

DASAR PEMROGRAMAN

Ade Hastuty Hasyim

DASAR PEMROGRAMAN

Ade Hastuty, ST., S.Kom. MT.

Buku ini ku persembahkan kepada :

Suamiku **AZIS MALLOMBASI DJALLE DAENG MATTAWANG**

Anak-anakku

Azizah Nurul Fadiillah Djalle Daeng Cinde

Alif Manrompai Djalle Daeng Mangka

Akbar Manrompai Hasmi Djalle Daeng Raja

I love You ALL



Kata Pengantar

Puji syukur penulis haturkan kepada Allah SWT yang maha mengetahui, yang telah memberi nikmat ilmu pengetahuan, nikmat sehat dan nikmat lain yang tidak dapat disebutkan satu-persatu, sehingga buku Dasar Pemrograman ini dapat diselesaikan. Hal yang baru pada buku ini adalah implementasi Algoritma yang di terjemahkan pada bahasa pemrograman Pascal dan Delphi.

Penulis menganggap di era informasi atau era digital seperti saat ini ranah pemrograman telah bergeser ke pemrograman berorientasi objek. Namun, mahasiswa tetap harus memahami bahasa pemrograman berorientasi proses untuk lebih menguasai algoritma.

Algoritma merupakan konsep dasar suatu program. Dalam pembuatan algoritmanya diperlukan daya nalar yang baik dan logis, karena algoritma merupakan susunan langkah-langkah penyelesaian masalah secara terstruktur. Studi kasus pada masing-masing bab di implementasikan menggunakan bahasa pemrograman Pascal dan Delphi, harapannya mahasiswa dapat memahami struktur dasar bahasa pemrograman dari alur algoritma yang dipelajari. Selanjutnya mahasiswa dapat mendalami dan meminati bahasa pemrograman sesuai dengan bidang pekerjaan yang akan ditekuni nanti ditambah bekal pengetahuan algoritma yang baik.

Akhirnya, penulis mengucapkan terimakasih kepada semua pihak yang telah memberikan konstribusi hingga buku ini selesai di tulis.

Kritik dan saran sangat penulis harapkan agar buku ini menjadi lebih baik lagi di edisi berikutnya. Amin YRA.

Parepare, 10 Oktober 2021

Penulis

Kata Pengantar

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan kekuatan dan hidayahnya atas terselesaiannya penulisan buku yang berjudul “**Dasar Pemrograman**”.

Saat ini ilmu pengetahuan berkembang pesat terutama dalam bidang teknologi dan informasi telah memberikan pengaruh terhadap kehidupan manusia. Pemrograman menjadi penunjang utama berbagai aktifitas baik menggunakan gawai, laptop dan desktop. Program yang baik tentu saja harus mengikuti kaidah-kaidah ilmu pemrograman. Buku ini mengajak pembaca mempelajari metodologi pemecahan masalah kedalam bentuk penulisan algoritma yang tepat.

Mempelajari pemrograman tentunya mencakup pada beberapa aspek penunjangnya diantaranya bahasa pemrograman. Aspek penting dalam bahasa pemrograman ialah aturan tata bahasa, penulisan perintah, prinsip kerja compiler serta kemampuan menganalisa semua perintah dalam bahasa tersebut.

Buku ini disusun secara runut dari pengantar dasar pemrograman, konsep data, teknik penyajian algoritma, konsep dasar pemrograman, struktur dasar algoritma, aturan penulisan teks algoritma, dan contoh penulisan algoritma dengan bahasa program delphi dan python. Sajian bahasan menarik dengan contoh-contoh yang tertuang dalam buku ini diharapkan sebagai bahan latihan untuk lebih memahami proses pemrograman dengan alur yang tepat melalui penerapan algoritma yang benar.

Penulis menganggap materi diatas sangat baik untuk seorang mahasiswa fakultas ilmu komputer, fakultas teknik dan fakultas lain yang tertarik pada pemrograman karena di era informasi ranah pemrograman telah bgeser ke pemrograman berorientasi objek. Tetapi sebagai dasar pengetahuan mahasiswa tetap harus memahami bahasa pemrograman berorientasi proses untuk lebih menguasai algoritma. Selanjutnya mahasiswa dapat mendalami satu atau dua bahasa pemrograman yang diminatinya untuk ditekuni agar menambah pengetahuan algoritma menjadi sangat mahir.

Pada kesempatan ini penulis mengucapkan banyak terimakasih kepada suamiku dan anak-anakku tercinta, keluarga, teman-teman dosen sejawat, serta mahasiswa-mahasiswa yang telah memberikan dukungan, saran, dan kritik untuk menyempurnakan buku ini.

Dan tak lupa penulis mengucapkan terimakasih kepada seluruh pembaca yang tertarik pada pemrograman dan menggunakan buku ini sebagai pengantar dan panduan belajar pemrograman dasar.

Penulis mengharapkan kritik dan saran yang membangun sehingga dapat menulis lebih baik lagi dalam buku selanjutnya. Semoga buku ini berguna dan bermanfaat bagi pembaca demi mencerdaskan dan memajukan bangsa dan negara.

Parepare, 09 September 2023

Ade Hastuty Hasyim

Dasar Pemrograman

Penulis :

Ade Hastuty Hasyim, ST., S.Kom., MT

Editor :

Muhammad Basri, ST., MT

Desain Cover & Layout :

Marlina, S.Kom., MT

Hak cipta dilindungi Undang-Undang
All right reserved

Cetakan 3 Edisi Revisi, Oktober 2023

Diterbitkan oleh :

CV. Bangun Bumitama

Jl. Pajjaiyyang Komp. BTN. Dewi Kumalasari Blok AD4 No.4
Kel. Daya, Kec. Biringkanaya, Makassar - Sulawesi Selatan
e-mail : bumitamapress@yahoo.com
Tel. 081355035326

Perpustakaan Nasional: Katalog Dalam Terbitan (KDT)

DASAR PEMROGRAMAN

Cet. 3 - Makassar: CV. Bangun Bumitama, 2023
vi, 148 hlm; 148mm x 217mm

ISBN 978-602-51846-4-2



9 786025 184642

Daftar Isi



Kata Pengantar	i
Daftar Isi	v
<hr/>	
BAB I Pengantar Dasar Pemrograman	1
1.1. Apa itu Algoritma	1
1.2. Defenisi Algoritma	1
1.3. Beda Algoritma & Program	2
1.4. Algoritma merupakan jantung ilmu informatika	4
1.5. Mekanisme pelaksanaan algoritma oleh pemrosesan	6
1.6. Belajar memprogram dan belajar bahasa pemrograman	7
1.7. Menilai sebuah Algoritma	10
1.8. Penyajian Algoritma	11
1.9. Struktur Dasar Algoritma	12
1.10. Tahapan dalam pemrograman	12
1.11. Soal dan penyelesaian	16
<hr/>	
BAB II Data, Variabel dan Operator	17
Pengertian Tipe Data	17
Pengertian Variabel	20
Pengertian Operator	21
<hr/>	
BAB III Teknik Penyajian Algoritma	25
3.1. Pengantar	25
3.2. Struktur English & Pseudocode	25
3.3. Flowchart	26
Soal dan pembahasan	32
<hr/>	
BAB IV Konsep Dasar Pemrograman	33
4.1. Elemen pemrograman pascal/Delphi	33
4.2. Struktur pemrograman pascal/Delphi	41
<hr/>	
BAB V Struktur Dasar Algoritma	43
5.1. Pengantar	43
5.2. Struktur dasar algoritma	43
5.3. Strategi perancangan puncak-turun	47
Soal dan pembahasan	49

BAB VI	Aturan Penulisan Teks Algoritma	53
6.1.	Pengantar	53
6.2.	Susunan teks algoritma	53
<hr/>		
BAB VII	Algoritma Runtunan	57
Konsep runtunan		57
Soal dan pembahasan		57
<hr/>		
BAB VIII	Algoritma Pemilihan	63
8.1.	Pengantar	63
8.2.	Struktur If-Then dan If-Then-Else	64
8.3	Struktur Case	70
<hr/>		
BAB IX	Algoritma Pengulangan	73
9.1.	Pengantar	73
9.2.	Struktur FOR	74
9.3.	Struktur WHILE	76
9.4.	Struktur REPEAT	77
<hr/>		
BAB X	Prosedur & Fungsi	79
10.1.	Pengantar	79
10.2.	Procedure	81
10.3.	Function	83
10.4.	Rekursi	85
<hr/>		
BAB XI	Array & Record	87
11.1.	Pengantar	87
11.2.	Array	88
11.3.	Record	94
11.4.	Soal dan pembahasan	95
<hr/>		
BAB XII	Pengantar Bahasa Program Phyton	103
<hr/>		
MODUL PRAKTIKUM		115
<hr/>		
DAFTAR PUSTAKA		

BAB I



Pengantar Dasar Pemrograman

1.1. Apa itu Algoritma

Ditinjau dari asal-usul katanya, kata Algoritma sendiri mempunyai sejarah yang aneh. Orang hanya menemukan kata algorism yang berarti proses menghitung dengan angka arab. Anda dikatakan algorist jika Anda menghitung menggunakan angka arab. Para ahli bahasa berusaha menemukan asal kata ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal kata tersebut yang berasal nama penulis buku arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi. Al-Khuwarizmi dibaca orang barat menjadi Algorism.

Al-Khuwarizmi menulis buku yang berjudul ***Kitab Al Jabar Wal-Muqabala*** yang artinya “**Buku pemugaran dan pengurangan**” (**The book of restoration and reduction**). Dari judul buku itu kita juga memperoleh akar kata “**Aljabar**” (Algebra). Perubahan kata dari algorism menjadi algorithm muncul karena kata algorism sering dikelirukan dengan arithmetic, sehingga akhiran -sm berubah menjadi -thm. Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa, maka lambat laun kata algorithm berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya. Dalam bahasa Indonesia, kata algorithm diserap menjadi **algoritma**.

1.2. Defenisi Algoritma

“*Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis*”. Kata logis merupakan kata kunci dalam algoritma. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan bernilai salah atau benar. Dalam beberapa konteks, algoritma adalah spesifikasi

urutan langkah untuk melakukan pekerjaan tertentu. Pertimbangan dalam pemilihan algoritma adalah, pertama, algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik.

Pertimbangan kedua yang harus diperhatikan adalah kita harus mengetahui seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil (hasil yang hanya berupa pendekatan). Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya. Ketiga adalah efisiensi algoritma. Efisiensi algoritma dapat ditinjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar (paling mendekati), tetapi jika kita harus menunggu berjam-jam untuk mendapatkan keluarannya, algoritma tersebut biasanya tidak akan dipakai, setiap orang menginginkan keluaran yang cepat. Begitu juga dengan memori, semakin besar memori yang terpakai maka semakin buruklah algoritma tersebut. Dalam kenyataannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya kita mengharapkan keluaran yang sama. Jika terjadi demikian, carilah algoritma yang paling efisien dan cepat.

1.3. Beda Algoritma dan Program

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman. **Wirth (1997)** menyatakan dalam bukunya bahwa : **Algoritma + Struktur Data = Program**

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa

pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, Demikian juga sebaliknya. Dalam pembuatan algoritma mempunyai banyak keuntungan di antaranya:

- Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.
- Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
- Apapun bahasa pemrogramannya, output yang akan dikeluaran sama karena algoritmanya sama.

Beberapa hal yang perlu diperhatikan dalam membuat algoritma :

- Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
- Tidak ada notasi yang baku dalam penulisan teks algoritma seperti notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.
- Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini dikarenakan teks algoritma tidak sama dengan teks program. Namun, supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.
- Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalannya.
- Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa pemrograman.

- Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasikan ke dalam notasi bahasa pemrograman.

Agar algoritma bisa ditranslasikan kedalam bahasa pemrograman, ada beberapa hal yang harus diperhatikan pada translasi tersebut, yaitu:

- Pendeklarasian variabel Untuk mengetahui dibutuhkannya pendeklarasian variabel dalam penggunaan bahasa pemrograman apabila tidak semua bahas pemrograman membutuhkannya.
- Pemilihan tipe data Apabila bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel maka perlu hal ini dipertimbangkan pada saat pemilihan tipe data.
- Pemakaian instruksi-instruksi Beberapa instruksi mempunyai kegunaan yang sama tetapi masing-masing memiliki kelebihan dan kekurangan yang berbeda.
- Aturan sintaksis Pada saat menuliskan program kita terikat dengan aturan sintaksis dalam bahasa pemrograman yang akan digunakan.
- Tampilan hasil Pada saat membuat algoritma kita tidak memikirkan tampilan hasil yang akan disajikan. Hal-hal teknis ini diperhatikan ketika mengkonversikan-nya menjadi program.
- Cara pengoperasian compiler atau interpreter. Bahasa pemrograman yang digunakan termasuk dalam kelompok compiler atau interpreter.

1.4. Algoritma Merupakan Jantung Ilmu Informatika

Algoritma adalah jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang mengarah ke dalam terminologi algoritma. Namun, jangan beranggapan algoritma selalu identik dengan ilmu komputer saja. Dalam kehidupan sehari-hari pun banyak terdapat proses yang dinyatakan dalam suatu algoritma.

Cara-cara membuat kue atau masakan yang dinyatakan dalam

suatu resep juga dapat disebut sebagai algoritma. Pada setiap resep selalu ada urutan langkah-langkah membuat masakan. Bila langkah-langkahnya tidak logis, tidak dapat dihasilkan masakan yang diinginkan. Ibu-ibu yang mencoba suatu resep masakan akan membaca satu per satu langkah-langkah pembuatannya lalu ia mengerjakan proses sesuai yang ia baca. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses (processor). Pemroses tersebut dapat berupa manusia, komputer, robot atau alat-alat elektronik lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau “mengeksekusi” algoritma yang menjabarkan proses tersebut.

No.	Proses	Algoritma	Contoh Langkah dalam Algoritma
1	Membuat kue	Resep kue	Masukkan telur ke dalam wajan, kocok sampai mengembang
2	Membuat pakaian	Pola pakaian	Gunting kain dari pinggir kiri bawah ke arah kanan sejauh 5 cm
3	Merakit mobil	Panduan merakit	Sambungkan komponen A dengan komponen B
4	Kegiatan sehari-hari	Jadwal harian	Pukul 06.00: mandi pagi, pukul 07.00: berangkat kuliah
5	Mengisi voucer HP	Panduan pengisian	Tekan 888, masukkan nomor voucer

Algoritma adalah deskripsi dari suatu pola tingkah laku yang dinyatakan secara primaifitif yaitu aksi-aksi yang didefinisikan sebelumnya dan diberi nama, dan diasumsikan sebelumnya bahwa aksi-aksi

Tabel 1.1. Contoh-contoh algoritma dalam kehidupan sehari-hari

tersebut dapat kerjakan sehingga dapat menyebabkan kejadian. Melaksanakan algoritma berarti mengerjakan langkah-langkah di dalam algoritma tersebut. Pemroses mengerjakan proses sesuai dengan algoritma yang diberikan kepadanya. Juru masak membuat kue berdasarkan resep yang diberikan kepadanya, pianis memainkan lagu berdasarkan papan not balok. Karena itu suatu algoritma harus dinyatakan dalam bentuk yang dapat dimengerti oleh pemroses. Jadi suatu pemroses harus :

- Mengerti setiap langkah dalam algoritma.
- Mengerjakan operasi yang bersesuaian dengan langkah tersebut.

1.5. Mekanisme Pelaksanaan Algoritma oleh Pemroses

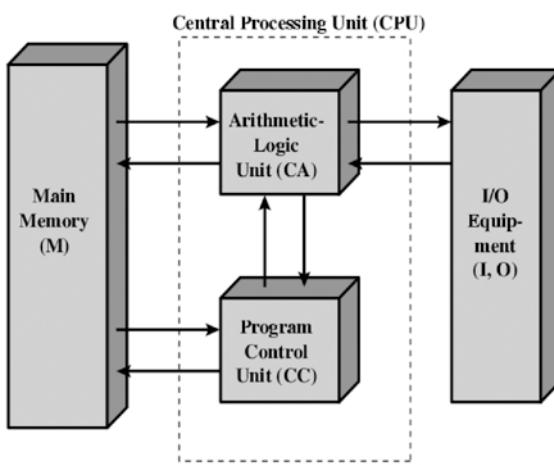
Komputer hanyalah salah satu pemroses. Agar dapat dilaksanakan oleh komputer, algoritma harus ditulis dalam notasi bahasa pemrograman sehingga dinamakan program. Jadi program adalah perwujudan atau implementasi teknis algoritma yang ditulis dalam bahasa pemrograman tertentu sehingga dapat dilaksanakan oleh komputer.

Kata “algoritma” dan “program” seringkali dipertukarkan dalam penggunaannya. Misalnya ada orang yang berkata seperti ini:

“program pengurutan data menggunakan algoritma selection sort”. Atau pertanyaan seperti ini: “bagaimana algoritma dan program menggambarkan grafik tersebut?”. Jika Anda sudah memahami pengertian algoritma yang sudah disebutkan sebelum ini, Anda dapat membedakan arti kata algoritma dan program. Algoritma adalah langkah-langkah penyelesaikan masalah, sedangkan program adalah realisasi algoritma dalam bahasa pemrograman. Program ditulis dalam salah satu bahasa pemrograman dan kegiatan membuat program disebut pemrograman (programming). Orang yang menulis program disebut pemrogram (programmer). Tiap-tiap langkah di dalam prog- ram disebut pernyataan atau instruksi. Jadi, program tersusun atas sederetan instruksi. Bila suatu instruksi dilaksanakan, maka operasi-operasi yang bersesuaian dengan instruksi tersebut dikerjakan komputer. Secara garis besar komputer tersusun atas empat komponen utama yaitu, piranti masukan, piranti keluaran, unit pemroses utama, dan memori. Unit pemroses utama (Central Processing Unit – CPU) adalah “otak” komputer, yang berfungsi mengerjakan operasi-operasi dasar seperti operasi perbandingan, operasi perhitungan, operasi membaca, dan operasi menulis. Memori adalah komponen yang berfungsi menyimpan atau mengingat. Yang disimpan di dalam memori adalah program (berisi operasi-operasi yang akan dikerjakan oleh CPU) dan data atau informasi (sesuatu yang diolah oleh operasi-operasi). Piranti masukan dan keluaran (I/O devices) adalah alat yang memasukkan data atau program ke dalam memori,

dan alat yang digunakan komputer untuk mengkomunikasikan hasil-hasil aktivitasnya. Contoh piranti masukan antara lain, papan kunci (keyboard), pemindai (scanner), dan cakram (disk). Contoh piranti keluaran adalah, layar peraga (monitor), pencetak (printer), dan cakram.

Mekanisme kerja keempat komponen di atas dapat dijelaskan sebagai berikut. Mula-mula program dimasukkan ke dalam memori komputer. Ketika program dilaksanakan (execute), setiap instruksi yang telah tersimpan di dalam memori dikirim ke CPU.



CPU mengerjakan operasi-operasi yang bersesuaian dengan instruksi tersebut. Bila suatu operasi memerlukan data, data dibaca dari piranti masukan, disimpan di dalam memori lalu dikirim ke CPU untuk

Gambar 1.1. Komponen-komponen Utama Komputer

operasi yang memerlukannya tadi. Bila proses menghasilkan keluaran atau informasi, keluaran disimpan ke dalam memori, lalu memori menuliskan keluaran tadi ke piranti keluaran (misalnya dengan menampilkannya di layar monitor).

1.6 Belajar Memprogram & Belajar Bahasa Pemrograman

Belajar memprogram tidak sama dengan belajar bahasa pemrograman. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami. Sedangkan belajar bahasa pemrograman berarti belajar memakai

suatu bahasa aturan-aturan tata bahasanya, pernyataan-pernyataannya, tata cara pengoperasian compiler-nya, dan memanfaatkan pernyataan-pernyataan tersebut untuk membuat program yang ditulis hanya dalam bahasa itu saja. Sampai saat ini terdapat puluhan bahasa pemrograman, antara lain bahasa rakitan (assembly), Fortran, Cobol, Ada, PL/I, Algol, Pascal, C, C++, Basic, Prolog, LISP, PRG, bahasa-bahasa simulasi seperti CSMP, Simscript, GPSS, Dinamo. Berdasarkan terapannya, bahasa pemrograman dapat digolongkan atas dua kelompok besar:

Bahasa pemrograman bertujuan khusus,

Yang termasuk kelompok ini adalah **Cobol** (untuk terapan bisnis dan administrasi), **Fortran** (terapan komputasi ilmiah), bahasa rakitan (terapan pemrograman mesin), **Prolog** (terapan kecerdasan buatan), bahasa-bahasa simulasi, dan sebagainya.

Bahasa perograman bertujuan umum,

Yang dapat digunakan untuk berbagai aplikasi. Yang termasuk kelompok ini adalah bahasa Pascal, Basic dan C. Tentu saja pembagian ini tidak kaku.

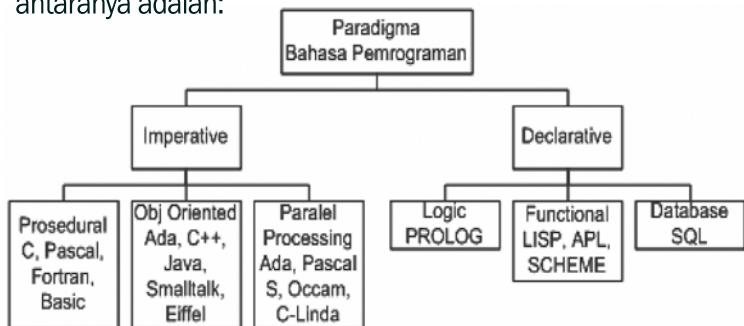
Bahasa-bahasa bertujuan khusus tidak berarti tidak bisa digunakan untuk aplikasi lain. Cobol misalnya, dapat juga digunakan untuk terapan ilmiah, hanya saja kemampuannya terbatas. Yang jelas, bahasa-bahasa pemrograman yang berbeda dikembangkan untuk bermacam-macam terapan yang berbeda pula.

Berdasarkan pada apakah notasi bahasa pemrograman lebih “dekat” ke mesin atau ke bahasa manusia, maka bahasa pemrograman dikelompokkan atas dua macam:

- **Bahasa tingkat rendah.** Bahasa jenis ini dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (translator). Contohnya adalah bahasa mesin. CPU mengambil instruksi dari memori, langsung mengerti dan langsung mengerjakan operasinya. Bahasa tingkat rendah bersifat primitif, sangat sederhana, orientasinya lebih dekat ke mesin, dan sulit dipahami manusia.

Sedangkan bahasa rakitan dimasukkan ke dalam kelompok ini karena alasan notasi yang dipakai dalam bahasa ini lebih dekat ke mesin, meskipun untuk melaksanakan instruksinya masih perlu penerjemahan ke dalam bahasa mesin.

■ **Bahasa tingkat tinggi**, yang membuat pemrograman lebih mudah dipahami, lebih “manusiawi”, dan berorientasi ke bahasa manusia (bahasa Inggris). Hanya saja, program dalam bahasa tingkat tinggi tidak dapat langsung dilaksanakan oleh komputer. Ia perlu diterjemahkan terlebih dahulu oleh sebuah translator bahasa (yang disebut kompilator atau compiler) ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU. Contoh bahasa tingkat tinggi adalah Pascal, PL/I, Ada, Cobol, Basic, Fortran, C, C++, dan sebagainya. Bahasa pemrograman bisa juga dikelompokkan berdasarkan pada tujuan dan fungsinya. Di antaranya adalah:



Gambar 1.2 Pembagian Bahasa Pemrograman

Secara sistematis berikut diberikan kiat-kiat untuk belajar memprogram dan belajar bahasa pemrograman serta produk yang dapat dihasilkan:

■ Belajar Memprogram

- ♥ Belajar memprogram: belajar bahasa pemrograman.
- ♥ Belajar memprogram: belajar tentang strategi pemecahan masalah, metodologi dan sistematika pemecahan masalah kemudian menuliskannya dalam notasi yang disepakati bersama.

- Belajar memprogram: bersifat pemahaman persoalan, analisis dan sintesis.
- Belajar memprogram, titik berat: designer program.

■ **Belajar Bahasa Pemrograman**

- Belajar bahasa pemrograman : belajar memakai suatu bahasa pemrograman, aturan sintaks, tatacara untuk memanfaatkan pernyataan yang spesifik untuk setiap bahasa.
- Belajar bahasa pemrograman, titik berat : coder.

■ Produk yang Dihasilkan Pemrogram

- Program dengan rancangan yang baik (metodologis, sistematis).
- Dapat dieksekusi oleh mesin.
- Berfungsi dengan benar.
- Sanggup melayani segala kemungkinan masukan.
- Disertai dokumentasi.
- Belajar memprogram, titik berat: designer program.

1.7. Menilai Sebuah Algoritma

Ketika manusia berusaha memecahkan masalah, metode atau teknik yang digunakan untuk memecahkan masalah itu ada kemungkinan bisa banyak (tidak hanya satu). Dan kita memilih mana yang terbaik di antara teknik-teknik itu. Hal ini sama juga dengan algoritma, yang memungkinkan suatu permasalahan dipecahkan dengan metode dan logika yang berlainan. Yang menjadi pertanyaan adalah bagaimana mengukur mana algoritma yang terbaik. Beberapa persyaratan untuk menjadi algoritma yang baik adalah:

- **Tingkat kepercayaannya tinggi (reability).** Hasil yang diperoleh dari proses harus berakurasi tinggi dan benar.
- **Pemrosesan yang efisien (cost rendah)**. Proses harus diselesaikan secepat mungkin dan frekuensi kalkulasi yang sependek mungkin.
- **Sifatnya general.** Bukan sesuatuyang hanya untuk menyelesaikan satu kasus saja, tapi juga untuk kasus lain yang lebih general.

- **Bisa dikembangkan (expandable).** Haruslah sesuatu yang dapat kita kembangkan lebih jauh berdasarkan perubahan requirement yang ada.
- **Mudah dimengerti.** Siapapun yang melihat, dia akan bisa memahami algoritma Anda. Susah dimengertinya suatu program akan membuat susah di-maintenance (kelola).
- **Portabilitas yang tinggi (portability).** Bisa dengan mudah diimplementasikan di berbagai platform komputer.
- **Precise (tepat, betul, teliti)** . Setiap instruksi harus ditulis dengan seksama dan tidak ada keragu-raguan, dengan demikian setiap instruksi harus dinyatakan secara eksplisit dan tidak ada bagian yang dihilangkan karena pemroses dianggap sudah mengerti. Setiap langkah harus jelas dan pasti.
- **Jumlah langkah atau instruksi berhingga dan tertentu** Artinya, untuk kasus yang sama banyaknya, langkah harus tetap dan tertentu meskipun datanya berbeda.
- **Efektif.** Tidak boleh ada instruksi yang tidak mungkin dikerjakan oleh pemroses yang akan menjalankannya. Contoh: Hitung akar 2 dengan presisi sempurna. Instruksi di atas tidak efektif, agar efektif instruksi tersebut diubah. Misal: Hitung akar 2 sampai lima digit di belakang koma.
- **Harus terminate.** Jalannya algoritma harus ada kriteria berhenti. Pertanyaannya adalah apakah bila jumlah instruksi berhingga maka pasti terminate?
- **Output yang dihasilkan tepat.** Jika langkah-langkah algoritmanya logis dan diikuti dengan seksama maka dihasilkan output yang diinginkan.

1.8. Penyajian Algoritma

Penyajian algoritma secara garis besar bisa dalam 2 bentuk penyajian yaitu tulisan dan gambar. Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu (misalnya bahasa Indonesia atau bahasa Inggris) dan pseudocode.

Pseudocode adalah kode yang mirip dengan kode pemrograman yang sebenarnya seperti Pascal, atau C, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram. Sedangkan algoritma disajikan dengan gambar, misalnya dengan flowchart. Teknik flowchart (diagram alur) akan dibahas lebih lanjut pada Bab 2.

1.9. Struktur Dasar Algoritma

Algoritma berisi langkah-langkah penyelesaian suatu masalah. Langkah-langkah tersebut dapat berupa runtunan aksi (sequence), pemilihan aksi (selection), pengulangan aksi (iteration) atau kombinasi dari ketiganya. Jadi struktur dasar pembangunan algoritma ada tiga, yaitu:

- ♥ **Struktur Runtunan:** digunakan untuk program yang pernyataannya sequential atau urutan.
- ♥ **Struktur Pemilihan:** digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.
- ♥ **Struktur Perulangan:** digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang.

1.10. Tahapan dalam Pemrograman

Langkah-langkah yang dilakukan dalam menyelesaikan masalah dalam pemrograman dengan komputer adalah:

- ♥ **Definisikan Masalah.** Berikut adalah hal-hal yang harus diketahui dalam analisis masalah supaya kita mengetahui bagaimana permasalahan tersebut:
 - Kondisi awal, yaitu input yang tersedia.
 - Kondisi akhir, yaitu output yang diinginkan.
 - Data lain yang tersedia.
 - Operator yang tersedia.
 - Syarat atau kendala yang harus dipenuhi.

Contoh kasus:

Menghitung biaya percakapan telepon di wartel. Proses yang perlu diperhatikan adalah:

- Input yang tersedia adalah jam mulai bicara dan jam selesai bicara.
- Output yang diinginkan adalah biaya percakapan.
- Data lain yang tersedia adalah besarnya pulsa yang digunakan dan biaya per pulsa.
- Operator yang tersedia adalah pengurangan (-), penambahan (+), dan perkalian (*).
- Syarat kendala yang harus dipenuhi adalah aturan jarak dan aturan waktu.

• **Buat Algoritma dan Struktur Cara Penyelesaian.** Jika masalahnya kompleks, maka dibagi ke dalam modul-modul. Tahap penyusunan algoritma seringkali dimulai dari langkah yang global terlebih dahulu. Langkah global ini diperhalus sampai menjadi langkah yang lebih rinci atau detail. Cara pendekatan ini sangat bermanfaat dalam pembuatan algoritma untuk masalah yang kompleks. Penghalusan langkah dengan cara memecah langkah menjadi beberapa langkah. Setiap langkah diuraikan lagi menjadi beberapa langkah yang lebih sederhana. Penghalusan langkah ini akan terus berlanjut sampai setiap langkah sudah cukup rinci dan tepat untuk dilaksanakan oleh pemroses.

• **Menulis Program Algoritma yang telah dibuat**, diterjemahkan dalam bahasa computer menjadi sebuah program. Perlu diperhatikan bahwa pemilihan algoritma yang salah akan menyebabkan program memiliki untuk kerja yang kurang baik. Program yang baik memiliki standar penilaian:

a. **Standar teknik pemecahan masalah**

■ **Teknik Top-Down.** Teknik pemecahan masalah yang paling umum digunakan. Prinsipnya adalah suatu masalah yang kompleks dibagi-bagi ke dalam beberapa kelompok masalah yang lebih kecil. Dari masalah yang

kecil tersebut dilakukan analisis. Jika dimungkinkan maka masalah tersebut akan dipilah lagi menjadi subbagian-subbagian dan setelah itu mulai disusun langkah-langkah penyelesaian yang lebih detail.

- **Teknik Bottom-Up.** Prinsip teknik bottom up adalah pemecahan masalah yang kompleks dilakukan dengan menggabungkan prosedur-prosedur yang ada menjadi satu kesatuan program sebagai penyelesaian masalah tersebut.

b. Standar penyusunan program

- Kebenaran logika dan penulisan.
- Waktu minimum untuk penulisan program.
- Kecepatan maksimum eksekusi program.
- Ekspresi penggunaan memori.
- Kemudahan merawat dan mengembangkan program.
- User Friendly.
- Portability.
- Pemrograman modular.

♥ Mencari Kesalahan

- Kesalahan sintaks (penulisan program).
- Kesalahan pelaksanaan: semantik, logika, dan ketelitian.

♥ Uji dan Verifikasi Program

Pertama kali harus diuji apakah program dapat dijalankan. Apabila program tidak dapat dijalankan maka perlu diperbaiki penulisan sintaksisnya tetapi bila program dapat dijalankan, maka harus diuji dengan menggunakan data-data yang biasa yaitu data yang diharapkan oleh sistem. Contoh data ekstrem, misalnya, program menghendaki masukan jumlah data tetapi user mengisikan bilangan negatif. Program sebaiknya diuji menggunakan data yang relatif banyak.

♥ Dokumentasi Program

Dokumentasi program ada dua macam yaitu dokumentasi internal dan dokumentasi eksternal. Dokumentasi internal adalah dokumentasi yang dibuat di dalam program yaitu setiap

kita menuliskan baris program sebaiknya diberi komentar atau keterangan supaya mempermudah kita untuk mengingat logika yang terdapat di dalam instruksi tersebut, hal ini sangat bermanfaat ketika suatu saat program tersebut akan dikembangkan. Dokumentasi eksternal adalah dokumentasi yang dilakukan dari luar program yaitu membuat user guide atau buku petunjuk aturan atau cara menjalankan program tersebut.

• **Pemeliharaan Program**

- Memperbaiki kekurangan yang ditemukan kemudian.
- Memodifikasi, karena perubahan spesifikasi.

Pemrograman Prosedural

Algoritma berisi urutan langkah-langkah penyelesaian masalah. Ini berarti algoritma adalah proses yang prosedural. Pada program prosedural, program dibedakan antara bagian data dengan bagian instruksi. Bagian instruksi terdiri dari atas runtunan (sequence) instruksi yang dilaksanakan satu per satu secara berurutan oleh sebuah pemroses. Alur pelaksanaan instruksi dapat berubah karena adanya pencabangan kondisional. Data yang disimpan di dalam memori dimanipulasi oleh instruksi secara beruntun. Kita katakan bahwa tahapan pelaksanaan program mengikuti pola beruntun atau prosedural. Paradigma pemrograman seperti ini dinamakan pemrograman prosedural. Bahasa-bahasa tingkat tinggi seperti Cobol, Basic, Pascal, Fortran, dan C/C++ mendukung kegiatan pemrograman prosedural, karena itu mereka dinamakan juga bahasa prosedural.

Selain paradigma pemrograman prosedural, ada lagi paradigma yang lain yaitu pemrograman berorientasi objek (Object Oriented Programming atau OOP). Paradigma pemrograman ini merupakan trend baru dan sangat popular akhir-akhir ini. Pada paradigma OOP, data dan instruksi dibungkus (encapsulation) menjadi satu. Kesatuan ini disebut kelas (class) dan instansiasi kelas pada saat run-time disebut objek (object). Data di dalam objek hanya dapat diakses oleh instruksi yang ada di dalam objek itu saja.

Paradigma pemrograman yang lain adalah pemrograman fungsional, pemrograman deklaratif, dan pemrograman konkuren. Buku ini hanya menyajikan paradigma pemrograman prosedural saja. Paradigma pemrograman yang lain di luar cakupan buku ini.

Soal dan Penyelesaian

Kasus 1 : Menghitung luas dan keliling lingkaran. Proses kerjanya sebagai berikut :

- Baca jari-jari lingkaran
 - Tentukan konstanta phi = 3.14
 - Hitung luas dan keliling
- $$L = \phi * r * r$$
- $$K = 2 * \phi * r$$
- Cetak luas dan keliling

Kasus 2 : Menghitung rata-rata tiga buah data

- a. Algoritma dengan struktur bahasa Indonesia
 - Baca bilangan a, b, dan c
 - Jumlahkan ketiga bilangan tersebut
 - Bagi jumlah tersebut dengan 3
 - Tulis hasilnya

b. Algoritma dengan pseudocode

input (a, b, c)

Jml = a+b+c

Rerata = Jml/3

Output (Rerata)

Kasus 3 : Algoritma konversi suhu dalam derajat Celcius ke derajat Kelvin
Penyelesaian menggunakan pseudocode:

Input (Celcius)

Kelvin = Celcius + 273

Output (Kelvin)

BAB II

Data, Variabel dan Operator

Sebuah bahasa pemrograman tidak terlepas pada Tipe Data, karena memiliki fungsi yang sangat penting yaitu untuk menyatakan jenis nilai yang dimiliki oleh sebuah variabel. Selain itu, bahasa pemrograman juga membutuhkan Variabel yang berfungsi untuk menyimpan nilai baik itu huruf, karakter atau angka dan Operator yang biasanya diidentikan dengan perhitungan yang bermain pada angka di tipe data integer atau yang memiliki nilai angka bulat.

Dalam Bahasa pemrograman terdapat tiga elemen dasar yang digunakan, yaitu: **tipe data, variabel, dan operator.**

Pengertian Tipe Data

Tipe data adalah himpunan nilai yang dapat dimiliki oleh sebuah data. Tipe data menentukan apakah sebuah nilai dapat dimiliki sebuah data atau tidak, serta operasi apa yang dapat dilakukan pada data tersebut.

Contoh tipe data dalam dunia nyata adalah bilangan bulat.

Jika sebuah data, misalnya umur harus berupa bilangan bulat maka dapat dipastikan bahwa 25, 13, 7 dapat menjadi nilai umur, sedangkan angka yang menggunakan koma seperti 7,5, 19.655 bukan merupakan contoh dari suatu nilai umur.

Contoh bilangan bulat ini dapat kita lihat dalam kasus sehari - hari khususnya dalam hal pencacahan (bilangan cacah : 1, 2, 3, 4, ... yang merupakan himpunan himpunan bilangan bulat).

Contoh :

Kasus 1 : Jumlah siswa dalam kelas ada 20. Angka 20 tersebut adalah bilangan bulat. Tidak akan ditemukan pernyataan : jumlah siswa dalam kelas ada 20,5.

Kasus 2 : Jumlah mobil yang diparkir di tempat parkir. Kita akan menggunakan bilangan bulat dalam kasus ini. Tidak pernah akan kita gunakan angka angka 20,72 atau 30/7 sebagai jumlah dari mobil yang sedang parkir.

Kasus 3 : Selain itu, misalnya data nama seseorang yaitu "Daeng Raja" yang merupakan sebuah deretan huruf dan lain sebagainya.

Dalam sebuah program, setiap variabel dan konstanta memiliki tipe data yang harus dideklarasikan di awal program. Pendeklarasi tipe data tersebut bertujuan untuk menentukan besarnya tempat dalam memori yang akan digunakan untuk menyimpan data pada tersebut saat program dijalankan.

Tipe data dasar adalah tipe data yang dapat langsung digunakan. Secara umum terdapat 2 tipe data dasar, yaitu numerik dan kategorik. Tipe data numerik terdiri atas angka/ kumpulan angka serta dapat mengalami operasi perhitungan, sedangkan tipe data kategorik dapat berupa angka maupun huruf namun tidak dapat mengalami operasi perhitungan.

Contoh Tipe Data Dasar

1) *Integer (bilangan bulat)*

Integer adalah tipe data dasar berupa bilangan yang tidak mengandung pecahan desimal. Tipe data ini juga memiliki urutan, sehingga dapat dibandingkan satu dengan lainnya.

Contoh integer:

- 8
- 5
- -10
- 135
- 2013

Secara teoritis, tipe data integer tidak memiliki batasan, yaitu dari minus tak hingga hingga plus tak hingga. Namun dalam pemrograman yang menggunakan bahasa pemrograman C++, secara umum dikenal beberapa macam tipe data integer, yaitu:

2) *Real (bilangan riil)*

Real adalah tipe data dasar berupa bilangan yang memiliki pecahan desimal. Dalam pemrograman, nilai dengan tipe data ini harus ditulis dengan sebuah titik sebagai pemisah bilangan utuh dan bilangan pecahannya.

Tipe data ini digunakan untuk perhitungan yang melibatkan bilangan pecahan, seperti perhitungan kosinus, akar persamaan, dan sebagainya. Tipe data ini juga memiliki urutan, sehingga dapat dibandingkan satu dengan lainnya.

Contoh real:

- 0.5
- 0.17
- -3.465
- 92.0
- 4.3000+E9

3) **Char (karakter)**

Char adalah tipe data dasar yang terdiri atas satu buah angka, huruf, tanda baca atau karakter khusus. Untuk menyimpan sebuah karakter, diperlukan 1 byte atau 8 bit tempat didalam memori.

Dalam sebuah program, penulisan tipe data char diawali dan diakhiri dengan tanda kutip ganda. Selain itu, terdapat sebuah karakter kosong yang disebut dengan null atau nil dan dituliskan sebagai "".

Contoh char:

- "20"
- "A"
- "?"
- "+"
- "/"

Perhatikan bahwa 20 adalah integer sedangkan "20" adalah char.

4) **String**

String adalah tipe data dasar yang berupa kumpulan karakter dengan panjang tertentu. Meskipun berupa kumpulan karakter, karena tipe data string sering digunakan dalam pemrograman, string dianggap sebagai tipe data dasar.

Untuk penyimpanan string didalam memori, dibutuhkan 1 byte untuk tiap karakternya. Serupa dengan penulisan karakter, penulisan sebuah string juga harus diawali dan diakhiri dengan tanda petik ganda. String juga mengenal null yang dituliskan dengan "".

Contoh string:

- "PAREPARE"
- "Universitas Muhammadiyah Parepare"
- "ABC3456"
- "AZIIZAH"
- "06012001"
- "D"

Perhatikan bahwa sebuah karakter tunggal ("D") juga merupakan string.

5) Boolean (*bilangan logika*)

Sebuah data boolean memiliki tepat dua buah kemungkinan nilai, direpresentasikan sebagai Benar dan Salah, atau True dan False, atau dapat juga dilambangkan dengan 1 dan 0. Tipe data ini dapat digunakan untuk pemilihan dengan kondisi-kondisi tertentu, dimana program harus memilih aksi apa yang akan dijalankan dengan parameter tertentu.

Tipe data ini paling sering digunakan untuk range yang memiliki dua buah nilai:

- Lulus - tidak lulus.
- Benar - salah.

Pengertian Variabel

Variabel atau peubah adalah obyek yang nilainya dapat berubah-ubah dalam sebuah program. Pada saat sebuah variabel dideklarasikan, program "memesan" tempat dengan ukuran tertentu (sesuai tipe datanya) pada memori untuk menyimpan nilai dari variabel tersebut.

Pemrogram dapat memberikan nama pada sebuah variabel untuk mempermudah pemanggilan variabel tersebut di dalam program. Pada saat mendeklarasikan sebuah variabel, pemrogram harus menyebutkan nama variabel dan tipe data dari variabel tersebut.

Dalam bentuk flowchart, deklarasi variabel digambarkan sebagai sebuah proses.

Misalnya sebagai berikut:

```
x : integer  
nama : string  
tinggiBadan: real
```

Contoh deklarasi variabel dalam psedeucode :

```
KAMUS DATA {awal deklarasi variabel}
```

```
x : integer  
nama: string  
tinggiBadan: real  
jenisKelamin : char  
status : boolean
```

Sebelum kita menuliskan program dalam bahasa delphi, ada baiknya kita mengenal terlebih dahulu struktur dan format penulisan program dalam bahasa delphi.

```
// Contoh Program  
Program Salam  
Begin  
    WriteLN ("Assalamualaikum");  
End.
```

Pengertian Operator

Operator adalah pengendali operasi yang akan dilakukan pada beberapa operan sehingga membentuk sebuah ekspresi. Secara umum, dalam sebuah ekspresi terdapat sebuah operator yang diapit dua operan.

Contohnya pada ekspresi:

$x + y$
x dan y adalah operan, sedangkan
“+” adalah operatornya

Terdapat tiga macam operator yang biasa digunakan dalam pemrograman, yaitu:

1. Operator Aritmatik

Operator ini membentuk perhitungan aritmatik. Kedua operan dari operasi aritmatik ini dapat berupa nilai integer atau real.

Operator yang termasuk tipe ini adalah:

Output dari operasi aritmatik akan memiliki tipe data yang sama dengan tipe data kedua operannya. Misalnya, jika sebuah bilangan integer dijumlahkan dengan bilangan integer lainnya maka outputnya adalah bilangan integer juga. Selain itu perlu diperhatikan pula bahwa sebuah operator aritmatik tidak dapat diterapkan pada dua bilangan dengan tipe data yang berbeda.

Contoh program dengan operasi aritmatik:

```
// Program Aritmatik
```

```
Deklarasi Var A, B, C, D : integer ;
```

```
Begin
```

```
    Write ("Nilai A :"); readLN (A);
```

```
    Write ("Nilai B :"); readLN (B);
```

```
    C:= A + B;
```

```
    D:= A - B;
```

```
    Write ("Hasil Penjumlahan : ", C); writeLN ;
```

```
    Write ("Hasil Pengurangan: ", D); writeLN ;
```

```
End.
```

Program di atas akan mengembalikan nilai hasil penjumlahan sesuai dengan inputan. Misalnya pada inputan pertama kita masukan 20 dan inputan kedua adalah 30 maka hasilnya adalah 50. Sedangkan untuk operasi pengurangan hasilnya 10. Outputnya adalah:

Masukan Nilai A : 30

Masukan Nilai B : 20

Hasil Penjumlahan 30 + 20 = 50

Hasil Pengurangan 30-20 = 10

2. Operator Assignment

Dalam pemrograman, Operator ini digunakan memasukan nilai kedalam sebuah variabel, tanpa menghilangkan atau mengosongkan nilai variabel sebelumnya. Contoh penggunaan operator ini adalah sebagai berikut :

3. Increase and Decrease

Penulisan ini dilambangkan dengan `++` (Increase) dan `--` (decrease). Operator ini berfungsi untuk menaikkan atau menurunkan satu satuan nilai pada sebuah variabel.

Contoh penggunaannya adalah pada contoh dibawah ini :

```
...
a++;
a += 1;
a = a + 1;
...
```

Ada dua macam penulisan operator ini, yaitu simbol dapat ditulis sebelum nama variabel dan setelah variabel. Adapun perbedaan antara keduanya adalah :

```
B = 3;
A = ++B;
// A = 4, B = 4

B = 3;
A = B++;
// A = 3, B = 4
```

4. Operator Relasional

Operator ini membandingkan dua operan dan hasilnya berupa nilai boolean (BENAR atau SALAH). Operasi relasional dapat dilakukan pada dua nilai dengan tipe data yang sama: tipe data integer, riil, char, string, maupun boolean.

5. Operator logika

Operator logika adalah operator yang digunakan untuk mengkombinasikan hasil ekspresi yang mengandung operator relasional.

BAB III

Teknik Penyajian

Algoritma

3.1 Pengantar

Algoritma dapat disajikan dengan dua teknik yaitu teknik tulisan dan teknik gambar. Teknik tulisan biasanya menggunakan metode structure English dan pseudocode, sedangkan teknik gambar biasanya menggunakan diagram alir (flow chart).

3.2 Structure English dan Pseudocode

Structure English merupakan alat yang cukup efisien untuk menggambarkan suatu algoritma. Basis dari structure english adalah bahasa Inggris, tetapi juga bisa digunakan bahasa Indonesia, sedangkan pseudocode berarti kode yang mirip dengan kode pemrograman sebenarnya. Pseudocode berasal dari kata pseudo yang berarti imitasi/mirip/menyerupai dan code yang berarti program.

Pseudocode berbasis pada kode program yang sesungguhnya seperti Pascal, C, C++. Pseudocode lebih rinci dari structure English misalnya dalam menyatakan tipe data yang digunakan.

Contoh struktur Indonesia

Baca data jam_kerja

Hitung gaji adalah jam_kerja dikalikan tarif

Tampilkan gaji

Pseudocode dengan Pascal :

Read jam_kerja

*Gaji := jam_kerja * tarif*

Write gaji

3.3 Flowchart (Diagram Alur)

Flowchart dalam Bahasa Indonesia diterjemahkan sebagai Diagram Alir. Dari dua kata ini, maka dapat kita bayangkan bahwa flowchart itu berbentuk diagram yang bentuknya dapat mengalirkan sesuatu. Hal ini memang benar, flowchart memang melukiskan suatu aliran kegiatan dari awal hingga akhir mengenai suatu langkah-langkah dalam penyelesaian suatu masalah. Masalah tersebut bisa bermacam-macam, mulai dari masalah yang sederhana sampai yang kompleks. Masalah yang kita pelajari tentu saja masalah pemrograman dengan menggunakan komputer, tetapi secara logika dapat kita awali dengan mengamati permasalahan dalam kehidupan sehari-hari kita. Contoh sederhananya adalah masalah menambal ban sepeda yang bocor.

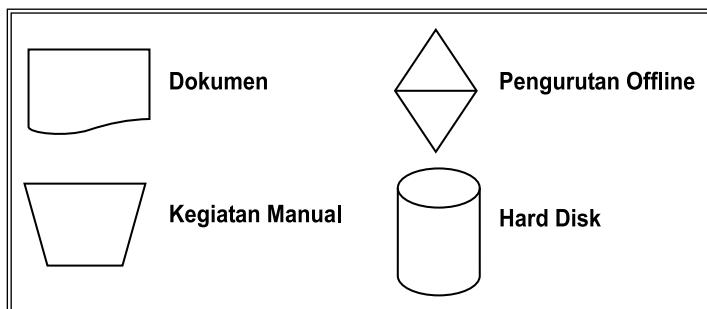
Dalam menambal ban sepeda yang bocor, tentu saja diperlukan langkah-langkah yang berurutan agar hasilnya dapat sesuai dengan apa yang kita inginkan, yaitu secangkir menambal ban. Demikian halnya dalam memprogram, diperlukan suatu algoritma (urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis) agar program yang kita buat dapat berjalan dan memberikan hasil yang valid. Nah, untuk merepresentasikan algoritma itulah kita gunakan flowchart. Flowchart biasanya dipelajari pada saat kita mulai mempelajari pemrograman. Mengapa demikian? Hal ini tak lain karena dengan mempelajari flowchart, kita diharapkan dapat berpikir secara logis, dapat menentukan komponen program (input dan output), serta memahami alur program. Flowchart merupakan teknik yang memudahkan kita dalam memprogram, dalam hal ini memudahkan dalam arti mengantisipasi agar tak ada komponen program yang tertinggal.

3.3.1 Defenisi Flowchart

Flowchart adalah representasi grafik dari langkah-langkah yang harus diikuti dalam menyelesaikan suatu permasalahan yang terdiri atas sekumpulan simbol, dimana masing-masing simbol merepresentasikan suatu kegiatan tertentu. Flowchart diawali dengan penerimaan input, pemrosesan input, dan diakhiri dengan penampilan output. Dengan kata lain flowchart merupakan suatu gambar yang menjelaskan urutan: pembacaan data, pemrosesan data, pengambilan keputusan terhadap data, penyajian hasil pemrosesan data.

Ada dua macam flowchart yang menggambarkan proses dengan komputer, yaitu:

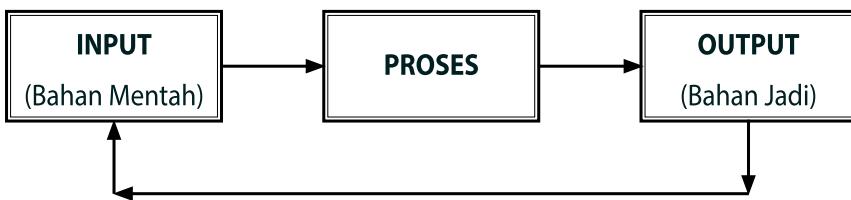
- **Flowchart sistem** yaitu bagan dengan simbol-simbol tertentu yang menggambarkan urutan prosedur dan proses suatu file dalam suatu media menjadi file di dalam media lain, dalam suatu sistem pengolahan data. Beberapa contoh Flowchart sistem:



- **Flowchart program** yaitu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses dan hubungan antar proses secara mendetail di dalam suatu program.

3.3.2 Kaidah-Kaidah Umum Pembuatan Flowchart Program

Dalam pembuatan flowchart Program tidak ada rumus atau patokan yang bersifat mutlak. Karena flowchart merupakan gambaran hasil pemikiran dalam menganalisis suatu masalah dengan komputer. Sehingga flowchart yang dihasilkan dapat bervariasi antara satu pemrogram dengan yang lainnya. Namun secara garis besar setiap pengolahan selalu terdiri atas 3 bagian utama, yaitu: Input, Proses pengolahan dan Output.



Untuk pengolahan data dengan komputer, urutan dasar pemecahan suatu masalah:

- **START**, berisi pernyataan untuk persiapan peralatan yang diperlukan sebelum menangani pemecahan persoalan.
- **READ**, berisi pernyataan kegiatan untuk membaca data dari suatu peralatan input.
- **PROSES**, berisi kegiatan yang berkaitan dengan pemecahan persoalan sesuai dengan data yang dibaca.
- **WRITE**, berisi pernyataan untuk merekam hasil kegiatan ke peralatan output.
- **END**, mengakhiri kegiatan pengolahan

Berikut merupakan beberapa contoh simbol flowchart yang disepakati oleh dunia pemrograman:

■ *Simbol Input*

Simbol input digambarkan dengan bangun jajar genjang. Simbol ini digunakan untuk melambangkan kegiatan penerimaan input. Dalam simbol ini, kita dapat menuliskan input yang diperlukan pada suatu waktu secara satu per satu maupun secara keseluruhan, tetapi biasanya input yang dimasukkan pada suatu waktu, dituliskan bersamaan secara keseluruhan dengan tujuan efisiensi ruang gambar.



Gambar 2.1 Simbol Input

■ *Simbol Proses*



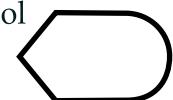
Simbol proses digambarkan dengan bangun persegi panjang. Simbol ini digunakan untuk melambangkan

Gambar 2.2. Simbol Proses kegiatan pemrosesan input.

Dalam simbol ini, kita dapat menuliskan operasi-operasi yang dikenakan pada input, maupun operasi lainnya. Sama seperti aturan pada simbol input, penulisan dapat dilakukan secara satu per satu maupun secara keseluruhan.

■ *Simbol Output*

Simbol output digambarkan dengan bangun seperti Gambar 2.3. Simbol ini digunakan untuk melambangkan kegiatan penampilan output. Dalam simbol ini, kita dapat menuliskan semua output yang harus ditampilkan oleh program. Sama seperti aturan pada dua simbol sebelumnya, penulisan dapat dilakukan secara satu per satu maupun secara keseluruhan.



Gambar 2.3 Simbol Output

■ *Simbol Percabangan*

Simbol percabangan digambarkan dengan bangun belah ketupat. Simbol ini digunakan untuk melambangkan percabangan, yaitu pemeriksaan terhadap suatu kondisi. Dalam simbol ini, kita menuliskan keadaan yang harus dipenuhi. Hasil dari pemeriksaan dalam simbol ini adalah "YES" atau "NO".

Gambar 2.4 Simbol Percabangan menghasilkan keadaan benar, maka jalur yang harus dipilih adalah jalur yang berlabel Yes, sedangkan jika pemeriksaan menghasilkan keadaan salah, maka jalur yang harus dipilih adalah jalur yang berlabel No. Berbeda dengan aturan pada tiga simbol sebelumnya, penulisan keadaan dilakukan secara satu per satu.

■ *Simbol Prosedur*

Simbol prosedur digambarkan dengan bangun seperti Gambar 6. Simbol ini berperan sebagai blok pembangun dari suatu program. *Gambar 2.5 Simbol Prosedur* memiliki suatu flowchart yang berdiri sendiri diluar flowchart utama. Jadi dalam simbol ini, kita cukup menuliskan nama prosedurnya saja, jadi sama seperti jika kita melakukan pemanggilan suatu prosedur pada program utama (main program). Sama dengan aturan pada simbol percabangan, penulisan nama prosedur dilakukan secara satu per satu.

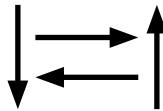


Gambar 2.5 Simbol Prosedur

■ *Simbol Garis Alir*

Simbol garis alir atau flow lines digambarkan dengan anak panah. simbol ini digunakan untuk menghubungkan setiap langkah dalam flowchart dan menunjukkan kemana arah aliran diagram.

Anak panah ini harus mempunyai arah dari kiri ke kanan atau dari atas ke bawah. Anak panah ini juga dapat diberi label, khususnya jika keluar dari symbol percabangan.



Gambar 2.6
Simbol Garis Alir

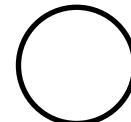
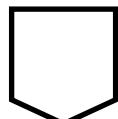
■ Simbol Terminator

Simbol terminator digambarkan dengan bangun seperti Gambar 8. Terminator berfungsi untuk menandai awal dan akhir dari suatu flowchart. Simbol ini biasanya diberi label START untuk menandai awal dari flowchart, dan label STOP untuk menandai akhir dari flowchart. Jadi dalam sebuah flowchart pasti terdapat sepasang terminator yaitu terminator start dan stop.



■ Simbol Konektor

Simbol konektor digunakan untuk menghubungkan suatu langkah dengan langkah lain dalam sebuah flowchart dengan keadaan on page atau off page. On page connector digunakan untuk menghubungkan suatu langkah dengan langkah lain dari flowchart dalam satu halaman, sedangkan off page connector digunakan untuk menghubungkan suatu langkah dengan langkah lain dari flowchart dalam halaman yang berbeda.



Gambar 2.8a
Simbol On-Page Connector

Connector ini biasanya dipakai saat media yang kita gunakan untuk menggambar flowchart tidak cukup luas untuk memuat gambar secara utuh, jadi perlu dipisahpisahkan. Dalam sepasang connector biasanya diberi label tertentu yang sama agar lebih mudah diketahui pasangannya.

Gambar 28.b
Simbol Off-Page
Connector

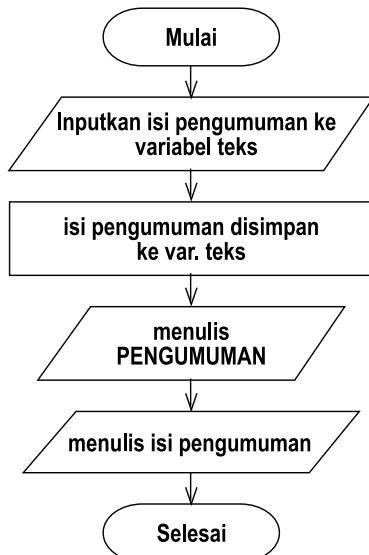
Soal dan Pembahasan

Kasus 1 :

Membuat Teks Pengumuman

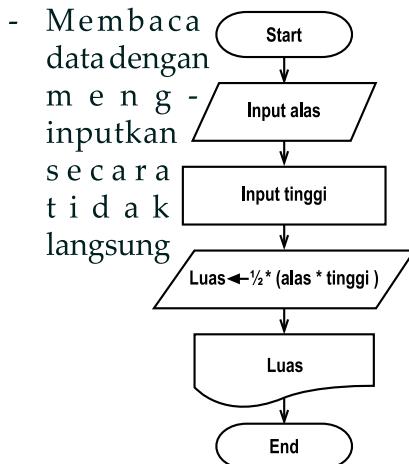
1. Mulai
2. Inputkan 'isi pengumuman'
3. Simpan isi pengumuman ke variabel 'teks'
4. Outputkan(tulis dilayar) tulisan "PENGUMUMAN"
5. Outputkanisipengumuman yang tersimpan dalam variable 'teks'
6. Selesai

Dibawah ini akan ditunjukkan diagram alir programnya / flowchart :

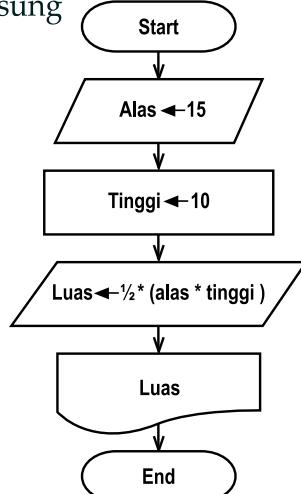


Kasus 2 :

Menghitung luas segitiga



- Membaca data dengan menginputkan secara langsung



BAB IV

Konsep Dasar Pemrograman

4.1 Elemen Pemrograman Pascal/Delphi

Pascal adalah bahasa pemrograman yang pertama kali dibuat oleh **Profesor Niklaus Wirth**, seorang anggota International Federation of Information Processing (IFIP) pada tahun 1971. Dengan mengambil nama dari matematikawan Perancis, **Blaise Pascal**, yang pertama kali menciptakan mesin penghitung, Profesor Niklaus Wirth membuat bahasa Pascal ini sebagai alat bantu untuk mengajarkan konsep pemrograman komputer kepada mahasiswanya. Selain itu, Profesor Niklaus Wirth membuat Pascal juga untuk melengkapi kekurangan-kekurangan bahasa pemrograman yang ada pada saat itu.

Sedangkan Borland Delphi dipersiapkan perusahaan Borland untuk menggantikan Turbo Pascal, karena semakin ramainya pemrograman berorientasi obyek (Object Oriented Programming) atau Visual. Borland Delphi adalah sebuah alat pengembangan aplikasi-aplikasi untuk sistem operasi Microsoft Windows. Delphi sangat berguna dan mudah digunakan untuk membuat suatu program berbasis *GUI* (*Graphical User Interface*) atau *console* (*mode teks*).

Borland Delphi atau biasa disebut Delphi saja, merupakan sarana pemrograman aplikasi visual. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Pascal atau yang kemudian disebut bahsa pemrograman Delphi. Delphi merupakan generasi penerus dari Turbo Pascal. Turbo Pascal yang diluncurkan pada tahun 1983 dirancang untuk dijalankan pada sistem operasi DOS (yang merupakan sistem operasi yang paling banyak disunakan pada saat itu). Sedangkan Delphi yang diluncurkan pertama kali tahun 1995 dirancang untuk beroperasi dibawah sistem operasi Windows.

Delphi terdapat 2 macam struktur yaitu struktur projrk dan struktur unit program Delphi:

1. Struktur Projek

Pada suatu projek yang anada bangun, terdapat sebuah file program utama yang berisi kode program untuk pengelolaan unit – unit. Kode program utama ini biasa juga disebut kode projek dan disimpan dalam file berekstensi .DPR.

2. Struktur Unit

Sebuah unit berisitipe-tipe, konstanta-konstanta, variabel dan rutin (fungsi dari prosedur). Setiap unit didefinisikan dalam file .PAS yang menangani unit tersebut.

Kelebihan Delphi :

1. Sifatnya freeware
2. Dikembangkan dengan bahasa Pascal, sehingga pagi pengguna yang terbiasa dengan dasar pemrograman turbo Pascal akan lebih familiar.
3. Komponen yang disediakan sudah cukup lengkap tanpa harus add component dari sumber lain.
4. Dokumentasi cukup lengkap.

Kelemahan Delphi :

1. Pengguna yang tidak memiliki dasar pemrograman dengan bahasa Pascal akan mengalami kesulitan untuk pertama kalinya.
2. Setiap komponen yang dimasukkan dalam form tampilan, akan diikutsertakan kode deklarasi dan inisialisasinya dalam list code. Sehingga apabila terjadi perubahan komponen, penamaan maupun kesalahan penulisan kode, program tidak mau membetulkan otomatis.
3. Apabila terdapat form/list code lain yang di-include-kan, harus dituliskan code/nama dari form/list code di bagian “uses” dan juga inisialisasi variablenya.

Delphi adalah suatu bahasa pemrograman (development language) yang digunakan untuk merancang suatu aplikasi program.

Kegunaan Delphi :

- Untuk membuat aplikasi windows.
- Untuk merancang aplikasi program berbasis grafis.
- Untuk membuat program berbasis jaringan (client/server).
- Untuk merancang program .Net (berbasis internet).

Sebelum kita membuat sebuah program, maka terlebih dahulu kita harus mengerti tentang elemen-elemen bahasa (Language elements) Turbo Pascal, seperti Reserved word, Statement, Type, Constants, Variabel, Tipe data, Label, Operator, dan lain-lain.

a. Reserved Word

Reserved word adalah kata-kata yang tidak dapat dijadikan menjadi identifier (pengenal), karena kata-kata tersebut sudah mempunyai arti tersendiri dalam Turbo Pascal/Delphi. Adapun kata-kata yang termasuk ke dalam identifier adalah:

And, asm, array, begin, case, const, constructor, destructor, div, do, downto, else, end, exports, file, for, function, goto, if, implementation, in, inherited, inline, interface, label, library, mod, nil, not, object, of, or, packed, procedure, program, record, repeat, set, shl, shr, string, then, to, type, unit, until, uses, var, while, with, xor.

b. Statement

Statement adalah salah satu dari berikut ini:

- Assignment (:=)
- Begin..end
- Case..of..else..end
- For..to/downto..do
- Goto
- If..then..else
- Inline(..)
- Procedure call
- Repeat..until
- While..do
- With..do

c. Type

Bentuk umum:

Type

Pengenal = tipe data;

.....

Pengenal = tipe data;

d. Const (Constant)

Constant yang disingkat dengan const adalah nilai konstanta (nilai tetap) yang dipasang dalam program.

Bentuk umum :

Const

Pengenal = ekspresi

.....

Pengenal = ekspresi

Const

Pengenal: type = nilai;

.....

Pengenal: type = nilai;

e. Var (Variabel)

Jika constant adalah nilai tetap, maka Variabel adalah nilai yang isinya dapat berubah-ubah. Dalam program, Variabel disingkat menjadi Var.

Bentuk umum :

Var

Pengenal, ... pengenal : Tipe data;

.....

Pengenal,... pengenal: Tipe data;

f. Tipe Data

Tipe atau jenis data dalam Turbo Pascal/Delphi dibagi kedalam 6 kelompok besar, antara lain:

1. Tipe simple:

▼ *Tipe ordinal* : dibagi kedalam 5 tipe :

Tipe	Range	Size
Shortint	128..127	8-bit
Integer	-32768..32767	16-bit
Longint	-2147483648..2147483647	32-bit
Byte	0.255	8-bit
Word	0.65535	16-bit

- ▼ **Tipe integer** : dibagi kedalam 5 bagian yaitu:

Tipe	Range	Format
Shortint	128..127	8-bit bertanda
Integer	-32768..32767	16-bit bertanda
Longint	-2147483648..2147483647	32-bit bertanda
Byte	0.255	8-bit tak bertanda
Word	0.65535	16-bit tak bertanda

Catatan : Semua tipe integer adalah tipe ordinal.

- ▼ **Tipe real** : dibagi kedalam 5 bagian yaitu:

Tipe	Range	Digit	Byte
Real	2.9e-39..1.7e38	11 - 12	6
Single	1.5e-45..3.4e38	7 - 8	4
Double	5.0e-324..1.7e308	15 - 16	8
Extended	3.4e4932..1.1e4932	19 - 20	10
comp	-9.2e18..9.2e18	19 - 20	8

Turbo Pascal/Delphi juga menyediakan 2 model floating point :

- Software floating point, {\$N-}
- 80×87 floating point, {\$N+}

- ▼ **Tipe char**

Char adalah semua tombol yang terdapat pada keyboard, atau lebih lengkapnya semua karakter yang terdapat pada kode ASCII. Apabila tipe char dijadikan konstanta, maka karakter yang dimasukkan harus

diapit oleh tanda kutip satu. Dan apabila karakter tersebut berupa tanda kutip satu, maka harus diapit oleh dua tanda kutip satu.

▼ **Tipe Boolean**

Ada 4 yang termasuk ke-dalam tipe Boolean : Boolean, wordbool, longbool, bytebool. Keempat tipe Boolean tersebut adalah tipe untuk kompatibilitas dengan Windows.

▼ **Tipe enumerated**

Bentuk umum:

Type

Nama = (pengenal,
Pengenal,...,
Pengenal);

▼ **Tipe subrange**

Bentuk umum:

Constant1 .. constant2

2. Tipe String

String adalah kumpulan dari beberapa karakter dan panjangnya tidak boleh melebihi 255 karakter. Jika string mengandung tanda kutip satu, maka tanda kutip tersebut harus diberi tanda kutip lagi.

Bentuk umum :

String [constant] Atau String

Ciri – ciri

Apabila panjang string tidak ditentukan maka panjangnya dianggap 255 karakter. Oleh karena itu, untuk menghemat memori, biasakanlah selalu menentukan panjang string yang akan dibuat.

3. Tipe Structured

Tipe structured adalah tipe yang terdiri lebih dari satu nilai.

Sedangkan tipe structured terdiri dari 5 tipe :

a) Tipe array

Bentuk umum:

Array [Indeks] of Tipe Data

b) Tipe file

Bentuk umum:

File of type Atau File

c) Tipe object

Tipe object adalah data berstruktur yang berisi komponen bilangan fixed.

Bentuk umum:

Object

Field;

Field;

.....

Method;

Method;

End;

d) *Tipe record*

Bentuk umum :

Record

Field;

Field;

.....

End;

e) *Tipe set*

Bentuk umum:

Set of Tipe Data

4. Tipe Pointer

Tipe pointer adalah tipe yang berisi alamat memori, dan berlambang ^. Anda dapat menunjuk sebuah nilai kedalam variable pointer dengan:

- Procedure New atau GetMem
- Operator @
- Fungsi Ptr

5. Tipe Procedural

Procedure dan Function adalah bagian Turbo Pascal/Delphi dalam membuat sebuah program. Melalui tipe Procedural, maka anda dapat memperlakukan Procedure dan Function sebagai object sehingga dapat dimasukkan kedalam sebuah variable dan parameter. Hasil function haruslah berupa string, real, integer, char, Boolean, atau pointer.

g. Label

Label adalah suatu deklarasi untuk membuat percabangan dalam proram. Label bisa berupa huruf, misalnya: AWAL, AKHIR, atau angka antara 0 and 999. Dan untuk menuju ke label yang telah dideklarasikan harus menggunakan instruksi GOTO.

Bentuk umum:

Label pengenal,.... pengenal;

h. Operator

Operator adalah lambing-lambing untuk melakukan perkalian, penjumlahan dan lain- lain seperti dalam kalkulator. Tetapi operator dalam computer lebih kompleks dibandingkan kalkulator. Jenis-jenis operator:

- * Operator penghubung (relational operators)
- * Operator arithmatik (arithmetic operators)
- * Operator logika (logical operators)
- * Operator pembanding (Boolean operators)
- * Operator string (string operators)
- * Operator set (set operators)
- * Operator @ (@ operators)
- * Operator Pchar (Pchar operators)

4.2 Struktur Pemrograman Pascal/Delphi

Struktur dasar dalam pemrograman pascal/Delphi :

Elemen-elemen dalam program harus sesuai dengan urutannya, beberapa elemen bisa dihilangkan bila tidak diperlukan. Seperti contoh dibawah, program yang ada merupakan program yang benar, tapi tidak melakukan apapun.

Komentar dapat disertakan dalam penulisan kode. Komentar tidak akan disertakan dalam kompilasi (compile) atau saat program dijalankan (execute). Penanda komentar adalah (*) dan diakhiri dengan (*). atau dapat pula dengan tanda { dengan akhiran }. Pemakaian komentar dalam bahasa pascal tidak boleh salah, karena akan menimbulkan masalah. Penulisan komentar yang salah seperti : {{ disini komentar }}

Pada saat code dicompile, akan memberikan pesan kesalahan karena compiler akan melihat tanda { yang pertama dan tanda } yang pertama pula, sehingga tanda } yang kedua akan dianggap kesalahan. Berikut beberapa contoh penulisan komentar yang benar :

```
PROGRAM NamaProgram (FileList);
CONST
(*Deklarasi Konstanta*)
TYPE
(*Deklarasi Type*)
VAR
(*Deklarasi Variabel*)
(*Definisi SubProgram*)
BEGIN
```

```
{ { disini komentar }
{ disini komentar } writeln('test komentar'); { komentar lagi }
(* disini komentar *)
```

Pemberian komentar akan mempermudah dalam memahami suatu kode program (source code). Bila kita menulis program tanpa memberikan komentar, saat kita membuka kembali kode yang kita tulis dalam jangka waktu berselang lama. Akan mempersulit kita memahami program yang kita buat sebelumnya (bila program sangat rumit).

```
PROGRAM
informatika;
BEGIN
```

Diketahui : Data input = Nama Mhs, Nama_MK, UTS, UAS

Data output = Nama Mhs, Nama, Nilai ujian (NA).

Rumus : NA = (UTS + UAS) / 2

Ditanya : Buatlah program sederhana untuk menghitung nilai ujian

Pembahasan :

```
Program Hitung_Nilai_Ujian; { Nama program)
Uses crt;
Var { Deklarasi variable }
Nama_Mhs: String [1..10];
Nama_MK: String;
UTS, UAS, NA: Integer;
Begin { Program utama}
Clrscr;
{Proses input data}
Write (' input nama mahasiswa='); Readln (Nama_Mhs);
Write (' input nama mata kuliah='); Readln (Nama_MK);
Write (' input nilai UTS='); Readln (UTS);
Write (' input nilai UAS='); Readln (UAS);
{Proses perhitungan}
NA:= (UTS/UAS)/2;
{Proses Output}
Writeln ('Nama mahasiswa= ,', Nama_Mhs: 10);
Writeln('Nama mata kuliah= ', Nama_MK);
Writeln('Nilai akhir = ', NA:4);
End. { Akhir Program}
```

BAB V

Struktur Dasar Algoritma

5.1 Pengantar

Pada dasarnya, algoritma merupakan deskripsi langkah-langkah pelaksanaan suatu proses. Setiap langkah penyelesaiannya disebut dengan pernyataan. Sebuah pernyataan menggambarkan aksi (action) algoritmik yang dapat dieksekusi. Efek dari penggerjaan suatu aksi dapat dilihat dengan membandingkan keadaan awal saat aksi belum dijalankan dan keadaan akhir setelah aksi dijalankan.

5.2 Struktur Dasar Algoritma

Dalam sebuah algoritma langkah-langkah penyelesaian masalahnya dapat berupa struktur urut (sequence), struktur pemilihan (selection), dan struktur pengulangan (repetition). Ketiga jenis langkah tersebut membentuk konstruktif suatu algoritma. Sebuah algoritma dapat dibangun dari tiga buah struktur dasar, yaitu :

- » *Runtungan (Sequence)*
- » *Pemilihan (Selection)*
- » *Pengulangan (Repetition)*

Runtunan (Sequence)

Sebuah runtunan terdiri dari satu atau lebih pernyataan. Setiap pernyataan akan dieksekusi secara berurutan sesuai dengan urutan penulisannya, yakni sebuah instruksi (pernyataan) akan dijalankan setelah instruksi sebelumnya selesai dijalankan. Sebagai contoh, runtunan dapat ditemui pada algoritma Tukar_isi_bejana berikut ini.

Langkah 1
Tuangkan isi bejana A ke dalam bejana C
↓
Langkah 2
Tuangkan isi bejana B ke dalam bejana A
↓
Langkah 3
Tuangkan isi bejana C ke dalam bejana B

Pemilihan (Sequence)

Adakalanya sebuah aksi dilakukan setelah kondisi tertentu terpenuhi. Misalnya, sebuah kendaraan berada di perempatan traffic light. Jika lampu traffic light berwarna merah, maka kendaraan tersebut harus berhenti. Keadaan ini dapat ditulis melalui pernyataan berikut :

Jika nilai ujian > 50 , maka
“Keterangan = D”

Pernyataan di atas dapat diubah dalam bentuk pernyataan pemilihan (selection statement) atau disebut juga pernyataan kondisional sebagai berikut :

if kondisi then
aksi

Maka, untuk contoh di atas dapat ditulis :

if nilai ujian > 50 then
“Keterangan=D”

Struktur pemilihan if-then hanya memberikan satu pilihan aksi bila kondisi (persyaratan) terpenuhi. Bentuk pemilihan lain yang sering digunakan adalah pemilihan satu dari dua buah aksi sesuai dengan kondisinya. Bentuk strukturnya adalah :

if kondisi then
aksi1
else
aksi2

Sebagai contoh, akan ditentukan bilangan terbesar diantara dua buah bilangan bulat x dan y. Maka, bentuk pemilihannya adalah :

if x > y then
 tulis x sebagai bilangan terbesar
else
 tulis y sebagai bilangan terbesar

Jika pilihan aksi yang dilakukan lebih dari dua, maka digunakan struktur pemilihan bersarang (nested if). Contohnya :

```
if lampu traf light berwarna merah then berhenti  
else  
if lampu traf light berwarna kuning then jalan hati-hati  
else  
    jalan terus
```

Contoh berikutnya dari pemilihan bersarang adalah menentukan bilangan terbesar dari 3 (tiga) buah bilangan x, y, z, yaitu :

```
if x > y then  
    if x > z then  
        tulis x sebagai bilangan terbesar  
    else  
        tulis z sebagai bilangan terbesar  
    else  
        if y > z then  
            tulis y sebagai bilangan terbesar  
        else  
            tulis z sebagai bilangan terbesar
```

Pengulangan (Repetition)

Jika pada suatu saat kita harus membuat teks yang berulang (dilakukan lebih dari 1 kali) dalam sebuah algoritma, maka dapat digunakan struktur pengulangan (repetition). Misalnya, akan dibuat teks "Teknik Informatika" sebanyak 10 kali. Maka algoritmanya dapat dibuat dengan menggunakan bentuk struktur pengulangan yaitu fordo sebagai berikut :

```
for i dari 1 sampai dengan 10 do  
    tulis Teknik Informatika  
end for
```

i merupakan pencacah pengulangan yang akan mencacah pengulangan dari 1 sampai dengan 10. Komputer akan melakukan pengulangan sebanyak pencacahan. Secara umum, struktur pengulangan tersebut dapat dibentuk sebagai berikut :

```
for pencacah pengulangan dari a sampai b do  
    aksi  
end for
```

artinya, aksi dilakukan sebanyak hitungan pencacah pengulangan yaitu dari a sampai b sebanyak $b - a + 1$ kali. Struktur pengulangan yang kedua adalah repeat-until. Repeat artinya “ulangi”, sedangkan until artinya “sampai” atau “hingga”. Bentuk umumnya adalah sebagai berikut :

```
repeat
    aksi
    until kondisi
```

artinya, aksi akan diulang hingga atau sampai kondisi (persyaratan) terpenuhi. Sebagai contoh, dari sejumlah data mahasiswa yang mengambil mata kuliah Algoritma Pemrograman I, akan dilakukan perubahan data alamat mahasiswa dengan NIM 006. Maka, bila ditelusuri algoritma yang mungkin adalah sebagai berikut :

```
lihat data mahasiswa pada posisi pertama
if data mahasiswa pertama memiliki NIM 006 then
    ubah data alamatnya
else
    lihat data mahasiswa pada posisi kedua
    if data mahasiswa kedua memiliki NIM 006 then
        ubah data alamatnya
    else
        ...
        ... dan seterusnya
```

Dari algoritma di atas, terlihat bahwa pengecekan (penyeleksian) apakah data mahasiswa pada posisi tertentu memiliki NIM 006 dilakukan secara terus menerus. Algoritma tidak memiliki kondisi kapan harus menghentikan proses pengecekan tersebut. Hal ini tentu saja tidak benar, mengingat sebuah algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas. Untuk kondisi seperti ini, maka dapat digunakan struktur pengulangan repeat-until. Bentuk algoritmanya adalah :

```
lihat data mahasiswa pada posisi pertama
repeat
    if data mahasiswa memiliki NIM 006 then
        ubah data alamatnya
    else
```

until lihat data mahasiswa pada posisi berikutnya
 data mahasiswa dengan NIM 006 telah ditemukan
 atau seluruh data mahasiswa telah diperiksa

Struktur pengulangan yang ketiga adalah while-do. While artinya “selama”, sedangkan do artinya “lakukan/ kerjakan”. Bentuk umumnya adalah sebagai berikut :

```
while kondisi do
    aksi
end while
```

artinya, selama kondisi (persyaratan) pengulangan masih terpenuhi (benar), maka aksi akan dilakukan. Perbedaannya dengan repeatuntil, jika pada repeat-until kondisi pengulangan akan dievaluasi (dicek) setelah aksi dikerjakan, sedangkan pada while-do kondisi pengulangan akan dicek sebelum aksi dikerjakan (di bagian awal pengulangan). Bentuk algoritma dengan menggunakan while-do adalah sebagai berikut.

```
lihat data mahasiswa pada posisi pertama
while data mahasiswa dengan NIM 006 belum ditemukan dan
data mahasiswa terakhir belum terlampaui do
    if data mahasiswa memiliki NIM 006 then
        ubah data alamatnya
    else
        lihat data mahasiswa pada posisi berikutnya
    end if
end while
```

5.3 Strategi Perancangan Puncak-Turun (Top-Down)

Tahap - tahap penyusunan sebuah algoritma seringkali dimulai dari langkah yang global (umum) terlebih dahulu. Langkah global ini kemudian diuraikan lagi hingga langkah yang lebih rinci.

Perancangan algoritma seperti ini dinamakan perancangan puncak-turun (top-down). Cara ini bermanfaat untuk membuat algoritma dari permasalahan yang rumit atau kompleks. Sebagai contoh, terdapat sejumlah data

(dimisalkan dengan N) dalam sebuah tabel yang akan diurutkan. Setiap data di dalam tabel disebut dengan elemen tabel. Pengurutan data akan dimulai dengan algoritma secara global (umum), yaitu sebagai berikut :

- * Cari nilai terbesar diantara N buah data
- * Tempatkan nilai terbesar tersebut pada posisi yang tepat
- * Ulangi dari langkah 1 untuk N-1 buah data yang lain

Pernyataan Cari nilai terbesar diantara N buah data masih terlalu global (umum). Algoritma tersebut tidak menyatakan bagaimana proses pencarian dilakukan. Karena itu, algoritma harus diuraikan lagi ke dalam langkah-langkah yang lebih rinci hingga pengurutan data dapat dilakukan. Untuk itu, langkah 1 akan diuraikan lebih rinci menjadi :

- * Asumsikan elemen pertama adalah nilai terbesar sementara (maks)

```
while belum mencapai elemen ke-N do
    tinjau elemen berikutnya
    if elemen ini lebih besar dari maks then
        ganti nilai maks dengan elemen ini
    end if
end while
```

Pernyataan Tempatkan nilai terbesar tersebut pada posisi yang tepat juga akan diuraikan lebih lanjut seperti berikut ini.

- * Tempatkan elemen ke-N ke dalam C
- * Tempatkan maks ke posisi elemen ke-N
- * Tempatkan elemen di dalam C ke posisi maks yang lama

Begitu pula dengan pernyataan ulangi dari langkah 1 untuk N- 1 buah data yang lain (langkah 3) akan diuraikan lagi menjadi langkah-langkah berikut.

- Kurangi N dengan 1
- Ulangi dari langkah 1.1

Secara keseluruhan algoritma pengurutan data di atas adalah :

- * Asumsikan elemen pertama adalah nilai terbesar sementara (maks)

```
while belum mencapai elemen ke-N do  
    tinjau elemen berikutnya  
    if elemen ini lebih besar dari maks then  
        ganti nilai maks dengan elemen ini  
    end if  
end while
```

- * Tempatkan elemen ke-N ke dalam C
- * Tempatkan maks ke posisi elemen ke-N
- * Tempatkan elemen di dalam C ke posisi maks yang lama
- * Kurangi N dengan 1
- * Ulangi dari langkah 1.1

Soal dan Pembahasan

Berikut ini diberikan beberapa contoh soal latihan dan pembahasannya :

1. Buatlah algoritma (dalam notasi kalimat deskriptif) untuk memperoleh informasi nomor telepon berdasarkan data alamat (berupa nama jalan dan nomornya) pada nomor penerangan lokal (108) PT. Telkom. Algoritma harus menjelaskan proses jika :
 - a. Nomor 108 sibuk
 - b. Alamat yang diberikan penelpon belum mempunyai sambungan telepon
2. Buatlah algoritma (dalam notasi kalimat deskriptif) untuk mengubah data alamat dan nomor telepon mahasiswa berdasarkan NIM.
3. Algoritma berikut membagikan sekantung permen secara adil kepada 3 orang anak dengan cara memberikan satu permen pada tiap anak secara berulang-ulang

Repeat

```
Berikan satu permen kepada anak pertama  
Berikan satu permen kepada anak kedua  
Berikan satu permen kepada anak ketiga
```

Until kantung permen kosong

Pada keadaan bagaimana algoritma tersebut gagal?

Pembahasan untuk soal-soal latihan di atas adalah :

1. Algoritma Mencari_nomor_telepon_ke_108

a. Hubungi nomor 108

b. Jika nomor 108 sibuk, maka algoritma berakhir. Jika tidak lanjut ke langkah c

c. Masukkan alamat yang dicari nomor teleponnya

d. Lihat data pertama di tabel pelanggan

e. While alamat yang dicari belum ditemukan dan data terakhir belum terlampaui do

If alamat ini sama dengan alamat yang dicari Then

Lihat data nomor teleponnya

If data nomor telepon tidak kosong Then

Berikan data nomor teleponnya

Else

Berikan informasi bahwa telepon belum terpasang

End if

Else

Lihat data di posisi berikutnya

End if

End while

f. If alamat yang dicari belum ditemukan Then Berikan informasi bahwa alamat yang dicari tidak ditemukan

End if

2. Algoritma Mengubah_alamat_dan_telepon_berd_NIM

a. Baca data NIM yang akan diubah data alamat dan nomor teleponnya

b. Lihat data pertama pada tabel mahasiswa

c. While data NIM yang dicari belum ditemukan dan data terakhir belum terlampaui do

If NIM ini sama dengan NIM yang dicari Then

Baca data alamat dan nomor telepon baru

Ubah data alamat dan nomor teleponnya

Else

Lihat data di posisi berikutnya

End if

End while

d. If data NIM yang dicari belum ditemukan Then

Berikan informasi bahwa data NIM tidak ditemukan

End if

BAB VI

Aturan Penulisan Teks Algoritma

6.1 Pengantar

Teks algoritma merupakan penjelasan atau deskripsi langkah-langkah dari penyelesaian masalah yang tersusun secara sistematis. Langkah-langkah tersebut tidak memiliki standar yang baku seperti pada bahasa pemrograman tetapi langkah-langkah tersebut mudah di mengerti oleh si pembacanya. Teks algoritma yang dibuat agar mudah di translasi ke bahasa pemrograman tertentu maka sebaiknya langkah-langkah dari teks algoritma yang dibuat berkoresponden dengan perintah-perintah bahasa pemrograman.

6.2 Susunan Teks Algoritma

Teks algoritma disusun oleh tiga bagian (blok): bagian kepala (header) algoritma, bagian deklarasi, dan bagian deskripsi algoritma. Setiap bagian disertai dengan komentar untuk memperjelas maksud teks yang dituliskan. Komentar adalah kalimat yang diapit oleh pasangan tanda kurung karawal ('{' dan '}').

■ Kepala / Judul Algoritma :

Bagian yang terdiri diri atas nama algoritma dan penjelasan tentang algoritma tersebut. Nama algoritma sebaiknya singkat. Untuk memisahkan antara kata dalam judul algoritma menggunakan tanda “_” bukanlah suatu keharusan. Anda dapat menuliskan LuasLingkaran atau Luas_Lingkaran. Tetapi sebaiknya anda tidak menggunakan spasi “ “ untuk memisahkan antara kata di dalam nama algoritma.

Contoh:

Judul

{ Komentar mengenai Algoritma seperti cara kerja program, Kondisi awal dan kondisi akhir dari algoritma }

* **Deklarasi**

Bagian untuk mendefenisikan semua nama yang dipakai dalam algoritma. Nama dapat berupa nama tetapan, nama peubah, nama tipe, nama prosedur dan nama fungsi.

* **Deskripsi**

Merupakan bagian inti dari suatu algoritma. Bagian ini bersisir uraian langkah-langkah penyelesaian masalah. Langkah-langkah ini dituliskan dengan notasi yang akan dijelaskan pada Bab berikutnya. Misalnya notasi “read” untuk menyatakan membaca data, notasi “write” untuk menyatakan menulis data dan sebagainya.

Contoh 1 :

Algoritma Luas_Kell_Lingkaran {judul algoritma} {menghitung luas dan keliling lingkaran untuk ukuran jari-jari tertentu. Algoritma menerima masukan jari-jari lingkaran, menghitung luas dan kelilingnya, dan mencetak luas lingkaran ke piranti keluaran <- ini spesifikasi algoritma}

Deklarasi : const phi = 3.14 {nilai ? }

R : real {jari-jari lingkaran}

Luas : real {luas lingkaran}

Keliling : real {keliling lingkaran}

Deskripsi : read (R)

Luas <- phi * R * R

Keliling <- 2 * phi * R

write(luas, keliling)

Contoh 2 :

Hit_5_faktorial {Judul dari algoritma}

Deklarasi {mengenalkan variabel “i” dan

“fak” bertipe bilangan bulat”}

i : integer

fak : integer

Deskripsi

```
i ← 0
fak ← 1
while (i<5)
    i ← i + 1
    fak ← fak * i
end while
Output(fak)
```

Soal dan Pembahasan

Buatlah notasi algoritmik untuk menghitung persamaan dibawah ini

$$\text{Fahrenheit} = \frac{9}{5} \times (C + 32)$$

$$\text{Reamur} = \frac{4}{5} \times (C + 32)$$

Dimana, temperatur derajat Celsius dibaca, hitung konversi suhu diatas, kemudian cetak Celsius, Fahrenheit, Reamur.

Algoritma Konversi_Suhu

{ Menghitung Fahrenheit dan Reamur }

Deklarasi

Celsius, Fahrenheit, Reamur : real

Deskripsi

Input (Celsius)

$$\text{Fahrenheit} = \frac{5}{9} * (\text{Celsius} + 32)$$

$$\text{Reamur} = \frac{4}{5} * (\text{Celsius} + 32)$$

Write (Celsius, Fahrenheit, Reamur)

BAB VII

Runtunan (Sequence)

Konsep Runtunan

Algoritma merupakan runtunan (sequence) satu atau lebih instruksi, yang berarti bahwa :

1. Tiap instruksi dikerjakan satu per satu
2. Tiap instruksi dilaksanakan tepat sekali, tidak ada instruksi yang diulang
3. Urutan instruksi yang dilaksanakan pemroses sama dengan urutan instruksi sebagaimana yang tertulis di dalam teks algoritmanya
4. Akhir dari instruksi terakhir merupakan akhir algoritma. Urutan instruksi di dalam algoritma adalah penting. Urutan instruksi menunjukkan urutan logik penyelesaian masalah. Bergantung pada masalahnya, urutan instruksi yang berbeda mungkin tidak ada pengaruhnya terhadap solusi persoalan, tetapi mungkin juga menghasilkan keluaran yang berbeda pula.

Contoh kasus 1 :

urutan instruksi tidak berpengaruh terhadap solusi persoalan.

Dibaca dua buah nilai integer dari piranti masukan, A dan B. hitung jumlah keduanya dan hasil kali keduanya, lalu cetak

jumlah dan hasil kali itu ke piranti keluaran.

Hasil algoritma tersebut sama saja jika urutan $C \leftarrow A+B$ dan $D \leftarrow A*B$ diubah sebagai berikut :

```
Algoritma RUNTUNAN_1
{ contoh algorima yang menghasilkan keluaran yang sam jika
urutan instruksi diubah. }

DEKLARASI
    A, B, C, D : integer

DESKRIPSI
    read(A,B)
    C=A+B
    D=A*B
    Write(C,D)
```

```

Algoritma RUNTUNAN_1
{ contoh algorima yang menghasilkan keluaran yang sam jika
urutan instruksi diubah. }

DEKLARASI
    A, B, C, D : integer

DESKRIPSI
    read(A,B)
    D:=A*B
    C:=A+B
    write(C,D)

```

Contoh kasus 2 :

urutan instruksi berpengaruh terhadap solusi persoalan. Diketahui dua buah nilai integer, masing-masing disimpan di dalam dua peubah, A dan B. bagaimana caramempertukarkan nilai A dan B ?

Misalnya, sebelum pertukaran nilai A=8, nilai B=5, maka setelah pertukaran, nilai A=5 dan B=8.

Proses pertukaran nilai akan salah jika anda tidak benar menuliskan urutan instruksi, misalnya :

```

temp:=A      { simpan nilai A di penampungan sementara,
temp }
        A:=B      { sekarang A dapat diisi dengan nilai B }
        B:=temp    { isi B dengan nilai A semula yang tadi
disimpan di temp }

Diubah urutannya sebagai berikut :

{ proses pertukaran }
        temp:=A      { simpan nilai A di penampungan sementara,
temp }
        B:=temp      { isi B dengan nilai A semula yang tadi
disimpan di temp }
        A:=B      { sekarang A dapat diisi dengan nilai B }

maka runtunan yang terakhir ini sama saja dengan runtunan :
        B:=A
        A:=B

```

```

Algoritma TUKAR_1
{ mempertukarkan nilai A dan B. Nilai A dan B dibaca dari piranti masukan.
Nilai A dan B dicetak ke piranti keluaran, baik sebelum pertukaran maupun
sesudah pertukaran. ALGORITMA YANG BENAR ! }

DEKLARASI
    A      : integer   { nilai pertama }
    B      : integer   { nilai kedua }
    Temp  : integer   { peubah bantu }

DESKRIPSI
    { baca nilai A dan B }
    Read(A,B)
    { cetak nilai A dan B sebelum pertukaran }
    write(A,B)
    { proses pertukaran }
    temp:=A      { simpan nilai A di penampungan sementara, temp }
    A:=B      { sekarang A dapat diisi dengan nilai B }
    B:=temp    { isi B dengan nilai A semula yang tadi disimpan di temp }
    { cetak nilai A dan B setelah pertukaran }
    Write(A,B)

```

Contoh Kasus 3 :

Dibaca waktu tempuh seorang pelari marathon dalam jam-menit-detik (hh:mm:ss). Diminta mengkonversi waktu tempuh tersebut ke dalam detik. Tuliskan algoritmanya.

Ingatlah

1 menit = 60 detik

1 jam = 3600 detik

Misalnya waktu tempuh seorang pelari marathon adalah 1 jam, 5 menit, 40 detik. Dalam detik, waktu tempuh seluruhnya adalah $(1 \times 3600) + (5 \times 60) + 40 = 3940$ detik.

Penyelesaian

```
Algoritma KONVERSI_JAM_KE_DETIK
{ dibaca jam-menit-detik (hh:mm:ss). Nilai jam-menit-detik
dikonversi ke dalam detik, lalu ditampilkan ke piranti
keluaran }

DEKLARASI
Type jam : record <hh : integer {0..23},      {jam}
                           mm : integer {0..59},    {menit}
                           ss : integer {0..59},    {detik}
                           >
J : jam
TotalDetik : integer

DESKRIPSI
read(J.hh,J.mm,J.ss)
TotalDetik - (J.hh*3600) + (J.mm*60) + J.ss
write(TotalDetik)
```

Jika anda mentranslasikan algoritma KONVERSI_JAM_KE_DETIK ke dalam bahasa pascal, anda harus memperhatikan tipe bilangan bulat yang digunakan. Karena ranah nilai tipe integer terbatas, maka ada kemungkinan hasil pengubahan jam-menit-detik ke total detik bernilai negatif, sebab nilai $(J.hh*3600) + (J.mm*60) + J.ss$ berada di luar rentang tipe integer. Tipe longint yang mempunyai ranah yang lebih besar dapat dipakai untuk masalah ini.

Jadi, program KONVERSI_JAM_KE_DETIK dalam bahasa pascal adalah sebagai berikut :

```
program KONVERSI_JAM_KE_DETIK;
{ dibaca jam-menit-detik (hh:mm:ss). Nilai jam-menit-detik dikonversi ke dalam detik, lalu ditampilkan ke piranti keluaran.}

uses wincrt;

(* DEKLARASI *)
type Jam = record
    hh : longint;      {jam}
    mm : longint;      {menit}
    ss : longint;      {detik}
end;

var
    J : Jam;
    TotalDetik : longint;

(* deskripsi *)
begin
    write('Jam :'); readln(J.hh);
    write('Menit:'); readln(J.mm);
    write('Detik:'); readln(J.ss);
    TotalDetik := (J.hh*3600) + (J.mm*60) + J.ss;
    writeln('Total detik = ', TotalDetik);
end.
```

Soal Dan Pembahasan:

Dibaca nama karyawan dan gaji pokok bulanannya. Gaji bersih yang diterima pegawai adalah: **gaji bersih = gaji pokok + tunjangan - pajak**

Tunjangan karyawan dihitung 20% dari gaji pokok, sedangkan pajak adalah 15% dari gaji pokok ditambah tunjangan. Nama karyawan dan gaji bersihnya dicetak ke piranti keluaran. Tuliskan algoritmanya.

Jawab:

$$\text{Tunjangan} = 0.2 * \text{gaji pokok}$$

$$\text{Pajak} = 0.15 * (\text{gaji pokok} + \text{tunjangan})$$

$$\text{Gaji bersih} = \text{gaji pokok} + \text{tunjangan} - \text{pajak}$$

Dari ketentuan di atas kita dapat tuliskan algoritma sebagai berikut:

Penyelesaian

Algoritma Gaji_Karyawan

DEKLARASI

Nama : string
GaPok, Tunjangan, GaBer : real

DESKRIPSI

```
Read (Nama, GaPok)
Tunjangan ← 0.2 * GaPok
Pajak ← 0.15 * (GaPok + Tunjangan)
Gaber ← GaPok + Tunjangan - Pajak
Write(nama, Gaber)
```

Program Mengitung_Gaji_Karyawan;

Var

 Nama : **string[25];**
 Gapok, tunjangan, gaber : **real;**

Begin

```
    Write('Masukkan Nama Karyawan : '); Readln(nama);
    Write('Masukkan Gaji Pokok : '); Readln (gapok);
    Tunjangan := 0.2 * gapok;
    Pajak := 0.15 (gapok + tunjangan);
    Gaber := gapok + tunjangan - pajak;
    Writeln('Gaji Bersih : ', gaber:7:2);
```

End.

BAB VIII

Pemilihan (Condition)

8.1. Pengantar

Struktur runtunannya hanya terdapat pada program sederhana. Pada umumnya, masalah yang akan diselesaikan memiliki beberapa alternative pelaksanaan aksi. Suatu aksi hanya dilakukan bila persyaratan atau kondisi tertentu dipenuhi. Kita katakan bahwa masalah tersebut memiliki beberapa kasus. Jadi, dalam memecahkan masalah, kita harus menganalisis kasus-kasus apa saja yang mungkin ada, lalu aksi apa yang dilakukan bila suatu kasus dimasuki. Adanya pemilihan kasus-kasus menyebabkan terjadinya pemilihan instruksi di dalam algoritma, bergantung pada kasus yang memenuhi. Menganalisis kasus dari suatu masalah adalah menentukan kondisi boolean (bernilai true atau false) untuk setiap kasus dan menentukan aksi yang dilakukan jika kondisi tersebut berlaku (memenuhi).

Kondisi boolean adalah ekspresi boolean yang bernilai true atau false bergantung pada nilai masing-masing operand yang terlibat di dalamnya. Ekspresi boolean dibentuk dengan mengkombinasikan operand yang bertipe sama dengan salah satu dari operator relasional : $=$, \neq , $<$, $>$, \leq , \geq , dan operator uner not. Contoh-contoh ekspresi boolean :

$x > y$	ketemu = true
$a \neq 10$	not berhenti
$m = n$	$(x > 0)$ and $(y < 0)$
$p \leq q$	
$a + b > 1$	
str = 'itb'	
$k \bmod 4 = 0$	

Aksi yang dikerjakan bila kondisi boolean dipenuhi dapat berupa pengisian nilai (assignment), kalkulasi, baca, tulis, dan sebagainya, bergantung pada masalahnya. Penentuan

kondisi boolean dan aksi yang dilakukan bergantung pada jumlah kasus yang terdapat pada masalah tersebut : satu kasus, dua kasus, atau lebih dari dua kasus.

8.2 Struktur If- Then dan If- Then-Else

8.2.1. Satu Kasus

Notasi algoritmik untuk analisis dengan satu kasus adalah dengan menggunakan struktur IF-THEN (jika-maka) :

```
if kondisi then  
    aksi  
endif
```

Aksi sesudah kata then (dapat berupa satu atau lebih aksi) hanya akan dilaksanakan bila kondisi bernilai benar (true). Bila kondisi bernilai salah (false), tidak ada aksi apapun yang dikerjakan. Kata endif sengaja ditambahkan untuk mempertegas awal dan akhir struktur if-then. Contoh-contoh :

a. if $x > 100$ then

```
    x x+1
```

```
endif
```

b. if ada=false then

```
    read (cc)
```

```
    write (cc)
```

```
endif
```

catatan :

contoh (b) dapat juga ditulis sebagai berikut :

```
if not ada then  
    read(cc)  
    write(cc)  
endif
```

Ingatlah bahwa aksi sesudah kata then akan dikerjakan hanya jika kondisi bernilai true (dalam hal ini, not ada bernilai true bila ada = false).

Contoh analisis :

Dibaca sebuah karakter. Diminta menuliskan pesan 'huruf hidup' jika karakter tersebut merupakan huruf vokal.

Penyelesaian :

```
| Algoritma HURUF_VOKAL
{ mencetak pesan "huruf vokal" bila sebuah karakter yang dibaca merupakan huruf hidup, asumsikan karakter yang dibaca adalah huruf kecil }

DEKLARASI
    c : char

DESKRIPSI
    read(c)
    if (c='a') or (c='i') or (c='u') or (c='e') or (c='o') then
        write('huruf hidup')
    endif
```

Jadi, program HURUF_VOKAL dalam bahasa pascal adalah sebagai berikut :

```
| program HURUF_VOKAL;
| uses wincrt;
|
| (* DEKLARASI *)
| var
|     c : char;
|
| (* DESKRIPSI *)
| begin
|     writeln('masukkan huruf');read(c);
|     if (c='a') or (c='i') or (c='u') or (c='e') or (c='o') then
|         write('huruf hidup')
|     end.
| end.
```

8.2.2. Dua Kasus

Notasi algoritma untuk analisis dengan dua buah kasus adalah dengan menggunakan struktur IF-THEN-ELSE (jika-maka-kalau tidak) :

```
if kondisi then  
    aksi1  
else  
    aksi2  
endif
```

Aksi1 akan dilaksanakan jika kondisi bernilai benar, tetapi jika kondisi bernilai salah, maka aksi2 yang akan dilaksanakan. Perhatikanlah bahwa "else" menyatakan ingkaran (negation) dari kondisi.

Contoh analisis :

Buatlah algoritma dan program yang membaca angka tahun masehi dari papan kunci, lalu menentukan apakah tahun tersebut merupakan tahun kabisat. Secara sederhana, tahun kabisat adalah tahun yang habis dibagi dengan 4. Pada tahun kabisat, bulan februari berjumlah 29 hari. Contoh tahun kabisat adalah 1996 dan 2000. Tahun 2002 bukan tahun kabisat karena tidak habis dibagi 4.

Penyelesaian :

Misalkan tahun masehi tersebut adalah Tahun.

Analisis kasus :

Kasus 1 : Tahun mod 4 = 0, maka tulis Tahun adalah tahun kabisat

Kasus 2 : Tahun mod 4 ≠ 0, maka tulis Tahun bukan tahun kabisat

```

Algoritma TAHUN_KABISAT
{ menentukan apakah suatu tahun merupakan tahun kabisat atau
bukan kabisat }

DEKLARASI
    Tahun : integer

DESKRIPSI
    read(Tahun)
    if Tahun mod 4 = 0 then
        write(Tahun, ' adalah tahun kabisat')
    else
        write(Tahun, ' bukan tahun kabisat')
    endif

```

Jadi, program TAHUN_KABISAT dalam bahasa pascal adalah sebagai berikut :

```

program TAHUN_KABISAT;
(* menentukan apakah suatu tahun merupakan tahun kabisat atau
bukan kabisat *)

uses wincrt;
(* DEKLARASI *)
var
    Tahun : integer;

(* DESKRIPSI *)
begin
    write('TAHUN ');read(Tahun);
    if Tahun mod 4 = 0 then
        write(' adalah tahun kabisat')
    else
        write(' bukan tahun kabisat')
    end.
end.

```

8.2.3. Tiga Kasus atau Lebih

Masalah yang mempunyai tiga buah kasus atau lebih tetap dapat dianalisis dengan struktur IF-THEN-ELSE sebagaimana halnya pada masalah dengan dua kasus.

Tiga Kasus :

```
if kondisi1 then  
    aksi1  
else  
    if kondisi2 then  
        aksi2  
    else  
        if kondisi3  
            aksi3  
        endif  
    endif
```

Empat Kasus :

```
if kondisi1 then  
    aksi1  
else  
    if kondisi2 then  
        aksi2  
    else  
        if kondisi3 then  
            aksi3  
        else  
            if kondisi 4  
                aksi4  
            endif  
        endif  
    endif
```

dan seterusnya.

Contoh analisis :

Buatlah algoritma dan program yang membaca temperatur air, T, (dalam suatu derajat celcius) pada tekanan normal, lalu menentukan apakah wujud air tersebut dalam keadaan padat ($T \leq 0^{\circ}\text{C}$), cair ($0 < T < 100$), atau gas ($T > 100$).

Penyelesaian :

Misalkan suhu air adalah T.

Analisis kasus :

Kasus 1 : $T \leq 0$, maka tulis “padat”

Kasus 2 : $0 < T < 100$, maka tulis “cair”

Kasus 3 : $T \geq 100$, maka tulis “uap”

```

|-----+
| Algoritma WUJUD_AIR
| {menentukan wujud air : padat, cair, atau gas, bergantung pada
| suhunya }

DEKLARASI
    T : real      { suhu air, dalam derajat celcius }

DESKRIPSI
    read(T)
    if T ≤ 0 then
        write('padat')                      { kasus 1 }
    else
        if ( T > 0 ) and ( T < 100 ) then
            write('cair')                   { kasus 2 }
        else
            if T ≥ 100 then
                write('gas atau uap');       { kasus 3 }
            endif
        endif
    endif
|-----+

```

Jadi, program WUJUD_AIR dalam penulisan IF-THEN-ELSE yang bertingkat-tingkat.

```

program WUJUD_AIR;
{ menentukan wujud air : padat, cair, atau gas, bergantung pada suhunya }

uses wincrt;

(* DEKLARASI *)
var
T : real;      { suhu air, dalam derajat celcius }

(* DESKRIPSI *)
begin
write('suhu ');read(T);
write('adalah ');
if T <= 0 then
write('padat')                      { kasus 1 }
else
if ( T > 0 ) and ( T < 100 ) then
write('cair')                   { kasus 2 }
else
if T ≥ 100 then
write('gas atau uap');       { kasus 3 }
end.
end.

```

8.3 Struktur If- Then dan If- Then-Else

Tidak semua bahasa pemrograman menyediakan struktur CASE (misalnya Bahasa Fortran). Bahasa pascal menyediakan struktur ini. Jika bahasa pemrograman tidak yang ekivalen.

Contoh analisis :

Buatlah algoritma dan program yang membaca angka bulan dan tahun, lalu menuliskan jumlah hari dalam bulan tersebut. Misalnya jika dibaca bulan 8 (agustus), maka jumlah harinya adalah 31.

Penyelesaian :

Kita harus mengidentifikasi bulan-bulan dan jumlah harinya sebagai berikut :

Bulan	Jumlah hari
1, 3, 5, 7, 8, 10,	31
12	
4, 6, 9, 11	30
2	29 (jika tahun kabisat), 28 (jika bukan kabisat)

```
Algoritma JUMLAH_HARI
{ menentukan jumlah hari dalam satu bulan }

DEKLARASI
    AngkaBulan : integer          { 1 . . 12 }
    Tahun       : integer          { > 0 }
    JumlahHari : integer

DESKRIPSI
    read(AngkaBulan,Tahun)
    case(AngkaBulan)
        AngkaBulan= [1, 3, 5, 7, 8, 10, 12] : JumlahHari=31
        AngkaBulan= [4, 6, 9, 11]           : JumlahHari=30
        AngkaBulan= 2 : case Tahun
            Tahun mod 4 = 0      : JumlahHari=29
            Tahun mod 4 ≠ 0      : JumlahHari=28
        endcase
    endcase
    write(JumlahHari)
```

Jadi, program JUMLAH_HARI dalam bahasa pascal adalah sebagai berikut :

```
program JUMLAH_HARI;
{ menentukan jumlah hari dalam satu bulan }

uses wincrt;

(* DEKLARASI *)
var
    AngkaBulan: integer;          { 1 . . 12 }
    Tahun      : integer;          { > 0 }
    JumlahHari: integer;

(* DESKRIPSI *)
begin
    write('Bulan (1-12) = ');readln(AngkaBulan);
    write('Tahun = ');readln(Tahun);
    case AngkaBulan of
        1, 3, 5, 7, 8, 10, 12 : JumlahHari:=31;
        4, 6, 9, 11           : JumlahHari:=30;
        2                      : if Tahun mod 4 = 0 then
                                    JumlahHari:=29
                                else
                                    JumlahHari:=28;
    {endif}
    end;
    writeln('Jumlah hari dalam bulan ',AngkaBulan,' adalah ',JumlahHari);
end.
```


BAB IX

Pengulangan (Looping Program)

9.1. Pengantar

Struktur pengulangan secara umum terdiri atas dua bagian :

- kondisi pengulangan, yaitu ekspresi Boolean yang harus dipenuhi untuk melaksanakan pengulangan. Kondisi ini ada yang dinyatakan secara eksplisit oleh pemrogram atau dikelola sendiri oleh komputer (implisit);
- badan (body) pengulangan, yaitu bagian algoritma yang diulang.

Disamping itu, struktur pengulangan biasanya disertai dengan bagian:

- Inisialisasi, yaitu aksi yang dilakukan sebelum pengulangan dilakukan pertama kali
- Terminasi, yaitu aksi yang dilakukan setelah pengulangan selesai dilaksanakan Inisialisasi dan terminasi tidak selalu harus ada, namun pada berbagai kasus inisialisasi umumnya diperlukan.

Struktur pengulangan secara umum :

yang dalam hal ini awal dan akhir pengulangan dinyatakan sebagai kata kunci yang bergantung pada struktur pengulangan yang digunakan. Selain itu, <inisialisasi> dan <terminasi> adalah bagian yang opsional.

Di dalam algoritma terdapat beberapa macam struktur pengulangan yang berbeda. Beberapa struktur dapat dipakai untuk masalah yang sama, namun ada notasi pengulangan yang hanya cocok dipakai untuk masalah tertentu. Pemilihan struktur pengulangan untuk masalah tertentu dapat mempengaruhi kebenaran algoritma. Pemilihan struktur pengulangan yang tepat bergantung pada masalah yang akan deprogram. Ada tiga (3) macam notasi struktur pengulangan, yaitu :

- Struktur FOR
- Struktur WHILE
- Struktur REPEAT

9.2. Struktur FOR

Struktur pengulangan FOR digunakan untuk menghasilkan pengulangan sejumlah kali yang dispesifikasikan. Jumlah pengulangan diketahui atau dapat ditentukan sebelum eksekusi. Untuk mengetahui sudah berapa kali pengulangan dilakukan, kita memerlukan sebuah peubah (variable) pencacah (counter). Perubah ini nilainya selalu bertambah satu setiap kali pengulangan dilakukan.

Jika cacah pengulangan sudah mencapai jumlah yang dispesifikasikan, maka proses pengulangan berhenti. Bentuk umum struktur FOR ada dua macam : menaik (ascending) atau menurun (descending).

FOR menaik :

```
for pencacah=nilai_awal to nilai_akhir do  
    aksi  
endfor
```

Keterangan :

- i. pencacah haruslah dari tipe data yang memiliki predecessor dan successor, yaitu integer atau karakter. Tipe riil tidak dapat digunakan sebagai pencacah.
- ii. Aksi adalah satu atau lebih instruksi yang diulang.
- iii. nilai_awal harus lebih kecil atau sama dengan nilai_akhir.
Jika nilai_awal lebih besar dari nilai_akhir, maka badan pengulangan tidak dimasuki.
- iv. Pada awalnya,pencacah diinisialisasi dengan nilai_awal. Nilai pencacah secara otomatis bertambah satu setiap kali pengulangan dimasuki, sampai akhirnya nilai pencacah sama dengan nilai_akhir.
- v. Jumlah pengulangan yang terjadi adalah nilai_akhir - nilai_awal + 1.

Contoh analisis :

Buatlah algoritma dan program mencetak angka 1, 2, .., N, yang dalam hal ini nilai N dibaca terlebih dahulu dari piranti masukan.

Jadi, program CETAK_N_ANGKA dalam bahasa pascal/delphi adalah sebagai berikut :

Hasil Output program :

```
program CETAK_N_ANGKA;
{ mencetak 1, 2, ..., N ke piranti keluaran }

uses wincrt;

(* DEKLARASI *)
var
    N : integer;
    k : integer;

(* DESKRIPSI *)
begin
    write('jumlah pengulangan = ');read(N);
    for k:=1 to N do           { ulangi sebanyak N kali }
        writeln(k);
    end.
end.
```

Pertanyaan :

Apa yang terjadi bila N=0? N=-1? N=1?

Jawab :

Jika N = 0 atau N = -1, proses pengulangan tidak terjadi, karena nilai akhir pencacahan pengulangan lebih besar dari nilai awalnya (1). Jika N = 1, pengulangan yang terjadi adalah 1 kali, karena $1 - 1 + 1 = 1$.

For menurun :

```
for pencacahan=nilai_akhir downto nilai_awal do
    aksi
endfor
```

Keterangan :

- i. Pencacahan haruslah dari tipe data yang memiliki predecessor dan successor, yaitu integer atau karakter. Tipe riil tidak dapat digunakan sebagai pencacahan.
- ii. Aksi adalah satu atau lebih instruksi yang diulang.
- iii. nilai_akhir harus lebih besar atau sama dengan nilai_awal.

- Jika nilai_akhir lebih kecil dari nilai_awal, maka badan pengulangan tidak dimasuki.
- iv. Pada awalnya, pencacahan diinisialisasi dengan nilai_akhir. Nilai pencacahan secara otomatis berkurang satu setiap kali aksi diulang, sampai akhirnya nilai pencacahan sama dengan nilai_awal.
 - v. Jumlah pengulangan yang terjadi adalah nilai_awal-nilai_akhir+1.

Contoh analisis :

Algoritma dan program peluncuran roket dengan hitung mundur, mulai dari 10, 9, 8, ..., 0.

```

Algoritma PELUNCURAN_ROKET
{ Hitung mundur peluncuran roket }

DEKLARASI
    k : integer

DESKRIPSI
    for k:=100 downto 0 do
        write(k)
    endfor
    write('Go!') {roket meluncur }
  
```

```

program PELUNCURAN_ROKET;
{ hitung mundur peluncuran roket }
uses wincrt;

(* DEKLARASI *)
var
    k : integer;

(* DESKRIPSI *)
begin
    for k:=100 downto 0 do
        writeln(k);
    write('Go!!!!'); { roket meluncur }
end.
  
```

9.3. Struktur WHILE

Bentuk umum :

```

while kondisi do
    aksi
endwhile
  
```

Aksi (atau runtunan aksi) akan dilaksanakan berulangkali selama kondisi benilai true. Jika kondisi bernilai false, badan pengulangan tidak akan dilaksanakan, yang berarti pengulangan selesai. Yang harus diperhatikan adalah pengulangan harus berhenti. Pengulangan yang tidak pernah berhenti menandakan bahwa logika algoritma tersebut salah. Pengulangan berhenti apabila kondisi bernilai false. Agar kondisi suatu saat bernilai false, maka di dalam badan pengulangan harus ada instruksi yang mengubah nilai peubah kondisi.

Contoh analisa :

Dibuat sejumlah data bilangan bulat positif dari piranti masukan. Banyaknya data tidak diketahui sebelumnya, tetapi akhir

pemasukan data adalah bila data yang dimasukkan bernilai -99. Bilangan -99 akan diinterpretasikan sebagai tanda berhenti proses pengisian data.

Kita diminta menghitung jumlah seluruh nilai yang dimaskkan (-99 tidak termasuk data yang dijumlahkan).

Sebagai ilustrasi :

- misalkan dibaca berturut-turut data: 10, 4, 5, 8, -99, maka jumlah seluruh nilai adalah $10 + 4 + 5 + 8 = 27$
- misalkan dibaca berturut-turut data : 9, -99, maka jumlah seluruh nilai adalah 9
- misalkan dibaca berturut-turut data : -99, maka jumlah seluruh nilai adalah 0

9.4. Struktur REPEAT

Bentuk umum :

```
repeat
    aksi
until kondisi
```

Notasi ini mendasarkan pengulangan pada kondisi boolean. Aksi di dalam badan kalang diulang samopai kondisi boolean bernilai true. Dengan kata lain, jika kondisi boolean masih false, pengulangan masih terus dilakukan. Karena proses pengulangan suatu saat harus berhenti, maka di dalam badan pengulangan harus ada aksi yang mengubah nilai peubah kondisi.

Struktur REPEAT memiliki makna yang sama dengan WHILE, dan dalam beberapa masalah kedua struktur tersebut komplemen satu sama lain.

Contoh analisa :

Algoritma dan program untuk menghitung jumlah angka dari 1 sampai N. Nilai N dibaca dari papan kunci. Misalnya N = 5, maka $1 + 2 + 3 + 4 + 5 = 15$.

```

Algoritma PENJUMLAHAN_DERET;
{ menjumlahkan deret
  1 + 2 + 3 + ... + N
dengan N adalah bilangan bulat positif yang dibaca dari piranti
masukan. jumlah deret dicetak ke piranti keluaran. }

DEKLARASI
  N, k, jumlah : integer;

DESKRIPSI
  read(N)      {banyaknya suku deret}
  jumlah:=0    { inisialisasi jumlah deret }
  k:=1         { suku deret yang pertama }
  repeat
    jumlah:=jumlah + k          {jumlah deret sekarang}
    k:=k+1                      {suku deret berikutnya}
  until k > N

```

```

program PENJUMLAHAN_DERET;
{ menjumlahkan deret
  1 + 2 + 3 + ... + N
dengan N adalah bilangan bulat positif yang dibaca dari piranti
masukan. jumlah deret dicetak ke piranti keluaran. }

uses wincrt;

(* DEKLARASI *)
var
  N, k, jumlah : integer;

(* DESKRIPSI *)
begin
  write('N = ');readln(N);  {banyaknya suku deret}
  jumlah:=0; { inisialisasi jumlah deret }
  k:=1;       { suku deret }
  repeat
    jumlah:=jumlah + k;        {jumlah deret sekarang}
    k:=k+1;                   {suku deret berikutnya}
  until k > N;

  writeln('jumlah deret = ', jumlah);
end.

```

10.1. Pengantar

Sebuah program yang baik adalah program yang membagi permasalahan utama menjadi bagian-bagian kecil dimana setiap bagian kecil ditangani oleh sebuah subprogram, cara ini disebut dengan modular programming (pemrograman terbagi/terpecah). Cara ini termasuk pemrograman terstruktur dan sangat didukung oleh bahasa Pascal maupun Delphi. Untuk itu, Pascal/Delphi telah menyediakan dua jenis subprogram, yaitu procedure dan function (prosedur dan fungsi).

Sebuah sistem program yang besar dibangun oleh ribuan baris bahkan sampai jutaan baris perintah. Jika baris perintah ini dibuat dalam sebuah program yang utuh, maka akan muncul beberapa kendala, seperti sulit membaca logika program, sulit mencari /memperbaiki kesalahan bahkan mungkin ada banyak perintah yang ditulis berulang-ulang. Hingga berakibat program menjadi lambat, bahkan tidak dapat berjalan ; disebabkan oleh memori yang tersedia terbatas.

Untuk mengatasinya, maka program besar tersebut harus dipecah-pecah menjadi beberapa bagian sub-program kecil berupa modul-modul yang terpisah dari program utama. Sub-program/modul ini dikendalikan oleh program utama, inilah yang disebut dengan pemrograman modular.

Dalam Pemrograman menggunakan Bahasa Pascal atau Delphi, kita dapat menyederhanakan program yang kompleks menjadi potongan-potongan program yang lebih sederhana. Dimana potongan-potongan program tersebut disebut dengan modul/subrutin yang dapat menjalankan suatu tugas tertentu.

Dengan **modular programming**, program lebih mudah dibaca dan dimengerti. Selain itu, pembahasan program dan penelusuran jalannya program (debugging) menjadi lebih mudah sebab dapat langsung diketahui subprogram mana yang berjalan tidak sesuai dengan yang diharapkan.

Pada dasarnya penggunaan procedure dengan function tidak jauh berbeda, berikut disampaikan perbedaan procedure dengan fungsi :

Persamaan Procedure dengan Function :

1. Jumlah parameter actual ketika pemanggilan kedua subrutin ini di program utama harus sama dengan jumlah parameter formal ketika pembuatan subrutin.
2. Type data parameter formal yang bersifat variabel lokal harus sama dengan type data variabel global yang menjadi rujukan pada parameter actual ketika pemanggilan subrutin di program utama

Perbedaan Procedure dengan Function :

1. Procedure bisa mengembalikan nilai/hasil dan bisa juga tidak mengembalikan nilai, tapi function wajib mengembalikan nilai keluaran.
2. Procedure membutuhkan suatu variabel khusus untuk menampung hasil/nilai ketika terjadi suatu proses perhitungan, tapi kalau function tidak membutuhkan karena pada fungsi berlaku ketentuan bahwa nama fungsi = nama/ variabel proses.
3. Pada Procedure proses pencetakan hasil proses berada dalam blok subrutinnya sendiri untuk kemudian tinggal dipanggil nama procedurenya di program utama, tapi pada function proses pencetakan hasil/nilai sekalian dibuat di program utama ketika pemanggilan function nya

Istilah-istilah dalam subrutin :

- a) **Parameter formal**, adalah parameter yang dibuat pada saat setelah pembuatan nama suatu subrutin. Penulisan parameter ini harus disertakan dengan type data dan berada didalam kurung. Parameter ini juga identik dengan variabel yang dibuat secara lokal.
- b) **Parameter actual**, adalah parameter yang dibuat ketika pemanggilan suatu subrutin di program utama. Penulisan parameter ini tidak boleh disertakan dengan type datanya. Parameter

- ini identik dengan variabel yang dibuat secara global.
- c) **Variabel Lokal**, adalah variabel yang dibuat didalam blok suatu subrutin sehingga hanya subrutin itu saja yang bisa mengenal dan menggunakannya
- d) **Variabel Global**, adalah variabel yang dibuat diatas program utama dan bersifat global/luas karena variabel ini bisa dikenal dan digunakan oleh sub-sub program yang berada didalamnya. Kemudian parameter itu sendiri adalah pengertiannya sama dengan variabel. Tapi istilah parameter hanya dipakai ketika membuat program dengan menggunakan konsep subrutin. Dimana parameter ini sendiri nantinya berfungsi untuk menerima inputan dari program utama dan nilai hasil proses untuk selanjutnya akan dikirim lagi ke program utama.

10.2. Prosedur

Prosedur adalah subprogram yang menerima masukan tetapi tidak mempunyai keluaran secara langsung. Pada dasarnya, struktur prosedur sama dengan struktur algoritma yang sudah anda kenal.

Procedur namaprosedur

{spesifikasi prosedur, berisi penjelasan tentang apa yang dilakukan oleh prosedur ini.
 {k.awal : keadaan sebelum prosedur dilaksanakan}
 {k.akhir : keadaan setelah prosedur dilaksanakan}

Deklarasi

{semua nama yang dipakai dalam prosedur dan hanya berlaku local di dalam prosedur di definisikan disini}

Deskripsi

{badan prosedur, yang berisi kumpulan instruksi}

Setiap prosedur mempunyai nama yang unik. Nama prosedur sebaiknya diawali dengan kata kerja karena prosedur berisi suatu aktivitas.

Notasi algoritma yang digunakan untuk mendefinisikan struktur prosedur (tanpa parameter) adalah :

Cara mendeklarasikan sebuah prosedur adalah sebagai berikut

Pendeklarasian prosedur tersebut adalah untuk prosedur yang tidak memerlukan parameter. Parameter adalah data masukan untuk subprogram yang

```
procedure A; { nama prosedur adalah A }
begin
{ statement }
end;
```

nantinya akan diproses lebih lanjut dalam subprogram tersebut. Dalam Pascal, dikenal dua macam parameter yaitu :

- Parameter nilai (*value parameter*), dan
- Parameter referensi (*reference parameter*).

Cara mendeklarasikan parameter tersebut adalah sebagai berikut :

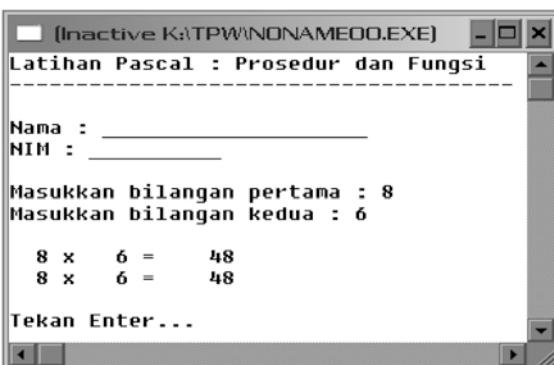
```
procedure B(X : integer; var Y : integer);
begin
{ statement }
end;
```

Pada deklarasi prosedur di atas, parameter X adalah parameter nilai sedang parameter Y adalah parameter referensi. Jadi, pendeklarasian parameter referensi didahului oleh reserved word var. Parameter referensi ini nantinya dapat dijadikan sebagai variabel keluaran dari prosedur. Untuk lebih memahami penggunaan prosedur dalam Pascal maupun Delphi, perhatikan contoh program di bawah ini

```
Program Prosedur;
uses wincrt;
var
Bil_1, Bil_2, Hasil : integer;
procedure Awal;
begin
writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
writeln('-----');
writeln('Nama : _____');
writeln('NIM : _____');
writeln;
end;
procedure Baca_Data;
begin
Write('Masukkan bilangan pertama : ');
Readln(Bil_1);
Write('Masukkan bilangan kedua : ');
Readln(Bil_2);
writeln;
end;
procedure Kali(A,B : integer);
var
I : integer;
begin
Hasil := 0;
for I := 1 to B do Hasil := Hasil + A;
end;
procedure Kalikan(A,B : integer; var C : integer);
var
I : integer;
begin
C := 0;
for I := 1 to B do C := C + A;
end;
begin
ClrScr;
Awal;
Baca_Data;
Kali(Bil_1, Bil_2);
writeln(Bil_1:3,' x ',Bil_2:3,' = ',Hasil:5);
Kalikan(Bil_1, Bil_2, Hasil);
writeln(Bil_1:3,' x ',Bil_2:3,' = ',Hasil:5);
writeln;
Write('Tekan Enter... ');
Readln;
end.
```

Hasil Output Program:

Perhatikan program tersebut. Dua prosedur terakhir memiliki kemiripan, bedanya hanya pada jumlah parameter dan variabel hasil perkaliannya.



```
 (Inactive K:\TPW\NONAME00.EXE)
Latihan Pascal : Prosedur dan Fungsi
-----
Nama : _____
NIM : _____
Masukkan bilangan pertama : 8
Masukkan bilangan kedua : 6
8 x   6 =      48
8 x   6 =      48
Tekan Enter...
```

Untuk lebih jelas, jalankan program dan perhatikan apa yang dilakukan oleh dua prosedur tersebut maka akan nampak perbedaan keduanya.

10.3. Fungsi

Fungsi adalah subprogram yang menerima masukan dan mempunyai keluaran secara langsung. Cara mendeklarasikan sebuah fungsi adalah sebagai berikut :

```
function A : integer; { nama fungsi adalah A dengan }
begin { tipe data keluaran adalah integer }
{ statement }
A := 3; { nilai yang dikeluarkan fungsi }
end;
```

Sebagaimana dalam prosedur, fungsi juga dapat diberikan parameter. Cara mendeklarasikan fungsi dengan parameter juga tidak jauh berbeda dengan pendeklarasian parameter pada prosedur.

Perbedaan utama antara prosedur dan fungsi adalah dalam menghasilkan keluaran. Walaupun prosedur bisa menghasilkan nilai keluaran, tetapi nilai tersebut tidak dapat diambil secara langsung, melainkan harus diambil melalui parameter referensi. Sedangkan keluaran dari fungsi dapat

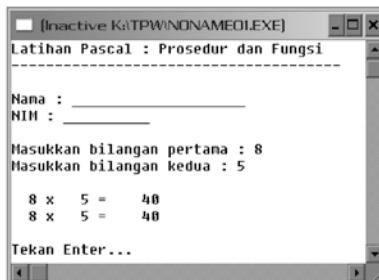
```
function B(X : integer) : integer;
begin
{ statement }
B := X * 2;
end;
```

diamambil langsung dari fungsi tersebut. Untuk lebih memahami perbedaan prosedur dan fungsi, perhatikan contoh berikut ini :

```
program Fungsi;
uses wincrt;
var
Bil_1, Bil_2, Hasil : integer;
procedure Awal;
begin
Writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
Writeln('-----');
function A : integer; { nama fungsi adalah A dengan }
begin { tipe data keluaran adalah integer }
{ statement }
A := 3; { nilai yang dikeluarkan fungsi }
end;
function B(X : integer) : integer;
begin
{ statement }
B := X * 2;
end;
Writeln('Nama : _____');
Writeln('NIM : _____');
Writeln;
end;
procedure Baca_Data;
begin
Write('Masukkan bilangan pertama : ');
Readln(Bil_1);
Write('Masukkan bilangan kedua : ');
Readln(Bil_2);
Writeln;
end;

function Kali(A,B : integer) : integer;
var
I,J : integer;
begin
J := 0;
for I := 1 to B do J := J + A;
Kali := J;
end;
procedure Kalikan(A,B : integer; var C : integer);
var
I : integer;
begin
C := 0;
for I := 1 to B do C := C + A;
end;
begin
ClrScr;
Awal;
Baca_Data;
Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Kali(Bil_1,Bil_2):5);
Kalikan(Bil_1, Bil_2, Hasil);
Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Hasil:5);
Writeln;
Write('Tekan Enter... ');
Readln;
end.
```

Hasil Output Program:



semula perintah :

Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Kali(Bil_1,Bil_2):5); diubah menjadi :
Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Kalikan(Bil_1,Bil_2,Hasil):5);

Perhatikan program tersebut. Prosedur Kalikan dan fungsi Kali mempunyai keluaran yang sama, tetapi cara mengambil keluarannya berbeda. Perhatikan dan jelaskan apa yang terjadi jika baris keempat dalam program utama yang

10.4. Rekursi

Dalam Pascal, ada satu kelebihan dalam cara pemanggilan subprogram. Pascal mengijinkan pemanggilan suatu subprogram dari dalam subprogram itu sendiri. Tidak semua bahasa pemrograman mengijinkan cara pemanggilan subprogram seperti itu karena akan banyak memakan memori. Untuk lebih jelasnya perhatikan potongan program di bawah ini :

```
procedure Z;  
begin  
{ statement }  
Z;  
end;
```

Pada baris terakhir prosedur Z, terdapat pemanggilan kembali terhadap prosedur Z, sehingga prosedur tersebut tidak akan pernah selesai dijalankan sebab begitu sampai pada baris terakhir dari prosedur, program akan

kembali lagi ke awal prosedur. Yang terjadi adalah semacam perulangan tanpa perintah perulangan Pascal, dan perulangan dengan cara ini disebut dengan rekursi. Rekursi berlaku terhadap semua subprogram dalam Pascal, yaitu prosedur dan fungsi.

Dengan adanya rekursi ini, banyak algoritma komputer menjadi lebih mudah dibuat programnya. Berikut ini adalah program menghitung suku banyak Legendre, salah satu contoh perhitungan yang dapat diselesaikan dengan menggunakan rekursi :

```

Program Rekursi;
uses wincrt;
var
Jum_Suku, I : integer;
Bil_X : real;
function Legendre(X : real; N : integer) : real;
var
Suku_1, Suku_2 : real;
begin
if N = 0 then
Legendre := 1
else if N = 1 then
Legendre := X
else
begin
Suku_1 := ((2*N - 1) * (X * Legendre(X, N-1))) / N;
Suku_2 := ((N-1) * Legendre(X, N-2)) / N;
Legendre := Suku_1 + Suku_2;
end;
end;
procedure Awal;
begin
writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
writeln('-----');
writeln;

```

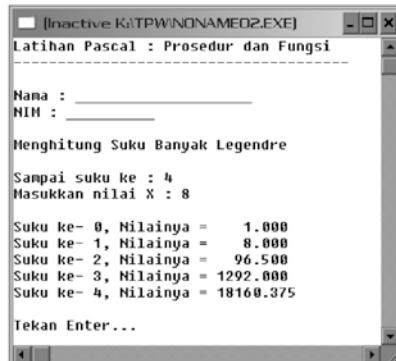
```

Writeln('Nama : _____');
Writeln('NIM : _____');
Writeln;
end;
procedure Baca_Data;
begin
writeln('Menghitung Suku Banyak Legendre');
writeln;
Write('Sampai suku ke : ');
Readln(Jum_Suku);
Write('Masukkan nilai X : ');
Readln(Bil_X);
writeln;
end;
begin
ClrScr;
Awal;
Baca_Data;
for I := 0 to Jum_Suku do
begin
writeln('Suku ke-',I:2,' Nilainya = ',Legendre(Bil_X, I):8:3);
end;
writeln;
Write('Tekan Enter...');


```

Hasil Output Program:

Untuk lebih jelas memahami program, jalankan program dengan F7. Perhatikan pula apa yang dilakukan oleh fungsi Legendre. Amati perubahan variabelvariabel yang terlibat dalam fungsi.



BAB XI

ARRAY(LARIK dan RECORD)

11.1. Pengantar

Bagi para pemrogram, efisiensi program merupakan hal utama yang harus diperhatikan, baik itu dalam hal kecepatan jalannya program, memori yang digunakan, banyak baris kode yang dituliskan dan juga ketepatan algoritma yang digunakan. Salah satu komponen yang harus dikuasai untuk memperoleh program yang baik adalah pengetahuan tentang array.

Dalam bahasa Pascal maupun Delphi, secara garis besar dikenal dua macam tipe data yaitu tipe data sederhana (primitive type) dan tipe data kompleks (complex type). Contoh tipe data sederhana adalah tipe numerik (integer dan real), tipe data karakter, tipe data boolean dan tipe data enumerasi. Contoh tipe data kompleks adalah string, array (lariik), record dan object. Tipe data sederhana adalah tipe data yang hanya mampu menyimpan satu nilai tiap satu variabelnya. Sebaliknya tipe data kompleks adalah tipe data yang mampu menyimpan lebih dari satu nilai dalam tiap satu variabelnya. Dalam buku ini hanya akan dibahas dua tipe data kompleks yaitu array dan record.

Array merupakan sebuah variabel yang dapat menyimpan lebih dari satu nilai yang memiliki tipe data sama. Hal ini berbeda dengan variabel biasa yang hanya mampu menampung satu buah nilai. Setiap nilai yang disimpan di dalam array disebut dengan elemen array, sedangkan nilai urut yang digunakan untuk mengakses elemennya disebut dengan **indeks array**.

Apabila kita akan membuat program untuk menyimpan sekumpulan data, misalnya data-data hasil penelitian yang berupa bilangan, dimana jumlah dari data tersebut puluhan, ratusan atau bahkan ribuan, apakah akan menggunakan variabel sebanyak data yang ada? Jawabannya tentu tidak, karena hal tersebut merupakan hal yang sangat tidak efisien. Penggunaan array dalam program akan membuat program lebih efisien dan mudah dipahami.

11.2. Array (Larik)

Adalah tipe terstruktur yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe yang sama. Banyaknya komponen dalam suatu Larik ditunjukkan oleh suatu Index yang disebut dengan tipe Index. Tiap-tiap komponen di Larik dapat diakses dengan menunjukkan nilai Indexnya atau Subscript.

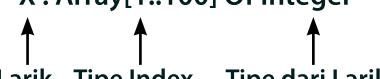
■ Deklarasi Larik

Suatu Larik yang akan dipergunakan didalam program Pascal harus dideklarasikan terlebih dahulu. Deklarasi dari Larik diadahului dengan kata cadangan Array diikuti oleh tipe Index yang diletakkan diantara tanda ‘[]’, diikuti lagi kata cadangan **Of** dan Tipe Lariknya.

Larik dapat bertipe data sederhana Byte, Word, Integer, Real, Boolean, Char atau String dan tipe Data Skalar atau Sub-range. Tipe Larik ini artinya isi dari Larik atau komponen-komponennya mempunyai nilai dengan tipe data tersebut. Tipe dari Larik ditunjukkan pada waktu mendeklarasikannya.

Contoh :

```
Var  
    X : Array[1..100] Of Integer
```



Larik X telah dideklarasikan sebagai Larik tipe Integer dengan Jumlah elemennya maksimum sebanyak 100 elemen. Nilai-nilai elemen Larik ini harus berisi nilai-nilai Integer. Misalnya elemen-elemen dari Larik X adalah :

```
X[1] := 25  
X[2] := 75  
X[3] := 8  
X[4] := 23
```



Bila nilai elemen ke-3 dari Larik X akan ditampilkan maka dapat dipergunakan statement :

Writeln(X[3])

■ Deklarasi Tipe Indeks

Index dari Larik menunjukkan maksimum banyaknya elemen-elemen dari Larik. Index Larik ini dapat berupa tipe Subrange atau tipe Skalar kecuali tipe Real.

● Deklarasi Tipe Index Subrange Integer

Pada deklarasi tipe Larik sebelumnya semuanya menggunakan tipe Index Subrange Integer sebagai berikut :

~ Type

Jangkauan = 1...5;

X = Array[Jangkauan] Of Char;

Var

Pada tipe Index Larik ini **Nilai Huruf : X** bahwa Larik X mempunyai elemen maksimum sebanyak 5 buah komponen yang ditunjukkan oleh Index terbawah berupa nilai Integer 1 dan Index teratas ditunjukkan oleh nilai Integer 5. Index dari Larik ini dapat dideklarasikan terlebih dahulu dibagian deklarasi tipe sebagai berikut :

~ Type

Jangkauan = 1...5; (tipe subrange)

Var

Nilai Huruf : Array[Jangkauan] Of Char;

Atau dapat ditulis :

Var

NilaiHuruf : Array [1...5] Of Char

Tipe Index Subrange Integer

Atau dapat ditulis :

```
~Type  
    X = Array[1...5] Of Char;  
Var  
    Nilai Huruf : X;
```

- **Deklarasi Tipe Index Subrange Byte**

Kalau Index dari Larik tidak sampai dengan 255, maka Index Larik ini dapat dideklarasikan dengan tipe Byte.

Contoh :

```
Var  
    X : Array[0...255] Of Real;
```

Karena nilai 0 sampai dengan 255 merupakan nilai Subrange Byte, maka deklarasi ini dapat juga ditulis :

```
Var  
    X : Array[Byte] Of Real;
```

- **Deklarasi Tipe Index Subrange Word**

Bila Index dari Larik mempunyai jangkauan Index dari 0 sampai dengan 65535, maka Index dari Larik ini dapat dideklarasikan dengan tipe Word.

Contoh :

```
Var  
    X : Array[Word] Of Integer;
```

- **Deklarasi Tipe Index Subrange Boolean**

Tipe Index dari Larik dapat juga bertipe Boolean. Nilai Boolean hanya mempunyai dua buah nilai saja, yaitu True dan False. Jadi Index Larik yang bertipe Boolean hanya mempunyai maksimum 2 buah elemen saja.

Contoh :

```
Type  
    Keterangan = String;  
Var  
    X : Array[Boolean] Of Keterangan;
```

● Deklarasi Tipe Index Subrange Char

Tipe Char sebenarnya adalah tipe subrange yang mempunyai nilai sebanyak 256 buah (dari 0 sampai dengan 255) sesuai dengan urutan kode ASCII. Indeks dari Larik dapat dideklarasikan berupa subrange Char.

Contoh :

```
~ Var  
  X : Array[Char] Of Integer;  
~ Var  
  X : Array['#'...''] Of Integer;
```

Dari deklarasi ini berarti Larik X mempunyai elemen sebanyak 91 buah elemen. Mulai dari elemen urutan ke ‘#’, ‘&’, ‘%’ dan seterusnya sampai urutan yang terakhir adalah urutan ke ‘’}. Misalnya urutan ketiga dari Larik X ini akan diisi dengan nilai 15, maka dapat dilakukan :

```
X[%] := 15;
```

● Deklarasi Tipe Indeks Skalar

Index dari Larik dapat berupa tipe scalar atau enumerated.

Contoh :

```
Var  
  Jumlah : Array[(Jan,Feb,Mar,Apr,Mei)] Of Integer;  
Begin  
  Jumlah[Jan] := 125  
  Jumlah[Feb] := 75  
  Jumlah[Mar] := 67  
  Jumlah[Apr] := 123  
  Jumlah[Mei] := 200  
  Writeln('Jumlah untuk bulan Februari =',Jumlah[Feb]);  
End.  
Output :  
  Jumlah untuk bulan Februari = 75
```

Deklarasi dari Larik ini dapat juga ditulis :

```
Type  
  Bulan = (Jan,Feb,Mar,Apr,Mei);  
Var  
  Jumlah : Array[Bulan] Of Integer;
```

- **Deklarasi Konstanta Larik**

Suatu Larik tidak hanya dapat berupa suatu Variabel yang dideklarasikan dibagian deklarasi variabel. Tetapi juga dapat berupa suatu konstanta yang di deklarasikan dibagian deklarasi Konstanta.

Contoh ini memperlihatkan deklarasi konstanta Larik tipe Integer yang mempunyai 5 buah elemen sebagai berikut :

```
Const  
      X : Array[1...5] Of Integer = (6,25,375,5,2);  
Var  
      I : Word;  
Begin  
      For I := 1 to 5 Do  
      Writeln('Nilai Konstanta Larik Ke', I, '=', X[I]);  
End.  
Output :  
      Nilai Konstanta Larik Ke 1 = 6  
      Nilai Konstanta Larik Ke 2 = 25  
      Nilai Konstanta Larik Ke 3 = 375  
      Nilai Konstanta Larik Ke 4 = 5  
      Nilai Konstanta Larik Ke 5 = 2
```

Contoh di bawah ini memperlihatkan deklarasi konstanta Larik tipe Boolean yang mempunyai 3 buah elemen saja.

```
Const  
      Nilai : Array[1...3] Of Boolean = (True, False, False);  
Var  
      I : Word;  
Begin  
      For I := 1 to 3 Do  
      Writeln('Nilai Ke', I, '=', Nilai[I]);  
End.  
Output :  
      Nilai Ke 1 = True  
      Nilai Ke 2 = False  
      Nilai Ke 3 = False
```

- **String Sebagai Larik Tipe Char**

Suatu String dapat dianggap sebagai suatu Larik tipe Char dengan Indeks dari 0 sampai dengan panjang dari String yang didefinisikan. Misalnya Nilai String sebagai berikut :

X := 'ABCD';

Maka :

```
X[1] := 'A';
X[2] := 'B';
X[3] := 'C';
X[4] := 'D';
```

Indeks ke 0 dari nilai String ini berisi nilai String yang menunjukkan panjang dari Stringnya. X[0] := #4; Karena panjang dari String ini berupa nilai karakter, untuk mendapatkan dalam bentuk nilai numeric, maka dapat dipergunakan Fungsi Standar Ord sebagai berikut :

Ord(X[0]) := 4;

Contoh :

```
Var
    I : Word;
    Nama : String;
Begin
    Write('Nama Anda ?');ReadIn(Nama);
    Writeln
    Writeln('Nama Anda Kalau Dibaca Terbalik Adalah :');
    For I := Ord>Nama[0] Downto 1 Do
        Write(Nama[I]);
End.
```

Output :

Nama Anda : Aryanto
Nama Anda Kalau Dibaca Terbalik Adalah :
Otnayra

● **Larik Dimensi Banyak**

Larik dapat juga berdimensi lebih dari satu yang disebut dengan Larik Dimensi Banyak (Multidimensional Array), yang dapat berdimensi dua (Two Dimensional Array), berdimensi Tiga (Three Dimensional Array) dan seterusnya. Pada pembahasan ini hanya akan dibahas Larik 2 Dimensi. Larik 2 Dimensi mewakili suatu bentuk tabel atau Matrik, yaitu Indeks yang pertama dapat menunjukkan Baris dan Indeks Kedua dapat menunjukkan Kolom dari Tabel atau Matrik. Bentuk Larik dimensi 2 berupa :

Nama Larik = Array[Tipe Indeks1, Tipe Indeks2] Of Tipe Larik

Contoh :

```
Var
    Tabel : Array[1...3,1...2] Of Byte;
    I,J : Byte;
Begin
    Tabel[1,1] := 5;
    Tabel[1,2] := 25;
    Tabel[2,1] := 200;
    Tabel[2,2] := 22;
    Tabel[3,1] := 75;
    Tabel[3,2] := 50;
    For I := 1 to 3 Do
        Begin
            For J := 1 to 2 Do
                Write(Tabel[I,J] : 10);
                Writeln;
            End;
        End.
Output :
      5          25
     200        22
      75          50
```

11.3. Record

Record adalah salah satu tipe data terstruktur bentukan yang digunakan untuk mempresentasikan sebuah objek yang tidak dapat dipresentasikan menggunakan tipe data dasar, seperti integer, real, boolean, character. Setiap record terdiri dari beberapa elemen yang disebut field. Setiap field menggambarkan informasi tertentu, dan tipe setiap field sudah dikenal, baik itu tipe dasar atau tipe bentukan lainnya.

Operasi atau manipulasi terhadap record hanya dapat dilakukan terhadap field-field pembentuknya. Pengacuan pada setiap field dilakukan dengan record selector. Operasi yang dapat dilakukan terhadap field-field tersebut sama dengan operasi yang dapat dikenakan terhadap tipe pembentuknya.

Record adalah tipe data kompleks yang elemen-elemennya boleh mempunyai tipe data yang berbeda. Record lebih kompleks daripada array karena record merupakan kumpulan beberapa variabel dengan tipe data yang berbeda. Berbeda dengan array yang tiap elemennya ditandai dengan nomer indeks maka record ditandai dengan nama variabel anggotanya. Cara mengakses elemen dari record dilakukan dengan menyebutkan nama variabel anggota setelah menyebutkan nama record yang akan diakses. Di antara nama record dan nama variabel anggota dipisahkan tanda titik (.).

Pendeklarasian record :

Type

```
Nama_record = record  
    Field1: tipe data1 ;  
    Field2: tipe data2 ;  
    .....  
    .....  
    Fieldn: tipe datan ;  
End ;
```

Contoh :

```
Type Barang = record  
    Nama : string[20] ;  
    Jenis : string [20]  
    Jumlah : integer ;  
End ;
```

Memasukkan data ke dalam record :

Untuk memberikan nilai dari masing-masing field maka kita harus menuliskan Nama_record.field := (nilainya);

Misalkan : dari contoh diatas kita akan mengisikan nama barang dengan Piring, jenis barang yaitu barang pecah belah dan jumlah barang 3 lusin maka kita harus menuliskan pada program utama

```
Barang.Nama := 'Piring' ;  
Barang.Jenis := 'Pecah belah' ;  
Barang.Jumlah:= 36 ;
```

Nilai-nilai dari field ini akan disimpan dalam record. Untuk melihat apakah benar data yang dimasukkan telah tersimpan dalam

record maka pada var kita deklarasikan suatu variable misal :

X : array[1..n] of Nama_record ; dari soal di atas yaitu X : array[1..n] of Barang ;

Maka apabila nanti kita lakukan pemanggilan dengan mengetikkan Write(Barang[i].nama), data dari field yang tersimpan dalam record tersebut akan ditampilkan

Contoh program :

PROGRAM DATABASE;

Uses crt;

TYPE mahasiswa=record

 nama: array[1..20] of string;

 nim: array[1..20] of string;

 alamat: array[1..20] of string;

 ipk: array[1..20] of real;

end;

VAR data1: mahasiswa;

PROCEDURE data(var mhs:mahasiswa; mhs1:mahasiswa);

Var i,n,no:integer;

 pilih,tekan:char;

Begin

 write('Masukan jumlah mahasiswa : ');readln(n);

 writeln;

 for i:= 1 to n do

 begin

 writeln('Masukan data mahasiswa ke - ',i);

 writeln;

 write('Nama Mahasiswa : ');readln(mhs.nama[i]);

 write('No. Mahasiswa : ');readln(mhs.nim[i]);

 write('Alamat Mahasiswa : ');readln(mhs.alamat[i]);

 write('IPK : ');readln(mhs.ipk[i]);

 writeln;

 end;

```

writeln;
writeln('DATA MAHASISWA');
writeln;
writeln('=====');
writeln('|'No':5>Nama':20,NIM':10,Alamat':20,IPK':10,'|:2);
writeln('=====');
for i:=1 to n do
writeln('|';i:5,mhs.nama[i]:20,mhs.nim[i]:10,mhs.alamat[i]:20,mhs.ipk[i]:10:2, '|:2);
writeln('=====');
writeln;
write('Ingin mencari data tertentu (y/n) ? ');readln(pilih);
writeln;
case pilih of
'y': begin
    tekan:='Y';
    while upcase(tekan)='Y' do
begin
    clrscr;
    writeln;
    writeln('MENU PILIHAN');
    writeln;
    writeln('[1] NAMA');
    writeln('[2] NIM');
    writeln('[3] ALAMAT');
    writeln('[4] IPK');
    writeln;
    write('Pilihan anda : ');readln(no);
case no of
1: begin
    write('Masukan Nama Mahasiswa : ');readln(mhs1.nama);
    writeln;
writeln('=====');
    writeln('|Nama':20,NIM':10,Alamat':20,IPK':10,'|:2);
writeln('=====');
for i:=1 to n do
if (mhs1.nama) = (mhs.nama[i]) then
begin
    writeln('|';mhs1.nama:20,mhs.nim[i]:10,mhs.alamat[i]:20,mhs.i
    pk[i]:10:2, '|:2);
    end;
writeln('=====');
    writeln;
    end;
2: begin
    write('Masukan No. Mahasiswa : ');readln(mhs1.nim);
    writeln;
writeln('=====');

```

```

writeln(';"Nama':20,'NIM':10,'Alamat':20,'IPK':10,'':2);
writeln('=====');
for i:=1 to n do
  if (mhs1.nim) = (mhs.nim[i]) then
    begin
writeln(';',mhs.nama[i]:20,mhs.nim[i]:10,mhs.alamat[i]:20,mhs.ipk[i]:1
0:2,'':2);
    end;
  writeln('=====');
  writeln;
  end;
3: begin
  write('Masukan Alamat Mahasiswa : ');readln(mhs1.alamat);
  writeln;
  writeln('=====');
  writeln(';"Nama':20,'NIM':10,'Alamat':20,'IPK':10,'':2);
  writeln('=====');
  for i:=1 to n do
    if (mhs1.alamat) = (mhs.alamat[i]) then
      begin
writeln(';',mhs.nama[i]:20,mhs.nim[i]:10,mhs1.alamat:20,mhs.ipk[i]:
10:2,'':2);
      end;
    writeln('=====');
    writeln;
    end;
4: begin
  write('Masukan IPK : ');readln(mhs1.ipk);
  writeln;
  writeln('=====');
  writeln(';"Nama':20,'NIM':10,'Alamat':20,'IPK':10,'':2);
  writeln('=====');
  for i:=1 to n do
    if (mhs1.ipk) = (mhs.ipk[i]) then
      begin
writeln(';',mhs.nama[i]:20,mhs.nim[i]:10,mhs.alamat[i]:20,mhs1.ipk:
10:2,'':2);
      end;
    writeln('=====');
    writeln;
    end;
  end;
  write('Ingin mencari data lagi (y/n) ? ');readln(tekan);
  writeln;
end; end; end; end;

```

Hasil Run Program :

Masukan jumlah mahasiswa : 4

Masukan data mahasiswa ke - 1

Nama Mahasiswa : Tumpal PS

No. Mahasiswa : 8051

Alamat Mahasiswa : KalBar

IPK : 3.5

Masukan data mahasiswa ke - 2

Nama Mahasiswa : Sri Sunarwati

No. Mahasiswa : 8244

Alamat Mahasiswa : Klaten

IPK : 3.4

Masukan data mahasiswa ke - 3

Nama Mahasiswa : Putu Eka A

No. Mahasiswa : 8239

Alamat Mahasiswa : Bali

IPK : 3.3

Masukan data mahasiswa ke - 4

Nama Mahasiswa : Timotius N

No. Mahasiswa : 8299

Alamat Mahasiswa : Tegal

IPK : 3.5

DATA MAHASISWA

No	Nama	NIM	Alamat	IPK	
1	Tumpal PS	8051	KalBar	3.50	
2	Sri Sunarwati	8244	Klaten	3.40	
3	Putu Eka A	8239	Bali	3.30	
4	Timotius N	8299	Tegal	3.50	

Ingin mencari data tertentu (y/n) ? y

MENU PILIHAN

[1] NAMA

[2] NIM

[3] ALAMAT

[4] IPK

Pilihan anda : 1

Masukan Nama Mahasiswa : Tumpal PS

	Nama	NIM	Alamat	IPK	
	Tumpal PS	8051	KalBar	3.50	

Ingin mencari data lagi (y/n) ? y

MENU PILIHAN

[1] NAMA

[2] NIM

[3] ALAMAT

[4] IPK

Pilihan anda : 2

Masukan No. Mahasiswa : 8299

Nama	NIM	Alamat	IPK
Timotius N	8299	Tegal	3.50

Ingin mencari data lagi (y/n) ? y

MENU PILIHAN

[1] NAMA

[2] NIM

[3] ALAMAT

[4] IPK

Pilihan anda : 3

Masukan Alamat Mahasiswa : Bali

Nama	NIM	Alamat	IPK
Putu Eka A	8239	Bali	3.30

Ingin mencari data lagi (y/n) ? y

MENU PILIHAN

[1] NAMA

[2] NIM

[3] ALAMAT

[4] IPK

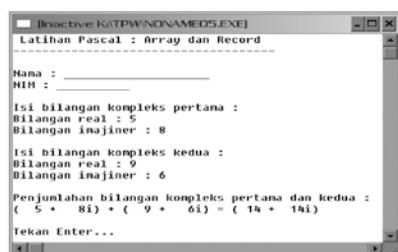
Pilihan anda : 4

Masukan IPK : 3.4

Nama	NIM	Alamat	IPK
Sri Sunarwati	8244	Klaten	3.40

Untuk lebih memahami penggunaan record dalam program, perhatikan contoh berikut ini :

```
program Jumlah_Kompleks;
uses wincrt;
Type
Kompleks = record
bil_real : integer;
bil_imaj : integer;
end;
var
K1, K2, H : kompleks;
procedure Awal;
begin
Writeln(' Latihan Pascal 3 : Array dan Record');
Writeln('-----');
Writeln;
Writeln('Nama : _____');
Writeln('NIM : _____');
Writeln;
end;
procedure JumlahKompleks(var Komp1, Komp2, KompHasil :
kompleks);
begin
KompHasil.bil_real := Komp1.bil_real + Komp2.bil_real;
KompHasil.bil_imaj := Komp1.bil_imaj + Komp2.bil_imaj;
end;
procedure BacaData(var Komp : kompleks);
begin
Write('Bilangan real : ');
Readln(Komp.bil_real);
Write('Bilangan imajiner : ');
Readln(Komp.bil_imaj);
end;
procedure TulisKompleks(var Komp : kompleks);
begin
Write('(''Komp.bil_real:3+'+''Komp.bil_imaj:3,i)'');
end;
begin
ClrScr;
Awal;
Writeln('Isi bilangan kompleks pertama :');
BacaData(K1);
Writeln;
Writeln('Isi bilangan kompleks kedua :');
BacaData(K2);
Writeln;
JumlahKompleks(K1, K2, H);
Writeln('Penjumlahan bilangan kompleks pertama dan kedua :');
Writeln('Penjumlahan bilangan kompleks pertama dan kedua :');
```



Perhatikan program di atas. Untuk lebih jelasnya, jalankan program dengan F7 sehingga akan terlihat urutan jalannya program. Perhatikan pula bagaimana cara mengakses elemen record seperti pada prosedur Jumlah Kompleks.

Soal Dan Pembahasan:

1. Program Fibonacci;

```
uses wincrt;
var
I: integer;
Data : array[1..10] of integer;
procedure Awal;
begin
Writeln('Praktikum DKP III : Array dan Record');
Writeln('-----');
Writeln;
Writeln('Nama : _____');
Writeln('NIM : _____');
Writeln;
end;
procedure Fibo;
begin
for I:= 1 to 10 do
begin
if I < 3 then
Data[I] := I - 1
else
Data[I] := Data[I-1] + Data[I-2];
end;
Writeln('Deret Fibonacci suku ke-1 hingga suku ke-10 :');
for I := 1 to 10 do Write(Data[I]:3);
Writeln;
end;
begin
ClrScr;
Awal;
Fibo;
Writeln;
Write('Tekan Enter...:');
Readln;
end.
```

program di atas adalah array berdimensi tunggal atau array berdimensi satu. Dengan demikian, dapat pula dideklarasikan variabel array dengan dimensi lebih dari satu atau array berdimensi banyak. Berikut adalah cara mendeklarasikan array berdimensi dua :

2. Program Jumlah_Matrik;

```
uses wincrt;
const
Orde = 3;
type
Matrik = array[1..orde,1..orde] of integer;
var
M1, M2, H : matrik;
I, J : integer;
procedure Awal;
begin
```

```
Writeln('Latihan Pascal 3 : Array dan Record');
Writeln('-----');
Writeln;
Writeln('Nama : _____');
Writeln('NIM : _____');
Writeln;
end;
procedure JumlahMatrik(var Mat1, Mat2, MatHasil : matrik);
begin
for I := 1 to orde do
for J := 1 to orde do
MatHasil[I,J] := Mat1[I,J] + Mat2[I,J];
end;
procedure BacaData(var Mat : matrik);
begin
for I:= 1 to orde do
for J := 1 to orde do
begin
Writeln('Nilai['I,'J] = ');
Readln(Mat[I,J]);
end;
end;
procedure TulisMatrik(var Mat : matrik);
begin
for I := 1 to orde do
begin
for J := 1 to orde do
begin
Writeln(Mat[I,J]:5);
end;
end;
end;
begin
ClrScr;
Awal;
Writeln('Isi matrik pertama :');
BacaData(M1);
Writeln;
Writeln('Isi matrik kedua :');
BacaData(M2);
Writeln;
JumlahMatrik(M1, M2, H);
Writeln('Penjumlahan matrik pertama dan kedua :');
TulisMatrik(H);
Writeln;
Write('Tekan Enter...:');
Readln;
end.
```

BAB XII

Pengantar Bahasa

Program pythonTM

Bahasa pemrograman Python pertama kali dikembangkan pada tahun 1991 oleh Guido van Rossum, seorang programmer Belanda. Hingga kini, Python telah berkembang pesat. Terbukti dari versi terbarunya, Python 3.10 yang dirilis 2021 lalu.

Bahasa pemrograman yang bisa dipelajari oleh pemula adalah Python. Python adalah bahasa pemrograman interpretative, high-level programming (pemrograman yang mendekati bahasa manusia), memiliki kode-kode pemrograman jelas dan mudah dipahami, alasan-alasan yang mendukungnya:

- Mudah Dipelajari, tanpa membutuhkan computer yang canggih dan mutakhir
- Memiliki Library Lengkap.
- Gratis (Open Source)
- Bersifat multiplatform (SO Windows, MacOs, Chrome, Linux)
- Meningkatkan produktivitas dalam pemrograman big data, data mining, deep learning, data science hingga machine learning, juga untuk aplikasi berbasis kecerdasan buatan (artificial intelligence).

Python disebut sebagai bahasa pemrograman interpreter karena ia mengeksekusi instruksi satu per satu, segera setelah diterima, bekerja dengan menggunakan kompiler. Ketika dijalankan, Pyhton akan bekerja lebih lambat jika dibandingkan dengan bahasa lain. Namun hal ini juga tergantung dari besar atau kecilnya program yang akan dibuat.

Python adalah bahasa pemrograman interpretatif yang dapat digunakan di berbagai sistem operasi baik Windows, Linux, MacOS atau Android dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python bisa dipakai untuk banyak hal (general purpose), seperti Aplikasi Desktop, Mobile, Web, menganalisis data (Data Science & Data Analysis), Artificial Intelligence, Machine Learning, dan Robotics, dan Internet of Things (IoT).

Python adalah bahasa tingkat tinggi dengan sintaks mirip bahasa manusia, sehingga mudah dipelajari. Bahkan jika dibandingkan dengan sesama bahasa program tingkat tinggi, perintah Python masih jauh lebih sederhana.

Contoh sintaksis untuk membuat program Hello World dengan perintah C vs Java vs Python berikut:

“Hello, World”

- C

```
#include <stdio.h>

int main(int argc, char ** argv)
{
    printf("Hello, World!\n");
}
```

- Java

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println("Hello, World!");
    }
}
```

- now in Python

```
print "Hello, World!"
```

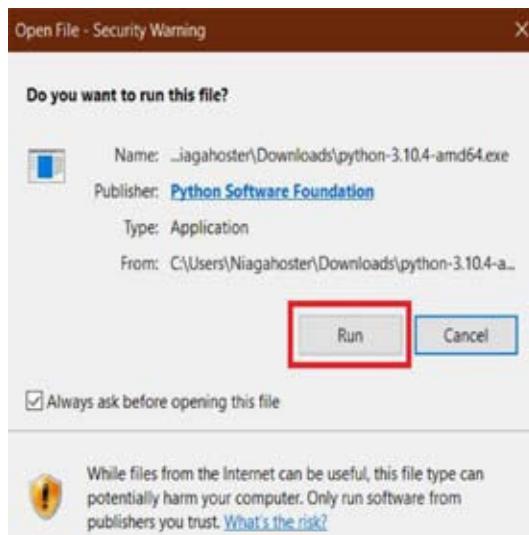
Dari sintaksis diatas nampak bahwa menulis sintaksis python lebih sederhana.

Bagaimana cara menginstal Python?

Untuk menginstal Python menggunakan Microsoft Store: Buka menu Mulai (ikon Windows kiri bawah), ketik "Microsoft Store", pilih tautan untuk membuka penyimpanan. Setelah toko terbuka, pilih Cari dari menu kanan atas dan masukkan "Python". Pilih versi Python mana yang ingin Anda gunakan, Di halaman ini, silakan unduh Python versi terbaru sesuai perangkat yang Anda gunakan. Sebagai contoh, kami akan mendownload Python 3.10.4 untuk Windows.



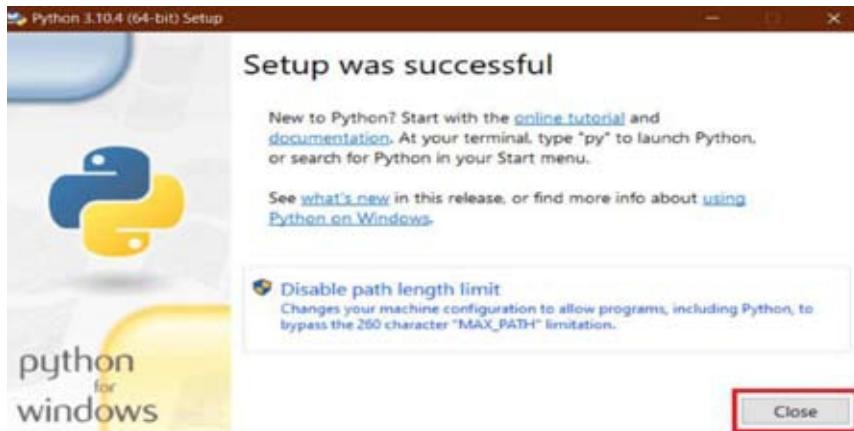
Silakan tunggu, proses download akan berlangsung. Jika sudah, klik dua kali file instalasi Python tersebut. Setelahnya, akan muncul tampilan berikut dan klik Run.



Di tampilan berikutnya, silakan klik centang pada opsi Add Python 3.10 to PATH. Kemudian, pilih opsi Install Now.



Proses instalasi Python akan berjalan. Jika sudah, Anda akan melihat tampilan berikut.



Itu artinya, Python sudah berhasil terinstall di komputer. Akhiri prosesnya dengan klik Close.

Mem-Validasi Instalasi Python

Sampai di tahap ini, tahap instalasi python sudah selesai. Setelah itu Anda bisa memvalidasi apakah Python sudah benar-benar terinstal di komputer atau belum dengan cara menggunakan software terminal bawaan milik komputer. Contohnya untuk Windows, Anda bisa memanfaatkan aplikasi Command Prompt.

Pertama, buka Command Prompt tersebut. Kemudian, tuliskan perintah di bawah ini:

```
python --version
```

Jika sudah, silakan tekan Enter. Anda akan melihat tampilan berikut. Itu artinya, Python sudah terinstall di komputer Anda.

A screenshot of a Microsoft Windows Command Prompt window titled "Command Prompt". The window shows the following text:

```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

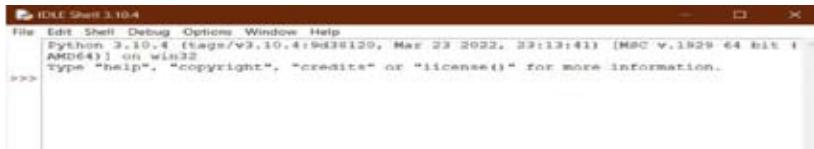
C:\Users\Niagahoster>python --version
Python 3.10.4

C:\Users\Niagahoster>
```

A red box highlights the command "python --version" and its output "Python 3.10.4".

Memulai Menggunakan Python

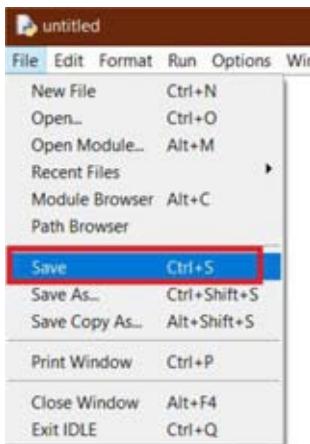
Setelah menginstal Python dan memvalidasinya, Anda bisa mulai belajar cara menggunakan Python. Python menyediakan teks editor bawaan bernama *IDLE*. Sehingga, Anda tidak perlu repot memasang teks editor lain. Cukup Anda, membuka aplikasi *IDLE Python*. Jika sudah, Anda akan melihat tampilan seperti berikut:



Pertama, silakan klik menu File > New File untuk membuat program sederhana Python. Anda akan dibawa menuju tampilan seperti ini:



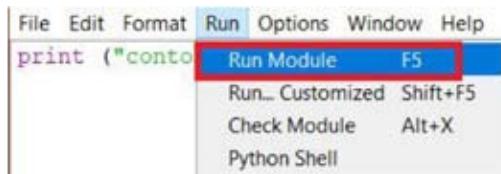
Sebelum menulis contoh program Python, ada baiknya Anda menyimpan file terlebih dahulu. Caranya gampang, tinggal klik File > Save dan simpan file sesuai kehendak Anda.



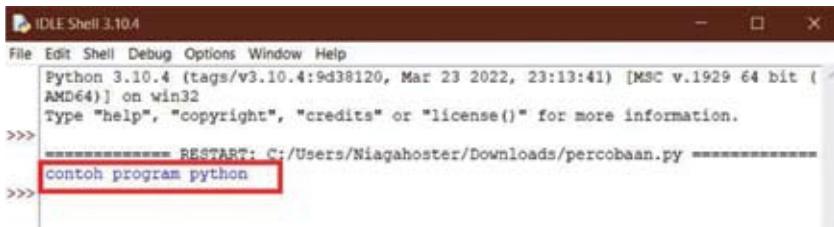
Jika sudah, silakan tulis sintaks Python Anda. Sebagai contoh, tulis perintah berikut:

```
File Edit Format Run Options Window Help  
print ("contoh program python")|
```

Setelah itu saatnya menjalankan program. Langsung saja klik menu Run > Run Module.

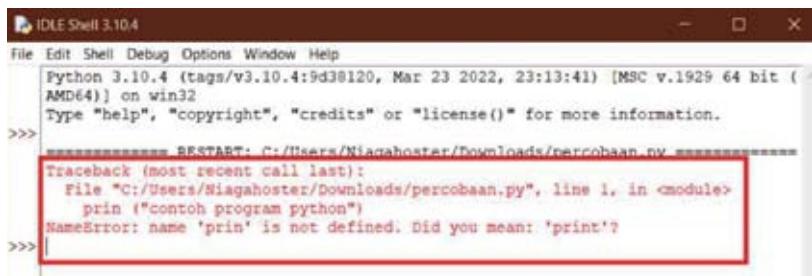


Berikutnya, Anda akan melihat tampilan berikut. Itu artinya, program sederhana Python berhasil dijalankan.



```
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d3f38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:/Users/Niagahoster/Downloads/percobaan.py =====
contoh program python
>>>
```

Sebaliknya jika yang muncul seperti ini, maka terdapat error di program Anda. Silakan periksa kembali sintaks Python Anda, lalu jalankan ulang program tersebut.



```
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d3f38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:/Users/Niagahoster/Downloads/percobaan.py =====
Traceback (most recent call last):
  File "C:/Users/Niagahoster/Downloads/percobaan.py", line 1, in <module>
    prin ("contoh program python")
NameError: name 'prin' is not defined. Did you mean: 'print'?
>>>
```

Tipe Data Python

Sebelum belajar berbagai tipe data Python, ada hal menarik di Python, karena Anda tidak perlu repot mengenalkan tipe data setiap kali menulis program. Sebab, Python sudah bisa mengenali tipe data yang Anda gunakan. Meski begitu, Anda tetap wajib belajar tipe data Python. Sebab, setiap tipe data punya aturan penulisannya masing-masing. Sehingga, Anda wajib mematuhi jika tidak ingin program Python Anda malah error.

Tipe Data Python

- Integer (Bilangan Bulat)
- Float (Bilangan Desimal)
- String (Teks & Karakter)
- Boolean (True & False)
- Tipe Data Khusus (Bilangan Kompleks)

Sebagai perbandingan, Anda perlu mendeklarasikan tipe data saat membuat program dalam bahasa pemrograman Java. seperti dibawah ini :

PYTHON TYPE DECLARATION

```
ivar = 10  
cvar = 'H'  
bvar = True  
fvar = 3.14
```

JAVA TYPE DECLARATION

```
int iVar = 10;  
char cVar = 'G';  
boolean bVar = true;  
float fVar = 3.14f;
```

Adapun 5 tipe data yang ada di Python, adalah :

1) Integer (Bilangan Bulat)

Sesuai namanya, integer digunakan untuk memasukkan nilai bilangan bulat.

Berikut contohnya: usia = 20, harga = 25000 dan suhu = -13

Untuk memeriksa tipe data di atas, gunakan fungsi type().

Contoh penggunaanya adalah sebagai berikut:

```
type (usia)  
type (harga)  
type (suhu)
```

Sementara itu, berikut hasilnya setelah dijalankan:

```
>>> usia = 20  
>>> harga = 25000  
>>> suhu = -13  
>>> type (usia)  
<class 'int'>  
>>> type (harga)  
<class 'int'>  
>>> type (suhu)  
<class 'int'>  
>>>
```

2) Float (Bilangan Desimal)

Tipe data Python yang selanjutnya adalah float. Kebalikan dari integer, float khusus digunakan untuk menampung nilai bilangan desimal.

Contoh pemakaian float yaitu:

```
phi = 3.14159  
berat = 50.75  
jumlah = -0.279
```

Sementara ini dia hasilnya setelah dilakukan pengecekan tipe data dengan type():

```
>>> phi = 3.14159  
>>> berat = 50.75  
>>> jumlah = -0.279  
>>> type (phi)  
<class 'float'>  
>>> type (berat)  
<class 'float'>  
>>> type (jumlah)  
<class 'float'>  
>>>
```

3) String (Teks & Karakter)

String adalah tipe data Python untuk menampung nilai teks, seperti huruf, tanda baca, dan karakter spesial lainnya. Ciri-ciri tipe data string adalah diapit tanda petik, baik itu petik satu ('') atau petik dua ("").

Contoh penggunaan string di bahasa pemrograman Python adalah:

```
nama = "mukidi"
```

```
gender = 'L'
```

```
motto = "saya ingin jadi juara ke-1!"
```

Hasil pengecekan tipe datanya adalah sebagai berikut:

```
>>> nama = "mukidi"
>>> gender = 'L'
>>> motto = "saya ingin jadi juara ke-1"
>>> type (nama)
<class 'str'>
>>> type (gender)
<class 'str'>
>>> type (motto)
<class 'str'>
>>>
```

4 Boolean (True & False)

Boolean adalah tipe data Python yang hanya bisa diisi oleh dua nilai, yaitu True dan False. Untuk menulis isian boolean, ada dua aturan yang harus dipatuhi. Yaitu, pakai kapital untuk huruf pertama dan tanpa tanda petik sama sekali.

Contohnya seperti berikut:

```
kanan = True
```

```
kiri = False
```

Jika diperiksa, berikut hasil tipe data di atas:

```
>>> kanan = True
>>> kiri = False
>>> type (kanan)
<class 'bool'>
>>> type (kiri)
<class 'bool'>
>>>
```

5) Tipe Data Khusus (Bilangan Kompleks)

Selain tipe data standar, Python juga sebenarnya menyediakan tipe data khusus, yaitu bilangan kompleks. Tipe data Python yang satu ini bisa Anda manfaatkan untuk memudahkan perhitungan matematika.

Namun untuk menggunakannya, Anda perlu menambahkan fungsi complex(bilangan bulat,bilangan imajiner). Contoh penerapannya di bahasa pemrograman Python adalah sebagai berikut:

```
x = complex (5,6)
```

```
y = complex (4,7)
```

```
x + y
```

Berikut hasil pemeriksaan sekaligus penjumlahan dari tipe data Python:

Dibawah ini diberikan beberapa Contoh Program Sederhana yang Dibuat Menggunakan Python, sebagai berikut :

1) Hello World

Python Hello World adalah program sederhana Python yang pertama. Program ini akan menampilkan tulisan ‘Hello World’ memakai fungsi print. Untuk itu, silakan tulis sintaks berikut:

```
# Mencetak tampilan Hello World!
print('Hello World!')
```

Nah, ini dia hasilnya setelah dijalankan:

```
>>> | Hello World!
```

2) Penjumlahan Dua Angka

Program sederhana Python selanjutnya adalah penjumlahan dua angka. Program ini memanfaatkan fungsi input untuk menerima inputan Anda. Namun, fungsi yang satu ini akan menghasilkan tipe data string. Untuk itu, Anda perlu mengkonversi string tersebut menjadi integer dengan fungsi int. Lalu, gunakan fungsi sum dan operator matematika ‘+’ untuk menjumlahkan dua angka yang diinput. Seperti ini bentuk sintaks Python untuk operasi penjumlahan :

```
# Memasukkan Inputan Angka
angka1 = input('Tulis angka pertama: ')
angka2 = input('Tulis angka kedua: ')

# Mengkonversi Angka lalu Menjumlahkannya
sum = int(angka1) + int(angka2)

# Menampilkan Hasil Penjumlahan
print('Hasil Penjumlahan {0} dan {1} adalah {2}'.format(angka1, angka2,
sum))
```

Jika dijalankan,
akan seperti ini:

```
Tulis angka pertama: 16
Tulis angka kedua: 37
Hasil Penjumlahan 16 dan 37 adalah 53
```

3) Penghitungan Luas Segitiga

Contoh program Python selanjutnya adalah penghitungan luas segitiga. Sama seperti program sebelumnya, program ini memakai fungsi input untuk mengambil inputan Anda.

Dari situ, Anda bisa menghitung luas segitiga sesuai dengan rumus yang ada. Jika sudah, tambahkan fungsi `%0.2f` untuk menampilkan dua angka di belakang koma pada hasil luas segitiga.

Selengkapnya, ini dia sintaks yang digunakan:

```
# Menginput Alas dan Tinggi Segitiga
alas = float(input('Tulis Alas Segitiga: '))
tinggi = float(input('Tulis Tinggi Segitiga: '))

# Hitung Luas Segitiga
luas = (alas * tinggi) / 2

#Menampilkan Hasil Perhitungan
print('Luas Segitiga adalah %0.2f' %luas)
```

Ketika dijalankan, tutorial Python ini akan menghasilkan tampilan berikut:

```
>>> Tulis Alas Segitiga: 34
      Tulis Tinggi Segitiga: 17
      Luas Segitiga adalah 289.00
```

4) Konversi Suhu

Konversi suhu adalah contoh program Python berikutnya. Program ini mengubah suhu dari Celcius menjadi Fahrenheit. Cara membuatnya sangat mudah, Anda cukup memasukkan rumus konversi suhu dari Celcius ke Fahrenheit.

Berikut contoh perintah yang dipakai:

```
# Menginput Suhu dalam Derajat Celcius
celcius = float(input("Tuliskan Suhu dalam Celcius: "))

# Menghitung Suhu dalam Derajat Fahrenheit
fahrenheit = (celcius * 1.8) + 32

#Menampilkan Hasil Konversi Suhu
print("%0.2f Derajat Celcius sama dengan %0.2f Derajat Fahrenheit"
      %(celcius,fahrenheit))
```

Hasil program sederhana Python ini adalah:

```
>>> Tuliskan Suhu dalam Celcius: 23  
23.00 Derajat Celcius sama dengan 73.40 Derajat Fahrenheit
```

5) Kalender Masehi

Contoh program Python yang terakhir adalah kalender masehi. Sesuai namanya, program ini akan menampilkan kalender masehi sesuai periode bulan yang Anda input.

Untuk mempraktikkan tutorial Python ini, Anda perlu mengimpor sebuah library bernama calendar. Lalu untuk memunculkan kalender dengan format bulan, gunakan fungsi month.

Nah, perintahnya adalah sebagai berikut:

```
# Mengimpor Modul Calendar  
import calendar  
  
# Menginput Tahun dan Bulan  
yy = int(input("Masukkan Tahun: "))  
mm = int(input("Masukkan Bulan: "))  
  
# Menampilkan Kalender Bulanan  
print(calendar.month(yy, mm))  
Jika dijalankan, berikut hasil program  
sederhana Python ini:
```

```
Masukkan Tahun: 2022  
Masukkan Bulan: 4  
April 2022  
Mo Tu We Th Fr Sa Su  
1 2 3  
4 5 6 7 8 9 10  
11 12 13 14 15 16 17  
18 19 20 21 22 23 24  
25 26 27 28 29 30
```

Seperti inilah bentuk sintaksis program python. Silahkan berlatih untuk mengembangkan diri Anda.

MODUL PRAKTIKUM

Implementasi ALGORITMA Kedalam Pemrograman DELPHI

Bahasa Pemrograman DELPHI merupakan pengembangan dari bahasa pemrograman PASCAL. Bahasa PASCAL maupun DELPHI mudah untuk dipelajari, mengingat bahasanya mendekati bahasa manusia. Pemrograman DELPHI menyediakan dua model pemrograman yaitu pemrograman konsol dan pemrograman visual.

Untuk praktik Pemanfaatan DELPHI saat ini diarahkan pada **penggunaan aplikasi konsol** (*console application*). Bertujuan agar mahasiswa mampu menerapkan algoritma yang sudah dipelajari sebelumnya dan sebagai alat pembuktian kebenaran susunan instruksi logikanya.

Ingin.....

- Apapun bahasa pemrograman yang digunakan tidak menjadi masalah, tapi harus diperhatikan bahwa konsep algoritma dimengerti dengan baik agar dapat membuat kode program yang rapi dan terstruktur.
- Beda **programmer** yang belajar dengan konsep dan **programmer yang** tidak belajar dengan konsep, terlihat pada kerapuhan dan struktur program yang dibuat.

Struktur Bagan Pascal/Delphi

1) **Bagian Judul Program**

2) **Bagian Deklarasi**

- ✖ Deklarasi Unit TPU {\$APPTYPE CONSOLE} Uses SysUtils;
- ✖ Deklarasi tipe data (TYPE)
- ✖ Deklarasi variabel (VAR)
- ✖ Deklarasi konstanta (CONST)
- ✖ Deklarasi label (LABEL)
- ✖ Deklarasi sub-program (PROCEDURE dan FUNCTION)

3) **Bagian Program Utama**

Berisikan baris –baris perintah (IPO).

Mengenal Delphi

1.1. Pengertian Delphi

Delphi adalah suatu bahasa pemrograman (**development language**) yang digunakan untuk merancang suatu aplikasi program.

a. Kegunaan Delphi

1. Untuk membuat aplikasi windows
2. Untuk merancang aplikasi program berbasis grafis
3. Untuk membuat program berbasis jaringan (client/server)
4. Untuk merancang program .Net (berbasis internet)

b. Keunggulan Delphi

1. IDE (*Integrated Development Environment*) atau lingkungan pengembangan aplikasi sendiri adalah satu dari beberapa keunggulan delphi, didalamnya terdapat menu – menu yang memudahkan kita untuk membuat suatu proyek program.
2. Proses Kompilasi cepat, pada saat aplikasi yang kita buat dijalankan pada Delphi, maka secara otomatis akan dibaca sebagai sebuah program, tanpa dijalankan terpisah.
3. Mudah digunakan, source kode delphi yang merupakan turunan dari pascal, sehingga tidak diperlukan suatu penyesuaian lagi.
4. Bersifat multi purpose, artinya bahasa pemrograman Delphi dapat digunakan untuk mengembangkan berbagai keperluan pengembangan aplikasi.

c. Sejarah Borland Delphi

1. Delphi versi 1 (berjalan pada windows 3.1 atau windows 16 bit)
2. Delphi versi 2 (Berjalan pada windows 95 atau delphi 32 bit)
3. Delphi versi 3 (berjalan pada windows 95 keatas dengan tambahan fitur internet atau web)
4. Perkembangan selanjutnya diikuti dengan Delphi versi 4, 5 dan 6.
5. Versi terkini dari delphi adalah versi 7 dengan tambahan fitur .net dengan tambahan file XML

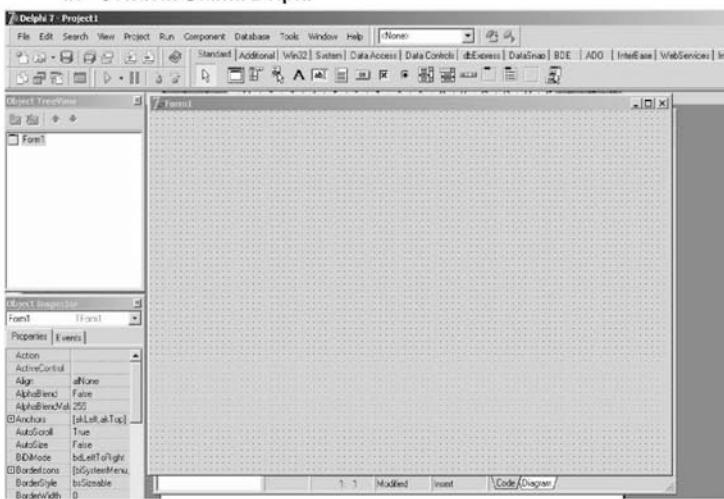
1.2. IDE DELPHI

a. Langkah – langkah mengaktifkan Delphi

- Klik start
- Pilih All program
- Pilih borland delphi
- Pilih dan klik Delphi 7

Atau bisa dengan masuk lewat menu Run (kombinasi tombol Start + R) kemudian mengetikkan perintah **delphi32**

b. Jendela Utama Delphi

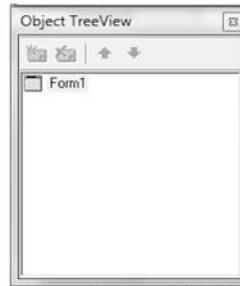


Gambar 1.1 Jendela Utama Delphi

c. Bagian – bagian dari Jendela Delphi

1. Object Tree View

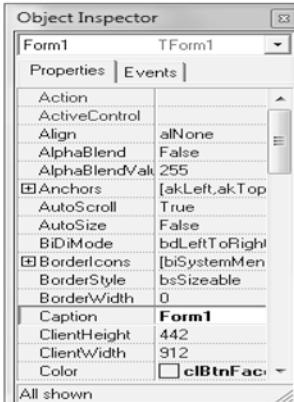
Merupakan sebuah diagram pohon yang menggambarkan hubungan logis menghubungkan semua komponen yang terdapat dalam suatu proyek program. Komponen tersebut meliputi form, modul atau frame. Fungsinya digunakan untuk menampilkan seluruh daftar komponen program dalam sebuah aplikasi program sesuai dengan penempatannya.



Gambar 1.2 Jendela Object Tree View

1. Object Inspector

Merupakan jendela yang digunakan untuk mengatur tampilan komponen pada form, misal bagaimana mengubah tulisan button pada command button menjadi Simpan, atau menghapus tulisan pada label dan mengganti nama menjadi Nama Mahasiswa atau memberikan perintah tertentu pada sebuah komponen sehingga ada interaksi ketika program dijalankan.



Gambar 1.3 Jendela Object Inspector

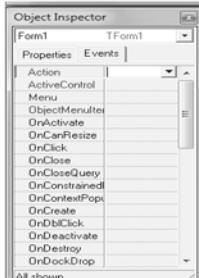
Secara Umum Object Inspector terbagi menjadi 2, yaitu :

a. Properties

Digunakan untuk mengatur tampilan pada sebuah komponen baik itu meliputi penggantian nama, warna, jenis huruf, border dan lain-lain.

b. Events

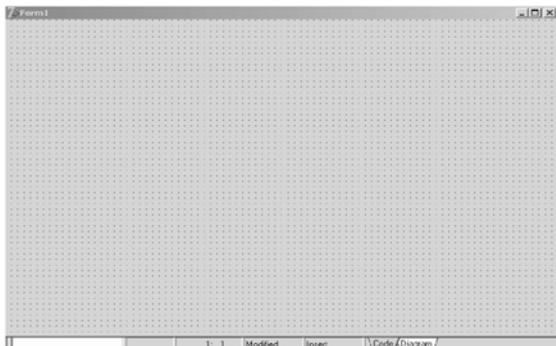
Merupakan jendela properties yang digunakan untuk memberikan fungsi yang lebih detail dari fungsi sebenarnya. Misalnya ketika tombol Simpan di klik maka program akan menjalankan perintah penyimpanan data. Dari kalimat tersebut ada event click untuk mengeksekusi sebuah tombol simpan. Perintah event click tersebut dapat diberikan melalui jendela events.



Gambar 1.4 Jendela Properties & Events

2. Form Designer

Merupakan tempat yang digunakan untuk merancang semua aplikasi program yang diambil dari komponen palette.



Gambar 1.5. Jendela Form Designer

3. Component Pallete

Merupakan kumpulan icon yang digunakan untuk merancang suatu aplikasi pada untuk membentuk sebuah aplikasi user interface. Dalam komponen palette semua icon dikelompokan dalam berbagai komponen sesuai dengan fungsi dan kegunaannya.



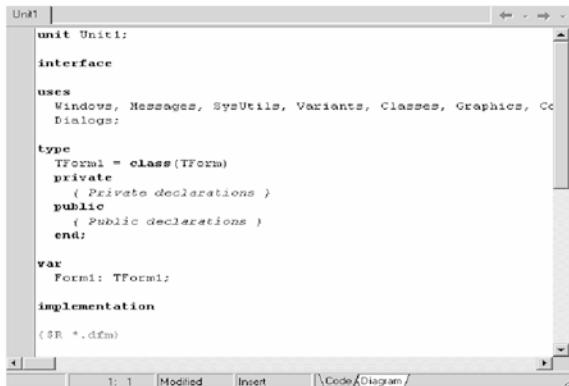
Gambar 1.6. Jendela Komponen Pallete

No	Icon	Name	Fungsi
1	Pointer	Pointer	Mengembalikan fungsi mouse ke defaultnya
2		Frame	Membentuk suatu frame terhadap obyek yang ada didalamnya
3		Main menu	Membuat menu Utama
4		Popup Menus	
5		label	Hanya untuk menampilkan Teks
6		Edit	Untuk menampilkan dan input data (1 baris)

No	Icon	Name	Fungsi
7		Memo	Sama seperti edit tetapi mempunyai kapasitas lebih besar (lebih dari 1 baris)
8		Button	Digunakan untuk melakukan eksekusi terhadap suatu proses
9		Checkbox	Digunakan untuk menentukan pilihan lebih dari satu
10		Radio Button	Digunakan untuk menentukan pilihan, tetapi hanya satu pilihan yang bisa digunakan
11		List Box	Menampilkan pilihan dalam bentuk list
12		Combo Box	Menampilkan pilihan dalam bentuk popup
13		Scroll Bar	Merupakan icon yang berupa garis status
14		Group Box	Digunakan untuk mengelompokan suatu icon
15		Radio Group	Digunakan untuk mengelompokan pilihan

4. Code Editor

Bagian dari delphi yang digunakan untuk menuliskan kode program. Pada bagian code editor terdapat 3 bagian utama yaitu = bagian *paling kiri* yang berisi berupa *angka* menunjukkan baris dan kolom. Keterangan *modified* menunjukkan bahwa telah terjadi modifikasi terhadap baris program. Dan *paling kanan* menunjukkan status keyboard tentang tombol *insert* atau *over write*.



```

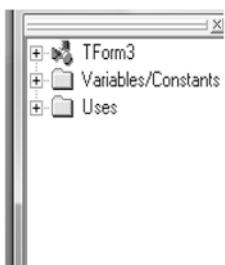
Unit1
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Dialogs;
type
  TForm1 = class(TForm)
    private
      { Private declarations }
    public
      { Public declarations }
    end;
var
  Form1: TForm1;
implementation
{$R *.dfm}

```

Gambar 1.7 Jendela Code Editor

5. Code Explorer

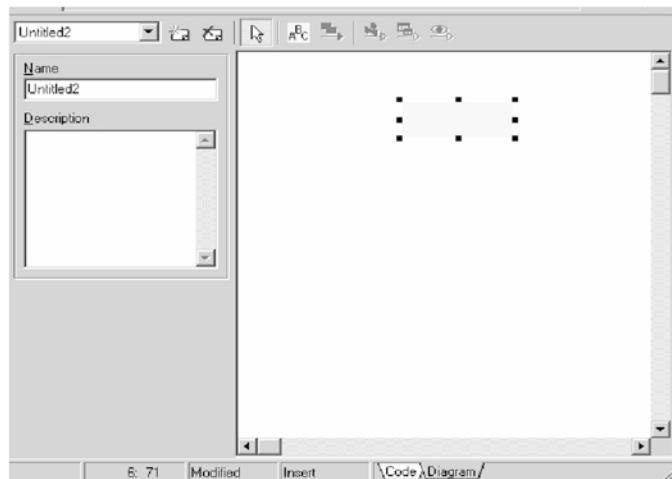
Jendela yang digunakan untuk menampilkan seluruh *variabel*, *type*, dan *routine* yang didefinisikan pada sebuah unit.



Gambar 1.8 Jendela Code Explorer

6. Code Diagram

Merupakan fasilitas pada delphi yang digunakan untuk mendesain sebuah diagram atas komponen – komponen yang digunakan dalam seatu rancangan aplikasi.



Gambar 1.9. Jendela Code Diagram

d. File-file Pendukung Project Delphi

1. File Project (.Dpr)

File ini disimpan dengan ber-ekstenion .dpr. File ini berisi informasi mengenai seluruh proyek program

2. File Unit (.Pas)

File ini merupakan kumpulan dari barisan kode program yang terdapat di jendela code editor, baik itu yang dituliskan oleh programmer maupun oleh system. Extention file ini adalah .pas

File Unit dibagi menjadi 3 :

a. Unit Form

Unit ini dibuat secara otomatis oleh Delphi. Satu unit untuk satu form.

b. Unit component

Unit yang terbentuk pada saat membuat komponen baru.

c. Unit umum

Unit yang dibuat untuk tipe data, variable, procedure dan class yang dapat digunakan dan diaplikasikan.

3. File Form (.Dfm)

Berisi tentang seluruh informasi yang ada kaitannya dengan form yang dibuat, meliputi tinggi, lebar, posisi form atau tentang komponen didalamnya. Penggunaan file ini tidak dianjurkan karena untuk pengaturan sudah disediakan **object inspector** sebagai media pengaturan semua komponen.

e. Aplikasi Console

Aplikasi console merupakan sebuah aplikasi yang tidak mempunyai form. Aplikasi ini berbasis text mode dan umumnya berjalan pada command prompt. Delphi menyediakan fasilitas untuk membangun aplikasi console. Aplikasi ini digunakan untuk membuat sourcecode object pascal.

Memulai Aplikasi Console yaitu dengan cara mengklik menu File-New-Other-Console Application.

↳ Beberapa aturan penulisan dalam Pascal/Delphi :

a. Akhir sebuah program Pascal ditandai dengan tanda baca titik (.) setelah END yang paling akhir.

b. Tanda titik koma (:) merupakan pemisah antar instruksi satu dengan lainnya.

c. Beberapa statement boleh ditulis menjadi satu baris dipisahkan dengan tanda baca titik koma (,)

Contoh : simpan := akhir; akhir := simpan + awal;

d. Baris komentar diletakkan diantara tanda (*) dan *) atau diantara tanda { dan }

Contoh : Var rerata : real; (*nilai rata-rata*) Nill : real; {nilai ujian}

↳ Input dan output (Standar I/O).

Dalam bahasa Pascal/Delphi untuk keperluan input (membaca input) digunakan identifier standar READ atau READLN. Sedangkan untuk keperluan output (mencetak output) digunakan identifier standar WRITE atau WRITELN.

1. READ (variabel input) ;

READLN (variabel input) ;

2. WRITE (variabel output) ;

WRITELN (variabel output);

Contoh

```
Var nama : string{15};  
    Begin  
        write(_nama : _);  
        readln(nama);  
        write(_nama saya:'nama ); readln;  
    end.
```

Dalam hal ini Anda akan belajar menulis program sesuai algoritma dengan menggunakan aplikasi console Delphi. Perhatikan dan ketik rapi listing berikut dan lihat hasilnya.

Kasus -1:

Buat algoritma dan program delphi yang dapat membaca 2 buah bilangan, dapat menentukan dan mencetak bilangan terbesar dari 2 buah bilangan tersebut. Kedua bilangan tidak boleh sama.

Pemecahan masalah

1. Input ada 2 buah bilangan (mis: X,Y)
2. Proses
 - a) Baca data : bilangan X, Y
 - b) Bandingkan jika $X > Y$ maka cetak keterangan $X > Y$ jika tidak maka cetak $Y > X$
3. Output : Keterangan bilangan terbesar dari 2 buah bilangan tersebut.
4. selesai

ALGORITMA (PSEODOCODE)

Program_Dua_Angka;

Var

X, Y : Angka (Int)

Begin

Read (X,Y)

if $X > Y$ then

 write ("Bilangan X besar dari bilangan Y")

else

 write ("Bilangan Y besar dari bilangan X")

endif

End.

Menggunakan Program Delphi

1. Pastikan Anda sudah menginstal DELPHI
2. Siapkan folder untuk menyimpan data program agar tidak tersebar dan dapat dengan mudah dalam pencarian program selanjutnya.
3. Saat Anda menggunakan **Delphi 2010** maka pilih **File New, other** kemudian tampil kotak dialog **New Items**, pilih **Delphi Project** dan item **Console Application**
4. Ketiklah program dibawah ini:

```
Program Latih01;
{$APPTYPE CONSOLE}
Uses SysUtils;
Var
  X, Y: Int;
Begin
  Write('Masukkan Bilangan A :') ; readLn (X);
  Write('Masukkan Bilangan B :') ; readLn (Y);
  If (A > B) then
    Writeln ('Bilangan A paling besar daripada B')
  Else { (B > A)} then
    Writeln ('Bilangan B paling besar daripada A');
  Readln;
End.
```

5. Setelah itu simpanlah program dengan langkah berikut ini : pilih **Save As**, pilih folder data Anda kemudian nama **Latih-1** dan pilih **save**.
6. Lakukan Compile untuk memastikan baris perintah sudah ditulis dengan benar dengan menekan tombol **CTRL+F9**
7. Menjalankan program Delphi dengan memilih menu **run/F9**
8. Mengedit program delphi dengan memilih menu **file open** nama file **Latih-1** klik open (maka program sudah dapat diperbaiki).

```

Program Latih02;
{ $APPTYPE CONSOLE }
Uses SysUtils;
Var
  A, B, C : LongInt;
Begin
  Write('Masukkan Bilangan A :) ; readln (A);
  Write('Masukkan Bilangan B :) ; readln (B);
  Write('Masukkan Bilangan C :) ; readln (C);
  If (A > B) and (A > C) then
    Writeln ('Bilangan A paling besar daripada B dan C')
  Else if (B > A) and (B > C) then
    Writeln ('Bilangan B paling besar daripada A dan C ')
  Else
    Writeln ('Bilangan C paling besar daripada A dan B ');
  Readln;
End.

```

Program ini dapat membaca tinggi, berat badan dan menghitung berat badan ideal seseorang.

```

Program Latih03;
{ $APPTYPE CONSOLE }
Uses SysUtils;
Var
  Nama, ket: string [45];
  Tinggi, Berat_badan, Berat_ideal :real;
Begin
  Write( 'Nama      :') ; readln (Nama);
  Write( 'Tinggi     :') ; readln (Tinggi);
  Write( 'Berat Badan :') ; readln (Berat_Badan);

  {menghitung berat badan ideal}
  Berat_ideal :=(Tinggi-110)*1.1;

  {menguji berat badan ideal}
  If Berat_badan > 1.1 * Berat_Ideal then
    Ket:='Ternyata anda KURUS, makan teratur ya!'
  Else
    Ket:='Nach ... ini baru berat badan ideal';
  {endif}

  {mencetak hasil}
  Writeln('Saudara', Nama, ' ',ket); readln;
End.

```

Perulangan FOR tersarang

Perulangan FOR tersarang adalah perulangan FOR yang berada pada perulangan yang lainnya. Perulangan yang lebih dalam akan diproses terlebih dahulu sampai habis, kemudian perulangan yang lebih luar baru akan bertambah, mengerjakan perulangan yang lebih dalam lagi mulai dari nilai awalnya dan seterusnya.

Contoh

Var a, b : integer;

begin

```
    for a:=1 to 3 Do
        begin
            for b:= 1 to 2 Do
                write (a:4, b:2);
            writeln;
        end;
```

end.

Hasil :

```
    1 1   1 2
    2 1   2 2
    3 1   3 2
```

Perulangan WHILE-DO

Penyeleksian kondisi digunakan untuk agar program dapat menyeleksi kondisi, sehingga program dapat menentukan tindakan apa yang harus dikerjakan, tergantung dari kondisi yang diseleksi tersebut. Perulangan WHILE-DO tidak dilakukan jika kondisi tidak terpenuhi.

Contoh :

VAR i : INTEGER;

Begin

```
    i := 0;
    while i< 5 Do
        begin
            write (i:3);
            i:=i+1; {sama jika menggunakan inc(i)}
        end;
```

end.

Hasilnya : 0, 1, 2, 3, 4

Perulangan REPEAT-UNTIL

REPEAT-UNTIL digunakan untuk mengulang statement-statement atau blok statement sampai (UNTIL) kondisi yang diseleksi di UNTIL tidak terpenuhi. Sintak dari statement ini adalah :

Contoh

```
Var i: interger;
Begin
    I:=0;
    Repeat
        I:=I+1; writeln(I);
    Until i=5;
End.
```

```
Program Kasus1;
{SAPP TYPE CONSOLE};
Uses SysUtils;
Var
    NIK    :string[10];
    Nama   :string[40];
    Gol    :string[3];
    Gapok:LongInt;
Begin
    Write ('No.Induk Karyawan :');Readln (nik);
    Write ('Nama Karyawan :');Readln (nama);
    Write ('Golongan :');Readln (gol);
    If gol='I' Then
        Gapok:=650000
    Else
        If gol='II' Then
            Gapok:=850000
        Else
            If gol='III' Then
                Gapok:=1250000
            Else
                gapok:=1500000;
    Write ('Besar gaji pokok :,gapok:7);readLn;WRITELN;
End.
```

Perulangan FOR tersarang

Perulangan FOR tersarang adalah perulangan FOR yang berada pada perulangan yang lainnya. Perulangan yang lebih dalam akan diproses terlebih dahulu sampai habis, kemudian perulangan yang lebih luar baru akan bertambah, mengerjakan perulangan yang lebih dalam lagi mulai dari nilai awalnya dan seterusnya.

Contoh

Var a, b : integer;

begin

```
    for a:=1 to 3 Do
        begin
            for b:= 1 to 2 Do
                write (a:4, b:2);
            writeln;end;
```

end.

Hasil :

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

Perulangan WHILE-DO

Penyeleksian kondisi digunakan untuk agar program dapat menyeleksi kondisi, sehingga program dapat menentukan tindakan apa yang harus dikerjakan, tergantung dari kondisi yang diseleksi tersebut. Perulangan WHILE-DO tidak dilakukan jika kondisi tidak terpenuhi.

Contoh :

VAR i : INTEGER;

Begin

```
    i := 0;
    while i< 5 Do
        begin
            write (i:3);
            i:=i+1; {sama jika menggunakan inc(i)}
        end;
```

end.

Hasilnya : 0, 1, 2, 3, 4

Perulangan REPEAT-UNTIL.

REPEAT-UNTIL digunakan untuk mengulang statement-statement atau blok statement sampai (UNTIL) kondisi yang diseleksi di UNTIL tidak terpenuhi. Sintak dari statement ini adalah :

Contoh

```
Var i: interger;
Begin
    I:=0;
    Repeat
        I:=I+1; writeln(I);
    Until i=5;
End.
```

```
Program Kasus1;
{SAPPTYPE CONSOLE};
Uses SysUtils;
Var
    NIK :string[10];
    Nama :string[40];
    Gol :string[3];
    Gapok:LongInt;
Begin
    Write ('No.Induk Karyawan :');Readln (nik);
    Write ('Nama Karyawan :');Readln (nama);
    Write ('Golongan :');Readln (gol);
    If gol='I' Then
        Gapok:=650000
    Else
        If gol='II' Then
            Gapok:=850000
        Else
            If gol='III' Then
                Gapok:=1250000
            Else
                gapok:=1500000;

    Write ('Besar gaji pokok :,gapok:7);readLn;WRITELN;
End.
```

Susun sebuah algoritma dan program delphi yang membaca data seorang mahasiswa yang mengikuti kuliah algoritma berupa Stb, nama dan nilai (angka) dan output berupa Stb, Nama, Nilai dan Predikat lulus bila Nilai>85=A, Nilai>70=B, Nilai >55=C, Nilai>40=D dan Nilai<39=E.

Algoritma

- Deklarasi Data mhs dalam record terdiri dari stb, nama (string) dan nilai (angka/int) serta predikat (char)
- Deskripsi
 - ▢ Input record mahasiswa (Stb, Nama dan Nilai)
 - ▢ Proses Nilai sbb:
 - Jika $85 \geq \text{Nilai} \leq 100$ maka predikat 'A'
 - Jika $70 \geq \text{Nilai} \leq 84$ maka predikat 'B'
 - Jika $55 \geq \text{Nilai} \leq 69$ maka predikat 'C'
 - Jika $40 \geq \text{Nilai} \leq 54$ maka predikat 'D'
 - Selain itu maka predikat 'E'
 - ▢ Output:
 - Stambuk
 - Nama
 - Nilai
 - Predikat

Program Latih07;

```
{$APPTYPE CONSOLE}
```

```
Uses SysUtils;
```

```
Var
```

```
    Nim :string[10];Nama :string[40];
    Nilai :Integer;Predikat:Char(1);
Begin
    {Input Data}
    WriteLn ('Menghitung Nilai Akhir Mahasiswa');WriteLn;
    Write('Masukkan Nim :'); readln (Nim);
    Write('Masukkan Nama :'); readln (Nama);
    Write('Masukkan Nilai :'); readln (Nilai);
    {Proses Data}
    Case Nilai of
        40 .. 54 : Predikat :='D';
        55 .. 69 : Predikat :='C';
        70 .. 84 : Predikat :='B';
        85 .. 100 : Predikat :='A';
    Else
        Predikat :='E';
    End;
    {Output Data}
    WriteLn ('Hasil Akhir :, Predikat);ReadLn;
End.
```

Latihan

1. Tuliskan apa yang Anda pahami dari Konsep Dasar Pemrograman
2. Tuliskan 2 contoh dari logika pemrograman dari kegiatan sehari-hari
3. Tuliskan pengertian dari ALGORITMA
4. Gambarkan flowchart aktifitas “Sedia Payung Sebelum Hujan”
5. Buat pseudocode untuk aktifitas di depan Traffic Light
6. Sebutkan 3 Konsep Dasar dari Struktur Algoritma
7. Buat flowchart dan listing program untuk menghitung luas dan isi tabung jika diketahui jari-jari dan tinggi tabung tersebut. rumus :
 - a. Luas = $\pi r^2 + 2 \times \pi r \times r$
 - b. Isi = $\pi r^2 \times t$

Soal 8

Buat flowchart dan program Delphi untuk menampilkan data nama, nik dan gaji total seorang karyawan. Untuk menghitung **gaji total = gaji pokok + tunjangan + status**. Sedangkan **Gaji pokok** di input 2.500.000 untuk semua golongan, **status** jika menikah 1.500.000 dan jika belum menikah 250.000, serta **tunjangan** disesuaikan dengan golongan karyawan, sebagai berikut :

- Jika golongan = ‘I’ maka tunjangan = 2.500.000
- Jika golongan = ‘II’ maka tunjangan = 3.250.000
- Jika golongan = ‘III’ maka tunjangan = 4.150.000
- Jika golongan = ‘IV’ maka tunjangan = 4.500.000

Gunakan perintah IF dan perintah PROCEDURE untuk soal diatas.

Soal 9

Buat program delphi untuk suatu aturan kelulusan sbb:

Data yang di input adalah Nilai tugas, nilai hadir, nilai UTS dan nilai UAS diterapkan persyaratan sebagai berikut :

- Jika nilai ujian tengah semester (UTS) lebih besar dari 70 maka siswa dinyatakan lulus dan Nilai jika kurang dari nilai UTS tersebut dinyatakan Mengulang
- **NA=(hadir X 15%) + (tgs X 25%) + (mid X 25%) + (final X 35%).**

Gunakan perintah tipe data record untuk soal

Soal 10

Buat program Delphi untuk mencetak nama BULAN dalam setahun. Nama BULAN di cetak erdasarkan no bulan yang diberikan, sbb :

Input no bulan (tipe data numeric)

Proses

- a) Baca data : no bulan
- b) Uji No bulan
 - No hari = 1 maka isi nama hari = “Januari”
 - No hari = 2 maka isi nama hari = “Februari”
 - Dst ... No hari = 12 maka isi nama hari = “Desember”

Output

Keterangan nama hari (tipe data string).

Jika menggunakan penanggalan Islam, bagaimana koding yang dihasilkan.

Soal.

1. Tuliskan apa yang Anda pahami dari Konsep Dasar Pemrograman
2. Tuliskan 2 contoh dari logika pemrograman dari kegiatan sehari-hari
3. Tuliskan pengertian dari ALGORITMA
4. Gambarkan flowchart aktifitas “Sedia Payung Sebelum Hujan”
5. Buat pseudocode untuk aktifitas di depan Traffic Light
6. Sebutkan 3 Konsep Dasar dari Struktur Algoritma
7. Buat flowchart dan listing program untuk menghitung luas dan isi tabung jika diketahui jari-jari dan tinggi tabung tersebut. rumus :
 - a. Luas = $\pi r^2 + 2 \times \pi r \times t$
 - b. Isi = $\pi r^2 \times t$

Soal 8

Buat flowchart dan program Delphi untuk menampilkan data nama, nik dan gaji total seorang karyawan. Untuk menghitung **gaji total = gaji pokok + tunjangan + status**.

Gaji pokok di input 2.500.000 untuk semua golongan, **status** jika menikah 1.500000 dan jika belum menikah 250.000, sedangkan **tunjangan** disesuaikan dengan golongan karyawan, sebagai berikut :

- Jika golongan = ‘I’ maka tunjangan = 2.500.000
- Jika golongan = ‘II’ maka tunjangan = 3.250.000
- Jika golongan = ‘III’ maka tunjangan = 4.150.000
- Jika golongan = ‘IV’ maka tunjangan = 4.500.000

Gunakan perintah IF dan perintah PROCEDURE untuk soal diatas.

Soal 9

Buat program delphi untuk suatu aturan kelulusan pada mata kuliah Algoritma & Pemrograman I

Data yang di input adalah Nilai tugas, nilai hadir, nilai UTS dan nilai UAS diterapkan persyaratan sebagai berikut :

- Jika nilai ujian tengah semester (UTS) lebih besar dari 70 maka siswa dinyatakan lulus dan Nilai jika kurang dari nilai UTS tersebut dinyatakan Mengulang
- **NA=(hadir X 15%) + (tgs X 25%) + (mid X 25%) + (final X 35%).**

Gunakan perintah tipe data record untuk soal

Soal 10

Buat program Delphi untuk mencetak nama BULAN dalam setahun. Nama BULAN di cetak erdasarkan no bulan yang diberikan, sbb :

Input no bulan (tipe data numeric)

Proses

- a) Baca data : no bulan
- b) Uji No bulan
 - No hari = 1 maka isi nama hari = “Januari”
 - No hari = 2 maka isi nama hari = “Februari”
 - Dst ... No hari = 12 maka isi nama hari = “Desember”

Output

Keterangan nama hari (tipe data string)

Jika menggunakan penanggalan Islam, bagaimana koding yang dihasilkan.

Soal 11

Buat flowchart dan listing program untuk menghitung konversi mata uang rupiah ke dalam mata uang dollar dan yen dimana mata uang rupiah sebagai data masukan. Rumus konversi mata uang :

- a. Dollar = Rupiah/14.000
- b. Yen = Rupiah / 5.500

Soal 12

Buat flowchart dan listing program untuk sebuah aturan untuk anak usia sekolah adalah sebagai berikut, jika usia anak sama atau lebih dari 7 tahun maka anak tersebut diperbolehkan masuk sekolah dasar dan apabila kurang dari 7 tahun maka anak tersebut tidak bersyarat.

Soal 13

Buat flowchart dan listing program delphi untuk suatu perhitungan sebagai berikut :

nilai $P = X + Y$. Jika P positif, maka $Q = X * Y$,
sedangkan jika negatif maka nilai $Q = X/Y$.
Tentukan nilai P dan Q .

Soal 14

Buat flowchart dan listing program untuk mencetak suku deret aritmatika menggunakan rumus $N = N + 4$ dan hasil OUTPUT adalah 3, 7, 11, 15, 19, 23, 27, 31, 35.

Soal 15

Buat program untuk suatu sebuah usaha fotokopi mempunyai aturan sebagai berikut :

- jika yang fotokopi statusnya adalah langganan, maka berapa lembar pun dia fotokopi, harga perlombarnya Rp. 75,-
- jika yang fotokopi bukan langganan, maka jika dia fotokopi kurang dari 100 lembar harga perlombarnya Rp. 100,-. Sedangkan jika lebih atau sama dengan 100 lembar maka harga perlombarnya Rp. 85,-.

Soal 16

Buat flowchart dan listing program dari sebuah univeritas yang memberlakukan yudisium cumlaude untuk mahasiswa yang lulus dengan IPK lebih besar sama dengan 3.5 dan masa kuliah tidak lebih dari 4 tahun.
(Input : IPK dan masa kuliah)

Gunakan Perintah PEMILIHAN (IF .. Then... Else atau Case ...Of)

Soal 17

Buat flowchart dan program Delphi yang dapat menampilkan Nama, Umur dan menyeleksi kelompok umur, yakni :

- a. Umur 0 – 5 tahun adalah ‘kelompok Balita’
- b. Umur 6 – 17 tahun adalah “kelompok Anak-anak”
- c. Umur 18 – 25 tahun adalah “kelompok Remaja”
- d. Umur 26 – 65 tahun adalah “kelompok Dewasa”
- e. Umur 66 tahun keatas adalah “kelompok Manula”

Program menampilkan keterangan kelompok umur.

Gunakan Perintah PEMILIHAN (IF .. Then... Else atau Case ...Of)

Soal 18

Buat flowchart dan listing program untuk mencetak suku deret aritmatika bilangan dari 1 s.d. 50 dengan rumus $N = N + 2$.

Output program adalah bilangan ganjil.

Gunakan Perintah PERULANGAN (FOR .. atau WHILE)

Soal 19

Buat flowchart dan listing program untuk menampilkan deret 20 bilangan dari variabel A=1 dan B=0, dimana $A=A+2$ dan $B=A \times A$.

Bagaimana hasilnya?

Gunakan Perintah PERULANGAN (FOR .. atau WHILE).

Perhatikan koding berikut ini untuk memahami program subrutin (program modular)

```
program Coba1;
{$APPTYPE CONSOLE}
uses
SysUtils;
var
x,y,z:byte; {variabel global}

procedure kali (a,b:byte); {parameter formal}
var
c:byte; {variabel lokal}
begin
c:= a*b;{proses perhitungan}
Writeln('Hasil a x b : ',c);//pencetakan hasil proses
end;

procedure garis;
begin
writeln('-----');
end;
function bagi(a,b:byte):byte;
begin
Bagi:=a div b;{nama proses = nama fungsi}
end;

begin
writeln('Program Matematika Sederhana');
garis;//pemanggilan procedure garis
write('Nilai 1 : ');readln(x);
write('Nilai 2 : ');readln(y);
garis;
kali(x,y);
garis;
writeln('Hasil a / b : ',bagi(x,y));
garis;
readln;
end.
```

Dari bentuk diatas, Buat flowchart dan program untuk menghitung

- a. luas segitiga,
- b. luas bujursangkar,
- c. luas persegi panjang dan
- d. ucapan program selesai.

Ke-4 sub program tersebut dijalankan pada program utama dan tidak berparameter.

Daftar Pustaka

Abdul Kadir, *Pemograman Turbo Pascal*, Andi, Yogyakarta, 2006

Adam Saputra, *Buku Sakti Pemrograman Python*, Anak Hebat Indonesia, Yogyakarta, 2022

Asep Kosasih, *Algoritma & Pemrograman dengan bahasa Delphi 5.0*, Yrama Widya, Bandung, 2006

Budi Rahardjo, *Teknik Pemrograman Pascal*, Informatika, Bandung, 2005.

Budi Sutedjo & Michael AN, *Algoritma & Teknik Pemrograman*, Andi, Yogyakarta, 2000

Cheltenham Computer Training, *C Programming*. United Kingdom (akses www.cctglobal.com), 1997.

Deitel & Deitel, *C How to Program 3rd edition*. prentice Hall ; New Jersey, 2001

Fathul Wahid, *Dasar-Dasar Algoritma dan Pemrograman*, Andi, Yogyakarta, 2004.

Harwikarya, dkk, *Dasar Pemrograman 2 (Implementasi Menggunakan Java, C++, MatLab & Pascal)*, Andi & Universitas Mercu Buana, Yogyakarta, 2017.

Jogiyanto, *Turbo Pascal*, Andi, Yogyakarta, 1989.

Joseph Teguh Santoso, *Proyek Coding dengan Python*, Universitas STEKOM, Semarang, 2022

Jubilee Enterprise, *Belajar Pemrograman Python Untuk Guru & Murid*, Elex Media Komputindo, Jakarta, 2022

Kusnassriyanto, *Belajar Pemrograman Delphi*, Modula, Bandung, 2011

P. Insap Santosa, *Struktur Data Menggunakan Turbo Pascal 6.0*, Andi, Yogyakarta, 2001

Rinaldi Munir, *Algoritma dan Pemrograman dalam bahasa Pascal dan C*, Informatika, Bandung 2007

Rosihan Ari Yuana, *Konsep dan Implementasi Pemrograman Python: Kasus Big Data*, Lokomedia, 2019

Suarga, Algoritma & Pemrograman, Andi, Yogyakarta, 2012

Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

Wendi Zarman & Mochamad Fajar Wicaksono, *Implementasi Algoritma Dalam Bahasa Python*, Informatika, 2022

<https://www.levatra.com/2017/02/pengertian-tipe-data-variabel-dan-operator-pemrograman.html>

DASAF PEMROGRAMA

Ade Hastuty Hasyim



Ade Hastuty Hasyim, ST., S.Kom., MT,
Lahir di Ujungpandang, 20 Januari 1972.
Pendidikan dasar diselesaikan di SD.
Pembangunan III Tauladan Jenderal
Sudirman Ujungpandang (1985),
pendidikan menengah ditempuh di
SMP Negeri 5 Ujungpandang (1988),
pendidikan menengah atas di SMA Negeri
1 Ujungpandang (1991). Pendidikan

S1 ditempuh di Jurusan Teknik Kimia Fak. Teknologi Industri
Universitas Muslim Indonesia Makassar (1996), Pendidikan
S1 di Jurusan Manajemen Informatika STMIK Makassar (2000),
Pendidikan S2 di Prodi Teknik Elektro Universitas Hasanuddin
Makassar (2006), Pendidikan S3 Prodi Teknologi Pendidikan
Universitas Negeri Jakarta. PNS pada Kementerian Agama RI
sebagai Dosen Ilmu Komputer di IAIN Parepare.

Keluarga : Suami Azis Mallombasi Djalle DM, dan memiliki 3 orang anak yakni
Azizah Nurul Fadhillah Djalle DC (2002), Alif Manrompai Djalle DM (2005) dan
Akbar Manrompai Hasmi Djalle DR (2013).

Karya Buku

1. Algoritma dan Pemrograman (2014) ISBN 978-602-70304-6-6
2. Desain WEB (2015) ISBN 978-602-70304-7-3
3. Melek Teknologi (2019) ISBN 978-623-72028-2-0

ISBN 978-602-51846-4-2



9 78602 184642



CV. Bangun Bumitama