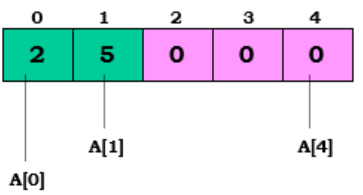
```
#include <stdio.h>
void main()
{
 int A[5 ] = { 25,12,17,10,15, 19 };
   -
   -
}
```

Akan terjadi *error*, karena jumlah data yang disimpan 6 buah, sedangkan jumlah elemen yang disediakan hanya 5 elemen.

2. Untuk array numerik, bila diisi hanya sebagian elemen (mulai dari elemen pertama), maka sisa elemen yang tidak diisi, oleh Bahasa C otomatis diisi dengan nol.

   Contoh:

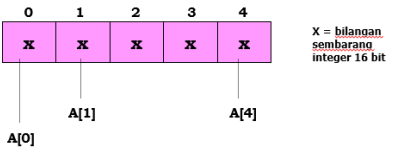
```
#include <stdio.h>
void main()
{
 int A[5 ] = { 2, 5 };
   -
   -
}
```



3. Bila menyiapkan array tanpa dengan isinya, maka lokasi array isinya tetap seperti semula, apa adanya waktu itu. Sehingga isinya tidak diketahui.

   Contoh:

```
#include <stdio.h>
void main()
{
 int A[5 ];
   -
   -
}
```



MENYIAPKAN ARRAY CHARACTER SATU DIMENSI LENGKAP DENGAN ISINYA

Contoh:

Algoritma:

Inisialisasi variabel Array C[5]=”ABCED”

Ilustrasi:

0	1	2	3	4
A	B	C	D	E

Penulisan dalam Bahasa C (Cara 1)

```
#include <stdio.h>
void main()
{
   char C[5] = “ABCDE”;
   -
   -
}
```

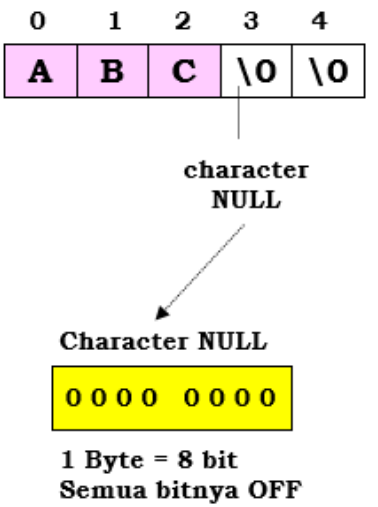
Penulisan dalam Bahasa C (Cara 2)

```
#include <stdio.h>
void main()
{
   char C[5] = { ‘A’, ‘B’, ‘C’, ‘D’, ‘E’ };
   -
   -
}
```

Catatan:

1. Bila diisi hanya sebagian elemen (mulai dari elemen pertama), maka sisa elemen yang tidak diisi, akan diisi dengan karakter NULL.
   Contoh:

```
#include <stdio.h>
void main()
{
   char C[5] = "ABC";
   -
   -
}
```



Bila dicetak dengan:

```
for(I=0; I <= 4;  I++)
     { printf ("%c", C[ I ] ); }
printf("Selesai");
```

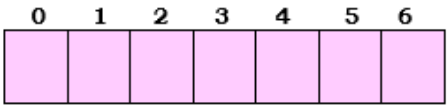
Tercetak :ABC▮▮Selesai

Dua karakter NULL

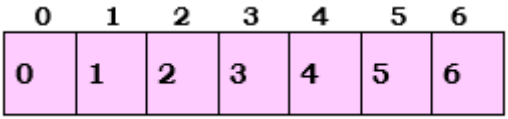
MENGISI ARRAY SATU DIMENSI DENGAN NILAI NUMERIK

Contoh 1:

Sudah ada array integer satu dimensi yang dibuat dengan int A[7], belum ada isinya dengan ilustrasi sebagai berikut:



Buatlah algoritma dan *flowchart* untuk mengisi array A sehingga hasilnya sebagai berikut:

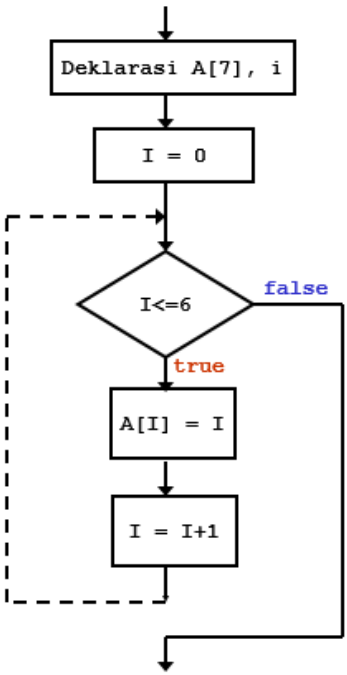


Jawab:

Algoritma:

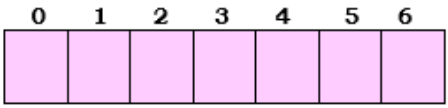
Deklarasi variabel array A[7], I
I = 0
Lakukan selama I<=6
    A[I] = I
I=I+1

Flowchart

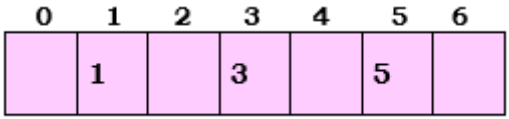


Contoh 2:

Sudah ada array integer satu dimensi yang dibuat dengan int A[7], belum ada isinya dengan ilustrasi sebagai berikut:



Buatlah algoritma dan *flowchart* untuk mengisi array A sehingga hasilnya sebagai berikut:

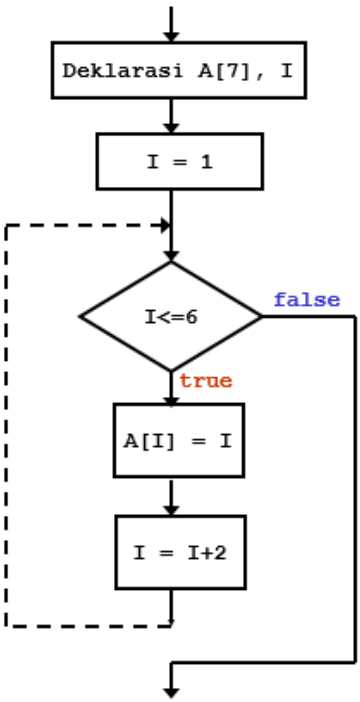


Jawab:

Algoritma:

Deklarasi variabel array A[7], I
I = 1
Lakukan selama I<=6
    A[I] = I
I=I+2

Flowchart



MENGISI ARRAY DENGAN DATA YANG DIKETIK MELALUI *KEYBOARD*

Contoh 1:

Dalam lembar dokumen ada data berupa nilai ujian mahasiswa sebagai berikut:

57, 75, 90, 55, 60, 62, 72, 58, 76, 69, 67, 82, 65, 48, 79, 64, 50, dst.

Sudah ada (sudah dibuat) array satu dimensi int A[11] belum ada isinya. Susun algoritma untuk menginput data dari luar melalui *keyboard*, dan mengisinya ke dalam array sehingga isi array menjadi:

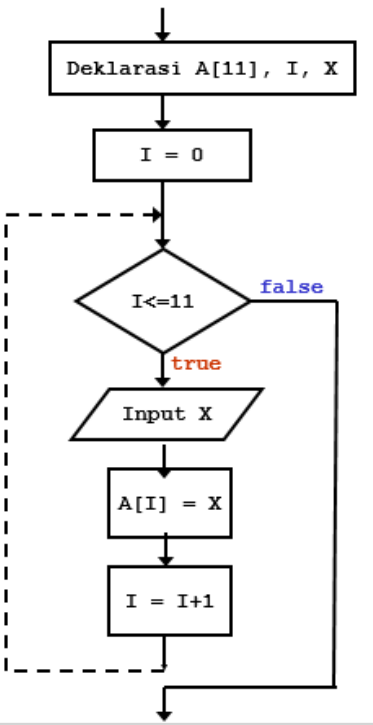
0	1	2	3	4	5	6	7	8	9	10
57	75	90	55	60	62	72	58	76	69	67

Jawab:

Algoritma:

Deklarasi variabel array A[11], I, X
I= 0
Lakukan selama I<=10
    Input X
    A[I] = X
    I=I+1

Flowchart



Program 9.1 Program Mengisi Array Melalui *Keyboard* dalam Bahasa C

```
#include<stdio.h>
void main()
{ int A[[11], X, I;

  for(I=0; I<=10; I++)
   { scanf("%i", &X );
     A[ I ] = X;
  }
}
```

Contoh 2:

Dalam lembar dokumen ada data berupa nilai ujian mahasiswa sebagai berikut:

57, 75, 90, 55, 60, 62, 72, 58, 76, 69, 67, 82, 65, 48, 79, 64, 50, dst.

Sudah ada (sudah dibuat) array satu dimensi int A[11] belum ada isinya. Susun algoritma untuk menginput data dari luar melalui *keyboard*. Bila data yang diinput nilainya < 60, tidak disimpan kedalam array, tetapi bila nilai yang diinput **>=60**, maka akan disimpan ke dalam array sehingga isi array menjadi:

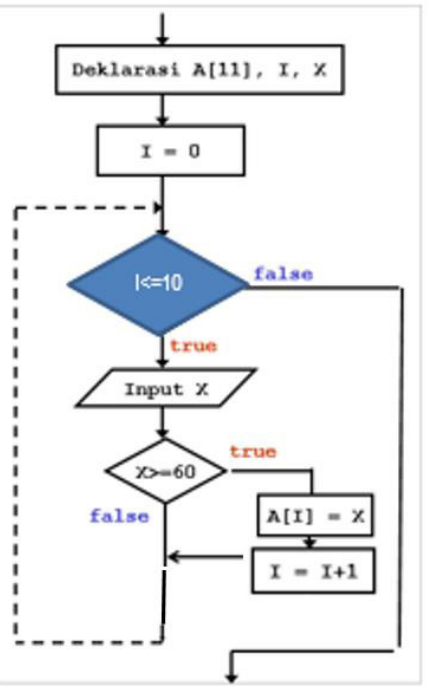
0	1	2	3	4	5	6	7	8	9	10
75	90	60	62	72	76	69	67	82	65	79

Jawab:

Algoritma:

Deklarasi variabel array A[11], I, X
I= 0
Lakukan selama I<=10
    Input X
    Cek apakah X>= 60
        Jika benar maka A[I] = X
        I=I+1

Flowchart



Program 9.2 Program Mengisi Array Melalui *Keyboard* untuk Isi Array >= 60

```
#include<stdio.h>
void main()
{int A[11],X,I;

 I=0
 while (I<=10)
 { scanf ("%i", &X);
   if (X>=60)
   {  A[I]= X;
      I++;
   }
}
```

MENCETAK ISI ARRAY SATU DIMENSI

Contoh 1:

Sudah ada array satu dimensi A[11], sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Susun algoritma dan *flowchart* untuk mencetak isi array sehingga tercetak dengan urutan sebagai berikut:

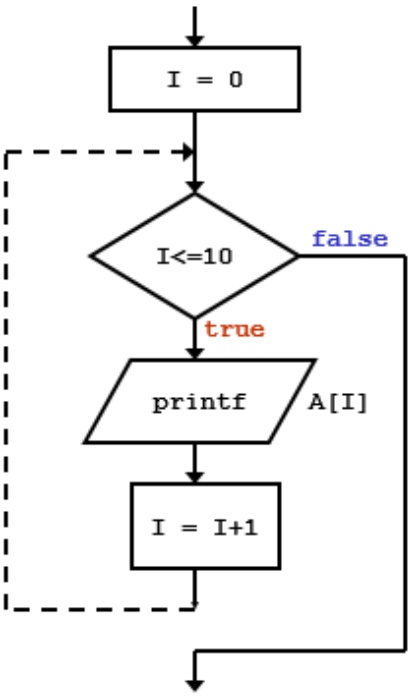
12 17 10 5 15 25 11 8 3 16 19

Jawab:

Algoritma:

Deklarasi variabel array A[11], I
I= 0
Lakukan selama I<=10
    Cetak A[I]
    I=I+1

Flowchart



Contoh 2:

Sudah ada array satu dimensi A[11], sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Susun algoritma dan *flowchart* untuk mencetak isi array sehingga tercetak dengan urutan sebagai berikut:

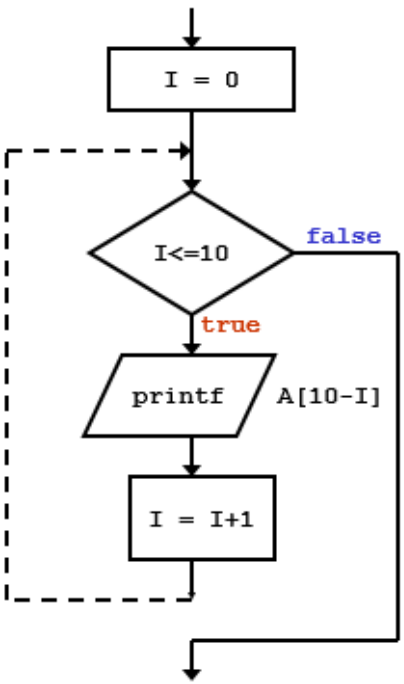
19 16 3 8 11 25 15 5 10 17 12

Jawab:

Algoritma:

Deklarasi variabel array A[11], I
I= 0
Lakukan selama I<=10
    Cetak A[10-I]
    I=I+1

Flowchart



Contoh 3:

Sudah ada array satu dimensi A[11], sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Susun algoritma dan *flowchart* untuk mencetak isi array yang nilainya **lebih besar dari 10** sehingga tercetak dengan urutan sebagai berikut:

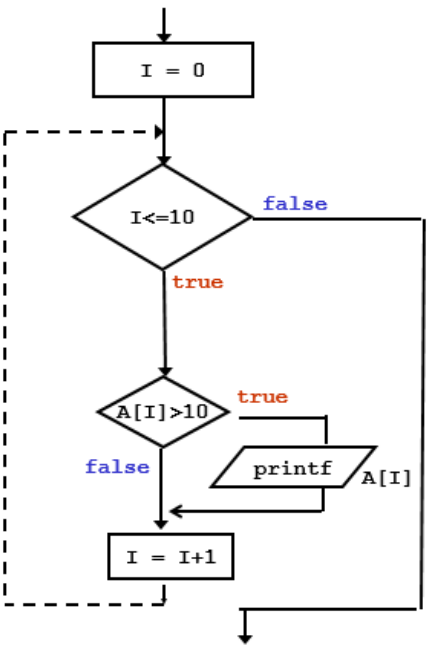
12 17 15 25 11 16 19

Jawab:

Algoritma:

Deklarasi variabel array A[11], I
I= 0
Lakukan selama I<=10
    Cek apakah A[I] > 10
        Jika benar maka
Cetak A[I]
    I=I+1

Flowchart



Program 9.3 Program Mencetak Isi Array yang Nilainya > 10

```
#include<stdio.h>
void main()
{ int A[[11]={12,17,10,5,15,25,11,8,3,16,19}, I;

  for(I=0; I<=10; I++)
   { if (A[I]>10)
        printf("%3i", A[I] );
   }
}
```

8.3. CONTOH ALGORITMA YANG MELIBATKAN ARRAY SATU DIMENSI SECARA SEDERHANA

Contoh 1:

Sudah ada array satu dimensi yang dibuat dengan int A[11]. Sudah ada isinya nilai-nilai numerik. Susun algoritma untuk mencetak Total isi array tersebut.

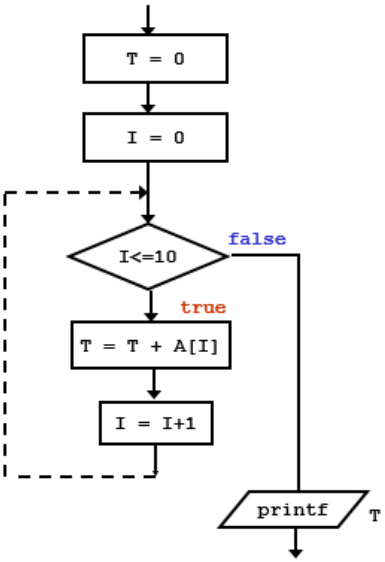
0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Jawab:

Algoritma:

Deklarasi variabel array A[11]={12,17,10,5,15,25,11,8,3,16,19}, T, I
T=0
I=0
Lakukan selama I<=10
    T = T + A[I]
I=I+1
Cetak T

***Flowchart*:**



Contoh 2:

Sudah ada array satu dimensi yang dibuat dengan char A[15]. Sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Susun algoritma untuk mencetak isi array tersebut tercetak dengan urutan sebagai berikut:

A B C D E
F G H I J
K L M N O

Jawab:

Algoritma:

Deklarasi variabel array C[15]={'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O'},
N, I, J

N=0
I=1
Lakukan selama I<=3
J=1
Lakukan selama J<=5
Cetak array A[N]
N++
J++
I++

Program 9.4 Program Mencetak Isi Array Karakter

```
#include<stdio.h>
void main()
{ char C[15]={'A','B','C','D','E','F','G','H','I','J','K','
L','M','N','O'}, I, J, N;
    N = 0;
    for(I=1; I <= 3; I++)
        { for(J=1; J<=5; J++)
              {  printf("%2c", A[ N ] );
                  N = N + 1;
               }
          printf("\n");
      }
}
```

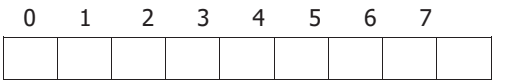
```
#include<stdio.h>
void main()
{ char
C[15]={'A','B','C','D','E','F','G','H','I','J','K','L','M'
,'N','O'}, I, J, N;
    N = 0;
    for(I=1; I <= 3; I++)
        { for(J=1; J<=5; J++)
              {  printf("%2c", A[ N ] );
                  N = N + 1;
               }
          printf("\n");
      }
}
```

KESIMPULAN

1. Array adalah sekumpulan variabel yang memiliki tipe data yang sama dan dinyatakan dengan nama yang sama.
2. Array menggunakan indeks integer untuk menentukan urutan elemen-elemennya, di mana elemen pertama dimulai dengan indeks 0, elemen kedua 1, dst.
3. Array memiliki ukuran yang tetap dalam arti tidak dapat membesar atau mengecil ukurannya setelah didefinisikan.

SOAL LATIHAN

1. Sudah ada array A satu dimensi yang dibuat dengan int A[9]. Belum ada isinya dengan ilustrasi sebagai berikut:



Gambarkan kembali array tersebut lengkap dengan isinya bila diisi dengan penggalan program sebagai berikut:

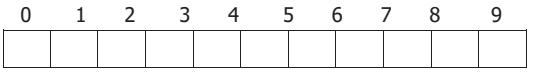
a.

```
I = 0;
while (I<=8)
{ A[I] = I
  I+=2;
}
```

b.

```
For (I=0;I<=8;I++)
{  A[I] = I*2+1
```

2. Sudah ada array A satu dimensi yang dibuat dengan int A[11]. Belum ada isinya dengan ilustrasi sebagai berikut:



Susun algoritma untuk mengisi array tersebut sehingga isinya menjadi sebagai berikut:

a.

0	1	2	3	4	5	6	7	8	9
1		2		3		4		5		6

b.

0	1	2	3	4	5	6	7	8	9
1		3		5		7		9		11

3. Sudah ada array A satu dimensi yang dibuat dengan int A[11]. Belum ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9

Dalam dokumen ada data yang tertulis dengan urutan sebagai berikut:
12, 15, 7, 10, 5, 2, 17, 25, 9, 20, 35, 28, 14, 11, 999 (999 sebagai *end of data*)
Susun algoritma untuk menginput data dalam dokumen di atas, satu persatu (mulai dari urutan paling kiri) dan menyimpannya kedalam array sehingga isi array menjadi:

a. Yang disimpan hanya bila nilai yang diinput yang bernilai **genap.**

0	1	2	3	4	5	6	7	8	9
12	10	2	10	28	14					

b. Yang disimpan bila data yang diinput bernilai **lebih besar dari 10**.

0	1	2	3	4	5	6	7	8	9
12	15	17	25	20	35	28	14	11		

BAB 9

MANIPULASI ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa mampu memahami beberapa operasi dasar array satu dimensi.

Subpokok Bahasan	:	9.1. Algoritma dasar manipulasi array satu dimensi 9.2. Penelusuran array satu dimensi 9.3. Contoh penyelesaian persoalan dengan array satu dimensi

Daftar Pustaka	:	Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media. Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205

9.1. ALGORITMA DASAR MANIPULASI ARRAY SATU DIMENSI

MENYALIN ISI ARRAY KE ARRAY LAIN

Contoh 1:

Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

A.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

B.

0	1	2	3	4	5	6	7	8	9	10

Susun algoritma untuk menyalin isi array A ke array B, sehingga isi array B sama dengan isi array A, seperti gambar berikut ini:

B.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Jawab:

Cara 1:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I
I=0
Lakukan selama I<=10
     array B[I] diisi dengan array A[I]
I=I+1
```

Ket: Disalin dari A[0] sampai dengan A[10]

Cara 2:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I, J
I=0
J=0
Lakukan selama I<=10
     array B[J] diisi dengan array A[I]
J=J+1
I=I+1
```

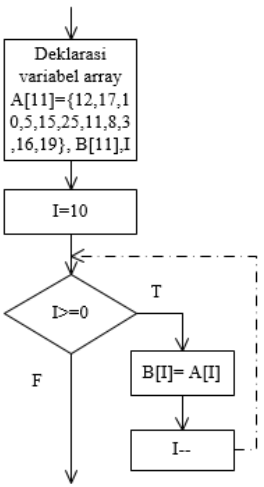
Ket: Disalin dari A[0] sampai dengan A[10]. A dengan indeks I, dan B dengan indeks J

Cara 3:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I
I=10
Lakukan selama I>=0
     array B[I] diisi dengan array A[I]
I=I-1
```

Ket: Disalin dari A[10] sampai dengan A[0]

***Flowchart* cara 3:**



Program 10.1 Program Menyalin Isi Array ke Array Lain Contoh 1 (Cara 3) Dalam Bahasa C

```
#include<stdio.h>
void main()

{ int A[[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11],I;

  for(I=10; I>=0; I--)
   {
     B[I] = A[I];
  }
}
```

Contoh 2:

Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

A.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

B.

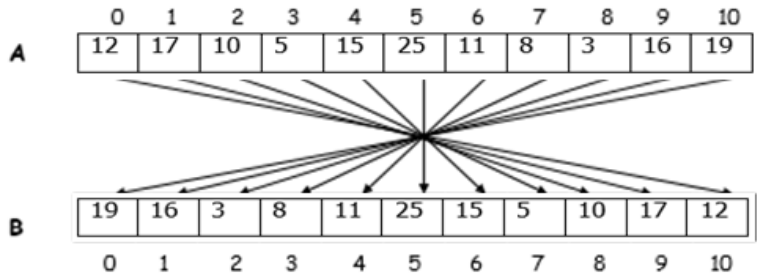
0	1	2	3	4	5	6	7	8	9	10

Susun algoritma untuk menyalin isi array A ke array B, sehingga isi array B sama dengan isi array A tetapi dengan urutan terbalik, seperti gambar berikut ini:

B.

0	1	2	3	4	5	6	7	8	9	10
19	16	3	8	11	25	15	5	10	17	12

Ilustrasi Proses:



Jawab:

Cara 1:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I

I=0
Lakukan selama I<=10
     array B[10-I] diisi dengan array A[I]
I=I+1
```

Ket: Disalin dari A[0] ke B[10] sampai dengan A[10] ke B[0]

Cara 2:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I, J

I=0
J=10
Lakukan selama I<=10
     array B[J] diisi dengan array A[I]
J=J-1
I=I+1
```

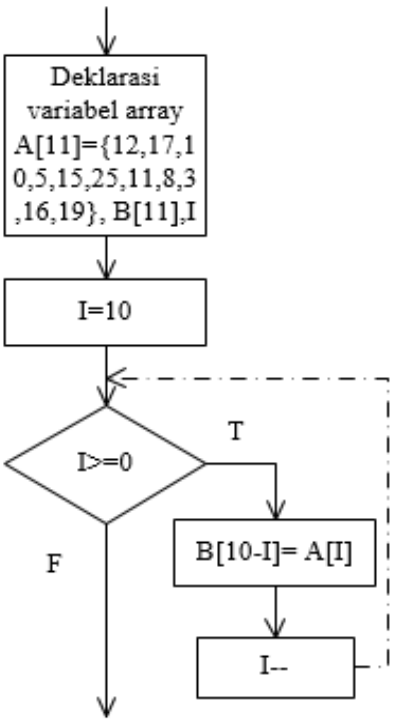
Ket: Disalin dari A[0] ke B[10] sampai dengan A[10] ke B[0] di mana A dengan indeks I, dan B dengan indeks J

Cara 3:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I
I=10
Lakukan selama I>=0
     array B[10-I] diisi dengan array A[I]
I=I-1
```

Ket: Disalin dari A[10] ke B[0] sampai dengan A[0] ke B[10]

***Flowchart* cara 3:**



Program 10.2 Program Menyalin isi Array ke Array Lain Contoh 2 (Cara 3) dalam Bahasa C

```
#include<stdio.h>
void main()

{ int A[[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11],I;

  for(I=10; I>=0; I--)
   {
     B[10-I] = A[I];
  }
}
```

Contoh 3:

Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

A.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

B.

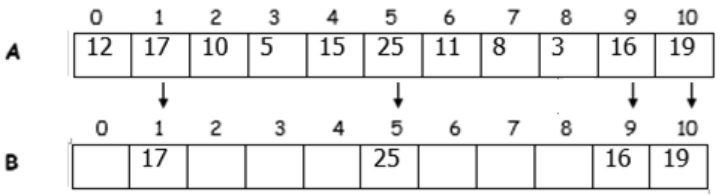
0	1	2	3	4	5	6	7	8	9	10

Susun algoritma untuk menyalin isi array A ke array B yang nilainya lebih besar dari 15 ke array B pada kolom yang sama, sehingga isi array B seperti berikut ini:

B.

	17				25				16	19

Ilustrasi Proses:



Jawab:

Cara 1:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I
I=0
Lakukan selama I<=10
     Cek apakah array A[I] > 15
          Jika ya, array B[I] diisi dengan array A[I]
I=I+1
```

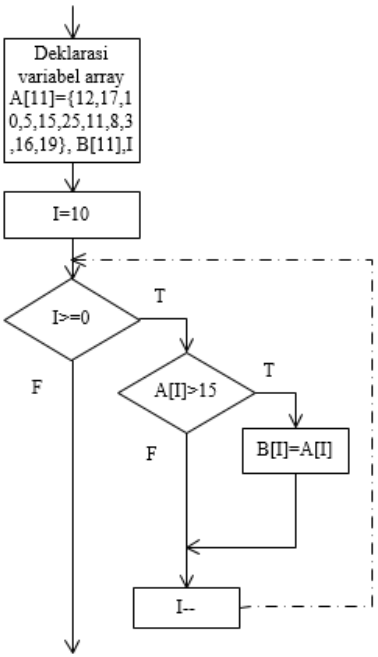
Cara 2:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I, J
I=0
J=0
Lakukan selama I<=10
     Cek apakah array A[I] > 15
          Jika ya, array B[J] diisi dengan array A[I]
J=J+1
I=I+1
```

Cara 3:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I

I=10
Lakukan selama I>=0
     Cek apakah array A[I] > 15
          Jika ya, array B[I] diisi dengan array A[I]
I=I-1
```

***Flowchart* cara 3:**



Program 10.3 Program Menyalin Isi Array ke Array Lain Contoh 3 (Cara 3) dalam Bahasa C

```
#include<stdio.h>
void main()
{ int A[[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11],I;

  for(I=10; I>=0; I--)
   {
      If (A[I]>15)
          B[I] = A[I];
  }
}
```

Contoh 4:

Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

A.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

B.

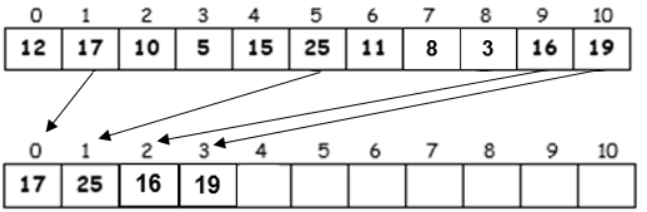
0	1	2	3	4	5	6	7	8	9	10

Susun algoritma untuk menyalin isi array A ke array B yang nilainya lebih besar dari 15 ke array B di mana kolom dari array B dimulai dari B[0], sehingga isi array B seperti berikut ini:

B.

0	1	2	3	4	5	6	7	8	9	10
17	25	16	19							

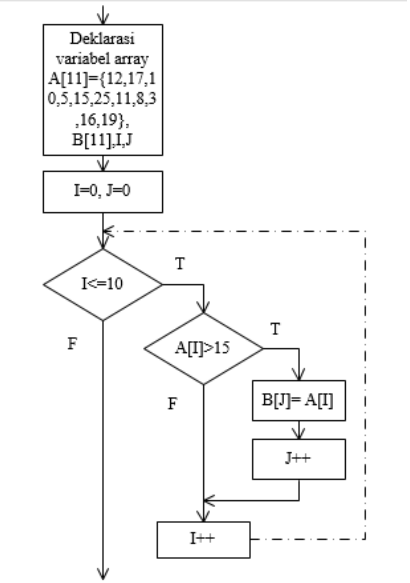
Ilustrasi Proses:



Jawab:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I, J

I=0
J=0
Lakukan selama I<=10
     Cek apakah array A[I] > 15
        { Jika ya, array B[J] diisi dengan array A[I]
          J=J+1 }
I=I+1
```

***Flowchart*:**



Program 10.4 Program Menyalin isi Array ke Array Lain Contoh 4 dalam Bahasa C

```
#include<stdio.h>
void main()
{ int A[[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11],I, J;
  I=0;
  J=0;
  while(I<=10)
   {
      If (A[I]>15)
          {B[J] = A[I];
           J++;}
      I++;
   }
}
```

9.2. PENELUSURAN ARRAY SATU DIMENSI

Contoh 1:

Sudah ada array A satu dimensi yang dibuat dengan int A[11], sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10
12	15	7	10	5	2	10	25	9	20	35

Apa yang tercetak bila array di atas dicetak dengan instruksi (penggalan program) berikut ini:

a.
```
For (I=0; I<=10; I++)
{ printf ("%4i", A[10-I];}
```

Jawab:

Perhatikan tabel berikut:

I	I<=10	Cetak A [10 – I]	I++
0	T	A[10]=35	1
1	T	A[9]=20	2
2	T	A[8]=9	3
3	T	A[7]=25	4
4	T	A[6]=10	5
5	T	A[5]=2	6
6	T	A[4]=5	7
7	T	A[3]=10	8
8	T	A[2]=7	9
9	T	A[1]=15	10
10	T	A[0]=12	11
11	F	-	

Hasil: 35 20 9 25 10 2 5 10 7 15 12

b.
```
For (I=0; I<=10; I++)
{ if (I%2 == 0)
    printf ("%4i", A[I];}
```

Jawab:

Perhatikan tabel berikut:

I	I<=10	I%2==0	Cetak A [I]	I++
0	T	T	A[0] = 12	**1**
1	T	F	-	**2**
2	T	T	A[2] = 7	**3**
3	T	F	-	**4**
4	T	T	A[4] = 5	**5**
5	T	F	-	**6**
6	T	T	A[6] = 10	**7**
7	T	F	-	**8**
8	T	T	A[8] = 9	**9**
9	T	F	-	**10**
10	T	T	A[10]= 35	**11**
11	F	-	-	-

Hasil: 12 7 5 10 9 35

Contoh 2:

Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

A.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

B.

0	1	2	3	4	5	6	7	8	9	10

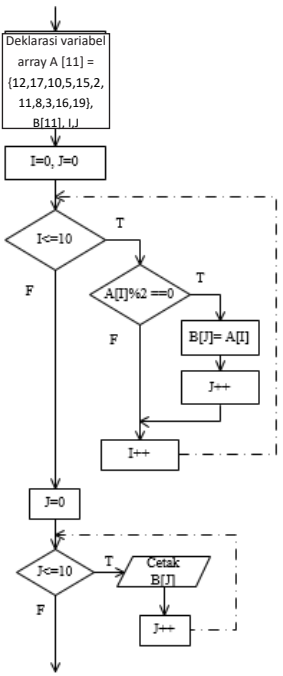
Susun algoritma untuk menyalin isi array A ke array B yang **nilainya merupakan bilangan genap** ke array B di mana kolom dari array B **dimulai dari B[0]** serta **tampilkan ke layar isi array B**. Berikut adalah ilustrasi isi array B:

0	1	2	3	4	5	6	7	8	9	10
12	10	8	16							

Jawab:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11],
I, J
I=0
J=0
Lakukan selama I<=10
     Cek apakah array A[I] % 2 == 0
        { Jika ya, array B[J] diisi dengan array A[I]
          J=J+1 }
I=I+1

//algoritma untuk mencetak isi array B
J=0
Lakukan selama J<=10
     Cetak array B[J]
J++
```

***Flowchart*:**



Program 10.5 Program Penelusuran Array Contoh 2 dalam Bahasa C

```
#include<stdio.h>
void main()
{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, B[11],I, J;
  I=0;
  J=0;
  while(I<=10)
   {
      If (A[I]%2 == 0)
          {B[J] = A[I];
           J++;}
      I++;
   }

   J=0;
   while (J<=0)
   {  printf ("%3i",B[J]);
   }
}
```

9.3. CONTOH PENYELESAIAN PERSOALAN DENGAN ARRAY SATU DIMENSI

Contoh 1:

Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

A.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

B.

0	1	2	3	4	5	6	7	8	9	10

Susun algoritma dan *flowchart* untuk menyalin isi array A yang **nilainya lebih kecil dari 17** ke array B sehingga isi array B menjadi sebagai berikut:

B

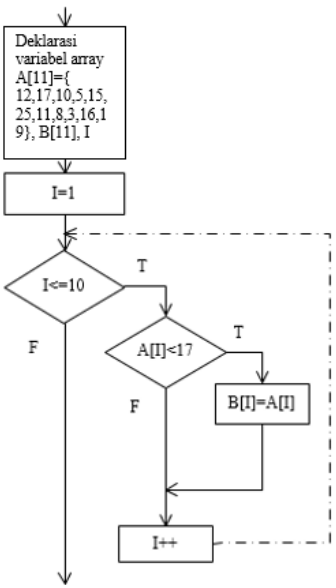
0	1	2	3	4	5	6	7	8	9	10
12		10	5	15		11	8	3	16	

Jawab:

Algoritma:

```
Deklarasi variabel
A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11], I

I=0
Lakukan selama I<=10
     Cek apakah array A[I] < 17
        Jika ya, array B[I] diisi dengan array A[I]
I=I+1
```

***Flowchart*:**



Program 10.6 Program Penyelesaian Persoalan Array Contoh 1a dalam Bahasa C

```
#include<stdio.h>
void main()
{ int A[11]={12,17,10,5,15,25,11,8,3,16,19}, B[11],I;

  I=0;
  while(I<=10)
   {
      If (A[I]<17)
          B[I] = A[I];
      I++;
   }
 }
```

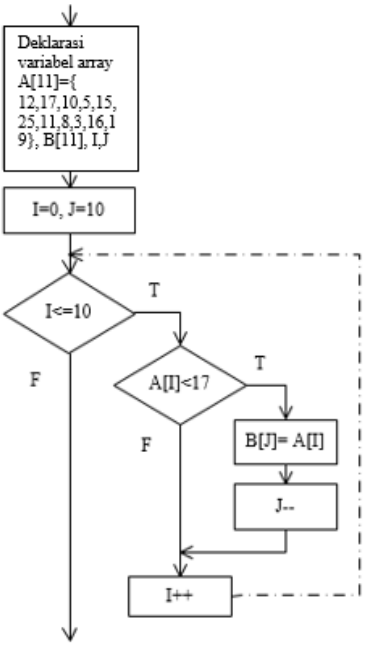
B	0	1	2	3	4	5	6	7	8	9	10
				16	3	8	11	15	5	10	12

Jawab:

Algoritma:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19},
B[11], I, J

I=0
J=10
Lakukan selama I<=10
     Cek apakah array A[I] < 17
        Jika ya, array B[J] diisi dengan array A[I]
        J--
I=I+1
```

***Flowchart*:**



Program 10.7 Program Penyelesaian Persoalan Array Contoh 1b dalam Bahasa C

```
#include<stdio.h>
void main()
{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, B[11],I,J;

  J=10;
  I=0;
  while(I<=10)
   {
      If (A[I]<17)
          { B[J] = A[I];
            J--;
          }
      I++;
   }
 }
```

KESIMPULAN

Konsep array bisa juga digunakan untuk mamanipulasi data, salah satunya dengan cara menyalin isi dari suatu array ke array lain.

SOAL LATIHAN

1. Sudah ada array A dan array B yang dibuat dengan int A[11], dan int B[11]. Array A sudah ada isinya, dan array B belum ada isinya dengan ilustrasi sebagai berikut:

a.

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

b.

0	1	2	3	4	5	6	7	8	9	10

Susun algoritma dan *flowchart* untuk menyalin isi array A yang merupakan **bilangan ganjil** ke array B sehingga isi array B menjadi sebagai berikut:

a.

B

0	1	2	3	4	5	6	7	8	9	10
17	5	15	25	11	3	19				

b.

B

0	1	2	3	4	5	6	7	8	9	10
				17	5	15	25	11	3	19

BAB 10

SEARCHING ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa mampu memahami konsep pencarian (*searching*) pada array satu dimensi.
Subpokok Bahasan	:	10.1. Pencarian pada array satu dimensi 10.2. Contoh algoritma pencarian sekuensial (*sequential searching*)
Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

SEARCHING ARRAY SATU DIMENSI

10.1. PENCARIAN PADA ARRAY SATU DIMENSI

PENGERTIAN SEARCH

Secara umum *search* diartikan mencari data dengan cara menelusuri tempat penyimpanan data tersebut. Tempat penyimpanan data dapat berupa array dalam *memory*, bisa juga berada dalam suatu *file* pada *external storage*. *Searching* adalah proses mencari atau pencarian. Data yang di-*search* adalah data yang berada dalam array satu dimensi. Untuk data yang berada dalam array satu dimensi, ada beberapa cara *searching* atau proses pencarian di antaranya:

1. *Sequential Search*
2. *Index Sequential Search*
3. *Binary Search*
4. *Fibonacci Search*

Dalam modul ini yang akan dibahas adalah ***Sequential Search*.**

10.2. CONTOH ALGORITMA PENCARIAN SEKUENSIAL

PENGERTIAN PENCARIAN SEKUENSIAL

Pencarian sekuensial merupakan metode pencarian data dalam array dengan cara membandingkan data yang dicari dengan data yang ada di dalam array secara berurutan.

Pencarian data dengan metode *sequential search* efektif untuk mencari data yang dalam posisi tidak terurut atau acak. Prosesnya bisa dijelaskan seperti berikut:

1. Menentukan data yang dicari.
2. Membaca data array satu per satu secara sekuensial.
3. Mulai dari data pertama sampai dengan data terakhir, kemudian data yang dicari dibandingkan dengan masing-masing data yang ada di dalam array. Jika data ditemukan, maka dapat dibuat statement bahwa data telah ditemukan. Namun jika tidak ditemukan, maka dibuat statement bahwa data tidak ditemukan.

SEQUENTIAL SEARCH MENCARI ADA ATAU TIDAK ADA SUATU NILAI DALAM ARRAY SATU DIMENSI

Contoh 1:
Sudah ada array A satu dimensi yang dibuat dengan int A[11], sudah ada isinya dengan ilustrasi sebagai berikut:

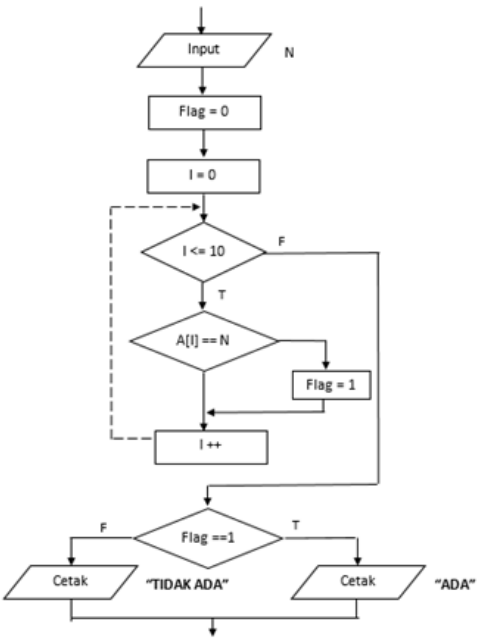
0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Susun algoritma dan flowchart untuk menginput sebuah nilai integer (misal N), kemudian periksa isi array, apakah ada isi array yang (nilainya) sama dengan N. Bila ada, cetak perkataan "**ADA**", bila tidak ada cetak perkataan "**TIDAK ADA**".

Jawab Cara 1:
Algoritma:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19},
N, I, Flag

Input Nilai N
Flag = 0
I=0
Lakukan selama I<=10
    Jika array A[I] == N
         Maka Flag = 1
    I=I+1

Cek apakah Flag == 1
   Jika Ya, Cetak "ADA"
   Jika Tidak, Cetak "TIDAK ADA"
```

***Flowchart*:**



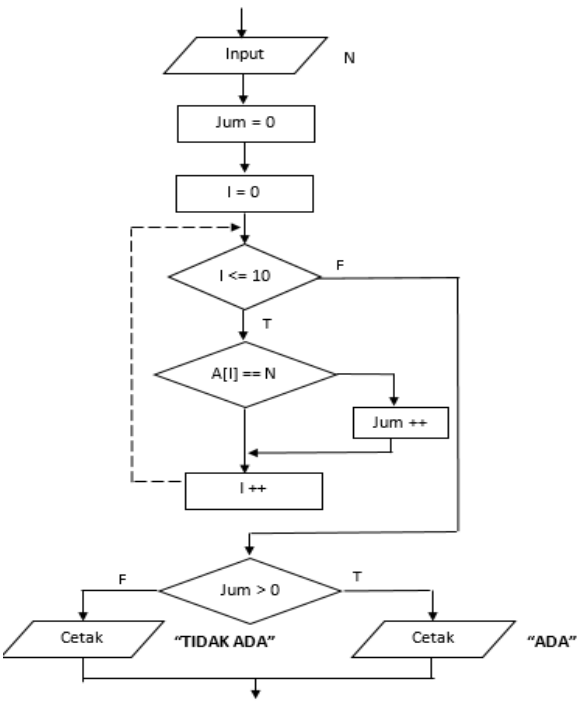
Program 11.1 Program Contoh 1 (Cara 1) dalam Bahasa C

```
#include<stdio.h>
void main()

{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, N,I, Flag;

  scanf ("%i", &N);
  Flag= 0;
  I=0;
  while (I<=10)
  {  if (A[I] == N)
        Flag = 1;
     I++;
  }

  if (Flag == 1)
     printf ("ADA");
  else
     printf ("TIDAK ADA");
}
```

Cara 2:
Algoritma:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19}, N,
I, Jum

Input Nilai N
Jum = 0
I=0
Lakukan selama I<=10
    Jika array A[I] == N
         Maka Jum=Jum+1
    I=I+1

Cek apakah Jum > 0
   Jika Ya, Cetak "ADA"
   Jika Tidak, Cetak "TIDAK ADA"
```

***Flowchart*:**



Program 11.2 Program Contoh 1 (Cara 2) dalam Bahasa C

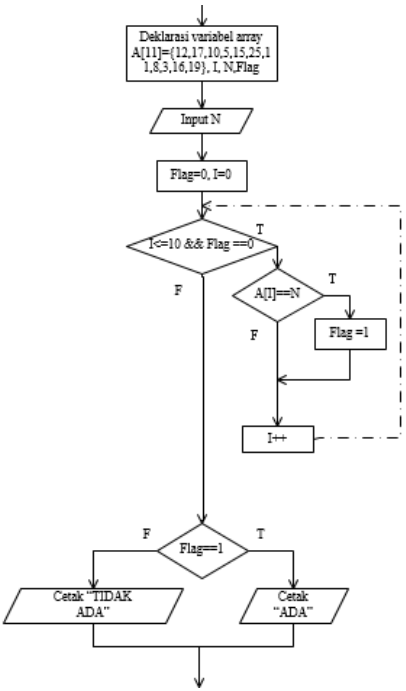
```
#include<stdio.h>
void main()

{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, N,I, Jum;

  scanf ("%i", &N);
  Jum= 0;
  I=0;
  while (I<=10)
  {  if (A[I] == N)
        Jum++;
     I++;
  }

  if (Jum > 0)
     printf ("ADA");
  else
     printf ("TIDAK ADA");
}
```

Cara 3:

Algoritma:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19},
N, I, Flag

Input Nilai N
Flag = 0
I=0
Lakukan selama I<=10 dan Flag == 0
    Jika array A[I] == N
         Maka Flag = 1
    I=I+1

Cek apakah Flag == 1
   Jika Ya, Cetak "ADA"
   Jika Tidak, Cetak "TIDAK ADA"
```

***Flowchart*:**



Program 11.3 Program Contoh 1 (Cara 3) dalam Bahasa C

```
#include<stdio.h>
void main()

{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, N,I, Flag;

  scanf ("%i", &N);
  Flag= 0;
  I=0;
  while (I<=10 && Flag == 0)
  {  if (A[I] == N)
        Flag = 1;
     I++;
  }

  if (Flag == 1)
     printf ("ADA");
  else
     printf ("TIDAK ADA");
}
```

Contoh 2:
Sudah ada array satu dimensi yang dibuat dengan int A[11]. Sudah ada isinya dengan ilustrasi sebagai berikut:

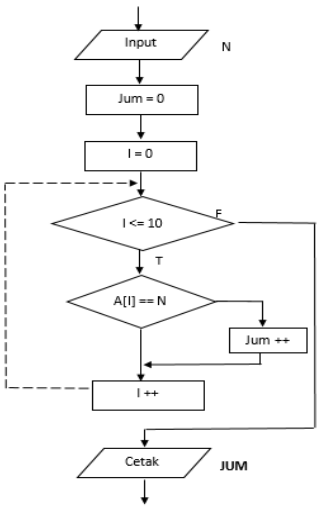
0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Susun algoritma dan flowchart untuk menginput sebuah nilai integer (misal N), kemudian periksa isi array, dan cetak ada berapa buah isi array yang nilainya sama dengan nilai N.

Jawab:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19}, N,
I, Jum

Input Nilai N
Jum = 0
I=0
Lakukan selama I<=10
    Jika array A[I] == N
         Maka Jum=Jum+1
    I=I+1

Cetak JUM
```

***Flowchart*:**



Program 11.4 Program Contoh 2 dalam Bahasa C

```
#include<stdio.h>
void main()

{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, N,I, Jum;

  scanf ("%i", &N);
  Jum= 0;
  I=0;
  while (I<=10)
  {  if (A[I] == N)
        Jum++;
     I++;
  }

  printf("%i Buah", Jum);
}
```

Contoh 3:
Sudah ada array satu dimensi yang dibuat dengan int A[11]. Sudah ada isinya dengan ilustrasi sebagai berikut:

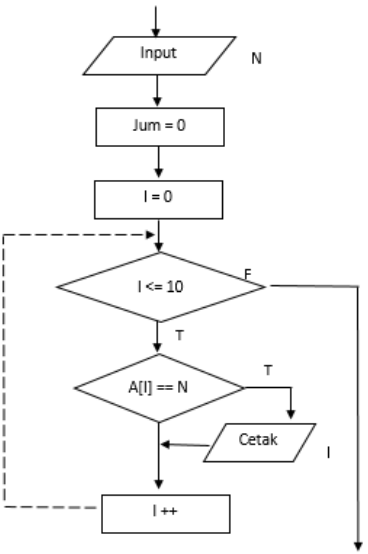
0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	8	3	16	19

Susun algoritma dan flowchart untuk menginput sebuah nilai integer (misal N), kemudian periksa isi array, dan cetak ada di lokasi mana saja isi array yang nilainya sama dengan N.

Jawab:

```
Deklarasi variabel A[11]={12,17,10,5,15,25,11,8,3,16,19}, N, I,
Jum
Input Nilai N
Jum = 0
I=0
Lakukan selama I<=10
    Jika array A[I] == N
         Cetak I
I=I+1
```

***Flowchart*:**



Program 11.5 Program Contoh 3 dalam Bahasa C

```
#include<stdio.h>
void main()

{ int A[[11]={ 12,17,10,5,15,25,11,8,3,16,19}, N,I, Jum;

  scanf ("%i", &N);
  Jum= 0;
  I=0;
  while (I<=10)
  {  if (A[I] == N)
        printf("%i", I);
     I++;
  }
}
```

SEQUENTIAL SEARCH MEMERIKSA APAKAH DI ANTARA 2 BUAH ARRAY ADA NILAI YANG SAMA

Contoh:

Sudah ada array satu dimensi A dan B yang dibuat dengan char A[5] dan char B[7]. Sudah ada isinya huruf-huruf kapital tanpa spasi dengan contoh ilustrasi sebagai berikut:

	0	1	2	3
A	B	O	G	O	R

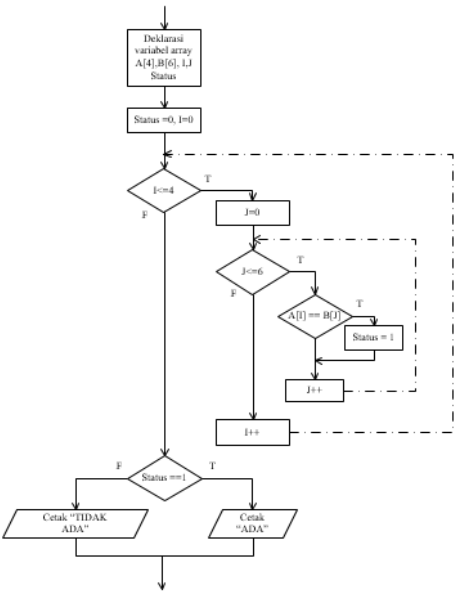
	0	1	2	3	4	5	
B	J	A	K	A	R	T	A

Susun algoritma dan flowchart untuk memeriksa apakah di antara isi array A ada yang sama dengan isi array B. Bila ada cetak perkataan “ADA”, bila tidak cetak perkataan “TIDAK ADA”.

Jawab:

```
Deklarasi variabel A[5]=”BOGOR”, B[7]=”JAKARTA”, Status, I, J

Status = 0
I=0
Lakukan selama I<=4
    J=0
    Lakukan selama J<=6
       Cek apakah A[I] == B[J]
           Jika Ya, maka Status = 1
        J++
I++

Cek jika Status == 1
   Jika benar, maka Cetak “ADA”
   Jika salah, maka cetak “TIDAK ADA”
```

***Flowchart*:**



Program 11.6 Program *Sequential Search* dalam Bahasa C

```
#include<stdio.h>
void main()

{ char A[5]="BOGOR", B[7]="JAKARTA";
  int Status,I, J;

  Status = 0;
  I=0;
  while (I<=4)
   { J=0;
    while (J<=6)
       { if (A[I] == B[J])
             Status = 1;
          J++;
       }
    I++;
   }

  if (Status == 1)
     printf("ADA");
   else
     printf("TIDAK ADA");
}
```

KESIMPULAN

1. Untuk mencari data dalam suatu deret array, kita bisa menggunakan konsep searching salah satunya yaitu sequential searching, yaitu mencari data secara urut dari suatu array.
2. Sequential searching memiliki kekurangan dan kelebihan. Kelebihannya adalah lebih mudah untuk diimplementasikan dalam pemrograman. Sedangkan kekurangannya, jika data yang terdapat dalam array jumlahnya banyak, maka akan diperlukan waktu yang lebih lama untuk membandingkan data yang dicari dengan jumlah data yang sangat banyak dalam suatu array.

SOAL LATIHAN

1. Sudah ada array A yang dibuat dengan int A[11], sudah ada isinya yaitu nilai-nilai ujian mahasiswa. Susun algoritma untuk memeriksa apakah ada mahasiswa yang mendapat nilai 90 ke atas. Bila ada cetak perkataan "ADA", bila tidak ada cetak perkataan "TIDAK ADA".

2. Sudah ada array A satu dimensi yang dibuat dengan int A[10]. Sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9
12	15	7	10	25	2	17	25	5	20

Susun algoritma untuk menginput sebuah nilai integer (misal N), kemudian periksa isi array.

a. Bila ada isi array yang nilainya sama dengan N, maka cetak perkataan "FOUND", bila tidak ada isi array yang nilainya sama dengan N, maka cetak perkataan "NOT FOUND".

b. Bila ada isi array yang nilainya sama dengan N, maka cetak perkataan "FOUND", dan mencetak ada berapa buah nilai yang sama dengan N.
Bila tidak ada isi array yang nilainya sama dengan N, maka cetak perkataan "NOT FOUND".

c. Bila ada isi array yang nilainya sama dengan N, maka cetak perkataan “FOUND”, dan mencetak ada berapa buah nilai yang sama dengan N, serta mencetak nomor lokasi tempat nilai yang sama dengan N berada.
Bila tidak ada isi array yang nilainya sama dengan N, maka cetak perkataan “NOT FOUND”.

BAB 11

PENELUSURAN ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa dapat memahami konsep penelusuran array untuk mencari nilai terbesar, nilai terkecil, dan kriteria lainnya di dalam array satu dimensi.
Subpokok Bahasan	:	11.1. Konsep perbandingan sebagai dasar penentuan nilai terbesar dan terkecil 11.2. Teknik pencarian nilai terbesar atau terkecil pada array satu dimensi 11.3. Teknik pencarian nilai terbesar atau terkecil menggunakan algoritma sekuensial 11.4. Teknik pencarian nilai terbesar atau terkecil menggunakan algoritma sentinel
Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

PENELUSURAN ARRAY SATU DIMENSI

11.1. KONSEP PERBANDINGAN

Perbandingan merupakan salah satu konsep algoritmik yang hampir selalu digunakan. Algoritma seperti penelusuran, pencarian dan pengurutan hampir selalu melibatkan proses perbandingan. Pada dasarkan perbandingan merupakan proses yang melibatkan operator perbandingan. Seperti sudah dijelaskan di modul pertemuan ke-3, operator perbandingan terdiri dari 6 (enam) jenis yang tersaji pada Tabel 1.

Tabel 1. Operator Perbandingan

Operator	Arti	Contoh	Keterangan
<	Kurang dari	x < y	Apakah x kurang dari y
<=	Kurang dari sama dengan	x <= y	Apakah x kurang dari sama dengan y
>	Lebih dari	x > y	Apakah x lebih dari y
>=	Lebih dari sama dengan	x >= y	Apakah x lebih dari sama dengan y
==	Sama dengan	x == y	Apakah x sama dengan y
!=	Tidak sama dengan	x != y	Apakah x tidak sama dengan y

Program 12.1 berikut ini merupakan contoh penerapan operator perbandingan untuk menampilkan bilangan terbesar dari dua buah bilangan bulan yang diinput.

Program 12.1 Contoh Penggunaan Operator Perbandingan

```
#include<stdio.h>
void main()
{
    int A,B;
    scanf("%i",&A);
    scanf("%i",&B);
    if(A>B)
        printf("%i",A);
    else
        printf("%i",B);
}
```

11.2. TEKNIK PENCARIAN NILAI TERBESAR/TERKECIL PADA ARRAY 1 DIMENSI

Pencarian nilai terbesar dan/atau terkecil dari sejumlah nilai yang tersimpan pada array 1 dimensi merupakan contoh kasus yang paling umum dalam menjelaskan konsep pencarian. Untuk melakukan pencarian nilai pada sebuah array, tentunya harus sudah menguasai 2 (dua) algoritma dasar, yaitu penelusuran array dan perbandingan nilai. Pada dasarnya, penelusuran array dapat dilakukan dengan perulangan baik menggunakan struktur for(), while(), maupun do..while(). Pada beberapa pemrograman, juga mengenal perulangan khusus untuk penelusuran array, yaitu foreach(), seperti ditemukan pada bahasa pemrograman PHP.

CONTOH SOAL 12.1 – NILAI TERBESAR

Sudah ada array satu dimensi yang dibuat dengan int A[11]. Sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	7	25	16	19

Susun program untuk mencari dan mencetak isi array yang nilainya terbesar. Untuk contoh data di atas, bila program dijalankan maka akan tercetak: **25**.

Pembahasan Soal dan Jawaban

Untuk menyelesaikan Soal 12.1 di atas, pertama-tama harus dilakukan penelusuran terhadap seluruh isi array. Penelusuran dapat dilakukan dengan memanfaatkan struktur perulangan. Nilai terbesar tidak diketahui posisinya, sehingga penelusuran dapat dilakukan dari sebelah kiri, maupun sebelah kanan. Selanjutnya untuk menentukan bilangan terbesar, dapat digunakan konsep perbandingan bilangan seperti sudah dijelaskan di atas. Berikut ini algoritma atau kerangka pikir untuk menyelesaikan Soal 12.1 di atas:

1. Inisialisasi array dan isinya.
2. Inisialisasi variabel max untuk menyimpan nilai terbesar.
3. Elemen pertama dari array anggap sebagai nilai terbesar.
4. Lakukan penelusuran array dari elemen kedua hingga terakhir.
5. Bandingkan setiap elemen array dengan isi variabel max. Jika lebih besar dari max, maka ganti nilai variabel max dengan nilai tersebut.
6. Selesai penelusuran, cetak isi variabel max sebagai variabel terbesar.

Berdasarkan algoritma (kerangka pikir) di atas, selanjutnya dapat diimplementasikan ke dalam bentuk program seperti pada Program 12.2 berikut ini.

Program 12.2 Mencari dan Mencetak Nilai Terbesar dari Array Satu Dimensi

```c
#include <stdio.h>
int main()
{
    int A[11]={12,17,10,5,15,25,11,7,25,16,19};
    int I, max;

    //cetak isi array
    for (I=0; I<11; I++) {
        printf("%3i", A[I]);
    }

    I = 1;
    max = A[0];
    while(I<=10) {
        if(A[I] > max) {
            max = A[I];
        }
        I++;
    }
    printf("\nNilai terbesar : %i", max);
    return 0;
}
```

11.3. TEKNIK PENCARIAN NILAI TERBESAR/TERKECIL DENGAN ALGORITMA SEKUENSIAL

Teknik pencarian nilai pada array dengan algoritma sekuensial sebenarnya sudah dijelaskan pada bagian 11.2 di atas. Teknik sekuensial pada dasarnya melakukan penelusuran array secara berurut baik dari sebelah kiri (indeks terkecil) maupun dari sebelah kanan (indeks terbesar). Sebagai teknik dasar, kelebihan teknik ini tentunya dari sisi kemudahan dalam memahaminya. Sementara kekurangannya adalah tidak efektif secara komputasi, terutama pada data yang sangat banyak dan untuk posisi nilai berada di bagian akhir dari array.

Perhatikan kembali contoh Soal 12.1. Kita dapat menyelesaikan soal tersebut dengan cara yang sedikit berbeda, namun tetap menggunakan teknik penelusuran sekuensial. Perhatikan Program 12.3 berikut ini.

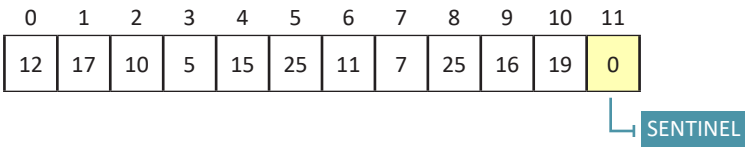
Program 12.3 Teknik Pencarian Nilai Terbesar dengan Algoritma Sekuensial

```c
#include <stdio.h>
int main()
{
    int A[11]={12,17,10,5,15,25,11,7,25,16,19};
    int I, J;

    //cetak isi array
    for (I=0; I<11; I++) {
        printf("%3i", A[I]);
    }

    I = 0;
    J = 0;
    while(I<=10) {
        if(A[I] > A[J]) {
            J = I;
        }
        I++;
    }
    printf("\nNilai terbesar : %i", A[J]);
    return 0;
}
```

Pada Program 12.3 di atas, tidak menggunakan variabel tertentu yang menyimpan nilai terbesar, namun yang disimpan adalah nomor indeks (variabel J) yang mengandung nilai terbesar. Secara penelusuran, Program 12.3 melakukan penelusuran dari awal array. Menurut Anda, antara Program 12.2 dan Program 12.3, mana yang lebih mudah dipahami?

11.4. TEKNIK PENCARIAN NILAI TERBESAR/TERKECIL DENGAN ALGORITMA SENTINEL

Pencarian dengan teknik SENTINEL merupakan pengembangan dari algoritma pencarian sekuensial. Teknik ini tetap membutuhkan penelusuran array secara berurutan (sekuensial), baik dari depan maupun dari belakang. Perbedaannya adalah, pada teknik sentinel, ditambahkan satu elemen di akhir array untuk menyimpan nilai yang dicari. Berdasarkan soal nomor 12.1, posisi elemen sentinel dapat diilustrasikan pada gambar berikut ini. Elemen array indeks ke-11 merupakan elemen sentinel yang akan menyimpan nilai yang dicari.



Program 12.4 berikut ini merupakan contoh program untuk mencari nilai TERKECIL dengan teknik sentinel.

Program 12.4 Teknik Pencarian pada Array dengan Teknik Sentinel

```
1  #include <stdio.h>
2  int main()
3  {
4      int A[12]={12,17,10,5,15,25,11,7,25,16,19};
5      int I;
6
7      //cetak isi array
8      for (I=0; I<11; I++) {
9          printf("%3i", A[I]);
10     }
11
12     I = 1;
13     A[11] = A[0];
14     while(I<=10) {
15         if(A[I] < A[11]) {
16             A[11] = A[I];
17         }
18         I++;
19     }
20     printf("\nNilai terkecil : %i", A[11]);
21     return 0;
22 }
```

Perhatikan deklarasi array pada baris ke-4 Program 12.4 di atas. Pada bagian deklarasi, ditambahkan satu elemen sentinel di bagian akhir

array. Pada awal penelusuran, elemen sentinel diisi dengan isi dari elemen pertama array (A[0]) sebagai nilai terkecil. Selanjutnya dilakukan penelusuran sekuensial dan perbandingan setiap eleman dengan nilai yang tersimpan pada elemen sentinel. Pada akhir perulangan, nilai TERKECIL akan tersimpan pada elemen sentinel yaitu A[11].

KESIMPULAN

Penelusuran array merupakan salah satu proses dasar di dalam algoritma dan pemrograman. Penelusuran array diperlukan di banyak algoritma yang melibatkan data dalam jumlah banyak. Beberapa contoh algoritma yang memerlukan konsep penelusuran array antara lain algoritma pencarian (*searching*), pengurutan (*sorting*), penggabungan (*merging*) dan pembagian/pemecahan (*splitting*/*dividing*). Oleh karena itu, memahami konsep penelusuran array merupakan salah satu kompetensi yang wajib dimiliki oleh seorang *programmer*.

SOAL LATIHAN

Soal 1.

Sudah ada array satu dimensi yang dibuat dengan int A[11]. Sudah ada isinya dengan ilustrasi sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10
12	17	10	5	15	25	11	7	25	16	19

Susun program untuk:

a. Mencari dan mencetak isi array yang nilainya TERKECIL. Untuk contoh data di atas, bila program dijalankan maka akan tercetak: **5**.

b. Mencetak ada berapa nilai TERKECIL dalam array tersebut. Untuk contoh data di atas, bila program dijalankan maka akan tercetak: **1**.

c. Mencetak berada di posisi (*index*) berapa, nilai TERKECIL dalam array tersebut. Untuk contoh data di atas, bila program dijalankan maka akan tercetak: **3**.

Soal 2.

Sudah ada array satu dimensi yang dibuat dengan int A[10], belum ada isinya. Susun program untuk menginput nilai 10 mahasiswa ke dalam array tersebut. Selanjutnya cetak keseluruhan isi array, cetak nilai TERBESAR, dan cetak JUMLAH mahasiswa yang mendapat nilai terbesar tersebut!

Soal 3.

Sudah ada array satu dimensi yang dibuat dengan int A[10], belum ada isinya. Susun program untuk menginput nilai 10 mahasiswa ke dalam array tersebut. Selanjutnya hitung dan cetak rata-rata nilai mahasiswa, serta cetak JUMLAH mahasiswa yang mendapat nilai di bawah rata-rata!

Soal 4 (*credit to HackerRank.com*)

Given five positive integers, find the minimum and maximum values that can be calculated by summing exactly four of the five integers. Then print the respective minimum and maximum values as a single line of two space-separated long integers.

For example, $arr = [1, 3, 5, 7, 9]$. Our minimum sum is $1 + 3 + 5 + 7 = 16$ and our maximum sum is $3 + 5 + 7 + 9 = 24$. We would print

```
16 24
```

Function Description

Complete the miniMaxSum function in the editor below. It should print two space-separated integers on one line: the minimum sum and the maximum sum of 4 of 5 elements.

miniMaxSum has the following parameter(s):

- arr: an array of 5 integers

Input Format

A single line of five space-separated integers.

Constraints

$1 \leq arr[i] \leq 10^9$

Output Format

Print two space-separated long integers denoting the respective minimum and maximum values that can be calculated by summing exactly four of the five integers. (The output can be greater than a 32 bit integer.)

Sample Input

```
1 2 3 4 5
```

Sample Output

```
10 14
```

Explanation

Our initial numbers are 1, 2, 3, 4, and 5. We can calculate the following sums using four of the five integers:

1. If we sum everything except 1, our sum is $2 + 3 + 4 + 5 = 14$.
2. If we sum everything except 2, our sum is $1 + 3 + 4 + 5 = 13$.
3. If we sum everything except 3, our sum is $1 + 2 + 4 + 5 = 12$.
4. If we sum everything except 4, our sum is $1 + 2 + 3 + 5 = 11$.
5. If we sum everything except 5, our sum is $1 + 2 + 3 + 4 = 10$.

Hints: Beware of integer overflow! Use 64-bit Integer.

BAB 12

PENGGABUNGAN (*MERGE*) ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa dapat memahami konsep manipulasi array satu dimensi, dan melakukan penggabungan dua atau lebih array menjadi array satu dimensi.
Subpokok Bahasan	:	12.1. Konsep dasar penggabungan (merge) array 12.2. Penggabungan dua buah array menjadi satu array 12.3. Penggabungan dua buah array menjadi satu array dengan kriteria tertentu
Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

PENGGABUNGAN (*MERGE*) ARRAY SATU DIMENSI

12.1. KONSEP PENGGABUNGAN (*MERGE*) ARRAY

Merge artinya menggabungkan. Dalam pemrograman, *merge* biasanya dimaksudkan menggabungkan dua *file* data dalam *external storage*. Yang akan dipelajari di modul ini adalah menggabungkan data dalam dua buah array satu dimensi. Logika menggabungkan dua buah array dapat juga diaplikasikan untuk menggabungkan dua buah *file* (Sjukani, 2014).

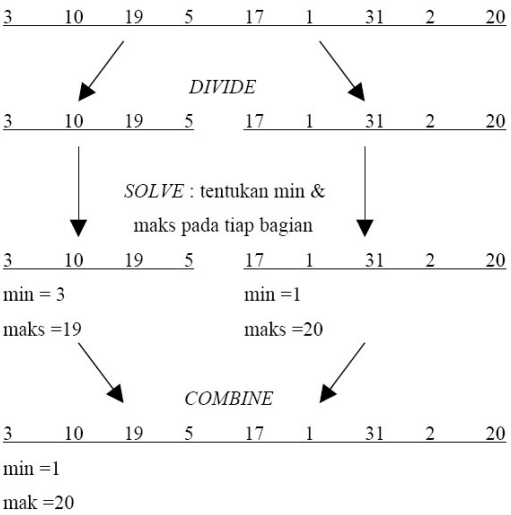
Konsep penggabungan (*merge*) array penting untuk dipelajari, karena menjadi dasar bagi berbagai algoritma untuk menyelesaikan berbagai permasalahan. Beberapa penggunaan konsep penggabungan (*merge*) dan pemecahan (*divide*) antara lain:

1. Algoritma *Divide-and-Conquer*.
   Algoritma *Divide and Conquer* merupakan algoritma yang sangat populer di dunia Ilmu Komputer. *Divide and Conquer* merupakan algoritma yang berprinsip memecah-mecah permasalahan yang terlalu besar menjadi beberapa bagian kecil sehingga lebih mudah untuk diselesaikan. Langkah-langkah umum algoritma *Divide and Conquer*: (1) ***Divide***: Membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama); (2) ***Conquer***: memecahkan (menyelesaikan) masing-masing masalah (secara rekursif); dan (3) ***Combine***: menggabungkan solusi masing-masing masalah sehingga membentuk solusi masalah semula.

2. Algoritma Pengurutan *Merge-Sort*.
   Merge-sort merupakan algoritma pengurutan dalam ilmu komputer yang dirancang untuk memenuhi kebutuhan pengurutan atas suatu rangkaian data yang tidak memungkinkan untuk ditampung dalam memori komputer karena jumlahnya yang terlalu besar. Algoritma ini ditemukan oleh John von Neumann pada tahun 1945.
   Algoritma pengurutan data *merge-sort* dilakukan dengan menggunakan cara *divide-and-conquer* yaitu dengan memecah kemudian menyelesaikan setiap bagian kemudian menggabungkannya kembali. Pertama data dipecah menjadi 2

bagian di mana bagian pertama merupakan setengah (jika data genap) atau setengah minus satu (jika data ganjil) dari seluruh data, kemudian dilakukan pemecahan kembali untuk masing-masing blok sampai hanya terdiri dari satu data tiap blok.

Gambar 13.1 menyajikan contoh penerapan konsep pemecahan dan penggabungan array untuk mencari nilai terbesar dan terkecil dari suatu array.



Gambar 13.1 Contoh Ilustrasi Penerapan Algoritma *Divide-and-Conquer* untuk Mencari Nilai Terkecil dan Terbesar

12.2. PENGGABUNGAN DUA BUAH ARRAY MENJADI SATU ARRAY

Pada bagian ini akan dijelaskan teknik penggabungan dua buah array menjadi satu array. Untuk mempermudah penjelasan, perhatikan contoh Soal 13.1 berikut ini.

CONTOH SOAL 13.1 – PENGGABUNGAN ARRAY

Sudah ada array **A** satu dimensi yang dibuat dengan **int A[5].** Dan array **B** yang dibuat dengan **int B[7].** Kedua buah array sudah ada isinya dengan ilustrasi sebagai berikut:

	0	1	2	3	4
A	12	17	10	5	15

	0	1	2	3	4	5	6
B	25	11	7	25	16	22	14

Sudah ada array C satu dimensi yang dibuat dengan int **C[12]**. Belum diisi dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	10	11
C												

Susun program untuk menyalin (menggabungkan) isi array A dan array B ke array C, sehingga isi array C menjadi:

	0	1	2	3	4	5	6	7	8	9	10	11
C	12	17	10	5	15	25	11	7	25	16	22	14

Berikut ini ilustrasi proses menyalin isi array A dan isi array B ke dalam array C

	0	1	2	3	4	5	6
B	25	11	7	25	16	22	14

	0	1	2	3	4
A	12	17	10	5	15

	0	1	2	3	4	5	6	7	8	9	10	11
C	12	17	10	5	15	25	11	7	25	16	22	14

Analisis dan Penyelesaian Soal

Untuk melakukan penggabungan array A dan B, pada dasarnya tinggal melakukan perulangan untuk setiap elemen array A[] dan B[] dan memindahkannya ke array C[].

Berikut ini algoritma atau kerangka pikir untuk menyelesaikan Soal 13.1 di atas:

1. Inisialisasi array A[] dan isinya.
2. Inisialisasi array B[] dan isinya.
3. Inisialisasi array C[] tanpa isi (kosong).
4. Lakukan penelusuran isi array A[] lalu masukkan ke array C[] secara berurutan.
5. Lakukan penelusuran isi array B[] lalu masukkan ke array C[] secara berurutan (indeks melanjutkan dari langkah 4).
6. Cetak isi array C.

Berdasarkan algoritma (kerangka pikir) di atas, selanjutnya dapat diimplementasikan ke dalam bentuk program seperti pada Program 13.1 berikut ini.

Program 13.1 Penggabungan Dua Array Menjadi Satu Array

```
1  #include <stdio.h>
2  int main()
3  {
4          int A[5] = {12,17,10,5,15};
5          int B[7] = {25,11,7,25,16,22,14};
6          int C[12] = {0};
7          int I, J;
8
9          //cetak isi array A
10         printf("Isi Array A : ");
11         for (I=0; I<5; I++) {
12                 printf("%3i", A[I]);
13         }
14         //cetak isi array B
15         printf("\nIsi Array B : ");
16         for (I=0; I<7; I++) {
17                 printf("%3i", B[I]);
18         }
19
```

```
20          J = 0; //index ini untuk array C
21          for(I=0; I<5; I++) {
22                  C[J] = A[I];
23                  J++;
24          }
25          for(I=0; I<7; I++) {
26                  C[J] = B[I];
27                  J++;
28          }
29
30          //cetak isi array C
31          printf("\nIsi Array C : ");
32          for (I=0; I<12; I++) {
33                  printf("%3i", C[I]);
34          }
35          return 0;
36  }
```

Penjelasan Program 13.1:

- Pada baris 4-7 dilakukan deklarasi array A[] dan B[] lengkap dengan isinya, array C[] masih kosong, dan beberapa variabel yang digunakan dalam program.
- Pada baris 9-18, program melakukan pencetakan isi dari array A[] dan B[] agar saat dijalankan kita dapat mengetahui isi array aslinya.
- Baris 20-28 merupakan proses utama program yang melakukan penelusuran ke setiap elemen array A[] dan B[], serta memindahkannya secara berurutan ke array C[]. Variabel yang digunakan untuk mengatur indeks dari array C[] adalah J.
- Bagian akhir program, yaitu baris 32-34, dilakukan pencetakan isi array C[] untuk memastikan bahwa seluruh isi array A[] dan B[] telah berhasil dipindahkan (digabungkan) dengan benar.

12.3. PENGGABUNGAN DUA BUAH ARRAY MENJADI SATU ARRAY DENGAN KRITERIA TERTENTU

Contoh Soal 13.1 di atas merupakan contoh penggabungan 2 buah array satu dimensi, yang mana seluruh data diikutsertakan dalam penggabungan. Pada kasus lainnya, terkadang diminta untuk menggabungkan 2 buah array, namun dengan kriteria penggabungan tertentu. Contoh Soal 13.2 merupakan contoh soal penggabungan array dengan kriteria tertentu.

CONTOH SOAL 13.2 – PENGGABUNGAN ARRAY DENGAN KRITERIA

Sudah ada array **A** satu dimensi yang dibuat dengan **int A[5].** Dan array **B** yang dibuat dengan **int B[7].** Kedua buah array sudah ada isinya dengan ilustrasi sebagai berikut:

	0	1	2	3	4
A	12	17	10	5	15

	0	1	2	3	4	5	6
B	25	11	7	25	16	22	14

Sudah ada array C satu dimensi yang dibuat dengan int **C[12]**. Belum diisi dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	10	11
C												

Susun program untuk menyalin (menggabungkan) isi array A yang bernilai ganjil, dan isi array B yang lebih besar dari 15, ke array C, sehingga isi array C menjadi:

	0	1	2	3	4	5	6	7	8	9	10	11
C	17	5	15	25	25	16	22					

Analisis dan Penyelesaian Soal

Untuk menyelesaikan Soal 13.2 pada dasarnya sama seperti menyelesaikan Soal 13.1. Keduanya melibatkan penelusuran array menggunakan

perulangan. Namun demikian, pada Soal 13.2 perlu ditambahkan kondisi untuk menyeleksi nilai A[] dan B[] yang akan dimasukkan ke array C[].

Berikut ini algoritma atau kerangka pikir untuk menyelesaikan Soal 13.2 di atas:

1. Inisialisasi array A[] dan isinya.
2. Inisialisasi array B[] dan isinya.
3. Inisialisasi array C[] tanpa isi (kosong).
4. Lakukan penelusuran isi array A[], jika nilai merupakan bilangan ganjil maka masukkan ke array C[] secara berurutan.
5. Lakukan penelusuran isi array B[], jika nilai merupakan bilangan yang lebih besar dari 15, masukkan ke array C[] secara berurutan (indeks melanjutkan dari langkah 4).
6. Cetak isi array C.

Berdasarkan algoritma (kerangka pikir) di atas, selanjutnya dapat diimplementasikan ke dalam bentuk program seperti pada Program 13.2 berikut ini.

Program 13.2 Penggabungan Dua Array Menjadi Satu Array dengan Kriteria Tertentu

```
1  #include <stdio.h>
2  int main()
3  {
4          int A[5] = {12,17,10,5,15};
5          int B[7] = {25,11,7,25,16,22,14};
6          int C[12] = {0};
7          int I, J;
8
9          //cetak isi array A
10         printf("Isi Array A : ");
11         for (I=0; I<5; I++) {
12                 printf("%3i", A[I]);
13         }
14         //cetak isi array B
15         printf("\nIsi Array B : ");
```

```
16          for (I=0; I<7; I++) {
17                  printf("%3i", B[I]);
18          }
19          J = 0; //index ini untuk array C
20          for(I=0; I<5; I++) {
21                  if (A[I]%2==1) {
22                          C[J] = A[I];
23                          J++;
24                  }
25          }
26          for(I=0; I<7; I++) {
27                  if (B[I] > 15) {
28                          C[J] = B[I];
29                          J++;
30                  }
31          }
32
33          //cetak isi array C
34          printf("\nIsi Array C : ");
35          for (I=0; I<12; I++) {
36                  printf("%3i", C[I]);
37          }
38          return 0;
39  }
```

Penjelasan Program 13.2:

Program 13.2 di atas pada dasarnya sama seperti Program 13.1. Perbedaannya terletak pada penambahan struktur kondisi pada baris 21-24 yang hanya memasukkan isi array A[] yang bernilai ganjil, serta kondisi pada baris 27-30 yang hanya memasukkan isi array B[] yang bernilai > 15. Perhatikan juga variabel J sebagai pengatur indeks array C[]. Variabel J hanya bertambah jika kriteria yang diminta terpenuhi, atau dengan kata lain, pertambahan nilai variabel J harus diletakkan di dalam kondisi IF.

KESIMPULAN

Konsep penggabungan (*merge*) array penting untuk dipelajari, karena menjadi dasar bagi berbagai algoritma untuk menyelesaikan berbagai permasalahan. Beberapa penggunaan konsep penggabungan (*merge*) dan

pemecahan (*divide*) antara lain: algoritma *divide-and-conquer*, algoritma pengurutan *merge-sort*, konversi bilangan desimal ke biner dengan teknik pembagian 8, dan lain-lain.

SOAL LATIHAN

Soal 1.

Sudah ada array X satu dimensi yang dibuat dengan int **X[4]** dan array Y yang dibuat dengan int **Y[6]**. Kedua buah array tersebut sudah ada isinya dengan ilustrasi sebagai berikut:

	0	1	2	3
X[4]	12	2	7	10

	0	1	2	3	4	5
Y[6]	15	4	16	20	25	30

Sudah ada array Z satu dimensi yang dibuat dengan int **Z[10]** belum ada isinya. Susun program bahasa C untuk menggabung nilai yang lebih kecil dari 10 untuk isi array **X** dan nilai yang ada dilokasi ganjil untuk isi array Y, sehingga isi array Z menjadi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9
Z [10]	2	7	4	20	30					

Berasal dari array X | **Berasal dari array Y**

Soal 2.

Pak Budi merupakan dosen di Universitas Budi Luhur yang mengajar 3 matakuliah yaitu Algoritma, Pemrograman Web, dan Metodologi Riset. Asumsikan bahwa jumlah mahasiswa di ketiga matakuliah tersebut adalah 20 mahasiswa dan asumsikan bahwa di ketiga matakuliah tersebut merupakan mahasiswa yang sama. Buatlah program untuk menginput

nilai ujian ketiga matakuliah tersebut untuk seluruh mahasiswa secara bergantian, lalu simpan dalam 3 array berbeda. Identitas mahasiswa direpresentasikan dalam bentuk nomor indeks dari array.
Selanjutnya, Pak Budi ingin memilih mahasiswa-mahasiswa berprestasi untuk diikutsertakan dalam lomba PkM, dengan kriteria:

1. Rata-rata nilai dari ketiga matakuliah lebih besar dari 80.
2. Tidak ada matakuliah yang mendapat nilai < 60.

Bantulah Pak Budi dengan membuat program untuk menyimpan nomor indeks mahasiswa berprestasi ke dalam suatu array satu dimensi sesuai dengan kriteria di atas. Cetaklah daftar mahasiswa berprestasi tersebut di layar (nomor indeksnya).

Ilustrasi isi array:

AL[] : 80 60 90 85 90 90 50 60 85 75 50 60 90 85 90 90 50 60 85 75

PW[] : 90 90 75 40 60 90 50 60 85 85 100 90 50 60 85 75 80 60 90 85

MR[] : 75 40 60 90 85 90 80 60 90 85 100 90 60 85 75 50 60 85 90 50

Isi array mahasiswa berprestasi dari ilustrasi di atas:

PRESTASI[] : 0 5 8 9 14 18

BAB 13

PEMECAHANAN (*SPLIT*) ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa dapat memahami konsep manipulasi array satu dimensi, dan melakukan penggabungan dua atau lebih array menjadi array satu dimensi.

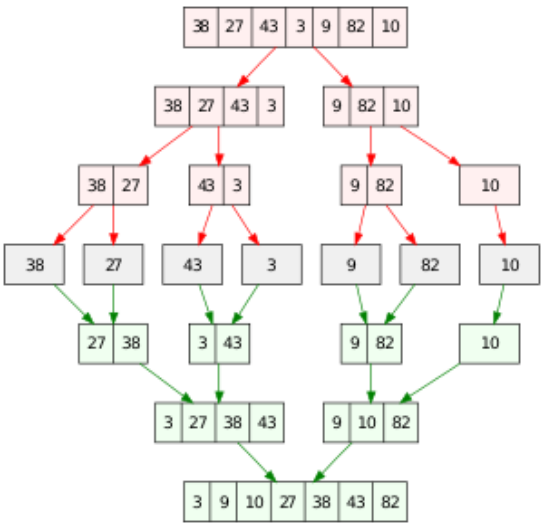
Subpokok Bahasan	:	13.1. Konsep dasar pemecahan (split) array 13.2. Pemecahan satu buah array menjadi dua buah array 13.3. Aplikasi pemecahan (split) dan penggabungan (merge) pada array

Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

PEMECAHAN (*SPLIT*) ARRAY SATU DIMENSI

13.1. KONSEP PEMECAHAN (*SPLIT*) ARRAY

***Split* array** merupakan operasi memecah suatu array menjadi beberapa array lain, baik dalam dimensi yang sama maupun berbeda. Pemecahan (*split*) array terkadang disebut juga dengan istilah pembagian (*divide*) array. Dalam pemrograman, operasi pemecahan array sering digunakan untuk menangani manipulasi data pada array yang berukuran besar. Misalnya, untuk melakukan pengurutan (*sorting*) array dalam jumlah besar, dapat dilakukan dengan memecah array menjadi beberapa array yang berukuran yang lebih kecil, lalu melakukan pengurutan pada setiap array, yang selanjutnya digabungkan kembali sebagai array utuh yang terurut. Gambar 14.1 menyajikan ilustrasi algoritma pengurutan *merge-sort* yang memanfaatkan operasi pemecahan (*split*) dan penggabungan (*merge*) array.



Gambar 14.1 Ilustrasi Algoritma *Merge-Sort* yang Memanfaatkan Operasi Pemecahan (*Split*) dan Penggabungan (*Merge*) Array
(Sumber: https://en.wikipedia.org/wiki/Merge_sort)

13.2. PEMECAHAN ARRAY SATU DIMENSI MENJADI DUA ARRAY

Pada bagian ini akan dijelaskan teknik pemecahan suatu array menjadi dua buah array berbeda. Untuk mempermudah penjelasan, perhatikan contoh soal 14.1 berikut ini.

CONTOH SOAL 14.1 – PEMECAHAN ARRAY

Sudah ada array **A** satu dimensi yang dibuat dengan **int A[12],** sudah ada isinya dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	10	11
A	12	17	10	15	25	11	7	25	16	22	14	5

Sudah ada array B[12] dan C[12] yang (dianggap) masih belum ada isinya. Susun program untuk memindahkan setiap isi array A yang bernilai **GANJIL** ke array **B** dan isi array A yang bernilai **GENAP** ke array **C**, sedemikian hingga isi array B dan C menjadi sebagai berikut:

B	17	15	25	11	7	25	5					
C	12	10	16	22	14							

Analisis dan Penyelesaian Soal

Untuk menyelesaikan Soal 14.1 di atas, pertama-tama kita harus dapat melakukan penelusuran pada array A[]. Selanjutnya menambahkan kondisi sesuai dengan permintaan soal tersebut untuk menyalin data ke array B[] dan C[]. Berikut ini algoritma atau kerangka pikir untuk menyelesaikan Soal 14.1 di atas:

1. Inisialisasi array A[] dan isinya.
2. Inisialisasi array B[] tanpa isi (kosong).
3. Inisialisasi array C[] tanpa isi (kosong).
4. Lakukan penelusuran isi array A[], jika nilainya GANJIL maka pindahkan nilainya secara berurutan ke array B[]. Namun jika nilainya GENAP maka pindahkan nilai tersebut secara berurutan ke array C[].

5. Cetak isi array B[] dan C[] untuk memastikan bahwa isi array sudah benar.

Berdasarkan algoritma (kerangka pikir) di atas, selanjutnya dapat diimplementasikan ke dalam bentuk program seperti pada Program 14.1 berikut ini.

Program 14.1 Pemecahan Satu Array Menjadi Dua Array

```
1  #include <stdio.h>
2  int main()
3  {
4          int A[12] = {12,17,10,15,25,11,7,25,16,22,14,5};
5          int B[12] = {0};
6          int C[12] = {0};
7          int I, J;
8
9          //cetak isi array A
10         printf("Isi Array A : ");
11         for (I=0; I<12; I++) {
12                 printf("%3i", A[I]);
13         }
14
15         J = 0;
16         for (I=0; I<12; I++) {
17                 if (A[I]%2==1) {
18                         B[J] = A[I];
19                         J++;
20                 }
21         }
22
23         J = 0;
24         for (I=0; I<12; I++) {
```

```
25                      if (A[I]%2==0) {
26                              C[J] = A[I];
27                              J++;
28                      }
29              }
30
31              //cetak isi array B
32              printf("\nIsi Array B : ");
33              for (I=0; I<12; I++) {
34                      printf("%3i", B[I]);
35              }
36
37              //cetak isi array C
38              printf("\nIsi Array C : ");
39              for (I=0; I<12; I++) {
40                      printf("%3i", C[I]);
41              }
42              return 0;
43      }
44
```

Penjelasan Program 14.1:

- Pada baris 4-7 dilakukan deklarasi array A[] lengkap dengan isinya, array B[] dan array C[] masih kosong, serta beberapa variabel yang digunakan dalam program.
- Pada baris 9-13, program melakukan pencetakan isi dari array A[] agar saat dijalankan kita dapat mengetahui isi array aslinya.
- Baris 15-21 merupakan proses penelusuran isi dari array A[], disertai pemeriksaan kondisi apakah nilai/bilangan bernilai GANJIL (perhatikan baris ke-17). Jika bernilai ganjil, maka pindahkan nilai tersebut ke array B[] secara berurutan. Perhatikan variabel yang digunakan untuk melakukan penelusuran A[] berbeda dengan variabel yang digunakan untuk mengisi array B[]. *Mengapa harus berbeda? Bisakah kalo dibuat satu variabel saja?*
- Baris 23-29 merupakan proses penelusuran isi dari array A[], disertai pemeriksaan kondisi apakah nilai/bilangan bernilai GENAP

(perhatikan baris ke-25). Jika bernilai genap, maka salin nilai tersebut ke array C[] secara berurutan. Perhatikan variabel yang digunakan untuk melakukan pengisian ke array B[], yaitu J, harus diberikan nilai awal dengan J=0 (baris ke-23). *Mengapa demikian? Bagaimana jika perintah tersebut dihilangkan?*

- Bagian akhir program, yaitu baris 31-41, dilakukan pencetakan isi array B[] dan C[] untuk memastikan bahwa seluruh isi array A[]telah berhasil disalin dengan benar.

Perhatikan kembali Program 14.1 di atas! Seperti sudah dijelaskan di atas, baris ke 15-29 merupakan proses penelusuran dan penyalinan array A[] ke array B[] dan C[]. Proses penelusuran berlangsung dua kali. Bagian program tersebut sebenarnya dapat digabungkan, sehingga program hanya memerlukan satu kali penelusuran array A[] dan secara komputasi menjadi lebih efektif.

Program 14.2 berikut ini merupakan potongan program yang menggabungkan proses penyalinan isi array A[] ke array B[] dan C[]. Yang perlu diperhatikan adalah penggunaan variabel sebagai pengatur indeks array B[] dan C[] harus berbeda, karena keduanya merupakan array berbeda.

Program 14.2 Potongan Program yang Menggabungkan Proses Penelusuran Array

```
15      J = 0; K = 0;
16      for (I=0; I<12; I++) {
17          if (A[I]%2==1) {
18              B[J] = A[I];
19              J++;
20          } else {
21              C[K] = A[I];
22              K++;
23          }
24      }
```

13.3. CONTOH APLIKASI PEMECAHAN dan PENGGABUNGAN ARRAY

Aplikasi yang melibatkan proses pemecahan dan penggabungan array sebenarnya banyak sekali. Pada modul ini dicontohkan penerapan konsep pemecahan (*split*) dan penggabungan (*merge*) array pada algoritma pengurutan *merge-sort*. Algoritma pengurutan tersebut sudah diilustrasikan pada Gambar 14.1 di atas. Anda dapat mencoba Program 14.3 berikut ini. Namun demikian, untuk penjelasan rinci mengenai algoritma pengurutan *merge-sort* tidak akan dijelaskan pada modul ini, akan tetapi akan dijelaskan pada matakuliah Algoritma dan Struktur Data 1.

Program 14.3 Contoh Algoritma Pengurutan *Merge-Sort*

```
1  #include “stdio.h”
2  int A[10];
3  void merge(int,int,int);
4  void merge_sort(int low,int high)
5  {
6          int mid;
7          if(low<high) {
8                  mid=(low+high)/2;
9                  merge_sort(low,mid);
10                 merge_sort(mid+1,high);
11                 merge(low,mid,high);
12         }
13 }
14 void merge(int low,int mid,int high)
15 {
16         int h,i,j,b[10],k;
17         h=low;
18         i=low;
18         j=mid+1;
20         while((h<=mid)&&(j<=high)) {
```

```
21                      if(A[h]<=A[j]) {
23                              b[i]=A[h]; h++;
24                      } else {
25                              b[i]=A[j]; j++;
26                      }
27                      i++;
28              }
29              if(h>mid) {
30                      for(k=j;k<=high;k++) {
31                              b[i]=A[k]; i++;
32                      }
33              } else {
34                      for(k=h;k<=mid;k++) {
35                              b[i]=A[k]; i++;
36                      }
37              }
38              for(k=low;k<=high;k++)
39                      A[k]=b[k];
40      }
41      int main()
42      {
43              int num = 10,i, M;
44              printf("MERGE SORT\n");
45              //input
46              printf("Input 10 bilangan:\n");
47              for (M=0; M<10; M++) {
48                      scanf("%i", &A[M]);
49              }
50
51              printf("Sebelum pengurutan");
52              for (M=0; M<10; M++) {
53                      printf("%4i", A[M]);
54              }
55
56              merge_sort(0,num-1);
57
58              printf("\n\nSetelah pengurutan");
```

```
59          for (M=0; M<10; M++) {
60                  printf("%4i", A[M]);
61          }
62          return 0;
63  }
```

KESIMPULAN

Konsep pemecahan (*split*) array penting untuk dipelajari, karena menjadi dasar bagi berbagai algoritma untuk menyelesaikan berbagai permasalahan. Salah satu penerapan konsep pemecahan array, bersama dengan konsep penggabungan array, adalah algoritma pengurutan *merge-sort* seperti sudah diilustrasikan pada Gambar 14.1 dan diterapkan pada Program 14.3.

SOAL LATIHAN

Soal 1.

Sudah ada array **N** satu dimensi yang dibuat dengan **int N[30],** sudah ada isinya berupa nilai mahasiswa dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	...	29
N	90	55	60	75	45	91	87	75	76	62	...	35

Sudah ada array LULUS[30] dan GAGAL[30] yang (dianggap) masih belum ada isinya. Susun program untuk memindahkan setiap isi array N yang bernilai **LULUS** ke array **LULUS[]** dan isi array N yang bernilai **GAGAL** ke array **GAGAL[]**. Nilai lulus adalah nilai yang lebih besar atau sama dengan 60.

BAB 14

MANIPULASI ARRAY KARAKTER (STRING)

Capaian Pembelajaran	:	Mahasiswa dapat memahami konsep manipulasi array satu dimensi, dan melakukan penggabungan dua atau lebih array menjadi array satu dimensi.
Subpokok Bahasan	:	14.1. Konsep manipulasi array karakter (string) 14.2. Manipulasi array karakter: teknik penelusuran, penggabungan (merge) dan pemecahan (split) array satu dimensi yang bertipe karakter
Daftar Pustaka	:	Gaddis, nd. 2011. Starting Out with C++ from Control Structures through Objects. 8th. Boston: Addison-Wesley. Institue of Distance & Open Learning, n.d. UNIT I Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 Sjukani, Moh. 2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

MANIPULASI ARRAY KARAKTER (STRING)

14.1. KONSEP MANIPULASI ARRAY KARAKTER (STRING)

Karakter dalam pemrograman merupakan representasi dari satu buah huruf, angka atau karakter tertentu. Setiap pemrograman memiliki tipe data khusus untuk menyimpan karakter, misalnya tipe data **char** di Bahasa C/C++ dan Java. Sementara itu, kumpulan dari karakter sering disebut sebagai **string**. Umumnya, bahasa pemrograman tidak menyediakan tipe data khusus untuk menyimpan string (kumpulan karakter). Namun ada beberapa yang menyediakan tipe data bentukan untuk string, seperti di Java memfasilitasi tipe data string dengan membentuk class String.

Di bahasa pemrograman C/C++ sendiri, tidak memiliki tipe data khusus untuk menyimpan string. Namun demikian, penanganan string dilakukan melalui struktur data array satu dimensi. Dengan kata lain, untuk menyimpan string atau teks, kita perlu membentuk array bertipe karakter (char) sejumlah karakter dari string yang akan disimpan.

Sebagai contoh, jika kita ingin mendeklarasikan suatu variabel C yang diisi dengan string “UBL”, maka dapat dilakukan dengan salah satu dari potongan perintah berikut ini:

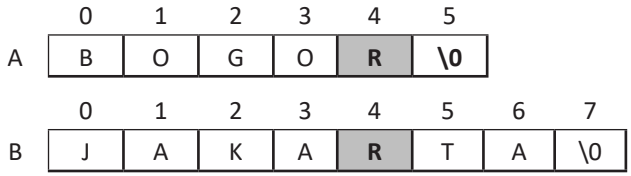
1. char C[3] = “UBL”;
2. char C[3] = {‘U’,’B’,’L’};
3. char C[3]; strcpy(C,”UBL”); **//cara ini membutuhkan header fungsi string.h**

14.2. MANIPULASI ARRAY KARAKTER (STRING)

Pada bagian ini dijelaskan mengenai beberapa contoh manipulasi array karakter (string). Manipulasi array karakter dapat berupa penelusuran array, pencarian isi array, pengurutan, penggabungan (*merge*) atau pemecahan (*split*). Contoh 15.1 di bawah ini merupakan contoh soal yang terkait dengan penelusuran dan pencarian isi array.

CONTOH SOAL 15.1 – PENELUSURAN ARRAY

Sudah ada dua buah array satu dimensi **A** dan **B** yang dibuat dengan **char A[6]** dan **char B[8],** sudah ada isinya dengan huruf-huruf kapital tanpa spasi. Ilustrasinya sebagai berikut:



Susun program untuk memeriksa apakah di antara isi array A ada yang sama dengan isi array B. Bila ada, cetak perkataan "**ADA**", bila tidak ada cetak perkataan "**TIDAK ADA**". Pada contoh di atas, maka akan tercetak perkataan "ADA" karena terdapat huruf yang sama di antara dua array, yaitu huruf R.

Analisis dan Penyelesaian Soal

Untuk menyelesaikan Soal 15.1 di atas, kita harus melakukan penelusuran setiap karakter yang tersimpan pada array A[], lalu dilakukan pencarian apakah karakter tersebut ada di array B[] atau tidak. Berikut ini algoritma atau kerangka pikir untuk menyelesaikan Soal 15.1 di atas:

1. Inisialisasi array A[] dan isinya.
2. Inisialisasi array B[] dan isinya.
3. Inisialisasi variabel flag = 0 untuk menyimpan status pencarian (0 = tidak ditemukan, 1 = ditemukan).
4. Lakukan penelusuran isi array A[], lalu lakukan penelusuran dan pemeriksaan isi dari B[]. Jika terdapat karakter dari array A[] yang sama dengan karakter dari array B[], maka ubah flag menjadi 1 dan hentikan penelusuran. Karena soal hanya meminta status "ADA" atau "TIDAK ADA", maka saat ditemukan, penelusuran tidak perlu dilanjutkan.
5. Jika isi flag = 0 maka cetak "TIDAK ADA", namun jika flag = 1 maka cetak "ADA".

Berdasarkan algoritma (kerangka pikir) di atas, selanjutnya dapat diimplementasikan ke dalam bentuk program seperti pada Program 15.1 berikut ini.

Program 15.1 Penelusuran dan Pencarian Karakter pada Array Satu Dimensi

```
1   #include <stdio.h>
2   int main()
3   {
4               char A[6] = “BOGOR”;
5               char B[8] = “JAKARTA”;
6               int I, J, flag=0;
7
8               //cetak isi array A
9               printf(“Isi Array A : “);
10              for (I=0; I<5; I++) {
11                      printf(“%3c”, A[I]);
12              }
13              //cetak isi array B
14              printf(“\nIsi Array B : “);
15              for (I=0; I<7; I++) {
16                      printf(“%3c”, B[I]);
17              }
18
19              for (J=0; J<5; J++) {
20                      for (I=0; I<7; I++) {
21                              if (A[J] == B[I]) {
22                                      flag = 1; break;
23                              }
24                      }
25              }
26
27              if(flag==1) {
28                      printf(“\nADA”);
29              } else {
30                      printf(“\nTIDAK ADA”);
31              }
32
33              return 0;
34  }
```

Penjelasan Program 15.1:

- Pada baris 4-6 dilakukan deklarasi array A[] dan B[] lengkap dengan isinya, serta beberapa variabel yang digunakan dalam program.
- Pada baris 8-17, program melakukan pencetakan isi dari array A[] dan B[] agar saat dijalankan kita dapat mengetahui isi array aslinya.
- Baris 19-25 merupakan proses penelusuran isi dari array A[]. Untuk setiap isi dari array A[], lakukan penelusuran ke array B[]. Jika terdapat karakter/isi dari array A[] yang sama dengan isi dari array B[] maka, ubah nilai variabel **flag** menjadi 1 (artinya ditemukan karakter yang sama). Dan selanjutnya keluar dari perulangan/penelusuran dengan perintah **break**. *Mengapa harus ditambahkan perintah break? Apakah jika tidak menggunakan perintah break, program tetap benar?*
- Bagian akhir program, yaitu baris 27-31, dilakukan pencetakan perkataan "ADA" jika isi flag = 1, dan "TIDAK ADA" jika isi flag = 0.

Selain penelusuran dan pencarian karakter di dalam array satu dimensi, proses pemecahan atau pemisahan string juga sering diperlukan dalam berbagai kasus. Salah satunya adalah proses *tokenizing*. Proses *tokenizing* adalah proses pemotongan string masukan berdasarkan tiap kata yang menyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata yang menyusun suatu dokumen. *Tokenizing* umumnya dipakai dalam berbagai algoritma *information retrieval*, *text mining*, dan sebagainya. Perhatikan contoh Soal 15.2 berikut ini!

CONTOH SOAL 15.2 – *TOKENIZING*

Sudah ada sebuah array satu dimensi **A** yang dibuat dengan **char A[1000]** belum ada isinya. Susun program untuk menginput satu kalimat dan menyimpannya di array A[]. Selanjutnya cetak setiap kata dari kalimat yang diinputkan pada baris-baris yang berbeda.

Contoh inputan: `algoritma itu mudah`

Luaran:

```
algoritma
itu
mudah
```

Analisis dan Penyelesaian Soal

Untuk menyelesaikan Soal 15.2, pertama-tama kita simpan kalimat yang diinputkan ke array A[]. Selanjutnya lakukan penelusuran setiap karakter yang tersimpan pada array A[], lakukan pencetakan dan jika menemukan karakter spasi, ganti dengan karakter pindah baris (\n). Karakter spasi merupakan karakter nomor 32. Berikut ini algoritma atau kerangka pikir untuk menyelesaikan Soal 15.2 di atas:

Program 15.2 *Tokenizing*.

```
1  #include “stdio.h”
2  int main()
3  {
4          char A[1000];
5          int i;
6          //input
7          printf(“Input kalimat: “);
8          gets(A);
9          i=0;
10         while(true) {
11                 if (A[i] == ‘\0’) {
12                         break;
13                 }
14                 if (A[i] != ‘ ‘) {
15                         printf(“%c”, A[i]);
16                 } else {
17                         printf(“\n”);
18                 }
19                 i++;
20         }
21         return 0;
22 }
```

KESIMPULAN

Penanganan atau manipulasi array karakter (string) merupakan salah satu kemampuan mendasar yang harus dikuasai oleh seorang *programmer*. Pada dasarnya penanganan array karakter (string) tidaklah berbeda dengan penanganan array yang berisi data lainnya. Hanya terdapat beberapa perbedaan dalam menangani data karakter/string seperti dalam hal pengisian variabel.

SOAL LATIHAN

Soal 1.

Sudah ada sebuah array satu dimensi A yang dibuat dengan char A[12]. Array tersebut sudah ada isinya berupa huruf-huruf kapital tanpa spasi. Susun program untuk mencetak jumlah huruf-huruf yang menyusun kata dalam array tersebut!

Contoh isi array: `{JAKARTABARAT}`

maka akan tercetak:

J 1
A 5
K 1
R 2
T 2
B 1

Soal 2.

Susun program untuk menginput sebuah teks (maksimal 2000 karakter) dan simpan ke dalam sebuah variabel array. Selanjutnya lakukan pemecahan setiap kata dari teks tersebut dan cetak dalam baris yang berbeda untuk setiap kata. Hilangkan setiap tanda baca.

Contoh inputan: `Saya cinta kampus Budi Luhur. UBL kampus cerdas berbudi luhur.`

maka akan tercetak:

```
Saya
cinta
kampus
Budi
Luhur
UBL
kampus
cerdas
berbudi
luhur
```

Algoritma adalah langkah-langkah yang diambil dalam menyelesaikan suatu pekerjaan. Suatu pekerjaan dapat diselesaikan dalam satu langkah, dua langkah, atau banyak langkah. Langkah-langkah harus tersusun secara logis agar pekerjaan dapat diselesaikan dengan benar. Di dunia komputer dan pemrograman, manusia bertugas untuk memberikan perintah kepada komputer, langkah per langkah yang akan dilaksanakan oleh komputer untuk menyelesaikan pekerjaan tersebut.

Buku berjudul ***Analisis dan Desain Algoritma*** ini membahas tentang algoritma dalam 14 bab. Buku ini diharapkan dapat menjadi referensi dan membuka wawasan pembaca terkait algoritma.

Penerbit Deepublish (CV BUDI UTAMA)
Jl. Kaliurang Km 9,3 Yogyakarta 55581
Telp/Fax : (0274) 4533427
Anggota IKAPI (076/DIY/2012)
cs@deepublish.co.id
Penerbit Deepublish
@penerbitbuku_deepublish
www.penerbitdeepublish.com

Kategori :

ISBN 978-634200004-5