ISBN 978-623-6833-67-4 (PDF)

ALGORITMA & PEMROGRAMAN DALAM BAHASA C++

Uce Indahyanti, M.Kom
Yunianita Rahmawati, M.Kom

BUKU AJAR
UNIVERSITAS MUHAMMADIYAH SIDOARJO

Algoritma & Pemerograman dalam Bahasa C++

UMSIDA

ALGORITMA & PEMROGRAMAN DALAM BAHASA C++

Uce Indahyanti, M.Kom
Yunianita Rahmawati, M.Kom

BUKU AJAR
UNIVERSITAS MUHAMMADIYAH SIDOARJO

BUKU AJAR
ALGORITMA DAN PEMROGRAMAN
DALAM BAHASA C++

Oleh
Uce Indahyanti, M.Kom
Yunianita Rahmawati, S.Kom. M.Kom

Diterbitkan Oleh : UMSIDA Press

UNIVERSITAS MUHAMMADIYAH SIDOARJO
2020

BUKU AJAR
ALGORITMA DAN PEMROGRAMAN DALAM BAHASA C++

Penulis :
Uce Indahyanti, M.Kom
Yunianita Rahmawati, M.Kom

ISBN :
978-623-6833-67-4

Editor:
Rohman Dijaya, S.Kom, M.Kom

Design Sampul dan Tata Letak:
Mochammad Nashrullah, S.Pd.
Amy Yoga Prajati, S.Kom

Penerbit:
UMSIDA Press
Anggota IKAPI No. 218/Anggota Luar Biasa/JTI/2019
Anggota APPTI No. 002 018 1 09 2017

Redaksi
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No 666B
Sidoarjo, Jawa Timur

Cetakan Pertama, September 2020

KATA PENGANTAR

Alhamdulillah, shalawat dan salam semoga selalu tercurahkan kepada Nabi Muhammad shallallahu ‘alaihi wa sallam. Puji syukur kami panjatkan ke hadirat Allah Ta’ala yang telah memberikan rahmat-Nya sehingga kami bisa menyelesaikan pembuatan buku ajar Algoritma dan Pemrograman Dalam Bahasa C++ ini.

Sesuai dengan Rencana Pembelajaran Semester (RPS) mata kuliah Algoritma dan Pemrograman Prodi Informatika, buku ajar ini membahas langkah-langkah pemecahan masalah yang disebut algoritma, struktur dasar, dan notasi algoritma menggunakan flowchart dan pseudo-code, serta cara menerjemahkannya ke dalam notasi bahasa pemrograman C++ sesuai kaidah yang benar.

Terima kasih kepada Dr. Hindarto, S.Kom., MT., Dekan Fakultas Saintek, Arif Senja Fitrani, S.Kom., M.Kom, Kaprodi Informatika, dan Mochamad Alfan Rosid, S.Kom., M.Kom., SekProdi Informatika yang telah memberikan arahan dan dukungan untuk menyusun buku ajar ini. Serta kepada semua pihak yang telah membantu penyelesaian buku ajar ini. Saran dan kritik sangat kami harapkan untuk mewujudkan buku ajar Algoritma dan Pemrograman yang lebih baik. Semoga bermanfaat.

Tim Penulis

DAFTAR ISI

BATANG TUBUH dan

SUB-CAPAIAN PEMBELAJARAN MATA KULIAH

BAB	Sub-Capaian Pembelajaran Mata Kuliah
BAB I PENGANTAR ALGORITMA & PEMROGRAMAN	1. Mahasiswa mampu menjelaskan definisi algoritma, pemrograman, jenis dan macam bahasa pemrograman, gambaran singkat pengolahan komputer (input-proses-output), serta contoh sederhana penerapan algoritma dalam kehidupan sehari-sehari..
BAB II NOTASI ALGORITMA	1. Mahasiswa mampu menyebutkan notasi algoritma berupa simbol-simbol flowchart dan notasi pseudo code.
BAB III STRUKTUR DASAR ALGORITMA	1. Mahasiswa mampu menjelaskan tiga struktur dasar algoritma, yaitu runtunan, pemilihan dan pengulangan. 2. Mahasiswa mampu menggunakan flowchart dan pseudo-code untuk menulis langkah-langkah pemecahan masalah sesuai struktur dasar yang diberikan.
BAB IV ELEMEN BAHASA PEMROGRAMAN C++	1. Mahasiswa mampu menyebutkan elemen-elemen pemrograman dalam bahasa C++ yaitu variable, tipe data, konstanta, ekspresi dan nilai. 2. Mahasiswa mampu menggunakan elemen pemrograman tersebut di atas sesuai kaidah yang benar.
BAB V TRANSLASI ALGORITMA	1. Mahasiswa mampu menerjemahkan semua struktur dasar algoritma yang dibuat ke dalam notasi bahasa C++ menggunakan perintah IF-

MENGGUNAKAN BAHASA C++	ELSE, SWITCH-CASE, FOR, dan WHILE-DO sesuai kaidah yang benar.
BAB VI PEMROGRAMAN MODULAR	1. Mahasiswa mampu menjelaskan prosedur, fungsi, variable global-lokal dan lingkupnya. 2. Mahasiswa mampu menerjemahkan prosedur dan fungsi termasuk algoritma rekursif ke dalam bahasa C++.
BAB VII ARRAY / LARIK	1. Mahasiswa mampu menjelaskan array / larik , array 1 dan 2 dimensi, dan array bertipe terstruktur (record). 2. Mahasiswa mampu memproses semua bentuk array menggunakan bahasa C++ sesuai kaidah yang benar.

BAB I

PENGANTAR ALGORITMA & PEMROGRAMAN

Sub Capaian Pembelajaran Mata Kuliah :

Mahasiswa mampu menjelaskan definisi algoritma, pemrograman, jenis dan macam bahasa pemrograman, gambaran singkat pengolahan komputer (input-proses-output), serta contoh sederhana penerapan algoritma dalam kehidupan sehari-sehari.

Pokok bahasan :

Terminologi, sejarah, contoh algoritma, pemrograman, dan bahasa pemrograman.

Di era digital seperti saat ini, hampir dapat dipastikan setiap orang dapat mengoperasikan komputer. Tapi pernahkah kita berpikir cara komputer menyelesaikan sebuah masalah, misalnya berupa sebuah perhitungan matematika untuk mencari luas lingkaran? Apakah komputer dapat langsung mengenali rumus mencari luas lingkaran tanpa dimasukkan beberapa parameter tertentu yang diperlukan? Tentu saja tidak, karena komputer hanyalah sebuah mesin yang bekerja berdasarkan data atau parameter yang dimasukkan ke dalamnya, sehingga dapat dihasilkan keluaran sesuai yang diinginkan. Mekanisme ini dikenal dengan I-P-O (input – proses – output). Komputer tidak serta merta dapat mengenali rumus luas lingkaran, dan tidak dapat melakukan proses perhitungan, tanpa melalui sebuah logika penyelesaian secara sistematis menurut standarisasi bahasa yang dikenali atau dimengerti oleh komputer. Nah, dari sanalah muncul konsep algoritma, program, dan bahasa pemrograman.

C. Definisi dan Konsep Algoritma

Istilah algoritma berasal dari kata “algorism” yang merujuk pada seorang ahli matematika muslim yang hidup di abad ke-19 bernama Ibnu Al-Khwarizmi. Pada perkembangannya, konsep pemikiran beliau dalam menguraikan dan memecahkan permasalahan secara logis dan matematis diterapkan sebagai sebuah metode algoritma dalam proses kerja komputer.

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Referensi lain menyebutkan algoritma adalah suatu prosedur yang merupakan urutan langkah-langkah yang berintegrasi, atau suatu metode khusus yang digunakan untuk menyelesaikan suatu masalah yang nyata (Webster Dictionary).

Referensi lain menyebutkan, Algoritma adalah susunan langkah-langkah untuk menyelesaikan suatu persoalan secara logis, efisien dan terstruktur. artinya menghasilkan solusi yang tepat untuk suatu masalah dengan menentukan dengan tepat (Cormen, 2013).

Logis adalah langkah-langkah tersebut benar. Efisien adalah setiap langkah langsung menuju ke tujuan. Terstruktur adalah langkah-langkah tersebut tersusun urut secara logis dan efisien.

Sebenarnya kita telah melakukan algoritma dalam kehidupan sehari-sehari, seperti mengirim email, menginstal aplikasi, mengambil uang di ATM, mengisi pulsa, membuat kue, mengendarai motor, merakit perabotan bongkar pasang, dan lain-lain. Semuanya dilakukan melalui langkah-langkah yang berurutan bukan?

Untuk lebih memberikan gambaran, berikut beberapa contoh algoritma dalam notasi deskriptif.

Contoh 1 : mengendarai sepeda motor :

1. Memasukkan kunci
2. Menyalakan mesin
3. Memasukkan gigi kesatu
4. Memutar pegangan gas
5. Menjalankan motor
6. Menaikkan kecepatan

Contoh 2 : menarik atau mengambil uang di ATM :

1. Memasukkan kartu ATM
2. Memasukkan nomer PIN

3. Memilih menu penarikan uang
4. Memilih besaran atau nominal yang diinginkan
5. Mengambil uang yang dikeluarkan dari mesin (ATM).

Contoh 3 : mempertukarkan isi dari dua gelas A dan B :

1. Tuangkan larutan dari gelas A ke gelas bantuan C (gelas A menjadi kosong)
2. Tuangkan larutan dari gelas B ke gelas A (gelas B menjadi kosong, dan gelas A terisi larutan dari gelas B)
3. Tuangkan larutan dari gelas C ke gelas B (isi kedua gelas tadi sudah tertukar)

Contoh 4 : menghitung luas segi tiga :

1. Masukan nilai alas

2. Masukan nilai tinggi

3. Hitung luas = (alas * tinggi) / 2

4. Cetak luas segitiga

Contoh 5 : menginstal aplikasi android melalui playstore

1. Masuk ke aplikasi playstore
2. Cari aplikasi pada mesin pencari di bagian header (atas)
3. Sistem menampilkan daftar aplikasi sesuai kata kunci yang diinputkan
4. Pilih aplikasi yang dikehendaki
5. Klik tombol download

6. Proses download berjalan
7. Jika gagal, akan muncul pesan download kembali, jika sukses muncul tombol instal
8. Klik tombol Instal, dan tunggu hingga proses instalasi selesai

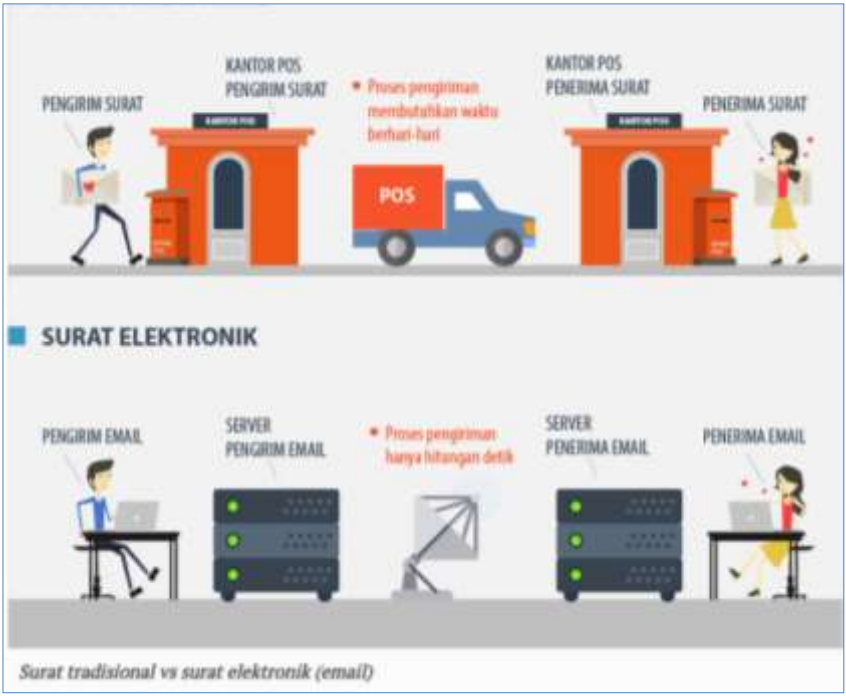
Contoh 6 : mengkoneksikan perangkat ke jaringan wifi

1. Pilih jaringan wifi yang tersedia
2. Masukan username dan password
3. Jika tidak sesuai, sistem menampilkan gagal koneksi
4. Jika sesuai , koneksi internet dapat digunakan.

Contoh 7: mengirim email

1. Login ke situs layanan email sesuai akun terdaftar
2. Pilih tulis email (pesan baru)
3. Masukan alamat email tujuan, subjek, dan tulis isi email
4. Klik tombol kirim

Menyambung contoh algoritma mengirim email di atas, coba perhatikan ilustrasi perbandingan pengiriman pesan secara tradisional melalui surat dengan pengiriman pesan melalui email di bawah ini.



Gambar 1.1. Perbandingan Algoritma Pengiriman Surat dengan Email

(Sumber : Niagahoster.co.id)

Dalam ilustrasi di atas, tampak algoritma pengiriman email tidak jauh berbeda dengan surat biasa. Yang membedakan adalah media penyimpan serta perantara pesannya. Jika surat tradisional ditulis di kertas kemudian dikirim melalui kantor pos dan membutuhkan waktu berhari -hari agar dapat tersampaikan, email ditulis melalui aplikasi layanan email dan disampaikan melalui jaringan internet. Proses penyampaian pesan pun hanya memakan waktu hitungan detik. Tentu ini sangat memudahkan, dan seiring perkembangan jaman, email tak hanya berfungsi untuk berkirim pesan, tapi juga

dapat digunakan untuk mengirim file grafis seperti dokumen, gambar, video, dan audio.

Algoritma mempunyai beberapa karakteristik, yaitu :

1. Menerima beberapa masukan.
2. Memproses masukan melalui langkah-langkah yang berurutan.
3. Setiap langkah harus didefinisikan dengan jelas, sederhana, dan efektif.
4. Urutan langkah tersebut harus terbatas dan berhenti.
5. Menghasilkan keluaran atau output

Algoritma yang terbaik akan menghasilkan output yang benar, tepat guna (efektif) dalam waktu yang relatif singkat dan penggunaan memori yang relatif sedikit.

D. Program dan Bahasa Pemrograman

Dari beberapa sumber dapat disimpulkan terdapat perbedaan diantara program dan pemrograman, berikut definisinya :

- program :
  susunan instruksi dalam bahasa komputer tertentu untuk menyelesaikan masalah (algoritma yang ditulis dalam bahasa komputer tertentu).

- pemrograman :
  aspek-aspek yang berhubungan dengan proses pembuatan program seperti metode, bahasa, tahap pembuatan.

Komputer adalah mesin yang menjalankan perintah-perintah dalam algoritma. Prinsip kerja komputer meliputi input → proses → output.

Sesuai prinsip kerja tersebut, algoritma dimasukkan ke dalam komputer, komputer membaca langkah-langkah intruksi di dalam algoritma, lalu mengerjakan operasi sesuai intruksi tersebut, dan dimunculkan hasilnya (output). Perintah atau intruksi tersebut harus ditulis dalam bahasa yang dipahami oleh komputer (program). Bahasa yang digunakan untuk menulis program itulah yang dinamakan bahasa pemrograman.

Jadi bisa disimpulkan bahasa pemrograman adalah bahasa yang menerjemahkan susunan perintah dalam bahasa komputer tertentu (kode program) yang diberikan oleh pengguna (user) kepada mesin komputer untuk mengerjakan suatu proses (menyelesaikan suatu permasalahan).

Bahasa pemrograman memiliki beberapa karakteristik, yaitu:

1. Memiliki tata bahasa dan aturan tertentu dalam penulisan perintah dan struktur program, pendeklarasian, serta pengoperasian translatornya.
2. Memiliki pustaka interupsi (interrupt library) untuk menerjemahkan perintah yang diinputkan, dan
3. Menggunakan translator yaitu interpreter atau compiler untuk menerjemahkan sintaks pemrograman ke dalam bahasa mesin komputer.

Saat ini terdapat puluhan bahasa pemrograman, yang dapat diklasifikasikan dengan banyak cara. Salah satunya klasifikasi berdasarkan tujuan aplikasinya berikut ini :

- Bertujuan khusus :

bahasa assembly (aplikasi pemrograman mesin), Prolog (aplikasi kecerdasan buatan), Simscript (aplikasi simulasi), dan lainnya.

- Bertujuan umum :
  Bahasa yang dapat digunakan untuk berbagai aplikasi seperti Visual Basic, Pascal, C++, Java, dan lainnya.

Klasifikasi lainnya berdasarkan tingkat pemahaman manusia terhadap bahasa pemrograman tersebut, yaitu :

- Bahasa tingkat rendah :
  Bahasa yang langsung dapat dikerjakan oleh komputer, tanpa melalui penerjemah. Tetapi bahasa ini sulit dipahami oleh manusia, contoh bahasa Assembly.

- Bahasa tingkat tinggi :
  Bahasa yang notasi dan sintaksnya dapat dipahami manusia. Tetapi bahasa ini tidak dapat langsung dikerjakan oleh komputer, perlu melalui penerjemah terlebih dahulu. Semua bahasa pemrograman **kecuali** bahasa mesin dan Assembly merupakan bahasa tingkat tinggi.

Bahasa tingkat tinggi seperti Pascal, Delphi, Visual Basic, C++, Java, Python, dan lainnya, membutuhkan bantuan translator untuk mengkonversi kode program yang dibuat oleh pengguna menjadi bahasa yang dimengerti oleh mesin komputer, yang dikenal dengan istilah translator. Berikut dua jenis translator :

- Interpreter :
  Kode program diterjemahkan tiap baris secara berurutan sampai akhir program. Walaupun ada kesalahan penulisan

kode atau error lainnya pada baris tertentu, baris program lainnya tetap dieksekusi.

- Compiler :
  Semua baris kode program akan diperiksa sampai tidak terjadi error, jika ada kesalahan kode maka proses eksekusi akan dihentikan dan letak kesalahan akan diberitahukan.

Buku ini membahas algoritma dan pemograman menggunakan bahasa pemrograman C++. Bahasa C++ merupakan turunan dari bahasa C yang merupakan induk besar dari banyak bahasa pemrograman yang berkembang saat ini, seperti PHP, Visual C, dan Java. Dan berikut beberapa kelebihan bahasa C++ berdasarkan referensi-referensi yang ada :

1. Bahasa C++ merupakan bahasa pemrograman yang popular.
2. Bahasa C++ banyak dibutuhkan dalam dunia kerja, khususnya yang bergerak di bidang embedded system.
3. Bahasa C++ banyak digunakan untuk membuat aplikasi-aplikasi canggih, contoh Google Earth dan Skype yang modul intinya menggunakan C++.

Perlu dipahami bahwa pemrograman berbeda dengan Bahasa pemrograman. Jika pemrograman merupakan metodologi pemecahan masalah, lalu menuliskan algoritma pemecahan masalahnya ke dalam notasi tertentu. Maka bahasa pemrograman merupakan cara menggunakan suatu bahasa komputer, dengan tata bahasa dan aturan tertentu dalam penulisan perintah dan struktur program, pendeklarasian, serta pengoperasian translatornya.

Algoritma sebagai hasil pemikiran konseptual merupakan alat bantu saja dalam mengkonversikan suatu permasalahan ke dalam bahasa

pemrograman (translasi). Agar algoritma bisa ditranslasikan kedalam bahasa pemrograman, ada beberapa hal yang harus diperhatikan, yaitu:

- Pendeklarasian variabel dan elemen Bahasa pemrograman lainnya.
- Pemilihan tipe data.
- Pemakaian instruksi-instruksi.
- Aturan sintaksis.
- Tampilan hasil.
- Cara pengoperasian compiler atau interpreter.

Sebelum mengerjakan latihan soal di akhir bab ini, mari kita bermain game di bawah ini untuk mengasah logika. Game yang diberi nama "petani serigala dan domba" ini diambil dari Game River Test Pro - Solver Lab di Android.

Dalam bukunya Rinaldi Munir " Algoritma dan Pemrograman Dalam Bahasa Pascal, C, dan C++ Edisi Keenam" membahas persoalan tersebut dan menjelaskan secara rinci tahapan penyelesaian permasalahannya atau algoritmanya.

Gambar 1.2. “Petani Serigala dan Domba”

(Sumber : Game River Test)

Berikut alur game “petani serigala dan domba” di atas :

Seorang petani tiba di tepi sungai dengan membawa seekor domba, seekor serigala, dan sekeranjang sayuran. Mereka hendak menyeberangi sungai dengan menaiki sebuah perahu kecil yang bersandar di tepi sungai. Perahu tersebut hanya dapat memuat satu bawaan saja dan tentu saja si petani tersebut setiap kali menyeberang. Petani berpikir keras cara menyeberangkan semua bawaannya dengan aman. Serigala tidak dapat ditinggal berdua dengan kambing karena akan dimangsanya, dan kambing tidak dapat ditinggal bersama sekeranjang sayur karena akan dimakannya.

Tuliskan algoritma untuk menyeberangkan semua bawaan petani tersebut dari sisi A sampai ke seberang sungai (sisi B) dengan selamat.

Penyelesaian :

Kondisi awal :

sisi A : P, S, D, K (petani, serigala , domba , keranjang sayur)

sisi B : -, -, -, - (kosong)

Algoritma dalam notasi deskriptifnya :

1. Petani menyeberangkan domba dari sisi A ke sisi B
   { sisi A : (-,S,-,K) sisi B : (P,-,D,-) }
2. Petani menyeberang kembali ke sisi A
   { sisi A : (P,S,-,K) sisi B : (-,-,D,-) }
3. Petani menyeberangkan serigala dari sisi A ke sisi B
   { sisi A : (-,-,-,K) sisi B : (P,S,D,-) }
4. Petani menyeberangkan kembali domba dari sisi B ke sisi A (ingat serigala tidak boleh ditinggal berdua dengan domba, karena akan dimangsanya)
   { sisi A : (P,-,D,K) sisi B : (-,S,-,-) }
5. Petani menyeberangkan sayuran dari sisi A ke sisi B
   { sisi A : (-,-,D,-) sisi B : (P,S,-,K) }
6. Petani menyeberang sendirian dari sisi B ke sisi A

{ sisi A : (-,S,-,K) sisi B : (P,-,D,-) }

7. Petani menyeberangkan domba dari sisi A ke sisi B
   { sisi A : (-,-,-,-) sisi B : (P,S,D,K) }

Setelah bermain logika di atas, sekarang kerjakan latihan soal di bawah ini untuk menambah pemahaman anda tentang konsep algoritma dan pemrograman.

Latihan Soal

1. Uraikan kegiatan anda mulai dari menyalakan komputer , mengetik dokumen, sampai dengan menyimpannya ke dalam perangkat keras anda, menjadi sebuah notasi deskriptif algoritma.
2. Tuliskan algoritma untuk membeli buku yang ditawarkan oleh sebuah toko online.
3. Tuliskan algoritma protokol kesehatan yang umum diterapkan di pusat pertokoan selama masa pandemi Covid- 19 ini.
4. Coba buat sebuah permainan logika (game) sederhana, kemudian tuliskan algoritma penyelesaian masalahnya !
5. Terdapat sederet bilangan bulat acak yaitu : 1 4 0 3 6 7, tuliskan cara anda mengurutkan deretan bilangan tersebut secara menaik dalam notasi deskriptif algoritma.
6. Bahasa pemrograman merupakan salah satu jenis perangkat lunak, selain sistem operasi, dan aplikasi. Jelaskan perbedaan di antara ketiganya.

BAB II

NOTASI ALGORITMA

Sub Capaian Pembelajaran Mata Kuliah :

Mahasiswa mampu menyebutkan notasi algoritma berupa simbol-simbol flowchart dan notasi pseudo code.

Pokok bahasan :

Teknik penyajian algoritma yang dinotasikan dalam bentuk flowchart dan pseudo-code.

Untuk menuliskan algoritma, perlu menggunakan notasi tertentu. Ada banyak cara menuliskan notasi algoritma, yang penting algoritma tersebut mudah dibaca dan dipahami. Notasi algoritma dapat berupa:

- pernyataan langkah-langkah dalam deretan kalimat deskriptif
- simbol-simbol flowchart
- pseudo-code

Pada bab ini akan membahas dua model penyajian algoritma, yaitu menggunakan notasi flowchart dan notasi pseudo-code. Penyajian algoritma yang dinotasikan dalam bentuk flowchart dan atau pseudo-code tersebut, bukanlah notasi baku sebagaimana pada notasi bahasa pemrograman.

A. Flowchart

Flowchart (bagan alir) adalah suatu bagan yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir. Flowchart terdiri dari symbol-simbol yang mewakili fungsi-fungsi langkah program dan garis alir (flowlines) yang menunjukkan urutan dari simbol-simbol yang akan dikerjakan. Ada beberapa jenis - Jenis flowchart diantaranya:

1. Bagan alir sistem (systems flowchart).
2. Bagan alir dokumen (document flowchart).
3. Bagan alir skematik (schematic flowchart).
4. Bagan alir program (program flowchart).
5. Bagan alir proses (process flowchart).

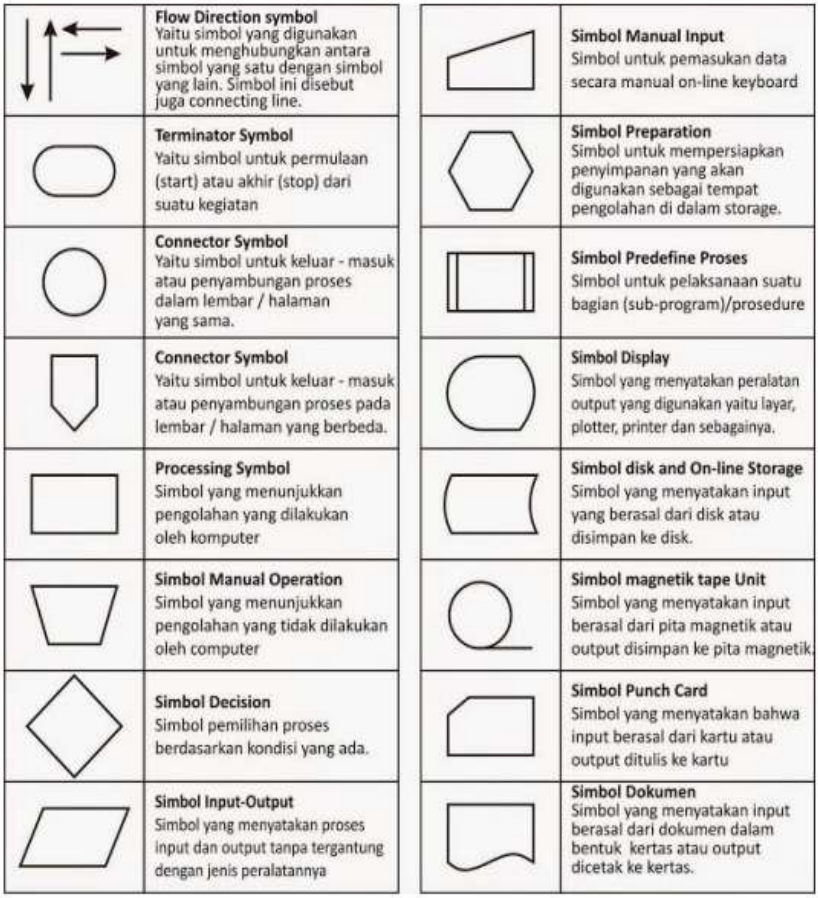
Keterangan :

1. System Flowchart atau bagan alir sistem dapat didefinisikan sebagai bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urut-urutan dari prosedur-prosedur yang ada di dalam sistem. Bagan alir sistem menunjukkan apa yang dikerjakan di sistem.
2. Document Flowchart atau disebut juga bagan alir formulir (form flowchart) atau paperwork flowchart merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya.
3. Schematic Flowchart atau bagan alir skematik merupakan bagan alir yang mirip dengan bagan alir sistem, yaitu untuk menggambarkan prosedur di dalam sistem. Perbedaannya adalah, bagan alir skematik selain menggunakan simbol-simbol bagan alir sistem, juga menggunakan gambar-gambar komputer dan peralatan lainnya yang digunakan. Maksud penggunaan gambar-gambar ini adalah untuk memudahkan komunikasi kepada orang yang kurang paham dengan simbol-simbol bagan alir. Penggunaan gambar-gambar ini memudahkan untuk dipahami, tetapi sulit dan lama menggambarnya.
4. Program Flowchart atau bagan alir program merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. Bagan alir program dibuat dari derivikasi bagan alir sistem. Bagan alir program dapat terdiri dari dua macam, yaitu bagan alir logika program (program logic flowchart) dan bagan alir program komputer terinci (detailed computer program flowchart). Bagan alir logika program digunakan untuk menggambarkan tiap-tiap langkah di dalam program komputer secara logika. Bagan alat-logika program ini dipersiapkan oleh analis sistem. Gambar berikut menunjukkan bagan alir logika program.

Bagan alir program komputer terinci (detailed computer program flow-chart) digunakan untuk menggambarkan instruksi-instruksi program komputer secara terinci. Bagan alir ini dipersiapkan oleh pemrogram.

5. Process Flowchart atau bagan alir proses merupakan bagan alir yang banyak digunakan di teknik industri. Bagan alir ini juga berguna bagi analis sistem untuk menggambarkan proses dalam suatu prosedur.

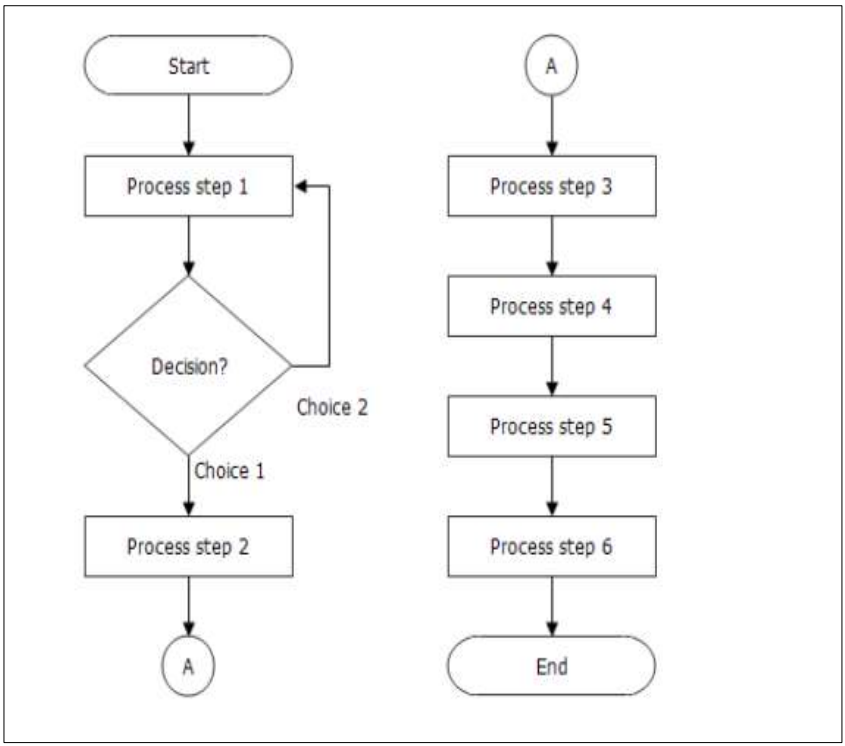
Berikut ini beberapa simbol flowchart menurut ANSI (American National Standard Institute):



Gambar 2.1. Simbol-simbol Flowchart

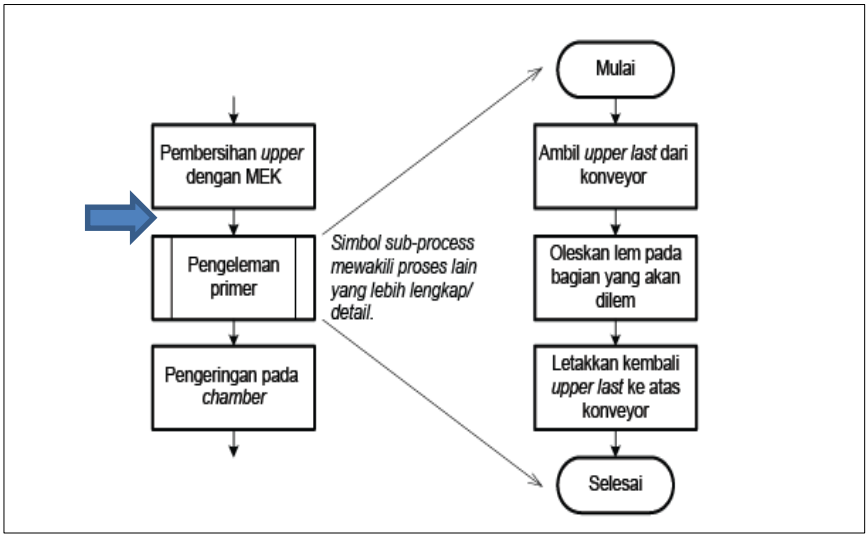
Penyajian algoritma yang dinotasikan dalam bentuk flowchart banyak digunakan, seiring bermunculan berbagai macam bahasa pemrograman. Dengan flowchart memudahkan setiap orang membaca dan memahami aliran penyelesaian masalah (algoritma) karena adanya standarisasi simbol seperti pada gambar di atas.

Berikut ini contoh notasi algoritma dalam bentuk flowchart.



Gambar 2.2. Contoh Flowchart

Dalam penyelesaian suatu masalah, kadang terdapat sebuah proses atau langkah yang kompleks dan besar di dalamnya. Untuk menggambarkan proses tersebut secara sederhana, dapat menggunakan sebuah simbol tertentu saja. Berikutnya proses tersebut dapat dipecah lagi menjadi langkah-langkah yang lebih kecil. Simbol yang dimaksud adalah simbol predefine process (lihat kembali gambar 2.1.), atau simbol sub-proses yang menandakan secara hirarki terdapat flowchart lain yang menjelaskan level proses yang lebih rinci. Berikut contoh penerapannya :



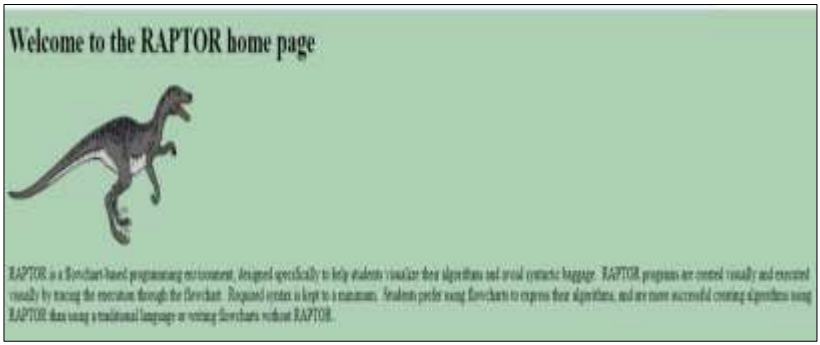
Gambar 2.3. Contoh Sub Proses Pada Pengeleman Sebuah Konveyor

Saat ini tersedia beberapa aplikasi pembuatan flowchart yang dapat dieksekusi. Tidak hanya sekedar dapat membuat atau menggunakan simbol-simbolnya saja dengan mudah, tetapi juga dapat mengetahui proses yang terjadi dalam tiap aliran flowchart yang telah dibuat,

mulai dari masukan atau input sampai dengan dihasilkan keluaran atau output (input – proses – output).

Berikut beberapa contoh aplikasi yang dapat digunakan dalam pembuatan atau penyajian algoritma menggunakan flowchart :

1. Raptor (https://raptor.martincarlisle.com/).
   Seperti dikutip dari halaman utama website resminya, perangkat lunak ini menyediakan lingkungan pemrograman berbasis flowchart (diagram alur), yang dirancang khusus untuk membantu pengguna memvisualisasikan algoritmanya.



Gambar 2.4. Tampilan Awal Aplikasi Raptor

2. Flowgorithm (http://www.flowgorithm.org/).
   Dikutip dari website resminya, aplikasi ini berbasis flowchart yang memungkinkan penggunanya untuk berkonsentrasi pada konsep pemrograman, dan flowgorithm dapat secara interaktif mengubah diagram alur yang dibuat pengguna ke dalam beberapa bahasa pemrograman.

Gambar 2.5. Tampilan Awal Aplikasi Flowgorithm

Dan masih ada beberapa aplikasi sejenis lainnya, tetapi buku ini tidak membahas lebih detil mengenai aplikasi-aplikasi tersebut, penyajian di atas sebagai tambahan pengetahuan saja.

B. Pseudo-code

Pseudo-code berasal dari kata pseudo yang artinya semu atau bukan sebenarnya. Pseudo-code adalah notasi yang mirip dengan notasi bahasa pemrograman tingkat tinggi. Pseudo-code bisa juga disebut kombinasi bahasa “biasa” dengan bahasa pemrograman.

Secara umum tidak ada notasi pseudo-code yang baku untuk menuliskan algoritma, boleh ditulis sesuai “versi” masing-masing pengguna. Artinya pseudo-code “tidak kaku” seperti halnya bahasa pemrograman yang harus ditulis tepat mengikuti kaidah-kaidah yang ada, yang dikenal dengan kode program (*source code*). Notasi baku dalam penulisan kode program diperlukan agar saat program

dijalankan komputer dapat mengenali setiap perintah yang ditulis dan mengeksekusinya.

Agar dapat dijalankan oleh komputer, maka notasi algoritma dalam bentuk pseudo-code harus diterjemahkan ke dalam kode program tertentu atau dalam bentuk notasi bahasa pemrograman, yang disebut dengan proses translasi. Pada beberapa bahasa pemrograman seperti Pascal dan C, antara perintah atau instruksi dengan instruksi berikutnya dipisahkan dengan tanda titik koma atau semicolon (;).

Seperti dua contoh pseudo-code di bawah ini, keduanya untuk menghitung luas lingkaran. Contoh 1 menggunakan "kode" mirip dengan bahasa pemrograman Pascal, dan contoh 2 menggunakan "kode" mirip dengan bahasa pemrograman C++ (turunan dari bahasa pemrograman C) :

Contoh 1 :

deklarasi

var r , luas : real;

algoritma:

read(r);

*luas <– 3.14 * r *r;*

write(luas);

Contoh 2:

deklarasi

float r, luas;

algoritma:

cin>>r;

*luas <– 3.14 * r *r;*

cout<<luas;

Dalam kaidah penulisan bahasa pemrograman yang akan dibahas pada bab berikutnya, terdapat bagian deklarasi variabel yang diperlukan untuk memecahkan permasalahan tertentu. Kemudian terdapat bagian utama (main program) yang berisi langkah-langkah pemecahan masalahnya (algoritma). Notasi pseudo-code mirip dengan notasi pemrograman, notasi ini mengadopsi beberapa keyword dan struktur dalam bahasa pemrograman, namun tidak "mematuhi" semua sintaks dan kaidah yang berlaku dalam bahasa pemrograman.

Pada contoh pseudo-code di atas, disebutkan variabel-variabel yang digunakan dalam bagian deklarasi yaitu variabel r (jari-jari) dan luas yang keduanya bertipe data integer (penjelasan mengenai variabel dan tipe data akan dibahas selengkapnya pada bab berikutnya). Kemudian pada bagian algoritma, disebutkan langkah-langkah

pemecahan masalahnya, yang dimulai dari membaca variabel r (yang diinputkan oleh user), lalu memasukkan nilai r pada rumus luas, dan terakhir mencetak hasilnya yang disimpan dalam variabel luas.

Latihan Soal :

1. Tuliskan algoritma dalam bentuk narasi deskriptif untuk mengambil uang di ATM.
2. Sajikan algoritma no. 1 di atas ke dalam notasi flowchart.
3. Di persimpangan jalan hampir dipastikan kita bisa temui lampu pengatur lalu lintas. Tuliskan algoritma langkah kerja lampu lalin tersebut dalam notasi pseudo code.
4. Buat flowchart untuk menghitung luas bujur sangkar dan mencetak hasilnya.
5. Tuliskan algoritma dalam notasi pseudo-code dan flowchart untuk mencetak teks “algoritma dan pemrograman” sebanyak 100 kali.

BAB III

STRUKTUR DASAR ALGORITMA

Sub Capaian Pembelajaran Mata Kuliah :

3. Mahasiswa mampu menjelaskan tiga struktur dasar algoritma, yaitu runtunan, pemilihan dan pengulangan.
4. Mahasiswa mampu menggunakan flowchart dan pseudo-code untuk menulis langkah-langkah pemecahan masalah sesuai struktur dasar yang diberikan.

Pokok bahasan :

1. Tiga struktur dasar algoritma : runtunan, pemilihan dan perulangan.
2. Penulisan struktur dasar algoritma menggunakan notasi algoritma flowchart dan pseudo-code.

Seperti yang telah disebutkan dalam bab 1, algoritma berisi langkah-langkah penyelesaian sebuah masalah. Langkah-langkah itu dapat diproses secara beruntun lurus ke bawah (struktur runtunan), atau dapat berpindah ke langkah tertentu jika memenuhi syarat tertentu (struktur pemilihan), atau dapat berulang sesuai *counter* yang dikehendaki (struktur perulangan). Berikut pembahasan lengkapnya.

E. Struktur Runtunan

Runtunan adalah struktur algoritma paling dasar yang berisi rangkaian instruksi yang diproses secara sekuensial, satu persatu, mulai dari intsruksi pertama sampai instruksi terakhir . Menurut Goldschlager dkk dalam buku Algoritma dan Pemrograman yang ditulis Rinaldi Munir dkk, pada dasarnya algoritma merupakan runtunan (sequence) satu atau lebih instruksi, yang berarti bahwa :

1. Tiap instruksi dikerjakan satu persatu;
2. Tiap instruksi dikerjakan hanya sekali, tidak ada yang diulang;
3. Urutan instruksi yang dikerjakan pemroses sama dengan urutan istruksi sebagaimana yang tertulis di dalam teks algoritmanya;
4. Instruksi terakhir merupakan akhir algoritma.

Beberapa contoh struktur algoritma runtunan :

1. Menghitung luas lingkaran
    - Masukkan nilai jari-jari (R)
    - Hitung L = 3.14 * R * R

- Cetak Luas Lingkaran (L)

2. Menjumlahkan dua bilangan
    - Masukkan nilai bilangan pertama (Bil 1)
    - Masukkan nilai bilangan kedua (Bil 2)
    - Hitung Hasil = Bil 1 + Bil 2
    - Cetak hasil penjumlahan (Hasil)

3. Menghitung pembayaran setelah potongan atau diskon :
    - Masukkan harga barang (hrg)
    - Masukkan potongan (p)
    - Hitung potongan harga (hrgpot) = hrg * p
    - Hitung pembayaran (bayar) = hrg – hrgpot
    - Cetak pembayaran (bayar)

F. Struktur Pemilihan

Pada struktur pemilihan atau percabangan, terdapat pemeriksaan kondisi / syarat yang harus dipenuhi, yang kemudian akan memilih perintah apa yang akan dilakukan jika syarat tersebut dipenuhi. Perintah tidak lagi dikerjakan secara beruntun seperti pada struktur runtunan, tetapi berdasarkan syarat yang harus dipenuhi.

Notasi algoritma untuk struktur pemilihan menggunakan dua konstruksi berikut ini :

1. IF – THEN
2. CASE

Berikut penjelasan masing-masing konstruksi di atas :

1. Konstruksi IF – THEN

Konstruksi IF – THEN dibagi lagi menjadi :

a) Satu aksi : IF - THEN
b) Dua aksi : IF – THEN – ELSE
c) Tiga aksi atau lebih : IF – THEN – ELSE

Berikut penjelasan masing-masing konstruksi IF di atas :

a) Satu aksi :

Bila kondisi yang diseleksi terpenuhi maka aksi atau *statement* yang mengikuti "then" yang akan diproses. Dan jika sebaliknya maka tidak diminta aksi apapun.

Konstruksi ini berbentuk :

If <kondisi> then

aksi

end if

Contoh :

Tuliskan algoritma untuk menampilkan keterangan "bilangan bulat positif" jika bilangan bulat yang diinputkan adalah positif.

Seperti diketahui bilangan positif merupakan bilangan yang nilainya lebih besar sama dengan nol (>=0). Jadi jika bilangan bulat yang diinputkan >= 0 maka akan muncul keterangan "bilangan bulat positif", tetapi jika < 0 tidak diminta aksi apapun.

Berikut algoritmanya :

if bil_bulat >= 0 then

write "bilangan bulat positif"

end if

b) Dua aksi :

Bila kondisi terpenuhi maka aksi atau statement 1 yang akan diproses, jika tidak maka statement 2 yang akan diproses.

Konstruksi ini berbentuk :

If <kondisi>

then <statement 1>

else <statement 2>

Contoh :

Tuliskan algoritma untuk menampilkan keterangan "bilangan genap" atau "bilangan ganjil". Jika bilangan bulat positif yang diinputkan habis dibagi 2, maka akan muncul "bilangan genap", tapi jika sebaliknya muncul "bilangan ganjil".

Dalam dunia pemrograman dikenal operator modulus yang disingkat mod. Operator tersebut berfungsi untuk mencari sisa pembagian. Macam-macam operator akan dibahas lebih detil pada bab lain.

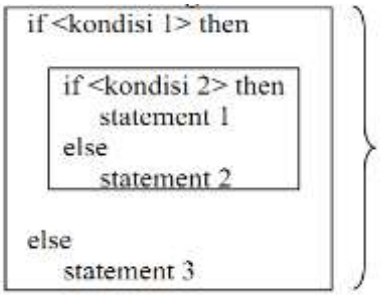
Berikut algoritmanya :

```
if (bil mod 2) = 0 then
  write "bilangan genap"
else
  write "bilangan ganjil"
```

c) Tiga aksi atau lebih:

Konstruksi IF – THEN pada tiga aksi atau lebih, dapat juga disebut IF Bersarang atau IF Bertingkat. Dimana terdapat

perintah IF di dalam IF (*nested if*). Ilustrasi konstruksi tersebut dapat dilihat pada gambar di bawah ini :



Gambar 3.1. Konstruksi IF – THEN Bersarang

Berikut penjelasan proses pada gambar di atas :

- Jika kondisi 1 terpenuhi maka cek kondisi 2 :
  - jika kondisi 2 terpenuhi maka kerjakan aksi atau statement 1,
  - jika kondisi 2 tidak terpenuhi maka kerjakan aksi atau statement 2.
- Jika kondisi 1 tidak terpenuhi maka kerjakan statement 3

Contoh :

Tuliskan algoritma untuk menghitung total pembayaran barang di sebuah swalayan, dengan ketentuan sebagai berikut :

- Jika total pembelian >= 100000, maka memperoleh diskon dengan syarat atau kondisi sebagai berikut :
    - Jika pembeli mempunyai kartu anggota, maka memperoleh diskon 20%,
    - jika bukan anggota atau member maka memperoleh diskon 10%
- Jika total pembelian < 100000, maka tidak memperoleh diskon (diskon = 0)

Berikut algoritmanya :

If beli >= 100000 then

{ If pembeli = member then

diskon = beli * 0.20

else if

diskon = beli * 0.10 }

else diskon = 0

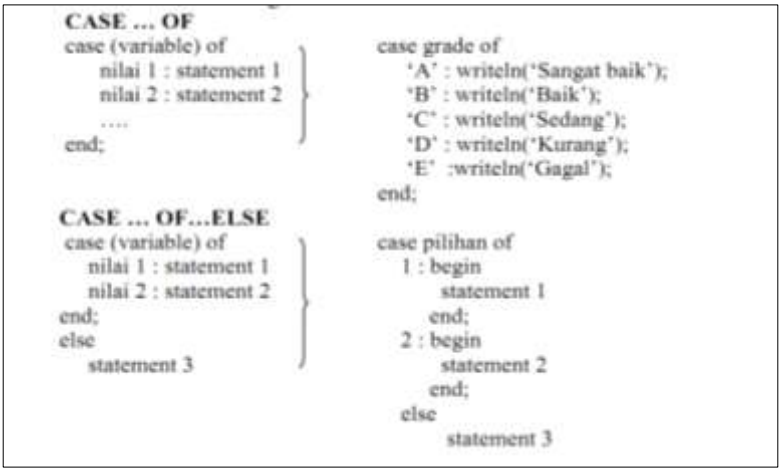
2. Konstruksi CASE

Konstruksi CASE digunakan untuk masalah dengan dua kasus atau lebih. Konstruksi ini dapat menyederhanakan perintah if-then-else yang bertingkat-tingkat ataupun bersarang untuk kasus tertentu.

Statement CASE dibagi menjadi 2 struktur:

a) CASE – OF
b) CASE – OF – ELSE

Perhatikan perbedaan struktur CASE – OF dengan CASE – OF – ELSE pada gambar di bawah ini :



Gambar 3.2. Konstruksi CASE – OF

Penjelasan gambar 3.2. di atas :

- Pada konstruksi case – of, variabel yang diinputkan dengan nama 'grade' akan dibaca (dicek) : jika 'A' maka dicetak keterangan 'Sangat Baik', jika 'B' maka dicetak keterangan 'Baik', dan seterusnya sampai 'E', lalu selesai.
- Pada konstruksi case – of – else, varibel yang diinputkan dengan nama 'pilihan' akan dibaca : jika pilihan = 1, maka dilakukan statement 1, jika 2 dilakukan aksi atau

statement 2. Tetapi jika pilihan yang diinputkan selain 1 dan 2, maka akan dilakukan statement 3

Contoh lain case – of dan case – of – else :

- Konversi angka menjadi nama hari menggunakan konstruksi case – of :

```
Case angka of

    1 : write ('Senin')

    2 : write ('Selasa')

    3 : write ('Rabu')

    4 : write ('Kamis')

    5 : write ('Jumat')

    6 : write ('Sabtu')

    7 : write ('Minggu')

End;
```

- Konversi grade (dalam bentuk huruf) menjadi keterangan pencapaian menggunakan konstruksi case – of :

```
case (grade) of

  'A' :

    begin

      writeln('Sangat Memuaskan');
```

```
    writeln('Pertahankan!');
   end;
  'B' :
   begin
   writeln('Memuaskan');
   writeln('Tingkatkan');
   end;
  'C','D' :
   begin
    writeln('Cukup Baik');
    writeln('Belajar lebih giat lagi ya');
   end;
  'E' :
   begin
    writeln('Maaf, anda tidak lolos');
    writeln('Tetap semangat, belajar lebih keras');
   end
 else
```

```
    writeln('Maaf, huruf yang anda masukkan salah');

    writeln('Masukkan grade berupa huruf: A, B, C, D, atau E saja');

   end;
```

G. Struktur Perulangan

Perulangan merupakan struktur dimana terdapat proses pengulangan perintah yang sama sebanyak n kali. Struktur ini merupakan salah satu kelebihan yang dimiliki oleh mesin komputer. Sebagai contoh untuk mencetak teks “Algoritma Pemrograman” sebanyak 100 kali pada layar monitor, hanya diperlukan beberapa baris perintah meggunakan teknik atau struktur perulangan tersebut. Tanpa harus menuliskan perintah yang sama sebanyak 100 kali.

Algoritma menggunakan struktur perulangan banyak dijumpai dalam kehidupan sehari-hari, contoh mencetak sekian ratus surat undangan yang sama, menampilkan sejumlah deret bilangan ganjil atau bilangan genap, atau deret bilangan prima, atau deret bilangan dengan rumus deret tertentu, mengambil sejumlah uang di ATM dengan pecahan tertentu, dan masih banyak contoh lainnya.

Terdapat dua jenis perulangan yang dikenal dalam algoritma, yaitu :

1. ***Unconditional looping*** yaitu perulangan yang tidak menyertakan kondisi tertentu sebagai syarat terjadinya perulangan perintah, contoh perulangan dengan **FOR - DO**.

2. ***Conditional looping*** yaitu perulangan dengan adanya kondisi atau syarat yang harus dipenuhi, contoh perulangan dengan **WHILE -DO** dan **REPEAT - UNTIL**.

1. Konstruksi perintah FOR - DO :

Konstruksi ini menetapkan jumlah perulangan sebelum perulangan tersebut dilakukan. Jumlah perulangan telah didefinisikan pada awal deklarasi perintah ini. Berikut format penulisan perintahnya dalam bentuk notasi pseudo-code :

for [variabel pengulang] ← *nilai_awal to nilai_akhir do*

statement 1

statement 2

…

end for

contoh perintah FOR – DO untuk mencetak teks " Algoritma Pemrograman" sebanyak 100 kali, dimana i adalah variabel pengulangnya :

for i ← *1 to 100 do*

write ("Algoritma Pemrograman")

end for

Konstruksi FOR-DO juga dapat digunakan untuk perulangan secara menurun (descending), yaitu menggunakan ***down to***. Contoh mencetak deret bilangan 5 4 3 2 1, maka format penulisan perintahnya sebagai berikut :

For x ← 5 ***down to*** *1 do*

Write (x)

End for

Variabel pengulang x sekaligus sebagai nilai yang akan dicetak, perhatikan perintah pada baris kedua di atas yaitu ***write(x).***

2. Konstruksi perintah WHILE - DO :

Konstruksi ini melakukan pengecekan kondisi perulangan terlebih dahulu sebelum melanjutkan ke perintah berikutnya, apakah sudah terpenuhi (true) sesuai yang disyaratkan atau tidak. Berikut format penulisan perintahnya :

while [kondisi] do

statement 1

statement 2

……

end while

contoh perintah WHILE – DO untuk mencetak teks " Algoritma Pemrograman" sebanyak 100 kali :

```
i = 1
while i <= 100 do
  write ("Algoritma Pemrograman")
  i = i + 1
end while
```

Pada awal deklarasi ditentukan nilai awal variabel i = 1, perintah berikutnya mengecek kondisi apakah nilai i <= 100, selama kondisi tersebut terpenuhi maka akan dicetak teks yang dimaksud. Variabel pengulang dinaikkan satu persatu (i = i +1), dan kembali ke perintah pengecekan kondisi, apakah nilai i yang terkini masih <= 100, jika terpenuhi maka teks "Algoritma Pemrograman" akan terus dicetak, sampai dengan kondisi menjad tidak terpenuhi yaitu i > 100, maka perintah perulangan akan dihentikan.

Konstruksi while-do juga dapat digunakan untuk perulangan dengan nilai menurun (descending). Sebagai contoh untuk mencetak deret bilangan : 5 4 3 2 1, maka format penulisan perintahnya :

```
i = 5
while i >= 1 do
  write ( i )
```

i = i - 1

end while

Perhatikan tiap baris perintah di atas, nilai awal variabel i ditentukan = 5, kondisi yang dicek apakah nilai i >= 1, jika terpenuhi maka perintah akan terus diulang, dan nilai pengulang diturunkan satu persatu (i = i -1). Perulangan akan dihentikan jika kondisi sudah tidak terpenuhi (false).

3. Konstruksi perintah REPEAT – UNTIL :

Konstruksi ini agak berbeda dengan WHILE – DO yang melakukan pengecekan kondisi di awal perulangan. Perulangan REPEAT – UNTIL dilakukan terlebih dahulu tanpa pengecekan kondisi, dan akan dihentikan sampai kondisi menjadi terpenuhi (true).

Berikut format penulisan perintahnya :

repeat

statement 1

statement 2

……

until [kondisi]

contoh perintah REPEAT - UNTIL untuk mencetak teks " Algoritma Pemrograman" sebanyak 100 kali :

```
i = 1
repeat
  write ("Algoritma Pemrograman")
  i = i + 1
until i > 100
```

Konstruksi repeat – until juga dapat digunakan untuk perulangan dengan nilai menurun (descending). Sebagai contoh untuk mencetak deret bilangan : 5 4 3 2 1, maka format penulisan perintahnya :

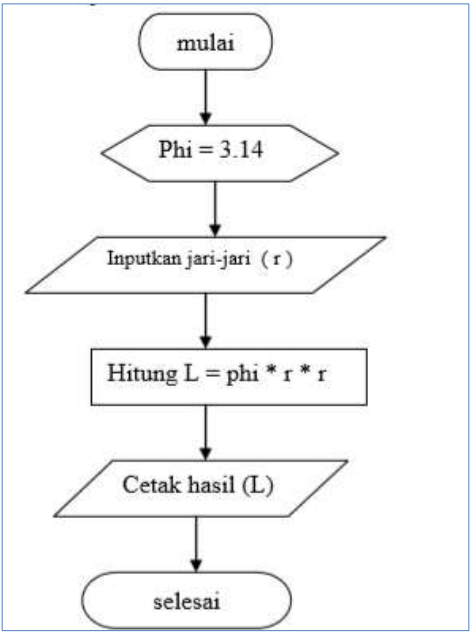
```
i = 5
repeat
  write ( i )
  i = i - 1
until i < 1
```

Perhatikan tiap baris perintah di atas, nilai awal variabel i ditentukan = 5, perintah terus diulang, nilai pengulang diturunkan satu persatu (i = i -1). Perulangan akan dihentikan jika kondisi sudah terpenuhi (true) yaitu nilai i < 1.

Dalam struktur perulangan juga terdapat struktur perulangan bersarang, dimana terdapat perulangan di dalam perulangan, contoh terdapat For -Do di dalam For-Do pada kasus membaca data matrik. Matrik atau array dua dimensi terdiri dari baris dan kolom, dimana pembacaan data dimulai dari baris ke-1 lanjut proses pembacaan data pada tiap kolom yang terdapat dalam baris ke-1 tersebut. Hal itu dilakukan berulang sampai 1 baris habis terbaca, lanjut ke baris ke-2. Pada baris ke-2 dilakukan pembacaan data pada kolom baris tersebut satu persatu, dan begitu seterusnya proses berulang sampai semua data pada semua baris dan kolom habis terbaca. Pembacaan data matrik atau array dua dimensi ini akan dibahas pada bab terkait selanjutnya.

H. Struktur Dasar Algoritma Dalam Bentuk Flowchart

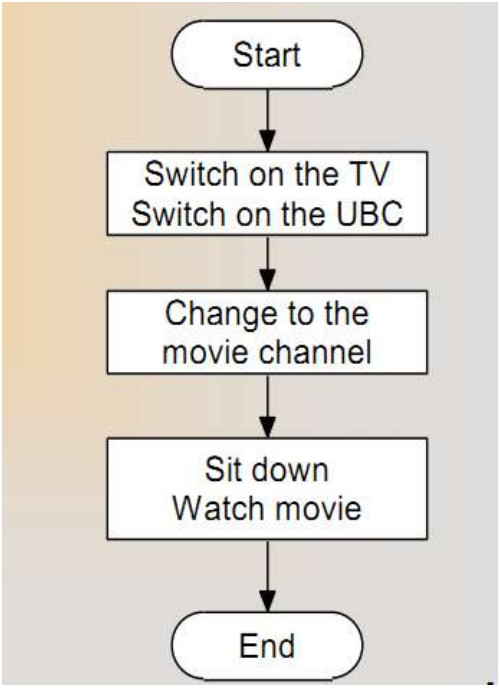
Struktur dasar algoritma juga dapat dinotasikan dalam bentuk flowchart. Di bawah ini diberikan beberapa contoh flowchart yang menggambarkan alur permasalahan dengan struktur runtunan, pemilihan, dan perulangan. Contoh pertama menunjukkan semua proses dilakukan secara berurutan dari atas sampai ke bawah sampai selesai. Tidak ada percabangan ataupun pengulangan di dalamnya.



Gambar 3.3. Flowchart Dengan Struktur Runtunan

Diawali dengan simbol terminal ‘mulai’, kemudian symbol untuk menentukan harga awal yaitu phi = 3.14, dilanjutkan memasukkan variabel r (jari-jari), dan dihitung dengan rumus L =phi * r * r, kemudian mencetak hasilnya (yang disimpan dalam variabel L), terakhir terminal selesai.

Contoh berikutnya di bawah ini juga menggunakan flowchart dengan struktur runtunan untuk persoalan menonton film di rumah.

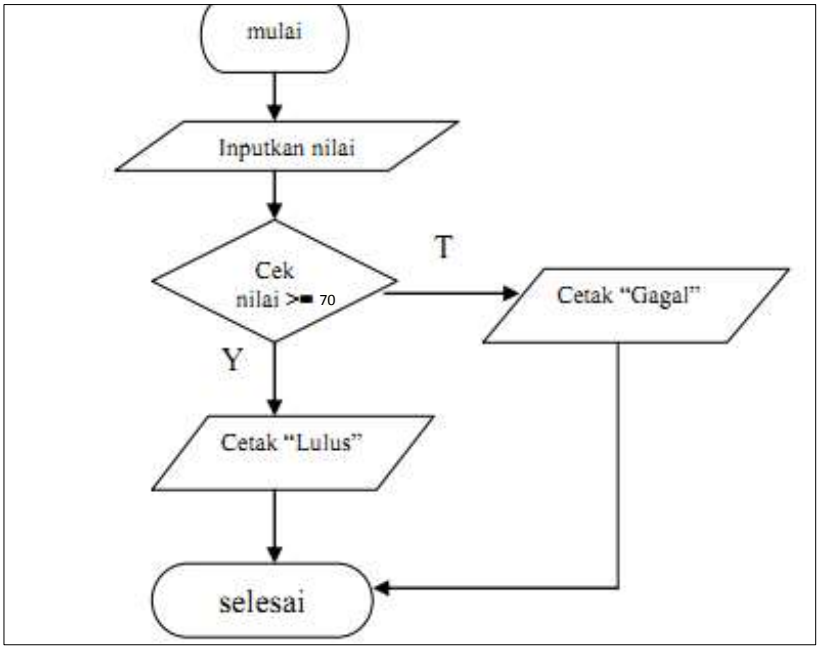


Gambar 3.4. Flowchart Struktur Runtunan

Alur penyelesaian masalah pada gambar 3.4. di atas, digambarkan dalam bentuk flowchart yang diawali oleh terminal start, lalu proses menyalakan UBC TV, proses ganti ke kanal film, proses duduk menonton film, kemudian diakhiri simbol terminal end.

Berikutnya contoh-contoh flowchart dengan struktur pemilihan atau percabangan, yang tentu mempunyai aliran bercabang untuk memilih atau mengecek kondisi tertentu. Contoh flowchart yang

menggambarkan persoalan menampilkan keterangan “Lulus” atau “Gagal” berdasarkan nilai yang dimasukkan.

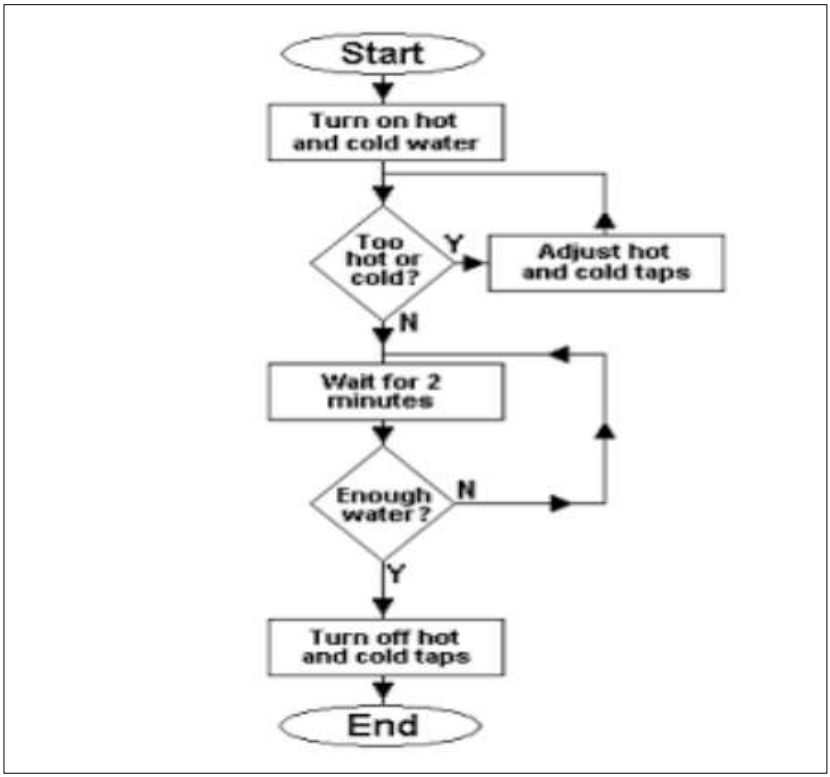


Gambar 3.5. Flowchart Struktur Pemilihan

Flowchart di atas menggambarkan alur pengecekan nilai yang diinputkan. Jika nilai yang diinputkan >= 70 maka dicetak keterangan “Lulus”, jika nilai < 70 maka dicetak keterangan “ Gagal”.

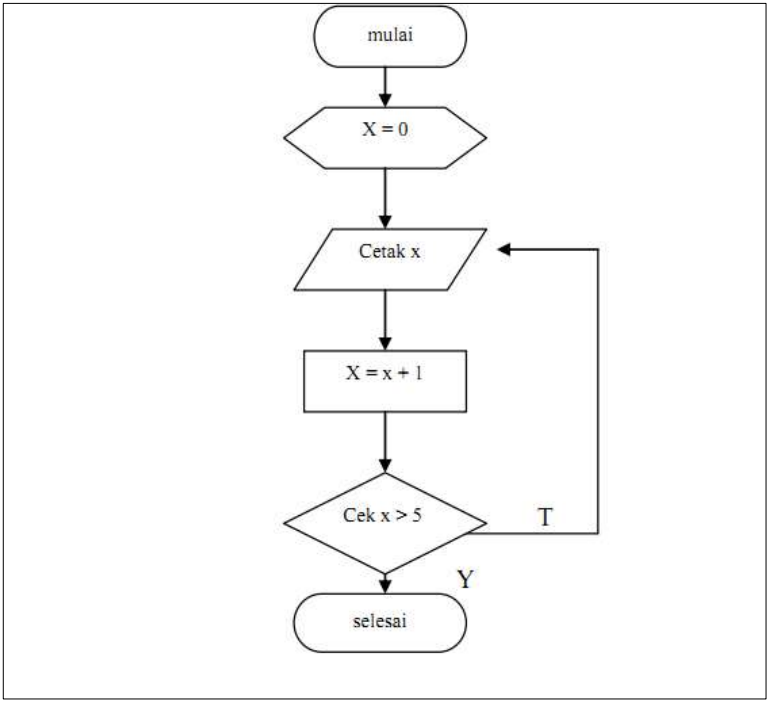
Contoh lainnya dapat dilihat pada gambar 3.6. dibawah ini, yaitu langkah memanaskan atau mendinginkan air pada sebuah mesin atau

semacam dispenser. Pada persoalan tersebut menggunakan kombinasi struktur pemilihan dan struktur perulangan. Di dalamnya terdapat percabangan untuk mengecek suhu air, apakah terlalu panas atau terlalu dingin? Jika yam aka akan disesuaikan suhunya, dan proses kembali ke perintah sebelumnya. Begitu juga pada percabangan untuk mengecek volume air yang akan dituangkan, apakah air dirasa cukup? Jika ya maka proses kembali ke perintah sebelumnya.



Gambar 3.6. Flowchart Struktur Perulangan 1

Berikutnya contoh-contoh flowchart dengan struktur perulangan. Di atas sudah dijelaskan, bahwa struktur perulangan merupakan struktur dimana terdapat proses pengulangan perintah yang sama sebanyak ***n*** kali. Beberapa persoalan yang dapat diselesaikan menggunakan struktur ini antara lain menampilkan deret bilangan seperti yang digambarkan pada flowchart di bawah ini.



Gambar 3.7. Flowchart Struktur Perulangan

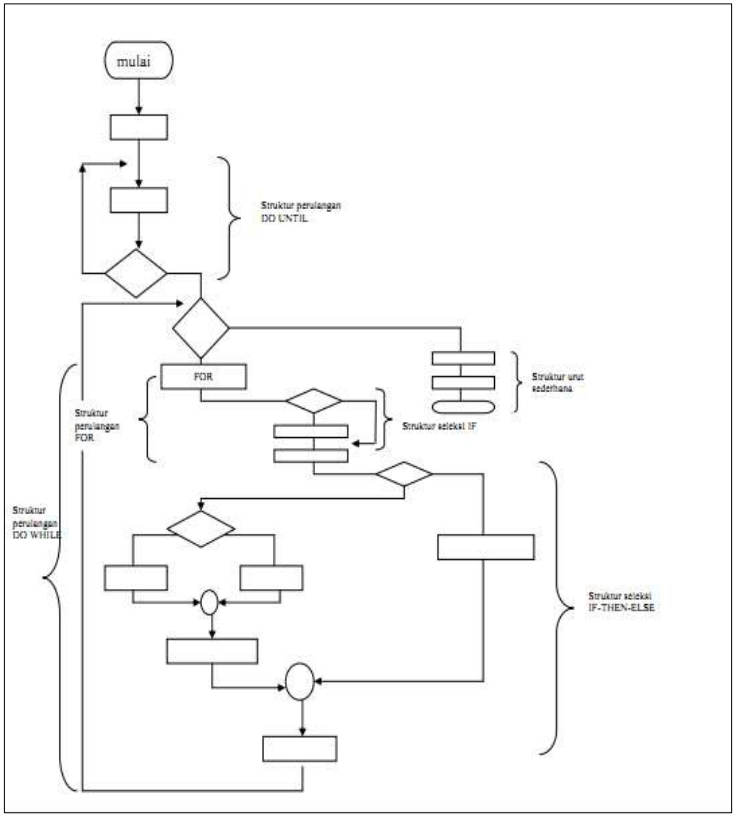
Flowchart di atas menggambarkan aliran langkah-langkah untuk menampilkan deret bilangan bulat 0 1 2 3 4 5.

Berikut algoritmanya dalam notasi deskriptif :

1. Mulai
2. Beri nilai awal 0 pada variabel X
3. Cetak nilai yang tersimpan dalam variabel X
4. Tambahkan 1 pada nilai X
5. Cek nilai X, apakah > 5 ?

   jika Ya maka proses selesai.
   jika Tidak maka proses kembali ke cetak nilai X, lalu ulangi lakukan langkah 4 dan 5, dan seterusnya sampai kondisi terpenuhi (X > 5).
6. Selesai.

Di bawah ini merupakan Ilustrasi alur program yang di dalamnya terdapat gabungan tiga bentuk struktur dasar algoritma, termasuk terdapat bentuk struktur yang bersarang :



Gambar 3.8. Ilustrasi Gabungan Bermacam Struktur Algoritma

Latihan soal :

1. Tuliskan flowchart untuk persoalan sebagai berikut :
   Sebuah perusahaan menerapkan seleksi calon karyawannya berdasarkan usia dengan ketentuan sebagai berikut :
   - Jika usia peserta seleksi antara 20 sampai 25 tahun (bisa menggunakan operator logika ***and***), maka cetak "DIterima"
   - Jika selain usia di atas maka cetak "Ditolak"
2. Untuk mencetak keterangan bilangan genap atau ganjil berdasarkan bilangan bulat yang diinputkan, menggunakan struktur dasar algoritma apa dan gambarkan diagram alirnya !
3. Tuliskan pseudo-code untuk mencetak deret bilangan genap mulai 2 sampai dengan 100 menggunakan konstruksi for – do.
4. Cari sebuah kasus atau permasalahan dimana algoritma penyelesaiannya mengkombinasikan struktur pemilihan/percabangan dengan struktur perulangan, kemudian gambarkan dalam bentuk diagram alir (notasi flowchart).

BAB IV

ELEMEN BAHASA PEMROGRAMAN C++

Sub Capaian Pembelajaran Mata Kuliah :

3. Mahasiswa mampu menyebutkan elemen-elemen pemrograman dalam bahasa C++ yaitu variable, tipe data, konstanta, ekspresi dan nilai.
4. Mahasiswa mampu menggunakan elemen pemrograman tersebut di atas sesuai kaidah yang benar.

Pokok bahasan :

1. Elemen-elemen dasar pemrograman dalam Bahasa C++ : variable, tipe data, konstanta, ekspresi dan nilai.
2. Penggunaan elemen dasar pemrograman sesuai kaidah yang benar.

Setiap proses dalam program komputer pada dasarnya akan memanipulasi objek atau data yang tersimpan di dalam memori, yang tentunya perlu diberi nama. Setiap objek dapat berupa variabel (peubah), konstanta, fungsi atau prosedur. Dan setiap data yang berupa variabel atau konstanta memiliki tipe data tertentu. Tipe data menentukan nilai yang dapat dimilikinya dan operasi yang dapat dilakukannya. Tipe data dapat dikelompokkan menjadi tipe data dasar yaitu tipe data yang dapat langsung dipakai, dan tipe data bentukan yang merupakan turunan dari tipe data dasar.

Variabel, konstanta, tipe data, ekspresi atau operator, dan nilai itulah yang merupakan elemen-elemen dasar pemrograman. Sebelum membahas satu persatu elemen-elemen tersebut di atas, perhatikan terlebih dahulu contoh program C++ dalam gambar berikut ini :

```
//*************************************************************
// This is a simple C++ program. It displays four lines
// of text, including the sum of two numbers.
//*************************************************************

#include <iostream>

using namespace std;

int main()
{
    int num;

    num = 6;

    cout << "My first C++ program." << endl;
    cout << "The sum of 2 and 3 = " << 5 << endl;
    cout << "7 + 8 = " << 7 + 8 << endl;
    cout << "Num = " << num << endl;

    return 0;
}
```

Sample Run: (When you compile and execute this program, the following four lines are displayed on the screen.)

```
My first C++ program.
The sum of 2 and 3 = 5
7 + 8 = 15
Num = 6
```

Gambar 4.1. Contoh Program Dalam Bahasa C++

Penjelasan program :

1. Penulisan "//" digunakan untuk memberikan keterangan program. Perintah ini tidak akan dieksekusi saat program dijalankan (run). Program di atas bertujuan untuk menjumlahkan dua bilangan.
2. Pada baris pertama terdapat perintah ***#include <iostream>*** yang digunakan untuk memanggil dan memanipulasi perintah input -output (***cin - cout***).
3. Maksud perintah berikutnya ***using namespace std;*** perintah ini digunakan untuk mendeklarasikan kepada compiler

bahwa user akan menggunakan semua fungsi/class/file yang terdapat dalam paket ***namespace std*.**

4. Pada baris ke 7 terdapat perintah int main() yang merupakan judul dari fungsi utama. Pada baris ke 8 terdapat tanda buka kurawal pembuka { yang menandai dari awal dari main program (fungsi utama). Tanda kurawal penutup } terdpaat pada baris terakhir dari program, yang menandai akhir main program. Perhatikan bahwa dalam C ++, tanda << adalah operator, yang disebut the stream insertion operator, contoh : cout << "My first C++ program." << endl; Itu adalah contoh pernyataan keluaran C ++, yang menyebabkan komputer mengevaluasi ekspresi setelah pasangan simbol << dan menampilkan hasilnya di layar.
5. Biasanya program C ++ berisi berbagai jenis ekspresi seperti aritmatika dan string. Misalnya, 7 + 8 adalah ekspresi aritmatika. Dan apapun yang berada dalam tanda kutip ganda adalah string. Misalnya, "Program C ++ pertama saya." dan "7 + 8 =" adalah string.
6. Perhatikan pernyataan output ini : cout << "Jumlah 2 dan 3 =" << 5 << endl; Pernyataan tersebut terdiri dari dua ekspresi. Ekspresi pertama adalah "Jumlah 2 dan 3 =" , dan ekspresi kedua terdiri dari angka 5. Ekspresi "Jumlah 2 dan 3 =" adalah string yang akan ditampilkan apa adanya sesuai yang tertulis. Ekspresi kedua, yang terdiri dari angka 5 dievaluasi menjadi 5. Dengan demikian, output dari pernyataan di atas adalah: Jumlah 2 dan 3 = 5.
7. Pernyataan berikutnya : cout << "7 + 8 =" << 7 + 8 << endl; Dalam pernyataan ini, ekspresi "7 + 8 =" merupakan string, yang akan menghasilkan keluaran atau output persis seperti yang tertulis. Pada pernyataan kedua, 7 + 8, ekspresi ini terdiri dari angka 7 dan 8 yang dikenali sebgai operator aritmatika oleh C ++. Oleh karena itu, hasil dari ekspresi 7 + 8 adalah jumlah dari 7 dan 8, yaitu 15. Jadi, output dari pernyataan sebelumnya adalah: 7 + 8 = 15.

8. Berikutnya : cout << "Num =" << num << endl; Pernyataan ini terdiri dari string "Num =", yang mengevaluasi dirinya sendiri, dan kata num. Pernyataan num = 6; menetapkan nilai 6 hingga num. Oleh karena itu, ekspresi num, setelah yang kedua <<, dievaluasi menjadi 6. Sekarang berarti output dari pernyataan sebelumnya adalah: Num = 6
9. Pernyataan terakhir, yaitu ***return 0***; mengembalikan nilai 0 ke sistem operasi ketika program berakhir.
10. Catatan tentang kata kunci (***keywords***) merupakan simbol kata. Beberapa simbol kata termasuk dalam keywords antara lain: int, float, double, char, const, void, return.
11. ***Reserved words*** atau kata - kata yang dicadangkan juga disebut kata kunci. Huruf-huruf yang membentuk kata khusus selalu lebih kecil. Seperti simbol khusus, masing-masing dianggap sebagai simbol tunggal. Selain itu, simbol kata tidak dapat didefinisikan ulang dalam program apa pun; yaitu, mereka tidak dapat digunakan untuk apa pun selain penggunaan yang dimaksudkan. MIsal : kata "***class***" tidak dapat dipakai sebagai nama variabel dalam program C++, karena ***class*** merupakan salah satu ***keywords.***

C++ Keywords					
and	*and_eq*	*asm*	*auto*	*bitand*	*bitor*
bool	*break*	*case*	*catch*	*char*	*class*
compl	*const*	*const_cast*	*continue*	*default*	*delete*
do	*double*	*dynamic_cast*	*else*	*enum*	*explicit*
export	*extern*	*false*	*float*	*for*	*friend*
goto	*if*	*inline*	*int*	*long*	*mutable*
namespace	*new*	*not*	*not_eq*	*operator*	*or*
or_eq	*private*	*protected*	*public*	*register*	*reinterpret_cast*
return	*short*	*signed*	*sizeof*	*static*	*static_cast*
struct	*switch*	*template*	*this*	*throw*	*true*
try	*typedef*	*typeid*	*typename*	*union*	*unsigned*
using	*virtual*	*void*	*volatile*	*wchar_t*	*while*
xor	*xor_eq*				

Gambar 4.2. Keywords C++

Dalam penulisan program secara umum, terdiri dari 2 bagian utama,

yaitu bagian deklarasi & bagian statement. Bagian deklarasi merupakan bagian program untuk mendefinisikan atau mendeklarasikan variabel, konstanta, dan tipe datanya. Selain itu bagian deklarasi dapat juga digunakan untuk memberi nilai awal suatu variable, dengan kata lain deklarasi digunakan untuk memperkenalkan suatu nama kepada *compiler*.

Deklarasi di dalam Bahasa Pascal, dimana tipe data disebutkan setelah penamaan variabel atau konstantanya. Di bawah ini variabel yang dideklarasikan adalah r dan luas yang bertipe data integer

(bilangan bulat), dan max yang merupakan konstanta bernilai tetap 100.

var r, luas : real;

const max = 100;

Dalam Bahasa C++ yang merupakan pengembangan dari Bahasa Pascal, tipe data disebutkan sebelum nama variabelnya. Di bawah ini variabel yang dideklarasikan adalah num dan count yang bertipe data integer, dan num diberi nilai awal 6. Kemudian sum dan product yang bertipe data float (bilangan decimal atau pecahan), terakhir variabel ch yang bertipe data karakter.

int num, count;

num = 6;

float sum, product;

char ch;

Sedangkan bagian statement merupakan bagian program yang berisi rangkaian perintah sesuai notasi algoritmanya. Rangkaian perintah tersebut pada umumnya dituliskan setelah pendeklarasian variabel.

A. Variabel dan Konstanta

Dalam dunia pemrograman, istilah variabel merujuk pada suatu tempat yang digunakan untuk menampung data di dalam memori

yang mempunyai nilai yang dapat berubah – ubah selama proses program. Suatu tempat atau lokasi di dalam memori tersebut menyimpan data yang akan diolah. Bisa diasumsikan bahwa sebuah varibel yang dideklarasikan dalam program, merupakan sebuah wadah yang dipesan oleh user untuk menampung nilai atau data yang sifatnya dapat berubah-ubah atau tidak tetap selama program dijalankan. Singkatnya variabel merupakan objek yang nilainya tidak tetap, dapat berubah sesuai proses yang dikenai terhadapnya. Nama variabel harus didefinisikan tipe datanya dalam bagian deklarasi.

Contoh :

Deklarasi :

int nilai, nomer;

float ipk;

string nama;

char gender;

Sedangkan konstanta adalah objek yang nilainya tetap, tidak berubah selama program dijalankan, notasinya menggunakan ***const***. Contoh :

const phi = 3.14; const kode = “xyz123”; const nilai_max = 100; dan sebagainya.

Penulisan variabel dan konstanta dalam Bahasa C++, juga dalam bahasa pemrograman lainnya, pada umumnya mempunyai aturan yang sama yaitu :

- karakter pertama harus berupa huruf
- karakter kedua dan seterusnya dapat berupa angka atau *underscore* (_)
- tidak mengandung spasi, operator, tanda baca, dan karakter khusus lainnya.

B. Tipe Data

Secara umum, terdapat dua jenis tipe data yaitu tipe data dasar dan tipe data bentukan. Yang termasuk dalam tipe data dasar adalah bilangan bulat (int), bilangan riil atau pecahan (float atau real), logika, karakter, dan string, dapat dilihat pada tabel 4.1. di bawah ini. Sedangkan tipe data bentukan disusun dari beberapa tipe data dasar seperti tipe terstruktur, contoh struct atau record.

Tabel 4.1. Tipe Data Dasar Dalam C++

Name	Size	Range
Char	1 byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	2 bytes	signed: -32768 to 32767 unsigned: 0 to 65535
Int	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
Bool	1 byte	true or false
Float	4 bytes	+/- 3.4e +/- 38 (~7 digits)
Double	8 bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8 bytes	+/- 1.7e +/- 308 (~15 digits
wchar_t	2 or 4 bytes	1 wide character

Tipe Data Dasar

Tiga tipe data dasar sederhana dalam bahasa C++ yang umum dipakai antara lain bilangan bulat, bilangan logika, dan karakter seperti dalam tabel berikut ini :

Tabel 4.2. Tipe Data Sederhana Yang Umum Dipakai

Data Type	Values	Storage (in bytes)
int	-2147483648 to 2147483647	4
bool	true and false	1
char	-128 to 127	1

Berikut macam-macam tipe dasar dan penjelasannya :

- Logika
  Tipe data logika disebut juga dengan Boolean, hanya mengenal dua nilai yaitu True (benar) dan False (salah). True dapat dinyatakan dengan angka 1, dan False dapat dinyatakan dengan angka 0, atau sebaliknya bergantung kesepakatan yang dibuat. Operator logika yang umum digunakan antara lain : ***and, or, not,*** dan ***xor.***

  Misal terdapat dua variabel X dan Y bertipe Boolean, dilakukan operasi logika ***and, or, not,*** dan ***xor*** terhadap kedua variabel tersebut. Maka cara mendeklarasikan kedua variabel tersebut adalah sebagai berikut :

  Deklarasi:
  X , Y : Boolean;

Operator ***and*** akan menghasilkan nilai benar jika variabel X dan Y **keduanya** bernilai benar, operator ***or*** akan menghasilkan nilai benar jika **salah satu** dari kedua variabel tersebut bernilai benar, dan operator ***xor*** akan bernilai benar jika X dan Y saling berlawanan nilai kebenarannya. Sedangkan ***not*** merupakan kebalikan nilai yang dikandung variabel tersebut.

Hasil operasi logika terhadap kedua variabel tersebut digambarkan dalam **tabel kebenaran** sebagai berikut :

Tabel 4.3. Tabel Kebenaran

X	**Y**	**X *and* Y**	**X *or* Y**	**X *xor* Y**	***not***	
True	False	False	True	True	X = True	Not (X) = False
False	True	False	True	True	X = False	Not (X) = True
False	False	False	False	False	Y = True	Not (Y) = False
True	True	True	True	False	Y = False	Not (Y) = True

Contoh operasi logika lainnya adalah jika X bernilai true, Y bernilai false, dan Z bernilai true, maka :

- (X or Y) or Z → hasilnya : (True) or True = True
- (X and Y) and Z → hasilnya : (False) and True = False
- (X or Y) and Z → hasilnya : (True) and True = True
- (X and Y) or Z → hasilnya : (False) or True = True
- X and (Y or Z) → hasilnya : True and (True) = True
- not (X and Z) → hasilnya : not (True) = False
- (Z xor Y) and Y → hasilnya : (True) and False = False

- Bilangan bulat atau integer
  Tipe data ini tidak mengandung nilai pecahan atau desimal, seperti 119, 7, -12, 620011, dan sebagainya. Tipe ini memiliki keterurutan, artinya nilai sebelumnya (predecessor) dan nilai sesudahnya (successor) dapat diketahui. Contoh predecessor dari 10 adalah 9, dan successornya adalah 11. Jika x adalah variabel bertipe integer, maka definisi keterurutan secara formalnya predecessor (a) = x – 1, dan successor (a) = x + 1.

  Deklarasi variabel bertipe data integer adalah sebagai berikut :

  d, e : integer ;
- Bilangan riil atau pecahan
  Tipe data ini mengandung pecahan decimal, seperti 0.25, 5.7, 31.0, 0.0008, 1.70004300E-3, dan lain-lain. Perhatikan

bilangan riil harus mengandung tanda titik (.) sebagai penanda decimal. Bilangan riil juga dapat dituliskan dengan notasi E yang berarti pangkat 10. Contoh 1.70004300E-3 berarti $1.70004300 \times 10^{-3}$.

Deklarasi variabel bertipe data riil (real atau float) adalah sebagai berikut:

g, h : real;

- Karakter

  Semua huruf dalam alfabet, tanda baca '.' ',' '?' dan lainnya, simbol khusus seperti '&' '%' '#' '@' dan lainnya, operator aritmatik ', dan angka yang diapit tanda petik seperti '0' '12'; termasuk dalam tipe data karakter. Begitu juga karakter kosong (null) yaitu karakter yang panjangnya nol.

  Deklarasi variabel bertipe karakter (char) adalah sebagai berikut :

  c, d : char;

- String

  String merupakan sekumpulan karakter dengan panjang tertentu. Pada hakikatnya string bukan tipe data dasar murni, karena disusun dari elemen-elemen bertipe karakter. Tapi tipe ini sering dipakai dalam pemrograman, yang kemudian diperlakukan sebagai tipe data dasar.

  Deklarasi variabel bertipe karakter (char) adalah sebagai berikut :

  nama : string;

Tipe Data Bentukan

Berbeda dengan tipe data dasar yang sudah dijelaskan di atas, tipe data bentukan merupakan tipe data yang dapat dibentuk sendiri atau didefinisikan sendiri sesuai kebutuhan pemrogram (user defined data type). Tipe data bentukan dapat terdiri lebih dari satu tipe data dasar, atau terdiri dari sekumpulan variabel dengan tipe sama atau berbeda yang berupa rekaman (record). Rekaman (record) disusun oleh satu atau lebih field, tiap field menyimpan tipe dasar tertentu. Nama record ditentukan sendiri oleh pemrogram, yang selanjutnya menjadi nama tipe baru.

Tipe bentukan ini juga dikenal dengan tipe terstruktur, sesuai namanya, tipe ini menyimpan lebih dari satu variabel bertipe sama maupun berbeda. Pendeklarasian tipe terstruktur dalam C++ menggunakan kata kunci ***struct.*** Berikut ini bentuk umum deklarasi dan contohnya :

Bentuk umum:

```
struct nama_struktur
{
   tipe_data variabel1;
   tipe_data variabel2;
   ...
};
```

Contoh:

```
struct nilai_mhs
{
    char nim[10];
    float nilai_tugas, nilai_kuis, nilai_uts, nilai_uas;
};
```

Atau nama variabel dipisah dari deklarasi tipe recordnya dimana nama variabel bisa lebih dari satu seperti pada fungsi main, perhatikan bentuk umum dan contohnya berikut ini :

Bentuk umum:

```
typedef  struct
{
    tipefield1 namafield1;
    tipefield2 namafield2;
    … …
    tipefieldn namafieldn;
} Namatipestruct;
```

namatipestruct namavar;

contoh:

```
typedef struct
{    int tanggal, bulan, tahun;
} data_lahir tgl_lhr;

typedef struct
{    char nama[25];
   data_lahir tgl_lhr;
} data_pasien;
data_pasien info_pasien;
```

Untuk mengakses atau memanggil sebuah variabel yang berada di dalam record, menggunakan bentuk umum "**namavar.namafield**" (dipisah dengan atnda titik). Perhatikan contoh dalam program sederhana berikut ini :

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

typedef struct
```

```
{ int tanggal, bulan, tahun;
} data_lahir tgl_lhr;

typedef struct
{ char nama[25];
data_lahir tgl_lhr;
} data_pasien;
data_pasien info_pasien;

int main()
{
strcpy(info_pasien.nama,"Budi");
info_pasien.tgl_lhr.tanggal = 10;
info_pasien.tgl_lhr.bulan = 2;
info_rekan.tgl_lahir.tahun = 2015;
cout << "Nama  Pasien : " <<info_pasien.nama;
cout << "\nTanggal Lahir Pasien :";
cout << "-" << info_pasien.tgl_lhr.bulan;
cout << "-" << info_pasien.tgl_lhr.tahun;
}
```

C. Ekspresi : Operand dan Operator

Perubahan nilai di dalam proses komputer sampai menjadi keluaran dilakukan melalui suatu perhitungan atau komputasi, yang dinyatakan dalam suatu ekspresi. Ekspresi terdiri dari operand dan operator, operand dapat berupa konstanta atau variabel yang dioperasikan dengan operator tertentu. Sedangkan operator berupa simbol atau lainnya yang menyatakan proses apa yang akan dilakukan pada suatu operand.

Contoh dari operator adalah +, -, *, /, ^ (perpangkatan), div (pembagian yang menghasilkan bilangan bulat atau integer), mod (sisa hasil pembagian integer), >, <, not, and, or, xor, dan lainnya.

Terdapat tiga macam ekspresi yaitu numerik, relasional, dan string.

- **Ekspresi numerik :**
  Pada ekspresi numerik, tipe operand dan hasilnya bertipe numerik, dan yang perlu diperhatikan dalam penulisan ekspresi numerik adalah terdapat tingkatan (hirarki) operator. Operator yang mempunyai tingkatan lebih tinggi dikerjakan terlebih dahulu, tetapi dapat berubah karena penggunaan tanda kurung. Berikut tingkatan operator aritmetika dimulai dari yang tertinggi :

1. ^
2. div, mod
3. /, *
4. +,-

Contoh-contoh ekspresi numerik dengan keterangan urutan pengerjaannya, baik **biner** (ekspresi dengan dua operand) maupun **uner** (ekspresi dengan satu operand) :

- a * (b + c) → di dalam tanda kurung dikerjakan dulu
- a ^ b * c – d → ((a ^ b) *c)) - d
- i + j * k – 5 + l*m → I + (j*k) – 5 + (l*m)
- (d + e) div 2
- a mod 2
- -a * (b+c) →**-a** adalah ekspresi uner

- **Ekspresi Relasional**

  Ekspresi relasional kadang disebut juga dengan ekspresi Boolean karena hasil ekspresinya bertipe boolean (true atau false). Ekspresi ini menggunakan operator **<, <=, >,>=,=, ==, !=** (tidak sama dengan), **not, and, or**, dan **xor**.

Contoh ekspresi boolean, dengan deklarasi variabel-variabel bertipe boolean dan integer di bawah ini :

hasil, nilai = Boolean

x , y = integer

misal hasil bernilai false, nilai bernilai true, x bernilai 4, dan y bernilai 6, maka hasil ekspresi boolean berikut ini adalah:

not nilai → false

hasil and nilai →false

nilai or hasil → true

x < 10 → false

nilai or (x=y) → true

- **Ekspresi String**
  Ekspresi string adalah ekspresi menggabungkan dua buah string dengan operator "+" (operator penyambungan atau penggabungan atau *concatenation*).

  Contoh :

  'Jl. Majapahit' + 'Sidoarjo' → Jl. Majapahit Sidoarjo

  'NIM' + ' 20200013' →NIM 20200013

D. Nilai : Input dan Output

Nilai merupakan besaran dari tipe data yang terdefinisi (tipe dasar atau tipe bentukan). Nilai dapat berupa data yang disimpan di dalam sebuah variabel (peubah), konstanta, hasil perhitungan, atau nilai yang dikirim oleh sebuah fungsi. Pada dasarnya algoritma memanipulasi nilai yang tersimpan di dalam sebuah variabel, yaitu mengisinya ke variabel lain, memakainya untuk perhitungan,

membaca nilai dari perangkat / piranti masukan, dan atau menulisnya ke piranti keluaran.

Contoh 1 :

Mengisi nilai ke variabel lain dan memakainya untuk perhitungan : misalkan num1, num2, dan num3 adalah tiga buah variabel bertipe integer, dan pernyataan berikut dieksekusi secara berurutan :

1. num1 = 18;
2. num1 = num1 + 27;
3. num2 = num1;
4. num3 = num2 / 5;
5. num3 = num3 / 4;

Tabel berikut menunjukkan nilai-nilai variabel setelah eksekusi setiap pernyataan. Tanda tanya artinya nilai yang belum diketahui - A? Menunjukkan bahwa nilainya tidak diketahui. Warna oranye di dalam kotak menunjukkan bahwa nilai variabel itu diubah.

Tabel 4.4. Nilai-nilai Variabel : Input dan Output

	Values of the Variables			Explanation
Before Statement 1	? num1	? num2	? num3	
After Statement 1	18 num1	? num2	? num3	
After Statement 2	45 num1	? num2	? num3	**num1 + 27 = 18 + 27 = 45.** This value is assigned to **num1**, which replaces the old value of **num1**.
After Statement 3	45 num1	45 num2	? num3	Copy the value of **num1** into **num2**.
After Statement 4	45 num1	45 num2	9 num3	**num2 / 5 = 45 / 5 = 9.** This value is assigned to **num3**. So **num3 = 9**.
After Statement 5	45 num1	45 num2	2 num3	**num3 / 4 = 9 / 4 = 2.** This value is assigned to **num3**, which replaces the old value of **num3**.

Contoh 2 :

Membaca nilai dari piranti masukan (keyboard) dan menulisnya ke piranti keluaran (monitor) :

read (x); → meminta user untuk memasukkan nilai melalui keyboard

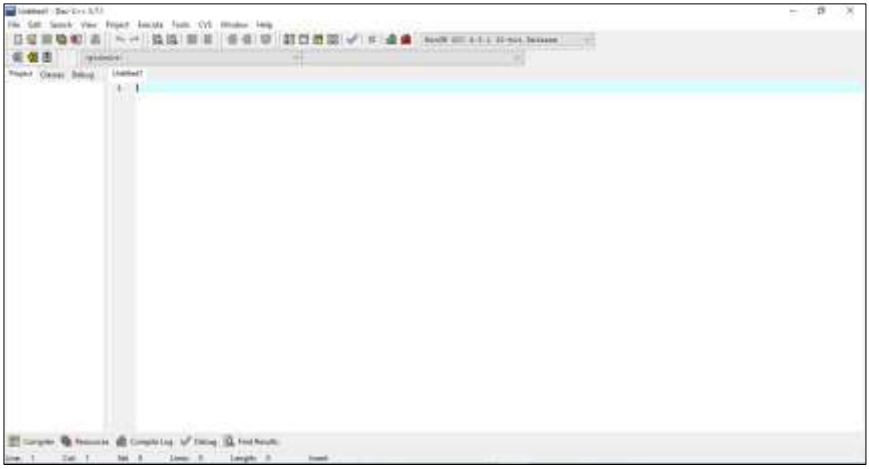
write (x); →mencetak isi atau nilai dari variabel x di atas, jika user memasukkan angka 30 maka akan tercetak angka 30 tersebut

di monitor

E. Memulai Membuat Program

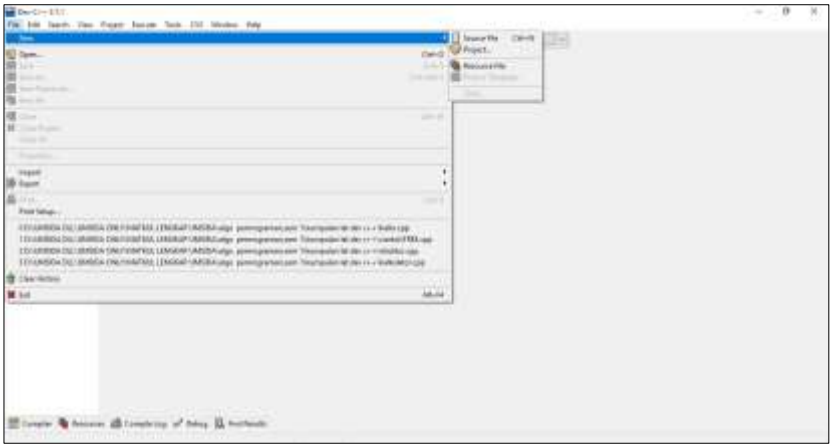
Untuk mengimplementasikan algoritma, struktur program, dan elemen-elemen pemrograman dalam bahasa C++ yang sudah dibahas di atas, diperlukan perangkat lunak compiler yang mendukung implementasi tersebut, antara lain IDE Dev C++ yang selanjutnya disingkat Dev C++ yang dibahas pada buku ini, IDE Visual Studio (Visual C++), dan aplikasi lainnya baik yang gratis maupun yang berbayar.

Aplikasi Dev C++ merupakan aplikasi gratis untuk bahasa pemrograman C dan C++ dibawah lisensi *General Public License* (GNU). Bagi pemula aplikasi ini sangat mudah digunakan, interface yang sederhana memudahkan pengguna untuk membuat, meng-compile dan menjalankan program dalam satu aplikasi sekaligus, tanpa perlu menginstal library atau plug-in tersendiri. Karena aplikasi tersebut sudah dilengkapi dengan *DM-GCC Compiler* yang merupakan bagian dari *GNU Compiler Collection* (GCC). Aplikasi tersebut dapat diunduh secara gratis di https://sourceforge.net/projects/orwelldevcpp/. Berikut tampilan awalnya:



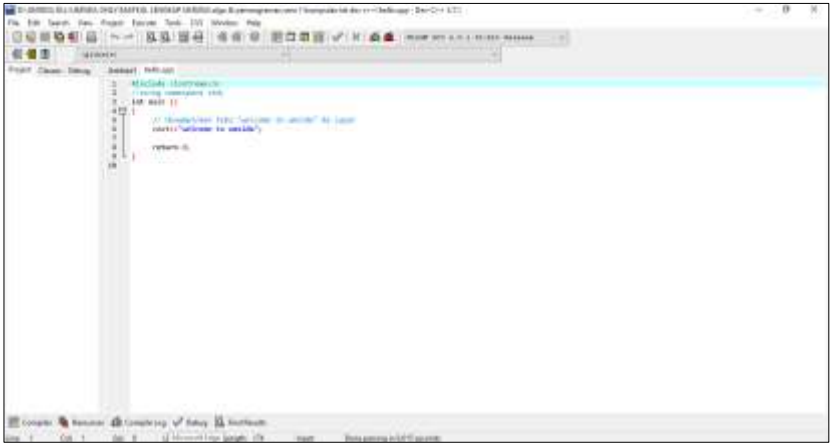
Gambar 4.3. Tampilan awal Dev C++

Selanjutnya pilih menu File – New – Source File, untuk memulai menulis program, seperti tampak dalam tampilan di bawah ini.



Gambar 4.4. Tampilan Menu File – New – Source File

Kemudian bisa langsung menulis perintah-perintah program dalam bahasa C++ pada kolom editor, seperti tampak dalam gambar di bawah ini.



Gambar 4.5. Tampilan Perintah Program

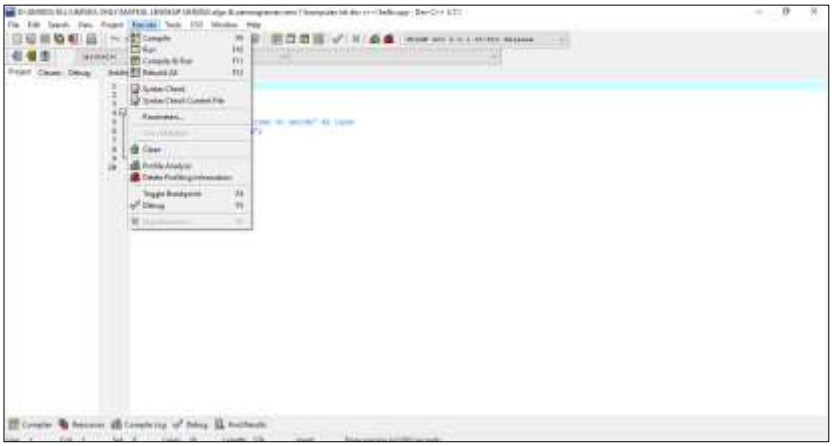
Program pada gambar di atas merupakan program sederhana C++ menampilkan teks "***welcome to umsida***" dengan susunan perintah sebagai berikut :

```
#include <iostream.h>

//using namespace std;

int main ()

{

    // Menampilkan teks "welcome to umsida" ke Layar
```

```
    cout<<"welcome to umsida";
    return 0;
}
```

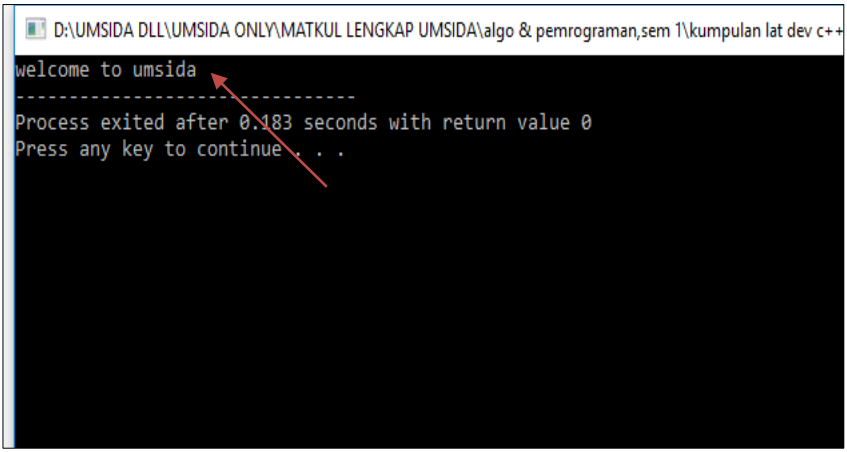
Perhatikan perintah pada tiap barisnya, simak kembali pembahasan perintah program yang telah dijelaskan di atas (bagian awal bab ini).

Selanjutnya simpan dengan nama "**hello**" (ekstensi cpp akan diberikan oleh Dev C++). Selanjutnya program dapat dijalankan dengan memilih menu Compile kemudian menu Run, seperti yang tampak dalam gambar di bawah ini.



Gambar 4.6. Tampilan Menu Compile - Run

Dan berikut tampilan output programnya, perhatikan bagian yang diberi tanda anak panah.



Gambar 4.7. Tampilan Output Program

Latihan soal bab 4

1. Jika nilai a = 3, b=10, c=-4, dan d = 3, tuliskan hasil dari operasi logika berikut ini, True (benar) atau False (salah):
   - (a > b) && (c < d) || (a == b)
   - (a != d) || (c > b) || (a == d)
2. Tuliskan rumus berikut dalam notasi algoritmik (ekspresi numerik):
   a. $V = \frac{4}{3} + 5\ (1 - a^2)$
   b. $X = \frac{-b+2c^2 2+4ab}{2c}$
3. Definisikan sebuah tipe data terstruktur untuk menyatakan data nasabah sebuah bank, yang terdiri dari nomor akun, nama, alamat, kota dan nomor telepon nasabah. Gunakan tipe data yang sesuai untuk setiap field yang ada.
4. Buat program sapaan sederhana yang mengembangkan dari program "hello.cpp" diatas dengan ketentuan sebagai berikut :
   - Menampilan tulisan " Hai, siapa nama anda?" di layar, kemudian meminta user memasukkan namanya.
   - Selanjutnya tampil tulisan "Di kota apa anda tinggal ? " , sama seperti ketentuan sebelumnya yang meminta user memasukkan kotanya.
   - Dan terakhir menuliskan pesan " Selamat datang", "<<nama>>", "<<kota>>"; field nama dan kota bertipe data string yang dibaca berdasarkan yang diinputkan user di atas.

BAB V

TRANSLASI ALGORITMA MENGGUNAKAN BAHASA PEMROGRAMAN C++

Sub – Capaian Pembelajaran Mata Kuliah :

Mahasiswa mampu menerjemahkan semua struktur dasar algoritma yang dibuat ke dalam notasi bahasa C++ menggunakan perintah IF - ELSE, SWITCH - CASE, FOR - DO, dan WHILE - DO sesuai kaidah yang benar.

Pokok bahasan :

1. Program berstruktur runtunan.
2. Program berstruktur pemilihan.
3. Program berstruktur perulangan.

Setelah mempelajari elemen-elemen pemrograman dan perangkat perangkat lunaknya, maka pada bab ini dibahas pembuatan program menggunakan elemen-elemen tersebut, sesuai struktur algoritma yang juga telah dibahas sebelumnya.

A. Program Berstruktur Runtunan

Pada bab 3 di atas telah dijelaskan bahwa runtunan merupakan salah satu struktur algoritma yang paling dasar, yaitu berisi rangkaian instruksi yang diproses secara sekuensial, satu persatu, mulai dari intsruksi pertama sampai instruksi terakhir. Contoh program ber struktur runtunan dapat dilihat pada gambar tembak layar (*screen shoot*) berikut ini.

```
// This program illustrates how data in the variables are
// manipulated.

#include <iostream>
#include <string>

using namespace std;

int main()
{
    int num1, num2;
    double sale;
    char first;
    string str;

    num1 = 4;
    cout << "num1 = " << num1 << endl;

    num2 = 4 * 5 - 11;
    cout << "num2 = " << num2 << endl;

    sale = 0.02 * 1000;
    cout << "sale = " << sale << endl;

    first = 'D';
    cout << "first = " << first << endl;

    str = "It is a sunny day.";
    cout << "str = " << str << endl;

    return 0;
}
```

Sample Run:

```
num1 = 4
num2 = 9
sale = 20
first = D
str = It is a sunny day.
```

Gambar 5.1. Contoh Program Berstruktur Runtunan

Program di atas dimulai dengan identifikasi beberapa variabel yang dibutuhkan, memberi harga awal dan expresi numerik, kemudian mencetak nama variabel beserta isi / nilainya masing-masing. Struktur perintah di dalamnya beruntun, berurutan dari atas ke bawah, dan tidak terdapat proses memilih / bercabang dan proses berulang.

Contoh program berstruktur runtunan berikutnya adalah program untuk menghitung akar persamaan kuadrat dari sebuah bilangan.

```
#include <iostream>
#include <conio.h>
#include <math.h>
void main () {
float a,b,c,x1,x2;
cout<<“masukkan nilai a : “;
cin>>a;
cout<<“masukkan nilai b : “;
cin>>b;
cout<<“masukkan nilai c : “;
cin>>c;
cout<<endl<<endl;
x1=(b+sqrt(b*b-4*a*c))/2*a*c;
x2=(b-sqrt(b*b-4*a*c))/2*a*c;
cout<<“x1=”<<x1<<endl<<endl;
cout<<“x2=”<<x2<<endl<<endl;
getch ();
}
```

Program tersebut dapat dikembangkan menjadi berstruktur pemilihan, menggunakan perintah if – else seperti pembahasan berikut ini.

B. Program Berstruktur Pemilihan

Pada struktur pemilihan atau percabangan, terdapat pemeriksaan kondisi / syarat yang harus dipenuhi, yang kemudian akan memilih perintah apa yang akan dilakukan jika syarat tersebut dipenuhi. Perintah tidak lagi dikerjakan secara beruntun seperti pada struktur runtunan, tetapi berdasarkan syarat yang harus dipenuhi.

Contoh program mencari akar persamaan kuadrat di bawah ini yang dikembangkan dengan memeriksa kondisi yang harus dipenuhi, dan memilih perintah tertentu jika kondisi terpenuhi (if – else).

```
#include <iostream>
#include <math.h>
using namespace std;
int main(){

    int a, b, c, D;
    float x1, x2;

    cout<<"Masukan nilai a : ";
    cin>>a;
    cout<<"Masukan nilai b : ";
    cin>>b;
```

```
        cout<<"Masukan nilai c : ";
        cin>>c;

        D=(b*b)-(4*a*c);

        if (D>0){
                x1 = (-b + sqrt(D)) / (2*a);
                x2 = (-b - sqrt(D)) / (2*a);
        }else if (D==0){
                x1 = (-b + sqrt(D)) / (2*a);
                x2 = x1;
        }

else {
        cout<<"\nAkar Imajiner"<<endl;
                exit(0);
        }
        cout<<"X1 = "<<x1<<endl;
        cout<<"X2 = "<<x2<<endl;
}
```

Program di atas menggunakan perintah if untuk menerjemahkan struktur pemilihan masing-masing nilai yang dihasilkan (variabel d). Terdapat method sqrt () yang diambil dari modul math. Method sqrt() berfungsi untuk menghitung akar kuadrat dari suatu bilangan. Selain itu terdapat juga fungsi exit() untuk mengentikan program agar tidak mengeksekusi pernyataan setelahnya.

Modul lainnya yang disertakan adalah iostram untuk menangani input/output program, terdapat 4 variabel dengan perincian sebagai berikut :

- Empat variabel bertipe integer yaitu : a, b, c, d
- Dua variabel bertipe float yaitu : x1, x2

Nilai a, b dan c dimasukkan dari keyboard saat program di jalankan. Program akan menghitung nilai diskriminan sesuai rumusnya D=(b*b)-(4a*c). Untuk menentukan akar-akar persamaan kuadrat dari nilai diskriminan.

Contoh dimasukkan nilai a=1, b=2 dan c= -3, maka akan program akan menghitung akar-akar persamaan x2+2x-3=0. Berikut tampilan outputnya :

```
Masukan nilai a : 1
Masukan nilai b : 2
Masukan nilai c : -3
X1 = 1
X2 = -3

--------------------------------
```

Gambar 5.2. Output Program Mencari Akar Persamaan Kuadrat

Contoh program dengan struktur pemilihan yang memadukan perintah if – else dengan switch – case :

```
#include <iostream>
using namespace std;
int main() {
        int jenis, lama, harga, tambahan, jam_berikutnya;

        cout<<"Menghitung Tarif Parkir Kendaraan"<<endl;
        cout<<"1. Mobil"<<endl;
        cout<<"2. Motor"<<endl;
        cout<<endl;
        cout<<"Masukan Jenis Kendaraan      : ";
        cin>>jenis;

        switch (jenis){
                case 1  :
                        harga=3000;
                        tambahan=1000;
```

```
                break;
        case 2  :
                harga=2000;
                tambahan=500;
                break;
        default :
                harga=0;
}

cout<<"Masukkan Lama Parkir (jam) : ";
cin>>lama;

if (lama>2){
        jam_berikutnya=((lama-2)*tambahan);
}else {
        jam_berikutnya=0;
}

cout<<"-------------------------------------"<<endl;
cout<<"Dua Jam Pertama  Rp:"<<harga<<endl;
cout<<"Jam Berikutnya Rp:"<<jam_berikutnya<<endl;
```

```
    cout<<"Total Bayar    Rp:"<<harga+jam_berikutnya<<endl;
}
```

C. Program Berstruktur Perulangan

Perulangan merupakan struktur dimana terdapat proses pengulangan perintah yang sama sebanyak n kali. Struktur ini merupakan salah satu kelebihan yang dimiliki oleh mesin komputer. Sebagai contoh untuk menampilan teks “Belajar Pemrograman” sebanyak 10 kali pada layar monitor, hanya diperlukan beberapa baris perintah meggunakan teknik atau struktur perulangan tersebut. Tanpa harus menuliskan perintah yang sama sebanyak 10 kali.

Contoh program di bawah ini adalah untuk menampilkan deret bilangan 0 1 2 3 4 5 menggunakan perintah for – do. Dimana perintah akan berulang selama nilai yang tersimpan dalam variabel (bil) berada dalam rentang 0 sampai dengan 5. Untuk lebih memahami materi, simak kembali pembahasan perintah for – do tersebut pada bab sebelumnya.

```
#include <iostream>
using namespace std;
main() {
 int bil;
 for(bil=0 ; bil<=5 ; bil++){
  cout << bil << " ";   }
}
```

Program selanjutnya merupakan pengembangan dari program di atas, yaitu dengan memasukkan jumlah deret yang ingin ditampilkan. Terdapat dua variabel yang digunakan yaitu bil dan deret. Salinlah program di bawah ini pada aplikasi Dev C++ dan perhatikan outputnya.

```
#include <iostream>
using namespace std;
main() {
 int bil, deret;
 cout << "Masukkan jumlah deret angka : ";
 cin >>deret;
 for(bil=0 ; bil<=deret ; bil++){
  cout << bil << " "; }
 }
```

Untuk menampilkan deret bilangan bulat tertentu, misal deret bilangan genap diantara 0 sampai dengan 100, maka diperlukan rumus deret yang ditulis dalam ekspresi numerik atau notasi algoritmik tertentu. Pada program di bawah ini, terdapat ekspresi numerik yang menggunakan operator modulus untuk mencari sisa pembagian (simbol %). Ekspresi numerik "***bil % 2 == 0***" menyatakan bilangan yang habis dibagi 2 merupakan bilangan genap.

```
#include <iostream>
using namespace std;
main()
 {
    int bil;
   for(bil=0 ; bil<=100 ; bil++)
  {
    bil % 2 == 0; bil++;
   cout << bil << " ";
    }
}
```

Contoh program selanjutnya berstruktur kombinasi pemilihan dan perulangan, yang merupakan pengembangan dari program deret bilangan genap di atas.

```
#include <iostream>
using namespace std;
int main() {
        int pil, batas;
```

```
cout<<"Masukan Pilihan [1.Ganjil] [2.Genap] : ";
cin>>pil;
cout<<"Masukan Batas : ";
cin>>batas;
cout<<endl;

if (pil==1)
        {
           cout<<"Deret Bilangan Ganjil"<<endl;
           for(int i=1;i<=batas;i++)
        {
           if (i % 2 != 0)
           cout<<i<<" ";
          }
       }
else {
           cout<<"Deret Bilangan Genap"<<endl;
           for(int i=1;i<=batas;i++){
           if (i % 2 == 0)
           cout<<i<<" ";
    }
```

```
    }
  }
```

Pada program di atas terdapat pilihan di dalamnya, jika pilihan = 1 maka akan ditampilkan deret bilangan ganjil, dan jika pilihan =2 maka akan ditampilkan deret bilangan genap. Dan terdapat dua variabel dengan “pil” dan “batas”, yang berfungsi untuk menyimpan nilai pilihan dan batas deret bilangan yang ingin ditampilkan.

Ekspresi numerik “***bil % 2 == 0***” menyatakan bilangan yang habis dibagi 2 alias bilangan genap. Sebaliknya untuk menyatakan bilangan yang tidak habis dibagi 2 alias bilangan ganjil, maka ekspresi numerik menjadi “***bil % 2 != 0***” (menggunakan operator relasional ***!=*** yang berarti “**tidak sama dengan**”).

Selain untuk menampilkan deret bilangan yang berurutan secara menaik (ascending) di atas, juga dapat ditampilkan secara menurun (descending), misal deret bilangan bulat 10 9 8 7 6 5 4 3 2 1, atau deret bilangan ganjil 9 7 5 3 1.

Berikut program untuk menampilkan deret bilangan menurun 10 sampai dengan 1 :

```
#include <iostream>
using namespace std;
main() {
 int bil;
```

```
 for(bil=10 ; bil<=1 ; bil--)
{
  cout << bil << " ";   }
}
```

Perhatikan perintah for yang tercetak tebal ***(bil=10; bil<=1; bil--)*** pada program di atas, yang berarti perintah perulangan dimulai dari 10 (bil=10) sebagai nilai awal, dilakukan berulang secara menurun satu persatu (bil--), dan berakhir saat variable bil <=1 (nilai akhir). Sehingga jika program tersebut dijalankan dapat ditampilkan deret bilangan bulat menurun 10 9 8 7 6 5 4 3 2 1.

Selain perintah for – do, untuk membuat program berstruktur perulangan juga dapat menggunakan perintah while dan do – while. Berikut contoh program berstruktur perulangan menggunakan while dan do-while.

```
#include <iostream>

using namespace std;

int main() {

        int a = 1;

        while(a <= 10) {

                cout<<a<<endl;

                a++;     }

      }
```

Program while di atas untuk menampilkan deret bilangan bulat dari 1 sampai dengan 10. Berbeda dengan perintah for-do dimana nilai awal dan nilai akhir ditulis dalam satu kesatuan perintah, maka pada perintah while diberi nilai awal saja (int a =1). Setelah itu terdapat perintah untuk mengecek kondisi yang ditentukan, dan perintah berikutnya akan dikerjakan selama kondisi tersebut bernilai benar (***while(a<=10)***).

Sedangkan pada program do-while, semua perintah dikerjakan berulang terlebih dahulu, dan dihentikan jika kondisi yang ditentukan sudah terpenuhi atau bernilai benar. Seperti yang tampak dalam pembahasan program di bawah ini.

Perhatikan outputnya pada gambar di bawah ini, mengapa hasilnya demikian ? Mari kita diskusikan.

```
#include <iostream>
using namespace std;
int main() {
    int a = 5;
        do {
                cout<<a<<endl;
        a++;

    } while(a <= 4);
```

```
    cout<<"Nilai terakhir a = "<<a<<endl;
}
```

Output :

nilai terakhir a = 6

Gambar 5.3. Output Program Do-While 1

Jika beberapa perintah pada program di atas diganti menjadi sebagai berikut, maka tampak perbedaan outputnya seperti yang tampak dalam gambar di bawah ini.

```
#include <iostream>
using namespace std;
int main() {
    int a = 5;
        do { cout<<a<<endl;
```

a--;

*} while(**a >= 4**);*

cout<<"Nilai terakhir a = "<<a<<endl; }

Output:

5

4

Nilai terakhir a = 3

Perhatikan perintah yang dicetak tebal, counter / pencacahnya diganti menjadi ***a--***(decrement / menurun), dimana pada program sebelumnya pencacahnya menaik ***a++*** (increment), dan kondisi yang ditentukan menjadi (a>=4). Maka outputnya menjadi seperti dalam gambar di bawah ini. Mengapa demikian? Mari kita didiskusikan juga.

Gambar 5.4. Output Program Do-While 2

1. Buat program input output sederhana, untuk menghitung dan menampilkan luas segitiga, di mana nilai alas dan tinggi dimasukkan oleh pengguna dari keyboard.
2. Buat program untuk menampilkan keterangan "bilangan positif" atau "bilangan negatif" berdasarkan bilangan tertentu yang dimasukkan oleh pengguna.
3. Buat program untuk menampilkan deret bilangan bulat dengan urutan menurun dari 100 sampai dengan 1, menggunakan perintah do-while.
4. Buat flowchart / pseudo code dan program untuk menghitung jumlah karakter yang dibaca secara berulang-ulang dari keyboard. Pembacaan berakhir jika karakter yang dibaca adalah karakter TITIK (tetapi titik tidak termasuk dalam hitungan jumlah karakter). Contoh jika karakter yang dibaca berturut-turut adalah 'a', 'm', 't', '.' maka jumlahnya = 3 (tidak termasuk tanda/karakter titik). Dibutuhkan variabel untuk menyimpan karakter yang dibaca dan pencacah (counter) dengan harga awal 0 tentunya, dan gunakan perintah while karena karakter dibaca di awal, dan jika ketemu titik maka selesai.

   Ilustrasi output :

   Inputkan karakter-karakternya : amit.

   Jumlah karakter = 4

5. Buat program untuk menuliskan teks lagu Anak Ayam Turun N dimana n adalah jumlah anak ayam semula (nilai n positif dan dibaca terlebih dahulu). Pada walnya nilai i (counter) = n, setiap kali pengulangan nilai i selalu dikurangi 1. Gunakan perintah for menurun, dan ketika i = 1 pencetakan string ditangani secra khusus karena lirik pada baris terakhir berbeda dengan baris sebelumnya (ada tambahan ".. tinggal induknya").

Contoh n = 3, maka lirik (tipe string) akan tercetak sbb :

Anak ayam turun 3

Anak ayam turun 3, mati satu tinggal 2

Anak ayam turun 2, mati satu tinggal 1

Anak ayam turun 1, mati satu tinggal induknya

BAB VI

PEMROGRAMAN MODULAR

Sub Capaian Pembelajaran Mata Kuliah :

3. Mahasiswa mampu menjelaskan prosedur, fungsi, variable global-lokal dan lingkupnya.
4. Mahasiswa mampu menerjemahkan prosedur dan fungsi termasuk algoritma rekursif ke dalam bahasa C++.

Pokok bahasan :

1. Konsep pemrograman modular.
2. Fungsi dan prosedur.
3. Prosedur rekursif

A. KONSEP PEMROGRAMAN MODULAR

Pemrograman modular adalah sebuah metode pembuatan program dengan cara memecah masalah menjadi beberapa kelompok masalah yang lebih kecil. Dengan membagi masalah menjadi beberapa modul maka masalah tersebut akan menjadi lebih sederhana sehingga program dapat menjadi lebih mudah disusun dan dipahami. Untuk menyusun program modular dapat menggunakan konsep fungsi, prosedur ataupun *subroutine.*

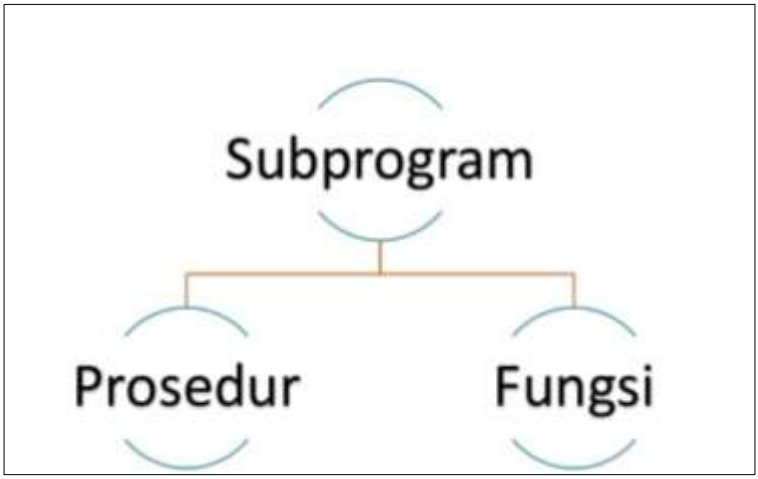
Keuntungan pemrograman modular :

1. Program yang kompleks bisa dibuat lebih ringkas
2. Mudah dibaca dan dimengerti
3. Mudah didokumentasi
4. Mengurangi dan mudah mencari kesalahan program, karena kesalahan yang terjadi bersifat "lokal".
5. Membantu memahami algoritma yang dibuat dan mengembangkannya.
6. Memudahkan penulisan bagian program yang sama, modularisasi menghindari penulisan bagian program yang berulang. Sub program dapat dipakai berulang kali dengan hanya memanggilnya tanpa menuliskan banyak yang diulang-ulang.

Secara umum terdapat dua bentuk modul program yaitu procedure/prosedur dan function/fungsi, berikut perbedaan keduanya :

1. Procedure bisa mengembalikan atau tidak mengembalikan nilai/hasil, sedangkan function wajib mengembalikan nilai keluaran.

2. Procedure membutuhkan suatu variabel khusus untuk menampung hasil/nilai ketika terjadi suatu proses perhitungan, sedangkan function tidak membutuhkan karena berlaku ketentuan bahwa nama fungsi = nama/variabel proses.
3. Pada procedure pencetakan hasil/nilai berada dalam blok subrutinnya sendiri yang selanjutnya dipanggil nama procedurenya di dalam program utama, sedangkan pada function proses pencetakan hasil/nilai dibuat di program utama ketika pemanggilan function-nya.



Gambar 6.1. Jenis Sub Program - Prosedur dan Fungsi

(Sumber : e-materiku)

Modul pada bahasa C++ dikenal dengan nama fungsi (function). Bahasa C terdiri dari fungsi-fungsi, baik yang langsung dideklarasikan dalam program ataupun dipisah di dalam header file. Fungsi yang selalu ada pada program C++ adalah fungsi main().

Selain bersifat modular, fungsi merupakan salah satu dasar penyusunan blok pada C++. Sebuah program C++ minimal mengandung sebuah fungsi, yaitu fungsi main(). Fungsi ini menjadi awal dan akhir eksekusi program C++. Badan fungsi dimulai dari tanda kurawal pembuka { hingga tanda kurawal penutup }, semua yang terletak didalam tanda { } tersebut disebut blok. Tanda () digunakan untuk mengapit argumen fungsi, yaitu nilai yang akan dilewatkan / dimasukkan ke dalam fungsi.

B. PROSEDUR

Bentuk Umum Prosedur :

```
void nama_prosedur (daftar_-parameter)
{
  /*kumpulan kode program di dalam prosedur*/
}
```

Contoh prosedur lengkap dengan program utamanya :

```
#include <iostream>
using namespace std;
// Deklarasi dua prosedur, int a sebagai parameter formal
```

```
void ContohProsedur(int a);

void ContohProsedur1(int a);


// Fungsi Utama atau main program

int main(){

    int panjang = 5;



//memanggil dua prosedur di atas, panjang sebagai parameter aktual

    ContohProsedur(panjang);

    ContohProsedur1(panjang);

    return 0;

}


//prosedur untuk menghitung dan mencetak luas persegi panjang

//int panjang sebagai parameter input, dan var panjang sudah diberi harga awal = 5 pada main program di atas
```

```
void ContohProsedur(int panjang)

{

    int lebar, luas;

    cout<<"\n\nMasukkan Lebar Persegi Panjang : ";cin>>lebar;

    luas=panjang*lebar;

    cout<<"Luas Persegi Panjang = "<<luas<<endl;

}


//prosedur untuk menghitung dan mencetak keliling persegi panjang

//int panjang sebagai parameter input, dan var panjang sudah diberi harga awal = 5 pada main program di atas


void ContohProsedur1(int panjang){

    int lebar, keliling;

    cout<<"\n\nMasukkan Lebar Persegi Panjang = ";cin>>lebar;

    keliling=(panjang+lebar)*2;

    cout<<"Keliling Persegi Panjang = "<<keliling<<endl;

}
```

Contoh output prosedur di atas :

(ingat, nilai panjang sudah ditentukan di dalam program utama = 5)

Masukkan Lebar Persegi Panjang = 4

Luas Persegi Panjang = 20

Masukkan Lebar Persegi Panjang = 6

Keliling Persegi Panjang = 22

C. FUNGSI

Bentuk Umum Fungsi :

Bentuk umum Fungsi adalah sebagai berikut:

```
TipeData NamaFungsi (DaftarParameter){
    /*kode program di dalam fungsi*/
    return nilaireturn;
}
```

Contoh fungsi lengkap dengan program utamanya:

```
#include <iostream>
using namespace std;

// Deklarasi Fungsi, int a sebagai parameter formal
int ContohFungsi(int a);

// Fungsi Utama
int main(){
    int luas1, luas2, totalluas;
    int panjang = 5;
```

```
    // memanggil fungsi ContohFungsi, panjang sebagai
parameter aktual

    luas1 = ContohFungsi(panjang);

    luas2 = ContohFungsi(panjang);

    total_luas = luas1 + luas2;


    cout<<"\n\nLuas Gabungan Kedua Persegi Panjang
adalah = "<<total_luas<<endl;

    return 0;

}
```

```
// Contoh Fungsi, int panjang sebagai parameter input

int ContohFungsi(int panjang){

    int lebar, luas;

    cout<<"\n\nMasukkan Lebar Persegi Panjang =
";cin>>lebar;

    luas=panjang*lebar;

    cout<<"Luas Persegi Panjang adalah "<<panjang<<" x
"<<lebar<<" = "<<luas;

    return luas;

}
```

Contoh output fungsi di atas :

(ingat, nilai panjang sudah ditentukan di dalam program utama = 5)

```
Masukkan Lebar Persegi Panjang = 2

Luas Persegi Panjang adalah 5 x 2 = 10


Masukkan Lebar Persegi Panjang = 5

Luas Persegi Panjang adalah 5 x 5 = 25


Luas Gabungan Kedua Persegi Panjang adalah = 35
```

D. REKURSIF

Rekursif adalah sub program yang memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Rekursif umumnya dipakai untuk permasalahan yang memiliki langkah penyelesaian yang terpola atau langkah-langkah yang teratur. Bila ada suatu permasalahan dan sudah diketahui algoritma penyelesaiannya, maka sub program rekursif dapat menjadi pilihan untuk digunakan.

Fungsi-fungsi yang dapat diubah ke bentuk rekursif antara lain perhitungan faktorial bilangan bulat positif n sebagai berikut :

Jika n > 1 → n! = n (n-1)!

Jika n=0 atau 1 → n! = 1

Kode programnya :

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

Penerapan lainnya pada perhitungan Fibonacci.Deret bilangan fibonacci adalah serangkaian deret angka yang susunannya merupakan penjumlahan dari dua angka sebelumnya.

Contoh deret Fibonacci :

0 1 1 2 3 5 8 13 21

Rumus deret Fibonacci dapat ditulis Fibo(n) = Fibo (n-1) + Fibo (n-2), artinya suku / deret ke-n perupakan penjumlahan dari dua suku sebelumnya.

Deret ke 0= Fibo(0) = 0

Deret ke 1 = Fibo(1) = 1

Deret ke 2 = Fibo(2) = 1

Deret ke 3 = Fibo(3) = 2

Deret ke 4 = Fibo(4) = Fibo(3) + Fibo(2) = 2 + 1 = 3, dan seterusnya

Berikut kode program untuk menampilkan deret bilangan Fibonacci menggunakan dua pendekatan yaitu non-rekursif dan rekursif. Bandingkan perbedaan di dalam kedua programnya.

```
//pendekatan non-rekursif
#include <iostream>
using namespace std;

int main()
{
   int n, f1 = 0, f2 = 1, lanjut = 0;

   cout << "Masukan jumlah deret Fibonacci :  ";
   cin >> n;
         cout<<endl;
   cout << "Deret Fibonacci: ";

   for (int i = 1; i <= n; ++i)
   {
      // mencetak dua deret Fibonacci pertama
      if(i == 1)
      {
         cout << " " << f1<<" ";
         continue;
      }
```

```
        if(i == 2)
        {
            cout << f2 << " ";
            continue;
        }
        lanjut = f1 + f2;
        f1 = f2;
        f2 = lanjut;
         // mencetak deret Fibonacci selanjutnya
        cout << lanjut << " ";
    }
    return 0;
}
```

```
//pendekatan rekursif
#include <iostream>
using namespace std;

//fungsi rekursif Fibo
int fibo(int m) {
   if (m == 0 || m ==1){
      return m;
    } else {
      return (fibo(m-1) + fibo(m-2));
    }
   }

int main() {

   int n, m= 0;
    cout << "masukan jumlah deret Fibonacci :  ";
   cin >> n;
   cout << "Deret Fibonacci: ";
   for (int i = 1; i <= n; i++){
      cout << fibo(m) <<" ";
```

```
        m++;
    }
            return 0;
}
```

Perhatikan fungsi rekursif yang diberi nama Fibo pada program di atas:

```
int fibo(int m) {
   if (m == 0 || m ==1){
      return m;
    } else {
      return (fibo(m-1) + fibo(m-2));
    }   }
```

Fungsi rekursif dengan nama Fibo tersebut membawa sebuah parameter yaitu variabel m. Di dalam fungsi tersebut terdapat percabangan if, jika nilai m = 0 atau 1 maka nilai yang kembali (return value) pada fungsi tersebut = nilai itu sendiri yaitu 0 dan 1.

Rumus deret bilangan Fibonacci merupakan penjumlahan dari dua bilangan sebelumnya, maka perlu memperoleh dua nilai awal yaitu 0 dan 1 agar dapat dijumlahkan, dan menjadi nilai pada deret selanjutnya. Apabila dijalankan akan menghasilkan output yang sama dengan program pertama (non-rekursif).

Penerapan fungsi rekursif lainnya dapat dilihat pada program di bawah ini, yaitu fungsi mencetak ke layar. Fungsi ini mencetak nilai dari paremeter yang dilempar kepadanya, jika nilai > 0 maka fungsi akan mencetak nilai dari parameter tersebut kemudian memanggil dirinya lagi, jika nilai <= 0 maka program berhenti.

```
void cetak(int n)
  {
    if (n>0)
    {
      printf ("\n cetak: %i",n);
     cetak(n-1);
    }
}
```

Latihan soal bab 6 :

1. Buat program modular menggunakan rekursif untuk perhitungan nilai pangkat. Di mana terdapat dua argumen yang diinputkan bertipe data sederhana, misal integer. Dua argumen yang dimaksud adalah nilai yang dipangkatkan dan pemangkatnya.
2. Sebuah fungsi juga dapat menggunakan tipe data gabungan seperti struktur (struct), string atau tipe data gabungan lainnya sebagai masukannya (argumen yang diinputkan).
   Coba cari dua contoh program yang menggunakan fungsi dan tipe data gabungan.
3. Berikan pendapat anda tentang kelebihan dan kekurangan dari dua program rekursif untuk menampilkan deret bilangan Fibonacci di atas (non rekursif vs rekursif).
4. Buatlah fungsi upper dan fungsi lower untuk mengubah besar/kecilnya huruf yang diinputkan sesuai fungsinya masing-masing.

BAB VII

ARRAY

Sub Capaian Pembelajaran Mata Kuliah :

3. Mahasiswa mampu menjelaskan array atau larik , array 1 dan 2 dimensi, dan array bertipe terstruktur (record).
4. Mahasiswa mampu memproses semua bentuk array menggunakan bahasa C++ sesuai kaidah yang benar.

Pokok Bahasan:

1. Konsep Array
2. Array berdimensi
3. Implementasi Array

Array atau larik adalah sebuah tipe data bentukan atau terstruktur yang terdiri dari sejumlah komponen dengan tipe yang sama. Dengan array dapat menyimpan banyak data dengan satu nama variabel array. Jumlah komponen ditunjukkan dengan nilai indeks atau dimensi dari array.

D. Deklarasi dan Pemrosesan Array

Array dapat berupa array berdimensi satu, dua, tiga atau lebih.

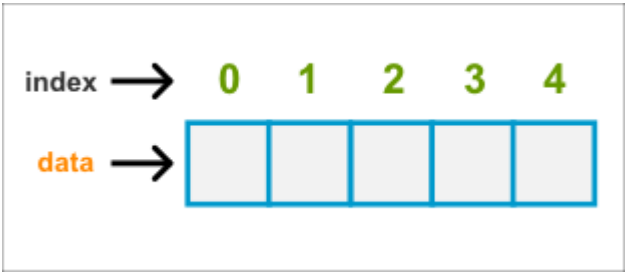
- Array berdimensi satu (one-dimensional array) mewakili bentuk suatu vektor.
- Array berdimensi dua (two-dimensional array) mewakili bentuk dari suatu matriks atau tabel
- Array berdimensi tiga (three-dimensional array) mewakili bentuk suatu ruang.

Contoh deklarasi array berdimensi satu :

var m : array[1..5] of integer;

keterangan

- M adalah nama array satu dimensi
- [1..5] adalah indeks array; yang dapat menampung maksimal 5 (lima) buah data
- indeks dimulai dari 0
- integer adalah tipe data array M



Gambar 7.1. Ilustrasi Array

Dari deklarasi tersebut di atas dapat digambarkan sebuah array dengan nama M mempunyai lima elemen atau dapat menyimpan lima buah data bertipe sama yaitu integer : ***m[1], m[2], m[3], m[4], m[5].***

Dan berikut psudo-code untuk menginputkan data array: ***read(m[1]); read(m[2]); read(m[3]); read(m[4]); read(m[5])***

Sebenarnya kapan array perlu digunakan? Jawabannya, jika kita memerlukan penyimpanan sementara untuk data-data yang bertipe sama di dalam memori, untuk selanjutnya data-data tersebut dimanipulasi (dihitung atau diterapkan oleh proses lainnya)

Contoh algoritma tanpa array:

Deklarasi

x,i : integer;

{baca 6 buah nilai yang bertipe integer dan simpan di x}

for i ← 1 to 6 do

read(x)

endfor

{cetak setiap nilai x}

for i ← 1 to 6 do

write(x)

endfor

Bila runtunan nilai x yang dibaca dari keyboard adalah : 20, 30, 40, 50, 60, 70 maka output dari algoritma di atas adalah: 70 70 70 70 70 70

Karena variabel x hanya dapat menampung satu buah nilai, dan nilai yang disimpan oleh x adalah selalu nilai yang terakhir yaitu 70, maka nilai 70 itulah yg akan dicetak pada setiap kali pengulangan.

Sekarang bandingkan dengan algoritma yang menggunakan array x yang berisi 6 buah data / elemen.

Algoritma dengan array:

Deklarasi

x : array [1..6] of integer

i : integer

{baca 6 buah nilai integer simpan di x}

for i ← 1 to 6 do

read(x[i])

endfor

{cetak setiap nilai x}

for I ← 1 to 6 do

write(x[i])

endfor

Keluaran dari algoritma di atas akan sesuai dengan data yang diinputkan yaitu: 20 30 40 50 60 70.

Selama pelaksanaan program, elemen array tetap menyimpan nilai-nilai yang dimaksud. Hal ini berguna jika kita ingin menggunakan nilai-nilai di dalam array tersebut untuk diproses lebih lanjut di bagian lain dalam algoritma berikut ini:

Contoh algoritma menggunakan array :

Deklarasi

```
x : array [1..6] of integer
i,jumlah : integer

 {baca 6 buah nilai integer simpan di x}
 for i ← 1 to 6 do
        read(x[i])
 endfor

 {cetak setiap nilai x}
 for I ← 1 to 6 do
 write(x[i])
 endfor
```

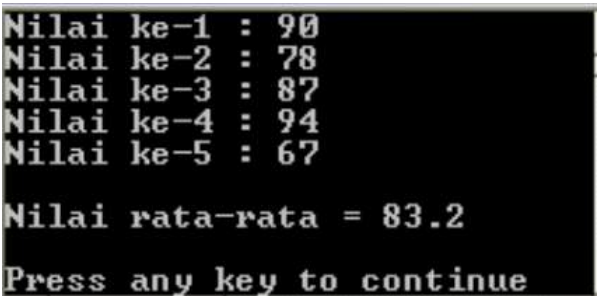
{pada bagian ini elemen array digunakan lagi, untuk menghitung nilai rata-rata seluruh elemen}

```
Jumlah ← 0
For I ← 1 to 6 do
Jumlah ← jumlah + x[i]
Endfor
Write(jumlah/6)
```

Contoh program lengkap untuk mengakses tiap elemen array:

```
#include <stdio.h>
#define MAKS 5
main()
{
int i;
float total=0, rata, nilai[MAKS];
for(i=0; i<MAKS; i++)
{
printf("Nilai ke-%d : ", i+1);
scanf("%f", &nilai[i]);
total = total + nilai[i]; //hitung jml total nilai
}
rata = total / MAKS; //hitung nilai rata2
//cetak nilai rata-rata
printf("\nNilai rata-rata = %g\n", rata);
}
```

Output program :



Gambar 7.2. Output Program Array

Sebuah array juga dapat diinisialisasi sekaligus pada saat dideklarasikan. Untuk mendeklarasikan array, nilai-nilai yang diinisialisasikan dituliskan di antara kurung kurawal ({}) yang dipisahkan dengan koma. Berikut contoh program inisialisasi array untuk menampilkan jumlah hari dalam setiap bulannya.

```
main()
{
int bln, thn, jhari;
int jum_hari[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30,
 31, 30, 31};
printf("Masukkan bulan (1..12) : ");
scanf("%d", &bln);
```

```
printf("Masukkan tahunnya : ");
scanf("%d", &thn);
    if(bln == 2)
    if(thn % 4 == 0) //thn kabisat
    jhari = 29;
else
    jhari = 28;


else
    jhari = jum_hari[bln-1];
printf("\nJumlah hari dalam bulan %d tahun %d
adalah %d hari\n",bln,thn,jhari);}
```

E. Matriks : Array Berdimensi dua

Array dapat dikategorikan sebagai tipe data terstruktur. Elemen array dapat distrukturkan lagi menjadi matriks. Matriks pada umumnya dikenal dengan array berdimensi dua (array berindeks dua). Jika didalam array 1 dimensi hanya menggunakan sebuah tanda [] (bracket), maka pada array 2 dimensi atau matriks terdapat 2 tanda [] tersebut. Matriks dapat digambarkan seperti tabel di mana terdapat baris dan kolom.

Berikut bentuk umum deklarasi array berdimensi dua (matriks) :

tipe_data nama_array[jumlah elemen baris][jumlah elemen kolom];

Contoh mendeklarasikan sebuah matriks bertipe data integer yang di dalamnya terdapat 3 baris dan 4 kolom.

int A[3][4];

Inisialisasi nilai array 2 dimensi (matriks) tersebut dapat dibuat manual atau diinput oleh pengguna saat program dijalankan. Karena array 2 dimensi mempunyai lebih dari satu bentuk index array, maka dalam inisialisasinya perlu menggunakan tanda {} untuk membentuk baris array, contoh sebagai berikut :

int A[3][4]={{3,4,8,0},{3,9,2,1},{6,3,0,2}};

Ilustrasi inisialisasi matriks di atas dapat digambarkan lewat tabel berikut:

Tabel 7.1. Ilustrasi Matriks Beserta Nilai Tiap Elemennya

A	Kolom Ke 0	Kolom Ke 1	Kolom Ke 2	Kolom Ke 3
Baris Ke 0	A [0] [0]	A [0] [1]	A [0] [2]	A [0] [3]
Baris Ke 1	A [1] [0]	A [1] [1]	A [1] [2]	A [1] [3]
Baris Ke 2	A [2] [0]	A [2] [1]	A [2] [2]	A [2] [3]

A	Kolom Ke 0	Kolom Ke 1	Kolom Ke 2	Kolom Ke 3
Baris Ke 0	3	4	8	0
Baris Ke 1	3	9	2	1
Baris Ke 2	6	3	0	2

F. Record : Array Bertipe Terstruktur

Array dapat menyimpan sejumlah elemen bertipe terstruktur (rekaman / record). Record disusun oleh satu atau lebih field. Tiap field menyimpan data dari tipe dasar tertentu atau tipe bentukan.

Record dalam Bahasa C++ dikenal dengan nama struct (lihat Kembali pembahasan tipe data bentukan pada bab IV di atas).

Contoh deklarasi record atau struct dengan nama “mahasiswa” yang mempunyai field : nama, prodi, ipk dengan tipe data berbeda-beda :

```
struct mahasiswa{
   string nama;
   string prodi;
   float ipk;};
```

Untuk memproses matriks digunakan algoritma berstruktur perulangan bersarang agar dapat membaca tiap nilai yang

terkandung di dalam elemen-elemennya, per baris dan per kolomnya. Berikut algoritmanya dalam notasi *pseudo-code* :

Deklarasi
i : integer; {indeks baris}
j : integer; {indeks kolom}

Algoritma
for i ← 1 to bar do
for j ← 1 o kol do
write(M[i,j]);
endfor
endfor

Translasi algoritma di atas ke dalam program lengkapnya:

```
#include <iostream>
#include <conio.h>
using namespace std;
int main() {

        int M[3][4]={{3,4,8,0},{3,9,2,1},{6,3,0,2}};
                for (int i=0;b<3;i++) {
                for (int j=0;k<4;j++) {
                        cout<<M[i][j]<<" ";
                }
                cout << endl;
                }
           getch();
        }
```

Program untuk memproses record atau struct :

```
#include <iostream>
#include <string>
using namespace std;

 struct mahasiswa{
   string nama;
   string prodi;
   float ipk;
};

int main(){

   mahasiswa mhs;
    mhs.nama="Wurijanto ";
   mhs.prodi=" Informatika";
   mhs.ipk=3.50;
   cout<<"Nama    : "<<mhs.nama<<endl;
   cout<<"Jurusan : "<<mhs.prodi<<endl;
   cout<<"IPK     : "<<mhs.ipk<<endl;
  return 0;
```

```
}
```

Selain matriks dan record, tipe data **string** pada dasarnya merupakan array karakter dengan panjang tertentu. Karena string adalah array, maka elemen-elemennya yang berupa karakter juga dapat diakses melalui indeks.

Contoh : 'prodi' → s[1] = 'p'

s[2] ='r'

s[3] = 'o'

s[4] ='d'

s[5] = 'i'

Contoh penerapan array karakter atau string, dapat dilihat dalam program menghitung panjang sebuah string s berikut ini. Program tersebut menggunakan fungsi yang diberi nama "panjang" :

```
int panjang(char s[])
{ int i;
  i = 0;
  while (s[i] != '\0')
 {
    i = i + 1 ;
 }
  return i
```

```
}
```

Pada program tersebut di atas, selama string s diakhiri dengan karakter null '\0' maka program menemukan akhir string (akhir kata yang diinputkan). Panjang string dihitung dengan membaca elemen-elemen string sampai ditemukan karakter null tersebut.

Bahasa C++ sudah menyediakan fungsi standar untuk mengembalikan (menghitung) panjang sebuah string, yaitu *strlen*. Fungsi ***strlen()*** ini didefinisikan di dalam file header *<string.h>* , contoh : *n = strlen(s);*

Latihan soal bab 7

1. Buat program array untuk mencari nilai terbesar dari sejumlah (n) bilangan yang diinputkan.
2. Buat program array untuk membaca sejumlah (m) data mahasiswa yang diinputkan. Data mahasiswa terdiri dari nim, nama, prodi, dan IPK. Gunakan struct dan perulangan !
3. Buat program untuk menghitung panjang sebuah string / kata yang diinputkan menggunakan fungsi ***strlen().***
4. Buat program untuk memasukkan nilai pada setiap elemen matrik [2,3] dan menampilkan hasilnya.
5. Buat program menghitung perkalian dua matrik, di mana baris dan kolom masing-masing matrik diinputkan oleh pengguna saat program dijalankan.

DAFTAR PUSTAKA

Abdul Kadir, Heriyanto, Algoritma Pemrograman Menggunakan C++ Edisi 1, Yogyakarta : Andi Publisher, 2005

Munir, Rinaldi, Algoritma dan Pemrograman Dalam Bahasa Pascal , C dan C++, Edisi Keenam, Bandung : Informatika, 2016

Modul Laboratorium Algoritma dan Pemrograman, Fakultas Teknik - Prodi Informatika - UMSIDA, 2017

Pengertian Dasar Logika & Algoritma , http://sumberbelajar.seamolec.org

Supardi, Yuniar, Cara Mudah Belajar C & Flowchart Lewat Praktek, Jakarta : Dinastindo, 2001

Sutedjo, Budi, Algoritma & Teknik Pemrograman, Yogyakarta : Andi Publisher, 2000

BIODATA PENULIS

Uce Indahyanti, **M.Kom** lahir di Situbondo – Jawa Timur, pada tanggal 11 Mei 1971. Penulis menamatkan pendidikan S1 di STIKOM Surabaya Jurusan Manajemen Informatika (1990 -1996), dan menamatkan S2 Jurusan Sistem Informasi di Institut Teknologi Sepuluh Nopember – ITS (2010-2012). Saat ini aktif berkarya sebagai dosen tetap Prodi Informatika Fakultas Saintek Universitas Muhammadiyah Sidoarjo.

Yunianita Rahmawati, M.Kom lahir di Magetan – Jawa Timur, pada tanggal 1 Juni 1985. Penulis menamatkan pendidikan S1 di STIKOM Surabaya Jurusan Sistem Informasi (2003-2008), dan menamatkan S2 Jurusan Teknologi Informasi di Sekolah Tinggi Teknik Surabaya - STTS (2011-2017). Saat ini aktif berkarya sebagai dosen tetap Prodi Informatika Fakultas Saintek Universitas Muhammadiyah Sidoarjo.