BAHAN AJAR TERSELEKSI

ALGORITMA DAN PEMROGRAMAN

(Aplikasi Bahasa Pemrograman PASCAL)

Yuhendra, S.T., M.T., Dr.Eng

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI PADANG

2013

DAFTAR ISI

Halaman

RENCANA PROGRAM DAN KEGIATAN PEMBELAJARAN SEMESTER
(RPKPS)

1. Mata Kuliah : ALGORITMA DAN PEMROGRAMAN
2. Kode Mata Kuliah / SKS : TIS2223 / 3
3. Semester : 2 (Dua)
4. Sifat Mata Kuliah : Wajib
5. Prasyarat : -
6. **Deskripsi Mata Kuliah**

   Matakuliah ini membahas tentang bagaimana cara mengatasi permasalahan-permasalahan yang ada dengan membuat algoritma pemrograman dan kemudian mengimplementasikannya ke dalam bahasa pemrograman (aplikasi bahasa pemrograman pascal)
7. **Standar Kompetensi**
   - Mahasiswa dapat memahami struktur dasar pembuatan alagoritma, data pemrograman dan dapat menerapkannya dalam pembuatan program yang efektif dan efisien.
   - Mahasiswa dapat membuat algoritma untuk memudahkan pembuatan program yang terstruktur.
   - Mahasiswa mempunyai pengalalman dalam praktek pemrograman dengan mampu merancang algoritma dengan struktur data yang sesuai.
8. **Kompetensi Dasar**

   Memahami perlunya dan menguasai algoritma dan flowchart dalam rangka membuat urut-urutan penyelesaian masalah lewat program dan dapat membuat algoritma, flowchart dan meninplementasikan kedalam bahasa pemrograman.

9. **Tujuan Pembelajaran**
   - Memperkenalkan konsep dan dasar-dasar algoritma,
   - Menjelaskan tentang teknik penyajian algoritma ke bentuk notasi flowchart atau pseudo code

- Mampu mengimplementasikan bebera kasus dari notasi algoritmik ke notasi bahasa pemrogaraman

10. Outcome Pembelajaran:

a. *Knowledge and Understanding*
   - Mengerti dan memahami konsep-konsep dasar algoritma, teknik penyajian algoritma, tipe-tipe data, teknik-teknik pemrograman seperti runtunan (sequence), pemilihan (selection), pengulangan (looping program), pembuatan prosedur dan fungsi, pemrosesan larik (array).
   - Mahasiswa termotivasi dan mampu mengikuti perkembangan terkini teknologi bahasa pemrogramn terkait dalam aplikasi-aplikasi dalam bentuk software (perangkat lunak).
   - Algoritma dan pemrograman merupakan kunci utama dalam membuat sebuah produk software dan aplikasi-aplikasi lainnya.

b. *Intellectual Skills*
   - Mahasiswa mampu menjelaskan konsep dasar pembuatan alagoritma, data pemrograman dan dapat menerapkannya dalam pembuatan program yang efektif dan efisien.
   - Mahasiswa mampu mentranlasikan notasi algoritma ke notasi bahasa pemrograman
   - Mahasiswa mampu merancang algoritma dengan struktur data yang sesuai dan dapat mengidentifikasikan kesalahan-kesahan dalam pembuatan program itu sendiri.
   - Mahasiswa mampu memecahkan contoh-contoh kasus yang sederhana sampai yang sulit. Melalui contoh kasus, diharapkan kemampuan dalam menulis algotitma dan merancang pemrograman meningkat.

c. *Practical Skills*

Practical skill akan didapatkan melalui praktikum pemrograman dengan mampu merancang algoritma dengan struktur data yang sesuai. Semakin banyak melakukan latihan-latihan pemrograman akan semakin terlatih cara pikir dalam menyelesaikan masalah.

11. Materi Pembelajaran

1. Pengantar Algoritma dan Pemrograman
2. Teknik Penyajian Algoritma
3. Konsep Dasar Pemrograman dan Tipe Data
4. Dasar-dasar Algoritma
5. Aturan Penulisan Teks Algoritma
6. Runtunan (Sequence)
7. Pemilhan (Selection)
8. Pengulangan (Looping Program)
9. Prosedur dan Fungsi
10. Larik (Array)

12. Jadual Tabel Kegiatan Mingguan

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
XII & XIII	**Pengulangan (Looping Program)** - Struktur For To Do - Struktur For Down To Do - Struktur WHILE Do - Struktur REPEAT Until	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis
XIV	**Prosedur dan Fungsi** - Mendefinisikan Prosedur - Pemanggilan Prosedur - Program dengan Prosedur atau Tanpa Prosedur - Prosedur dengan Parameter atau Tanpa Parameter	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

	- Mendefenisikan Fungsi - Pemanggilan Fungsi - Prosedur atau Fungsi			
XV	**Larik (Array)** - Array Satu Dimensi - Cara mengcu elemen array Satu dimensi - Array Dua Dimensi - Cara mengcu elemen array Dua dimensi - Array Multidimensi - Tipe Data Bentukan	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis
XVI	**UJIAN AKHIR SEMESTER (UAS)**			

13. Evaluasi Hasil Pembelajaran

Evaluasi hasil pembelajaran pada mata kuliah ini dilakukan dengan penilalaain sebagai berikut:

- Penilaian terhadap Quiz , Tugas mandiri dan Kelompok
- Penilaian terhadap Praktikum
- Mengadakan Ujian Tengan Semester (UTS)
- Mengadakan Ujian Akhir Semester (UAS)

Distribusi Penilaian:

Komponen	**Bobot Nilai**
Quiz, Tugas Mandiri, Tugas Kelompok	15%
Praktikum	25%
Ujian Tengah Semester	30%
Ujian Akhir Semester	30%

Skor Penilaian Akhir:

No	Nilai Mahasiswa	Bobot Nilai
1	A	80 – 100
2	B	65 – 79
3	C	55 – 64
4	D	45 – 54
5	E	< 45

14. Bahan, sumber informasi dan referensi

Sumber Informasi

- Konsultasi langsung atau melalui e-mail
- Mahasiswa didorong untuk mempergunakan kemajuan teknologi informasi (Internet, Newsgroup, Perpustakaan Online, Handbook) untuk mendapatkan bahan-bahan penunjang.

Referensi

1. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung : Informatika. 2007
2. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
3. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006

SATUAN ACARA PERKULIAHAN

(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: 1 dan 2

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B. Kompetensi Dasar

Mahasiswa mampu mengenal lingkungan, bahasa pemrograman Pascal. Dapat menggunakan bahas pemrograman pascal untuk pemecahan masalahnya.

C. Pokok Bahasan

Pengantar Algoritma dan Pemrograman

D. Sub Pokok Bahasan

- Pengantar Algoritma
- Dasar-dasar Allgoritma
- Penyajian Algoritma

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Menjelaskan kontrak perkuliahan 2. Penjelasan materi, relevansi referensi 3. Memberi motivasi	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	1. Menjelaskan tentang Pengertian Algoritma 2. Menjelaskan perbedaan Algoritma dan Program 3. Menjelaskan Mekanisme Pelaksanaan Algoritma oleh Pemroses 4. Menjelaskan perbedaan Belajar Memprogram dan Belajar Bahasa Pemrograman 5. Menjelaskan Cara Penyajian Algoritm	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan, dan memberikan kesimpulan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang Algoritma dan Pemrograman

2. Pertanyaan Langsung

   - Jekaskan pengertian Algoritma secara umum
   - Jelaskan perbedaan Algoritma dengan Program

G. Referensi

4. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung : Informatika. 2007
5. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
6. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
7. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : 1 dan 2

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
I	**Pendahuluan** - Kontrak Perkuliahan - Penjelasan Materi Teoritik dan Praktiukm - Daftar Pustaka dan Relevansi sumber-sumber lain.	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB I
Pengantar Algoritma dan 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 dikeluarkan 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 ditranslasika 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 mengkonversikannya 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. Algoritma adalah deskripsi dari suatu pola tingkah laku yang dinyatakan secara primitif yaitu aksi-aksi yang didefenisikan sebelumnya dan diberi nama, dan diasumsikan sebelumnya bahwa aksi-aksi 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.

Tabel 1.1. Contoh-Contoh Algoritma dalam Kehidupan Sehari-hari

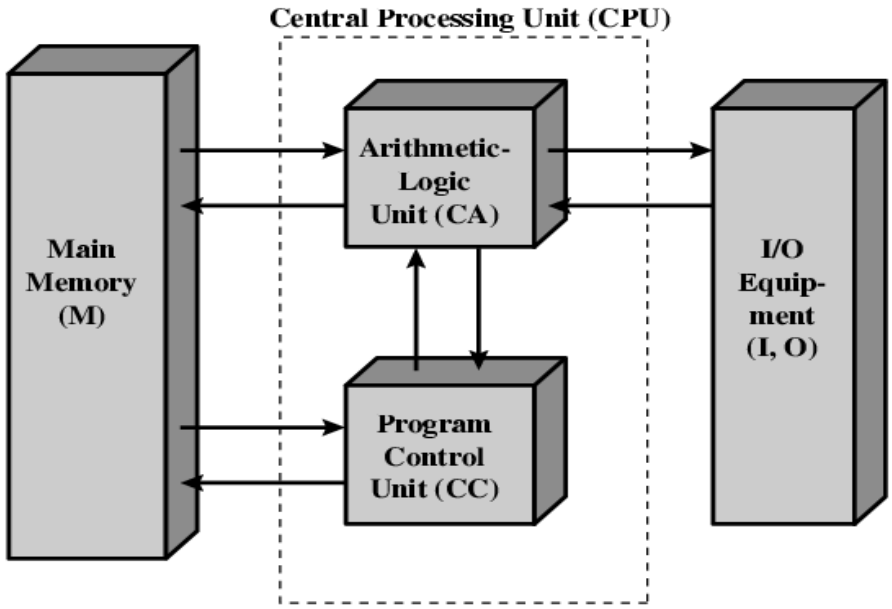
No.	Proses	Algoritma	Contoh Langkah dalam Algoritma
1	Membuat kue	Resep kue	Masukkan telur ke dalam wajan, kocok sampai mengembang
2	Membuat pakaian	Pola pakaian	Gunting kain dari pinggir kiri bawah ke arah kanan sejauh 5 cm
3	Merakit mobil	Panduan merakit	Sambungkan komponen A dengan komponen B
4	Kegiatan 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

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 program disebut **pernyataan** atau **instruksi.** Jadi, program

tersusun atas sederetan instruksi. Bila suatu instruksi dilaksanakan, maka operasioperasi 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 mengingatingat. Yang disimpan di dalam memori adalah program (berisi operasioperasi 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.



Gambar 1.1 Komponen-komponen Utama Komputer

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 operasioperasi yang bersesuaian dengan instruksi tersebut. Bila suatu operasi memerlukan data, data dibaca dari piranti masukan, disimpan di dalam memori lalu dikirim ke CPU untuk 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 dan Belajar Bahasa Pemrograman

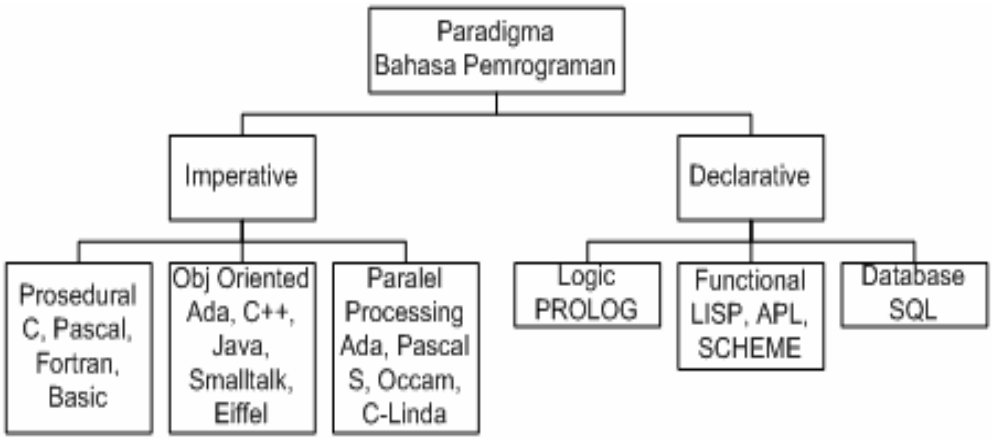
Belajar memprogram tidak sama dengan belajar bahasa pemrograman. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami. Sedangkan belajar bahasa pemrograman berarti belajar memakai suatu bahasa aturan-aturan 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 pemrogram, 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. Bahasabahasa 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 (*realibility*). 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 sesuatu yang 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 instruksinya 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 subbagiansubbagian 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 katakana 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 pemrogramanini 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 presedural 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)

3. Algoritma konversi suhu dalam derajat Celcius ke derajat Kalvin

Penyelesaian menggunakan *pseudocode*:

Input (Celcius)

Kalvin = Celcius + 273

Output (Kalvin)

.

SATUAN ACARA PERKULIAHAN
(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: 3

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

a. Mengerti macam-macam komponen flowchart beserta fungsinya.
b. Mengetahui dan memahami Kaidah-kaidah dalam dalam pembuatan flowchart

C. Pokok Bahasan

Teknik Penyajian Algoritma

D. Sub Pokok Bahasan

- Pengantar dan defenisi Flowchart
- Dasar-dasar komponen flowchart dan fungsinya
- Aturan dalam penulisan flowchart

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya dengan memberikan quiz 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	1. Menjelaskan tentang komponen-komponen dalam pembuatan flowchart 2. Menjelaskan fungsi dari masing-masing komponen 3. Memberikan sejumlah contoh-contoh permasalahan sederhana yang dinaotasikan ke bentuk flowchart.	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan, dan memberikan kesimpulan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang Flowchart

2. Pertanyaan Langsung
   - Jekaskan Fungsi dari komponen flowchart
   - Membuat flowchart berdasarkan permasalahan .

G. Referensi

1. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung : Informatika. 2007
2. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
3. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
4. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : 3

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
III	**Teknik Penyajian Algoritma** - Defenisi Flowchart - Macam-macam komponen flowchart - Kaidah-Kaidah Umum Pembuatan *Flowchart* Program	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer, LCD, papan tulis dan alat tulis

BAB II
Teknik Penyajian Algoritma

2.1 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).

2.2 Structure English dan Pseudocode

Structure English merupakan alat yang cukup efisienuntuk menggambarkan suatu algoritma. Basis dari structure english adalah bahasa inggris, tetapi juga bisa digunakan bahasa indonesia, sedangkan pseudocodeberarti kode yang mirip dengan kode pemrograman sebenarnya. Pseudocodeberasal dari kata pseudo yang berarti imitasi/mirip/menyerupai dan code yang berarti program. Pseudocodeberbasis pada kode program yang sesungguhnya seperti Pascal, C, C++. Pseudocodelebih 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
```

2.3 **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 berfikir 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.

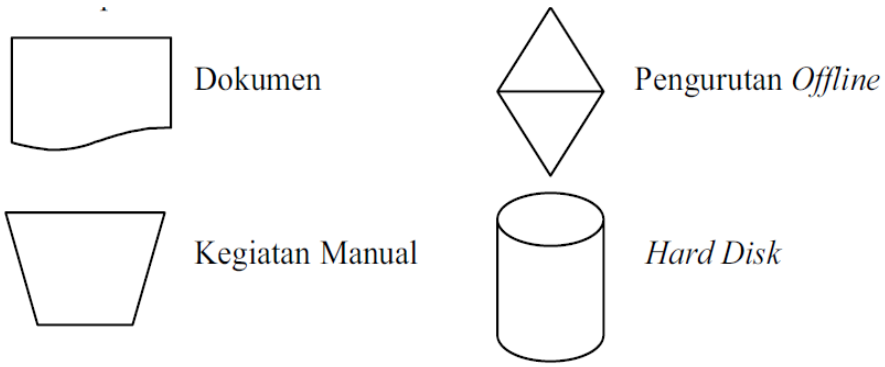
2.3.1 Defenisi Flowchart

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:

❶ *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:



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

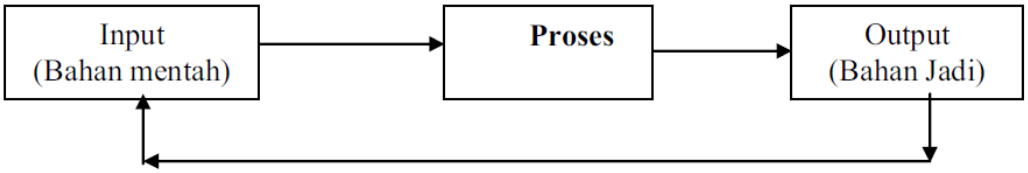
2.3.2 Kaidah-Kaidah Umum Pembuatan *Flowchart* Program

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*



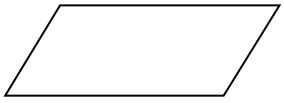
Untuk pengolahan data dengan komputer, urutan dasar pemecahan suatu masalah:

- 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 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.

Gambar 2.2. Simbol Proses

☞ ***Simbol Output***

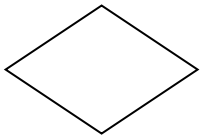
Simbol output digambarkan dengan bangun seperti Gambar 4. Simbol ini digunakan untuk melambangkan kegiatan penampilan output. Dalam simbol ini, kita dapat menuliskan semua output yang harus ditampilkan oleh program. Sama seperti aturan pada dua simbol sebelumnya, penulisan dapat dilakukan secara satu 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". Jika pemeriksaan 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.



Gambar 2.4 Simbol Percabangan

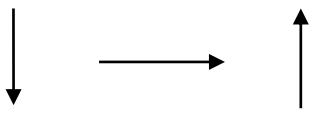
☞ ***Simbol Prosedur***

Simbol prosedur digambarkan dengan bangun seperti Gambar 6. Simbol ini berperan sebagai blok pembangun dari suatu program. 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 symbol 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

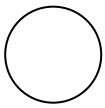
☞ **Simbol Terminator**

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.

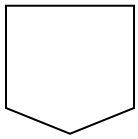
Gambar 2.7 Simbol Terminator

☞ **Simbol Konektor**

Simbol konektor digunakan untuk menghubungkan suatu langkah dengan langkah lain dalam sebuah flowchart dengan keadaan on page atau off page. On page connectordigunakan 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. 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 2.8a Simbol On-Page Connector



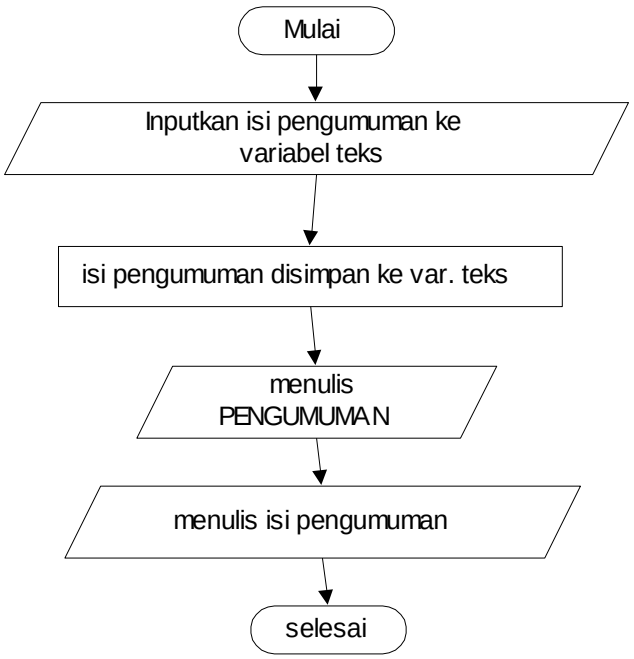
Gambar 28.b Simbol Off-Page Connector

Soal dan Pembahasan

Kasus 1 : Membuat Teks Pengumuman

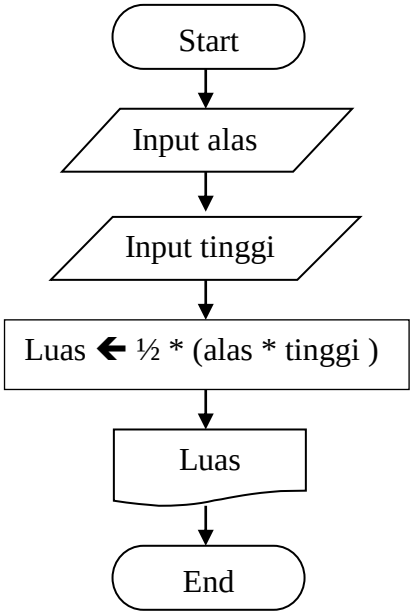
1. Mulai
2. Inputkan 'isi pengumuman'
3. Simpan isi pengumuman ke variabel 'teks'
4. Outputkan(tulis dilayar) tulisan "PENGUMUMAN"
5. Outputkan isi pengumuman yang tersimpan dalam variable 'teks'
6. Selesai

Dibawah ini akan ditunjukkan diagram alir programnya / flowchart :

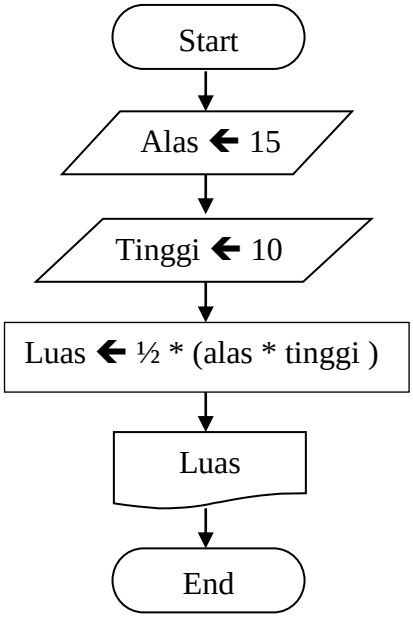


Kasus 2: Menghitung luas segitiga

- Membaca data dengan menginputkan secara tidak langsung



- Membaca data dengan menginputkan secara langsung



SATUAN ACARA PERKULIAHAN

(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: 4 dan 5

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

a. Mengenal Elemen-elemen pemrograman Pascal
b. Mengetahui dan memahami Kaidah-kaidah pemograman Pascal dalam bentuk struktur dan tipe-tipe data

C. Pokok Bahasan

Konsep Dasar Pemrograman dan Tipe Data

D. Sub Pokok Bahasan

- Pengatar Pemrograman Pascal
- Tipe dan Struktur Data
- Struktur Pemrograman

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya dengan memberikan quiz 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	1. Menjelaskan tentang elemen-elemen pada Program Pascal beserta fungsi dari masing-masing elemen. 2. Menjelaskan Struktur dan Tipe-tipe Data 3. Memberikan sejumlah contoh-contoh permasalahan sederhana yang dinotasikan ke bentuk struktur pemrograman Pascal	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan, dan memberikan Tugas	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang Pemrograman Komputer, terutama bahasa pemrograman Pascal

2. Pertanyaan Langsung
   - Jekaskan Tipe-tipe data beserta contohnya
   - Membuat program sederhana dengan menggunakan kaidah struktur pemrograman Pascal .

G. Referensi

5. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I.* Bandung: Informatika. 2007
6. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
7. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
8. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN
(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : 4 dan 5

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
IV & V	**Konsep Dasar Pemrograman dan Tipe Data** - Pengatar Pemrograman Pascal - Tipe dan Struktur Data - Struktur Pemrograman	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB III
Konsep Dasar Pemrograman Pascal

3.1 Elemen Pemrograman Pascal

Pascal adalah bahasa pemrograman yang pertama kali di buat 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. 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.

✍ ***Reserved Word***

Reserved word adalah kata – kata yang tidak dapat dijadikan menjadi identifier (pengenal), karena kata – kat tersebut sudah mempunyai arti tersendiri dalam Turbo Pascal. 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.

✍ ***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

✍ ***Type***

Bentuk umum:

Type

Pengenal = tipe data;

…….

Pengenal = tipe data;

✍ ***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;

✍ ***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;

✍ *Tipe Data*

Tipe atau jenis data dalam Turbo Pascal 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 kedal 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 kedalm 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 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 empat yang termasuk kedalam 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 dianggap255 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 dalam mebuat 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 kelabel 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

3.2 Struktur Pemrograman Pascal

Struktur dasar dalam pemrograman pascal :

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

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.

```
PROGRAM
informatika;
BEGIN
```

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 :

{ { 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).

Soal dan Pembahasan :

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}
```

SATUAN ACARA PERKULIAHAN
(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: 6

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

a. Mengenal struktur dasar pembuatan algoritma
b. Mengetahui dan memahami Kaidah-kaidah algoritma dalam bentuk runtunan, pemilihan dan pengulang.

C. Pokok Bahasan

Dasar-dasar Algoritma

D. Sub Pokok Bahasan

- Struktur Dasar Algoritma
- Pengantar Runtunan, Pemilihan dan Pengulangan
- Strategi perancangan algoritma

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Membahas tugas mandiri 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	1. Menjelaskan tentang sruktur dasar dalam pembuatan algoritma 2. Menjelaskan konsep dasar runtunan, pemilihan dan pengulangan beserta contoh. 3. Membuat notasi algoritmik ke bentuk programming berdasarkan kasus yang diberikan.	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan, dan memberikan Tugas	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

2. Pertanyaan tidak langsung

   Meminta kepada mahasiswa untuk meberikan komentar tentang struktur dasar algoritma

2. Pertanyaan Langsung

   - Membuat algoritma sederhana dengan menggunakan kaidah dasar pembuatan algoritma.

G. Referensi

9. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung: Informatika. 2007
10. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
11. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
12. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : 6

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
VI	**Dasar-dasar Algoritma** - Struktu Dasar Algoritma - Runtunan, Pemilihan dan Pengulangan - *Strategi Perancangan Puncak-Turun (Top-Down*	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB IV
Struktur Dasar Algoritma

4.1 Pengantar

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

4.2 Struktur Dasar Algoritma

Dalam sebuah algoritma langkah-langkah penyelesaianmasalahnya dapat berupa struktur urut (sequence), struktur pemilihan (selection), dan struktur pengulangan (repetition). Ketiga jenis langkah tersebut membentuk konstrukti 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 traffic light berwarna merah then
        berhenti
else
        if lampu traffic 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 for-do 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
        lihat data mahasiswa pada posisi berikutnya
until *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 repeat-until, 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
<u>while</u> data mahasiswa dengan NIM 006 belum ditemukan dan data mahasiswa terakhir belum terlampaui <u>do</u>
if data mahasiswa memiliki NIM 006 then
ubah data alamatnya
else
lihat data mahasiswa pada posisi berikutnya
end if
end while

4.3 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

SATUAN ACARA PERKULIAHAN

(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: VII

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

a. Mengenal aturan-aturan dalam penulisan teks algoritma
b. Mengetahui dan memahami cara translasi dari teks algoritma ke dalam teks program Pascal

C. Pokok Bahasan

Aturan Penulisan Teks Algoritma

D. Sub Pokok Bahasan

- Teks Alogoritma
- Translasi Teks Algoritma ke Program Pascal

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	4. Menjelaskan tentang bagaimana cara penulisan teks algoritma. 5. Menjelaskan struktur teks algoritma berupa kepala algoritma yang terdiri dari deklarasi dan deskripsi. 6. Menjelaskan cara translasi teks algoritma ke notasi bahasa pemrograman Pascal	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan	Membuat rangkuman	Papan tulis dan alat tulis

F. Evaluasi

3. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang aturan penulisan teks algoritma

2. Pertanyaan Langsung

   - Bagaimana melakukan translasi dari notasi teks algoritma ke bentuk notasi bahasa pemrograman .

G. Referensi

13. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung: Informatika. 2007
14. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
15. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
16. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : 7

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
VII	**Aturan Penulisan Teks Algoritma** - Struktur Teks Algoritma - Translasi Teks Algoritma ke dalam Teks Program Pascal - Contoh-contoh kasus	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer, LCD, papan tulis dan alat tulis

BAB V
Aturan Penulisan Teks Algoritma

5.1 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

5.2 Susunan Teks Algoritma

Teks algoritma disusun oleh tiga bagian (blok): bagain 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 bersis 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 lingkaranke 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

{Jjudul 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

Fahrenheit = 9/5 x (C + 32)

Reamur = 4/5 x (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)

Fahrenheit = 5/9 * (Celsius +32)

Reamur = 4/5 * (Celsius +32)

Write (Celsius, Fahrenheit, Reamur)

SATUAN ACARA PERKULIAHAN

(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: IX

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat menjelaskan konsep struktur dasar runtunan dan menerapkan dalam pembuatan program

C. Pokok Bahasan

Runtunan (Sequence)

D. Sub Pokok Bahasan

- Konsep Runtunan
- Pengaruh urutan instruksi

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya dengan memberikan quiz 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	1. Menjelaskan tentang konsep runtunan 2. Menjelaskan pengaruh urutan dalam runtunan 3. Memberikan sejumlah contoh-contoh runtunan	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang runtunan

2. Pertanyaan Langsung

   - Membuat program sederhana dengan menggunakan konsep rununan.

G. Referensi

17. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung: Informatika. 2007

18. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
19. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
20. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN
(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : IX

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
IX	**Runtunan (Sequence)** - Konsep Runtunan - Pengaruh Ururan Instruksi	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB VI

Runtunan (Sequence)

6.1 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 tehadap 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.

```
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)
```

Hasil algoritma di atas sama saja jika urutan C←A+B dan D←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)
       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 cara mempertukarkan nilai *A* dan *B* ? Misalnya, sebelum pertukaran nilai *A*=8, nilai *B*=5, maka setelah pertukaran, nilai *A*=5 dan *B*=8.

```
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)
```

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

{ 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 }

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

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 x 3600) + (5 x 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:

Tunjangan = 0.2 * gaji pokok

Pajak = 0.15 * (gaji pokok + tunjangan)

Gaji bersih = gaji pokok + tunjangan – 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.
```

SATUAN ACARA PERKULIAHAN

(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: X dan XI

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

- Menjelaskan konsep struktur dasar seleksi kondisi
- Menjelaskan *statement* yang digunakan dalam penyeleksian kondisi dan menerapkannya dalam pembuatan algoritma

C. Pokok Bahasan

Pemilihan (Selection)

D. Sub Pokok Bahasan

- Statemen IF - Then
- Statemen IF-Then-Else
- Statemen Case – Of
- Statemen Case – Of - Else

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya dengan memberikan quiz 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	Menjelaskan tentang struktur pemilihan (selection) seperti Statemen IF – Then, Statemen IF-Then-Else, Statemen Case – Of dan Statemen Case – Of – Else beserta contoh kasus.	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang pemilihan

2. Pertanyaan Langsung

   - Membuat program sederhana dengan menggunakan konsep pemilihan

G. Referensi

21. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung: Informatika. 2007
22. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
23. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
24. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : 10 dan 11

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
X & XI	**Pemilhan (Selection)** - Struktur If- Then - Struktur If- Then-Else - Struktur Case - of - Struktur Case - of – Else	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB VII
Pemilihan (Condition)

7.1 Pengantar

Struktur runtunan 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 pemilahan 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 : =, ≠, <, >, ≤, ≥, dan operator uner not.

Contoh-contoh ekspresi boolean :

x > y

a ≠ 10

m = n

p ≤ q

a + b > 1

str = 'itb'

k mod 4 = 0

ketemu = true

not berhenti

(x > 0) and (y < 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.

7.2 Struktur If- Then dan If- Then-Else

✍ 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) di atas 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 tersebutmerupakan 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.
```

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 merupan 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 merupan 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.
```

✍ 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
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
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 ≤ 0°C), cair (0 < T < 100), atau gas (T > 100).

Penyelesaian

Misalkan suhu air adalah T.

Analisis kasus :

Kasus 1 : T ≤ 0, maka tulis “padat”

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

Kasus 3 : T ≥ 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.
end.
```

7.3 Struktur Case - of dan Case - of - 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, 12	31
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←31
              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.
```

SATUAN ACARA PERKULIAHAN

(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: XII dan XIII

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

a. Menjelaskan pengertian pengulangan proses program
b. Mengerti dan memahami konsep kounter dan akumulator serta penerapannya dalam pembuatan program
c. Menjelaskan statement yang digunakan dalam pengulangan proses program dan menerapkannya dalam pembuatan program

C. Pokok Bahasan

Perulangan (Looping Program)

D. Sub Pokok Bahasan

- Statemen For To Do
- Statemen For Downto Do
- Statemen While Do
- Statemen Repeat Until

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya dengan memberikan quiz 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	Menjelaskan tentang struktur pemilihan (selection) seperti Statemen For To Do, Statemen For Downto Do dan Statemen While Do, Statemen Repeat Until beserta contoh kasus.	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang Pengulangan

2. Pertanyaan Langsung

   - Membuat program sederhana dengan menggunakan konsep Pengulangan

G. Referensi

25. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung : Informatika. 2007
26. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
27. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
28. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : XII Dan XIII

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
XII & XIII	**Pengulangan (Looping Program)** - Struktur For To Do - Struktur For Down To Do - Struktur WHILE Do - Struktur REPEAT Until	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB VIII
Pengulangan (Looping Program)

8.1 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 :

```
<inisialisasi>
awal pengulangan
    badan pengulangan
akhir pengulangan
<terminasi>
```

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 :

❶ ***Struktur FOR***
❷ ***Struktur WHILE***
❸ ***Struktur REPEAT***

8.2 Struktur FOR

Struktur pengulangan FOR digunakan untuk menghasilkan pengulangan sejumlah kali yang dispesifikasikan. Jumlah pengulangan diketahui atau dapat ditentukan sebelum eksekusi. Untuk mencacah sudah berapa kali pengulangan dilakukan, kita memerlukan sebuah peubah (variable) pencacah (counter). Peubah 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 tejadi 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 adalah sebagai berikut :

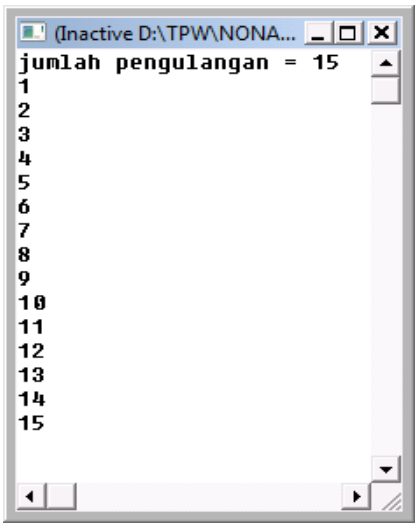
Hasil Output Program:

```
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 pencacah pengulangan lebih besar dari nilai awalnya (1).

Jika N = 1, pengulangan yang terjadi adalah 1 kali, karena 1 – 1 + 1 = 1.

For menurun :

```
for pencacah←nilai_akhir downto nilai_awal 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_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, pencacah diinisialisasi dengan nilai_akhir. Nilai pencacah secara otomatis berkurang satu setiap kali aksi diulang, sampai akhirnya nilai pencacah sama dengan nilai_awal.

v. Jumlah pengulangan yang terjadi adalah nilai_awal-nilai_akhir+1.

Contoh analisis :

Algorima 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:=10 downto 0 do
         writeln(k);
     write('Go!!!!');      { roket meluncur }
end.
```

Struktur WHILE

Bentuk umum :

```
while kondisi do
       aksi
endwhile
```

Aksi (atau runtunan aksi) akan dilaksanakanberulangkali 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

```
Algoritma JUMLAH_DATA;
{ menghitung jumlah seluruh nilai bilangan bulat
positif yang dibaca dari piranti masukan. akhir
pembacaan data: -99 }

DEKLARASI
   x : integer      { data yang dibaca }
   jumlah : integer { pencatat jumlah seluruh data }

DESKRIPSI
     jumlah←0       {inisialisasi penjumlah bilangan}
     read(x)
     while x ≠ -99 do
         jumlah←jumlah + x
         read(x)
     endwhile
{ kondisi di akhir pengulangan: x = -99}
write(jumlah)       {terminasi}
```

```
program JUMLAH_DATA;
{ menghitung jumlah seluruh nilai bilangan bulat positif yang dibaca dari
piranti masukan. akhir pembacaan data: -99 }

uses wincrt;

(* DEKLARASI *)
var
   x, jumlah : integer;


(* DESKRIPSI *)
begin
     jumlah:=0;
     write('Ketikkan sembarang bilangan bulat ( -99 mengakhiri )');
     readln(x);
     while x <> -99 do
           begin
                jumlah:=jumlah + x;
                write('Ketikkan sembarang bilangan bulat ( -99 mengakhiri )');
                readln(x);
     end;

     writeln('Jumlah seluruh nilai = ',jumlah);
end.
```

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 dicetek 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 dicetek 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.
```

SATUAN ACARA PERKULIAHAN
(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: XIV

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

- Menjelaskan konsep dasar dan definisi prosedur dan fungsi
- Mengerti dan memahami cara deklarasi dan pemanggilan prosedur dan fungsi
- Menjelaskan ruang lingkup variabel dan cara pengiriman parameter
- Membuat algoritma yang memuat prosedur dan fungsi

C. Pokok Bahasan

Prosedur dan Fungsi

D. Sub Pokok Bahasan

- Mendefenisikan Prosedur dan Fungsi
- Cara Pemanggilan Prosedur dan Fungsi
- Mendefenisikan Prosedur dan Fungsi

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1.Mereview materi sebelumnya 2.Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	Menjelaskan tentang konsep prosedur dan fungsi Menjelaskan cara mendefinisikan dan memanggil program dengan prosedur atau fungsi	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang prosedur dan fungsi

2. Pertanyaan Langsung

   - Membuat program sederhana dengan menggunakan konsep prosedur dan fungsi

G. Referensi

29. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung : Informatika. 2007
30. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
31. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
32. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN
(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : XIV

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
XIV	**Prosedur dan Fungsi** - Mendefinisikan Prosedur - Pemanggilan Prosedur - Program dengan Prosedur atau Tanpa Prosedur - Prosedur dengan Parameter atau Tanpa Parameter - Mendefenisikan Fungsi - Pemanggilan Fungsi - Prosedur atau Fungsi	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer, LCD, papan tulis dan alat tulis

BAB IX

Prosedur dan Fungsi

9.1 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. Untuk itu, Pascal telah menyediakan dua jenis subprogram, yaitu procedure dan function (prosedur dan fungsi). Dengan modular programming, program lebih mudah dibaca dan dimengerti. Selain itu, pembenahan program dan penelusuran jalannya program (debugging) menjadi lebih mudah sebab dapat langsung diketahui subprogram mana yang berjalan tidak sesuai dengan yang diharapkan.

9.2 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. 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 :

```
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}
```

Cara mendeklarasikan sebuah prosedur adalah sebagai
berikut :

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

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

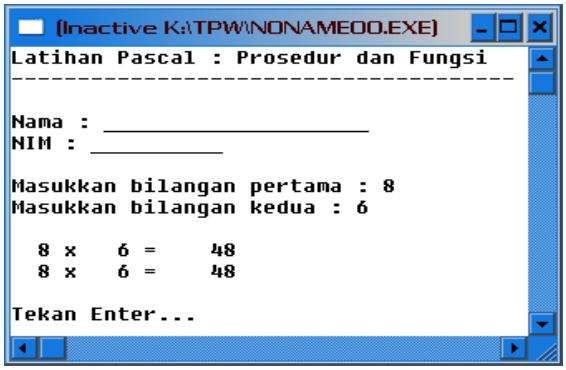
Untuk lebih memahami penggunaan prosedur dalam Pascal, perhatikan contoh program di bawah ini :

Program **Prosedur;**
uses **wincrt;**
var

```
Bil_1, Bil_2, Hasil : integer;
procedure Awal;
begin
Writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
Writeln('-------------------------------------');
Writeln;
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 di atas. Dua prosedur terakhir memiliki kemiripan, bedanya hanya
pada jumlah parameter dan variabel hasil perkaliannya. Untuk lebih jelas, jalankan program dan perhatikan apa yang dilakukan oleh dua prosedur tersebut maka akan nampak perbedaan keduanya.

9.3 F u n g s i

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.

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

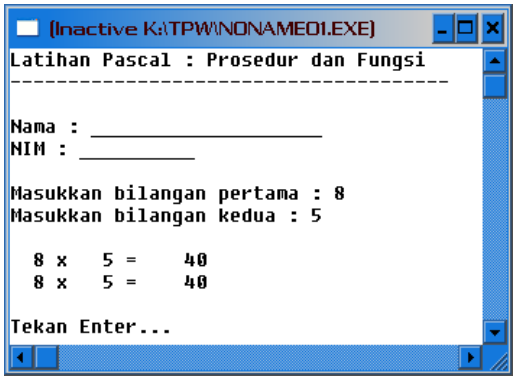
Perbedaan utama antara prosedur dan fungsi adalah dalam menghasilkan keluaran.
Walaupun prosedur bisa menghasilkan nilai keluaran, tetapi nilai tersebut tidak dapat
diambil secara langsung, melainkan harus diambil melalui parameter referensi.
Sedangkan keluaran dari fungsi dapat diambil langsung dari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

```
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;
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:



Perhatikan program di atas. 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 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);

9.3. 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 di atas, terdapat pemanggilan kembali terhadap prosedur Z, sehingga prosedur di atas 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.

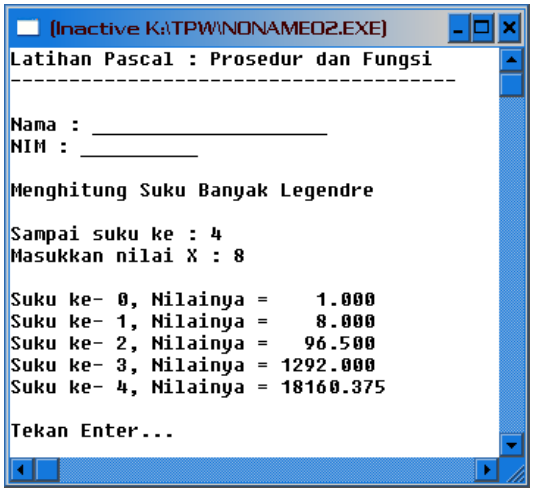
Dengan adanya rekursi ini, banyak algoritma komputer menjadi lebih mudah dibuat programnya. Berikut ini adalah program menghitung suku banyak Legendre, salah satu contoh perhitungan yang dapat diselesaikan dengan menggunakan rekursi :

```
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...');
```

```
Readln;
end.
```

Hasil Output Program:



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

SATUAN ACARA PERKULIAHAN
(SAP)

Mata Kuliah	: Algoritma & Pemrograman Komputer
Kode Mata Kuliah	: TIS2223
SKS	: 3
Semester	: 2
Waktu Pertemuan	: 2x3x50 Menit
Pertemuan	: XV

A. Standar Kompetensi

Setelah perkuliahan ini, mahasiswa dapat memecahkan masalah dengan algoritma yang berbasis pada bahasa Pascal dan dapat mengimplementasikan konsep dasar pemrograman terstruktur

B.Kompetensi Dasar

Mahasiswa dapat:

- Menjelaskan pengertian dan deklarasi array
- Membuat algoritma yang memuat operasi matriks

C. Pokok Bahasan

Array (Larik)

D. Sub Pokok Bahasan

- Array Satu Dimensi
- Cara mengcu elemen array Satu dimensi
- Array Dua Dimensi
- Cara mengcu elemen array Dua dimensi
- Array Multidimensi
- Tipe Data Bentukan

E. Kegiatan Belajar Mengajar

Tahap Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat
Pendahuluan	1. Mereview materi sebelumnya 2. Menjelaskan materi perkuliahan yang akan dipelajari	Mendengarkan dan memberikan komentar	Komputer, LCD, papan tulis dan alat tulis
Penyajian	1. Menjelaskan tentang konsep Array satu dimensi, dua dimensi dan multi dimensi 2. Menjelaskan cara elemen array satu, dua dan multidimensi	Berperan aktif, memperhatikan, mencatat, membrikan komentar dan mengajukan pertanyaan.	Komputer, LCD, papan tulis dan alat tulis
Penutup	Mengajukan pertanyaan	Membuat rangkuman Mengerjakan tugas rumah	Papan tulis dan alat tulis

F. Evaluasi

1. Pertanyaan tidak langsung

   Memninta kepada mahasiswa untuk meberikan komentar tentang Array

2. Pertanyaan Langsung

   - Membuat program sederhana dengan menggunakan konsep Array

G. Referensi

33. Munir, Rinaldi. *Algoritma dan Pemrograman, Jilid I*. Bandung : Informatika. 2007
34. P.Insap Santosa, Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001
35. Abdul Kadir, Pemograman Turbo Pascal, Andi Offset, Yogyakarta, 2006
36. Suryadi, Algoritma dan Pemrograman, Penerbit Gunadarma, Jakarta, 1997

RENCANA KEGIATAN BELAJAR MINGGUAN

(RKBM)

Mata Kuliah : Algoritma Dan Pemrograman

Kode : TIS2223

Semester : II

Waktu : 1 x 3 x 50 Menit

Pertemuan : XV

Minggu (Ke-)	**Topik (Pokok Bahasan)**	**Metode Pembelajaran**	**Estimasi Waktu (Menit)**	**Media**
XV	**Larik (Array)** - Array Satu Dimensi - Cara mengcu elemen array Satu dimensi - Array Dua Dimensi - Cara mengcu elemen array Dua dimensi - Array Multidimensi - Tipe Data Bentukan	Ceramah, Diskusi Kelas	1 x 3 x 50’	Komputer , LCD, papan tulis dan alat tulis

BAB X

Array (Larik) Dan Record

10.1 Pengantar

Dalam bahasa Pascal, 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 (larik), 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 latihan ini hanya akan dibahas dua tipe data kompleks yaitu array dan record.

10.2 A r r a y

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 didahului 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 Subrange. 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
```

Nama Larik Tipe Index Tipe dari Larik

Larik X telah dideklarasikan sebagai Larik tipe Integer dengan Jumlah elemnnya 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
```

Nilai Integer

Index(Subscript)

Nama Larik

Bila nilai elemen ke-3 dari Larik X akan ditampilkan makan 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 :

```
Var
        NilaiHuruf : Array [1…5] Of Char
```

Tipe Index Subrange Integer

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

```
~ Type
        Jangkauan = 1…5;        (tipe subrange)
  Var
        NilaiHuruf : Array[Jangkauan] Of Char;
```

Atau dapat ditulis :

```
~ Type
        Jangkauan = 1…5;
        X         = Array[Jangkauan] Of Char;
  Var
        NilaiHuruf : X;
```

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 :

  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 :

Contoh 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 ?');Readln(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
```

10.3 R e c o r d

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 dalah ecord 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:mahasiswi);
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,mhs1.nim: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;
{====================PROGRAM
UTAMA=======================}
BEGIN
      clrscr;
      data(data1,data2);
      readln;
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        |
=========================================
```

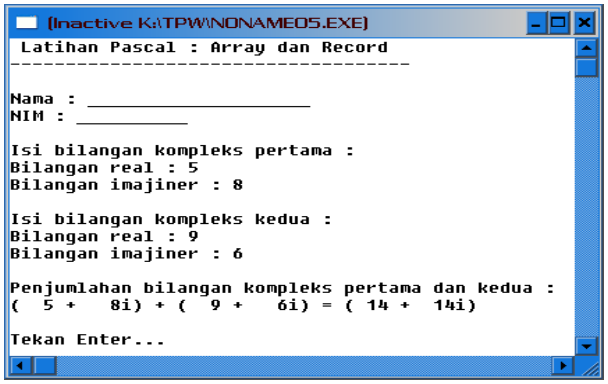
Untuk lebih memahami penggunaan record dalam program, perhatikan contoh berikut ini :

```
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 :');
TulisKompleks(K1);
Write(' + ');
TulisKompleks(K2);
Write(' = ');
TulisKompleks(H);
```

```
Writeln;
Writeln;
Write('Tekan Enter...');
Readln;
end.
```



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 JumlahKompleks.

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
Write('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
```

```
Write(Mat[I,J]:5);
end;
Writeln;
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.
```

JJIAN TENGAH SEMESTER GENAP 2012/2013

INSTITUT TEKNOLOGI PADANG

Mata Kuliah : Algoritma & Pemrograman
Jur/Prog.Studi : T.Informatika /S1
Dosen Penguji : Yuhendra,ST.,MT., Dr.Eng

Tanggal :
Waktu : 75 Menit
Sifat Ujian : Tutup Buku

Soal-01

a. Apa beda program, bahasa pemrograman, pemrogram (Programer) dan Pemrograman

Jawab:

- Program : berisi urutan langkah-langkah penyelesaian masalah
- Bahasa Pemrograman: Program ditulis dengan menggunakan bahasa pemrograman tertentu (Pascal, C, Fortran, dll)
- Pemrogram: Orang yang membuat program
- Pemrograman:Kegiatan merancang dan menulis suatu program

b. Apa itu algoritma dan sebutkan syara-syarat apa saja yang diperlukan dalam merancang algoritma

Jawab:

- Algoritma adalah: Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis.
- Syarat Algoritma yang baik:
  - Tingkat kepercayaannya tinggi (*realibility*).
  - Pemrosesan yang efisien (*cost* rendah).
  - Sifatnya general
  - Bisa dikembangkan (*expandable*).
  - Mudah dimengerti.
  - Portabilitas yang tinggi (*portability*)..
  - *Precise* (tepat, betul, teliti).
  - Efektif.
  - *Output* yang dihasilkan tepat.

Soal -02

a. Buatlah contoh suatu algoritma dalam kehidapan sehari-hari sesuai dengan kaidah yang logis dan tepat.

Jawab:

Algoritma mengirim surat

- Menulis surat
- Surat dimasukkan ke dalam amplop tertutup
- Amplop dikasih alamat penerima dan pengirim
- Amplop ditempeli perangko secukupnya.
- Pergi ke Kantor Pos terdekat untuk mengirimkannya

b. Diberikan kasus untuk menghitung luas dan keliling lingkaran sebagai berikut:

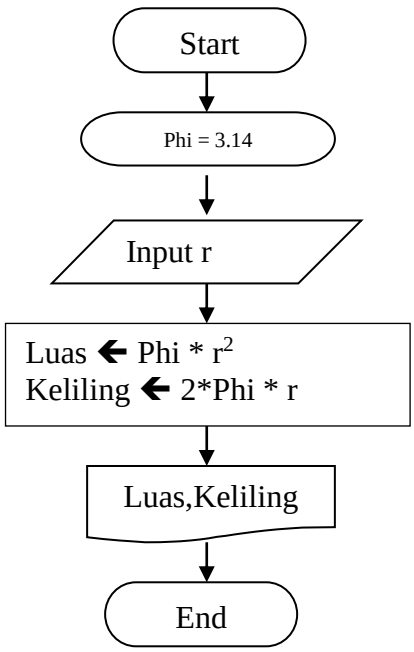
- Konstanta phi = 3.14
- Persamaan luas dan keliling:

  L(Luas) = phi*r*r

  K (Kelililing) = 2*phi*r dimana r=jari-jari

Ditanya: Buat flowchart dan notasi algoritma ke notasi pemrograman untuk menhgitung luas dan keliling

Jawab:



Algoritma Luas _dan _Keliling

{Menghitung luas dan keliling lingkaran untuk ukuran jari-jari tertentu. Algoritma menerima masukan jari-jari lengkaran}

Const {harga tetapan}

Phi : 3.14

Deklarasi

r : integer {peubah data bilangan bulat}

Luas, Keliling: real { Peubah data bilanga pecahan}

Deskripsi:

Read (r) { menginputkan nilai jari-jari}

Luas ← Phi*r^2 { Menghitung nilai luas}

Keliling← 2*Phi* r { Menghitung nilai keliling}

Write (Luas) { mencetak hasil luas}

Write (Keliling) {mencetak hasil keliling}

```
Program Hitung_Luas_Keliling;
Uses Crt;
Const Phi:3.14;
    Var
        r : integer;
        Luas, Keliling : real;
Begin{ Program utama}
Clrscr;
    Write('masukkan nilai jari-jari=?'; Readln (r);
    Luas:= phi*SQR(r);
    Keliling:=2*phi*r;
    Writeln(' nilai luas diperoleh= ', Luas:4');
    Writeln(' nilai keliling diperoleh= ', Keliling:4');
End.
```

Soal -03

Dibaca suhu air (T) (dalam satuan derajat Celsius). Buatkan algoritma untuk menenentukan apakah wujud air tersebut dalam keadaan Padat (suhu ≤ derajat Celsius), cair (suhu antara 0 sampai 100), atau gas (suhu > 100).

Jawab:

Algoritma Wujud_Air
Deklarasi
    T: real
Deskripsi:

```
        Read(T)
        If T ≤ 0, Then  Write (‘Padat’)
                Else
                        If  (T>0) and ( T< 100) Then Write (‘Cair’)
                                Else
                        If T ≥ 100 Then Write (‘Gas atau Uap’)
        Endif
EndIf
```

UJIAN AKHIR SEMESTER GENAP 2012/2013

INSTITUT TEKNOLOGI PADANG

Mata Kuliah : Algoritma & Pemrograman	Tanggal :
Jur/Prog.Studi : T.Informatika /S1	Waktu : 75 Menit
Dosen Penguji : Yuhendra,ST.,MT., Dr.Eng	Sifat Ujian : Buka Buku

Soal

Diketahui :

Indeks nilai mahasiswa ditentukan berdasarkan nilai ujian (N) yang diraihnya. Ketentuan pemberian nilai bobot adalah sebagai berikut:

Nilai ujian ≥ 80 , Bobot nilai = A

70 ≤ nilai ujian < 80 , Bobot nilai = B

55 ≤ nilai ujian < 70 , Bobot nilai = C

50 ≤ nilai ujian < 55 , Bobot nilai = D

Nilai ujian < 40 , Bobot nilai = E

Petunjuk

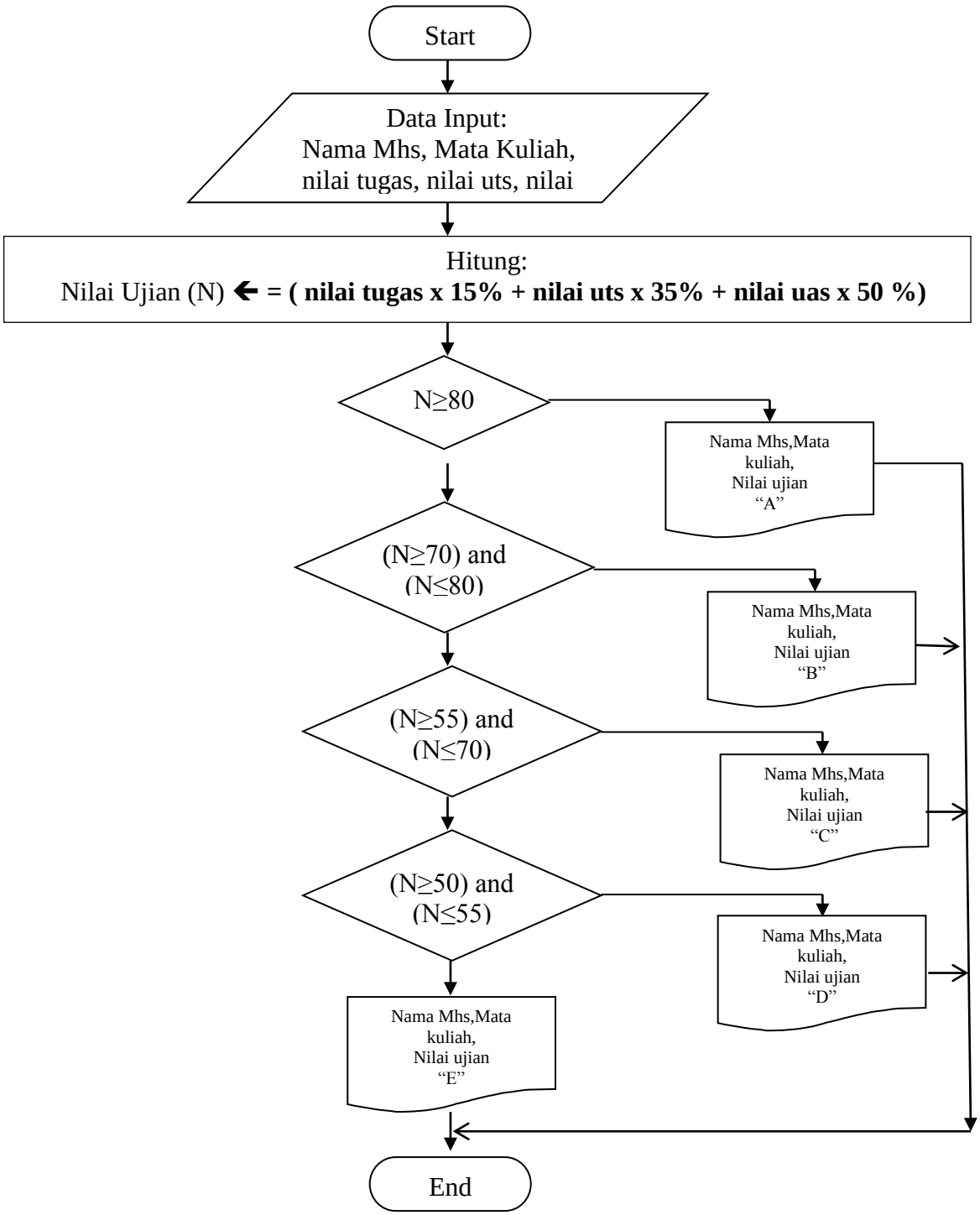
- Data input = Nama Mhs [I], Mata Kuliah [I], nilai tugas [I], nilai uts [I], nilai uas [I]
- Data output= Nama Mhs, maka kuliah, nilai ujian, bobot nilai
- Persamaan , **Nilai Ujian = (nilai tugas x 15% + nilai uts x 35% + nilai uas x 50 %)**
- Gunakan ***Statement IF … Then….Else*** atau ***Case ….of ….Else***
- Gunakan ***Tipe Array (Larik)*** untuk menyatakan indeknya (I)

Ditanya:

a. Buatlah diagram alur (flowchart) untuk menghitung nilai ujian mahasiswa !
b. Buatlah notasi algoritmiknya
c. Ubahlah tranlasi dari algoritmik ke dalam bentuk notasi bahasa pemograman untuk menghitung nilai ujian mahasiswa.

Penyelesaian

a) Flowchart



b). Notasi Algoritmik

```
Algoritma  Nilai_Ujian
Deklarasi
        Nama_Mhs :  Array [1..10] of char
        Mata_Kuliah: Array[1..N] of char
        Nilai_Tugas, UTS, UAS: Array [1..N] of Integer
        NU :Array[1..N] of integer
        N:integer
        Bobot : Char
Deskripsi
        Read (Nama_Mhs, Mata_kuliah, Nilai_Tugas, UTS,UAS)
        NU ← ( nilai tugas [I]*0.15) + (nilai uts [I]* 0.35) + (nilai uas[I] *0.5)
        If   (NU[I] ≥ 80) Then  Bobot  ← 'A' Else
        If (NU[I] ≥70) and (NU[I]≤80) Then  Bobot  ← 'B' Else
        If (NU[I]≥55) and(NU[I]≤70) Then  Bobot  ← 'C' Else
        If (NU[I]≥50) and(NU[I]≤55) Then  Bobot  ← 'D'
        Else  Bobot  ← 'E'
                Endif
            Endif
        Endif
  Endif
        Write (Nama_Mhs, Mata_Kuliah, NU, Bobot)
```

C) Notasi Bahasa Pemograman

```
Program  Nilai_Ujian;
Uses Crt;
 Const N=5;
Var {Deklarasi Variabel}
        Nama_Mhs :  Array [1..10] of char;
        Mata_Kuliah: Array[1..N] of char;
        Nilai_Tugas, UTS, UAS: Array [1..N] of Integer;
        NU :Array[1..N] of integer;
        N:integer;
        Bobot : Char;
```

```
Begin {Program Utama}
Clrscr;
{Program input data}
        Write (masukkan Nama mahasiswa= ?'); Readln (Nama_Mhs);
        Write (masukkan Nama Matakuliah= ?'); Readln (Nama_MataKuliah[I]);
        Write (masukkan Nilai Tugas= ?'); Readln (Nilai_Tugas[I]);
        Write (masukkan Nilai UTS= ?'); Readln (UTS[I]);
        Write (masukkan Nilai UAS= ?'); Readln (UAS[I]);
{Proses Perhitungan}
        NU :=( nilai tugas [I]*0.15) + (nilai uts [I]* 0.35) + (nilai uas[I] *0.5);
{Proses Penyeleksian}
        If   (NU[I] ≥ 80) Then  Bobot:= 'A'
                Else
        If (NU[I] ≥70) and (NU[I]≤80) Then  Bobot:= 'B'
                Else
        If (NU[I]≥55) and(NU[I]≤70) Then  Bobot := 'C'
                 Else
        If (NU[I]≥50) and(NU[I]≤55) Then  Bobot:= 'D'
                Else
         Bobot := 'E'
                End;
            End;
        End;
  End;
{Proses output}
        Writeln ('Nama Mahasiswa ='  , Nama_Mhs[I] : 10);
        Writeln ('Nama Mata kuliah  = ' , Mata_kuliah[I]);
        Writeln ('Nama ujian akhir  = ' , NU[I] );
        Writeln ('Bobot nilai  = ' , Bobot);
End.
```