

ISBN:



BUKU AJAR MATA KULIAH

ALGORITMA DAN PEMROGRAMAN

PENYUSUN: Drs. Wahyu Pujiyono, M.Kom.

HAK CIPTA

BUKU AJAR MATA KULIAH ALGORITMA PEMROGRAMAN

Copyright© 2020,

Drs. Wahyu Pujiyono, M.Kom.

Hak Cipta dilindungi Undang-Undang

Dilarang mengutip, memperbanyak atau mengedarkan isi buku ini, baik sebagian maupun seluruhnya, dalam bentuk apapun, tanpa izin tertulis dari pemilik hak cipta dan penerbit.

Diterbitkan oleh:

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Penulis : Drs. Wahyu Pujiyono, M.Kom.

Editor : Laboratorium Teknik Informatika, Universitas Ahmad Dahlan

Desain sampul : Laboratorium Teknik Informatika, Universitas Ahmad Dahlan

Tata letak : Laboratorium Teknik Informatika, Universitas Ahmad Dahlan

Ukuran/Halaman : 21 x 29,7 cm / 227 halaman

Didistribusikan oleh:



Laboratorium Teknik Informatika

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Indonesia

DAFTAR ISI

Kata Pengantar	i
Daftar Isi	iii
Bab 1 : Komputer dan Bahasa Pemrograman	1
1.1 Komputer dan Bahasa Pemrograman	1
1.2 Bahasa Pemrograman Terstruktur	2
1.2 Struktur Program Bahasa C++	3
1.4 Variable	3
1.5 Jenis / Tipe Data	4
1.6 Pernyataan	4
1.7 Program : dari Konsep sampai Eksekusi	4
1.8 Mengenal Pemrograman berorientasi Objek	8
1.9 Pengantar Algoritma	10
1.10 Flowchart	11
Proyek 1 : menggambar persegi panjang	13
Latihan	14
Workshop Pemrograman	16
Bab 2 : INPUT dan OUTPUT Operator Overloading dan FILE	21
2.1 Pendahuluan	21
2.2 Memberi masukan : tradisional dan class	21
2.3 Workshop Pemrograman : Operator Overloading	24
2.4 File	28
2.5 Hirarki Data	28
2.6 File dan Stream	29
2.7 Kepustakaan Standar untuk File	30
2.8 Membuat File Akses Sekuensial	30
2.9 Membaca Data dari File Akses Sekuensial	33
2.10 File Akses Random (FAR)	35
2.11 Membuat File Akses Random	36

2.12 Menulis Data Secara Random pada FAR	37
2.13 Membaca Data Secara Random pada FAR	39
2.14 Rangkuman Fungsi Untuk Operasi File (C)	40
Bab 3 : Sekuen	41
3.1 Pendahuluan	41
3.2 Operator Aritmetika	41
3.3 Pemrograman	42
3.4 Mengkonversi ke Model OOP	43
3.5. Workshop algoritma dan class	59
Bab 4 : Pemilihan/ Kontrol Program	67
4.1 Pendahuluan	67
4.2 Ekspresi Relasional	67
4.3 Operator Relasional	71
4.4 Operator Logika	71
Latihan	84
Uji Pemrograman	88
Bab 5 : Perulangan/ Loop	91
5.1 Pendahuluan	91
5.2 Konsep Counter	91
5.3 Konsep Total	92
5.4 Pemrograman	93
5.5 Sentinel	99
5.6 Latihan	110
5.7 Fungsi Rekursif	111
5.8 Iteratif Versus Rekursif	113
5.9 Macam-macam Metode Rekursi	117
5.10 Studi Kasus	119
5.11 Latihan	120
5.12 Uji Pemrograman	124
5.13 Project	125

Bab 6 : Sub Program OOP dan Rekursi	126
6.1 Prosedur (Procedure)	126
6.2 Fungsi (Function)	129
6.3 Parameter Dalam Subprogram	130
6.4 Latihan	133
Bab 7 : Array Satu Dimensi	135
7.1 Pendahuluan	135
7.2 Deklarasi Array	135
7.3 Membaca Elemen Array	136
7.4 Mencetak Elemen Array	138
7.5 String	152
7.6 Workshop	157
7.7 Latihan	160
Bab 8 : Searching Dan Sorting	161
8.1 Pencarian Linier (Linear search)	161
8.2 Pencarian Biner (Binary Search)	163
8.3 Pengantar Pengurutan (Sorting)	165
8.4 Bubble Sort	165
8.5 Insertion Sort	169
8.6 Selection Sort	172
8.7 Merge Sort	177
8.8 Quick Sort	179
8.9 Latihan	182
8.10 Kasus	184
Bab 9 : Array Dua Dimensi, Kasus Matriks	186

9.1 Pendahuluan	186
9.2 Konstruksi Tipe Data Matriks	186
9.3 Membaca Elemen Matriks	186
9.4 Mencetak Elemen Matriks	187
9.5 Workshop	203
9.6 Latihan	203
Bab 10 : Pointer, Pengenalan Struktur Data	207
10.1Pendahuluan	207
10.2Definisi Variabel Pointer	208
10.3Menciptakan Variabel Dinamis	210
10.4Menghapus Pointer	210
10.5Linked List	211
10.6Membuat Daftar Berantai (<i>Linked List</i>)	211
10.7Menampilkan Isi Daftar Berantai	213
10.8Implementasi Pointer Pada Komputer	214
10.9Latihan	215
Daftar Pustaka	216

PENGANTAR

Alhamdulillah, edisi Kedua dari dikat **Algoritma dan Pemrograman** dapat tersaji di hadapan anda. Dibanding dengan edisi pertama, buku ini dirancang untuk dengan pengembangan materi algoritma dan penggunaan Bahasa pemrograman C++ yang lebih luas. Di samping itu, lembaran workshop di akhir bab diharapkan dapat membantu mahasiswa, khususnya dan pembaca pada umumnya.

Aspek-aspek yang dipelajari menyangkut algoritma, flowchart, dan bahasa pemrograman. Translasi ke dalam bahasa pemrograman disajikan dalam bahasa C++. Bahasa C++ dikenal sebagai bahasa yang “smart”, cocok untuk mengeksplorasi kemampuan bahasa sampai mendalam. Bahasa C++ juga dikenal sebagai basis untuk mempelajari bahasa pemrograman yang maju seperti Java dan Symbian C++.

Kompilator yang digunakan adalah Borland C++ Builder X, produk dari Borland International, Inc. Kompilator ini juga dapat digunakan untuk mengkompilasi aplikasi berbasis Symbian. Bagi mahasiswa yang berminat, diberikan kesempatan untuk mengenal Symbian C++ lebih awal.

Buku ini disusun dalam 9 bab, yaitu :

- **Bab 1 : Pendahuluan**, berisi tentang pengertian algoritma, pemrograman dan aturan-aturan formal penulisan algoritma, flowchart dan program. Juga diperkenalkan pendekatan pemrograman berorientasi objek.
- **Bab 2 : Input dan Output**, berisi tentang aspek masukan dan keluaran baik di layar monitor maupun file. Penggunaan operator overloading dala C++ lebi ditekankan.
- **Bab 3 : Sekuen**, berisi tentang aliran program yang tersusun secara runtut dan terurut di mana satu pernyataan dilakukan lebih dulu sebelum pernyataan lain dieksekusi
- **Bab 4 : Pemilihan/ Kontrol Program**, berisi tentang pernyataan pilihan. Pernyataan ini penting karena adanya beberapa alternatif yang harus dipilih berdasarkan kondisi
- **Bab 5 : Perulangan/ Loop**, berisi tentang pernyataan perulangan tanpa adanya pernyataan goto, untuk menekankan pentingnya membangun program secara terstruktur. Dalam bab juga diberikan teknik perulangan dengan rekursi.

- **Bab 6 : Subprogam OOP dan Rekursi**, berisi tentang pentingnya parameter dalam subprogam sebagai saluran komunikasi dengan progam pemanggilnya. Subprogam rekursi memberikan alternatif ringkas untuk menggantikan cara iteratif.
- **Bab 7 : Array Satu Dimensi**, berisi tentang pengelompokan data sejenis
- **Bab 8 : Searching dan Sorting**, berisi tentang teknik pencarian linier (Linear search) dan biner (binary search), sedangkan bagian Sorting membahas algoritma Bubble Sort, Insertion Sort, Selection Sort, Quick Sort dan Merge Sort
- **Bab 9 : Array Dua Dimensi, Kasus Matriks**, berisi tentang algoritma yang menyangkut masalah matriks khususnya
- **Bab 10 : Pointer**, berisi tentang teknik pemrograman menggunakan pointer, dan beberapa algoritma operasi dengan pointer.

Edisi diktat kali ini telah mengalami beberapa revisi bagian yang dirasa tidak sinkron dengan antar komponen: algoritma, flow chart, dan tranlasi ke dalam C++ nya. Ucapan terimakasih kepada bu Dewi Pramudi Ismi, S.T., M.CompSc yang telah memberikan kritik dan saran yang membangun. Belajar algoritma tidak hanya belajar menyelesaikan masalah yang terkait dengan dunia komputer. Belajar algoritma juga berarti belajar menyelesaikan masalah kehidupan dengan cara logis dan dapat dipertanggungjawabkan. Mengenal banyak ragam algoritma berarti semakin beragam pengalaman yang diperoleh. Walau cukup banyak mahasiswa yang “gerah” karena menyita banyak waktu, namun kami yakin, dari sisi kuantitas akan cukup memberikan kebanggaan tersendiri di samping, tentu saja, semakin banyak macam algoritma yang dipelajari, pengalaman menghadapi kasus-kasus akan semakin kaya.

Semoga bermanfaat.

Yogyakarta, 20 Januari 2020

WP

BAB 1

KOMPUTER DAN BAHASA PEMROGRAMAN

1.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menguasai ketrampilan mengetik 10 jari, menguasai penggunaan Raptor, dan menguasai dasar-dasar pengembangan algoritma.

1.2 Komputer Dan Bahasa Pemrograman

Komputer adalah suatu mesin yang melakukan tugas yang sederhana berdasarkan instruksi-instruksi tertentu. Karena tugas-tugas ini dapat dilakukan dengan kecepatan dan ketelitian yang tinggi menjadikan komputer sebagai peralatan yang sangat berguna. Namun demikian untuk menggunakan komputer tersebut sebenarnya sangat sulit, berhubung perintah-perintah yang harus disusun (perintah-perintah itu disebut program) harus berupa kode-kode biner, yaitu : 1 (satu) dan 0 (nol). Sebagai contoh, misalnya akan membuat perintah :

ADD A,B,C

maka perintahnya adalah

0110000100100011

artinya "tambahkan bilangan di lokasi memory A ke bilangan di lokasi memory B dan taruh hasilnya di lokasi memory C". Perintah-perintah tersebut seluruhnya terdiri dari karakter 1 dan 0, dan bahasa yang ditulis seperti ini disebut bahasa mesin. Maka dapat dibayangkan betapa sulitnya menulis program dalam bahasa mesin ini.

Saat ini kita dapat berkomunikasi dengan komputer dengan menggunakan bahasa yang kita mengerti. Hal ini dapat dilakukan karena para ahli telah berhasil membuat kamus yang disebut dengan bahasa pemrograman yang akan menterjemahkan bahasa yang kita buat menjadi bahasa mesin, kamus ini disebut dengan COMPILER. Proses penterjemahan bahasa manusia ke bahasa mesin ini disebut dengan kompilasi. Adapun bahasa-bahasa pemrograman tersebut antara

lain :

Bahasa Pemrograman	Tipe	Dibuat
FORTRAN	Prosedural	1950
BASIC	Prosedural	1960
LISP	Fungsional	1950
Prolog	Deklaratif	1970
Ada	Prosedural	1970
SmallTalk	Berorientasi objek	1970
Pascal	Prosedural	1970
C	Prosedural	1970
C++	Berorientasi objek	1980

1.3 Bahasa Pemrograman Terstruktur

Bahasa Pascal dirancang oleh Niklaus Wirth pada tahun 1970. Turbo Pascal merupakan salah satu versi bahasa Pascal yang dikembangkan oleh Borland International. Bahasa Pascal sendiri termasuk bahasa tingkat tinggi (*high level language*). Instruksinya ditulis dengan menggunakan kata dalam bahasa Inggris. Bahasa Pascal ini cukup mudah untuk difahami karena Pascal ditunjang oleh bentuk program yang terstruktur yang tersusun atas sejumlah blok. Blok-blok yang kecil selanjutnya dapat dipakai untuk membuat blok yang lebih besar. Suatu permasalahan dapat dipecah menjadi bagian-bagian yang kecil sehingga dengan gampang dapat dikodekan. Kesalahan yang terjadi di dalam program akan mudah ditelusuri. Di samping itu program akan mudah dimodifikasi tanpa khawatir menimbulkan efek sampingan terhadap bagian lain dari program. Dalam Pascal, blok lebih dikenal dengan sebutan subprogram, yang dibedakan atas prosedur dan fungsi, sedangkan bahasa C dibangun atas fungsi-fungsi.

Instruksi-instruksi yang diberikan kepada komputer baik mempergunakan bahasa Pascal (Turbo Pascal), bahasa Basic, bahasa Fortran, dll, sebenarnya tidak dapat dimengerti oleh CPU (*Central Processing Unit*) suatu komputer. Hal ini dapat dipergunakan karena setiap bahasa mempunyai suatu program (*software*) yang dapat menterjemahkan bahasa bersangkutan ke bahasa mesin komputer (bahasa yang dapat diikuti/dimengerti langsung oleh CPU suatu komputer. Software demikian disebut dengan *compiler* (kompilator). Dengan demikian setiap bahasa mempunyai compiler masing-masing. Bahasa yang biasa dipakai dalam menyampaikan instruksi kepada komputer seperti bahasa Pascal (Turbo Pascal), Basic, Turbo C, dll disebut bahasa tingkat tinggi (*High Level Language*), sedangkan bahasa yang dapat langsung dapat diikuti oleh CPU komputer disebut bahasa tingkat rendah (*Low Level Language*). Instruksi-instruksi dalam bahasa tingkat tinggi jauh lebih rumit dari instruksi-instruksi sederhana yang

dapat diikuti langsung oleh CPU suatu komputer. Mempelajari bahasa Pascal (termasuk bahasa tingkat tinggi yang lain) tidak harus mengetahui perihal teknis dari mesin komputer secara mendalam, yang penting adalah memahami kaidah-kaidah dari bahasa tersebut. Bahasa tingkat tinggi berorientasi pada bahasa manusia, jadi mudah menulis program, mudah dibaca dan dirawat oleh manusia.

Manusia berkomunikasi dengan mesin komputer lewat operating system (OS), yaitu suatu program yang mengontrol dan mengelola program-program lain yang disampaikan kepada komputer. Untuk memecahkan suatu masalah dengan komputer, bagian yang paling sulit sesungguhnya adalah mendapatkan langkah-langkah penyelesaian yang jelas dan lengkap sehingga dapat memberi solusi dari masalah tersebut, yang disebut dengan **algoritma** dari masalah tersebut.

1.4 Struktur Program Bahasa C++

Skema dari program Pascal adalah sebagai berikut :

Algoritmik	Bahasa C++
Algoritma nama_algoritma { kepala algoritma, berisi penjelasan seperlunya } Deklarasi { berisi variabel yang terlibat } Deskripsi { berisi detail algoritma }	#include <iostream.h> [deklarasi subprogram] main() { [deklarasi variabel] pernyataan; ... return 0; }

1.5 Variable

Variable (perubah) merupakan suatu nama yang menyiratkan lokasi memori komputer yang dapat digunakan untuk menyimpan nilai, di mana isinya dapat diubah-ubah. Variable dapat dipandang sebagai abstraksi dari lokasi. Hasil evaluasi dari variable adalah nilai dari variable itu. Nilai dari suatu variable diubah dengan *assignment statement*. Sebuah *assignment statement* terdiri dari sebuah variable di sebelah kirinya dan suatu ekspresi di sebelah kanannya.

Algoritmik	Bahasa C++
Deskripsi jumlah \leftarrow B1 + B2	jumlah = B1 + B2;

Variabel jumlah diubah nilainya menjadi nilai dari ekspresi B1 + B2 setelah dievaluasi.

Dalam suatu program Pascal maupun bahasa C, setiap variable yang akan digunakan terlebih dahulu dideklarasikan, di mana setiap variable harus mempunyai tipe. Deklarasi variable berguna untuk memberi informasi kepada compiler serta membantu programmer untuk berpikir secara jelas dan berencana.

Algoritmik	Bahasa C++
Deklarasi B1, B2, jumlah : integer	int B1, B2, jumlah;

1.6 Jenis / Tipe Data

Pada waktu sebuah variable dideklarasikan, maka tipenya sekaligus ditentukan. Tipe dari suatu variable menyatakan :

- Jenis nilai yang dapat disimpan dalam lokasi memori untuk variable tersebut, (membatasi himpunan nilai-nilai yang dapat dipunyai variable tersebut),
- Jenis operasi yang dapat dilakukan terhadap variable bersangkutan.

1.7 Pernyataan

Pernyataan adalah deretan instruksi yang akan dieksekusi oleh komputer. Pernyataan dalam bahasa Pascal terdiri dari 2 macam, yaitu :

- pernyataan sederhana,
- pernyataan majemuk (*compound statement*). Dalam bahasa Pascal, pernyataan majemuk diawali dengan **begin** dan diakhiri dengan **end**, sedangkan dalam bahasa C diapit { dan }.

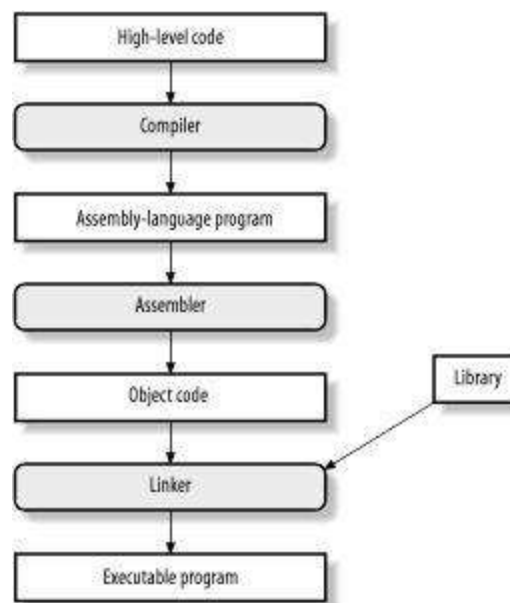
Contoh :

Bahasa C++	Keterangan
<pre>#include <iostream.h> main() { int i = 1; while ((i <= 5)) { cout << "Bilangan : " << i << "\n"; i++; } return 0; }</pre>	<p>Penyertaan header untuk input dan output</p> <p>Fungsi yang dipanggil saat program dieksekusi</p> <p>Deklarasi dan pernyataan sederhana</p> <p>awal pernyataan majemuk , ditandai dengan {</p> <p>kenaikan nilai i</p> <p>akhir pernyataan majemuk</p> <p>program berakhir dengan sukses</p>

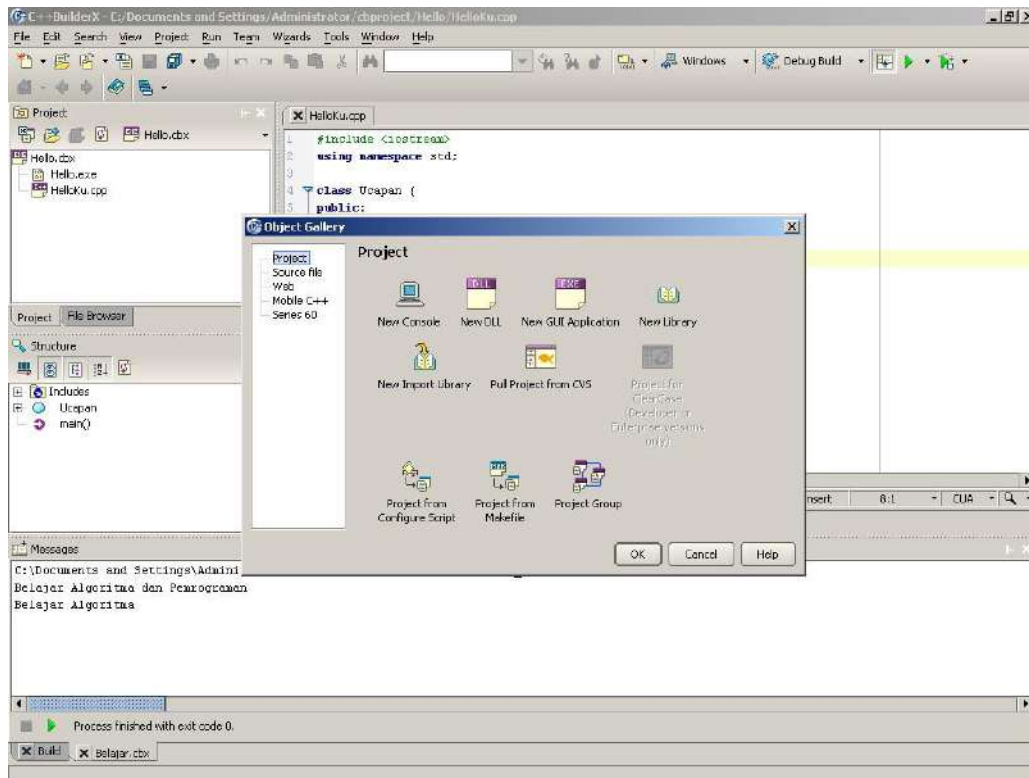
1.8 Program : dari Konsep sampai Eksekusi

Program diawali dari ide seorang programmer. Ketika memjumpai sebuah masalah, dia memikirkan cara menyelesaikan masalah tersebut. Apabila masalahnya rumit, dia akan membagi masalah tersebut menjadi bagian yang lebih kecil untuk diselesaikan.

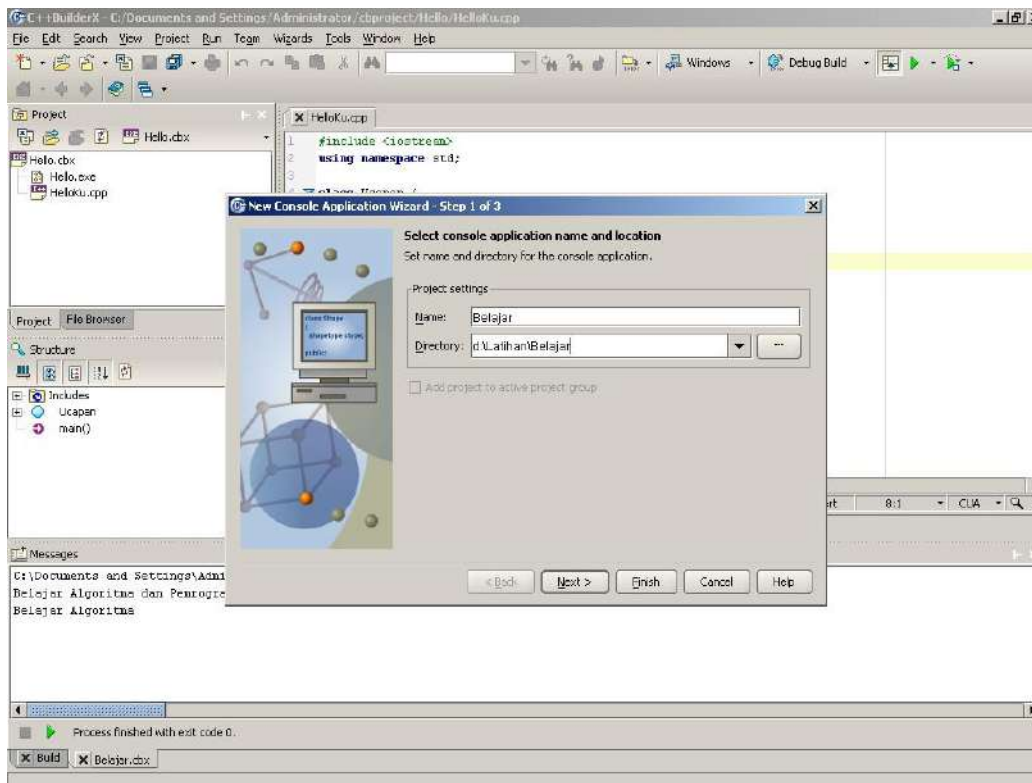
Setelah tahapan penyelesaian ditemukan, dia akan menuliskannya ke sebuah (atau beberapa) file. Selanjutnya file-file ini disebut source file (file sumber) atau source code (kode sumber) yang diketik menggunakan editor teks, seperti notepad atau teks editor yang telah disediakan oleh software kompilator (IDE). File-file ini akan diubah menjadi file objek oleh kompilator. Objek file dikumpulkan dan digabung dengan *routine predefined* dari kepustakaan standar (misalnya untuk urusan input/ output adalah *iostream.h*) oleh *linker* menjadi program yang dapat dieksekusi (*executable program*). Pada dasarnya, *executable program* adalah sekumpulan perintah dari bahasa mesin. Berikut ini diberikan gambar langkah-langkah perubahan dari kode sumber menjadi *executable program*.



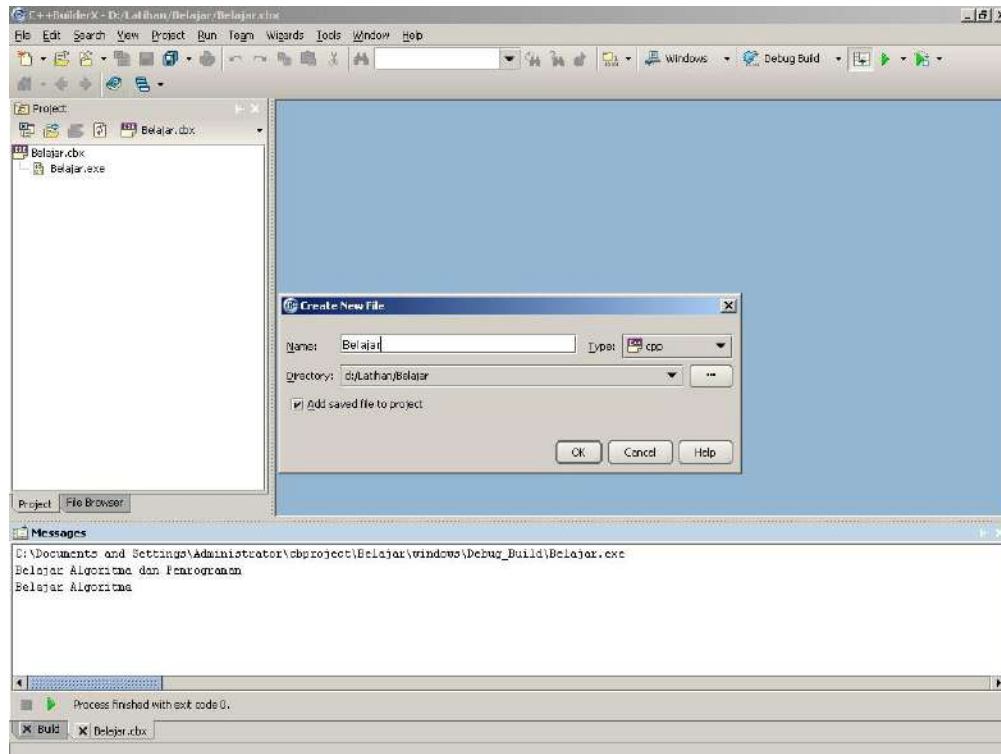
Program C++ dapat dikompilasi dengan berbagai macam kompilator, di antaranya :Turbo C++, Borland C++, gcc (dalam sistem operasi Linux), Visual C++ 6.0 dan Borland C++ Builder X. Untuk membuat program, ketika window awal muncul pilih File | New, tampilan window Borland C++ Builder X seperti berikut ini :



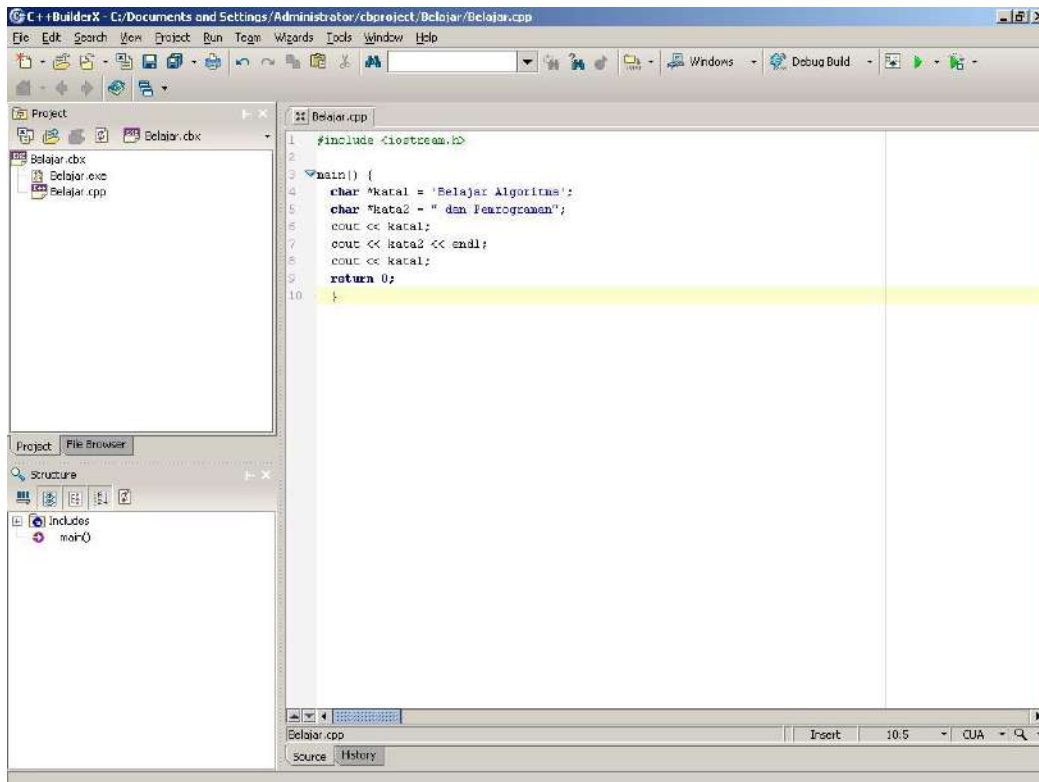
Pilih New Console, karena kita akan membuat program yang nantinya digunakan pada prompt DOS. Setelah tombol Ok ditekan akan muncul tampilan :



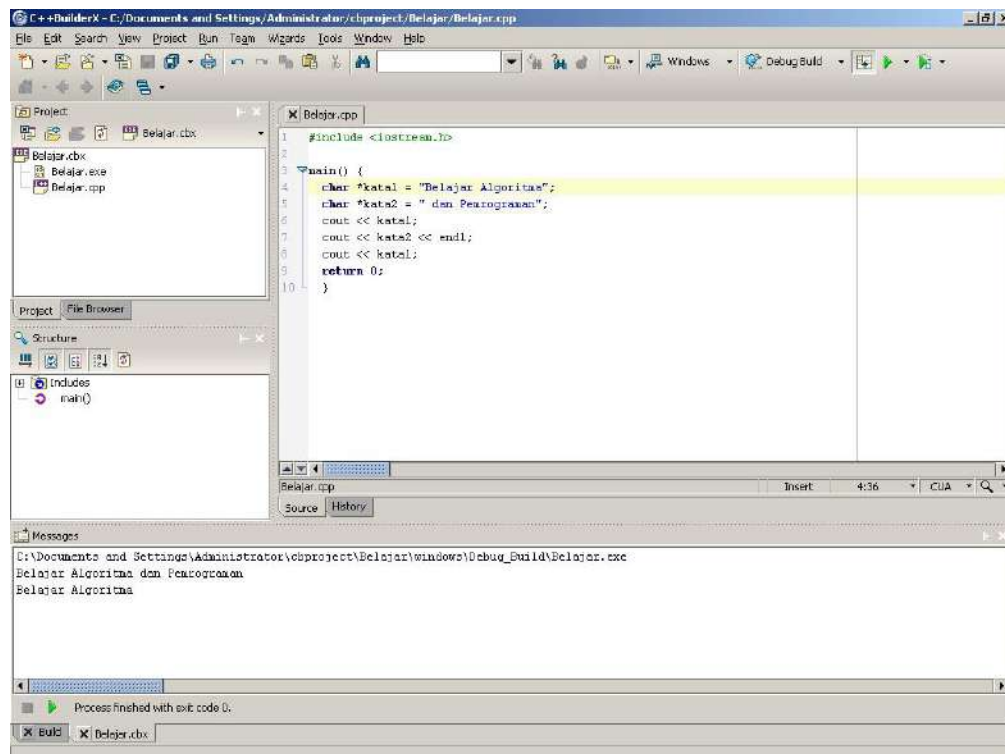
Beri nama project kita dengan Belajar, kemudian isi bagian Directory di mana kita akan menyimpan project, misalnya di D:\Latihan\Belajar. Lalu tekan Finish. Untuk menyertakan file dalam project, pilih File | New File, sehingga tampilannya :



Beri nama filenya Belajar. Pilih dengan Type : cpp. Ada banyak pilihan macam file yang digunakan dalam pemrograman menggunakan bahasa C++. Tekan Ok. Kemudian ketikkan program yang akan kita buat.



Untuk mengkompilasinya pilih menu Run | Run Project atau dengan menekan F9, maka akan muncul gambar berikut :



Kemudian ketiklah program berikut ini.

Baris	belajar.cpp	Penjelasan
-------	-------------	------------

1.	#include <iostream.h>	Menyertakan file iostream.h untuk keperluan input dan output
2.		
3.	main() {	Fungsi main() adalah fungsi yang pertama dijalankan saat program dieksekusi
4.	char *kata1 = 'Belajar Algoritma';	kata1 dan kata2 variabel bertipe string (char *)
5.	char *kata2 = " dan Pemrograman";	mencetak kata1
6.		mencetak kata2
7.	cout << kata1;	fungsi main mengembalikan nilai 0 bertipe int
8.	cout << kata2 << endl;	
	cout << kata1;	
	return 0;	
	}	

Window messages berisi informasi eksekusi program atau pesan-pesan yang muncul saat kompilasi, misal pesan kesalahan

1.9 Mengenal Pemrograman berorientasi Objek

Secara normal, program seperti di atas sudah mencukupi untuk menampilkan nilai variabel bertipe string. Namun ada baiknya kita mulai mengenal kelas sebagai representasi kumpulan objek. Kita akan mengumpulkan objek string dalam sebuah kelas yang diberi nama Kalimat. Prototipe kelas Kalimat dapat dibuat sebagai berikut :

1.	class Kalimat {
2.	public:
3.	Kalimat(char *kata_awal = "") : kata(kata_awal) {
4.	cout << "Selamat datang\n"; }
5.	void cetak() { cout << kata; }
6.	private:
7.	char *kata;
8.	};

Kelas Kalimat terdiri dari 2 bagian yaitu bagian public dan bagian private. Bagian public dapat diakses oleh fungsi lain, sebagai contoh fungsi main dapat mengakses fungsi cetak(). Bagian public ini merupakan antar muka (*interface*) kelas dengan kelas atau fungsi lain. Dengan antar muka ini, objek kelas dapat berkomunikasi dengan objek dari kelas lain.

Bagian private dari kelas Kalimat digunakan untuk menampung data. Di sini dideklarasikan variabel kata bertipe char * (atau lazim dikenal sebagai string). Bagian private biasanya berisi data yang tidak dapat langsung diakses oleh fungsi dari kelas lain. Bagian ini terlindung dari bagian yang tidak diberi hak akses langsung. Sifat ini dinamakan sifat

pengkapsulan (*encapsulation*). Fungsi-fungsi yang hanya digunakan oleh suatu kelas dan tidak berguna bagi kelas lain juga dapat ditempatkan pada bagian *private*.

Ada fungsi istimewa yang namanya sama dengan nama kelas. Fungsi yang demikian dinamakan konstruktor (*constructor*). Konstruktor digunakan untuk memberi nilai awal (inisialisasi) objek yang baru dibuat. Apabila dalam suatu kelas tidak dideklarasikan secara eksplisit, kompilator akan secara otomatis membuatnya. Ini dinamakan *default constructor*. Dalam kelas *Kalimat* di atas, parameter *kata_asal* akan digunakan untuk memberi nilai awal variabel *kata*. Apabila objek baru dibuat tanpa parameter, variabel *kata* akan diisi dengan string kosong (""). Apabila objek baru dibuat dengan parameter, variabel *kata* akan diisi dengan string parameter itu. Sebagai contoh :

Kalimat baru;

Objek baru akan mempunyai nilai kata kosong. Lain halnya dengan deklarasi berikut :

Kalimat isi("Selamat");

Objek isi akan mempunyai nilai kata "Selamat".

Program di bawah ini akan menggantikan program 1. Program ini akan dilengkapi dengan kelas *Kalimat*.

1.	#include <iostream.h>
2.	#include <conio.h>
3.	
4.	class Kalimat {
5.	public:
6.	Kalimat(char *kata_awal = "") : kata(kata_awal) {
7.	cout << "Selamat datang\n"; }
8.	void cetak() { cout << kata; }
9.	private:
10.	char *kata;
11.	};
12.	
13.	main() {
14.	Kalimat kata1("Belajar Algoritma");
15.	Kalimat kata2(" dan Pemrograman");
16.	
17.	kata1.cetak();
18.	kata2.cetak();
19.	cout << endl;
20.	kata1.cetak();
21.	getch();
22.	return 0;
23.	}

1.10 Pengantar Algoritma

Untuk menyusun sebuah algoritma diperlukan pengetahuan yang mendalam tentang permasalahan yang dihadapi. Coba kita analisis perbedaan dua pertanyaan berikut ini.

Berapakah nilai $1 + 2$?

dan

Bagaimanakah algoritma menjumlah dua buah bilangan bulat ?

Untuk menjawab pertanyaan pertama, dengan mudah kita bisa menjawab nilai $1 + 2$ adalah 3. Mengapa ? Karena :

1. operan pertama, operan kedua sudah diketahui
2. operator sudah ditentukan
3. susunan operan dan operatornya sudah tertentu.

Untuk menjawab pertanyaan kedua,

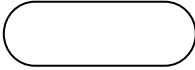

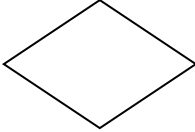


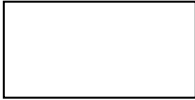
1. harus ditentukan inputnya yaitu bilangan pertama dan bilangan kedua, yang keduanya bertipe integer (bilangan bulat), misalnya $bil1$ dan $bil2$
2. dipilih operator yang sesuai dengan permasalahan, yaitu $+$
3. kemudian disusun suatu ekspresi, yaitu rangkaian operan dan operator yang mempunyai nilai tertentu, yaitu $bil1 + bil2$
4. kemudian $bil1 + bil2$ dievaluasi, misalnya ditempatkan pada variabel output jumlah, ditulis jumlah $\leftarrow bil1 + bil2$.

Mengapa susunan suatu ekspresi itu penting ? Pada masalah di atas kebetulan operator $+$ mempunyai sifat komutatif, sehingga penulisan $bil1 + bil2$ mempunyai arti yang sama dengan $bil2 + bil1$. Tetapi bila operatornya $-$, oleh karena operator $-$ tidak mempunyai sifat komutatif maka $bil1 - bil2$ berbeda maknanya dengan $bil2 - bil1$. Jadi susunan ekspresi menjadi penting karena akan memberikan makna yang spesifik sesuai dengan tujuan penyelesaian masalahnya. Secara umum, susunan algoritma mengikuti pola IPO (Input Proses Output). Jadi algoritma menjumlah dua bilangan bulat adalah sebagai berikut :

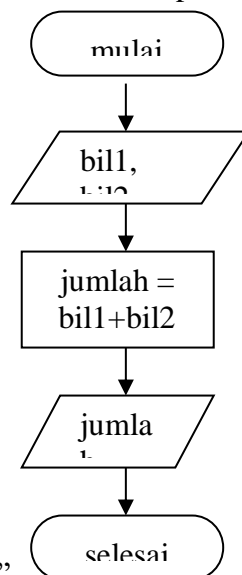
1. baca $bil1$ dan $bil2$ **(input)**
2. jumlah $\leftarrow bil1 + bil2$ **(proses)**
3. tulis(jumlah) **(output)**

2.1 Flowchart

Kadang-kadang perlu digambarkan bagaimana arus data dari algoritma yang sudah dibuat, terutama kalau algoritma sudah cukup kompleks. Untuk itu algoritma dapat disajikan dalam bentuk flowchart. Simbol yang diperlukan di antaranya :

No.	Simbol	Makna
1.		start/mulai end/selesai
2.		input output
3.		kondisi
4.		nilai awal / inisialisasi
5.		perulangan for
6.		proses / penugasan

Algoritma menjumlah dua bilangan bulat di atas dapat dibuat flowchartnya sebagai berikut :



Listing program dengan versi “C like”.

1.	#include <iostream.h>
2.	#include <conio.h>
3.	
4.	void main() {
5.	int a = 5, b = 10;

6.	int hasil;
7.	cout << "Nilai a : " << a << endl;
8.	cout << "Nilai b : " << b << endl;
9.	hasil = a+b;
10.	cout << "Hasil penjumlahan : " << hasil;
11.	getch();
12.	}

Listing program dengan versi menggunakan Class.

1.	#include <iostream.h>
2.	#include <conio.h>
3.	
4.	class Hitung {
5.	friend ostream& operator<<(ostream&, Hitung&);
6.	public:
7.	Hitung();
8.	void jumlahkan() { hasil = a+b; }
9.	private:
10.	int a, b;
11.	int hasil;
12.	};
13.	
14.	Hitung::Hitung() {
15.	cout << "Belajar menghitung\n";
16.	a = 5;
17.	b = 10;
18.	}
19.	
20.	ostream& operator<<(ostream& out, Hitung& a) {
21.	a.jumlahkan();
22.	out << "Nilai a : " << a.a << endl;
23.	out << "Nilai b : " << a.b << endl;
24.	out << "Hasil penjumlahan : " << a.hasil;
25.	return out;
26.	}
27.	void main() {
28.	Hitung X;
29.	cout << X;
30.	getch();
31.	}

Mari kita amati perpindahan style di atas. Kita mulai dari komposisi data.

C Style	C++ Style
int a = 5, b = 10;	class Hitung {
int hasil;	private:
	int a, b;

	int hasil; };
--	---------------

Kemudian proses yg diimplementasikan ke dalam method :

C Style	C++ Style
hasil = a+b;	void jumlahkan() { hasil = a+b; }

Untuk menginisialisasi bagian data member, perhatikan :

C Style	C++ Style
int a = 5, b = 10;	Hitung::Hitung() {
	cout << "Belajar menghitung\n";
	a = 5;
	b = 10; }

Proyek 1 : menggambar persegi panjang

Apabila kita ingin menggambar persegi panjang, tidaklah masalah apakah kita mulai dengan garis atas lebih dulu baru komponen yang lain atau dari garis samping kiri dulu baru komponen yang lain. Mari kita selidiki langkah-langkah menggambar persegi panjang yang mungkin dapat dilakukan.

Komponen persegi panjang :

1. garis bagian atas
2. garis bagian samping kiri
3. garis bagian samping kanan
4. garis bagian bawah

Urutan yang mungkin dilakukan :

- Kemungkinan I : 1, 2, 3, dan 4
- Kemungkinan II : 1, 3, 2, dan 4
- Kemungkinan III : 1, 2, 4, dan 3
- Kemungkinan IV : 1, 2, 3, dan 4, dst

Ternyata kita dapat melakukannya sebanyak $4! = 24$ cara. Kesimpulan yang dapat diambil adalah membuat algoritma dari sebuah masalah tidaklah mungkin hanya satu.

Latihan

Selain contoh-contoh program di atas, ketiklah kemudian kompilasilah program-program berikut ini. Bila terdapat kesalahan, cobalah amati dan terjemahkan kesalahan yang muncul

dalam window debug lalu betulkan kesalahan yang terjadi. Dalam setiap program, buatlah dokumentasi seperlunya.

No. 1	
1.	#include <iostream.h>
2.	#include <conio.h>
3.	
4.	void main() {
5.	cout << "Belajar memprogram\n";
6.	getch();
7.	}

No. 2	
1.	#include <iostream.h>
2.	#include <conio.h>
3.	
4.	class Cetak {
5.	public:
6.	Cetak() { cout << "Belajar memprogram\n"; }
7.	};
8.	
9.	void main() {
10.	Cetak X;
11.	getch();
12.	}

No. 2	
1.	#include <iostream.h>
2.	#include <conio.h>
3.	int jumlahkan(int x, int y) {
4.	return x+y;
5.	}
6.	
7.	void main() {
8.	int a = 5, b = 10;
9.	int hasil;
10.	cout << "Nilai a : " << a << endl;
11.	cout << "Nilai b : " << b << endl;
12.	cout << "Hasil penjumlahan : " << jumlahkan(a,b);
13.	getch();
14.	}

No. 3	
1.	#include <iostream.h>
2.	#include <conio.h>
3.	

4.	void inisialisasi(int&, int&);
5.	void cetak_nilai_awal(int, int);
6.	int jumlahkan(int, int);
7.	void cetak_hasil(int);
8.	
9.	void main() {
10.	int a, b;
11.	int hasil;
12.	
13.	inisialisasi(a, b);
14.	cetak_nilai_awal(a, b);
15.	hasil = jumlahkan(a, b);
16.	cetak_hasil(hasil);
17.	getch();
18.	}
19.	
20.	void inisialisasi(int &x, int &y) {
21.	x = 5;
22.	y = 10;
23.	}
24.	
25.	void cetak_nilai_awal(int x, int y) {
26.	cout << "Nilai a : " << x << endl;
27.	cout << "Nilai b : " << y << endl;
28.	}
29.	
30.	int jumlahkan(int x, int y) {
31.	return x+y;
32.	}
33.	
34.	void cetak_hasil(int output) {
35.	cout << "Hasil penjumlahan : " << output;
36.	}

Workshop Pemrograman

1. Buatlah program yang menampilkan biodata pribadi anda.
2. buatlah program yang meminta input dari user berupa tanggal, bulan dan tahun kelahiran dan menampilkannya dengan format tanggal-bulan-tahun.

3. buatlah program yang menampilkan huruf I yang semuanya menggunakan karakter '*' dengan tinggi 6 karakter dan lebar 3 karakter.
4. buatlah program untuk menghitung luas dan keliling lingkaran beserta volume bola. Buat deklarasi terpisah antara variabel dan konstanta yang diperlukan.
5. buatlah program yang menampilkan pilihan-pilihan menu dari sebuah restoran atau cafe.

6. Buatlah program yang merepresentasikan cerita berikut. Ibu memberi uang kepada adik sebesar 20 ribu rupiah. Ibu meminta adik untuk membeli 3 kg beras. Setiap kilogram beras harganya Rp 5.500.- Berapakah uang kembalian yang diterima adik ?

7. Ulangi untuk kasus no 6. Apabila banyak beras tidak diketahui, berapa kg maksimal beras yang dapat dibeli ?

|

-
- halaman : 20

10. Anda mendapat undangan dari seorang teman untuk menghadiri pesta ulang tahunnya. Pesta ulang tahun tidak diselenggarakan di rumahnya, namun berada di tempat yang belum pernah ada kunjungi. Rancanglah suatu rencana dari membaca undangan sampai ke tempat acara pesta ulang tahun diadakan.

BAB II

INPUT DAN OUTPUT

OPERATOR OVERLOADING dan FILE

2.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan berbagai algoritma yang melibatkan pernyataan, berbagai pernyataan yang berkaitan dengan input/output.

2.2 Pendahuluan

Apabila kita ingin membeli sebuah TV, salah satu pertimbangan kita adalah aspek masukan dan keluaran. Masukan yang kita pertimbangkan antara lain : apakah TV termasuk menggunakan remote control atau tidak, TV tersebut dapat memperoleh sumber dari DVD Player atau tidak, berapa banyak slot untuk menerima masukan dari media lain.

Sedangkan pertimbangan berkaitan dengan keluaran yang kita pertimbangkan adalah : berapa lebar layar yang kita inginkan, apakah layar TV tersebut digital atau analog, layarnya datar (flat) atau melengkung, suaranya mono, stereo atau bahkan surround.

Pada kebanyakan produk, aspek masukan dan keluaran memegang peranan yang sangat penting. Pengguna akan dapat membayangkan bagaimana dia akan menggunakan peralatan tersebut dengan fungsional dan nyaman. Pengguna juga dapat memperkirakan kualitas keluaran sesuai dengan yang dikehendaki atau tidak.

Dalam sebuah sistem, secara umum terdiri dari input, proses, output. Masukan (input) merupakan bahan yang nantinya akan diproses untuk menghasilkan informasi (output). Dalam bab ini akan dibahas tentang bagaimana memformat masukan dan keluaran dengan baik (sesuai dengan kaidah user interface dan analisis kebutuhan).

2.3 Memberi masukan : tradisional dan class

Misalkan kita ingin mempunyai model dialog seperti gambar di bawah ini :

```
Program menghitung nilai F=m.a
Selamat berkarya
Masukkan nilai m : 
Masukkan nilai a :
```

Tanda `█` menandakan pada posisi tersebut pengguna mengetikkan masukan. Kita dapat merancang dialog di atas dengan pernyataan berikut ini.

```
cout << "Program menghitung nilai F=m.a\n";
cout << "Selamat berkarya\n";
cout << "Masukkan nilai m : ";
cin >> m;
cout << "Masukkan nilai a : ";
cin >> a;
```

Pernyataan keluaran dalam C++ adalah operator `<<`. Sedangkan `cout` merupakan variabel yang lazim digunakan untuk menampung hasil dari operator keluaran `<<`. Pernyataan masukan dalam C++ adalah operator `>>`. `cin` merupakan variabel yang lazim digunakan untuk menampung hasil dari operator masukan `<<`.

Apabila program yang kita buat sederhana, maka dialog masukan dapat kita tempatkan pada fungsi `main()`. Sehingga dialog di atas menjadi :

```
void main() {
    cout << "Program menghitung nilai F=m.a\n";
    cout << "Selamat berkarya\n";
    cout << "Masukkan nilai m : ";
    cin >> m;
    cout << "Masukkan nilai a : ";
    cin >> a;

    .. // proses

    getch();
}
```

Apabila kita ingin memisahkan dialog masukan dari fungsi main, kita dapat membuat fungsi terpisah. Misalkan nama fungsinya adalah masukan_data. Implementasinya sebagai berikut :

```
void masukan_data() {  
    cout << "Program menghitung nilai F=m.a\n";  
    cout << "Selamat berkarya\n";  
    cout << "Masukkan nilai m : ";  
    cin >> m;  
    cout << "Masukkan nilai a : ";  
    cin >> a;  
}
```

Andaikan kita mempunyai sebuah kelas, namakan kelas Gaya yang kita definisikan sebagai berikut :

```
class Gaya {  
    friend ostream& operator<<(ostream&, const Gaya&);  
    friend istream& operator>>(istream&, Gaya&);  
public:  
    // berisi methods atau fungsi  
private:  
    int m, a;  
    int F;  
    // berisi data-data  
};
```

Kita dapat membuat dialog masukan dengan membuat operator berbeban lebih (operator overloading) untuk operator >> untuk kelas Gaya sebagai berikut :

```
istream& operator>>(istream& in, Gaya& masukan) {  
    cout << "Program menghitung nilai F=m.a\n";  
    cout << "Selamat berkarya\n";  
    cout << "Masukkan nilai m : ";  
    in >> masukan.m;  
    cout << "Masukkan nilai a : ";  
    in >> masukan.a;  
    return in;  
}
```


Objek masukan nantinya akan digantikan oleh variabel cin saat kita memanggilnya dalam fungsi main().

```
void main() {  
    Gaya X;  
    cin >> X;  
  
    // proses  
  
    getch();  
}
```

Namun akan muncul pertanyaan : “ Apakah dengan membuatnya menjadi operator overloading kode menjadi lebih panjang ?”. Memang benar. Bila kita memposisikan diri sebagai “pembuat” memang terasa pekerjaan menjadi berat. Namun bila kita sebagai pengguna (lihat dalam fungsi main), maka cara pemanggilannya menjadi sangat sederhana tidak ubahnya memberikan masukan pada variabel sederhana seperti int, float atau char.

Apalagi ditunjang dengan karakteristik dari operator overloading yang khas. Sekali kita tahu bagaimana membuat operator overloading, kitapun dapat dengan mudah membuat operator overloading untuk kelas-kelas yang lain.

Bentuk umum operator overloading masukan.

```
istream& operator>>(istream& variabelPenampung, namaKelas& objekDari_namaKelas)  
{  
    // pernyataan yang diperlukan dalam dialog  
    return variabelPenampung;  
}
```

Perlu diingat, operator overloading merupakan friend dari sebuah kelas. Dengan demikian operator overloading dapat mengakses bagian private dari kelas tersebut. Caranya dengan menyebut objek dari kelas diikuti operator . (titik) kemudian variabel bagian private yang dikehendaki. Sebagai contoh, pernyataan :

in >> masukan.m;

mempunyai arti : pengguna akan memasukkan data ke variabel m objek masukan dari kelas Gaya.

Selanjutnya, ragam dialog dalam contoh-contoh akan disajikan menggunakan operator overloading masukan dan keluaran.

Workshop Pemrograman : Operator Overloading

1. Buatlah program yang meminta user memasukkan jam, menit dan detik kemudian menampilkannya dengan format jam:menit:detik. Deklarasikan jam dan menit berupa integer, sedangkan menit berupa floating point. Buat masukan dan keluaran menggunakan operator overloading << dan >> sesuai yang dikehendaki.

- a. Buat class waktu, termasuk operator overloading input dan output

```
class waktu {  
    friend ostream& operator<<(ostream&, const waktu&);  
    friend istream& operator>>(istream&, waktu &);  
public:  
    // berisi methods atau fungsi  
  
private:  
  
  
};
```

- b. Buat implementasi operator overloading input dan output

- c. Buat main function

Untuk soal selanjutnya ulangi langkah-langkah sebagaimana soal nomor 1.

2. Buatlah program yang meminta user memasukkan bagian penyebut dan pembilang dari sebuah bilangan rasional berbentuk p/q . Setiap memasukkan bilangan rasional outputnya berbentuk p/q . Misalnya, masukan 1 dan 2 maka tampilan outputnya $\frac{1}{2}$.
3. Buatlah program yang meminta user memasukkan bagian ribuan, ratusan, puluhan dan satuan. Misalnya : 1000, 200, 30, 4. Tampilan yang dikehendaki adalah 1234.

4. Buatlah program untuk merepresentasikan operasi-operasi aritmatika : penjumlahan, pengurangan, perkalian dan pembagian. User diminta memasukkan 2 buah bilangan bulat, kemudian menampilkan sajian lengkapnya dari semua operasi. Sebagai contoh, user memasukkan 1 dan 2 maka tampilan outputnya :

$$1 + 2 = 2$$

$$1 - 2 = -1$$

$$1 * 2 = 2$$

$$1 : 2 = \frac{1}{2}$$

```
class Hitung {
```

```
}
```

5. Buatlah program untuk merepresentasikan bilangan bulat menjadi bilangan scientific number berbentuk 1.23E1 untuk menyatakan bilangan 12.3. Masukan bilangan bulat 4 digit, keluaran berbentuk aEb, dengan a adalah bilangan antara 0 dan 10, sedangkan b dari 0 sampai 4.

2.4 File

Penyimpanan data dalam variabel dan array bersifat sementara; data yang seperti itu akan hilang bila program selesai. File digunakan untuk penyimpanan data secara permanen dalam jumlah yang besar. File disimpan pada peralatan penyimpanan sekunder yang paling lazim adalah disk (bisa disket atau harddisk). Khusus dalam bab ini akan dibahas file akses sekuensial dan file akses random dari sudut pandang bahasa C. Walaupun demikian, prosedur atau fungsi dalam bahasa Pascal yang berkaitan dengan operasi file juga diberikan.

2.5 Hirarki Data

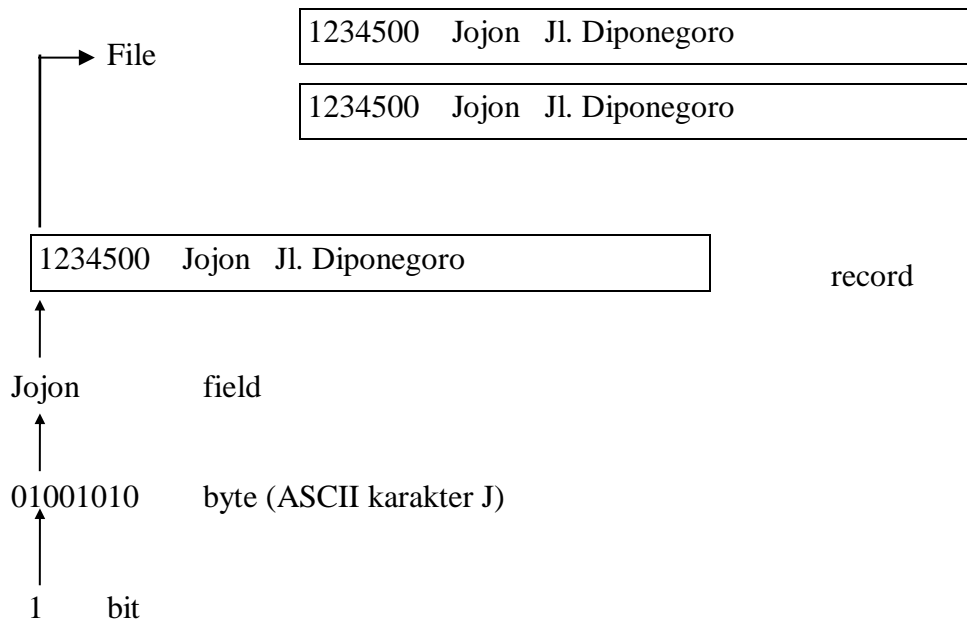
Semua data yang diproses komputer yang disederhanakan menjadi kombinasi 0 dan 1. Jadi secara mendasar, data yang diproses komputer adalah manipulasi 0 dan 1. Susunan data yang demikian dinamakan bit (*binary digit*).

Karakter disusun atas bit. Sedangkan field disusun atas karakter-karakter. Field adalah kelompok data yang mempunyai arti. Misalnya, suatu field berupa huruf besar dan kecil yang merepresentasikan nama orang. Data diproses oleh komputer berbentuk hirarki data di mana item data menjadi besar dan kompleks dalam struktur dari bit-bit menjadi karakter (byte), menjadi field dst.

Suatu record (yaitu **struct** dalam C) tersusun atas field-field. Sebagai contoh : dalam suatu sistem pembayaran gaji pegawai, record dapat terdiri dari field :

1. SSN (Sosial Security Number)
2. Nama
3. Alamat
4. Gaji Pokok
5. Tunjangan
6. Pajak, dsb.

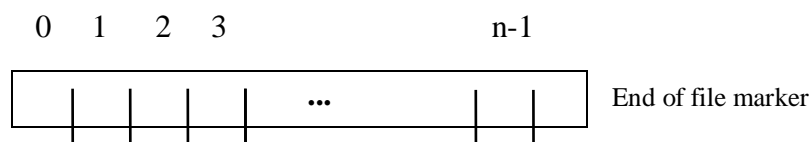
Dengan demikian, record adalah kelompok field yang berhubungan satu dengan lainnya. Suatu file adalah kelompok rekod yang saling berhubungan. Dalam suatu perusahaan, setiap pegawai direpresentasikan dalam satu record. Bila terdapat 50 pegawai maka terdapat 50 record.



Untuk memudahkan mendapatkan record tertentu dalam suatu file, paling tidak terdapat satu field dalam suatu record yang dipilih sebagai field kunci. Kunci record mengidentifikasi suatu record sebagai pemilik terhadap seseorang atau entitas. Misalnya dalam contoh di atas dipilih **SSN** sebagai kunci record. Biasanya, record-record diurutkan dari yang terkecil sampai dengan yang terbesar. Kelompok file yang saling terhubung dinamakan basis data. Kumpulan program yang dirancang untuk membuat dan mengatur basis data disebut *database management system* (DBMS).

2.6 File dan Stream

C memandang setiap file sebagai stream sekuensial byte. Setiap file berakhir dengan *end-of-file marker* (tanda akhir file). C memandang file n byte terlihat seperti di bawah ini.



Bila file dibuka, stream dihubungkan dengan file. Tiga file dan stream terhubungnya secara otomatis dibuka bila program mulai dieksekusi yaitu *standard input* (**stdin**), *standard output* (**stdout**), dan *standard error* (**stderr**). Stream menyediakan saluran komunikasi antara file dan program. Standard input memungkinkan program untuk membaca data dari keyboard, dan stream standard output memungkinkan program untuk menampilkan data pada layar. Membuka suatu file mengembalikan pointer ke struktur FILE (didefinisikan pada **stdio.h**) yang berisi informasi yang digunakan untuk memproses file. Struktur tersebut termasuk *file descriptor*, yaitu

indeks ke dalam sistem operasi array yang dinamakan *open array table*. Setiap elemen array berisi *file control block* (FCB) di mana sistem operasi menggunakannya untuk mengurus file tertentu.

2.7 Kepustakaan Standar untuk File

<i>Function</i>	<i>Keterangan</i>
fgetc	membaca 1 karakter dari file. fgetc menerima argumen pointer FILE dari file di mana karakter akan dibaca. Fgetc(stdin) sama dengan getchar().
fputc	Menulis 1 karakter ke file. fputc menerima argumen 1 karakter untuk ditulis dan pointer untuk file di mana karakter akan ditulis. fputc('a', stdout) ekuivalen dengan putchar('a').
fgets	membaca 1 baris dari file
fputs	menulis 1 baris ke file

2.8 Membuat File Akses Sekuensial

Perhatikan contoh program di bawah ini.

Translasi 2.1. : Membuat file sekuensial

```
#include <stdio.h>

main(){
    int account;
    char nama[30];
    float balance;
    FILE *cfPtr;

    if ((cfPtr = fopen("clients.dat", "w")) == NULL)
        printf("File tidak dapat dibuka\n");
    else {
        printf("Masukkan account, nama dan balance.\n");
        printf("Masukkan EOF untuk mengakhiri input.\n");
        printf("> ");
        scanf("%d%s%f", &account, nama, &balance);

        while (!feof(stdin)) {
            fprintf(cfPtr, "%d %s %.2f\n", account, nama, balance);
            printf("> ");
            scanf("%d%s%f", &account, nama, &balance);
        }
    }
}
```



```

        fclose(cfPtr);
    }
    return 0;
}

```

Output :

```

Masukkan account, nama dan balance.
Masukkan EOF untuk mengakhiri input.
> 1 Seno 200
> 2 Seto 350
> 3 Agus 0
> 4 Amir -100
> ^Z

```

Pernyataan

```
FILE *cfPtr;
```

menyatakan bahwa **cfPtr** adalah pointer ke struktur FILE. Setiap file mempunyai struktur FILE sendiri. Setiap file terbuka harus mempunyai deklarasi pointer secara terpisah tipe FILE yang digunakan untuk merujuk ke file.

Pernyataan

```
if ((cfPtr = fopen("clients.dat", "w")) == NULL)
```

menyatakan “clients.dat” digunakan program dan menyambung baris komunikasi dengan file. Pointer file **cfPtr** mengassign pointer ke struktur FILE untuk membuka file dengan **fopen**. Function **fopen** mempunyai dua argumen yaitu nama file dan modus pembukaan file. Modus “w” menunjukkan bahwa file dibuka untuk *ditulis*. Jika file tidak ada maka **fopen** akan membuat file. Jika sudah ada, file yang ada akan ditimpa. Struktur **if** digunakan untuk menentukan apakah pointer file **cfPtr** adalah NULL (jika file tidak dapat dibuka). Jika NULL, pesan kesalahan akan ditampilkan dan program berakhir, jika tidak input diproses dan ditulis ke file.

Di bawah ini adalah berbagai variasi tanda akhir file dari berbagai sistem komputer.

<i>Sistem Komputer</i>	<i>Kombinasi Kunci</i>
Sistem UNIX	<return><ctrl> d
IBM PC & kompatibel	<ctrl> z
Macintosh	<crtl> d
VAX (VMS)	<ctrl> z

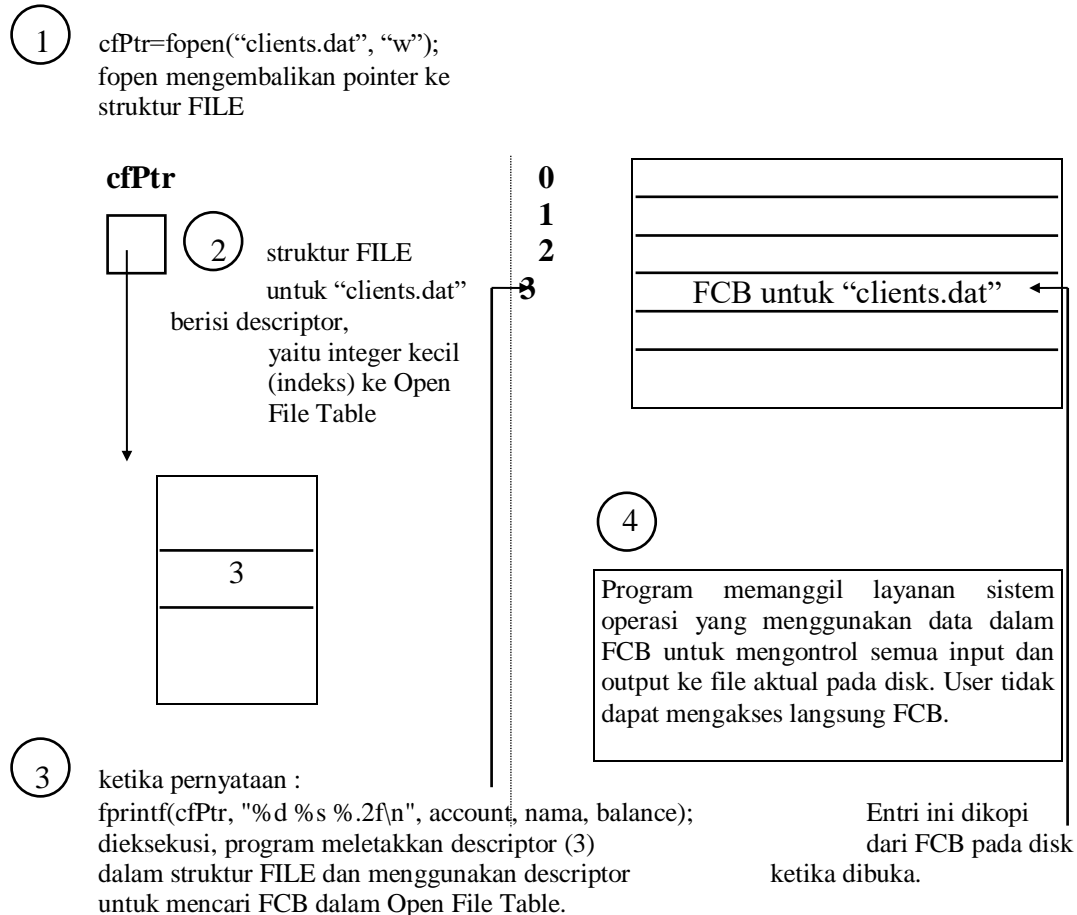
Pernyataan

```
while (!feof(stdin))
```

menggunakan function **fEOF** untuk menentukan apakah tanda akhir file diset untuk file dengan merujuk **stdin**. Tanda akhir file menginformasikan program bahwa tidak terdapat data lagi untuk diproses.

Sesudah user memasukkan tanda akhir file, program menutup **clients.dat** dengan **fclose** dan selesai. Function **fclose** menerima argumen pointer file (bukan nama file). Jika **fclose** secara eksplisit tidak diberikan, sistem operasi akan menutup file secara normal.

Hubungan antara pointer FILE, struktur FILE dan FCB dapat digambarkan seperti di bawah ini. Ketika file “clients.dat” dibuka, FCB untuk file dikopi ke memori.



Program bisa memproses satu atau lebih file. Setiap file dalam program harus mempunyai nama yang unik dan akan mempunyai pointer file berbeda yang *direturn* oleh **fopen**. Semua pernyataan function yang memproses file sesudah file dibuka harus merujuk ke file dengan pointer file yang sesuai. File bisa dibuka dengan berbagai macam modus :

Modus	Keterangan
w	membuat file atau menghapus isi file sebelum menulis data
r	untuk membuka file yang ada
a	untuk menambah record pada akhir file yang ada

r+	membuka file untuk membaca dan menulis
w+	membuat file untuk membaca dan menulis, jika file ada isi akan dihapus
a+	membuka atau membuat file untuk update; penulisan dilakukan pada akhir file

Jika kesalahan muncul ketika membuka file, **fopen** akan mengembalikan NULL.

2.9 Membaca Data dari File Akses Sekuensial

Data disimpan dalam file sehingga data bisa diterima untuk pemrosesan ketika diperlukan.

Translasi 2.2. : Membaca dan mencetak file sekuensial

```
#include <stdio.h>

void main(){
    int account;
    char nama[30];
    float balance;
    FILE *cfPtr;

    if ((cfPtr = fopen("clients.dat", "r")) == NULL) printf("File tidak dapat dibuka\n");
    else {
        printf("%-10s%-13s\n", "Account", "Nama", "Balance");
        fscanf(cfPtr, "%d%s%f", &account, nama, &balance);

        while (!feof(cfPtr)) {
            printf("%-10d%-13s7.2f\n", account, nama, balance);
            fscanf(cfPtr, "%d%s%f", &account, nama, &balance);
        }
        fclose(cfPtr);
    }
}
```

Output :

Account	Nama	Balance
1	Seno	200.00
2	Seto	350.00
3	Agus	0.00
4	Amir	-100.00

Pernyataan

```
fscanf(cfPtr, "%d%s%f", &account, nama, &balance);
```

membaca record dari file. Function **fscanf** ekivalen dengan function **scanf** kecuali **fscanf** menerima pointer file sebagai argumen dari mana file dibaca.

Untuk menerima data dari file secara sekuensial, program mulai membaca dari awal file, dan membaca semua data secara berturutan sampai data yang diperlukan ditemukan. Pernyataan `rewind(cfPtr);`

menyebabkan pointer posisi file program, yang menunjukkan sejumlah byte berikutnya dalam file untuk dibaca atau ditulis, dikembalikan posisinya ke awal file, yaitu byte 0. Pointer posisi file sesungguhnya bukanlah pointer, tetapi suatu harga integer yang menentukan lokasi byte dalam file pada saat dibaca atau ditulis berikutnya muncul. Hal ini disebut sebagai *file offset*. Pointer posisi file adalah anggota dari struktur FILE yang bersesuaian dengan setiap file.

Translasi 2.3. : Contoh Kasus <i>retrieve</i> Data

```
#include <stdio.h>
void main() {
    int request, account;
    float balance;
    char nama[30];
    FILE *cfPtr;
    if ((cfPtr = fopen("clients.dat", "r")) == NULL)
        printf("File tidak dapat dibuka.\n");
    else {
        printf("Masukkan request : \n"
            " 1 - Daftar account dengan balance NOL\n"
            " 2 - Daftar account dengan balance kredit\n"
            " 3 - Daftar account dengan balance debet\n"
            " 4 - Selesai\n> ");
        scanf("%d", &request);
        while (request != 4) {
            fscanf(cfPtr, "%d%s%f", &account, nama, &balance);
            switch(request) {
                case 1:
                    printf("\nAccount dengan balance NOL : \n");
                    while (!feof(cfPtr)) {
                        if (balance == 0) printf("%-10d%-13s%7.2f\n", account, nama, balance);
                        fscanf(cfPtr, "%d%s%f", &account, nama, &balance);
                    } break;
                case 2:
                    printf("\nAccount dengan balance kredit : \n");
                    while (!feof(cfPtr)) {
                        if (balance < 0) printf("%-10d%-13s%7.2f\n", account, nama, balance);
                        fscanf(cfPtr, "%d%s%f", &account, nama, &balance);
                    }
            }
        }
    }
}
```

```

        } break;
    case 3:
        printf("\nAccount dengan balance debet :\n");
        while (!feof(cfPtr)) {
            if (balance > 0) printf("%-10d%-13s%7.2f\n", account, nama, balance);
            fscanf(cfPtr, "%d%s%f", &account, nama, &balance);
        } break;
    }
    rewind(cfPtr);
    printf("\n> ");
    scanf("%d", &request);
}
printf("Selesai.\n");
fclose(cfPtr);
}
}

```

Output :

```

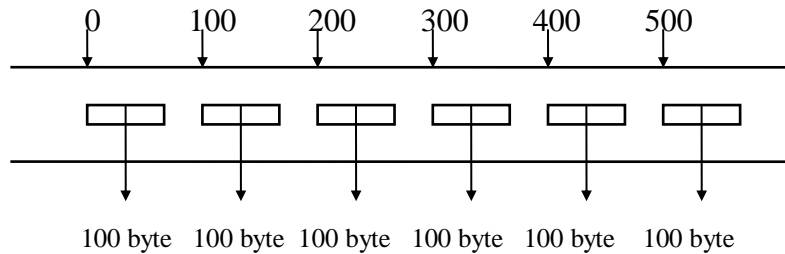
Masukkan request :
1 - Daftar account dengan balance NOL
2 - Daftar account dengan balance kredit
3 - Daftar account dengan balance debet
4 - Selesai
> 1
Account dengan balance NOL :
3    Agus    0.00
> 2
Account dengan balance kredit :
4    Amir   -100.00
> 3
Account dengan balance debet :
1    Seno    200.00
2    Seto    350.00
> 4
Selesai.

```

2.10 File Akses Random (FAR)

Record dalam suatu file dengan function output terformat **fprintf** tidak perlu mempunyai panjang yang sama. Namun, record individu dari file akses random mempunyai panjang tertentu dan dapat diakses langsung tanpa mencari melalui record yang lain, sehingga proses pencarian akan lebih cepat.

Oleh karena setiap record mempunyai panjang yang sama, lokasi yang tepat dari suatu record relatif terhadap awal file dapat dihitung sbg fungsi kunci record.



Data dapat disisipkan dalam file akses random tanpa menghapus data yang lain dalam file. Data yang disimpan sebelumnya jugadpt *diupdate* atau dihapus tanpa menulis keseluruhan file.

2.11 Membuat File Akses Random

Function **fwrite** mentransfer sejumlah awal byte tertentu pada lokasi tertentu dalam memori ke file. Data ditulis awal pada lokasi dalam file ditunjukkan oleh pointer posisi file. Function **fread** mentransfer sejumlah byte tertentu dari lokasi dalam file yang ditentukan oleh pointer posisi file ke daerah awal memori dengan alamat tertentu.

Pernyataan

```
fprintf(fPtr, "%d", number);
```

dapat menulis sedikitnya 1 digit dan paling banyak 11 digit (10 digit ditambah tanda) untuk 4 byte integer. Pernyataan

```
fwrite(&number, sizeof(int), 1, fPtr);
```

selalu menulis 4 byte dari variabel **number** ke file yang direpresentasikan oleh **fPtr**. Function **fread** dapat digunakan untuk membaca 4 byte itu ke variabel integer **number**. Argumen ketiga dari **fread** maupun **fwrite** ad banyaknya elemen dalam array yang akan dibaca dari disk atau ditulis ke disk. **fwrite** di atas menulis integer tunggal ke disk sehingga argumen ketiga adalah 1 (sebagaimana kalau 1 elemen array ditulis).

Function **fwrite** dapat digunakan untuk menulis beberapa elemen array objek. Untuk menulis beberapa elemen array programmer menyediakan pointer ke array sebagai argumen pertama dalam pemanggilan ke **fwrite** dan menentukan banyaknya elemen untuk ditulis sebagai argumen ketiga dalam pemanggilan **fwrite**.

Translasi 2.4. : Membuat file akses random secara sekuensial

```
#include <stdio.h>
struct clientData {
    int acctNum;
    char lastName[15];
    char firstName[10];
```

```

float balance;
};
void main() {
    int i;
    struct clientData blankClient = {0, "", "", 0.0};
    FILE *cfPtr;
    if ((cfPtr = fopen("credit.dat", "w")) == NULL) printf("File tidak dapat dibuka.\n");
    else {
        for (i=1; i<=100; i++) fwrite(&blankClient, sizeof(struct clientData), 1, cfPtr);
        fclose(cfPtr);
    }
}

```

2.12 Menulis Data Secara Random pada FAR

Program di bawah ini menulis data ke file “**credit.dat**” yang menggunakan kombinasi **fseek** dan **fwrite** untuk menyimpan data pada lokasi tertentu dalam file. Function **fseek** mengatur pointer posisi file ke posisi tertentu dalam file kemudian **fwrite** menulis data.

Translasi 2.5. : Menulis ke file akses random

```

#include <stdio.h>
struct clientData {
    int acctNum;
    char lastName[15];
    char firstName[10];
    float balance; };
void main() {
    struct clientData client;
    FILE *cfPtr;
    if ((cfPtr = fopen("credit.dat", "r+")) == NULL) printf("File tidak dapat dibuka.\n");
    else {
        printf("Masukkan account (1 - 100, 0 untuk selesai)\n> ");
        scanf("%d", &client.acctNum);
        while (client.acctNum != 0) {
            printf("Masukkan lastname, firstname, balance\n> ");
            scanf("%s%s%f", &client.lastName, &client.firstName, &client.balance);
            fseek(cfPtr, (client.acctNum - 1) * sizeof(struct clientData), SEEK_SET);
            fwrite(&client, sizeof(struct clientData), 1, cfPtr);
            printf("Masukkan account\n> ");
            scanf("%d", &client.acctNum);
        }
    }
    fclose(cfPtr);
}

```

```
}
```

Output :

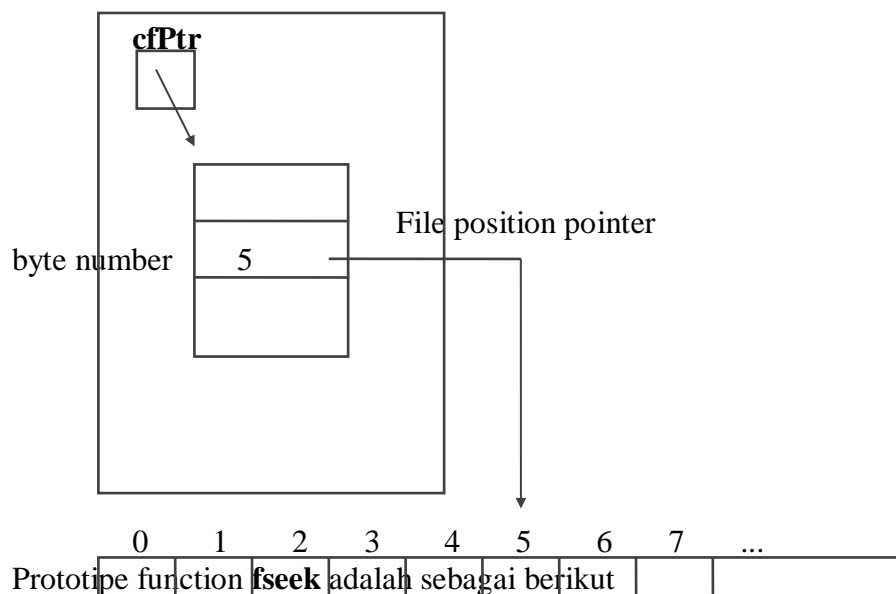
```
Masukkan account (1 - 100, 0 untuk selesai)
> 3
Masukkan lastname, firstname, balance
> nugroho eko 100
Masukkan account
> 7
Masukkan lastname, firstname, balance
> santoso insap 500
Masukkan account
> 0
```

Pernyataan

```
fseek(cfPtr, (client.acctNum - 1) * sizeof(struct clientData), SEEK_SET);
```

meletakkan pointer posisi file untuk file yang dirujuk oleh **cfPtr** ke lokasi byte yang dihitung oleh **(client.acctNum - 1) * sizeof(struct clientData)**. Harga dari ekspresi ini dinamakan *offset* atau *displacement*. Konstanta simbolik **SEEK_SET** menunjukkan bahwa pointer posisi file diletakkan relatif terhadap awal file oleh sejumlah offset.

Gambar di bawah ini memperlihatkan pointer posisi file ke struktur FILE dalam memori. Pointer posisi file menunjukkan offset 5 byte dari awal file.



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


di mana **offset** adalah sejumlah byte dari lokasi **whence** dalam file yang ditunjuk oleh **stream**. Argumen **whence** dapat mempunyai 1 dari 3 harga yaitu **SEEK_SET**, **SEEK_CUR**, atau **SEEK_END** yang menunjukkan dari posisi mana dalam file pencarian akan dimulai.

SEEK_SET : pencarian dimulai dari awal file

SEEK_CUR : pencarian dimulai dari lokasi saat ini dalam file

SEEK_END : pencarian dimulai dari akhir file.

Ketiga konstanta simbolik tersebut ada pada file header **stdio.h**.

2.13 Membaca Data Secara Random pada FAR

Function **fread** membaca sejumlah byte tertentu dari file ke dalam memori. Pernyataan :

```
fread(&client, sizeof(struct clientData), 1, cfPtr);
```

membaca sejumlah byte yang ditentukan oleh **sizeof(struct clientData)** dari file yang dirujuk oleh **cfPtr** dan menyimpan data dalam struktur **client**. Byte yang dibaca dari lokasi yang ditentukan oleh pointer posisi file. Function **fread** dapat digunakan untuk membaca beberapa elemen array dengan ukuran tertentu dengan menyediakan suatu pointer ke array di mana elemen akan disimpan, dan dengan menunjukkan jumlah elemen yang dibaca. Pernyataan di atas menentukan 1 elemen akan dibaca. Untuk membaca lebih dari elemen, tentukan jumlah elemen dalam argumen ketiga dari pernyataan **fread**.

Translasi 2.6. : Membaca file akses random secara sekuensial

```
#include <stdio.h>
struct clientData {
    int acctNum;
    char lastName[15];
    char firstName[10];
    float balance; };
void main() {
    struct clientData client;
    FILE *cfPtr;

    if ((cfPtr = fopen("credit.dat", "r")) == NULL) printf("File tidak dapat dibuka.\n");
    else {
        printf("%-6s%-16s%-11s%10s\n", "Acct", "Last Name", "First Name", "Balance");
        while (!feof(cfPtr)) {
            fread(&client, sizeof(struct clientData), 1, cfPtr);
            if (client.acctNum != 0)
                printf("%-6d%-16s%-11s%10.2f\n", client.acctNum,
                    client.lastName, client.firstName, client.balance);
        }
    }
}
```

```

    }
}
fclose(cfPtr); }

```

Output program 6.

```

Acct Last Name    First Name  Balance
3  nugroho      eko        100.00
7  santoso      insap      500.00

```

2.14 Rangkuman Fungsi Untuk Operasi File (C)

Function	Arti	Mode	Arti
fopen	membuka file	r	membuka file untuk dibaca
		w	membuat file untuk penulisan. Jika file ada, isinya akan dihapus
		a	membuka atau membuat file. Penulisan ditambahkan pada akhir file
		r+	membuka file untuk update (pembacaan dan penulisan)
		w+	membuat file untuk update. Jika file ada, isinya akan dihapus
		a+	membuka atau membuat file untuk update. Penulisan ditambahkan pada akhir file
feof	fungsi untuk menentukan apakah indicator pada akhir file atau tidak. Untuk IBM PC, akhir file ditandai dengan <ctrl> z		
fclose	fungsi untuk menutup file		
fprintf	menulis data ke file. Seperti printf tetapi fprintf menerima argumen file pointer untuk file di mana akan ditulis		
fscanf	membaca record dari file. Seperti scanf tetapi fscanf mengambil argumen pointer file dari mana data dibaca.		
rewind	menyebabkan posisi pointer file pada awal file		
fread	membaca blok (sejumlah byte tertentu) dari file		
fwrite	menulis blok (sejumlah byte tertentu) ke file		
fseek	mengatur posisi pointer file ke posisi tertentu pada file didasarkan pada lokasi awal pencarian dalam file.	Mode	
		SEEK_SET	mulai dari awal file
		SEEK_CUR	mulai dari posisi saat ini pada file
		SEEK_END	mulai dari akhir file

BAB 3

SEKUEN

3.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan bentuk kondisional : pernyataan logika dan tabel kebenarannya

Pernyataan if, if-else, dan if-else if, if bersarang, mengubah arus (block statements vs. single line).

3.2. Pendahuluan

Apabila kita mengerjakan sesuatu hal, ada kalanya tata cara mengerjakannya harus berurutan. Misalnya, kita ingin membuat rumah. Urutan membuat pondasi, membuat dinding dan memasang atap tentulah tidak dapat dibalik menjadi membuat dinding, membuat pondasi, dan memasang atap atau memasang atap, membuat pondasi, dan membuat dinding.

Pada sisi lain, ada suatu pekerjaan yang tidak membutuhkan tata urutan tertentu untuk menyelesaikannya. Misalkan kita ingin membuat cocktail. Sudah tersedia pepaya, nanas, bengkoang, sirup dan es. Tidaklah menjadi soal apabila kita sudah mengetahui proporsi dari setiap komponen, urutannya tidaklah harus pada aturan tertentu.

Sekuen (*sequence*) adalah sederetan pernyataan-pernyataan yang urutan dan pelaksanaan eksekusinya runtut, yang lebih dahulu ditemukan (dibaca) akan dikerjakan (dieksekusi) lebih dulu. Bila urutan pernyataan dibalik, akan mempunyai makna yang berbeda.

3.3 Operator Aritmetika

Rumus-rumus aljabar biasanya melibatkan operator aritmetika, di antaranya :

Aritmetika	Arti	C++	Sifat
+	penjumlahan	+	komutatif
-	pengurangan	-	non komutatif

*	perkalian	*	komutatif
/	pembagian	/	non komutatif
div	pembagian integer	/	non komutatif
modulo	sisanya pembagian	%	non komutatif

Operator yang mempunyai sifat komutatif, urutan pengerjaannya bisa dilakukan sebarang, sedangkan operator yang mempunyai sifat non komutatif urutan pengerjaannya harus diperhatikan. Khusus operator /, apabila kedua operan bertipe int, maknanya sebagai pembagian integer. Apabila salah satu (atau kedua) operan bertipe float, maknanya sebagai pembagian biasa.

Sebagai contoh :

```
int a = 7, b = 2;
float c = 2.0;
cout << a/b;      // hasilnya 3
cout << a/c;      // hasilnya 3.5
```

3.4 Pemrograman

Dalam bahasa C++, pernyataan yang berkaitan dengan operasi dasar adalah sebagai berikut :

Pernyataan	Algoritmik	Bahasa C++
input	Read	cin >>
output	Write	cout <<
penugasan	←	=
akhir pernyataan		;

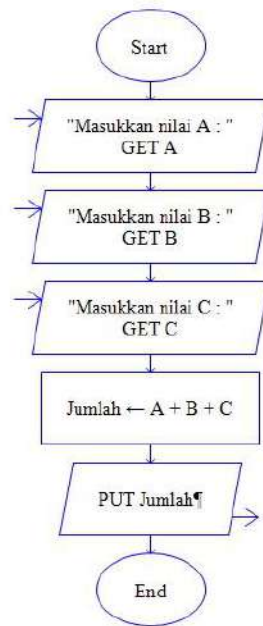
Kasus 3.1.

Hitunglah jumlah dari 3 buah bilangan bulat!

Analisis :

Untuk menjumlahkan 3 buah bilangan bulat, operator yang tepat adalah operator penjumlahan +. Operator ini mempunyai sifat komutatif sehingga kita tidak perlu memperhatikan tata cara pengerjaannya.

Algoritma mencari jumlah 3 bilangan bulat {menghitung jumlah 3 bilangan bulat, algoritma menerima masukan 3 buah bilangan bulat, menjumlahkan, lalu mencetak hasil penjumlahannya}
Deklarasi a, b, c : integer {input} jumlah : integer {output}
Deskripsi read(a, b, c) jumlah ← a + b + c

Flowchart 3.1**Translasi 3.1**

	Bahasa C++
1.	#include <iostream.h>
2.	main() {
3.	int a,b,c,jumlah;
4.	cout << "Bilangan 1 : "; cin >> a;
5.	cout << "Bilangan 2 : "; cin >> b;
6.	cout << "Bilangan 3 : "; cin >> c;
7.	jumlah = (a + b + c);
8.	cout << "Jumlah 3 bilangan = " << jumlah;
9.	return 0;
10.	}

3.5 Mengkonversi ke Model OOP

Tahapan untuk membuat model langsung menjadi model OOP adalah sebagai berikut :

1. mempersiapkan kelas yang cocok, termasuk nama yang sesuai dengan masalah
2. mempersiapkan konstruktor dari kelas, terutama apabila dikehendaki adanya :
 - a. komentar/ informasi awal
 - b. nilai awal dari data
3. menentukan variabel yang terlibat meliputi : variabel input, variabel proses dan variabel output. Semua variabel ini diletakkan pada bagian private dari kelas

4. mempersiapkan dialog input dan operator overloadingnya
5. mempersiapkan model informasi (output) yang dikehendaki dan operator overloadingnya
6. menentukan fungsi (methods) untuk memanipulasi variabel input
7. menyediakan implementasi dari kelas (membuat main function)

Langkah 1 sampai dengan 3 dapat dirangkum dengan membuat prototipe kelas dari masalah menjumlahkan 3 bilangan bulat berikut ini :

class Hitung {
friend ostream& operator<<(ostream&, const Hitung&); //prototipe operator overloading output
friend istream& operator>>(istream&, Hitung&); //prototipe operator overloading input
public:
Hitung(); // konstruktor
void hitung_jumlahnya(); // prototipe methods { jumlah = (a + b + c); }
private:
int a,b,c; // variabel input
int jumlah; // variabel proses / output
};

Operator overloading input maupun output hampir dapat dikatakan baku. Artinya bentuknya serupa, hanya tergantung dari nama kelas. Bagian ini khusus dibuat sebagai friend oleh karena dikehendaki dapat mengakses bagian private melalui objek, yaitu memberi nilai atau Mencetak nilai.

Konstruktor merupakan methods yang khas karena namanya sama dengan nama kelas. Isinya dapat kita tentukan sesuai dengan kebutuhan.

Bagian private umumnya berisi data. Bagian ini dikatakan private karena tidak dapat diakses oleh bagian di luar kelas. Ini merupakan salah satu ciri OOP yaitu information hiding. Pembungkusan bagian data dan fungsi yang memanipulasi data dalam satu kelas juga merupakan cirri dari OOP yaitu pengkapsulan (encapsulation). Methods diletakkan pada bagian public oleh karena merupakan bagian yang digunakan sebagai antar muka dengan kelas atau fungsi di luar kelas.

2. mempersiapkan dialog input dan operator overloadingnya

Misalkan diinginkan dialog input :

Bilangan 1 :	<input type="text"/>
Bilangan 2 :	<input type="text"/>
Bilangan 3 :	<input type="text"/>

Dengan tanda menyatakan tempat untu memasukkan data dari keyboard. Dialog di atas dapat diimplementasikan :

istream& operator>>(istream& in, Hitung& masukan) {
cout << "Masukkan nilai a : ";
in >> masukan.a;
cout << "Masukkan nilai b : ";
in >> masukan.b;
cout << "Masukkan nilai c : ";
in >> masukan.c;
return in;
}

Variabel in saat digunakan nilainya akan diberikan oleh operator >> pada variabel cin, misalnya digunakan oleh main function :

main() {
Hitung X;
cin >> X; // cin menerima nilai dari variabel in pada friend operator overloading input
...
}

mempersiapkan model informasi (output) yang dikehendaki dan operator overloadingnya

Nilai a : ●●●
Nilai b : ●●●
Nilai c : ●●●
Jumlah 3 integer di atas : ●●●

Dengan tanda ●●● berarti informasi yang diberikan program kepada pengguna. Kita dapat merancang model output sebagai berikut :

ostream& operator<<(ostream& out, const Hitung& keluaran) {
out << "Nilai a : " << keluaran.a << endl;
out << "Nilai b : " << keluaran.b << endl;
out << "Nilai c : " << keluaran.c << endl;
out << "Jumlah 3 integer di atas : " << keluaran.jumlah << endl;
return out;
}

Variabel out saat digunakan (misalnya pada main function) nilainya akan diberikan oleh operator << pada variabel cout.

main() {
Hitung X;
...
cout << X; // cout menerima nilai dari variabel out pada friend operator overloading output
...
}

8. menentukan fungsi (methods) untuk memanipulasi variabel input

Kita telah menyiapkan prototipe methods dalam kelas yang telah dibuat sebelumnya dengan pernyataan :

```
void hitung_jumlahnya();
```

Implementasi dari prototipe methods di atas dengan cara menyisipkan nama kelas di antara return type fungsi dan nama fungsi diikuti operator scope ::.

```
void Hitung::hitung_jumlahnya() {  
    // isi fungsi  
}
```

Fungsi selengkapnya disajikan berikut ini :

void Hitung ::hitung_jumlahnya() {
jumlah = (a + b + c); // isi fungsi
}

Kalau diperhatikan, kita tidak perlu lagi menggunakan parameter untuk mengakses bagian private data. Secara umum dapat disimpulkan bahwa bagian private dari kelas bersifat global (dapat diakses) oleh anggota kelas yang lain.

9. menyediakan implementasi dari kelas (membuat main function)

main function biasanya digunakan untuk menguji penggunaan kelas. Perancangan fungsi main dapat dilakukan dengan model top down.

1. Buat objek dari kelas Hitung
2. masukkan data yang diperlukan
3. proses data masukan
4. keluarkan hasilnya

Untuk kelas Hitung main functionnya adalah sebagai berikut :

main() {
Hitung X; // langkah 1
cin >> X; // langkah 2
X.hitung_jumlahnya(); // langkah 3
cout << X; // langkah 4
getch();
return 0;
}

Apabila dalam suatu fungsi tidak ada return typenya, maka kompilator akan menganggap bertipe int. Sehingga diakhir main function terdapat pernyataan :

```
return 0;
```

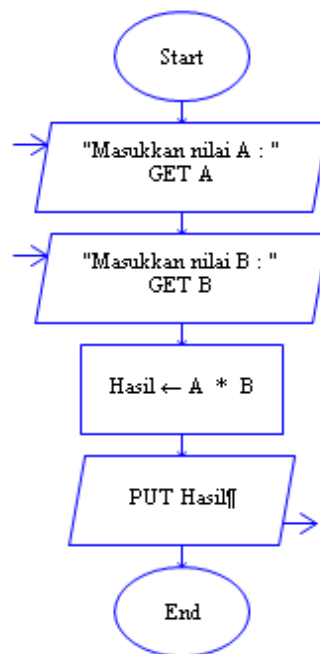

yang menyatakan program berakhir sukses (tidak ada kesalahan). Fungsi getch() digunakan untuk menahan laju program sementara sampai pengguna menekan ENTER. Fungsi getch() dipanggil dengan menyertakan file conio.h.

Kasus 3.2.

Buat algoritma dan program untuk mencari hasil kali dari dua buah bilangan !

Algoritma hasil_kali { mencari hasil kali dari dua buah bilangan bulat, algoritma menerima masukan nilai bilangan1 dan bilangan2, lalu mengalikan kedua bilangan tersebut, dan mencetak hasil kalinya }
Deklarasi a,b : integer (input) hasil : integer (output)
Deskripsi read(a,b) hasil \leftarrow a*b write(hasil)

Flowchart 3.2.



Translasi 3.2.

Bahasa C++
<pre> #include <iostream.h> #include <conio.h> class Perkalian { friend ostream& operator<<(ostream&, const Perkalian&); friend istream& operator>>(istream&, Perkalian&); public: </pre>

```

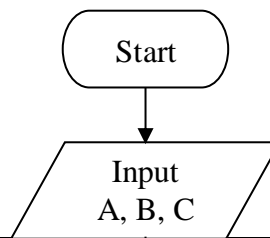
Perkalian();
void hitung_perkaliannya(){ hasil = (a * b); }
private:
    int a,b;
    int hasil;
};
Perkalian::Perkalian() {
    cout << "Program mengalikan 2 integer\n";
    cout << "Selamat berkarya\n";
}

istream& operator>>(istream& in, Perkalian& masukan) {
    cout << "Masukkan nilai a : ";
    in >> masukan.a;
    cout << "Masukkan nilai b : ";
    in >> masukan.b;
    return in;
}

ostream& operator<<(ostream& out, const Perkalian& keluaran) {
    out << "Nilai a : " << keluaran.a << endl;
    out << "Nilai b : " << keluaran.b << endl;
    out << "Hasil kali integer di atas : " << keluaran.hasil << endl;
    return out;
}

main() {
    Perkalian X;
    cin >> X;
    X.hitung_perkaliannya();
    cout << X;
    getch();
    return 0;
}

```



Oleh karena hanya satu pernyataan, maka pernyataan ini tidak perlu dibuat terpisah dari kelasnya dan dibuat inline.

Kasus 3.3.

Buat algoritma dan program untuk mengkonversi dari meter ke cm dan inchi !

Analisis :

Konversi satuan jarak ditentukan sebagai berikut

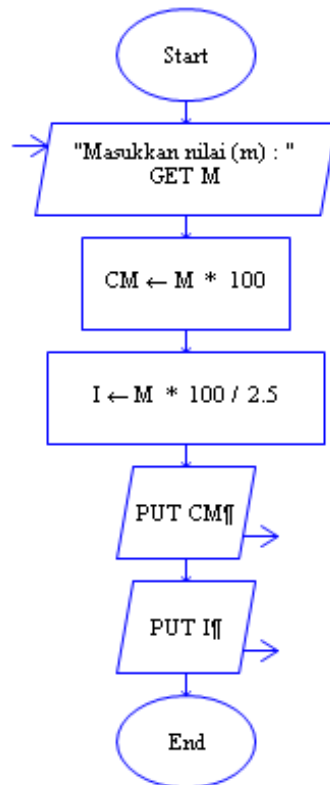
1 meter = 100 cm

1 inci = 2.54 cm

Algoritma 3.3.

Algoritma konversi jarak { mengkonversi mulai jarak dari meter ke cm dari inchi, masukan m ke cm, lalu masukkan ke inchi }
Deklarasi m : real(input) cm : real(output) inci : real(output)
Deskripsi read(m) cm \leftarrow m *100 inci \leftarrow m*100/ 2.54 write(cm) write(inci)

Flowchart 3.3.



Translasi 3.3.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Konversi {</pre>

```

    friend ostream& operator<<(ostream&, const Konversi&);
    friend istream& operator>>(istream&, Konversi&);
public:
    Konversi();
    void konversi_ke_cm(){ cm = (m * 100); }
    void konversi_ke_inci(){ inci = cm/2.54; }
private:
    float m;
    float cm, inci;
};

Konversi::Konversi() {
    cout << "Program konversi m ke cm dan inci\n";
    cout << "Selamat berkarya\n";
}

istream& operator>>(istream& in, Konversi& masukan) {
    cout << "Masukkan nilai m : ";
    in >> masukan.m;
    masukan.konversi_ke_cm();
    masukan.konversi_ke_inci();
    return in;
}

ostream& operator<<(ostream& out, const Konversi& keluaran) {
    out << "Nilai m : " << keluaran.m << endl;
    out << keluaran.m << " m = " << keluaran.cm << " cm " << endl;
    out << keluaran.m << " m = " << keluaran.inci << " inci " << endl;
    return out;
}

void main() {
    Konversi X;
    cin >> X;
    cout << X;
    getch();
}

```

Kasus 3.4.

Carilah keliling dan luas lingkaran yang telah diketahui jari-jarinya.

Analisis :

Untuk mencari keliling dan luas lingkaran digunakan rumus :

$$\text{keliling} = 2 \Pi r$$

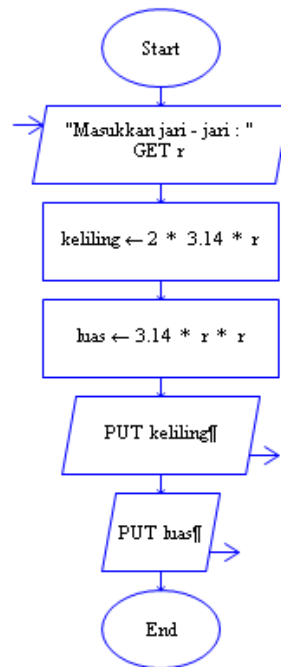
$$\text{luas} = \Pi r^2$$

dengan $\Pi = 3.14$ dan r adalah jari-jari lingkaran.

Algoritma 3.4.

Algoritma Lingkaran { Mencari keliling dan luas lingkaran yang telah diketahui jari-jarinya (misal r) }.
Deklarasi Konstanta $\text{phi} = 3.14$ r : integer { input } keliling, luas : real { output }
Deskripsi read(r) keliling $\leftarrow 2 * \text{phi} * r$ luas $\leftarrow \text{phi} * r * r$ write(keliling, luas)

Flowchart 3.4.



Translasi 3.4.

Bahasa C++
<pre> #include <iostream.h> void main() { float phi=3.14; int r; float keliling, luas; cout << "Jari-jari lingkaran : "; cin >> r; keliling = 2*phi*r; luas = phi*r*r; </pre>

```
cout << "Keliling = " << keliling << endl;
cout << "Luas    = " << luas << endl;
}
```

Kasus 3.5.

Carilah konversi suhu dari Celcius menjadi Reamur, Fahrenheit dan Kelvin.

Analisis :

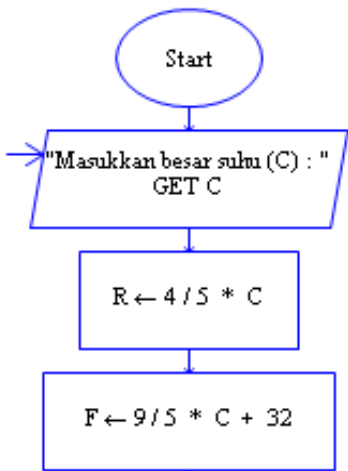
Rumus konversi dari Celcius menjadi Reamur, Fahrenheit dan Kelvin adalah sebagai berikut :

$$reamur = \frac{4}{5} celcius$$
$$fahrenheit = \frac{9}{5} celcius + 32$$
$$kelvin = celcius + 273$$

Algoritma 3.5.

Algoritma Menghitung_Konversi_Suhu { Membaca integer derajat Celcius (C), menghitung ekivalensinya dalam derajat Reamur (R), Fahrenheit (F), dan Kelvin (K) dan menampilkannya di layar }.		
Deklarasi		
C	: integer	{ derajat Celcius }
R	: real	{ derajat Reamur }
F	: real	{ derajat Fahrenheit }
K	: real	{ derajat Kelvin }
Deskripsi		
read (C)		
R ← 4/5 * C		{ Rumus R = 4/5 * C }
F ← 9/8 * C + 32		{ Rumus F = 9/5 * C + 32 }
K ← C + 273		{ Rumus K = C + 273 }
write (R,F,K)		

Flowchart 3.5.



Translasi 3.5.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Suhu { friend ostream& operator<<(ostream&, Suhu&); friend istream& operator>>(istream&, Suhu&); public: void konversi_keR() { r = 4/5.0 * c; } void konversi_keF() { f = 9/5.0 * c + 32; } void konversi_keK() { k = c + 273; } private: int c; // variabel input float r, f, k; // variabel output }; istream& operator>>(istream& in, Suhu& A) { cout << "Masukkan suhu derajat celcius : "; in >> A.c; return in; } ostream& operator<<(ostream& out, Suhu& A) {</pre>

```

A.konversi_keR();
A.konversi_keF();
A.konversi_keK();
cout << A.c << " celcius = " << A.r << " reamur" << endl;
cout << A.c << " celcius = " << A.f << " fahrenheit" << endl;
cout << A.c << " celcius = " << A.k << " kelvin" << endl;
return out;
}
void main() {
    Suhu X;
    cin >> X;
    cout << X;
}

```

Kasus 3.6.

Setiap bilangan bulat selalu bisa ditulis dalam bentuk :

$$m = q.n + r$$

dengan $n < m$, q adalah kuosen dan r adalah residu (sis). Buatlah algoritma untuk merepresentasikan m dan dalam bentuk $m = q.n + r$. Sebagai contoh :

integer $m = 73$ dan $n = 7$ dapat ditulis sebagai $73 = 10 \times 7 + 3$, yaitu $r = 3$.

Analisis :

Input : n dan m ($n < m$),

Kuosen q dapat diperoleh dengan pembagian integer. Operator yang tepat untuk itu adalah div, sedangkan r dapat diperoleh dengan menggunakan operasi modulo.

Output q dan r dalam bentuk $m = q.n + r$.

Algoritma 3.6.

Algoritma Aljabar

{membaca masukan nilai n dan m dengan $n < m$ kemudian menuliskan outputnya berbentuk $m = q.n + r$ }

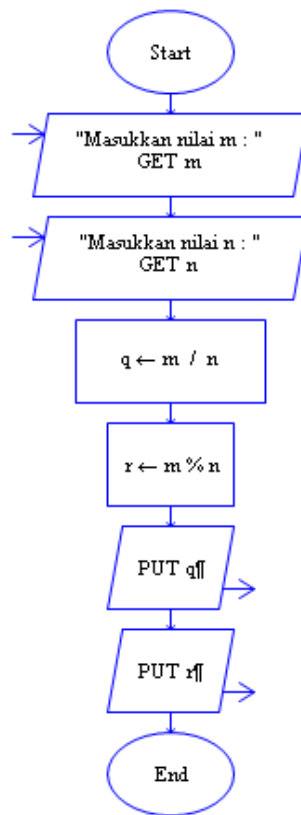
Deklarasi

m, n	: integer	{ input }
q, r	: integer	{ output }

Deskripsi

read (m, n)	($n < m$)
$q \leftarrow m \text{ div } n$	{ mendapatkan nilai q }
$r \leftarrow m \text{ mod } n$	{ mendapatkan nilai r }
write (q, r)	

Flow Chart 3.6.



Translasi 3.6.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Aljabar { friend ostream& operator<<(ostream&, const Aljabar&); friend istream& operator>>(istream&, Aljabar&); public: Aljabar(); void hitung(){ q = m / n; // mendapatkan nilai q r = m % n; // mendapatkan nilai r } private: int m,n; // input int q,r; // output }; Aljabar::Aljabar() { cout << "Membaca input nilai n dan m dengan ketentuan n<m\n"; cout << "dan menampilkan output berbentuk m = q.n + r.\n\n"; } istream& operator>>(istream& in, Aljabar& masukan) {</pre>

```

    cout << "Masukkan nilai m = ";
    in >> masukan.m;
    cout << "Masukkan nilai n = ";
    in >> masukan.n;
    masukan.hitung();
    return in;
}

ostream& operator<<(ostream& out, const Aljabar& keluaran) {
    out << "Nilai q adalah = " << keluaran.q << endl;
    out << "Nilai r adalah = " << keluaran.r << endl << endl;
    out << "Jadi, " << keluaran.m << " = " << keluaran.q << " x ";
    out << keluaran.n << " + " << keluaran.r;
    return out;
}

void main() {
    Aljabar X;
    cin >> X;
    cout << X;
}

```

Algoritma pada dasarnya tidak tergantung pada suatu bahasa pemrograman tertentu. Secara sederhana, bila menguasai suatu bahasa pemrograman tertentu, konversi ke dalam bahasa pemrograman yang lain hanyalah menyesuaikan dengan “dialek” (aturan-aturan) bahasa yang bersangkutan. Dengan demikian, yang terpenting adalah bagaimana mengkonstruksikan algoritma yang benar.

Kasus 3.7.

Buatlah algoritma untuk menghitung Luas Segitiga.

Analisis :

Input : t (tinggi) dan a (alas)

$$luas = \frac{l * a}{2}$$

Algoritma 3.7.

Algoritma Menghitung_Luas_Segitiga.

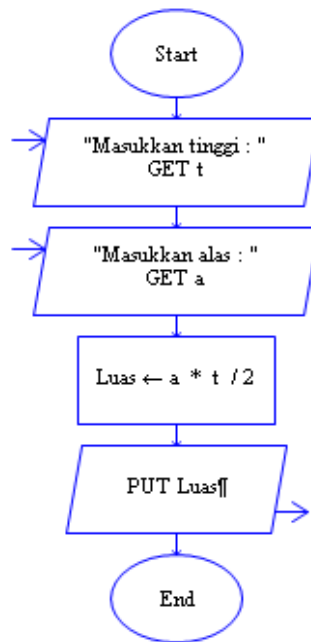
{Membaca data real berupa panjang alas segitiga (a) dan tinggi segitiga (t) dan menghitung luasnya dengan panjang alas dan tinggi tertentu. Luas segitiga dihitung dengan rumus $L = \frac{1}{2} at$. Nilai L dicetak sebagai output ke piranti keluaran.}

Deklarasi

a	: real	{ panjang alas segitiga, dalam satuan cm }
t	: real	{ tinggi segitiga, dalam satuan cm }
L	: real	{ luas segitiga, dalam satuan cm^2 }

Deskripsi

read (t)
read (a)
 $L \leftarrow (a*t)/2$
write (L)

Flowchart 3.7.**Translasi 2.7.****Bahasa C++**

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

class Segitiga {
    friend ostream& operator<<(ostream&, Segitiga&);
    friend istream& operator>>(istream&, Segitiga&);
public:
    void hitung_luas() { luas = a * t / 2.0 ; }
private:
    int a, t ;    // variabel input
    float luas;  // variabel output
};

istream& operator>>(istream& in, Lingkaran& X) {
    cout << "Menghitung luas segitiga" << endl ;
    cout << "Masukkan panjang alas segitiga : " ;
    in >> X.a;
    cout << "Masukkan tinggi segitiga : " ;
    in >> X.t;
    return in;
}
```

```

}
ostream& operator<<(ostream& out, Lingkaran& Y) {
    Y.hitung_luas();
    out << "Luas segitiga = " << Y.luas << endl;
    return out;
}
void main() {
    Segitiga D;
    cin >> D;
    cout << D;
}

```

Output :

Menghitung Luas Segitiga.
 Masukkan panjang alas segitiga : 25 cm
 Masukkan tinggi segitiga : 12 cm
 Luas segitiga = 150.00 cm²

Pada pernyataan

$$L = (a*t)/2.0;$$

pembagian dengan 2.0 (perhatikan tanda titik nol setelah angka 2) “memaksa” hasil evaluasi $(a*t)/2.0$ akan bertipe float, oleh karena kita telah mendeklarasikan L bertipe float. Hal ini perlu karena bahasa C tidak secara otomatis menjadikan hasil evaluasinya bertipe float. Bila kita tidak membagi dengan 2.0 melainkan dengan 2, karena a dan t masing-masing bertipe int, maka hasil evaluasi keseluruhan juga bertipe int. Kemudian baru hasil ini dikonversi ke tipe float dalam L. Dalam bahasa C, hasil evaluasi $15/2$ harusnya menghasilkan 7.5, tetapi karena bertipe int, maka nilai desimal akan “dipotong”, sehingga hasilnya menjadi 7. Lalu bila disimpan pada variabel bertipe float akan dikonversi menjadi 7.0. Jadi akan terdapat kesalahan. Pemrograman bahasa C akan “waspada” terhadap kasus seperti ini.

Workshop algoritma dan class

1. Buatlah analisis dan algoritma untuk mencari titik tengah sebuah garis yang ujung titiknya adalah A(x1,y1) dan B(x2,y2).

Gambar lebih dulu, di sini contohnya !

Tulis rumus jarak 2 titik :

Formulasikan algoritmanya :

2. Buatlah algoritma untuk mencari isi bola bila diketahui jari-jari bola.

Input :

Rumus isi bola :

Algoritma :

3. Buatlah analisis dan algoritma untuk mencari hipotenusa dari segitiga Pythagoras bila diketahui sisi siku-sikunya. (Petunjuk : gunakan fungsi **sqrt** yang menyatakan **akar dari**).

Tulis rumusnya lebih dulu :

4. Buatlah analisis dan algoritma untuk menghitung konversi detik menjadi format jam:menit:detik

Buat contoh kasus, misalnya konversikan 1230 detik menjadi ... jam ... menit ... detik.

Rumuskan algoritmanya :

5. Buatlah analisis dan algoritma untuk menghitung konversi dari jam:menit:detik ke detik
Buat contoh kasus, misalnya konversikan 2 jam 13 menit 43 detik menjadi ... detik.

Rumus yang diperlukan :

Rumuskan algoritmanya :

6. Buatlah analisis dan algoritma untuk menghitung selisih 2 waktu. Output ditampilkan dalam bentuk jam:menit:detik. Asumsikan menggunakan sistem jam 24-an.
Buat contoh kasus, misalnya berapa selisih waktu antara jam 3.45 sore dengan jam 10.23 malam. Selesaikan secara manual lebih dulu.

Rumus yang digunakan, tulis di sini.

Rumuskan algoritmanya :

- .
7. Buatlah analisis dan algoritma untuk menghitung jumlah komponen sejumlah uang menjadi pecahan-pecahannya. Misalkan Rp 188.875,- menjadi : 1 seratus ribuan, 1 lima puluh ribuan, 1 dua puluh ribuan, 1 puluhan ribu, 1 lima ribuan, 3 ribuan, 1 lima ratusan, 1 dua ratusan, 1 ratusan, 1 lima puluhan, 1 dua puluh lima an
- Rumus yang digunakan :

Rumuskan algoritmanya :

8. [Pengamatan] Buatlah analisis dan algoritma untuk menghitung luas sebuah plat CD.

Diameter lingkaran luar = cm.

Diameter lingkaran dalam = cm.

Rumus luas plat CD :

Buat algoritmanya :

9. Buatlah analisis dan algoritma untuk menghitung operasi bilangan rasional :
penjumlahan, pengurangan, perkalian, pembagian dan kebalikan.

Buat contoh kasus, misalnya dua bilangan rasional $\frac{2}{3}$ dan $\frac{4}{7}$.

Hasil penjumlahan manual =

Hasil pengurangan manual =

Hasil perkalian manual =

Hasil pembagian manual =

Hasil kebalikan manual =

Rumus yang digunakan, tulis di sini :

10. Buatlah analisis dan algoritma untuk mengkonversi bilangan biner 4 digit menjadi bilangan desimal. Operator overloading input mencegah user untuk memasukkan bilangan selain 0 dan 1. Deklarasi variabel input dan output adalah bertipe integer.
- Konversikan 1011 ke desimal :

Buat algoritmanya :

11. Buatlah analisis dan algoritma untuk menjumlah dua bilangan scientific berbentuk aEb dengan $0 < a < 10$ dan $0 \leq b \leq 4$.
- Hitung $2.1E2 + 3.31E1 =$ secara manual lebih dulu.

Buat algoritmanya :

12. Buatlah analisis dan algoritma untuk menghitung jumlah dari dua tanggal yang dimasukkan user. Misalnya, berapa hari dari 6/1/90 sampai 8/3/92 ? Anggap satu tahun 365 hari dan anggap semua komponen tanggal pertama **SELALU LEBIH KECIL** dari komponen tanggal kedua.

Hitung secara manual kasus dalam soal.

Rumus yang digunakan, tulis di sini :

Buat algoritmanya :

BAB 4

PEMILIHAN/ KONTROL PROGRAM

4.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan berbagai algoritma yang melibatkan pernyataan logika dan menentukan tabel kebenarannya dan menggunakan untuk menentukan pengambilan keputusan.

4.2 Pendahuluan

Dalam persoalan sehari-hari, hampir selalu ada kondisi di mana kita harus memilih, di antara alternatif-alternatif yang ada. Misalkan kita berada pada sebuah super market. Kita berniat ingin membeli sepatu. Sebelum berangkat kita mempersiapkan sejumlah uang. Untuk membeli sepatu ada sejumlah pertimbangan, misalnya jenis sepatu : sepatu olah raga atau sepatu kulit, warna : hitam atau coklat, harga : mulai dari 30.000 sampai maksimal 300.000 (dana maksimal untuk beli sepatu), dan selera (model). Dari sekian banyak kriteria kita akan membuat prioritas. Untuk mengambil keputusan, akan selalu dibandingkan dengan kondisi yang ada, cocok atau tidak. Bila cocok, akan diteruskan pada pertimbangan berikutnya. Sampai akhirnya, apabila semua kondisi yang cocok sudah terpenuhi, sepatu yang diinginkan akan dibeli.

Sebagai contoh, seorang mahasiswa memperoleh nilai 75. Apakah dengan nilai tersebut mahasiswa itu lulus ? Jika batas kelulusan minimal 60 maka mahasiswa tersebut lulus ujian. Jika kurang dari 60 maka mahasiswa tersebut tidak lulus ujian.

Salah satu kemampuan komputer adalah dapat melakukan proses pemilihan dari beberapa alternatif sesuai dengan kondisi yang diberikan.

4.3 Ekspresi Relasional

Biasanya suatu kondisi dinyatakan dengan membandingkan suatu keadaan dengan suatu nilai. Untuk itu diperlukan suatu operator perbandingan, yaitu :

Algoritmik	Arti	C++
<	lebih kecil	<
>	lebih besar	>
<=	lebih kecil sama dengan	<=
>=	lebih besar sama dengan	>=
≠	tidak sama dengan	!=
=	sama dengan	=
or	atau	
and	dan	&&

Apabila ada 2 operan atau lebih yang dihubungkan dengan operator pembandingan dan mempunyai makna tunggal, penulisan ini dinamakan ekspresi relasional. Ekspresi relasional hanya mempunyai 2 alternatif nilai yaitu benar atau salah. Bila kondisi sesuai, ekspresi bernilai benar, bila tidak, ekspresi bernilai salah.

Contoh 1 :

Menentukan air sudah mendidih atau belum bila diketahui suhu air saat diuji.

Jika suhu < 100 derajat C maka kesimpulannya air belum mendidih else air sudah mendidih

Contoh 2 :

Menentukan kelulusan siswa. Jika syarat lulus nilai harus lebih dari 60 maka dapat dibuat ekspresi :

Jika nilai < 60 maka siswa tidak lulus

Else siswa lulus.

Ada kalanya, kita akan menentukan keputusan pada kondisi dengan range tertentu.

Contoh 3 :

Misalkan akan dibeli sepatu bila harga sepatu antara 75.00 sampai 200.000. Jika Ternyata sepatu yang ditawarkan seharga 350.000 maka kita tidak jadi beli.

Ekspresi majemuk, alternatif ganda

Contoh 4 :

Menentukan keadaan siang atau malam bila diketahui jam saat ini.

Jika jam antara 0 sampai 4 dikatakan dini hari

Jam 4 sampai 12 dikatakan pagi hari

Jam 12 sampai 15 dikatakan siang hari

Jam 15 sampai 18 dikatakan sore hari

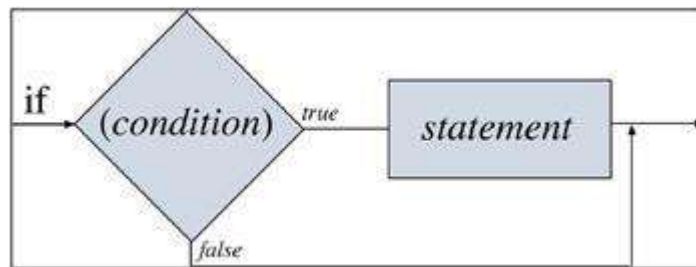
Jam 18 sampai 24 dikatakan malam hari

Pemrograman

Bahasa pemrograman C++ juga mendukung pernyataan bersyarat. Tata cara penulisannya tersaji berikut ini :

Algoritmik	C++
if (kondisi) then aksi end if	if (kondisi) aksi;
if (kondisi) then aksi1 else aksi2 end if	if (kondisi) aksi1; else aksi2;
pernyataan majemuk	{ ... }

Dalam flowchart, pernyataan if di atas dapat digambarkan sebagai :

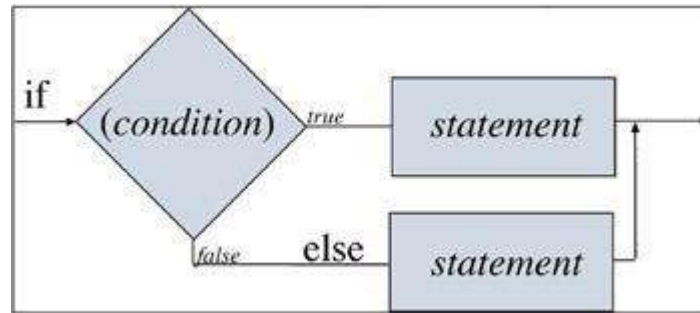


Pada pernyataan :

```
if (kondisi) then  
aksi  
end if
```

aksi merupakan statement yang akan dikerjakan bila kondisi true. Bila kondisi bernilai false maka tidak ada satupun pernyataan dalam if tersebut yang akan dieksekusi.

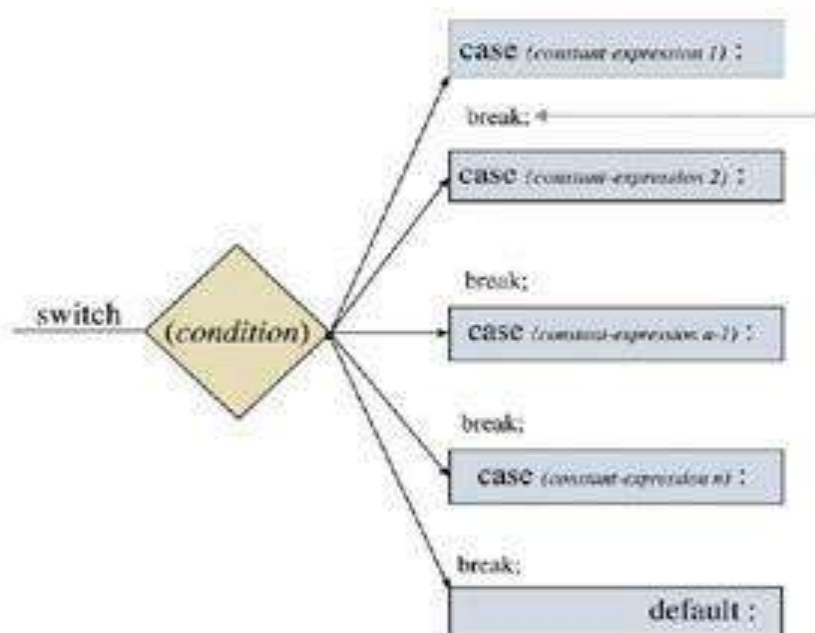
Dalam flowchart, pernyataan if – else di atas dapat digambarkan sebagai :



Aksi1 akan dikerjakan bila kondisi true, sedangkan aksi2 akan dikerjakan bila kondisi false. Baik aksi, aksi1 maupun aksi2 bisa merupakan pernyataan tunggal maupun pernyataan majemuk. Selain itu, bila kondisi bertipe ordinal, seperti integer, byte atau yang lain (kecuali real atau string), bisa digunakan pernyataan di bawah ini :

Algoritmik	C++
<pre> case (nama) <label1> : aksi1 < label2> : aksi2 < label3> : aksi3 ... < labelN> : aksiN else : aksiX endcase </pre>	<pre> switch (nama) { case label1 : aksi1; break; case label2 : aksi2; break; case label3 : aksi3; break; ... case labelN : aksiN; break; default : aksix; } </pre>

Dalam flowchart, pernyataan switch di atas dapat digambarkan sebagai :



di mana konstanta1 sampai dengan konstantaN dapat pula berupa range suatu nilai, misalnya untuk menyatakan $60 < \text{nilai} \leq 80$ dapat ditulis sebagai 61..80 (dengan menggunakan operator range ..)

4.4 Operator Relasional

Untuk membandingkan dua keadaan diperlukan operator pembandingan. Dalam C++, operator relasional adalah :

Operator	Arti
>	Lebih besar dari
>=	Lebih besar atau sama dengan
<	Lebih kecil dari
<=	Lebih kecil atau sama dengan
=	Sama dengan
!=	Tidak sama dengan

4.5 Operator Logika

Operator or dan and termasuk operator binari, yaitu operator yang memerlukan dua operan, di mana setiap operan bisa bernilai benar atau salah. Untuk itu masing-masing mempunyai 4 kemungkinan nilai, yaitu :

p	Q	p or q
benar	Benar	benar
benar	Salah	benar
salah	benar	benar
salah	salah	salah

p	q	p and q
benar	benar	benar
benar	salah	salah
salah	benar	salah
salah	salah	salah

Operator not atau ingkaran dalam C++ adalah !. Lihat contoh pemrograman berikut ini :

```
int a = 0;
if(!a)
    cout<<"Pernyataan benar."<<endl;
else
    cout<<" Pernyataan salah."<<endl;
```

Dalam C++, nilai false diwakili oleh integer , sementara nilai true diwakili nilai selain nilai 0.

Kasus 4.1.

Tentukanlah bilangan terbesar antara dua bilangan bulat.

Algoritma 4.1.

Algoritma Maksimum

{ Menentukan nilai terbesar antara dua bilangan bulat }

Deklarasi

A, B : integer

Deskripsi

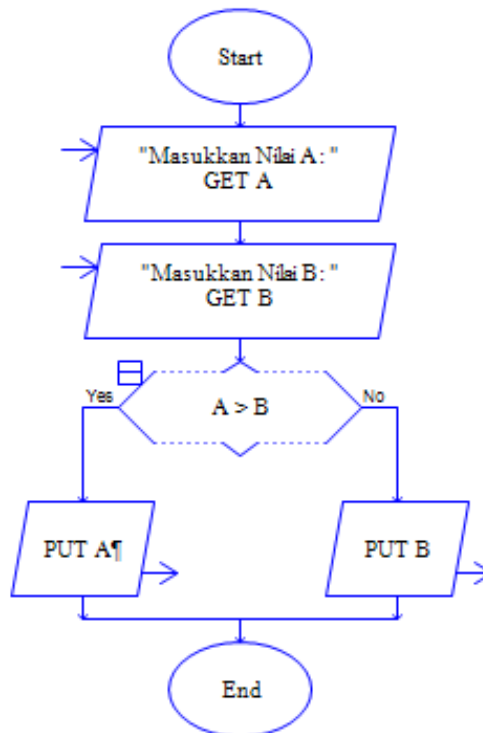
read (A, B)

if (A > B) then write ('Bilangan terbesar adalah = 'A)

else write ('Bilangan terbesar adalah = 'B)

endif

Flow Chart 4.1.



Translasi 4.1.

Bahasa C++

```
#include <iostream.h>
#include <conio.h>
class Banding {
    friend istream& operator>>(istream&, Banding&);
public:
    Banding() {};
    void bandingkan() {
        if (A > B)
            cout << "Bilangan terbesar : " << A;
        else
            cout << "Bilangan terbesar : " << B;
    }
private:
    int A, B;
};

istream& operator>>(istream& in, Banding& bilangan){
    cout << "Bilangan pertama = ";
    in >> bilangan.A;
    cout << "Bilangan kedua = ";
    in >> bilangan.B;
    return in;
};

main() {
    Banding bilangan;
    cin >> bilangan;
    bilangan.bandingkan();
    getch();
    return 0;
}
```

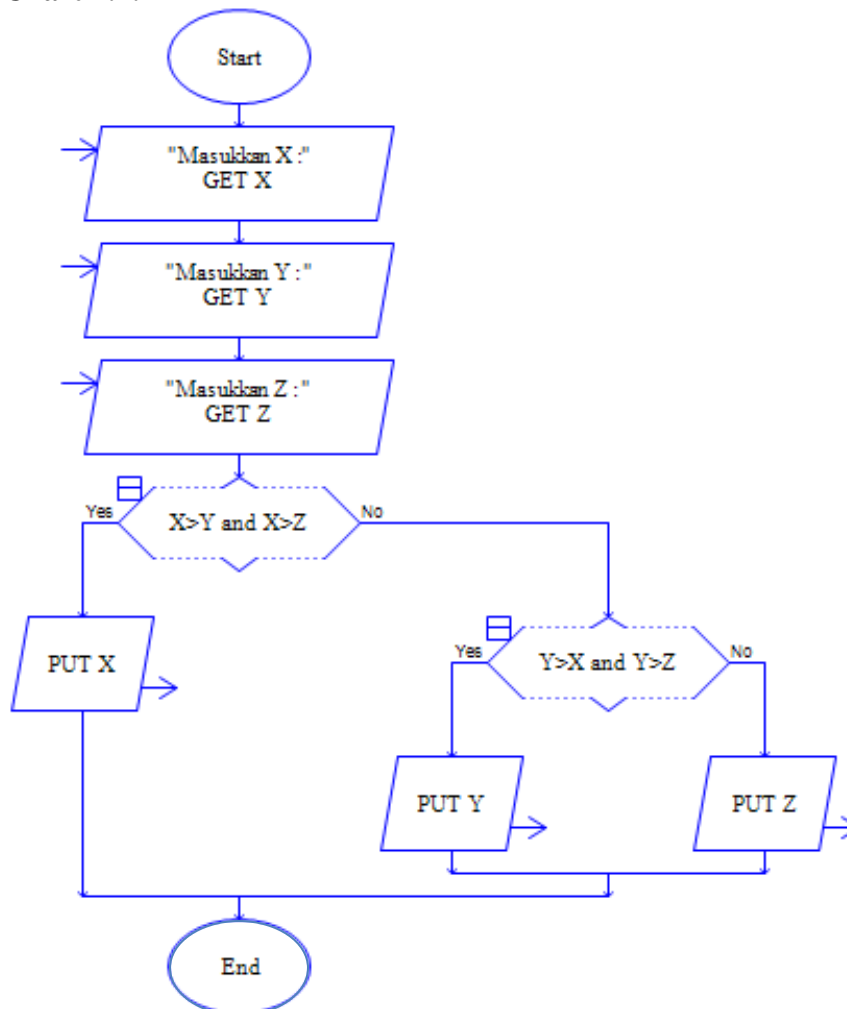
Kasus 4.2.

Tentukanlah bilangan terbesar antara 3 bilangan bulat.

Algoritma 4.2.a.

Algoritma Nilai_ Maksimum { Membaca tiga buah bilangan bulat, menentukan bilangan terbesar di antara tiga buah bilangan tersebut dan menampilkannya ke layar }
Deklarasi x, y, z : integer
Deskripsi read (x, y, z) if (x > y) and (x > z) then write ('Bilangan terbesar adalah = ',x) else if (y > x) and (y > z) then write ('Bilangan terbesar adalah = ',y) else write ('Bilangan terbesar adalah = ',z) end if

Flow Chart 4.2.



Translasi 4.2.a.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Banding { friend istream& operator>>(istream&, Banding&); public: Banding() {}; void bandingkan() { if ((x>y) && (x>z)) cout << "Bilangan terbesar : " << x; else if ((y>x) && (y>z)) cout << "Bilangan terbesar : " << y; else cout << "Bilangan terbesar : " << z; } private: int x, y, z; }; istream& operator>>(istream& in, Banding& bilangan){ cout << "Bilangan pertama = "; in >> bilangan.x; cout << "Bilangan kedua = "; in >> bilangan.y; cout << "Bilangan ketiga = "; cin >> bilangan.z; return in; }; main() { Banding bilangan; cin >> bilangan; bilangan.bandingkan(); getch(); return 0; }</pre>

Pembandingan ini tentu saja menjadi sangat kompleks bila bilangan yang dibandingkan lebih dari 3. Sebagai alternatif, algoritma di bawah ini memberikan penyelesaian lebih baik. Idenya : bila kita punya satu bilangan maka bilangan tersebut pastilah terbesar (atau terkecil), karena memang hanya sebuah bilangan saja. Bilangan berikutnya tinggal dibandingkan dengan nilai terbesar yang saat ini diperoleh.

Algoritma 4.2.b.

Algoritma Nilai_ Maksimum {Membaca tiga buah bilangan bulat, menentukan bilangan terbesar di antara tiga buah bilangan tersebut dan menampilkannya ke layar}
Deklarasi x, y, z : integer maks : integer
Deskripsi <u>read</u> (x, y, z) maks \leftarrow x if (y > maks) then maks \leftarrow y end if if (z > maks) then maks \leftarrow z end if write ('Bilangan terbesar adalah = ',maks)

Translasi 4.2.b.

```
#include <iostream.h>
#include <conio.h>
class Banding {
    friend istream& operator>>(istream&, Banding&);
public:
    Banding() {};
    void bandingkan() {
        int maks = x;
        if (y > maks) maks = y;
        if (z > maks) maks = z;
        cout << "Bilangan terbesar : " << maks;
    }
private:
    int x, y, z; };
istream& operator>>(istream& in, Banding& bilangan){
    cout << "Bilangan pertama = ";
    in >> bilangan.x;
    cout << "Bilangan kedua = ";
    in >> bilangan.y;
    cout << "Bilangan ketiga = ";
    cin >> bilangan.z;
    return in;
};
main() {
    Banding bilangan;
    cin >> bilangan;
    bilangan.bandingkan();
    getch();
    return 0;
}
```

Kasus 4.3.

Carilah akar-akar persamaan kuadrat.

Analisis :

Persamaan kuadrat adalah persamaan dengan bentuk umum $Ax^2 + Bx + c = 0$, dan tentu saja dengan $A \neq 0$. Akar persamaan kuadrat diperoleh dengan rumus :

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Bila $b^2 - 4ac < 0$ akan diperoleh akar imajiner.

Algoritma 4.3.

Algoritma Persamaan_Kuadrat

{Menghitung akar-akar persamaan kuadrat $Ax^2+Bx+C = 0$ }

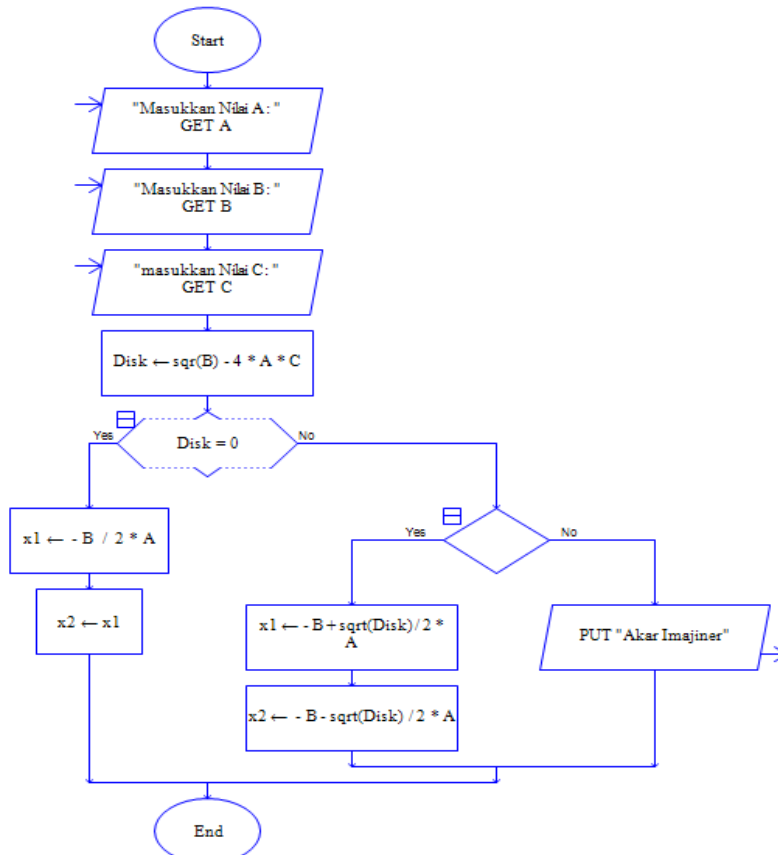
Deklarasi

A, B, C	: integer	{koefisien-koefisien persamaan}
disk	: longint	{nilai diskriminan}
x1, x2	: real	{nilai-nilai akar untuk disk ≥ 0 }

Deskripsi

```
read (A, B, C)
disk  $\leftarrow B*B - 4*A*C$ 
if (A = 0) then write ('Bukan Persamaan Kuadrat')
else if disk > 0 then
    x1 ( -B + sqrt (disk)/2*A
    x2 ( -B - sqrt(disk)/2*A
else if disk = 0 then
    x1 ( -B/2*A
    x2 ( x1
else write ('Akar imajiner')
end if
write (x1,x2)
```

Flow Chart 4.3.



Translasi 4.3.

Bahasa C++

```
#include <iostream.h>
#include <math.h>
#include <conio.h>
class Akar {
    friend ostream& operator<<(ostream&, Akar&);
    friend istream& operator>>(istream&, Akar&);
public:
    Akar();
    int disk() { return B*B-4*A*C; }
    float akar1() {return (-B+sqrt(disk()))/(2*A); }
    float akar2() {return (-B-sqrt(disk()))/(2*A); }
    void hitung_akar();
    void cetak_disk() { cout << "diskriminan = " << disk() << endl; }
    void cetak_akar() {
        cout << "x1 = " << akar1() << endl;
        cout << "x2 = " << akar2() << endl;
    }
private:
    int A, B, C; // input
    float x1, x2; // akar 1 dan akar 2
};

ostream& operator<<(ostream& out, Akar& keluaran) {
    keluaran.cetak_disk();
    if (keluaran.disk() >= 0) keluaran.cetak_akar();
    else out << "Akar imajiner";
    return out;
}

istream& operator>>(istream& in, Akar& masukan) {
    cout << "Koefisien pangkat 2 : "; in >> masukan.A;
    cout << "Koefisien pangkat 1 : "; in >> masukan.B;
    cout << "Koefisien pangkat 0 : "; in >> masukan.C;
    return in;
}

Akar::Akar() {
    cout << "Menghitung akar persamaan kuadrat\n";
}
```

```

void Akar::hitung_akar() {
    if (A == 0) {
        cout << "bukan pers. kuadrat.\n";
        cout << "Harga akar = " << -C/B; } else {
        if (disk() > 0) {
            x1 = akar1();
            x2 = akar2();
        } else if (disk() == 0) {
            x1 = akar1();
            x2 = x1;
        }
    }
}

void main() {
    Akar kasus;
    cin >> kasus;
    kasus.hitung_akar();
    cout << kasus;
}

```

Output :

```

Koefisien pangkat 2 : 1
Koefisien pangkat 1 : -1
Koefisien pangkat 0 : -20
diskriminan = 81
x1 = 5.5
x2 = -3.5

```

Kasus 4.4.

Konversikan nilai angka menjadi nilai huruf dengan ketentuan sebagai berikut :

Nilai Angka	Nilai huruf
0 – 20	E
21 – 40	D
41 – 60	C
61 – 80	B
81 – 100	A

Algoritma 4.4.

```

Algoritma Konversi_Nilai
{Mengkonversikan nilai angka menjadi nilai huruf}

```

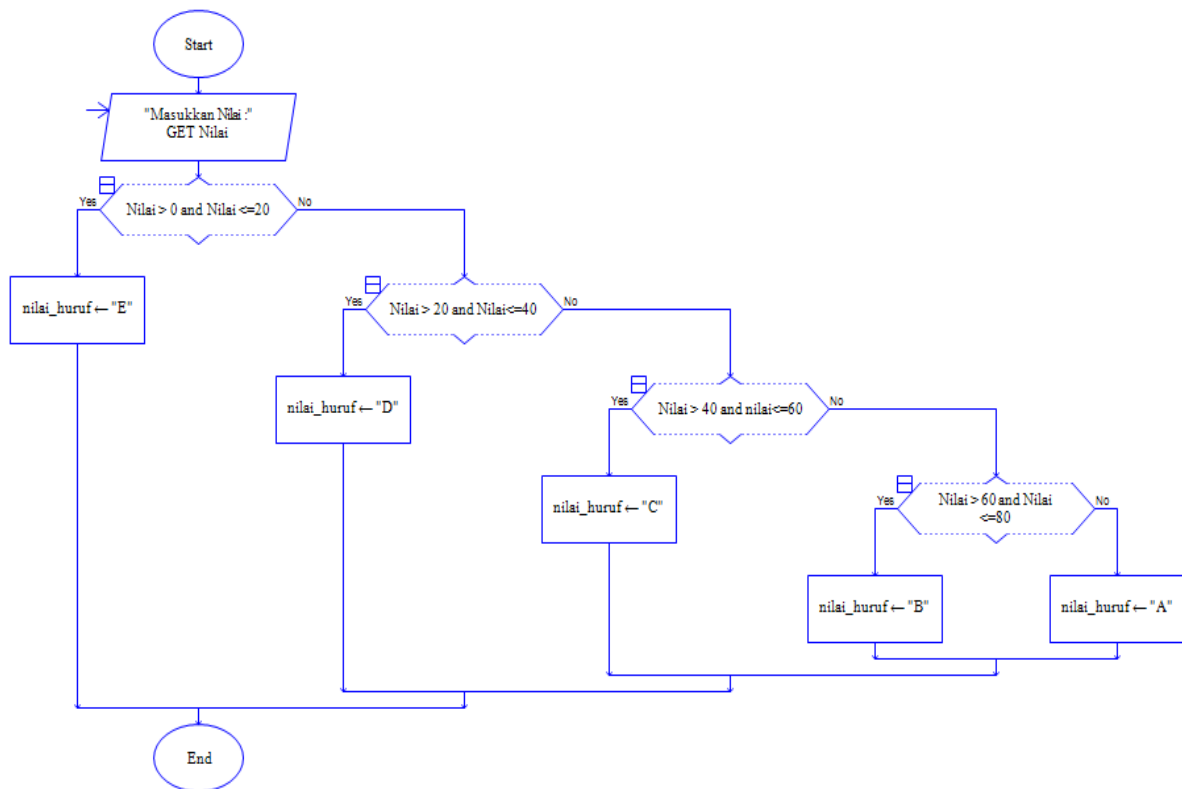
Deklarasi

nilai : integer
nilai_huruf : char

Deskripsi

```
read (nilai)
if (nilai > 0) and (nilai <= 20) then nilai_huruf ← 'E'
else if (nilai > 20) and (nilai <= 40) then nilai_huruf ← 'D'
else if (nilai > 40) and (nilai <= 60) then nilai_huruf ← 'C'
else if (nilai > 60) and (nilai <= 80) then nilai_huruf ← 'B'
else nilai_huruf ← 'A'
endif
write (nilai_huruf)
```

Flowchart 4.4.



Translasi 4.4.

Bahasa C++

```
#include <iostream.h>
#include <conio.h>
class Nilai {
```

```

friend ostream& operator<<(ostream&, Nilai&);
friend istream& operator>>(istream&, Nilai&);
public:
    Nilai() {};
    void konversikan() {
        if ((nilai > 0) && (nilai <= 20))
            nilai_huruf = 'E';
        else if ((nilai > 20) && (nilai <= 40))
            nilai_huruf = 'D';
        else if ((nilai > 40) && (nilai <= 60))
            nilai_huruf = 'C';
        else if ((nilai > 60) && (nilai <= 80))
            nilai_huruf = 'B';
        else nilai_huruf = 'A';
    }
private:
    int nilai;
    char nilai_huruf;
};

istream& operator>>(istream& in, Nilai& masukan) {
    cout << "Masukkan nilai angka = ";
    in >> masukan.nilai;
    return in;
};

ostream& operator<<(ostream& out, Nilai& keluaran) {
    out << "Nilai angka : " << keluaran.nilai << endl;
    out << "Nilai huruf : " << keluaran.nilai_huruf;
    return out;
}

void main() {
    Nilai angka;
    cin >> angka;
    angka.konversikan();
    cout << angka;
}

```

Kasus 4.5.

Buatlah algoritma, flowchart, dan program untuk mengkonversi hari ke-1 adalah hari Senin sampai dengan hari ke-7 adalah Minggu. Misalkan dimasukkan nilai 5, outputnya adalah hari Jum'at.

Algoritma 4.5.

Algoritma menentukan_hari { masukan integer 1 sampai 7, akan ditentukan hari apa yang sesuai }
Deklarasi hari_ke : integer; hari : string;
Deskripsi read(hari_ke); case (hari_ke) of 1 : hari ← 'Senin' 2 : hari ← 'Selasa'; 3 : hari ← 'Rabu'; 4 : hari ← 'Kamis'; 5 : hari ← 'Jum"at'; 6 : hari ← 'Sabtu'; else hari ← 'Minggu'; endcase write(hari)

Translasi 4.5.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Kalender { friend ostream& operator<<(ostream&, Kalender&); friend istream& operator>>(istream&, Kalender&); public: Kalender() {}; void adalah(); private: int hari_ke; char *hari; }; void Kalender::adalah() { switch (hari_ke) { case 1: hari = "Senin"; break; case 2: hari = "Selasa";break; case 3: hari = "Rabu"; break; case 4: hari = "Kamis"; break; case 5: hari = "Jum'at";break; case 6: hari = "Sabtu"; break; default: hari = "Minggu";break; }</pre>

```

    }
}

istream& operator>>(istream& in, Kalender& masukan) {
    cout << "Masukkan hari ke-";
    in >> masukan.hari_ke;
    return in;
}

ostream& operator<<(ostream& out, Kalender& keluaran) {
    out << "Hari ke- " << keluaran.hari_ke << " adalah "
        << keluaran.hari << endl;
    return out;
}

main() {
    Kalender tanggal;
    cin >> tanggal;
    tanggal.adalah();
    cout << tanggal;
    getch();
    return 0;
}

```

Latihan

1. Buatlah flowchart dan translasi program dari algoritma 3.2.b. di atas.

2. Sempurnakan algoritma 3.3. dengan akar imajiner berbentuk :

$$\mathbf{a + bi}$$

dengan a dan b bilangan real dan bila harga $b^2 - 4ac < 0$.

Perhatikan bahwa nilai a dan b dihitung **terpisah**.

Rumus menghitung nilai a :

Rumus menghitung nilai b :

Operator Overloading untuk output :

3. Buatlah program dari algoritma 3.4. menggunakan case ... of .
4. Buatlah algoritma, flowchart dan program untuk mengecek apakah pemakai memasukkan bilangan bulat atau bilangan real.
- Lebih dulu, carilah perbedaan dari 1.0 dan 1 kemudian 1 dan 1.23.
- Perbedaan :

Algoritma :

5. Buatlah algoritma, flowchart dan program untuk menentukan banyaknya hari dalam suatu bulan dan tahun yang diberikan. Pertimbangkan pula adanya tahun kabisat di mana bulan Februari mempunyai jumlah 29 hari. Gunakan case .. of dan if .. then .. else .. yang tepat untuk setiap kasus.
6. Buatlah algoritma, flowchart dan program untuk mengecek apakah karakter yang dimasukkan itu merupakan huruf besar, huruf kecil atau digit (bilangan). Gunakan fungsi ORD(char) untuk menyatakan nomor urut dalam tabel ASCII.

Algoritma :

7. Seorang dosen ingin mengkonversikan nilai angka hasil ujian mahasiswa menjadi nilai huruf dengan ketentuan :

Nilai antara	Nilai Huruf
0-60	F
61-70	D
71-80	C

81-90	B
91-100	A

8. Diberikan ketentuan pecahan uang di Indonesia. User memasukkan sebuah bilangan, program harus dapat memecahnya menjadi pecahan sesuai dengan mata uang di Indonesia.

Misalkan dimasukkan nilai uang 127.675 rupiah. Cobalah memecah secara manual nilai uang di atas.

Algoritma :

Program :

9. Seorang pegawai bekerja selama 5 hari kerja, yaitu Senin sampai Jum'at. Setiap hari dia masuk jam 08.00 dan pulang jam 16.00. Kecuali Jum'at dia pulang jam 11.00. Apabila dia bekerja lebih dari 30 jam per bulan maka setiap 5 jam akan memperoleh uang lembur sebesar Rp 30.000,- bila kurang dari 5 jam maka akan dihitung Rp 4.000,-/jam. Buat program dengan masukan bulan yang diinginkan, dan outputnya berupa besarnya uang lembur pegawai tersebut.

Perhitungan secara manual :

Uji Pemrograman

1. Pernyataan if. Tulislah sebuah program yang mendeklarasikan dua variabel yaitu a dan b, yang nilai telah diinisialisasi (terserah anda). Tulislah 10 pernyataan if dengan kondisi berikut :

(a <= b)

!(b == a)
(3 < a) && (3 < b)
(a b)
((++a) == (--b)) b)
(a ^ b)
((a && b && (!0)) true)
(b == 10)
(int c = b)
(a (!b))

2. Pernyataan if -- else. Tulislah sebuah program yang mendeklarasikan dua variabel karakter yaitu a dan b, yang nilai telah diinisialisasi dengan ' '. Tulislah 10 pernyataan if – else dengan berikut :

(a = 'y')
(a == 'Y')
(!(b = 'n'))
(b == 'n')
('c' <= b)
((('A' <= a) && (a <= 'Z'))
(a == b)
(a != b)
((b >= a) (a <= '1'))
(a && b && '0')

BAB 5

PERULANGAN/ LOOP

5.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan berbagai algoritma yang melibatkan perulangan baik iterative maupun rekursif.

5.2 Pendahuluan

Dalam kehidupan sehari-hari, sering kali kita temukan kegiatan yang dilakukan berulang-ulang. Dimulai pada kondisi tertentu, dilakukan langkah berulang, kondisi selesai. Sebagai contoh, untuk menghabiskan sepiring nasi (kondisi awal) kita suap satu sendok demi satu sendok. Setiap kali langkah makan satu sendok, banyaknya nasi dalam piring berukuran satu sendok. Kita berhenti makan apabila makanan yang ada dalam piring habis (kondisi selesai).

Contoh lain, pada suatu saat kita ingin membeli 5 liter bensin. Saat awal, konter besarnya bensin yang keluar adalah 0 (kondisi awal). Berangsur-angsur, liter demi liter (langkah), sampai pada liter ke-5, bensin tidak lagi keluar (kondisi selesai).

Perulangan bisa menaik (kasus membeli bensin), namun bisa pula menurun (kasus makan sepiring nasi). Pada dasarnya, baik menaik maupun menurun, tergantung dari sudut pandang yang dipilih. Sebagai contoh, pada kasus makan sepiring nasi, kita dapat menghitung, berapa sendok yang telah kita makan pada suatu saat. Setiap kali makan sesendok nasi, maka jumlah nasi yang dimakan bertambah satu sendok. Untuk kasus membeli bensin, tinggal berapa liter yang akan keluar dari mesin bisa kita identifikasi. Setiap kali 1 liter bensin keluar, maka sisa bensin yang keluar dari mesin berkurang satu.

Namun adakalanya, baik berkurang maupun bertambah jumlahnya, tidak dapat ditentukan kapan (sudah berapa kali) perulangan dilakukan. Pada kasus membeli bensin, kita

tahu kapan berhenti secara pasti, yaitu bila mesin telah mengeluarkan bensin sebanyak 5 liter. Namun pada kasus makan sepring nasi, kondisi berhenti apabila nasi dalam piring telah habis. Kita tidak peduli, berapa sendok nasi yang telah kita makan.

Salah satu kelebihan komputer adalah dapat melakukan perhitungan berulang dengan sangat cepat. Oleh karena banyak persoalan melibatkan langkah-langkah yang sering diulang maka salah satu aspek algoritma yang penting adalah mempelajari aspek-aspek perulangan.

5.3 Konsep Counter

Pada kasus membeli bensin, misalkan i menyatakan berapa bensin yang telah keluar dari mesin.

Nilai i	Nilai i sesudah ditambah	Nilai i sebelum ditambah
0 liter	1 liter \leftarrow	0 liter + 1 liter
1 liter	2 liter	1 liter + 1 liter
2 liter	3 liter	2 liter + 1 liter
3 liter	4 liter	3 liter + 1 liter
4 liter	5 liter, stop !!!	4 liter + 1 liter
5 liter		

Apabila kolom kedua menyatakan kondisi i sesudah 1 liter bensin keluar dari mesin maka kita dapat membuat penugasan seperti berikut ini :

$$i \leftarrow i + 1$$

Arti dari penugasan di atas adalah i di sebelah kiri merupakan harga i **yang baru** setelah harga i **sebelah kanan** ditambah dengan 1. Dengan demikian, i merupakan counter (pencacah) di mana setiap kali pernyataan penugasan itu dieksekusi, harga i akan bertambah 1.

5.4 Konsep Total

Pada suatu saat seorang dosen ingin menjumlahkan berapa nilai total dari ujian seorang mahasiswa. Banyaknya soal ujian adalah 4 soal. Misalkan nilai dari soal pertama adalah 30, soal kedua 20, soal ketiga 5, dan soal terakhir 10. Tahapan penjumlahan total nilai yang diperoleh dapat digambarkan sebagai berikut :

Nilai total	Total sesudah ditambah nilai	Total sebelum ditambah nilai
-------------	------------------------------	------------------------------

0 (awal)	30	0 + 30 (nilai 1)
30	50	30 + 20 (nilai 2)
50	55	50 + 5 (nilai 3)
55	65	55 + 10 (nilai 4)
65 (nilai total terakhir)		

Pada dasarnya, pola dari konsep total tidak berbeda dengan konsep konter. Dalam konsep konter, elemen penambah adalah 1, sementara pada konsep total, elemen penambah tidak sama dengan 1 (bisa merupakan pola bisa tidak). Untuk kasus di atas, penambahan nilai tidak terpola. Untuk penambahan yang terpola, misalnya kita diminta untuk menghitung jumlah deret :

$$1 + 2 + 3 + \dots + n = \sum_{i=1}^n i$$

Setiap kali langkah penjumlahan dilakukan, elemen penambah bertambah 1 dari sebelumnya. Pola deret di sebelah kiri dapat dirangkum menggunakan tanda sigma (\sum) untuk deret dengan operasi penjumlahan, tanda \prod digunakan untuk deret dengan operasi perkalian.

Tahapan penjumlahan dapat dilakukan dengan menggunakan tabel seperti kedua contoh di atas. Namun bisa juga menggunakan proses analisis seperti berikut ini. Harga awal dari nilai kumulatif penjumlahan i dimulai dengan 0 (mengingatkan kita pada unsur identitas penjumlahan adalah 0). Misalkan :

$$\text{jumlah} \leftarrow 0$$

Dengan menggunakan salah satu bentuk perulangan, misalnya for, kita dapat menghitung jumlah deret di atas dengan pernyataan :

$$\text{for } i \leftarrow 1 \text{ to } n \text{ do}$$

$$\text{jumlah} \leftarrow \text{jumlah} + i$$

Dalam hal ini, kita dapat melihat bahwa elemen penambah dapat diwakili dengan struktur for oleh karena nilai i selalu berubah bertambah satu setiap kali pernyataan penjumlahan dilakukan. Konsep total ini merupakan konsep dasar untuk penerapan yang melibatkan harga kumulatif. Konsep inipun dapat digunakan untuk operator perkalian. Misalkan kita diminta menghitung :

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \prod_{i=1}^n i$$

Harga awal dari kumulatif perkalian dimulai dengan 1 (mengingatkan kita pada unsur identitas perkalian adalah 1). Misalkan :

$$\text{kali} \leftarrow 1$$

kemudian dengan perulangan yang mirip (hanya mengganti operator penjumlahan (+) dengan perkalian (*)) kita peroleh :

```
for i ← 1 to n do
    kali ← kali * i
```

5.5 Pemrograman

Hampir semua bahasa pemrograman mempunyai pernyataan yang berkaitan dengan perulangan.

Algoritmik	C++
for i ← awal to akhir do aksi end for	for (i = awal; i <= akhir; i++) aksi;
for i ← awal downto akhir do aksi end for	for (i = awal; i >= akhir; i--) aksi;
while (kondisi) do aksi end while	while (kondisi) aksi;
repeat aksi until (kondisi)	do { aksi; } while (kondisi);

Aksi dapat berupa pernyataan tunggal maupun majemuk. Pernyataan majemuk dalam bahasa C++ diawali dengan “{” dan diakhiri dengan “}”.

Pada dasarnya, ketiga bentuk di atas mempunyai struktur yang sama. Untuk pernyataan for menaik, konversi while dan do – while nya sebagai berikut :

for	while	do – while
for (i = awal; i <= akhir; i++) aksi;	i = awal; while (i <= akhir) { aksi; i++; }	i = awal; do { aksi; i++; } while (i <= akhir);

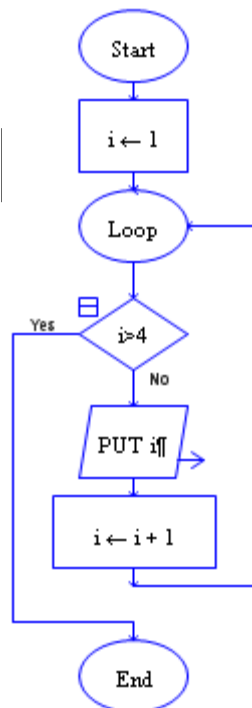
Kasus 5.1.

Cetaklah bilangan 1 sampai 4 menggunakan perulangan.

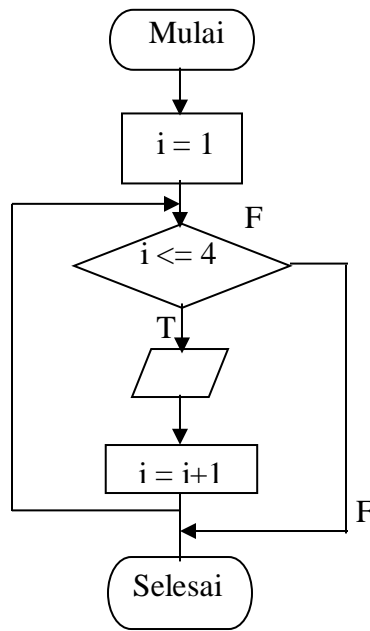
Algoritma 5.1

Algoritma Cetak_Angka {Mencetak angka 1, .., 4 ke piranti keluaran}		
Deklarasi i : integer		
for loop	while loop	repeat until loop
Deskripsi for i \leftarrow 1 to 4 do write (i) endfor	Deskripsi i \leftarrow 1 while (i \leq 4) do write (i) i \leftarrow i + 1 endwhile	Deskripsi i \leftarrow 1 repeat write (i) i \leftarrow i + 1 until (i > 4)

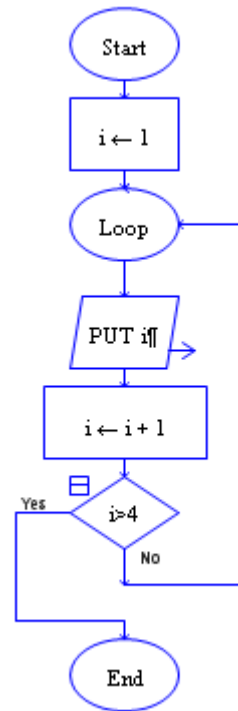
Flow Chart 5.1.



Struktur for



Struktur while - do



Struktur repeat - until

Translasi 5.1.b. Bahasa C++

for loop	while loop	do while loop
<pre> #include <iostream.h> main() { int i; for (i=1; i<=4; i++) cout << " " << i; return 0; } </pre>	<pre> #include <iostream.h> main() { int i=1; while (i <= 4) { cout << " " << i; i++; } return 0; } </pre>	<pre> #include <iostream.h> main() { int i=1; do { cout << " " << i; i++; } while (i <= 4); return 0; } </pre>

Kasus 5.2.

Cetaklah bilangan ganjil dari 0 sampai 10 menggunakan perulangan (for, while – do, repeat – until).

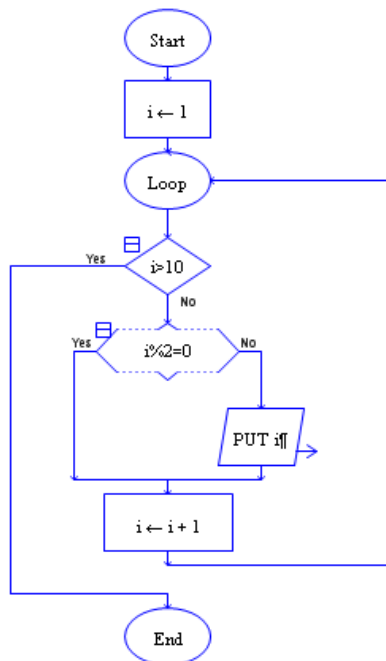
Ide :

Bilangan ganjil dari 0 sampai 10 diawali dengan 1, kemudian bertambah dengan 2 atau bilangan ganjil adalah bilangan yang bila dibagi 2 bersisa 1.

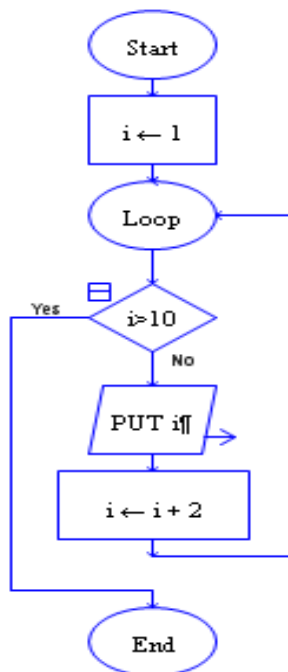
Algoritma 5.2.

Algoritma Cetak Ganjil { Mencetak bilangan ganjil dari 0 sampai 10 ke piranti keluaran }		
Deklarasi <i>i</i> : integer		
Deskripsi for <i>i</i> ← 0 to 10 do if (<i>i</i> mod 2 = 1) then write (<i>i</i>) endif endfor	Deskripsi <i>i</i> ← 1 while (<i>i</i> ≤ 10) do write (<i>i</i>) <i>i</i> ← <i>i</i> + 2 endwhile	Deskripsi <i>i</i> ← 1 repeat write (<i>i</i>) <i>i</i> ← <i>i</i> + 2 until (<i>i</i> > 10)

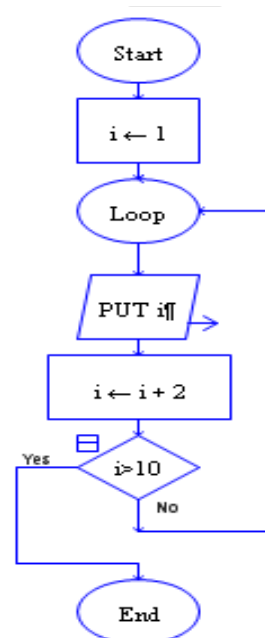
Flow Chart 5.2.



Struktur for



Struktur while - do



Struktur repeat – until

Translasi 5.2.

Struktur for	Struktur while	Struktur do - while
void main() {	void main() {	void main() {
for (int i = 0; i <=10; i++)	int i = 1;	int i = 1;
if (i % 2 == 1)	while (i <= 10) {	do {
cout << i << “ “;	if (i % 2 == 1)	if (i % 2 == 1)
}	cout << i << “ “;	cout << i << “ “;
	i++; }	i++; }
	}	while (i <= 10);
		}

Kasus 5.3.

Carilah rata-rata dari n bilangan bulat positif.

Analisis :

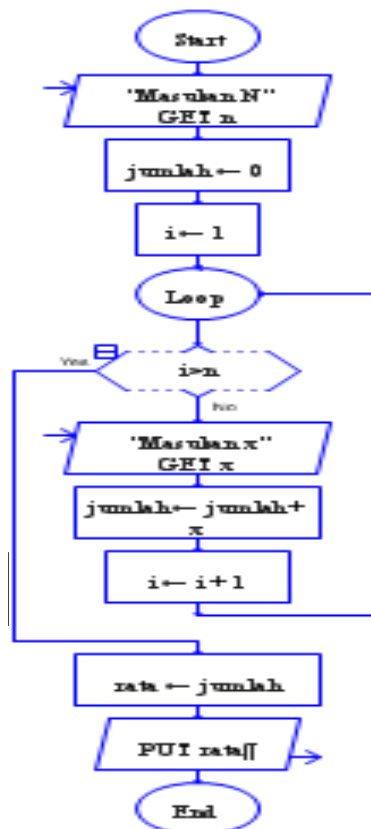
Rumus rata-rata adalah :
$$\frac{\sum_{i=1}^n x_i}{n}$$

yaitu jumlah data dibagi dengan banyaknya data, dengan x_i adalah data ke-i.

Algoritma 5.3.

Algoritma mencari rata-rata {Diberikan n data kemudian dicari rata-ratanya}
Deklarasi i, n, jumlah, x : integer rata : real
Deskripsi read(n) jumlah \leftarrow 0 for i \leftarrow 1 to n do read(x) jumlah \leftarrow jumlah + x endfor rata \leftarrow jumlah/n write(rata)

Flowchart 5.3.



Translasi 5.3.

Bahasa C++

```
#include <iostream.h>
#include <conio.h>
class Rata {
    friend ostream& operator<<(ostream&, Rata&);
    friend istream& operator>>(istream&, Rata&);
public:
    float hitung_rata();
private:
    int n;    // banyak data
    int data; // variabel untuk menyimpan data
};

istream& operator>>(istream& in, Rata& A) {
    cout << "Banyak data : ";
    in >> A.n;
    return in;
}

float Rata::hitung_rata() {
    int jumlah = 0;
    float rata;
    for (int i = 1; i<=n; i++) {
        cout << "Data ke-" << i << " : ";
        cin >> data;
        jumlah = jumlah + data;
    }
    rata = (float)jumlah/n;
    return rata;
}

ostream& operator<<(ostream& out, Rata& A) {
    out << "Rata-rata = " << A.hitung_rata();
    return out;
}

void main() {
    Rata X;
    cin >> X;
    cout << X;
    getch();
}
```

5.6 Sentinel

Kadang banyaknya masukan tidak diketahui, tetapi sifat datanya diketahui. Seperti kasus 4.3. di mana semua data adalah bilangan bulat positif, maka untuk menghentikan masukan, kita dapat menggunakan harga lain (bisa negatif atau 0). Setiap masukan akan dicek dengan harga lain tersebut (yang dinamakan sentinel/pengawal).

Kasus 5.4.

Hitunglah rata-rata dari bilangan bulat positif, di mana banyak data ditentukan dari data yang dimasukkan.

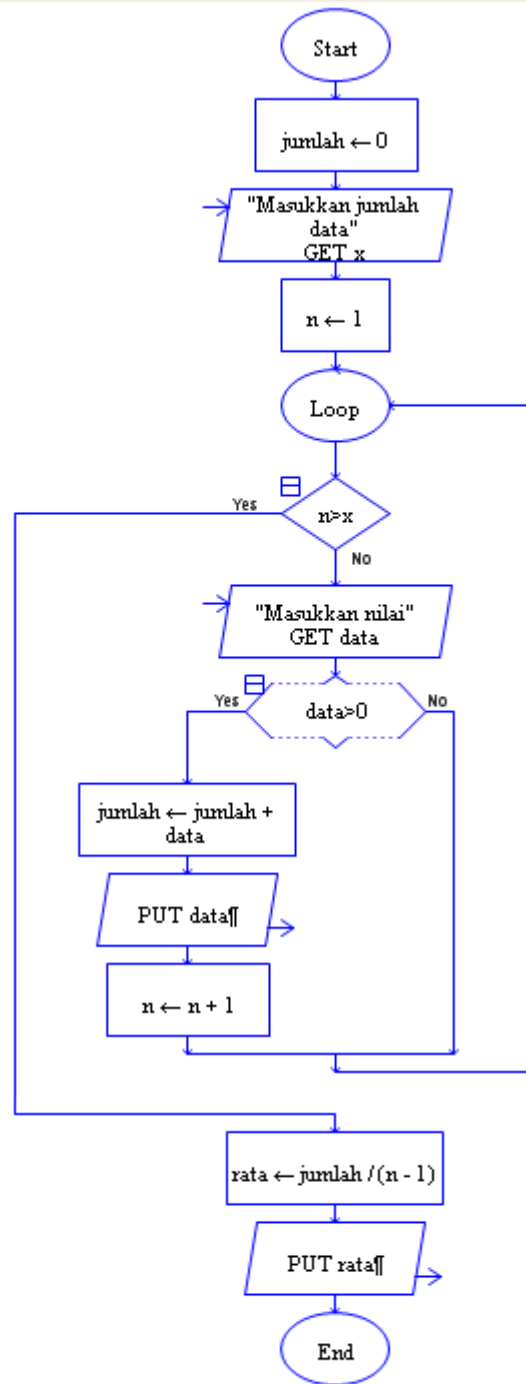
Algoritma 5.4.

Algoritma mencari rata-rata {Diberikan data bilangan bulat positif kemudian dicari rata-ratanya}
Deklarasi n, jumlah, x : integer rata : real
Deskripsi jumlah \leftarrow 0 read(x) n \leftarrow 1 while (x>0) do jumlah \leftarrow jumlah + x read(x) n \leftarrow n+1 endfor rata \leftarrow jumlah/(n-1) write(rata)

Penjelasan :

Terlihat rumus yang digunakan agak berbeda, yaitu pembaginya (n-1). Hal disebabkan pada saat nilai terakhir dimasukkan, nilai ini masih digunakan untuk validasi masukan (apakah masih ada masukan berikutnya atau tidak) sehingga banyaknya data selalu “kelebihan” 1. Untuk itu, pembagi dalam rumus rata-rata “haruslah” dikurangi dengan 1.

Flowchart 5.4.



Translasi 5.4.

Bahasa C++
<pre>float Rata::hitung_rata() { int jumlah = 0; float rata; cout << "Data ke-1 : "; cin >> data; n = 1; while (data>0) {</pre>

```

    jumlah = jumlah + data;
    cout << "Data ke-" << n+1 << " : ";
    cin >> data;
    n++;
}
rata = (float)jumlah/(n-1);
cout << "Data ke-" << n << " yaitu : " << data << " tidak termasuk perhitungan\n";
return rata;
}

```

Kasus 5.5.

Tentukan nilai dari :

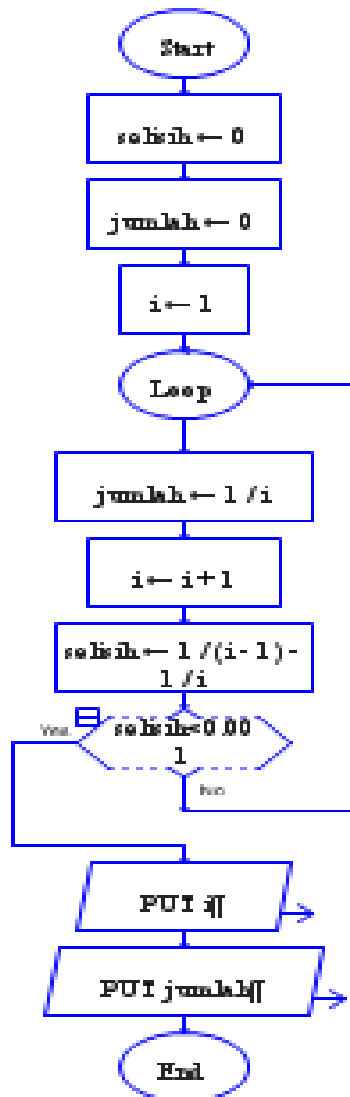
$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} = \sum_{i=1}^n \frac{1}{i}$$

sampai selisih 2 suku tidak lebih dari 0.001. Tentukan pula nilai n terakhir.

Algoritma 5.5.

Algoritma mencari jumlah deret
Deklarasi n, jumlah, x : integer rata : real
Deskripsi jumlah \leftarrow 0 i \leftarrow 1 repeat jumlah \leftarrow jumlah + (1/i) i \leftarrow i+1 selisih \leftarrow (1/i)-(1/(i-1)) until abs(selisih) < 0.001 write(jumlah, i)

Flowchart 5.5.



Translasi 5.5.

Bahasa C++
<pre> #include <iostream.h> #include <conio.h> #include <math.h> class Deret { friend ostream& operator<<(ostream&, Deret&); public: float hitung_Deret(); private: int n; // sampai n suku float hasil; // variabel untuk menyimpan jumlah deret }; </pre>

```

float Deret::hitung_Deret() {
    float selisih, jumlah = 0.0;
    int i=1;
    do {
        jumlah += 1.0/i;
        i++;
        selisih = 1.0/(i-1) - 1.0/i;
    } while (selisih >= 0.001);
    cout << "Perulangan dilakukan sebanyak : " << i << endl;
    return jumlah;
}

ostream& operator<<(ostream& out, Deret& A) {
    out << "Jumlah deret = " << A.hitung_Deret();
    return out;
}

void main() {
    Deret X;
    cout << X;
}

```

Kasus 5.6.

Carilah nilai dari $n!$ (n faktorial).

Analisis :

Nilai n faktorial secara matematis didefinisikan sebagai berikut :

$$n! = 1 \times 2 \times 3 \times \dots \times n = \prod_{i=1}^n i$$

dengan $0! = 1$ atau $1! = 1$.

Algoritma 5.6.

Algoritma mencari n faktorial

{Dicari n faktorial dengan masukan n }

Deklarasi

i, n : integer

faktorial : longint

Deskripsi

read(n)

faktorial $\leftarrow 1$

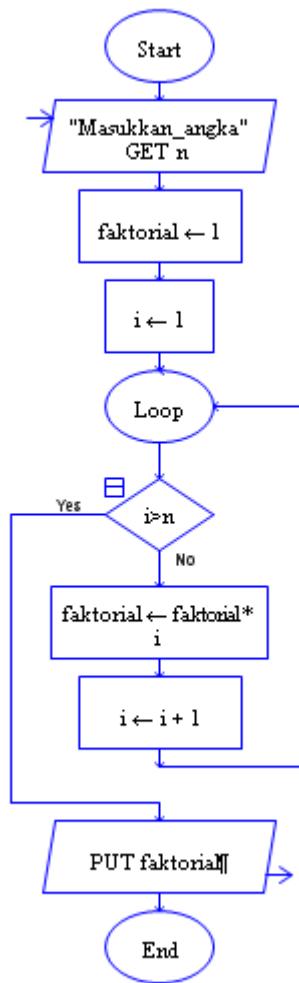
for $i \leftarrow 1$ to n do

 faktorial \leftarrow faktorial * i

endfor

write(faktorial)

Flowchart 5.6.



Translasi 5.6.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Operator { friend ostream& operator<<(ostream&, Operator&); friend istream& operator>>(istream&, Operator&); public: long faktorial(); private: int n; long hasil; }; long Operator::faktorial() { long fak = 1;</pre>

```

    for (int i = 1; i <= n; i++)
        fak = fak * i;
    return fak;
}

ostream& operator<<(ostream& out, Operator& a) {
    a.hasil = a.faktorial();
    out << "Hasil cara iterasi : " << a.n << "! adalah " << a.hasil;
    return out;
}

istream& operator>>(istream& in, Operator& a) {
    cout << "Masukkan integer n : ";
    in >> a.n;
    return in;
}

void main() {
    Operator run;
    cin >> run;
    cout << run;
    getch();
}

```

Kasus 5.7.

Dengan menggunakan fungsi ln dan exp, buatlah fungsi untuk menghasilkan nilai x^y .

Analisis :

Dengan menggunakan sifat logaritma :

$$\ln(x^y) = y * \ln(x)$$

$$\exp(\ln(x^y)) = \exp(y * \ln(x))$$

sehingga :

$$x^y = \exp(y * \ln(x))$$

Algoritma 5.7.

Fungsi Pangkat(input x, y : integer)
{ Menghitung nilai dari x pangkat y }

Deskripsi

pangkat $\leftarrow \exp(y * \ln(x))$

Translasi 5.7.

Bahasa C++
<pre> #include <iostream.h> #include <math.h> float pangkat(int x, int y) { return(exp(y*log(x))); } main() { float hasil; int a, b; cout << "Menghitung hasil perpangkatan\n"; cout << "Tulis sebuah bilangan : "; cin >> a; cout << "Mau dipangkat berapa : "; cin >> b; hasil = pangkat(a,b); cout << a << " pangkat " << b << " = " << hasil; return 0; } </pre>

Kasus 5.8.

Hitunglah nilai dari x^y dengan x bilangan real dan y bilangan bulat.

Analisis :

$$x^y = x \cdot x \cdot x \cdot \dots \cdot x \text{ (sebanyak } y \text{ kali)} = \prod_{i=1}^y x$$

Algoritma 5.8.

Algoritma Pangkat {Diberikan masukan x dan y, dihitung nilai dari x pangkat y}
Deklarasi x, y, i : integer { input } pangkat : integer { output }
Deskripsi read (x,y) pangkat \leftarrow 1 for i \leftarrow 1 to y do pangkat \leftarrow pangkat * x enfor write (pangkat)

Contoh penggunaan :

1.	#include <iostream.h>
2.	
3.	int pangkat(int x, int y) {
4.	int hasil=1;
5.	for (int i=1; i<=y; i++)
6.	hasil*=x;
7.	return hasil;
8.	}
9.	
10.	void eja(int x) {
11.	switch (x) {
12.	case 1 : cout << " satuan "; break;
13.	case 2 : cout << " puluhan + "; break;
14.	case 3 : cout << " ratusan + "; break;
15.	case 4 : cout << " ribuan + "; break;
16.	case 0 : cout << " "; break;
17.	}
18.	}
19.	
20.	void urai(int n) {
21.	int pilihan=4;
22.	int mulai=10, hasil;
23.	cout << n << " = ";
24.	while (n!=0) {
25.	hasil=n/pangkat(mulai,pilihan-1);
26.	n%=pangkat(mulai,pilihan-1);
27.	if (hasil) {
28.	cout << hasil;
29.	eja(pilihan);
30.	}
31.	pilihan--;
32.	}
33.	}
34.	
35.	void main() {
36.	int bilangan;
37.	cout << "Masukkan bilangan bulat : ";
38.	cin >> bilangan;
39.	urai(bilangan);
40.	}

Kasus 5.9.

Buatlah fungsi perkalian 2 bilangan bulat dengan menggunakan operator penjumlahan.

Analisis :

$a \times b = a + a + a + \dots + a$ (sebanyak b kali)

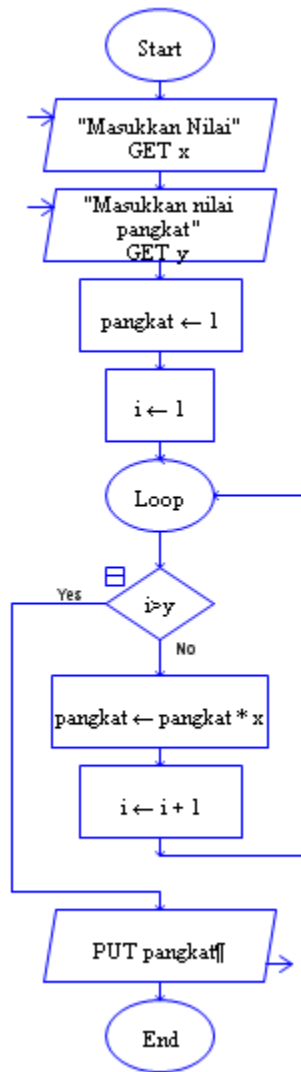
Algoritma 5.9.

Fungsi kali(input a, b : integer) : integer { Menghitung hasil perkalian a dan b menggunakan Operator Penjumlahan }
Deklarasi hasil, i : integer
Deskripsi hasil \leftarrow 0 for i \leftarrow 1 to b do hasil \leftarrow hasil + a kali \leftarrow hasil

Translasi 5.9.

Bahasa c++
<pre>#include <iostream.h> int kali(int m, int n) { int i, hasil=0; for (i=1; i<=n; i++) hasil = hasil + m; return(hasil); } main() { int a, b; cout << "Masukkan bilangan : "; cin >> a; cout << "Akan dikali dengan : "; cin >> b; cout << "Nilai " << a << " x " << b << " = " << kali(a,b); return 0; }</pre>

Flowchart 5.9.



Translasi 5.9.

Bahasa C++
<pre>#include <iostream.h> main() { int x, y, i; int pangkat; cout << "Menghitung hasil perpangkatan\n"; cout << "Tulis sebuah bilangan : "; cin >> x; cout << "Mau dipangkat berapa : "; cin >> y; pangkat = 1; for (i = 1; i <= y; i++) pangkat = pangkat * x;</pre>


```

cout << x << " pangkat " << y << " = "
    << pangkat;
return 0;
}

```

Latihan

1. cetaklah bilangan 1 sampai dengan 4 menggunakan **downto**.
2. buatlah translasi dalam bahasa C untuk algoritma 4.5. Apakah diperlukan modifikasi algoritma ?
3. Buatlah algoritma dan program untuk mencetak bilangan yang habis dibagi 3 dan 5 antara 1 sampai dengan 100.
4. Hitunglah nilai dari :

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{n}$$

5. Buatlah algoritma dan program untuk menghitung nilai dari permutasi dan kombinasi n buah bola yang diambil r bola.

Petunjuk :

Rumus permutasi dan kombinasi adalah sebagai berikut :

$$P(n, r) = \frac{n!}{(n-r)!} \quad \text{dan} \quad C(n, r) = \frac{n!}{r!(n-r)!}$$

6. Buatlah algoritma dan program untuk mengkonversi bilangan desimal menjadi bilangan biner. **Petunjuk** : Gunakan operator mod dan div !
7. Buatlah algoritma, flowchart dan program untuk menampilkan permutasi dari 3 huruf. Misalnya diberikan : **abc**

output :

abc

bca

bac, dan seterusnya

5.7 Fungsi Rekursif

Rekursif berasal dari bahasa Latin recur (re = kembali, curre : eksekusi). Fungsi rekursif adalah fungsi yang melakukan proses perulangan dengan cara memanggil dirinya sendiri. Ini berbeda dengan versi iteratif yang menggunakan perulangan for, while maupun repeat until. Fungsi rekursif dapat dipandang sebagai sebuah “operator”. Misalkan kita lihat kasus 5.4. di mana secara iteratif n faktorial didefinisikan sebagai :

$$n! = 1 \times 2 \times 3 \times \dots \times n = \prod_{i=1}^n i$$

n faktorial didefinisikan secara rekursif sebagai berikut :

$$\begin{aligned} n! &= 1, && \text{untuk } n = 0 \text{ atau } n = 1 \\ &= n \cdot (n-1)!, && n > 1. \end{aligned}$$

Dapat dilihat ternyata pada sisi kanan masih terdapat operator yang sama dengan sisi kiri. Dengan demikian, nilai n faktorial baru bisa diperoleh bila (n-1) faktorial telah diperoleh. Tentu saja n faktorial “lebih kompleks” dibanding dengan (n-1) faktorial. Pencarian ini berakhir bila sudah sampai pada nilai konstan, yakni telah dicapai harga n=0 atau n=1 yaitu 0!=1 atau 1!=1.

Untuk itu dapat diambil kesimpulan bahwa untuk fungsi rekursif ada 2 syarat, yaitu :

1. kasus penyetop. Dalam kasus ini terdapat nilai konstan
2. kasus pemanggilan rekursif. Dalam kasus ini terdapat pemanggilan fungsi itu sendiri, tetapi harus mengarah kepada kasus penyetop.

Kasus 5.10.

Buatlah fungsi faktorial secara rekursif untuk mencari n!.

Algoritma 5.10.

Fungsi faktorial(input n : integer) : longint
Deskripsi if (n=0) or (n=1) then faktorial \leftarrow 1 else faktorial \leftarrow n * faktorial(n-1)

class Operator {
friend ostream& operator<<(ostream&, Operator&);
friend istream& operator>>(istream&, Operator&);
public:
long faktorial();
long faktorial(int);
private:
int n;
long hasil;
};
long Operator::faktorial() {
long fak = 1;
for (int i = 1; i<=n; i++)
fak = fak * i;
return fak;
}
long Operator::faktorial(int n)
{ if ((n==0) (n==1)) return(1);
else return (n*faktorial(n-1));
}

Kasus 5.11.

Diberikan deret Fibonacci sebagai berikut :

1, 1, 2, 3, 5, 8, ...

Buatlah fungsi yang menghitung suku ke-n dari deret Fibonacci dengan menggunakan cara rekursif.

Analisis :

Suku ke-n dari deret Fibonacci diperoleh dengan rumus :

$$\text{fibonacci}(n) = \text{fibonacci}(n-1) + \text{fibonacci}(n-2)$$

dengan nilai awal untuk n=1 dan n=2 berharga 1.

Algoritma 5.11.

fungsi fibonacci (input n : integer) : integer
Deskripsi if (n = 1) or (n = 2) then fibonacci \leftarrow 1 else fibonacci \leftarrow fibonacci(n-1) + fibonacci(n-2) endif

Translasi 5.11.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Operator { friend ostream& operator<<(ostream&, Operator&); friend istream& operator>>(istream&, Operator&); public: long fibonacci(int); private: int n; }; int fibonacci (int n) { if ((n == 1) (n == 2)) return(1); else return(fibonacci(n-1) + fibonacci(n-2)); } ostream& operator<<(ostream& out, Operator& a) { for (int i = 1; i <= a.n; i++) out << fibonacci(i) << " "; return out; } istream& operator>>(istream& in, Operator& a) { cout << "Sampai suku ke : "; in >> a.n; return in; } main() { Operator run; cin >> run; cout << run; getch(); }</pre>

5.8 Iteratif Versus Rekursif

Dengan pengalaman memprogram, seorang programmer akan tahu, kapan menggunakan perulangan iteratif dan kapan menggunakan perulangan rekursif. Beberapa kasus berikut akan memberikan wawasan memprogram dengan kedua versi, yaitu versi iteratif dan versi rekursif.

Kasus 5.12.

Cetaklah suatu kalimat dengan cara iteratif maupun cara rekursif.

Translasi 5.12.a. Versi Iteratif

Bahasa C++
<pre>#include <iostream.h> #include <string.h> void balik(char *s) { int i; for (i=strlen(s)-1; i>=0; i--) cout << s[i]; } main(){ char *kata = "Algoritma"; balik(kata); return 0; }</pre>

Translasi 5.12.b. Versi Rekursif

Bahasa C++
<pre>#include <iostream.h> #include <string.h> void balik(char *s) { if (*s != '\0') { balik(&s[1]); cout << s[0]; } } main() { char *kata = "Algoritma"; balik(kata); return 0; }</pre>

Kasus 5.13.

Buatlah algoritma iteratif dan rekursif untuk menghitung gcd dari dua bilangan bulat positif.

Analisis :

Jika $n \neq 0$ dan m integer non negatif, kita dapat menulis $m = q.n + r$ untuk suatu integer non negatif q dan r dengan $0 \leq r < n$.

Contoh :

Jika $n = 3$, $m = 16$ maka $16 = 5(3) + 1$, yaitu $q = 5$ dan $r = 1$.

Jika $n = 10$, $m = 3$ maka $3 = 0(10) + 3$, yaitu $q = 0$ dan $r = 3$.

Untuk mencari nilai gcd dari dua integer. kita bisa menggunakan cara menulis di atas. Misalkan kita mau cari gcd(190,34).

$$34 \mid 190 \quad \rightarrow 190 = 5(34) + 20, r = 20$$

$$20 \mid 34 \quad \rightarrow 34 = 1(20) + 14, r = 14$$

$$14 \mid 20 \quad \rightarrow 20 = 1(14) + 6, r = 6$$

$$6 \mid 14 \quad \rightarrow 14 = 2(6) + 2, r = 2$$

$$2 \mid 6 \quad \rightarrow 6 = 3(2) + 0, r = 0 \text{ stop !}$$

Harga $r \neq 0$ terakhir dicapai adalah $r = 2$. Inilah hasil dari gcd(190,34). Versi rekursif, gcd didefinisikan sebagai berikut :

$$\text{gcd}(c,d) = c, \text{ jika } d = 0$$

$$= \text{gcd}(c-d, d), \text{ jika } d > 0 \text{ dan } c > d.$$

Juga berlaku hukum komutatif, $\text{gcd}(c,d) = \text{gcd}(d,c)$.

Algoritma 5.13.

fungsi gcd(c, d : integer) : integer	
versi iteratif	versi rekursif
Deskripsi while (d > 0) do $r \leftarrow c \bmod d$ $c \leftarrow d$ { menyimpan harga r terakhir } $d \leftarrow r$ { harga r terakhir untuk menghentikan perulangan } endwhile gcd $\leftarrow c$	Deskripsi if (d=0) then gcd $\leftarrow c$ else if (c<d) then gcd $\leftarrow \text{gcd}(d,c)$ else gcd $\leftarrow \text{gcd}(c-d, d)$

Translasi 5.13.a. Versi Iteratif

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class GCD { friend ostream& operator<<(ostream&, GCD&); friend istream& operator>>(istream&, GCD&); public: int HitungGCD(int, int); private: int x, y; }; int GCD::HitungGCD(int c, int d) { int r; while (d > 0) { r = c % d; c = d; d = r; } return (c); } istream& operator>>(istream& in, GCD& a) { cout << "Masukkan bilangan pertama : "; in >> a.x; cout << "Masukkan bilangan kedua : "; in >> a.y; return in; } ostream& operator<<(ostream& out, GCD& a) { out << "gcd(" << a.x << "," << a.y << ") = "; out << a.HitungGCD(a.x, a.y); return out; } void main() { GCD run; cin >> run; cout << run; getch(); }</pre>

Translasi 5.13.b. Versi Rekursif

Bahasa C++
<pre>int HitungGCD(int c, int d) { if (d==0) return(c); if (c<d) return(HitungGCD(d,c)); return(HitungGCD(c-d, d)); }</pre>

5.9 Macam-macam Metode Rekursi

Ada 3 jenis rekursi, yaitu :

1. Going Down Recursion (rekursi menurun), yaitu parameter menurun nilainya sampai dicapai kasus berhenti
2. Going Up Recursion (rekursi menaik), yaitu parameter menaik nilainya sampai dicapai kasus berhenti
3. Two Half (rekursi separuh-separuh), rekursi dibagi menjadi 2 bagian, di mana setiap bagian juga merupakan subprogram rekursi.

Ilustrasi ketiga jenis rekursi tersebut diperlihatkan seperti pohon berikut ini.

Going Down Recursion	pemanggilan rekursi	return value
GDR(5,10)	↓ GDR(5,9)+10*10	↑ 25+36+49+64+81+100=355
GDR(5,9)	↓ GDR(5,8) + 9*9	↑ 25+36+49+64+81=255
GDR(5,8)	↓ GDR(5,7) + 8*8	↑ 25+36+49+64=174
GDR(5,7)	↓ GDR(5,6) + 7*7	↑ 25+36+49=110
GDR(5,6)	↓ GDR(5,5) + 6*6	↑ 25+36=61
GDR(5,5)	5*5	↑ 25

Going Up Recursion	pemanggilan rekursi	return value
GUR(5,10)	↓ GUR(6,10)+ 5*5	↑ 100+81+64+49+36+25=355
GUR(6, 10)	↓ GUR(7,10) + 6*6	↑ 100+81+64+49+36=330
GUR(7, 10)	↓ GUR(8,10) + 7*7	↑ 100+81+64+49=294
GUR(8, 10)	↓ GUR(9,10) + 8*8	↑ 100+81+64=245
GUR(9, 10)	↓ GUR(10,10) + 9*9	↑ 100+81=181
GUR(10, 10)	10*10	↑ 100

Two Half	rekursif call	return value		rekursif call	return value
TF(5,10)	↓ TF(5,7) + TF(8,10)	↑ 110+245=355			
TF(5,7)	↓ TF(5,6) + TF(7,7)	↑ 61+49=110	TF(8,10)	↓ TF(8,9)+TF(10,10)	↑ 145+10*10=245
TF(5,6)	↓ TF(5,5) + TF(6,6)	↑ 25+36=61	TF(8,9)	↓ TF(8,8)+TF(9,9)	↑ 64+81=145
TF(5,5)		↑ 5*5=25	TF(8,8)		↑ 8*8=64
TF(6,6)		↑ 6*6=36	TF(9,9)		↑ 9*9=81

Translasi 5.14.

Bahasa C++
<pre> #include <iostream.h> #include <conio.h> class Rekursi { friend ostream& operator<<(ostream&, Rekursi&); public: Rekursi() { m =5; n=10; } int JumlahKuadratIteratif(int, int); int GoingDownRecursion(int, int); int GoingUpRecursion(int, int); int TwoHalf(int, int); private: int m, n; }; int Rekursi::JumlahKuadratIteratif(int m, int n) { int i, jumlah; jumlah = 0; for (i=m; i<=n; i++) jumlah = jumlah+i *i; return jumlah; } int Rekursi::GoingDownRecursion(int m, int n) { if ((m >= n)) return n * n; else return GoingDownRecursion(m,n-1)+n*n; } </pre>

```

int Rekursi::GoingUpRecursion(int m, int n)
{
    if ((m >= n)) return m * m;
    else return GoingUpRecursion(m + 1, n) + m * m;
}

int Rekursi::TwoHalf(int m, int n)
{
    int mid;
    if ((m == n)) return m * m;
    else {
        mid = (m + n) / 2;
        return TwoHalf(m, mid)+TwoHalf(mid+1,n);
    }
}

ostream& operator<<(ostream& out, Rekursi& a) {
    out << "Hasil = " << a.JumlahKuadratIteratif(a.m, a.n) << endl;
    out << "Hasil = " << a.GoingDownRecursion(a.m, a.n) << endl;
    out << "Hasil = " << a.GoingUpRecursion(a.m, a.n) << endl;
    out << "Hasil = " << a.TwoHalf(a.m, a.n);
    return out;
}

main() {
    Rekursi run;
    cout << run;
    getch();
    return 0;
}

```

Studi Kasus

Dalam bisnis, banyak dijumpai bukti transaksi misalnya kuitansi. Seseorang harus bisa mengeja nilai uang dengan ucapan. Misalnya nilai uang sebesar 17.500 akan diucapkan sebagai **tujuh belas ribu lima ratus rupiah.**

Buat contoh kasus nilai uangnya :

Uraikan (dengan operasi aritmatika tertentu) komponen nilai uangnya :

Apabila ada pernyataan yang berulang kali dijumpai, kumpulkan dalam perintah loop yang sesuai

Algoritma :

Input :

Proses :

Output :

Latihan

1. Buatlah fungsi fibonacci dengan cara iteratif.
2. Buatlah fungsi fibonacci dengan 2 cara rekursif yang lain.
3. Algoritma perkalian dengan cara penjumlahan pada algoritma 5.3. belum sempurna karena belum mencakup semua kemungkinan, misalnya untuk harga b negatif. Buatlah

fungsi perkalian dengan cara penjumlahan dengan menyempurnakan algoritma 5.3. di atas.

4. Buatlah algoritma rekursif dari algoritma 5.3.
5. Diberikan suatu bilangan bulat positif. Cetaklah bilangan bulat tersebut secara terbalik, secara iteratif maupun rekursif.
6. Buatlah suatu subprogram yang mengembalikan nilai maksimum pertama (misal m_1) dan nilai maksimum kedua (misal m_2 , dan $m_1 \neq m_2$) dari array dengan n bilangan bulat.
7. Algoritma 5.5 menggunakan fungsi untuk menghitung n faktorial secara rekursif. Buatlah algoritma menghitung n faktorial dengan menggunakan prosedur !
8. **[Pengamatan : membayar barang belanja di kasir]** Amatilah kejadian di mana seorang pembeli (yang membeli barang cukup banyak) pada sebuah toko swalayan. Perhatikan apa yang dilakukan pembeli dan apa yang dilakukan oleh kasir ketika mengambil barang belanjaan untuk menghitung berapa harga keseluruhan yang harus dibayar pembeli. Laporkan konsep counter dan konsep total yang anda temukan pada kejadian tersebut.

Konsep counter :

- a.
- b.
- c.

Konsep total :

- a.
- b.
- c.

Langkah yang dilakukan saat belanja :

- a. mulai : kondisi awal sebelum belanja

- b. proses : kegiatan apa saja yang dilakukan saat belanja

- c. akhir : berapa uang yang diserahkan ke kasir, berapa jumlah barang yang dibeli beserta jumlah uang yang harus dibayar dan uang kembalian (jika ada).

9. **[Proyek]** Buatlah suatu class yang akan mengeksplorasi kalimat. Kalimat dapat disimpan dalam bentuk array (kumpulan) karakter.

- a. masukan user diterima huruf demi huruf sampai diakhiri tanda akhir baris (sentinel)
- b. buat method untuk menghitung statistik :
 - i. huruf hidup (vokal)
 - ii. huruf mati (konsonan)
 - iii. banyak kata
 - iv. huruf terbanyak dalam kalimat.
- c. buat method untuk mengubah huruf awal setiap kata menjadi huruf besar

```
class Kalimat {  
    friend  
    friend  
public :
```

```
private :
```

```
};
```

```
Operator overloading input :
```

Operator overloading output :

Method statistik :

Uji Pemrograman

1. Pernyataan do – while. Gunakan pernyataan do – while untuk menghitung harga berikut.
Nilai n dimasukkan oleh user lewat keyboard. Tuliskan di sebelah kanan setiap lambang

a.
$$\sum_{i=1}^n i^2$$

b.
$$\sum_{i=1}^n i + 2$$

c.
$$\left(\sum_{i=1}^n i \right) \left(\sum_{j=1}^n j \right)$$

2. Tulis kembali poin 1 dengan menggunakan pernyataan for.

Project

Buat program kalkulator sientifik. Selain operasi $+$, $-$, $*$, $/$ dan $\%$, sekurang-kurangnya juga termasuk fungsi trigonometri (baik menggunakan radian maupun derajat).

BAB 6

SUB PROGRAM OOP DAN REKURSI

6.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan berbagai algoritma yang melibatkan perulangan baik iterative maupun rekursif.

Pada dasarnya, manusia adalah makhluk yang lemah dan berkemampuan terbatas. Untuk mengerjakan sesuatu hal yang besar dan kompleks, manusia membagi pekerjaan menjadi bagian yang lebih kecil yang memungkinkan untuk diselesaikan. Bagian yang lebih kecil dan telah diselesaikan itu disebut dengan modul. Dalam algoritma pun juga terdapat paradigma untuk membagi pekerjaan besar menjadi pekerjaan yang lebih kecil. Prinsip ini dikenal dengan *divide and conquer* (bagi lalu selesaikan).

6.2 Prosedur (Procedure)

Prosedur dikategorikan sebagai sub algoritma (atau subprogram) yang menghasilkan output lebih dari satu atau hanya sekedar melakukan tindakan tertentu.

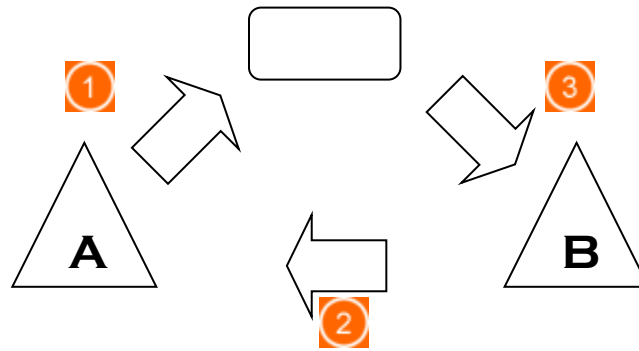
Kasus 5.1.

Buatlah prosedur untuk menukar nilai dari dua variabel.

Analisis :

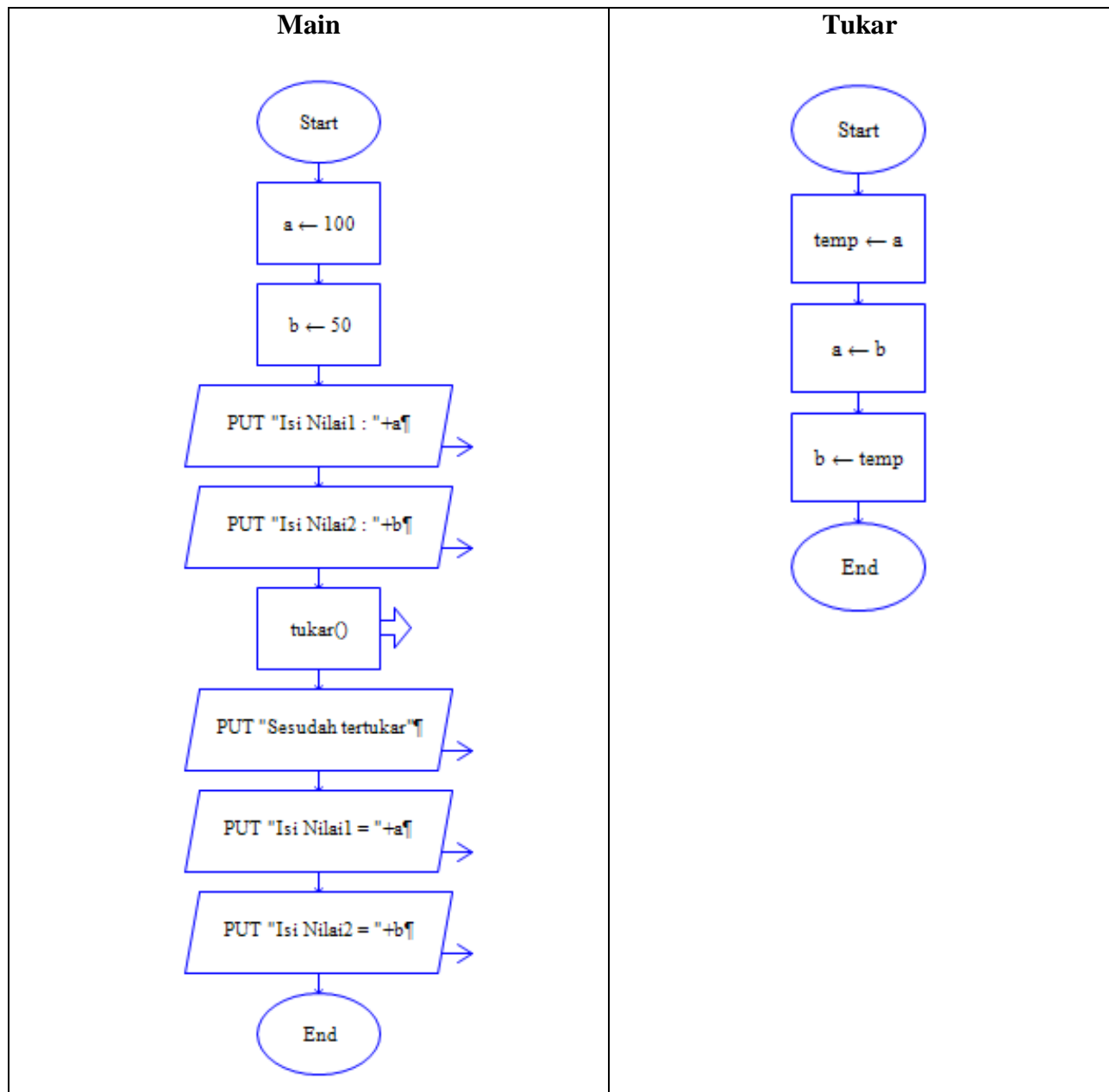
Untuk memindah posisi huruf A pada posisi huruf B dan sebaliknya, maka kita perlu tempat sementara (temp) sebelum huruf A menempati posisi huruf B. Karena kita tidak bisa begitu saja menaruh huruf ke posisi huruf B, yang akan berakibat huruf B akan “hilang”. Langkah-langkahnya adalah sebagai berikut :

TEMP



No.	Langkah	Algoritma
1.	temp diisi A	temp \leftarrow A
2.	(A kosong) A diisi B	A \leftarrow B
3.	(B kosong) B diisi temp	B \leftarrow temp

Flowchart 5.1



Algoritma 5.1.

prosedur tukar (input/output a : integer; b : integer) { menukar isi dua nilai a menjadi nilai b, demikian pula sebaliknya }
Deklarasi temp : integer
Deskripsi temp ← a a ← b b ← temp

Translasi 5.1.

Bahasa C++

```
#include <iostream.h>

void tukar (int *a, int *b)
{ int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

main() {
  int a = 100, b = 50;
  cout << "Sebelum Tukar\n";
  cout << "Isi Nilai1  = " << a << endl;
  cout << "Isi Nilai2  = " << b << endl;
  tukar (&a,&b);
  cout << "Sesudah Tukar\n";
  cout << "Isi Nilai1  = " << a << endl;
  cout << "Isi Nilai2  = " << b << endl;
  return 0;
}
```

Ada baiknya, bila kita akan memanipulasi suatu data “pribadi” yaitu data yang tidak sembarang subprogram dapat mengaksesnya, kita mulai menggunakan tipe data abstrak yang dalam C++ menggunakan **class**. Class akan mempunyai data dan *member function*. Translasi 5.1. di atas dalam C++ dapat menggunakan class sebagaimana berikut ini.

```
#include <iostream.h>

class data {
private :
  int a, b;
public :
  data();
  void tukar();
  void cetak();
};

data::data()
{
  a = 100; b = 50;
}

void data::tukar()
{ int temp;
  temp = a;
  a = b;
  b = temp;
}
```

```

void data::cetak()
{
    cout << "Isi Nilai1  = " << a << endl;
    cout << "Isi Nilai2  = " << b << endl;
}

main() {
    data angka;
    cout << "Sebelum Tukar\n";
    angka.cetak();
    angka.tukar ();
    cout << "Sesudah Tukar\n";
    angka.cetak();
    return 0;
}

```

Integer a dan b di atas tidak dapat diakses oleh fungsi yang bukan *member function*. Sehingga diperlukan fungsi-fungsi pendukung untuk memanipulasi dan menampilkan data. Konstruktor digunakan untuk memberikan nilai awal variabel a dan b.

6.3 Fungsi (Function)

Sebagaimana pengertian fungsi dalam matematika, fungsi dalam algoritma adalah sub program yang menghasilkan satu nilai. Perbedaan penulisan fungsi dalam bahasa Pascal dan bahasa C adalah sebagai berikut :

Karakteristik	Bahasa Pascal	Bahasa C
mulai dengan ...	keyword function	tipe hasil
nilai akhir dari fungsi	disimpan pada nama fungsi	return(hasil akhir)
Prototype	function nama(deklarasi variabel) : tipe_hasil;	tipe_hasil nama(deklarasi variabel)
Contoh	function rata(x : larik; n : integer) : real;	float rata(larik x, int n)

Fungsi dipanggil dengan dua cara :

1. langsung digunakan, misalnya pada saat mencetak hasil
2. di-assign ke suatu variabel

Bahasa Pascal	Bahasa C++
writeln('Hasil hitung rata-rata = ',rata(A,n));	cout << "Hasil hitung rata-rata = " << rata(A,n);
hitung_rata := rata(A, n);	hitung_rata = rata(A, n);

6.4 Parameter Dalam Subprogram

Dalam subprogram dapat dideklarasikan variabel sendiri, yang nantinya akan digunakan. Variabel yang dideklarasikan dalam subprogram bersifat **lokal**, artinya hanya dikenal dalam procedure itu sendiri, dan tidak dikenal di luar procedure. Ada 2 jenis parameter yang dideklarasikan dalam subprogram, yaitu

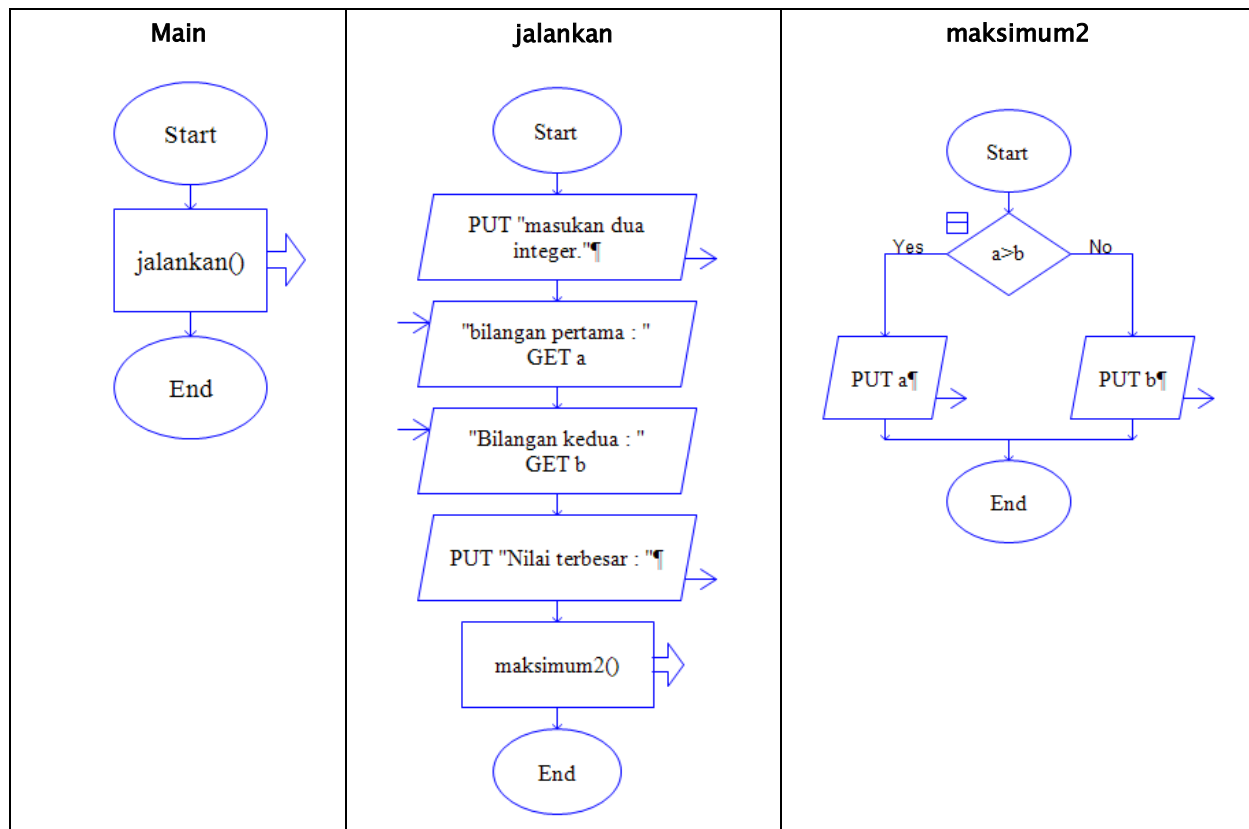
1. parameter nilai (value parameter) : variabel yang dikirimkan tidak mengalami perubahan sekeluar dari subprogram)
2. parameter variabel (variable parameter) : variabel yang dikirimkan tidak mengalami perubahan sekeluar dari subprogram)

Jenis parameter	Bahasa Pascal	Bahasa C++
parameter nilai	----	menggunakan const
parameter variable	menggunakan var	melalui pointer

Kasus 5.2.

Buatlah fungsi yang menentukan nilai terbesar dari 2 bilangan bulat.

Flowchart 5.2.



Algoritma 5.2.

Fungsi maksimum2(input a, b : integer) : integer

Deskripsi

if (a>b) then maksimum2 \leftarrow a
else maksimum2 \leftarrow b

Translasi 5.2.

Bahasa Pascal	Bahasa C
<pre>program nilai_maksimum_2_bilangan; uses wincrt; var x, y : integer; function maksimum2(a,b : integer) : integer; begin if (a>b) then maksimum2 := a else maksimum2 := b; end; begin writeln('Masukkan dua integer : '); write('Bilangan pertama : '); readln(x); write('Bilangan kedua : '); readln(y); write('Nilai terbesar:',maksimum2(x,y)); end.</pre>	<pre>#include <iostream.h> int maksimum2(int a, int b) { if (a>b) return(a); else return(b); } main() { int x, y; cout << "Masukkan dua integer. \n"; cout << "Bilangan pertama : "; cin >> x; cout << "Bilangan kedua : "; cin >> y; cout << "Nilai terbesar : " << maksimum2(x,y); return 0; }</pre>

Bila menggunakan class, kita dapat membuat program dalam bahasa C++ berikut :

```
#include <iostream.h>
class data {
public :
    int a, b;
public :
    data();
    void tukar();
    void cetak();
    int maksimum2(int a, int b);
    void jalankan();
};

data::data()
{
    a = 100; b =50;
}

int data::maksimum2(int a, int b)
{
    if (a>b) return(a);
```

```

    else return(b);
}

void main() {
    data angka;
    angka.jalankan();
}

void data::jalankan()
{
    cout << "Masukkan dua integer. \n";
    cout << "Bilangan pertama : "; cin >> a;
    cout << "Bilangan kedua : "; cin >> b;
    cout << "Nilai terbesar : " << maksimum2(a, b);
}

```

Bila baris untuk memasukkan data dari keyboard dihilangkan, data dari konstruktor default akan digunakan. Untuk selanjutnya, pembaca dipersilahkan untuk membuat translasi ke bahasa pemrograman berikutnya dengan menggunakan class.

Translasi 5.5.

Bahasa Pascal	Bahasa C
<pre> program Faktorial_Fungsi_Rekursif; uses wincrt; var bil : integer; function faktorial(N : integer):longint; begin if (N = 0) or (N = 1) then faktorial := 1 else faktorial := N * faktorial(N-1); end; begin write (' Masukkan integer n : '); readln(bil); writeln (bil,'!= ',faktorial(bil)); end. </pre>	<pre> #include <iostream.h> long faktorial(int n) { if ((n==0) (n==1)) return(1); else return (n*faktorial(n-1)); } main() { int n; long hasil; cout << "Masukkan integer n : "; cin >> n; hasil = faktorial(n); cout << "Nilai " << n << "!= " << hasil; return 0; } </pre>

Latihan

Buatlah procedure untuk:

1. Menghitung ganjil dan genap
2. Menghitung luas persegi panjang
3. Menghitung keliling persegi panjang
4. Menghitung volum lingkaran
5. Menhitung keliling lingkaran
6. Menghitung luas segitiga
7. Menghitung keliling segitiga
8. Menghitung tinggi segitiga
9. Menghitung kombinasi
10. Menghitung permutasi

BAB 7

ARRAY SATU DIMENSI

7.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan berbagai algoritma menggunakan array (variabel berindeks) baik satu ataupun dua dimensi.

7.2 Pendahuluan

Array adalah struktur data yang mengandung type data yang mempunyai type sama. Suatu array adalah sekelompok memori yang berhubungan. Array mempunyai nama dan type yang sama. Untuk merujuk lokasi tertentu atau elemen dalam array; nama array dan angka posisi (disebut subscript atau indeks) dari elemen tersebut dalam array. Sebagai contoh :

Algoritmik	Bahasa C++
$c[2] \leftarrow 2$	$c[2] = 2;$

adalah cara mengisi pada indeks 2 array c.

Array c di bawah ini mengandung 5 elemen. Elemen pertama array adalah elemen kenol (bukan ke-1). Bilangan posisi yang diapit [] dinamakan subscript. Subscript harus berupa integer atau ekspresi integer.

Nama array	
c[1]	-45
c[2]	6
c[3]	0
c[4]	72
c[5]	1543
↑ bilangan posisi	

Jika suatu program menggunakan ekspresi sebagai subscript maka ekspresi dievaluasi dulu untuk menentukan subscript. Jika $a=1$ dan $b=3$ maka $c[a+b]$ adalah array c yang bernilai 72.

7.3 Deklarasi Array

Deklarasi array ditentukan dengan tipe dari setiap elemen dan banyaknya elemen yang diperlukan oleh setiap array sehingga komputer mempersiapkan sejumlah memori. Deklarasi array adalah sebagai berikut :

Algoritmik	Bahasa C++
c : array [1..5] of integer	int c[5];

memberitahu komputer untuk menyediakan 5 elemen integer array c. Array bisa saja dideklarasikan untuk berisi tipe data yang lain. Sebagai contoh, array tipe char dapat digunakan untuk menyimpan string karakter.

7.4 Membaca elemen array

Oleh karena indeks array umumnyaurut teratur, biasanya digunakan perulangan for untuk mengakses elemen array. Perhatikan perbedaan dialek dalam bahasa Pascal dan bahasa C di bawah ini. Diasumsikan :

type larik = array [1..20] of integer;

Bahasa C++
<pre>void baca_data(int A[], int n) { int i; for (i = 0; i < n; i++) { cout << "Data ke-%d : " << i+1; cin >> A[i]; } }</pre>

Indeks dimulai dari 0 sampai dengan (n-1). Default parameternya adalah **berubah**. Sehingga dalam bahasa Pascal, bila diinginkan adanya perubahan pada array perlu ditambahkan **var**. Sedangkan dalam bahasa C bila tidak diinginkan perubahan dalam array perlu ditambahkan **const**.

Selain dimasukkan lewat keyboard, elemen array juga dapat diberi nilai awal. Perhatikan contoh program berikut :

Bahasa C++
<pre>#include <iostream.h></pre>

```

int data[3];
void main()
{
    data[0] = 5;
    data[0] = 10;
    data[0] = 15;
    int jumlah;
    jumlah = data[0] + data[1] + data[2];
    cout << "Jumlah 3 data : " << jumlah;
}

```

Kita juga bisa menginisialisasi isi array. Apabila jumlah elemen dalam deklarasi array lebih besar daripada banyaknya nilai awal, sisa tempat dalam array akan diisi dengan 0.

Bahasa C++
<pre> #include <iostream.h> int data[10] = { 5, 10, 15}; void main() { int jumlah = 0; for (int i = 0 ; i<3; i++) jumlah = jumlah + data[i]; cout << "Jumlah 3 data : " << jumlah; } </pre>

7.5 Mencetak Elemen Array

Pada subprogram dalam bahasa C++ di bawah ini menggunakan kualifier **const** karena untuk mencetak data tidak diperlukan adanya perubahan pada data tersebut.

Bahasa C++
<pre> void cetak_data(const int A[], int n) { int i; for (i = 0; i < n; i++) cout << A[i] >> " "; cout << "\n"; } </pre>

Pada dasarnya, array merupakan fasilitas untuk mengelompokkan data yang bertipe sama. Ini berguna bila data tersebut dapat dimanfaatkan kembali. Sebagai contoh akan diperlihatkan pada kasus berikut ini.

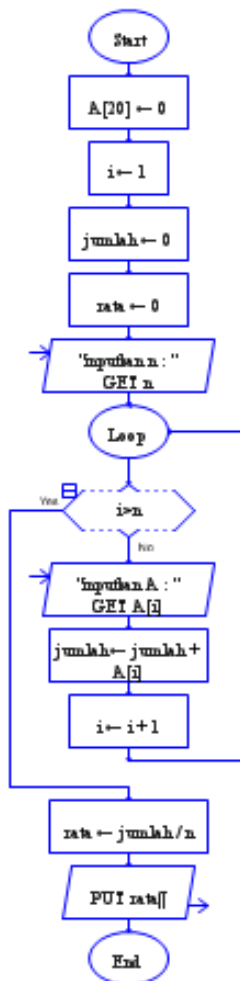
Kasus 6.1.

Carilah rata-rata dari n bilangan bulat dengan menggunakan array.

Algoritma 6.1.

Fungsi rata (input x : array [1..10] of integer, n : integer) : real {Diberikan n data kemudian dicari rata-ratanya}
Deklarasi i, jumlah : integer
Deskripsi jumlah \leftarrow 0 for i \leftarrow 1 to n do jumlah \leftarrow jumlah + x[i] endfor rata \leftarrow jumlah/n

Flowchart 6.1



Translasi 6.1.

1.	#include <iostream.h>
2.	class HitungRATA {
3.	friend ostream& operator<<(ostream&, HitungRATA&);

4.	friend ostream& operator>>(ostream&, HitungRATA&);
5.	public:
6.	float rata();
7.	private:
8.	int n; // banyaknya data
9.	int A[20]; // array untuk menyimpan data
10.	};
11.	
12.	ostream& operator>>(ostream& in, HitungRATA& a)
13.	{
14.	cout << "Banyaknya data : ";
15.	in >> a.n;
16.	for (int i = 0; i < a.n; i++)
17.	{
18.	cout << "Masukkan data ke- : " << i+1 << " > ";
19.	in >> a.A[i];
20.	}
21.	return in;
22.	}
23.	
24.	float HitungRATA::rata()
25.	{
26.	float total=0;
27.	for (int i = 0; i<n; i++) total = total + A[i];
28.	return(total/n);
29.	}
30.	
31.	ostream& operator<<(ostream& out, HitungRATA& a) {
32.	out << "Rata-rata dari " << a.n
33.	<< " bilangan adalah : " << a.rata();
34.	return out;
35.	}
36.	
37.	main() {
38.	HitungRATA run;
39.	cin >> run;
40.	cout << run;
41.	return 0;
42.	}

Hampir tidak ada bedanya dengan algoritma 4.3. Karena tujuan akhirnya hanya untuk mencari nilai rata-rata dari n bilangan bulat. Tentu akan sangat berguna bila nilai rata-rata tersebut beserta datanya digunakan kembali sebagaimana kasus di bawah ini.

Kasus 6.2.

Carilah nilai deviasi standar dari n buah data.

Analisis :

Rumus deviasi standar adalah :

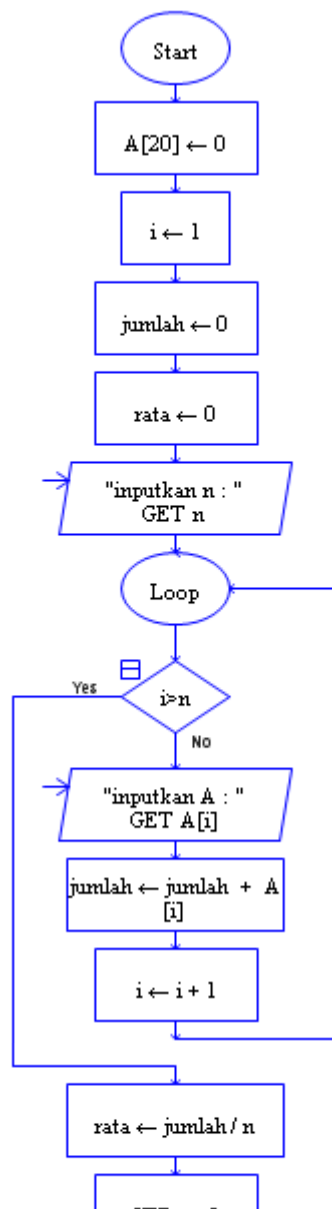
$$std = \sqrt{\frac{(x_i - \bar{x})^2}{n-1}}$$

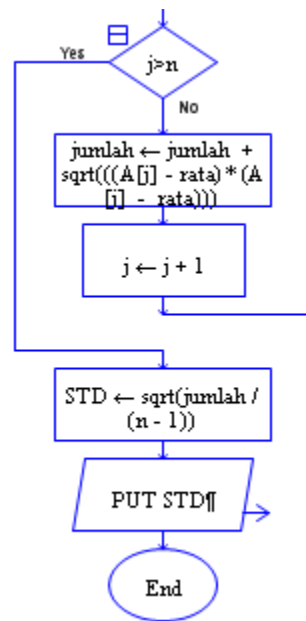
Terlihat bahwa nilai rata-rata dan datanya digunakan kembali.

Algoritma 6.2.

Fungsi std(input x : array [1..10] of integer, n : integer, rata : real) : real {Diberikan n data kemudian dicari rata-ratanya}	
Deklarasi i, jumlah : integer	
Deskripsi jumlah \leftarrow 0 for i \leftarrow 1 to n do jumlah \leftarrow jumlah + sqr(x[i]-rata) endfor std \leftarrow sqrt(jumlah/(n-1))	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: auto;"> x[i] dipakai kembali </div>

Flowchart 6.2





Translasi 6.2.

1.	#include <iostream.h>
2.	#include <math.h>
3.	class HitungStatistik {
4.	friend ostream& operator<<(ostream&, HitungStatistik&);
5.	friend istream& operator>>(istream&, HitungStatistik&);
6.	public:
7.	float rata();
8.	float STD();
9.	private:
10.	long sqr(int n) { return(n*n); }
11.	int n; // banyaknya data
12.	int A[20]; // array untuk menyimpan data
13.	};
14.	
15.	istream& operator>>(istream& in, HitungStatistik& a)
16.	{
17.	cout << "Banyaknya data : ";
18.	in >> a.n;
19.	for (int i = 0; i < a.n; i++)
20.	{
21.	cout << "Masukkan data ke- : " << i+1 << " > ";
22.	in >> a.A[i];
23.	}
24.	return in;
25.	}
26.	
27.	float HitungStatistik::rata()

28.	{
29.	float total=0;
30.	for (int i = 0; i<n; i++) total = total + A[i];
31.	return(total/n);
32.	}
33.	
34.	float HitungStatistik::STD ()
35.	{ float rerata = rata();
36.	float jumlah=0.0;
37.	for (int i = 0; i<n; i++)
38.	jumlah = jumlah + sqr(A[i] - rerata);
39.	return(sqrt (jumlah/(n-1)));
40.	}
41.	
42.	ostream& operator<<(ostream& out, HitungStatistik& a) {
43.	out << "Rata-rata dari " << a.n
44.	<< " bilangan adalah : " << a.rata() << endl;
45.	out << "Standar deviasi= " << a.STD();
46.	return out;
47.	}
48.	
49.	Main() {
50.	HitungStatistik run;
51.	cin >> run;
52.	cout << run;
53.	return 0;
54.	}

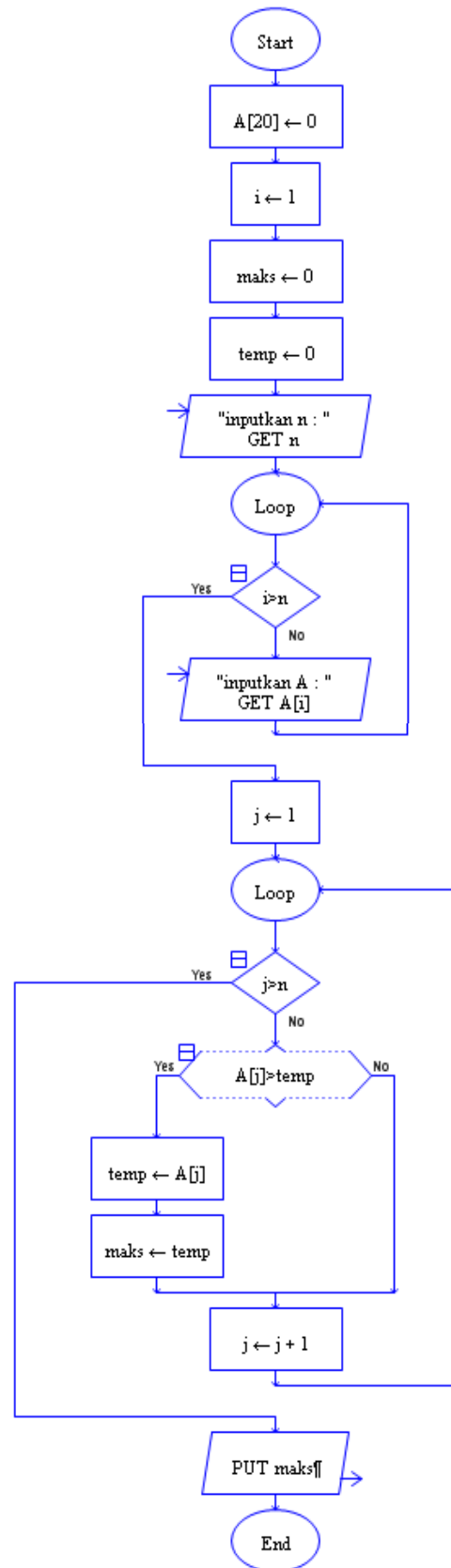
Kasus 6.3.

Dengan menggunakan algoritma 5.2., buatlah algoritma untuk menentukan nilai maksimum dari n bilangan.

Algoritma 6.3.

function maksimum(input data : array [1..10] of integer, n : integer):integer
Deklarasi i, temp : integer
Deskripsi temp \leftarrow data[1] for i \leftarrow 2 to n do temp \leftarrow maksimum2(temp, data[i]) maksimum \leftarrow temp

Flowchart 6.3.



Translasi 6.3.

1.	#include <iostream.h>
2.	#include <math.h>
3.	class HitungStatistik {
4.	friend ostream& operator<<(ostream&, HitungStatistik&);
5.	friend istream& operator>>(istream&, HitungStatistik&);
6.	public:
7.	int maksimum();
8.	private:
9.	int maksimum2(int, int);
10.	int n; // banyaknya data
11.	int A[20]; // array untuk menyimpan data
12.	};
13.	
14.	istream& operator>>(istream& in, HitungStatistik& a)
15.	{
16.	cout << "Banyaknya data : ";
17.	in >> a.n;
18.	for (int i = 0; i < a.n; i++)
19.	{
20.	cout << "Data ke- : " << i+1 << " > ";
21.	in >> a.A[i];
22.	}
23.	return in;
24.	}
25.	
26.	int HitungStatistik::maksimum2(int a, int b)
27.	{ if (a>b) return(a);
28.	else return(b);
29.	}
30.	
31.	int HitungStatistik::maksimum()
32.	{ int temp = A[0];
33.	for (int i = 1; i<n; i++)
34.	temp = maksimum2(temp, A[i]);
35.	return(temp);
36.	}
37.	
38.	ostream& operator<<(ostream& out, HitungStatistik& a) {
39.	out << "Nilai terbesar: " << a.maksimum();
40.	return out;
41.	}

42.	
43.	main() {
44.	HitungStatistik run;
45.	cin >> run;
46.	cout << run;
47.	return 0;
48.	}

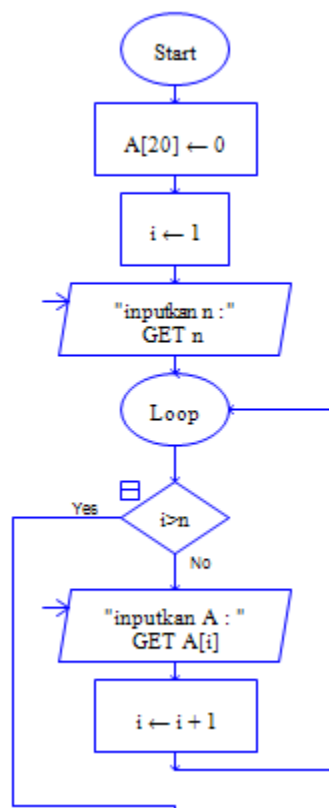
Kasus 6.4.

Buatlah algoritma untuk menentukan nilai maksimum dan minimum dari n bilangan.

Algoritma 6.4.

procedure maks_min(input data : larik; n : integer; output m1, m2 : integer)
Deklarasi i : integer
Deskripsi m1 \leftarrow data[1] m2 \leftarrow data[1] for i \leftarrow 2 to n do if (data[i] > m1) then m1 \leftarrow data[i]; if (data[i] < m2) then m2 \leftarrow data[i]; endfor

Flowchart 6.4.



9.	int m1, m2; // m1 untuk maksimum, m2 untuk min
10.	int n; // banyaknya data
11.	int A[20]; // array untuk menyimpan data
12.	};
13.	
14.	istream& operator>>(istream& in, HitungStatistik& a)
15.	{
16.	cout << "Banyaknya data : ";
17.	in >> a.n;
18.	for (int i = 0; i < a.n; i++)
19.	{
20.	cout << "Data ke- : " << i+1 << " > ";
21.	in >> a.A[i];
22.	}
23.	return in;
24.	}
25.	
26.	void HitungStatistik::maks_min()
27.	{ m1 = m2 = A[0];
28.	for (int i=1; i<n; i++)
29.	{
30.	if (A[i] > m1) m1 = A[i];
31.	if (A[i] < m2) m2 = A[i];
32.	}
33.	}
34.	
35.	ostream& operator<<(ostream& out, HitungStatistik& a) {
36.	a.maks_min();
37.	out << "Nilai terbesar : " << a.m1 << endl;
38.	out << "Nilai terkecil : " << a.m2;
39.	return out;
40.	}
41.	
42.	main() {
43.	HitungStatistik run;
44.	cin >> run;
45.	cout << run;
46.	return 0;
47.	}

Kasus 6.5.

Tentukan modus dari n buah data berupa bilangan bulat, di mana besar datanya antara 1 sampai dengan 10.

Analisis :

Modus adalah bilangan atau data yang paling sering muncul. Dengan kata lain, frekuensi data terbesar lah yang dicari. Jadi langkah penyelesaian masalahnya adalah :

1. setiap jenis data dihitung frekuensi kemunculannya
2. dari frekuensi-frekuensi tersebut dicari frekuensi terbesarnya.

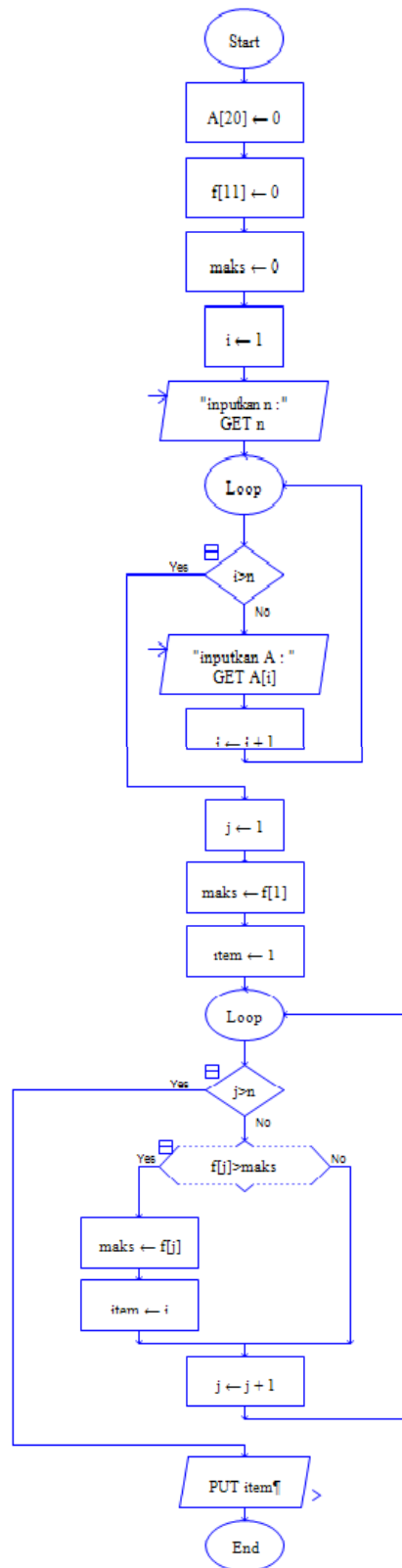
Algoritma 6.5.a.

procedure maksimum(data : larik; n : integer; output maks, item : integer) { procedure ini hasil modifikasi dari algoritma (...) karena selain nilai maks dari larik data, juga perlu diketahui besar datanya item }
Deklarasi i : integer
Deskripsi maks \leftarrow data[1] item \leftarrow 1 for i \leftarrow 2 to n do if (data[i] > maks) then maks \leftarrow data[i]; item \leftarrow i; endif endfor

Algoritma 6.5.b.

procedure frekuensi(data : larik; n : integer; output f : larik) { data akan diambil nilai frekuensi f-nya }
Deklarasi i : integer
Deklarasi for i \leftarrow 1 to n do f[data[i]] := f[data[i]] + 1 { dengan prinsip memasukkan bola ke keranjang yang sesuai dengan nomornya } endfor

Flowchart 6.5.



Translasi 6.5.

1.	#include <iostream.h>
2.	#include <math.h>
3.	class HitungStatistik {
4.	friend ostream& operator<<(ostream&, HitungStatistik&);
5.	friend istream& operator>>(istream&, HitungStatistik&);
6.	public:
7.	HitungStatistik();
8.	void hitung_modus();
9.	private:
10.	void maksimum();
11.	void frekuensi();
12.	int maks, item;
13.	int n; // banyaknya data
14.	int A[20]; // array untuk menyimpan data
15.	int f[11]; // array untuk menampung frekuensi
16.	};
17.	
18.	HitungStatistik::HitungStatistik()
19.	{ for (int i=0; i<20; i++) f[i] = 0; }
20.	
21.	istream& operator>>(istream& in, HitungStatistik& a) {
22.	cout << "Banyaknya data : ";
23.	in >> a.n;
24.	for (int i = 0; i < a.n; i++) {
25.	cout << "Data ke- : " << i+1 << " > ";
26.	in >> a.A[i];
27.	}
28.	return in;
29.	}
30.	
31.	void HitungStatistik::maksimum()
32.	{
33.	maks = f[0];
34.	item = 1;
35.	for (int i=0; i<n; i++)
36.	if (f[i] > maks) {
37.	maks = f[i];
38.	item = i;
39.	}
40.	cout << "Modus = " << item;
41.	}

42.	
43.	void HitungStatistik::frekuensi()
44.	{
45.	for (int i=1; i<n; i++) ++f[A[i]];
46.	}
47.	
48.	void HitungStatistik::hitung_modus() {
49.	cout << "Frekuensi running\n";
50.	frekuensi();
51.	maksimum();
52.	}
53.	
54.	ostream& operator<<(ostream& out, HitungStatistik& a) {
55.	cout << "Mulai ...\n";
56.	a.hitung_modus();
57.	out << "Nilai modus : " << a.item;
58.	return out;
59.	}
60.	
61.	main() {
62.	HitungStatistik run;
63.	cin >> run;
64.	cout << run;
65.	return 0;
66.	}

7.6 String

String dapat dipandang sebagai sederetan karakter (misalnya dengan panjang 255) atau array [1..255] of char. Sedangkan string dengan panjang satu disebut sebagai karakter (char). Dalam C++, string dapat dipandang sebagai array dari karakter. Setiap elemen dalam string dapat diakses dengan operator []. Perhatikan program berikut ini.

```
#include <iostream.h>
#include <conio.h>
void main() {
    char S[10] = "Algoritma";
    for (int t = 0; t < strlen(S); t++)
        cout << S[t] << " ";
}
```

Pada program di atas, setiap karakter yang ada pada array S dicetak ke layar. Dengan demikian, kita bisa melakukan manipulasi pada setiap elemen array.

Kasus 6.6.

Buatlah suatu algoritma untuk mengecek apakah suatu kata atau kalimat merupakan palindrom atau tidak. Palindrom adalah suatu kata atau kalimat yang dibaca dari kiri ke kanan sama dengan kalau dibaca dari kanan ke kiri.

Analisis :

Misalnya kata yang akan dicek adalah “kasur rusak”. Maka huruf pertama dicek, apakah sama dengan huruf pertama terakhir atau tidak, demikian pula untuk huruf kedua dicek dengan huruf kedua terakhir sampai huruf yang di tengah kalimat. Bila pengecekan selalu sama maka kalimat tersebut adalah palindrom. Jika terdapat satu huruf saja yang tidak sama, kalimat tersebut bukanlah palindrom.

Algoritma 6.6.

function palindrom(input s : string) : boolean
Deklarasi i, pj : integer
Deskripsi palindrom \leftarrow true { diasumsikan benar, dibuktikan dengan penyangkalan } pj \leftarrow length(s) for i \leftarrow 1 to (pj div 2) do if (s[i] \neq s[pj-i+1]) then palindrom \leftarrow false

Translasi 6.6.

1.	#include <iostream.h>
2.	#include <string.h>
3.	#include <conio.h>
4.	class Palindrom {
5.	friend ostream& operator<<(ostream&, Palindrom&);
6.	friend istream& operator>>(istream&, Palindrom&);
7.	public:
8.	int cek_palindrom();
9.	private:
10.	char *s;
11.	};
12.	
13.	ostream& operator<<(ostream& out, Palindrom& a) {
14.	if (a.cek_palindrom())
15.	out << a.s << " adalah palindrom";
16.	else out << a.s << " bukan palindrom";
17.	return out;
18.	};

19.	
20.	istream& operator>>(istream& in, Palindrom& a) {
21.	cout << "Masukkan sebuah kalimat : ";
22.	in >> a.s;
23.	return in;
24.	};
25.	
26.	int Palindrom::cek_palindrom()
27.	{ int i, pj;
28.	pj = strlen(s);
29.	for (i=0; i<=pj/2 ;i++)
30.	if (s[i] != s[pj-i-1]) return 0;
31.	return 1;
32.	}
33.	
34.	main() {
35.	Palindrom kata;
36.	cin >> kata;
37.	cout << kata;
38.	getch();
39.	return 0;
40.	}

E.1. Subprogram yang berkaitan dengan string

Bahasa C++	Arti
strlen	mengembalikan panjang string
strcpy	menyalin string
strcat	menggabung string
	menghapus sebagian string
	menyisipkan suatu string ke dalam string
strstr	mengembalikan posisi substring dalam string
toupper	mengubah menjadi huruf besar

E. 2. Penggunaan

Translasi 6.7. Mencoba strlen

Bahasa C++
<pre>#include <iostream.h> #include <string.h> main() { char *kata = "Algoritma"; int p;</pre>

```
p = strlen(kata);
cout << "Panjang kata : " << kata << " = " << p;
return 0;
}
```

Translasi 6.8. Mencoba strcpy

Bahasa C++
<pre>#include <iostream.h> #include <string.h> typedef char *string; string copy(string str, int dari, int panjang) { char buf[50]; strcpy(buf,str+dari-1); buf[panjang] = 0; return buf; } void main() { char *kata="Algoritma", *hasil; hasil = copy(kata, 3, 5); cout << "Hasil copy : " << hasil; }</pre>

Translasi 6.9. Mencoba strcat

Bahasa C++
<pre>#include <iostream.h> #include <string.h> main() { char *kata1 = "Algoritma "; char *kata2 = "Pemrograman"; strcat(kata1, kata2); cout << "Kata gabungan : %s",kata1); return 0; }</pre>

Translasi 6.10. Mencoba delete

Bahasa C++
<pre>#include <iostream.h> #include <string.h> #define delete(str,posisi,panjang) strcpy(str+posisi-1,str+posisi+panjang-1) main() { char *kata="Algoritma";</pre>

```

delete(kata, 3, 5);
cout << "Hasil setelah dihapus : %s",kata);
return 0; }

```

Translasi 6.11. Mencoba upcase

Bahasa C++
<pre> #include <iostream.h> #include <string.h> void ubah_ke_huruf_besar(char *s) { int i; for (; *s != '\0'; s++) *s = toupper(*s); } main() { char *kata="Algoritma dan Pemrograman"; ubah_ke_huruf_besar(kata); cout << "%s\n",kata); cout << "Panjang kata = %d",strlen(kata)); return 0; } </pre>

Kasus 6.7. Penyandian Kalimat

Dalam kriptografi cukup banyak algoritma untuk menyandikan teks (kalimat). Salah satunya adalah dengan cara menggeser posisi huruf. Sebagai contoh, huruf 'A' bila digeser ke kanan sebanyak 5 tempat akan menjadi huruf 'F', huruf 'B' menjadi 'G' dan seterusnya. Lima huruf terakhir melingkar kembali ke huruf awal, misalnya 'X' menjadi 'C'.

Untuk mengembalikan ke teks asli, teks yang telah digeser hurufnya ke kanan tadi, kita geser sebaliknya yaitu ke kiri sebanyak 5 kali.

Program selengkapnya adalah sebagai berikut.

```

#include <iostream.h>
#include <conio.h>
#include <cstring.h>
class Sandi {
    friend ostream& operator<<(ostream&, Sandi&);
public:
    Sandi() { kalimat = "Algoritma dan Pemrograman";}
    void enkripsi(char *);

```

```

    void dekripsi(int *);
private:
    char *kalimat;
    int *kalimatTersandi; };

void Sandi::enkripsi(char *kalimat) {
    for (int i=0; kalimat[i] != '\0'; i++)
        if (kalimat[i] != ' ')
            if ( kalimat[i]%2==0 ) kalimatTersandi[i] = kalimat[i] + 3;
            else kalimatTersandi[i] = kalimat[i] + 5;
        else kalimatTersandi[i] = 0;
    }

void Sandi::dekripsi(int *kalimatTersandi) {
    for (int i=0; kalimat[i] != '\0'; i++)
        if (kalimatTersandi[i] != 0)
            if ( kalimat[i]%2==1 ) kalimat[i] = kalimatTersandi[i] - 5;
            else kalimat[i] = kalimatTersandi[i] - 3;
        else kalimat[i] = ' ';
    }

ostream& operator<<(ostream& out, Sandi& snd) {
    cout << "Kata Asli : " << snd.kalimat;
    snd.enkripsi(snd.kalimat);
    cout << "\nSetelah disandikan : ";
    for (int i=0; i<=strlen(snd.kalimat); i++)
        out << snd.kalimatTersandi[i] << " ";
    snd.dekripsi(snd.kalimatTersandi);
    cout << "\nSetelah dikembalikan : " << snd.kalimat;
    return out;
}

void main() {
    Sandi X;
    cout << X;
}

```

Workshop

1. Buatlah algoritma dan subprogram yang cara kerjanya seperti procedure **insert** di atas.

Misalkan : kata1 = “Algoritma Pemrograman”

kata2 = “dan “

Outputnya :

kata3 = “Algoritma dan Pemrograman”

Coba lakukan secara manual huruf demi huruf dari kata2 yang disisipkan ke kata1.

Perhatikan letak indeks dari kata1 yang akan memperoleh tambahan kata2.

Cara manual :

Algoritma :

Subprogram :

2. Buatlah algoritma dan program untuk mengubah string menjadi bergantian huruf besar dan kecil. Contoh :

Algoritma dan Pemrograman → AlGoRiTmA dAn PeMrOgRaMaN

Analisis :

Perhatikan bahwa manipulasi elemen (dalam hal ini karakter) array tertuju pada indeksnya. Buat angka indeks di bawah setiap karakter di atas :

Karakter	A	l	G	o	R	i	T	m	A		d	A	n		P	e	M	r	O	g	R	a	M	a	N
Indeks																									

Cari keberaturan letak karakter yang berubah. Kemudian turunkan menjadi algoritma.

Rumus letak karakter yang berubah :

Algoritma :

3. Buatlah algoritma dan program untuk menghitung jumlah kata dalam suatu kalimat.

Analisis :

Dalam sebuah kalimat, apa yang digunakan untuk mulai menghitung satu kata ke kata lain ?

Jawab :

Buat contoh kalimat di bawah ini.

Hitung banyaknya kata dalam kalimat secara manual. Banyak kata =

Algoritma :

4. Diberikan class Vektor berikut ini.

```
class Vektor {
    friend ostream& operator<<(ostream&, Vektor&);
    friend istream& operator>>(istream&, Vektor&);
    friend class SPL;
public:
    Vektor();
    void penjumlahan_vektor(const Vektor& A, const Vektor& B);
    void perkalian_vektor(float k, const Vektor& A);
    void beri_nilaiBanyak(int);
private:
    int elemen[100];
    int banyak; };

```

Sebuah vektor yang disimpan menggunakan array, dapat dipandang sebagai vektor baris maupun vektor kolom. Method `penjumlahan_vektor` didefinisikan sebagai penjumlahan elemen-elemen Vektor A dan Vektor B pada indeks yang sama. Method `perkalian_vektor` didefinisikan sebagai perkalian Vektor A dengan skalar k. Sementara method `beri_nilaiBanyak` menunjukkan berapa banyak elemen dari suatu vektor. Lengkapi isi method di atas sehingga sesuai dengan definisi yang diberikan.

Misal :

Vektor A

2	-3	7	1	-8
---	----	---	---	----

dan

Vektor B

0	1	5	-3	5
---	---	---	----	---

Hasilnya :

Vektor C

--	--	--	--	--

```
void Vektor::beri_nilaiBanyak(int i) {
```

```
}
```

```
void Vektor::penjumlahan_vektor(const Vektor& A, const Vektor& B) {
```

```
}
```

```
void Vektor::perkalian_vektor(float k, const Vektor& A) {
```

```
}
```

Latihan

1. Buatlah algoritma dan subprogram yang cara kerjanya seperti function **pos** di atas.
2. Buatlah algoritma dan function dalam bahasa C untuk menggantikan strlen, yaitu mengetahui panjang suatu string
3. Buatlah algoritma dan function untuk mengubah huruf besar menjadi huruf kecil, kemudian buatlah function untuk mengubah string menjadi string yang hurufnya huruf besar semua.

BAB 8

SEARCHING DAN SORTING

8.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu mengembangkan dan menjelaskan berbagai algoritma pengurutan (sorting) dan pencarian (searching).

8.2 Pencarian Linier (Linear search)

Andaikan terdapat array **array** dengan banyak data sebanyak **ukuran**. Misalkan kita ingin mencari data **kunci** dalam array **array**. Prinsip pada pencarian linier, setiap data pada **array** akan dibandingkan dengan **kunci** sampai pada data yang terakhir (kasus terburuk). Bila pada posisi ke-i data sama dengan **kunci**, berarti data ditemukan pada posisi ke-i. Bila sampai akhir data, data tidak juga ditemukan berarti **kunci** tidak ada pada **array**.

Algoritma 7.1.

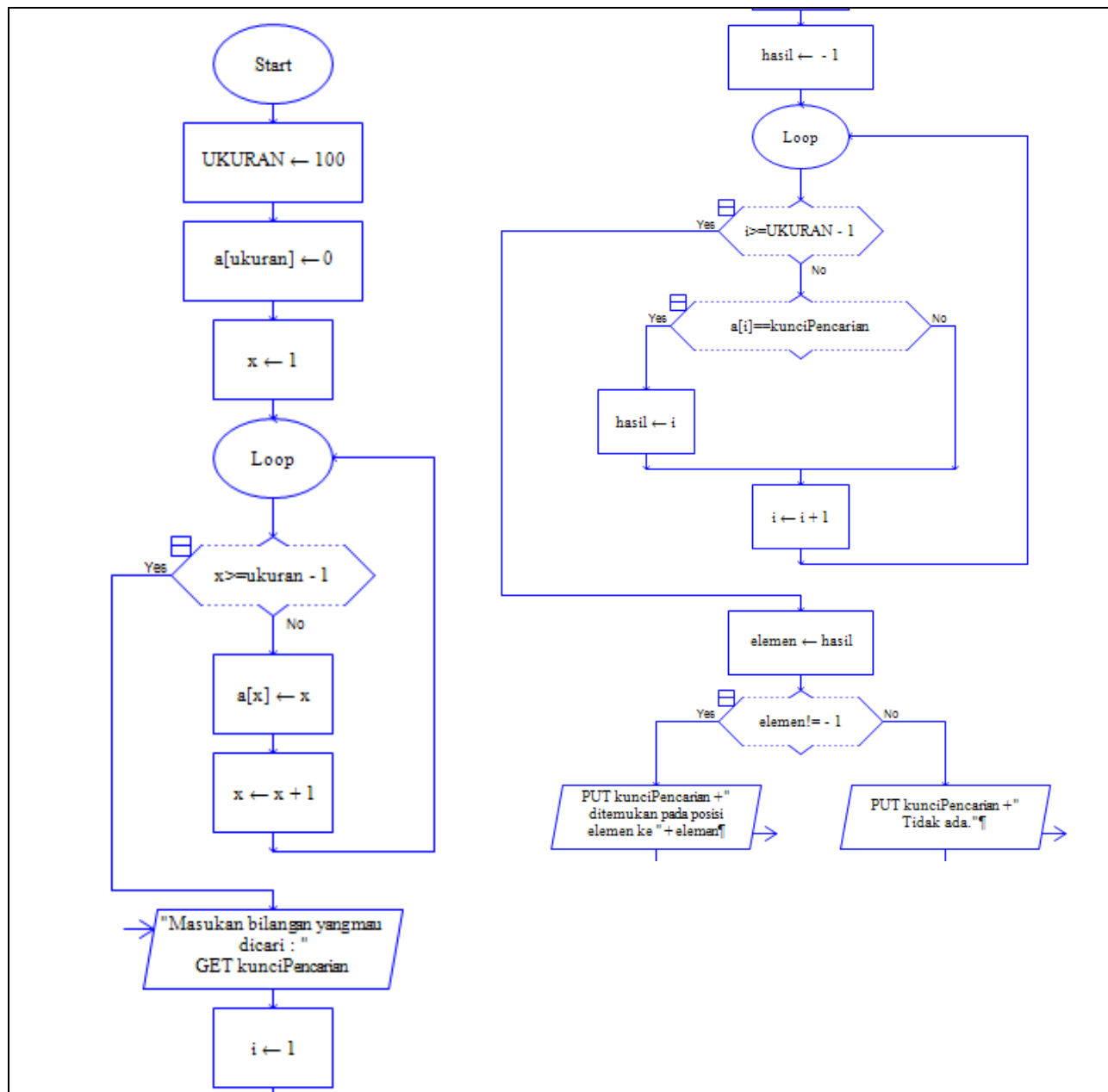
function pencarianLinier(input array : larik; kunci, ukuran : integer) : integer
Deklarasi ketemu : boolean i, n : integer
Deskripsi for i \leftarrow 1 to ukuran do if (array[i] = kunci) { Setiap elemen array dicocokkan dengan kunci } pencarianLinier \leftarrow i { Bila cocok i menunjukkan tempat dimana ditemukan } endif endfor pencarianLinier \leftarrow -1 { Bila tidak ada satupun yang cocok, -1 menunjukkan kegagalan } end

Translasi 7.1.

Bahasa C++

```
#include <iostream.h>
#define UKURAN 100
int pencarianLinier(int array[], int kunci, int n)
{   int i;
    for (i=0; i<=n-1; ++i)
        if (array[i] == kunci)
            return i;
    return -1;
}
void main(){
    int a[UKURAN], x, kunciPencarian, elemen;
    for (x=0; x<=UKURAN-1; x++) a[x] = 2*x;
    cout << "Bilangan yang mau dicari : ";
    cin >> kunciPencarian;
    elemen = pencarianLinier(a,kunciPencarian,UKURAN);
    if (elemen != -1)
        cout << kunciPencarian << " ditemukan pada posisi elemen ke " << elemen;
    else
        cout << kunciPencarian << " tidak ada.";
}
```

Flowchart 7.1



8.3 Pencarian Biner (Binary Search)

Pada algoritma pencarian biner, data sudah dalam keadaan terurut (untuk mudahnya diasumsikan urut naik). Contoh dalam kehidupan sehari-hari, seperti orang mencari nomor telepon pada buku telepon. Setiap kali pencarian, **kunci** akan selalu dibandingkan dengan data yang berada di tengah (*middle*), bila sama berarti data ketemu, bila tidak, akan “dilihat” apakah data ada di sebelah “kiri” (artinya data lebih kecil dari data di tengah) atau di sebelah “kanan” (artinya data lebih besar dari data di tengah). Bila data ada di sebelah kiri, dilakukan pencarian dengan cara yang sama (sementara data yang berada di sebelah kanan akan diabaikan). Jadi, setiap kali pencarian, data selalu “dibelah” menjadi dua bagian (biner), sampai pada “titik

tertentu” (bila sama dengan titik tengah, pencarian tidak dilakukan lagi, bila tidak, sampai pada perbandingan terakhir data juga tidak sama, berarti data tidak ditemukan pada array **array**).

Algoritma 7.2.

function pencarianBiner(input array : larik; kunci, low, high : integer) : integer
Deklarasi ketemu : boolean i, middle : integer
Deskripsi ketemu \leftarrow false while (low \leq high) and (not ketemu) do middle \leftarrow (low+high) div 2 if (kunci = array[middle]) then ketemu \leftarrow true { data pencarian = data di tengah } else if (kunci < array[middle]) then high \leftarrow middle – 1 {data akan dicari lagi di sebelah kiri} else low \leftarrow middle + 1 {data akan dicari lagi di sebelah kanan} endif endwhile if ketemu then pencarianBiner := middle else pencarianBiner := -1; endif

Translasi 7.3.

Bahasa C++
<pre> #include <iostream> using namespace std; #define UKURAN 15 #define FALSE 0 #define TRUE 1 void cetakHeader(void) { int i; cout << "\nSubscript : \n"; for (i=0;i<=UKURAN-1;i++) cout << i << " "; cout << "\n"; for (i=1; i <= 4*UKURAN; i++) cout << "-"; cout << "\n"; } void cetakBaris(int b[], int low, int mid, int high) { int i; for (i=0; i<=UKURAN-1; i++) if (i<low i>high) cout << " "; else if (i==mid) cout << "*" << b[i]; else cout << b[i] << " "; cout << "\n"; } </pre>


```

int pencarianBiner(int b[], int kunciPencarian, int low,
int high)
{
    int i, middle;
    bool ketemu = FALSE;
    while ( (low <= high) &&(!ketemu)) {
        middle = (low+high) / 2;
        cetakBaris(b, low, middle, high);
        if (kunciPencarian == b[middle]) {
            ketemu = TRUE;
            return middle; }
        else if (kunciPencarian < b[middle])
            high = middle - 1;
        else low = middle + 1;
    }
    return -1;
}

int main() {
    int a[UKURAN], i, kunci, hasil;
    for (i=0; i<=UKURAN-1; i++) a[i] = 2*i;
    cout << "Masukkan bilangan antara 0-28 : ";
    cin >> kunci;
    cetakHeader();
    hasil=pencarianBiner(a,kunci,0,UKURAN-1);
    if (hasil != -1) cout << kunci
        << " ditemukan pada posisi : "<< hasil;
    else
        cout << kunci << " tidak ditemukan";
    return 0;
}

```

Output

Masukkan bilangan antara 0-28 : 16

Subscript :

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

```

-----
0 2 4 6 8 10 12 14* 16 18 20 22 24 26 28
          16 18 20 22* 24 26 28
          16 18* 20
          16*

```

16 ditemukan dalam array pada posisi : 8

8.4 Pengantar Pengurutan (Sorting)

Pengurutan atau sorting berarti menyusun elemen-elemen dengan urutan tertentu, yaituurut naik (ascending) atauurut turun (descending). Urutan naik berarti susunan elemen dari yang terkecil sampai dengan yang terbesar, misal elemen {3, 1, 7, 5} diurutkan naik menjadi {1, 3, 5, 7}. Sebaliknya, urutan turun berarti susunan elemen dari yang terbesar sampai dengan yang terkecil, misal elemen {3, 1, 7, 5} diurutkan turun menjadi {7, 5, 3, 1}. Susunan terurut akan memberikan susunan data yang lebih berarti.

8.5 Bubble Sort

Metode pengurutan gelembung (bubble sort) mempunyai perilaku seperti gelembung di mana bila akan diurutkan naik, nilai yang besar akan naik (indeks besar) sementara nilai yang kecil akan turun (ke indeks yang kecil). Setiap data (misalnya data pertama) akan dibandingkan dengan data yang ada di sebelahnya (dari data kedua sampai selesai) . Bila data pertama tersebut lebih besar dari data yang ada pada data sesudahnya, dilakukan penukaran tempat atau posisi data. Demikian, untuk data kedua sampai dengan data terakhir dilakukan dengan cara serupa.

Sebagai ilustrasi, proses bubble sort adalah sebagai berikut :

Data awal :	[8, 4, 7, 3, 1, 2, 6, 5]	8 \leftrightarrow 4
fase 1	[4, 7, 3, 1, 2, 6, 5, 8]	7 \leftrightarrow 3
fase 2	[4, 3, 1, 2, 6, 5, 7, 8]	4 \leftrightarrow 3, 4 \leftrightarrow 1, 4 \leftrightarrow 2, 6 \leftrightarrow 5
fase 3	[3, 1, 2, 4, 5, 6, 7, 8]	3 \leftrightarrow 1, 3 \leftrightarrow 2
fase 4	[1, 2, 3, 4, 5, 6, 7, 8]	
fase 5	[1, 2, 3, 4, 5, 6, 7, 8]	
fase 6	[1, 2, 3, 4, 5, 6, 7, 8]	
fase 7	[1, 2, 3, 4, 5, 6, 7, 8]	
fase 8	[1, 2, 3, 4, 5, 6, 7, 8]	

Algoritma 7.3.

Procedure Bubble_Sort (input/output data : larik, input banyak : byte)
Deklarasi larik = array [1..100] of integer j,k : integer
Deskripsi for i \leftarrow 1 to banyak-1 do for j \leftarrow 1 to banyak-i do if data[j] > data[j + 1] then tukar(data[j], data[j + 1])

```
endif
endfor
endfor
```

Translasi 7.3.

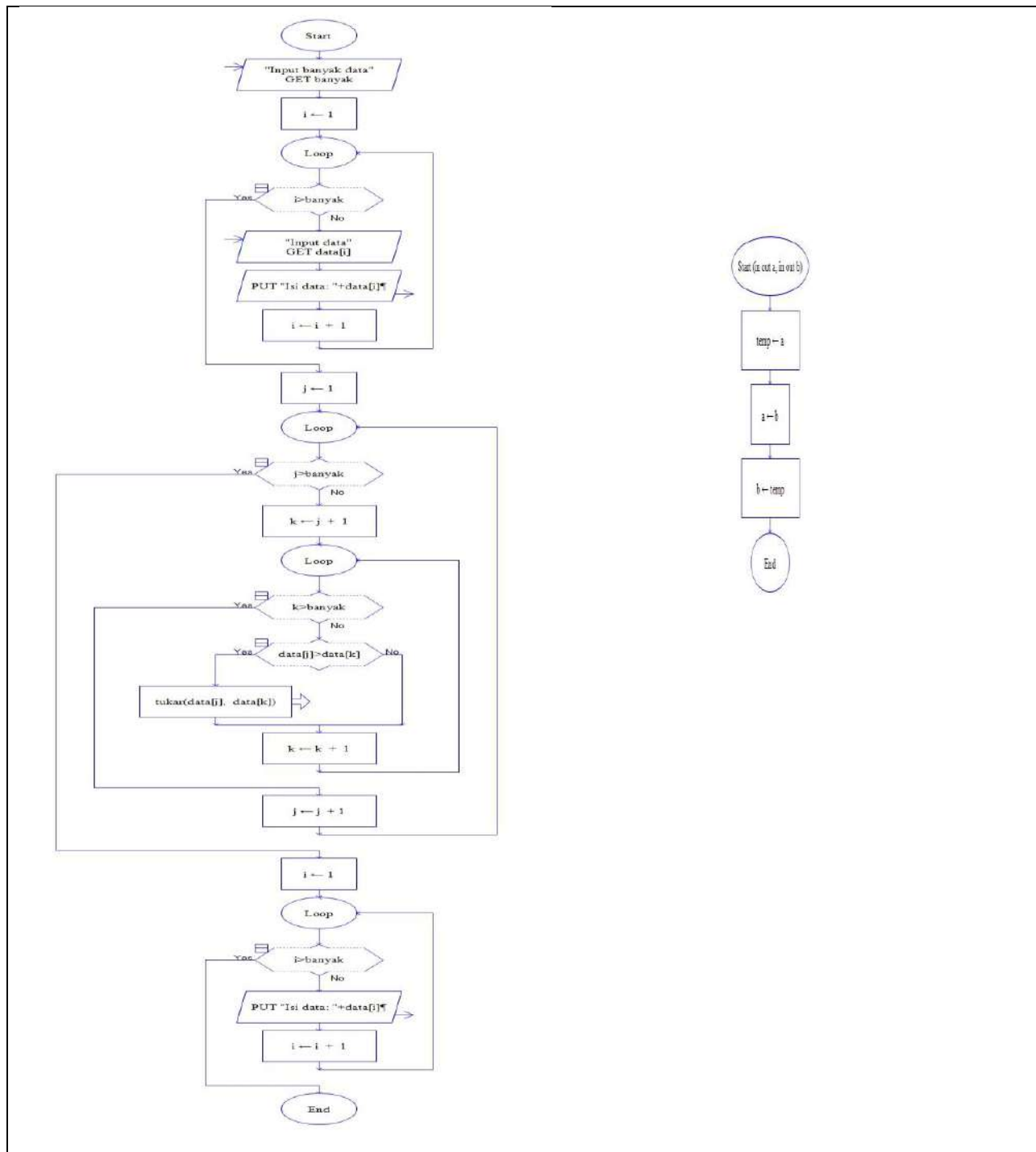
Bahasa C++

```
#include <iostream.h>
void baca_data(int A[], int n) {
    int i;
    for (i = 0; i < n; i++)
    {   cout << "Data ke-" << i+1 << " : ";
        cin >> A[i];
    }
}
void cetak_data(const int A[], int n) {
    int i;
    for (i = 0; i < n; i++)
        cout << A[i] << " ";
    cout << "\n";
}
void tukar (int *a, int *b)
{ int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

void bubble_sort (int x[], int n)
{ int i, j;
  for (i = 0; i < n-1; i++)
    for (j = 1; j < n-1; j++)
      if (x[i] > x[j]) tukar(&x[i], &x[j]);
}

void main() {
    int data[10], n;
    cout << "Banyak data : ";
    cin >> n;
    baca_data(data,n);
    cetak_data(data,n);
    bubble_sort(data,n);
    cetak_data(data,n);
}
```

Flowchart 7.3



8.6 Insertion Sort

Untuk menjelaskan algoritma pengurutan dengan metode penyisipan, bayangkan setumpuk kartu ada di meja, dengan nomor :

[8, 4, 7, 3, 1, 2, 6, 5]

Pertama, kartu 8, diperoleh di tangan kiri :

[8]

Kedua, diambil kartu nomor 4, karena lebih kecil dari 8, ditaruh di sebelah kiri, sehingga diperoleh :

[4, 8]

Ketiga, diambil kartu nomor 7, dibandingkan dengan kartu pertama, 7 lebih besar dari 4 dan lebih kecil dari 8, sehingga 7 diletakkan di antara 4 dan 8 :

[4, 7, 8]

Mulai langkah ketiga ini, kita dapat menarik kesimpulan, bahwa “tempat yang tepat” berarti “kartu yang saat ini diambil akan diletakkan berada di antara yang kecil dan yang besar dari kartu yang diambil tersebut”.

Urutan selengkapnya adalah sebagai berikut :

Data awal :	[8, 4, 7, 3, 1, 2, 6, 5]
fase 1, 4 masuk	[4 , 8, 7, 3, 1, 2, 6, 5]
fase 2, 7 masuk	[4, 7 , 8, 3, 1, 2, 6, 5]
fase 3, 3 masuk	[3 , 4, 7, 8, 1, 2, 6, 5]
fase 4, 1 masuk	[1 , 3, 4, 7, 8, 2, 6, 5]
fase 5, 2 masuk	[1, 2 , 3, 4, 7, 8, 6, 5]
fase 6, 6 masuk	[1, 2, 3, 4, 6 , 7, 8, 5]
fase 7, 5 masuk	[1, 2, 3, 4, 5 , 6, 7, 8]
fase 8	[1, 2, 3, 4, 5, 6, 7, 8]

Algoritma 7.4.

procedure insertion_sort(input/output data:larik; input n:integer)
Deklarasi k, j, temp : integer
Deskripsi for j \leftarrow 1 to n-1 do temp \leftarrow data [j]; i \leftarrow j-1; while (temp <= data [i]) and (i >= 1) do data [i+1] \leftarrow data [i]; i \leftarrow i-1; endwhile if (temp >= data [i]) then data [i+1] \leftarrow temp else data [i+1] \leftarrow data [i]; data [i] \leftarrow temp;

```
endif  
endfor
```

Translasi 7.4.

Bahasa C++

```
#include <iostream.h>  
#include <conio.h>  
class Sorting {  
    friend istream& operator>>(istream& in, Sorting& a);  
    friend ostream& operator<<(ostream& out, Sorting& a);  
public:  
    void baca_data();  
    void cetak_data();  
    void bubble_sort();  
    void insertion_sort();  
    void selection_sort();  
private:  
    void minimum(int , int, int&);  
    void tukar (int&, int&);  
    int data[10], n;  
};  
istream& operator>>(istream& in, Sorting& a) {  
    cout << "Banyak data : ";  
    in >> a.n;  
    for (int i = 0; i < a.n; i++) {  
        cout << "Data ke-" << i+1 << " : ";  
        in >> a.data[i];  
    }  
    return in;  
}  
ostream& operator<<(ostream& out, Sorting& a) {  
    for (int i = 0; i < a.n; i++)  
        out << a.data[i] << " ";  
    out << "\n";  
    return out;  
}  
void Sorting::tukar (int &a, int &b)  
{ int temp;  
    temp = a;  
    a = b;  
    b = temp;  
}  
void Sorting::insertion_sort()  
{
```

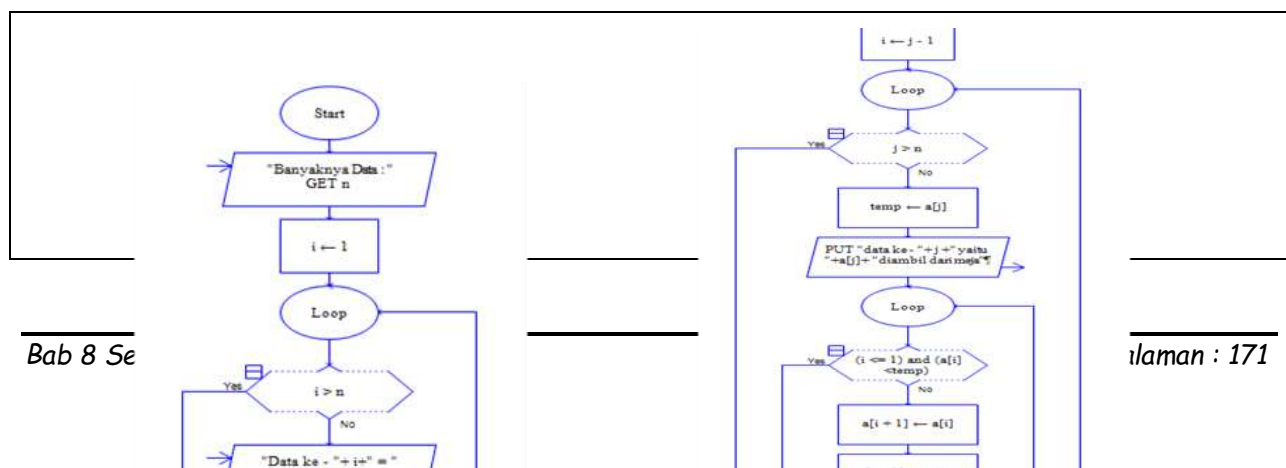
```

int i, j, temp;
cout << "Data pertama sudah ada ditangan kiri, yaitu : " << data[0] << '\n';
for(j = 1; j < n; j++)
{
    temp = data[j];
    cout << "\nData ke-" << j+1 << " yaitu " << data[j] << " diambil dari meja\n";
    cout << "Dilakukan penggeseran elemen :\n";
    for(i = j - 1; (i >= 0) && (data[i] > temp); i--)
    {
        cout << "data pd posisi ke-[" << i+1 << "] digeser ke posisi [" << i+2 << "]\n";
        data[i+1] = data[i];
    }
    cout << "Data baru masuk pada posisi ke-[" << i+2 << "]\n";
    data[i+1] = temp; //Insert key into proper position
    cout << "Data saat ini adalah : ";
    for (int k=0; k<=j; k++) cout << data[k] << " ";
    getch();
}
}

void main() {
    Sorting angka;
    cin >> angka;
    angka.insertion_sort();
    cout << "\nHasil akhir adalah : \n";
    cout << angka;
}

```

Flowchart 7.4



8.7 Selection Sort

Algoritma pengurutan dengan metode seleksi dapat diilustrasikan demikian. Misalkan diberikan data awal :

[8, 4, 7, 3, 1, 2, 6, 5]

Data pertama adalah 8. Akan dicari (atau tepatnya “dipilih/diseleksi”) **data terkecil** dari data kedua sampai terakhir yang terkecil untuk menempati posisi pertama ($i=1$) ini. Data terkecil ditemukan pada posisi ke-5 ($t=5$). Maka data pertama ditukar dengan data ke-5, menjadi :

[1, 4, 7, 3, 8, 2, 6, 5]

Langkah ini diulang untuk data kedua ($i=2$). Ditemukan data terkecil pada posisi ke-6 ($t=6$). Data ke-2 ditukar dengan data ke-6, menjadi :

[1, 2, 7, 3, 8, 4, 6, 5]

Fase selengkapnya dapat dilihat pada data berikut ini.

Data awal :	[8, 4, 7, 3, 1, 2, 6, 5]	1 terkecil, $8 \leftrightarrow 1$
fase 1	[1, 4, 7, 3, 8, 2, 6, 5]	2 terkecil, $4 \leftrightarrow 2$

fase 2	[1, 2 , 7, 3, 8, 4, 6, 5]	3 terkecil, $7 \leftrightarrow 3$
fase 3	[1, 2, 3 , 7, 8, 4, 6, 5]	4 terkecil, $7 \leftrightarrow 4$
fase 4	[1, 2, 3, 4 , 8, 7, 6, 5]	5 terkecil, $8 \leftrightarrow 5$
fase 5	[1, 2, 3, 4, 5 , 7, 6, 8]	6 terkecil, $7 \leftrightarrow 6$
fase 6	[1, 2, 3, 4, 5, 6 , 7, 8]	7 terkecil, tetap
fase 7	[1, 2, 3, 4, 5, 6, 7 , 8]	
fase 8	[1, 2, 3, 4, 5, 6, 7, 8]	

Algoritma 7.5.a.

Procedure minimum(input A : larik; dari, n : integer; output tempat : integer) { mencari tempat di mana elemen terkecil ditemukan}
Deklarasi i, min : integer
Deskripsi min \leftarrow A[dari]; tempat \leftarrow dari; for i \leftarrow dari+1 to n do if A[i] < min then min \leftarrow A[i]; tempat \leftarrow i; endif endfor

Algoritma 7.5.b.

Procedure selection_sort(output A : larik; input n : integer)
Deklarasi i, t : integer
Deskripsi for i \leftarrow 1 to n do minimum(A, i, n, t); tukar(A[i], A[t]); {tukar tempat elemen saat ini j dengan elemen terkecil yang ditemukan t} endfor

Translasi 7.5.

Bahasa C++
#include <iostream.h> void baca_data(int A[], int n) { int i; for (i = 0; i < n; i++)

```

    {   cout << "Data ke- %d : ",i+1);
        cin >> A[i]);
    }
}

void cetak_data(const int A[], int n)
{   int i;
    for (i = 0; i < n; i++)
        cout << "%d ",A[i]);
    cout << "\n";
}

void tukar (int *a, int *b)
{ int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

void minimum(int A[], int dari, int n, int * tempat)
{ int i, min;
  min = A[dari];
  *tempat = dari;
  for (i = dari+1; i<n; i++)
      if (A[i] < min)
      {   min = A[i];
          *tempat = i;
      }
}

void selection_sort(int A[], int n)
{ int i, t;
  for (i = 0; i<n; i++)
  {   minimum(A, i, n, &t);
      tukar(&A[i], &A[t]);
  }
}

main() {
  int data[10], n;
  cout << "Banyak data : ";
  cin >> n;
  baca_data(data,n);
  cetak_data(data,n);
  selection_sort(data,n);
  cetak_data(data,n);
  return 0;
}

```

}

Flowchart 7.5

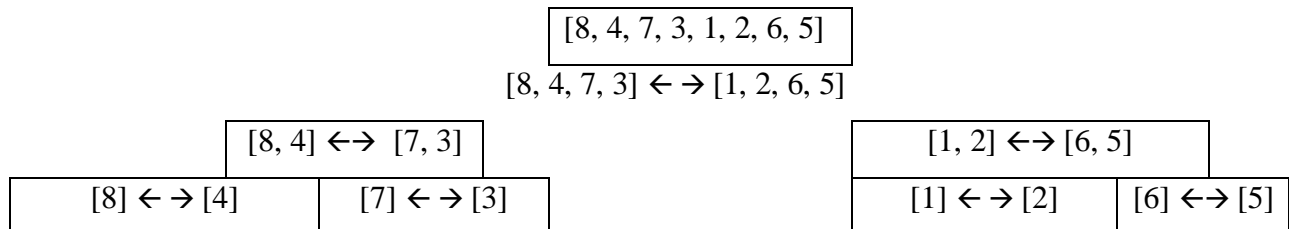
8.8 Merge Sort

Pengurutan dengan metode merge sort ini dikerjakan dengan cara rekursif. Langkah dari algoritma ini adalah sebagai berikut :

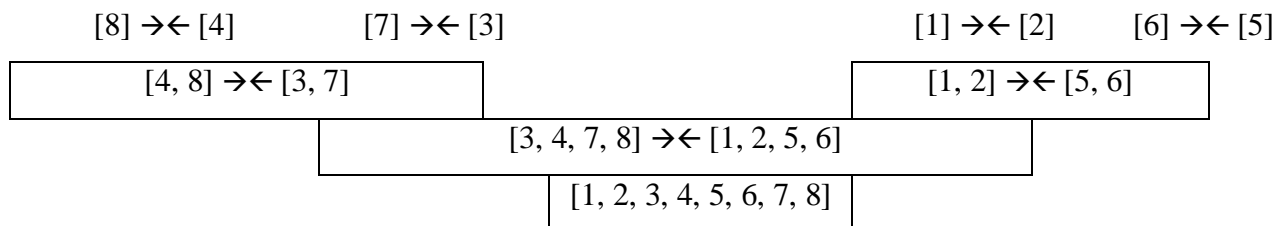
1. barisan data dibagi menjadi 2 subbarisan
2. sort secara rekursif
3. gabung hasil langkah 2 dari 2 subbarisan yang terurut menjadi barisan yang terurut.

Langkah-langkah pengurutan dengan metode penggabungan akan diperlihatkan berikut ini :

Fase pembagian



Fase penggabungan



Translasi 7.6.

Bahasa C++
<pre>#include <iostream.h> typedef int larik[10]; void masuk_data(int A[], int n) { int i; for (i = 0; i < n; i++) { cout << "Data ke-%d : ",i+1); cin >> A[i]; } } void cetak_data(const int A[], int n) { int i; for (i = 0; i < n; i++)</pre>

```

        cout << "%d ",A[i]);
    cout << "\n");
}

void merge(larik a, int kiri, int tengah, int kanan)
{
    int bagianKiri, posTemp, banyakElemen, i;
    larik temp;

    bagianKiri = tengah - 1;
    posTemp = kiri;
    banyakElemen = kanan - kiri + 1;
    while ((kiri <= bagianKiri) &&
           (tengah <= kanan))
        if ((a[kiri] <= a[tengah]))
        {
            temp[posTemp] = a[kiri];
            posTemp = posTemp + 1;
            kiri = kiri + 1;
        }
        else
        {
            temp[posTemp] = a[tengah];
            posTemp = posTemp + 1;
            tengah = tengah + 1;
        }

    /* kopi bagian kiri */
    while ((kiri <= bagianKiri)) {
        temp[posTemp] = a[kiri];
        posTemp = posTemp + 1;
        kiri = kiri + 1;
    }
    /* kopi bagian kanan */
    while ((tengah <= kanan)) {
        temp[posTemp] = a[tengah];
        posTemp = posTemp + 1;
        tengah = tengah + 1;
    }
    /* kopi kembali ke array asal */
    for (i = 1; i <= banyakElemen; i++)
    {
        a[kanan] = temp[kanan];
        kanan = kanan - 1;
    }
}

void merge_sort(larik A, int kiri, int kanan)
{
    int tengah;

```

```

    if ((kiri < kanan))
    {
        tengah = (kiri + kanan) / 2;
        merge_sort(A, kiri, tengah);
        merge_sort(A, tengah + 1, kanan);
        merge(A, kiri, tengah + 1, kanan);
    }
}

main() {
    int n;
    larik data;
    cout << "Berapa data array : ";
    cin >> n;
    masuk_data(data, n);
    cetak_data(data, n);
    merge_sort(data, 0, n-1);
    cetak_data(data, n);
    return 0;
}

```

8.9 Quick Sort

Pada pengurutan dengan metode quick sort, langkah-langkahnya adalah sebagai berikut :

1. dipilih sebuah elemen, yaitu elemen pivot (**p**). Elemen ini akan digunakan sebagai pembanding elemen-elemen yang lain untuk disusun dengan komposisi :

< p	p	> p
------------	----------	------------

yaitu semua elemen di sebelah kiri p, elemen tersebut lebih kecil dari p, sedangkan semua elemen yang berada di sebelah kanan p, semua lebih besar dari p. Elemen pivot bisa dipilih sembarang asal konsisten. Untuk algoritma yang akan dijelaskan berikut ini, elemen pivot dipilih elemen yang berada di tengah.

2. lakukan hal yang sama pada elemen sebelah kiri dan elemen sebelah kanan dari p.

No.	Isi array	pivot	i	j	data[i]	data[j]	Setelah ditukar
1.	[8, 4, 7, 3, 1, 2, 6, 5]	3	1	6	8	2	[2, 4, 7, 3, 1, 8, 6, 5]
2.	[2, 4, 7, 3, 1, 8, 6, 5]		2	5	4	1	[1, 4, 7, 3, 4, 8, 6, 5]
3.	[2, 1, 7, 3, 4, 8, 6, 5]		3	4	7	3	[2, 1, 3, 7, 4, 8, 6, 5]
4.	[2, 1, 3]	1	1	2	2	1	[1, 2, 3]
5.	[1, 2, 3]	2	2	2	2	2	[2]
6.	[7, 4, 8, 6, 5]	8	6	8	8	5	[7, 4, 5, 6, 8]
7.	[7, 4, 5, 6, 8]	4	4	5	7	4	[4, 7, 5, 6]

8.	[4, 7, 5, 6]	5	5	6	7	5	[5, 7, 6]
9.	[5, 7, 6]	7	6	7	7	6	[6, 7]
10.	[6, 7]						
11.	[1, 2, 3, 4, 5, 6, 7, 8]						

Pada langkah 3, kolom terakhir menunjukkan posisi data seperti yang dikehendaki, yaitu semua elemen di sebelah kiri elemen pivot semua lebih kecil dari elemen pivot, dan semua elemen di sebelah kanan pivot, semua lebih besar dari elemen pivot.

Algoritma 7.7.

procedure quick_sort(output data : larik; input L, R : integer)
Deklarasi i, j, p : integer
Deskripsi p ← data[(L+R) div 2] i ← L j ← R { mulai membuat partisi } while (i ≤ j) do while (data[i] < p) do i ← i+1 endwhile while (data[j] > p) do j ← j-1 endwhile if (i < j) then tukar(data[i], data[j]) i ← i+1 j ← j-1 endif endwhile if (L < j) then quick_sort(data,L,j) endif if (i < R) then quick_sort(data,i,R) endif

Translasi 7.7.

Bahasa C++
<pre>#include <iostream.h> void masuk_data(int A[], int n) { int i; for (i = 0; i < n; i++) { cout << "Data ke-<i>%d</i> : ",i+1); cin >> A[i]; } } void cetak_data(const int A[], int n) {</pre>


```

    int i;
    for (i = 0; i < n; i++)
        cout << "%d ",A[i];
    cout << "\n";
}
void tukar (int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
void quick_sort(int data[], int L, int R) {
    int i, j, p;
    p = data[(L+R) / 2];
    i = L;
    j = R;
    while (i<=j) {
        while (data[i] < p) i++;
        while (data[j] > p) j--;
        if (i<=j)
        {
            tukar(&data[i], &data[j]);
            i++;
            j--;
        }
    }
    if (L < j) quick_sort(data,L,j);
    if (i < R) quick_sort(data,i,R);
}

void main() {
    int data[10], n;
    cout << "Banyak data : ";
    cin >> n;
    masuk_data(data,n);
    cetak_data(data,n);
    quick_sort(data,0,n-1);
    cetak_data(data,n);
}

```

Latihan

Pada tahun 1959, D. L. Shell menemukan cara pengurutan dengan mempertimbangkan jarak data yang akan diurutkan. Metode ini kemudian dikenal dengan Shell Sort. Misalkan data yang akan diurutkan sebanyak n . Langkah-langkah adalah sebagai berikut :

1. diambil jarak = $n \div 2$, misalkan $i = \text{jarak} + j$, untuk $j=1$ sampai $n \div 2$, bila $\text{data}[j] > \text{data}[i]$, tukar letak kedua data.
2. lakukan langkah 1 sampai jarak = 1.

Sebagai ilustrasi, algoritma Shell Sort dapat digambarkan sebagai berikut :

Isi array	jarak	i (=jarak + j)	j	data[i]	data[j]	Setelah ditukar
[8, 4, 7, 3, 1, 2, 6, 5]	4	5	1	1	8	[1, 4, 7, 3, 8, 2, 6, 5]
		6	2	2	4	[1, 2, 7, 3, 8, 4, 6, 5]
		7	3	6	7	[1, 2, 6, 3, 8, 4, 7, 5]
[1, 2, 6, 3, 8, 4, 7, 5]	2	7	5	7	8	[1, 2, 6, 3, 7, 4, 8, 5]
[1, 2, 6, 3, 7, 4, 8, 5]	1	4	3	3	6	[1, 2, 3, 6, 7, 4, 8, 5]
		6	5	4	7	[1, 2, 3, 6, 4, 7, 8, 5]
		8	7	5	8	[1, 2, 3, 6, 4, 7, 5, 8]
		5	4	4	6	[1, 2, 3, 4, 6, 7, 5, 8]
		7	6	5	7	[1, 2, 3, 4, 6, 5, 7, 8]
		6	5	5	6	[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8]						

Buatlah algoritma dan program pengurutan dengan metode Shell Sort.

Kasus

Buatlah aplikasi kamus Inggris – Indonesia dengan minimal setiap huruf terdapat 5 kata yang diterjemahkan. Rancanglah melalui tahapan berikut :

1. Tahap 1 membuat class Kamus

```
class Kamus {  
    friend ostream& operator<<(ostream&, Kamus &);  
    friend istream& operator>>(istream&, Kamus &);  
public :  
    // berisi method sorting dan binary searching  
  
  
private :  
    // berisi sekurang-kurangnya dua data  
  
};
```

2. Tahap 2 : membuat implementasi class

3. Tahap 3 : running program

- a. memasukkan data dari keyboard
- b. menyimpan data ke file
- c. melakukan pencarian data : kasus ada dalam kamus (atau tidak ada)

Latihan :

1. Perhatikan bahwa Anda dapat melewati banyak nomor dalam daftar dan masih berada dalam urutan menaik yaitu sebagai berikut :

3 4 6 17 21 24 32 43

Angka-angka ini meningkat saat Anda bergerak melalui daftar dari kiri ke kanan. Bangunlah sebuah array yang berisi angka-angka tersebut ? Kemudian lakukan pencarian biner (Binary Search) untuk memeriksa apakah angka yang kita cari ada dalam daftar array tersebut ?

2. Jika terdapat sebuah *array* yang elemennya berindeks 1 sampai dengan 15. Masing-masing elemen berturut-turut berisi nilai sebagai berikut:

1, 2, 8, 25, 30, 49, 50, 55, 60, 61, 68, 70, 72, 84, 90.

- a. Jelaskan langkah-langkah pencarian nilai 49 dalam *array* tersebut dengan metode pencarian biner, sehingga menghasilkan indeks elemen *array* tempat ditemukannya nilai tersebut.
 - b. Jelaskan langkah-langkah pencarian nilai 71 dalam *array* tersebut dengan metode pencarian biner, sehingga menghasilkan kesimpulan bahwa nilai tersebut tidak berhasil ditemukan.
3. Urutkan deret angka berikut dengan bubble sort :

7 4 5 8 10

Tuliskan hasil tiap langkah (step).

4. Periksalah daftar 6 angka di bawah ini :

14 32 5 12 61 7

Ketika Anda melihat daftar tersebut, Anda segera dapat melihat bahwa 5 adalah angka terkecil didaftar. Tugas ini lebih sulit untuk komputer. Jadi untuk itu harus dibuat program untuk menemukan nilai minimum tersebut. Buatlah program selection sort dan lakukan sorting secara manual (step by step) !

5. Urutkan deret angka berikut dengan selection sort dan tuliskan hasil tiap langkah (step) :

21 16 25 8 19 4 1

6. Diketahui deret angka sebagai berikut :

5 2 4 6 1 3

Dari deret angka tersebut, lakukan pengurutan dari yang paling kecil sampai paling besar menggunakan insertion sort !

7. Mari kita lihat daftar nomor dari sebuah array untuk melihat bagaimana cara merge sort bekerja :

32 12 5 18 31 4 25 7

[0] [1] [2] [3] [4] [5] [6] [7]

Lakukan sorting dari data dalam array di atas menggunakan merge sort sehingga nomor paling kecil berada paling depan samapai yang paling besar berada paling belakang !

8. Diketahui deretan data sebagai berikut :

8 1 4 9 7 3 5 2 7

Urutkan data tsb. memakai **Merge sort**, agar elemen terkecil berada paling depan (urutan pertama), semakin ke belakang semakin besar !

9. Ada beberapa kumpulan data sebagai berikut :

2 8 3 5 6 4 11 1 9

Urutkan kumpulan data di atas menggunakan quick sort serta gambarkan step by step dari sorting tersebut !

10. Urutkan data yaitu [2 8 7 1 3 5 6 4] dengan menggunakan **Quick Sort**, agar elemen terkecil berada paling depan (urutan pertama), semakin ke belakang semakin besar !

BAB 9

ARRAY DUA DIMENSI, KASUS MATRIKS

9.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu menjelaskan berbagai algoritma menggunakan array (variabel berindeks) baik satu ataupun dua dimensi.

9.2 Pendahuluan

Sebagai ilustrasi, array dua dimensi dapat digambarkan sebagai sebuah matriks :

Misalkan matriks A berordo 2 x 3 :

$$A = \begin{bmatrix} 1 & -2 & 3 \\ -4 & 0 & -1 \end{bmatrix}$$

Dalam memori, setiap elemen matriks A dapat dipetakan seperti :

	kolom j=1	kolom j=2	kolom j=3
baris i=1	A[1,1] = 1	A[1,2] = -2	A[1,3] = 3
baris i=2	A[2,1] = -4	A[2,2] = 0	A[2,3] = -1

Dengan demikian, suatu matriks dapat dibuat implementasinya dengan menggunakan array berdimensi dua.

9.3. Konstruksi Tipe Data Matriks

Matriks dikonstruksikan dengan menggunakan array berdimensi dua. Indeks array pertama dapat dipandang sebagai baris, indeks array kedua dapat dipandang sebagai kolom dari matriks. Dengan demikian tipe baru untuk matriks adalah sebagai berikut :

Bahasa C++
#define baris 2

```
#define kolom 2
int matriks[baris][kolom];
```

dengan banyak baris dan kolom dapat ditentukan lewat konstanta yang didefinisikan tersendiri.

9.4 Membaca Elemen Matriks

Oleh karena menggunakan array dua dimensi, maka diperlukan dua buah perulangan, yaitu dengan dua buah indeks yang berbeda.

Algoritmik	Bahasa C++
procedure baca_matriks (output mat : matriks; input baris,kolom:integer) Deklarasi i,j : integer Deskripsi for i ← 1 to baris do for j ← 1 to kolom readln(mat[i,j]) endfor endfor	void baca_matriks (int mat[10][10], int baris, int kolom) { int i,j; for (i=0; i<baris; i++) for (j=0; j<kolom; j++) cin >> mat[i][j]; }

9.5. Mencetak Elemen Matriks

Algoritmik	Bahasa C++
procedure cetak_matriks (input mat : matriks; input baris,kolom:integer) Deklarasi i,j : integer Deskripsi for i:=1 to baris do for j:=1 to kolom readln(mat[i,j]) endfor endfor	void cetak_matriks (const int A[10][10], int baris, int kolom) { int i,j; for (i=0; i<baris; i++) { for (j=0; j<kolom; j++) cout << A[i][j]; cout << endl; } }

Kasus 8.1:

Buatlah algoritma dan program untuk menjumlahkan dua buah matriks.

Analisis :

Dua buah matriks dapat dijumlahkan jika dua matriks tersebut berukuran sama, misalkan m baris dan n kolom. Untuk mengetahui bagaimana dua buah matriks dapat dijumlahkan, dapat diilustrasikan seperti berikut ini. Misalkan matriks A dan B berukuran 2 x 2. Matriks C diperoleh sebagai hasil penjumlahan matriks A dan matriks B.

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Ternyata setiap elemen c_{ij} , $i=1,2$ dan $j=1,2$ diperoleh dari penjumlahan $a_{ij} + b_{ij}$, yaitu hasil penjumlahan dari elemen yang bersesuaian.

Algoritma 8.1.

procedure matriks_jumlah(input matriks1,matriks2 : matriks; baris, kolom : integer; output jumlah : matriks)
Deklarasi i,j : integer
Deskripsi for i \leftarrow 1 to baris do for j \leftarrow 1 to kolom do jumlah[i,j]:=matriks1[i,j] + matriks2[i,j] endfor endfor

Translasi 8.1.

Bahasa C++
<pre>#include <iostream.h> void baca_matriks (int mat[10][10], int baris, int kolom) { int i,j; for (i=0; i<baris; i++) for (j=0; j<kolom; j++) { cout << "Data [" << i+1 << "," << j+1 << "] : "; cin >> mat[i][j]; } } void matriks_jumlah(const int matriks1[10][10], const int matriks2[10][10], int baris, int kolom, int jumlah[10][10]) { int i,j; for (i=0; i<baris; i++) for (j=0; j<kolom; j++) jumlah[i][j]=matriks1[i][j]+matriks2[i][j]; }</pre>

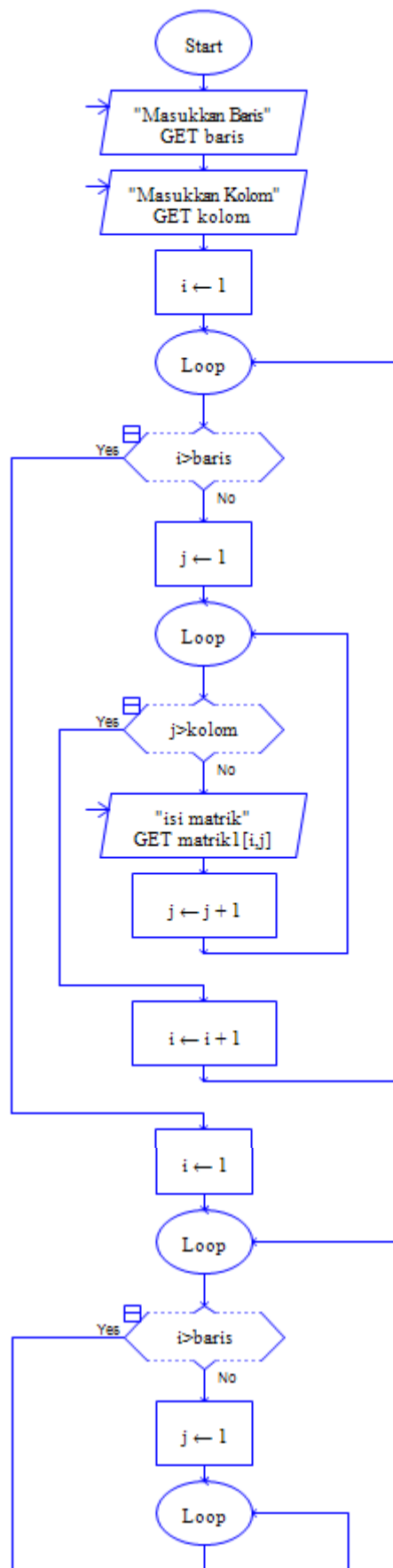

```

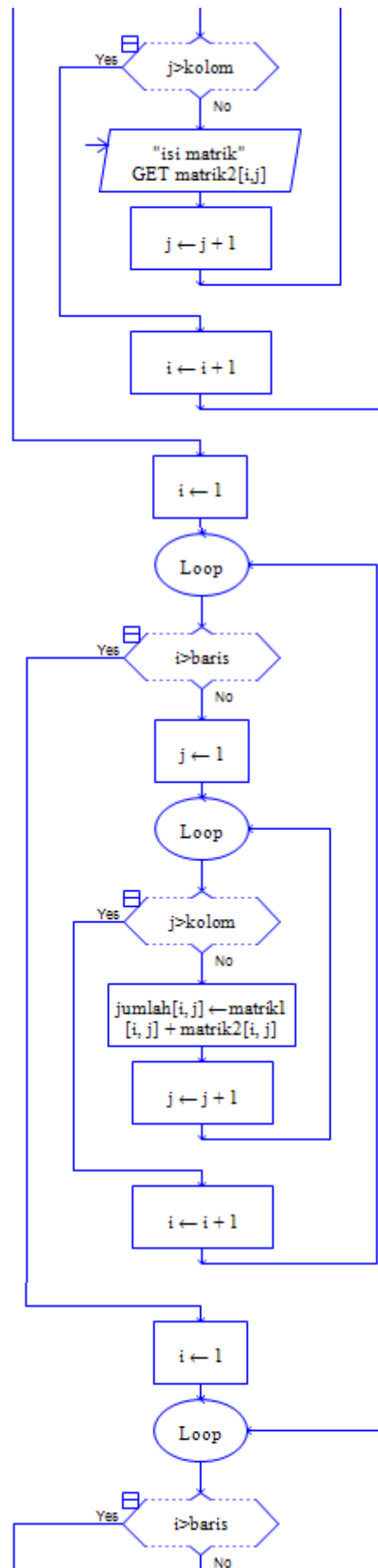
}

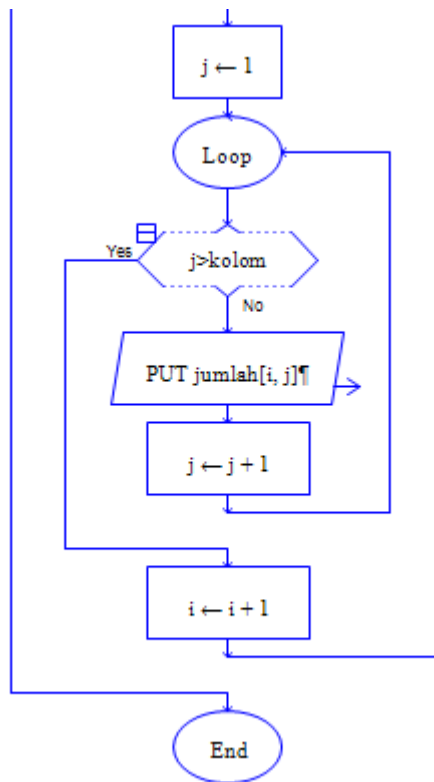
void cetak_matriks (const int A[10][10], int baris, int kolom)
{   int i,j;
    for (i=0; i<baris; i++)
    {   for (j=0; j<kolom; j++)
        cout << A[i][j];
        cout << endl;
    }
}

void main() {
    int m, n;
    int matriks1[10][10], matriks2[10][10];
    int jumlah[10][10];
    cout << "Banyak baris : ";
    cin >> m;
    cout << "Banyak kolom : ";
    cin >> n;
    cout << "Data matriks ke-1 \n";
    baca_matriks(matriks1,m,n);
    cetak_matriks(matriks1,m,n);
    cout << "Data matriks ke-2 \n";
    baca_matriks(matriks2,m,n);
    cetak_matriks(matriks2,m,n);
    matriks_jumlah(matriks1,matriks2,m,n,jumlah);
    cout << "Hasil Penjumlahan : \n";
    cetak_matriks(jumlah,m,n);
}

```







Kasus 8.2.

Buatlah algoritma dan program untuk mengalikan dua buah matriks.

Analisis :

Syarat dua matriks dapat dikalikan adalah ukuran kolom matriks pertama haruslah sama dengan ukuran baris matriks kedua. Cara kerja perkalian dua matriks dapat diambarkan sebagai berikut. Misal matiks A berukuran 2 x 3 dan matriks B berukuran 3 x 1 dan akan menghasilkan matriks C berukuran 1 x 2. Misalkan pula $A_{m \times p}$ dan $B_{p \times n}$, maka $m = 2$, $p = 3$, dan $n = 1$.

$$\begin{bmatrix} c_{11} \\ c_{21} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} * \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \end{bmatrix}$$

Matriks C diperoleh dengan cara :

$$c_{11} = a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31}$$

dan

$$c_{21} = a_{21} * b_{11} + a_{22} * b_{21} + a_{23} * b_{31}$$

dari dua baris persamaan tersebut kita dapat menemukan suatu pola, yaitu untuk memperoleh elemen $c[i,j]$ maka suku perkalian dari elemen a dengan baris i dan b dengan kolom j. Sedangkan kolom elemen a dan baris elemen b ternyata mempunyai perulangan tersendiri, yaitu k, dengan k

= 1, 2, 3. Dapat dilihat bahwa k mengikuti perulangan p, sedangkan i mengikuti perulangan m dan j mengikuti perulangan n. Setiap persamaan memperlihatkan adanya kumulatif penjumlahan.

Algoritma 8.2.

procedure kali_matriks(input matriks1,matriks2 : matriks; baris, kolom, barkol : integer; output mat_kali : matriks)
Deklarasi i,j,k : integer
Deskripsi for i \leftarrow 1 to baris do for j \leftarrow 1 to kolom do mat_kali[i,j] \leftarrow 0; for k \leftarrow 1 to barkol do mat_kali[i,j] \leftarrow mat_kali[i,j] + matriks1[i,k] * matriks2[k,j] endfor endfor endfor

Translasi 8.2.

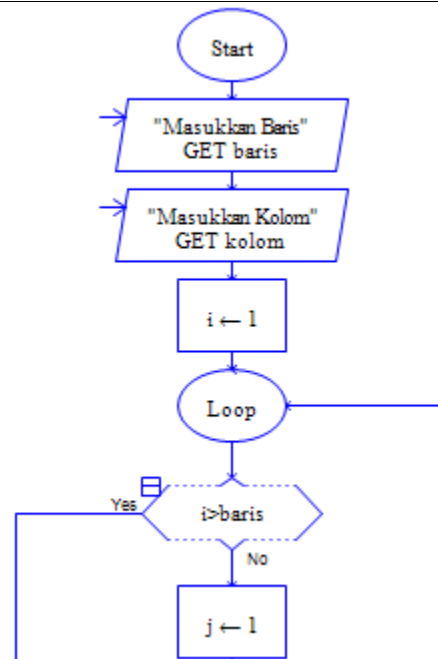
Bahasa C++
<pre>#include <iostream.h> void baca_matriks (int mat[10][10], int baris, int kolom) { int i,j; for (i=0; i<baris; i++) for (j=0; j<kolom; j++) { cout << "Data [" << i+1 << "," << j+1 << "] : "; cin >> mat[i][j]; } } void kali_matriks(const int matriks1[10][10], const int matriks2[10][10], int baris, int kolom, int barkol, int mat_kali[10][10]) { int i,j,k; for (i=0; i<baris; i++) for (j=0; j<kolom; j++) { mat_kali[i][j] = 0; for (k=0; k<barkol; k++) mat_kali[i][j] = mat_kali[i][j] + matriks1[i][k] * matriks2[k][j]; } }</pre>

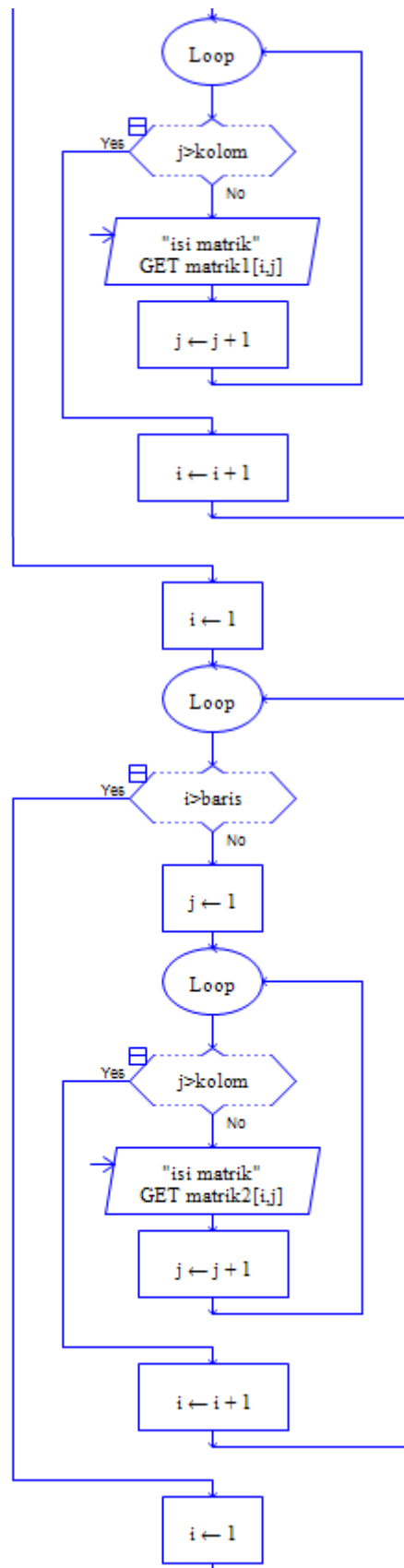
```
void cetak_matriks (const int A[10][10], int baris, int kolom)
```

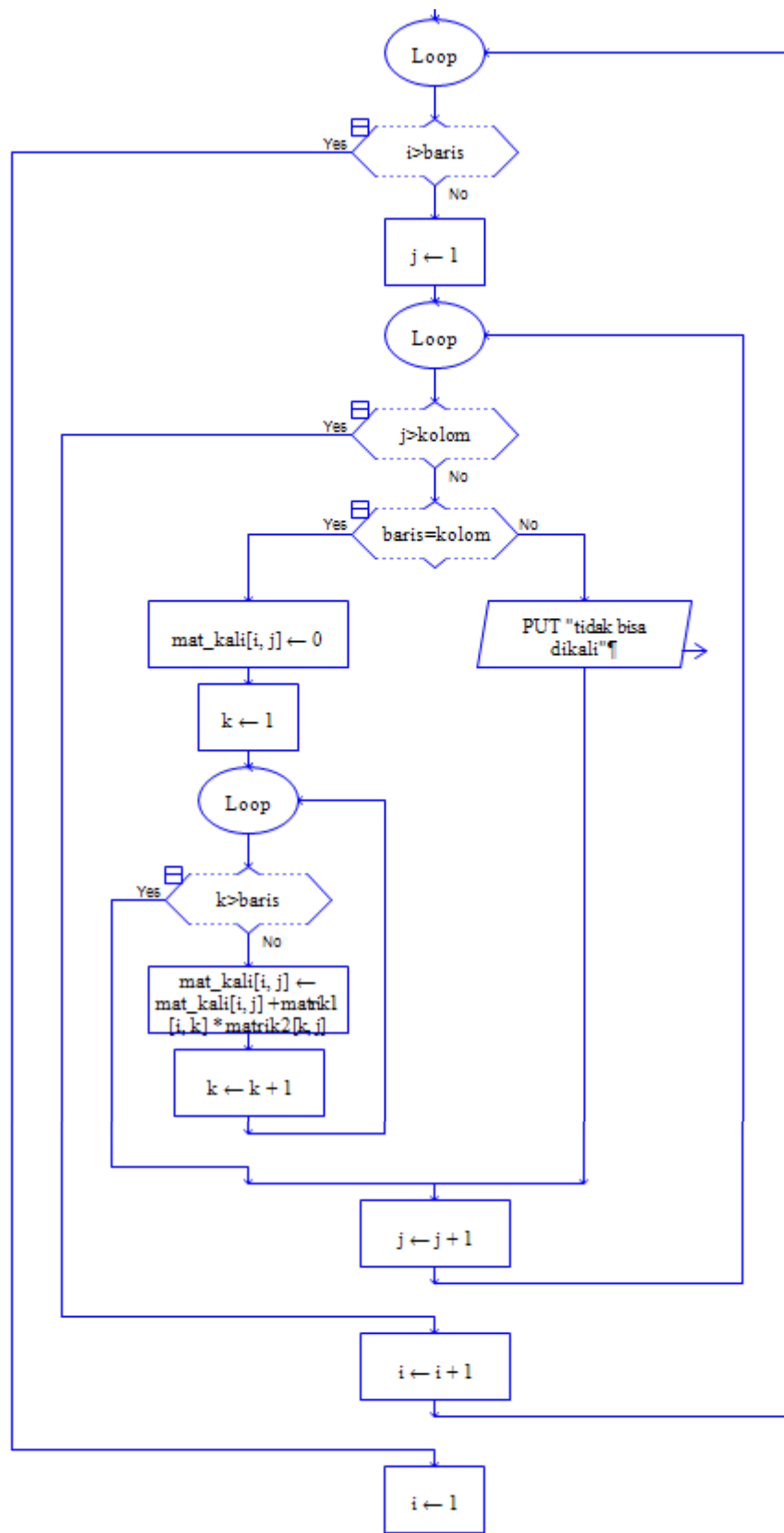
```
{  int i,j;
    for (i=0; i<baris; i++)
    {  for (j=0; j<kolom; j++)
        cout << A[i][j];
        cout << endl;
    }
}
```

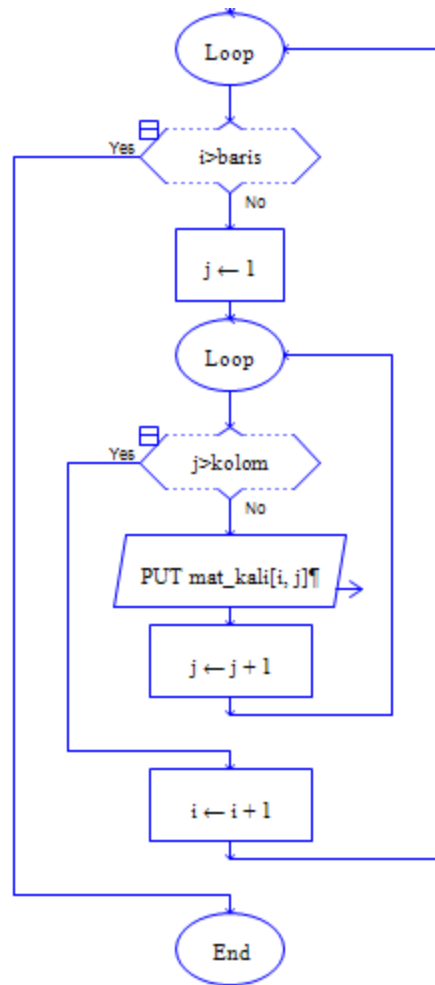
```
void main() {
```

```
    int m, n, p;
    int matriks1[10][10], matriks2[10][10];
    int hasil[10][10];
    printf("Baris matriks ke-1 : ");
    scanf("%d",&m);
    printf("Banyak kolom : "); scanf("%d",&p);
    printf("Kolom matriks ke-2 : ");
    scanf("%d",&n);
    printf("Data matriks ke-1 \n");
    baca_matriks(matriks1,m,p);
    cetak_matriks(matriks1,m,p);
    printf("Data matriks ke-2 \n");
    baca_matriks(matriks2,p,n);
    cetak_matriks(matriks2,p,n);
    kali_matriks(matriks1,matriks2,m,n,p,hasil);
    printf("Hasil Perkalian : \n");
    cetak_matriks(hasil,m,n);
}
```









Kasus 8.3.

Buatlah algoritma dan program untuk menyelesaikan sistem persamaan linier

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

...

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

Analisis :

Salah satu cara menyelesaikan sistem persamaan linier adalah dengan menggunakan metode eliminasi Gauss. Pertama, bentuk sistem persamaan linier di atas dikonversikan ke dalam bentuk :

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

Matriks A berukuran nxn, sedangkan vektor (kolom) x maupun b berukuran n. Dengan demikian, dapat dibuat type matriks dan vektor sebagai berikut :

type **matriks** = record <

baris, kolom : integer

elemen : array [1..maks,1..maks] of real >

vektor = record <

baris : integer

elemen : array [1..maks] of real >

Kemudian, pada matriks A dilakukan operasi elementer baris¹ sedemikian sehingga akan diperoleh matriks segitiga atas :

$$\begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ 0 & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a'_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}$$

dengan melakukan substitusi dari nilai paling bawah akan diperoleh penyelesaian dari sistem persamaan linier semula.

$$x_n = \frac{b'_n}{a'_{nn}}$$

$$x_{n-1} = \frac{b'_n - a'_{n-1n}x'_n}{a'_{n-1n-1}}$$

...

$$x_{n-1} = \frac{b'_n - a'_{1n}x'_n - \cdots - a'_{12}x'_2}{a'_{11}}$$

Misalkan sistem persamaan linier yang diberikan sebagai berikut :

$$x_1 + 2x_2 + 3x_3 = 1$$

$$x_1 + 3x_2 = 2$$

$$x_1 + 4x_2 + 3x_3 = 3$$

¹ lihat di buku *Aljabar Linier*, karangan H. Anton, penerbit Erlangga

dibentuk menjadi :

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 0 \\ 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

sehingga diperoleh matriks yang bersesuaian adalah :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 0 \\ 1 & 4 & 3 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

dengan operasi baris elementer, kita jumlahkan baris ke-2 dengan (-1) x baris pertama diperoleh :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -3 \\ 1 & 4 & 3 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

jumlahkan baris ke-3 dengan (-1) x baris pertama diperoleh :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -3 \\ 0 & 2 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

jumlahkan baris ke-3 dengan (-2) x baris ke-2 diperoleh :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 6 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

dengan substitusi diperoleh :

$$x_3 = 0,$$

$$x_2 = 1,$$

$$x_1 = 1 - 2x_2 = 1 - 2 = -1$$

sehingga penyelesaian sistem persamaan linier semula adalah :

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

Algoritma 8.3.

procedure gauss(input a : matriks; b : vektor; output x : vektor)
Deklarasi i, j, k : integer temp, s : real
Deskripsi for i ← 1 to a.baris - 1 do for k ← i+1 to a.baris do temp ← a.elemen[k,i] / a.elemen[i,i] for j ← i+1 to a.baris do a.elemen[k,j] ← a.elemen[k,j] - temp*a.elemen[i,j] endfor b.elemen[k] ← b.elemen[k] - temp*b.elemen[i] a.elemen[k,i] ← 0.0 endfor x.baris ← a.baris for i ← a.baris downto 1 do s ← b.elemen[i] for j ← i+1 to a.baris do s ← s - a.elemen[i,j] * x.elemen[j] x.elemen[i] ← s/a.elemen[i,i] endfor endfor

Translasi 8.3.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> #include "matriks.h" #include "vektor.h" #define maks 25 #define false 0 #define true 1 class SPL { friend ostream& operator<<(ostream&, SPL&); friend istream& operator>>(istream&, SPL&); public: SPL(); void gauss(); void cetak(); private: Matriks koef; Vektor konstanta; int banyak; Vektor X; };</pre>

```

SPL::SPL() {
    banyak = 3;
}

ostream& operator<<(ostream& out, SPL& A)
{ Vektor hasil;
  A.gauss();
  A.cetak();
  return out;
}

istream& operator>>(istream& in, SPL& A) {
    cout << "Jumlah persamaan: ";
    in >> A.banyak;
    A.koef.beri_nilaiBaris(A.banyak);
    A.koef.beri_nilaiKolom(A.banyak);
    cin >> A.koef;
    A.konstanta.beri_nilaiBanyak(A.banyak);
    cin >> A.konstanta;
    return in;
}

void SPL::gauss()
{ int i, j, k;
  float temp, s;
  error = false;
  for (i = 0; i < banyak - 1; i++)
      for (k = i + 1; k < banyak; k++) {
          if ((koef.A[i][i] == 0.0)) {
              error = true;
              return;
          }
          temp = koef.A[k][i] / koef.A[i][i];
          for (j = i + 1; j < banyak; j++)
              koef.A[k][j] = koef.A[k][j] - temp * koef.A[i][j];
          konstanta.elemen[k] = konstanta.elemen[k] - temp * konstanta.elemen[i];
          koef.A[k][i] = 0.0;
      }
    X.banyak = banyak;
    for (i = banyak-1; i >= 0; i--)
    { s = konstanta.elemen[i];
      for (j = i + 1; j < banyak; j++)
          s -= koef.A[i][j] * konstanta.elemen[j];
      if ((koef.A[i][i] == 0.0)) {

```

```

        error = true;
        return;
    }
    X.elemen[i] = s / koef.A[i][i];
}
}

void SPL::cetak() {
    int i;
    if ((error))
        printf("Persamaan tidak dapat diselesaikan.\n");
    else {
        cout << "\n";
        cout << "Penyelesaian SPL : \n";
        cout << X;
    }
}

void main() {
    SPL X;
    cout << X;
    getch();
}

```

Workshop

Perhatikan dan pelajari algoritma 8.3. Procedure Gauss. Kemudian lakukan langkah demi langkah untuk Sistem Persamaan Linier berikut :

$$3x_1 + 2x_2 = 18$$

$$-x_1 + 2x_2 = 2$$

Isilah nilai setiap variabel berikut :

i	k	Temp	elemen	baris	s

Latihan

1. Buatlah algoritma dan program untuk mengalikan matriks dengan vektor. Vektor adalah matriks berdimensi satu (bisa matriks baris maupun matriks kolom). Implementasikan matriks dengan array berdimensi dua, dan vektor dengan menggunakan matriks berdimensi satu.
 - a. Buat contoh matriks sembarang, misal berordo 2x2 :
 - b. Buat contoh vektor :
 - c. Kalikan matriks poin a dan vektor poin b elemen demi elemen. Jangan menuliskan hasil akhir saja, tapi tuliskan pula perkalian (dan penjumlahan) setiap elemen.

d. Identifikasilah dan kelompokkan setiap langkah dengan variabel perulangan !

e. Tulis lengkap algoritma yang anda peroleh :

2. Diberikan suatu matriks berordo n . Buatlah algoritma dan program untuk mencetak matriks identitas berordo n .
3. Buatlah algoritma dan program untuk menghasilkan transpose suatu matriks.
4. Buatlah algoritma dan program mengalikan matriks dengan suatu skalar (konstanta).
5. Buatlah algoritma dan program invers matriks berordo dua. Lalu kalikan matriks asal dengan matriks inversnya untuk mengecek apakah hasil kalinya merupakan matriks identitas atau bukan.
 - a. Buat contoh matriks sembarang, berordo 2×2 :

 - b. Hitung determinan matriks di atas

c. Tulis matriks adjoin dari poin a

d. Hitung invers matriks berordo 2×2 :

BAB 10

POINTER, PENGENALAN STRUKTUR DATA

10.1 Tujuan Instruksional

Tujuan instruksional terbagi menjadi 2 dalam SAP yaitu Tujuan Instruksional Umum (TIU) dan Tujuan Instruksional Khusus (TIK).

A. Tujuan Instruksional Umum

Menjelaskan kepada mahasiswa teknik informatika agar memahami tahapan penyelesaian masalah komputasi, logika pemrograman dan implementasinya dalam sebuah bahasa pemrograman.

B. Tujuan Instruksional Khusus

Mampu membedakan penggunaan array dan pointer dan menerapkannya dalam pemrograman.

10.1 Pendahuluan

Pointer merupakan salah satu jenis data terstruktur. Dengan menggunakan pointer, suatu variabel dapat diciptakan atau dihapus selama pengeksekusi program, variabel demikian dinamakan sebagai variabel dinamis. Variabel dinamis adalah suatu variabel yang akan dialokasikan pada saat diperlukan saja, yaitu saat program dieksekusi. Dengan menggunakan variabel dinamis ini, dimungkinkan untuk membuat struktur data dinamis seperti *link list*, *queue*, *stack* dan *tree*.

Pada struktur data dinamis, terdapat variabel yang disebut variabel pointer (secara singkat disebut pointer), yaitu variabel yang menunjuk alamat memory pada variabel dinamis. Secara umum dapat dikatakan bahwa variabel pointer merupakan suatu variabel yang menyimpan **alamat** dari suatu objek. Karena itu sebenarnya variabel pointer bukan berisi data, melainkan berisi alamat dari suatu data. Secara singkat pointer artinya menunjuk ke suatu lokasi data. Pointer harus dideklarasikan dan diberi nama serta dikaitkan dengan suatu tipe. Tipe ini disebut tipe dasar atau tipe domain dari pointer tersebut, di mana tipe ini menentukan tipe dari variabel dinamis yang akan ditunjuk oleh pointer yang bersangkutan. Untuk ilustrasi, pointer akan digambarkan sebagai anak panah dan lokasi memory sebagai segi empat.

Dalam variabel dinamis, nilai data yang ditunjuk oleh suatu pointer biasa disebut dengan simpul/node. Simpul biasanya berupa suatu record (dapat juga tipe data yang lain kecuali yang menyangkut file).

10.2 Definisi Variabel Pointer

Tipe pointer didefinisikan dengan sintaks sebagai berikut :

Bahasa C++
int *P;
*P = 5;

Di sini P merupakan variabel pointer yang menunjuk variabel bertipe integer. P[^] (atau *P) menyatakan objek yang ditunjuk oleh pointer P. Tipe dari sebuah pointer sesuai dengan tipe objek yang ditunjuknya.

10.3 Menciptakan Variabel Dinamis

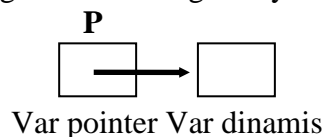
Pointer biasa dipakai untuk menunjuk variabel dinamis, tetapi pointer dapat juga dipakai untuk menunjuk variabel statis. Variabel dinamis tidak diberi nama seperti variabel statis. Oleh karena itu variabel dinamis tidak perlu dideklarasikan. Satu-satunya cara untuk mengakses variabel dinamis adalah dengan menggunakan pointer.

Variabel dinamis dibuat dengan menggunakan pernyataan berikut :

Bahasa C++
int *P;
P = new int;

P akan menunjuk ke variabel dinamis yang dibuat. Variabel dinamis yang baru dibuat itu tipenya sesuai dengan tipe dasar **P**. Contoh di atas menyebabkan suatu alamat memori dari variabel dinamis bertipe integer disimpan dalam variabel pointer P.

Jika digambarkan diagramnya adalah sebagai berikut:

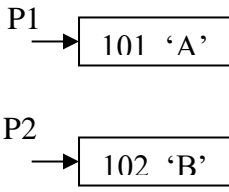
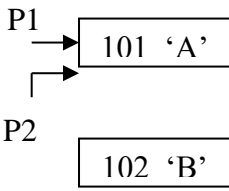
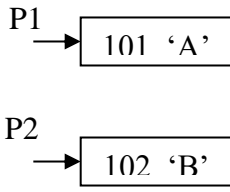
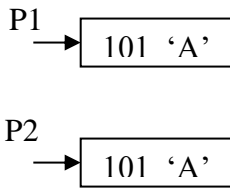


Berikut ini disajikan contoh program yang menggunakan pointer:

Nomor baris	Bahasa C++
1.	#include <iostream.h>
2.	class Mhs {
3.	public :
4.	int nim;
5.	char nilai;

6.	};
7.	typedef Mhs *Ptrmhs;
8.	
9.	void main() {
10.	Ptrmhs P1, P2, P3;
11.	P1 = new Mhs;
12.	P1->nim = 101;
13.	P1->nilai = 'A';
14.	cout << "P1 : " << P1->nim << " " << P1->nilai << endl;
15.	P2 = new Mhs;
16.	P2 = P1; /* salin alamatnya */
17.	cout << "P2 : " << P2->nim << " " << P2->nilai << endl;
18.	P2->nim = 102;
19.	P2->nilai = 'B';
20.	cout << "P1 : " << P1->nim << " " << P1->nilai << endl;
21.	cout << "P2 : " << P2->nim << " " << P2->nilai << endl;
22.	P3 = new Mhs;
23.	*P3 = *P1; /* salin isinya */
24.	cout << "P1 : " << P1->nim << " " << P1->nilai << endl;
25.	cout << "P3 : " << P3->nim << " " << P3->nilai << endl;
26.	P3->nim = 103;
27.	P3->nilai = 'C';
28.	cout << "P1 : " << P1->nim << " " << P1->nilai << endl;
29.	cout << "P3 : " << P3->nim << " " << P3->nilai << endl;
30.	}

Perbedaan antara $P1 = P2$ dengan $*P1 = *P2$, adalah:

$P2 = P1;$	$*P2 = *P1;$
<p>Semula :</p>  <p>Setelah :</p> 	<p>Semula :</p>  <p>Setelah :</p> 

Jika X dan Y adalah pointer, maka arti pernyataan $X = Y;$ adalah mengubah X sedemikian sehingga **X menunjuk ke objek yang sama dengan yang ditunjuk oleh Y**. X dan Y berisi nilai yang sama. Sedangkan arti dari pernyataan: $*X = *Y;$ adalah variabel yang ditunjuk oleh pointer X diberi nilai yang sama dengan nilai dari variabel yang ditunjuk pointer Y.

Variabel-variabel yang dapat dipakai selama pelaksanaan dari sebuah program dapat dibedakan atas:

1). Variabel Statis

- Dideklarasikan dan diberi nama seperti biasa program ditulis.
- Tempat (memory) untuk variabel-variabel ini tetap ada selama programnya sedang jalan.
- Tidak dapat dibuat atau dihapus sewaktu program sedang jalan.

2). Variabel Dinamis

- Dibuat dan dapat dihapus sewaktu program sedang jalan.
- Karena variabel dinamis ini tidak ada pada waktu program di-*compile*, maka variabel-variabel ini tidak dapat diberi nama seperti variabel-variabel statis.

10.4 Menghapus Pointer

Pointer yang telah dialokasikan (dibentuk) bisa didealokasikan (dihapus) kembali pada saat program dieksekusi. Setelah suatu pointer dihapus, maka lokasi yang semula ditempati oleh

simpul yang ditunjuk oleh pointer tersebut akan bebas, sehingga dapat ditempati oleh variabel lain.

Cara menghapus pointer adalah sebagai berikut :

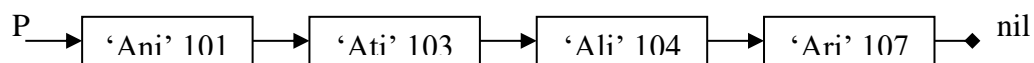
Bahasa C++
<code>delete(var_pointer);</code>

10.4 Linked List

Biasanya pointer dipakai untuk menunjuk variabel dinamis yang bertipe record, di mana record ini mempunyai komponen (field) yang bertipe pointer. Record seperti ini disebut simpul (node). Sebagai contoh:

Bahasa C++
<pre>typedef struct { char Nama[20]; char Nim[10]; PMhs Link; } Mhs; typedef Mhs *PMhs; PMhs P;</pre>

Variabel pointer Link berfungsi sebagai pengait atau penghubung dengan simpul berikutnya, sehingga membentuk suatu untaian yang dinamakan sebagai daftar berantai (*Linked List*). Ciri daftar berantai adalah terdapat pointer yang berfungsi sebagai pointer kepala (*Head*) dari daftar berantai. Kemudian pada akhir daftar berantai terdapat pointer yang tidak menunjuk ke simpul yang lain yang disebut dengan Nil (Pascal) atau NULL (C) (Nil/NULL adalah konstanta dalam pointer). Contoh :



Pada contoh di atas P adalah pointer kepala yang menunjuk awal daftar berantai.

E.1. Membuat Daftar Berantai (*Linked List*)

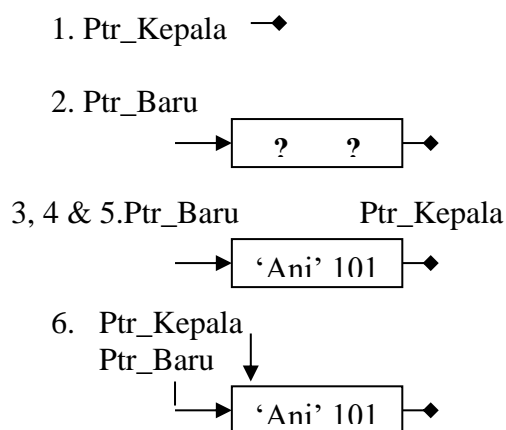
Pembuatan daftar berantai dimulai dengan mengatur agar pointer yang merupakan kepala dari rantai tidak menunjuk ke lokasi manapun.

Algoritmanya adalah sebagai berikut:

Algoritma 9.1.

Procedure Membuat Daftar Berantai(output Ptr_Kepala : PMhs)
<p>Deklarasi</p> <pre>Type PMhs = ^Mhs Mhs = Record < Nama : String[20] Nim : String[10] berikut : PMhs > Ptr_Baru : pMhs</pre>
<p>Deskripsi</p> <ol style="list-style-type: none">1. Ptr_Kepala \leftarrow NULL2. NEW(Ptr_Baru)3. Ptr_Baru->Nama \leftarrow 'ANI'4. Ptr_Baru->Nim \leftarrow '101'5. Ptr_Baru->berikut \leftarrow Ptr_Kepala6. Ptr_Kepala \leftarrow Ptr_Baru7. Jika masih menambah simpul, kembali kelangkah 2;

Langkah-langkah tersebut jika digambarkan adalah sebagai berikut:



Translasi 9.1.

Bahasa C++
<pre>#include <iostream.h> #include <conio.h> class Mhs { public: void Buat_berikut_list(); char Nama[20]; char Nim[10]; Mhs *berikut; };</pre>


```

void Mhs::Buat_berikut_list(){
    Mhs *Ptr_Kepala = NULL;
    Mhs *Ptr_Baru;
    char lagi;
    do {
        Ptr_Baru = new Mhs;
        cout << "\nNama Mahasiswa : ";
        cin >> Ptr_Baru->Nama;
        cout << "Nim Mhs   : ";
        cin >> Ptr_Baru->Nim;
        Ptr_Baru->berikut = Ptr_Kepala;
        Ptr_Kepala = Ptr_Baru;
        cout << "Tambah (y/t) : ";
    } while ((lagi=getch()) != 't');
    cout << "\nAnda telah memasukkan data : ";
    Mhs *cetak = Ptr_Kepala;
    // mencetak isi link list
    while (cetak != NULL) {
        cout << "\nNama Mahasiswa : " << cetak->Nama;
        cout << "\tNim Mhs   : " << cetak->Nim;
        cetak = cetak->berikut;
    }
}

void main() {
    Mhs *simpul;
    simpul->Buat_berikut_list();
    getch();
}

```

Catatan :

Fungsi **getch()** digunakan untuk menerima karakter dari keyboard.

E.2. Menampilkan Isi Daftar Berantai

Untuk menampilkan isi daftar berantai yang telah dibuat, diperlukan suatu pointer lain (Ptr_Sementara) yang digunakan untuk menunjuk pointer kepala. Hal ini dimaksudkan agar pointer kepala tetap menunjuk lokasi yang sama baik sebelum atau sesudah isi daftar berantai ditampilkan.

Algoritma 9.2.

procedure Menampilkan Isi Daftar Berantai(input Ptr_Kepala : PMhs)

Deklarasi

```

Type PMhs = ^Mhs
Mhs = Record <
    Nama : String[20]
    Nim  : String[10]
    berikut : PMhs >

```

Deskripsi

```

while (Ptr_Kepala <> NULL) do
    write(Ptr_Kepala->Nama)
    writePtr_Kepala->Nim)
    Ptr_Kepala ← Ptr_Kepala->berikut { ke node berikutnya }
Endwhile

```

Translasi 9.2.**Bahasa C++**

```

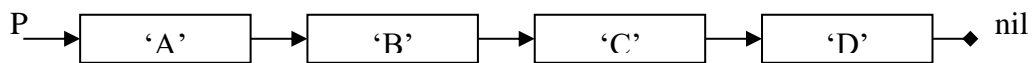
void Lihat_Linked_list(mhs* Ptr_Kepala) {
    while (Ptr_Kepala != NULL) {
        cout << "\nNama Mahasiswa : " << Ptr_Kepala ->Nama;
        cout << "\tNim Mhs    : " << Ptr_Kepala ->Nim;
        Ptr_Kepala = Ptr_Kepala ->berikut;
    }
}

```

10.5 Implementasi Pointer Pada Komputer

Memori utama dari sebuah komputer dapat dipandang sebagai sebarisan lokasi-lokasi memori yang sangat panjang. Setiap lokasi ini dapat menyimpan seuntai angka-angka biner yang dapat diinterpretasikan sebagai data dengan tipe tertentu. Setiap lokasi memori itu diberi satu nomor urut yang disebut sebagai alamat (*address*) dari lokasi memori tersebut.

Misalnya: sebuah link list yang gambarnya intuitifnya sebagai berikut:



Implementasinya dalam memori mungkin seperti berikut, dengan mengingat bahwa 1 karakter menempati memori sebanyak 1 byte atau sama dengan 8 bit, sedangkan setiap variabel bertipe pointer menempati 4 byte atau 32 bit :

...	...
P	1001
1001	'A'
	1006
1006	'B'
	1011
1011	'C'
	1015
1015	'D'
	1020
1020	-1
...	...

Latihan

Modifikasilah struktur data dari algoritma 9.1. dan algoritma 9.2. sehingga sesuai dengan tabel di bawah ini dengan menggunakan link list.

No.	NIM	Nama	Ujian		Nilai	
			Mid	Akhir	Akhir	Huruf
1.	990510001	Khoirul Anam	80	95	90	A
2.	990510002	Siti Zulaiha	45	30	35	D
3.	990510003	Nur Rohmah	50	50	50	C
4.	990510004	Agus Muhammad	90	60	70	B
5.	990510005	Nur Iskandar	40	10	20	E

DAFTAR PUSTAKA

Cormen, H. Thomas, dkk. *Introduction to Algorithms Third Edition*, London, The MIT Press, 2009

Farrel, Mary, *Computer Programming for Teens*, USA, Thomson Learning Inc., 2008

Gozali, William, dan Alham Fikri Aji, *Pemrograman Kompetitif Dasar Versi 1.4*, Indonesia, Ikatan Alumni TOKI

Hanly, Jeri. Hall and Koffmann, Elliot B. *Problem Solving and Program Design in C*, Addison Wesley, 2007

R.G. Dromey, *How Solve It by Computer*, London, Prentice Hall, 1982

Sahni, Sartaj, *Data Structures, Algorithms and Applications in C++ Second Edition*, India, Universities Press, 2005

Schildt, Herbert, *C++ A Beginner's Guide Second Edition*, Osborn, McGraw Hill, 2004

Schildt, Herb, *C++ Programming Cookbook*, Osborn, McGraw Hill, 2008