ALGORITMA & PEMROGRAMAN MENGGUNAKAN MATLAB

Dr. Hindarto, MT
Ade Eviyanti, S.Kom., M.Kom

ISBN 978-623-6833-71-1 (PDF)

BUKU AJAR
UNIVERSITAS MUHAMMADIYAH SIDOARJO

UMSIDA
DARI SINI PENCERAHAN BERSEMI

ALGORITMA & PEMROGRAMAN MENGGUNAKAN MATLAB

Dr. Hindarto, MT
Ade Eviyanti, S.Kom., M.Kom

BUKU AJAR
UNIVERSITAS MUHAMMADIYAH SIDOARJO

BUKU AJAR
ALGORITMA & PEMROGRAMAN MENGGUNAKAN MATLAB

Oleh
Dr. Hindarto, S.Kom, MT.
Ade Eviyanti, S.Kom, M.Kom.

UNIVERSITAS MUHAMMADIYAH SIDOARJO
2020

BUKU AJAR
ALGORITMA & PEMROGRAMAN MENGGUNAKAN MATLAB

Penulis:
Dr. Hindarto, S.Kom, MT.
Ade Eviyanti, S.Kom, M.Kom.

ISBN :
978-623-6833-71-1

Editor:
Muhammad Suryawinata, S.Pd, M.Kom.

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

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

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

Cetakan Pertama, September 2020

KATA PENGANTAR

Alhamdulillah kami panjatkan kehadirat Allah SWT, serta sholawat dan salam tercurahkan kepada junjungan nabi kita nabi Muhammad SAW. Buku Algoritma & Pemrograman menggunakan Matlab ini dimaksudkan sebagai pegangan kegiatan belajar mengajar di Fakultas Teknik Umsida, khususnya untuk program studi Teknik Industri dan Teknik Mesin. Pada buku ini diuraikan tentang berbagai jenis tipe data, variabel, dan konstanta serta pemakaiannya dalam algoritma. Mengetahui dan memahami operator aritmetika dan logika dan penggunaannya dalam algoritma. Menggunakan bahasa pemrograman Matlab untuk pemecahan masalahnya.

Penulis menyadari bahwa pada penyusunan buku Algoritma & Pemrograman menggunakan Matlab ini jauh dari sempurna, baik dari segi penyusunan, bahasan, ataupun penulisannya. Oleh karena itu penulis mengharapkan kritik dan saran yang sifatnya membangun, guna menjadi acuan dalam bekal pengalaman untuk lebih baik dimasa yang akan datang. Semoga buku ini dapat dipergunakan sebagai salah satu acuan, petunjuk maupun pedoman bagi pembaca.

Sidoarjo, 30 Juli 2020

Penulis

DAFTAR ISI

Hal

CAPAIAN PEMBELAJARAN MATA KULIAH ALGORITMA & PEMROGRAMAN :

1. Mahasiswa mampu mengenal lingkungan, bahasa pemrograman Matlab. Dapat menggunakan bahasa pemrograman Matlab untuk pemecahan masalahnya.
2. Mahasiswa Mengerti berbagai jenis tipe data, variabel, dan konstanta serta pemakaiannya dalam algoritma.
3. Mahasiswa mengetahui dan memahami operator aritmetika dan logika dan penggunaannya dalam algoritma.
4. Mahasiswa mengerti dan memahami fungsi input dan output dan menerapkan dalam pembuatan algoritma
5. Mahasiswa dapat menjelaskan konsep struktur dasar runtunan dan menerapkan dalam pembuatan program
6. Mahasiswa dapat Menjelaskan konsep struktur dasar seleksi kondisi.
7. Mahasiswa Dapat Menjelaskan *statement* yang digunakan dalam penyeleksian kondisi dan menerapkannya dalam pembuatan algoritma.
8. Mahasiswa dapat menjelaskan pengertian pengulangan proses program.
9. Mahasiswa Mengerti dan memahami konsep kounter dan akumulator serta penerapannya dalam pembuatan program

10. Mahasiswa dapat menjelaskan statement yang digunakan dalam pengulangan proses program dan menerapkannya dalam pembuatan program.
11. Mahasiswa dapat memecahkan permasalahan dengan algoritma dan program
12. Mahasiswa dapat menjelaskan konsep dasar dan definisi prosedur
13. Mahasiswa mengerti dan memahami cara deklarasi dan pemanggilan prosedur
14. Mahasiswa dapat menjelaskan ruang lingkup variabel dan cara pengiriman parameter
15. Mahasiswa dapat membuat algoritma yang memuat prosedur.
16. Mahasiswa dapat menjelaskan konsep dasar dan definisi fungsi.
17. Mahasiswa mengerti dan memahami cara deklarasi dan pemanggilan fungsi.
18. Mahasiswa dapat menjelaskan pengertian dan deklarasi array.
19. Mahasiswa dapat membuat algoritma yang memuat operasi matriks.
20. Mahasiswa dapat menjelaskan pengertian searching dan berbagai metode yang digunakan.
21. Mahasiswa dapat membuat algoritma yang memuat searching.

BAB I
LOGIKA DAN ALGORITMA

A. Definisi Logika dan Algoritma

Logika, seperti yang digunakan oleh komputer, menggunakan fungsi Logika seperti DAN, ATAU, TIDAK, di mana jawabannya adalah Benar atau Salah. Jika Benar, hasilnya adalah "1" dan jika salah hasilnya adalah "0." String satu dan nol melewati serangkaian gerbang AND di mana seri unik dikaitkan dengan karakter atau instruksi.

Algoritma digunakan dengan cara yang sama untuk membuat keputusan Benar atau Salah. Algoritma IF-THEN akan melakukan satu hal jika "IF" benar, tetapi lanjutkan dengan "THEN", jika tidak benar kemudian bisa menjadi IF berikutnya lagi. Banyak keputusan akan didasarkan pada pemeriksaan matematika: Lebih besar dari, Kurang dari, jika demikian ini dan itu, jika tidak melakukan sesuatu yang lain (Munir, R. 1999).

Apabila kita melihat kode yang berfungsi, tetapi seharusnya tidak berfungsi. Kode mengambil keuntungan dari kesalahan di bagian lain dari program. Kode berfungsi hingga kesalahan lainnya diperbaiki. Algoritma itu tergantung pada definisi dari masing – masing individu. Biasanya menganggap suatu algoritma sebagai terputus dari implementasi tertentu. Algoritma setidaknya dapat secara teori bekerja dalam bahasa yang berbeda, pada berbagai jenis komputer. Jadi algoritma non-logis adalah kontradiksi. Algoritma dapat memiliki logika yang tidak jelas, tetapi akan selalu ada logika terkait untuk memenuhi hasil yang diinginkan. Bahkan, membuat deskripsi yang cukup baik dari suatu algoritma dan logika diperlukan untuk memenuhi hasil tertentu.

Definisi informal dapat berupa "seperangkat aturan yang secara tepat mendefinisikan urutan operasi", yang akan

mencakup semua program komputer, termasuk program yang tidak melakukan perhitungan numerik, dan (misalnya) prosedur birokrasi apa pun yang ditentukan. Secara umum, sebuah program hanya sebuah algoritma jika berhenti pada akhirnya. Contoh prototipikal dari suatu algoritma adalah algoritma Euclidean, yang digunakan untuk menentukan pembagi umum maksimum dua bilangan bulat; contoh (ada yang lain) dijelaskan oleh diagram alur di atas dan sebagai contoh di bagian selanjutnya.

Logika (dari bahasa Yunani Kuno) adalah studi sistematis tentang bentuk-bentuk inferensi, hubungan yang mengarah pada penerimaan satu proposisi, kesimpulan, berdasarkan seperangkat proposisi lain, tempat. Secara lebih luas, logika adalah analisis dan penilaian argumen. Tidak ada kesepakatan universal mengenai definisi yang tepat dan batasan-batasan logika, dan inilah mengapa masalah ini masih tetap menjadi salah satu subjek utama penelitian dan perdebatan di bidang filsafat logika. Namun, secara tradisional termasuk klasifikasi argumen, eksposisi sistematis dari bentuk logis, validitas dan kesehatan penalaran deduktif, kekuatan penalaran induktif, studi bukti dan kesimpulan formal (termasuk paradoks dan fallacy), dan studi sintaksis dan semantik.

Algoritma menyerupai resep. Resep memberi tahu Anda cara menyelesaikan tugas dengan melakukan sejumlah langkah. Misalnya, untuk membuat kue, langkah-langkahnya adalah: memanaskan lebih dulu oven; mencampur tepung, gula, dan telur secara menyeluruh; tuang ke loyang; Dan seterusnya. Namun, "algoritma" adalah istilah teknis dengan makna yang lebih spesifik daripada "resep", dan menyebut sesuatu sebagai algoritma berarti bahwa properti berikut semuanya benar.

Algoritma adalah deskripsi yang jelas yang menjelaskan apa yang harus diimplementasikan. Dalam resep, langkah seperti "Panggang sampai selesai" tidak jelas karena tidak

menjelaskan apa yang berarti "selesai". Deskripsi yang lebih eksplisit seperti "Panggang sampai keju mulai menggelembung" lebih baik. Dalam algoritma komputasi, langkah seperti "Pilih nomor besar" tidak jelas: apa yang besar ? 1 juta, 1 miliar, atau 100 ? Apakah nomor harus berbeda setiap kali, atau dapatkah nomor yang sama digunakan pada setiap kali.

Algoritma mengharapkan satu set input yang ditentukan. Misalnya, mungkin memerlukan dua angka di mana kedua angka lebih besar dari nol. Atau mungkin memerlukan kata, atau daftar angka nol atau lebih.

Algoritma menghasilkan serangkaian output yang ditentukan. Mungkin menampilkan yang lebih besar dari dua angka, versi huruf besar semua kata, atau versi diurutkan dari daftar angka.

Algoritma dijamin untuk mengakhiri dan menghasilkan hasil, selalu berhenti setelah waktu yang terbatas. Jika suatu algoritma berpotensi berjalan selamanya, itu tidak akan sangat berguna karena Anda mungkin tidak pernah mendapatkan jawaban.

Sebagian besar algoritma dijamin untuk menghasilkan hasil yang benar. Jarang berguna jika suatu algoritma mengembalikan jumlah terbesar 99% dari waktu, tetapi 1% dari waktu algoritma gagal dan mengembalikan jumlah terkecil sebagai gantinya.

Jika suatu algoritma memaksakan persyaratan pada inputnya (disebut prasyarat), persyaratan itu harus dipenuhi. Misalnya, prasyarat mungkin bahwa suatu algoritma hanya akan menerima angka positif sebagai input. Jika prasyarat tidak terpenuhi, maka algoritme dibiarkan gagal dengan menghasilkan jawaban yang salah atau tidak pernah berakhir (Kadir, A dan Heriyanto, 2005).

Membuat penggunaan logika dan algoritma untuk menghitung luas lingkaran :

1. Tentukan nilai dari jari-jari lingkaran.
2. Tentukan nilai dari π (phi).
3. Hitung luas lingkaran dengan cara nilai jari-jari dikali jari-jari dikali nilai phi.
4. Maka luas dari lingkaran yang dicari akan ditemukan.
5. Selesai.

B. Cara Penulisan Algoritma

Ada beberapa karakteristik yang harus diikuti oleh setiap algoritma. Ada Enam karakteristik berbeda yang berhubungan dengan berbagai aspek algoritma. Karakteristik tersebut adalah sebagai berikut (Kadir, A dan Heriyanto, 2005) :

1. Masukan ditentukan
   Input adalah data yang akan ditransformasikan selama perhitungan untuk menghasilkan output. Algoritma harus memiliki 0 atau lebih input yang terdefinisi dengan baik.
2. Output ditentukan
   Output adalah data yang dihasilkan dari perhitungan (hasil yang Anda inginkan). Algoritme harus memiliki 1 atau lebih output yang terdefinisi dengan baik, dan harus sesuai dengan output yang diinginkan. Ketepatan output juga mengharuskan Anda mengetahui jenis data apa, berapa banyak dan apa bentuk output seharusnya (atau bahkan jika akan ada output pada semua).
3. Kepastian
   Algoritma harus menentukan setiap langkah dan urutan langkah-langkah harus diambil dalam proses. Ketegasan berarti menentukan urutan operasi untuk mengubah input menjadi output. Algoritma harus jelas dan tidak ambigu. Rincian setiap langkah juga harus dijabarkan

(termasuk cara menangani kesalahan). Ini harus berisi segala sesuatu yang kuantitatif dan bukan kualitatif.

4. Efektivitas
   Agar suatu algoritma menjadi efektif, itu berarti bahwa semua langkah yang diperlukan untuk mencapai output harus layak dengan sumber daya yang tersedia. Itu tidak boleh mengandung langkah-langkah yang tidak perlu dan berlebihan yang bisa membuat algoritma tidak efektif.
5. Keterbatasan
   Algoritme harus berhenti, akhirnya. Menghentikan mungkin berarti Anda mendapatkan output yang diharapkan ATAU Anda mendapatkan respons yang tidak ada solusi yang mungkin. Algoritma harus berakhir setelah jumlah langkah yang terbatas. Algoritma tidak boleh tak terbatas dan selalu berakhir setelah jumlah langkah yang pasti.
6. Independen
   Algoritme harus memiliki petunjuk langkah demi langkah, yang harus independen dari kode pemrograman apa pun. Ini harus sedemikian rupa sehingga dapat dijalankan pada salah satu bahasa pemrograman.

Sedang sifat algoritma adalah:

1. Simbol atau sintaks dari suatu bahasa pemrograman tertentu tidak digunakan.
2. Bebas dalam menentukan bahasa pemrograman.
3. Seluruh bahasa manapun Notasi-notasinya dapat digunakan.
4. Untuk merepresentasikan suatu urutan kejadian secara logis dan diterapkan di semua kejadian sehari-hari dapat menggunakan algoritma yang dipakai.

C. Runtunan (sequence)

Sequencing adalah urutan khusus di mana instruksi dilakukan dalam suatu algoritma.

Misalnya, algoritma yang sangat sederhana untuk menyikat gigi mungkin terdiri dari langkah-langkah ini:

1. Letakkan pasta gigi di atas sikat gigi
2. Gunakan sikat gigi untuk membersihkan gigi
3. Bilas sikat gigi

Setiap langkah adalah instruksi yang harus dilakukan. Sequencing adalah urutan pelaksanaan langkah-langkah.

Mengapa sequencing itu penting?

Sangat penting bahwa langkah-langkah dalam suatu algoritma dilakukan dalam urutan yang benar - jika tidak, algoritma tersebut tidak akan berfungsi dengan benar. Misalkan langkah-langkah untuk algoritma pembersihan gigi berada dalam urutan ini:

1. gunakan sikat gigi untuk membersihkan gigi
2. letakkan pasta gigi di atas sikat gig
3. bilas sikat gigi

Sikat gigi masih akan digunakan untuk membersihkan gigi dan pasta gigi masih akan diletakkan di atas sikat. Tetapi karena langkah 1 dan 2 berada dalam urutan yang salah, gigi tidak akan dibersihkan dengan pasta gigi, dan pasta gigi akan terbuang sia-sia.

Manusia akan menyadari bahwa mereka lupa menambahkan pasta gigi pada awal proses, tetapi komputer tidak akan tahu bahwa ada sesuatu yang salah.

D. Pemilihan (selection)

Algoritma untuk menemukan angka terkecil k (atau terbesar) dalam daftar atau array. Angka itu disebut statistik urutan k. Ini mencakup berbagai kasus untuk menemukan elemen minimum, maksimum, dan median dalam daftar atau array. Untuk menemukan elemen minimum (atau maksimum)

dengan mengulangi daftar, kami melacak elemen minimum (atau maksimum) saat ini yang terjadi sejauh ini dan terkait dengan jenis seleksi.

Di bawah ini adalah berbagai cara untuk memilih elemen k terkecil (atau terbesar) dalam daftar tidak berurutan :

1. Seleksi dengan Menyortir
   Menyortir daftar atau larik kemudian memilih elemen yang diperlukan dapat mempermudah pemilihan. Metode ini tidak efisien untuk memilih elemen tunggal tetapi efisien ketika banyak pilihan harus dibuat dari array yang hanya perlu menyortir array. Untuk pemilihan dalam daftar tertaut adalah O (n) bahkan jika daftar tertaut diurutkan karena kurangnya akses acak.
2. Seleksi Berbasis Partisi
   Untuk pemilihan berbasis partisi, Algoritma pilih Cepat digunakan. Ini adalah varian dari algoritma quicksort. Dalam keduanya, kami memilih elemen pivot dan menggunakan langkah partisi dari algoritma quicksort mengatur semua elemen lebih kecil dari pivot di sebelah kirinya dan elemen-elemen lebih besar dari pada di sebelah kanannya. Tetapi sementara Quicksort berulang di kedua sisi partisi, Quickselect hanya berulang di satu sisi, sisi di mana elemen k yang diinginkan hadir. Algoritma berbasis partisi dilakukan di tempat, yang menghasilkan penyortiran sebagian data. Mereka dapat dilakukan di tempat dengan tidak mengubah data asli dengan mengorbankan ruang bantu O (n).
3. Median dipilih sebagai poros
   Algoritma seleksi-median dapat digunakan untuk melakukan algoritma seleksi atau algoritma pengurutan, dengan memilih median array sebagai elemen pivot dalam algoritma Quickselect atau Quicksort. Dalam praktiknya, overhead perhitungan pivot sangat penting, sehingga

algoritma ini umumnya tidak digunakan, tetapi teknik ini menarik secara teoritis dalam mengaitkan algoritma seleksi dan pengurutan. Median array adalah pivot terbaik untuk menyortir array karena secara merata membagi data menjadi dua bagian, dan dengan demikian menjamin pengurutan yang optimal, dengan asumsi algoritma pemilihan optimal.

E. Pengulangan (repetition)

Pengulangan memungkinkan sebagian dari suatu algoritma atau program komputer dieksekusi beberapa kali depednet pada beberapa kondisi yang dipenuhi. Kejadian pengulangan biasanya dikenal sebagai loop. Fitur penting dari pengulangan adalah bahwa setiap loop memiliki kondisi terminasi untuk menghentikan pengulangan, atau hasil yang jelas adalah bahwa loop tidak pernah menyelesaikan eksekusi. Ini dikenal sebagai loop tak terbatas dan jelas tidak diinginkan. Kondisi terminasi dapat diperiksa atau diuji pada awal atau akhir loop, dan masing-masing dikenal sebagai pre-test atau post-test. Berikut ini adalah deskripsi masing-masing jenis loop ini :

1. Pengulangan pra-tes loop

Loop pra-uji dinamakan demikian karena kondisi harus dipenuhi pada awal loop atau badan loop tidak dieksekusi. Konstruk ini sering disebut loop dijaga. Tubuh loop dijalankan berulang kali saat kondisi terminasi benar.

Contoh Pseudocode :

```
WHILE condition is true
  processes
ENDWHILE
```

2. Pengulangan post-test loop

Loop setelah diuji mengeksekusi tubuh loop sebelum menguji kondisi terminasi. Konstruk ini sering disebut sebagai loop

tidak dijaga. Tubuh loop berulang kali dieksekusi sampai kondisi terminasi benar. Perbedaan penting antara pre-test dan post-test loop adalah bahwa pernyataan dari post-test loop dieksekusi setidaknya satu kali, bahkan jika kondisinya semula benar, sedangkan badan loop pra-tes mungkin tidak pernah dieksekusi jika kondisi terminasi ini benar adanya. Pandangan dekat pada representasi dari dua tipe loop membuat titik ini jelas.

Contoh Pseudocode :

```
REPEAT
  process
UNTIL  condition is true
```

3. FOR NEXT or counted loop

Loop dihitung atau UNTUK loop BERIKUTNYA dapat dianggap sebagai kasus pengulangan khusus dan, tergantung pada bahasa di mana mereka diimplementasikan, diimplementasikan sebagai pengulangan pra-tes atau pasca-tes. Untuk mendemonstrasikan loop FOR NEXT sebagai flowchart, ia harus memiliki tujuan, contoh kita akan mencetak tabel 12 kali.

Contoh Pseudocode :

```
FOR i = 1 to 12 STEP 1
  Display "12 x " i " = " (12 * i)
NEXT i
```

F. Perbedaan Algoritma, pseudocode dan Program

Pendekatan logis sistematis yang merupakan prosedur selangkah demi selangkah yang didefinisikan dengan baik yang memungkinkan komputer untuk menyelesaikan masalah merupakan Algoritma.

Versi yang lebih sederhana dari pemrograman sederhana yang menggunakan frase pendek untuk menulis kode untuk suatu

program sebelum diimplementasikan dalam bahasa pemrograman tertentu adalah Pseudocode.

Program adalah kode persis yang ditulis untuk masalah mengikuti semua aturan bahasa pemrograman.

Algoritma digunakan untuk memberikan solusi untuk masalah tertentu dalam bentuk langkah-langkah yang didefinisikan dengan baik. Setiap kali Anda menggunakan komputer untuk memecahkan masalah tertentu, langkah-langkah yang mengarah ke solusi harus dikomunikasikan dengan benar ke komputer. Saat mengeksekusi algoritma pada komputer, beberapa operasi seperti penambahan dan pengurangan digabungkan untuk melakukan operasi matematika yang lebih kompleks. Algoritma dapat diekspresikan menggunakan bahasa alami, diagram alur, dll. Contoh untuk pemahaman yang lebih baik, sebagai seorang programmer untuk mengetahui program Pencarian Linear.

Algoritma pencarian linear:

1. Mulai dari elemen paling kiri dari bilangan dan satu per satu bandingkan x dengan setiap elemen bilangan
2. Jika x cocok dengan suatu elemen, kembalikan indeks.
3. Jika x tidak cocok dengan elemen apa pun, kembalikan -1.

Di sini, kita dapat melihat bagaimana langkah-langkah program pencarian linear dijelaskan dalam bahasa yang sederhana.

Pseudocode adalah metode yang dapat digunakan untuk mewakili suatu algoritma untuk suatu program. Ia tidak memiliki sintaksis spesifik seperti bahasa pemrograman mana pun dan karenanya tidak dapat dijalankan di komputer. Ada beberapa format yang digunakan untuk menulis kode pseudo dan kebanyakan dari mereka mengambil struktur dari bahasa seperti C, Lisp, FORTRAN, Matlab, dll.

Banyak algoritma waktu disajikan menggunakan pseudocode karena dapat dibaca dan dipahami oleh programmer yang terbiasa dengan bahasa pemrograman yang berbeda. Pseudocode memungkinkan Anda untuk memasukkan beberapa struktur kontrol seperti While, If-then-else, Ulangi-sampai, untuk dan huruf besar-kecil, yang hadir dalam banyak bahasa tingkat tinggi.
Suatu program adalah serangkaian instruksi untuk diikuti oleh komputer. Mesin tidak dapat membaca program secara langsung, karena hanya memahami kode mesin. Tetapi Anda dapat menulis hal-hal dalam bahasa komputer, dan kemudian kompiler atau juru bahasa dapat membuatnya dimengerti oleh komputer.

Algoritma vs Psuedocode vs Algoritma

1. Algoritma didefinisikan sebagai urutan langkah-langkah yang terdefinisi dengan baik yang memberikan solusi untuk masalah yang diberikan, sedangkan kodesemu adalah salah satu metode yang dapat digunakan untuk mewakili suatu algoritma.
2. Sementara algoritma umumnya ditulis dalam bahasa alami atau bahasa Inggris biasa, pseudocode ditulis dalam format yang mirip dengan struktur bahasa pemrograman tingkat tinggi. Program di sisi lain memungkinkan kita untuk menulis kode dalam bahasa pemrograman tertentu.

Jadi, seperti yang digambarkan di atas, dapat dengan jelas melihat bagaimana algoritma digunakan untuk menghasilkan pseudocode yang dikembangkan lebih lanjut dengan mengikuti sintaks tertentu dari bahasa pemrograman untuk membuat kode program.

G. Membuat Program dan Bahasa Pemrograman

Membuat suatu program mempunyai perbedaan denagn kalau kita belajar bahasa pemrograman. Kalau membuat program, kita belajar cara atau strategi untuk menyelesaikan suatu masalah. Penyelesaian tersebut dituangkan dengan algoritma dan dapat dipahami sehingga dapat dituangkan dalam pemrograman. Kalau Pemrograman merupakan seni untuk membuat algoritma dalam suatu bahasa pemrograman dan dapat dijalankan di dalam sebuah komputer. Beberapa hal yang dapat dilakukan untuk memulai belajar pemrograman:

1. Pikirkan tentang apa yang ingin Anda lakukan dengan pengetahuan pemrograman Anda.

Membantu Anda menentukan apa yang harus dipelajari dan berapa banyak yang perlu Anda pelajari. Apakah Anda tertarik dengan desain web? Apakah Anda ingin membuat video game? Apakah Anda ingin mengembangkan aplikasi ponsel cerdas? Apakah Anda ingin berkarir di industri teknologi? Apakah Anda menikmati pemecahan masalah? Apakah Anda lebih tertarik pada pemrograman front-end atau pemrograman back-end?

Pemrogram front-end bekerja pada hal-hal seperti antarmuka pengguna grafis (GUI) dan hal-hal yang berinteraksi dengan pengguna. Bahasa populer untuk pemrogram front-end termasuk HTML, CSS, dan Javascript.

Pemrogram back-end bekerja pada hal-hal seperti database, scripting, dan arsitektur program, dan hal-hal yang terjadi di belakang layar. Bahasa pemrograman populer untuk pengguna back-end termasuk Ruby, Python, PHP, dan alat-alat seperti MySQL dan Oracle.

2. Pikirkan tentang platform apa yang Anda minati

Apakah Anda ingin mengembangkan perangkat lunak untuk komputer? Apakah Anda lebih tertarik dengan aplikasi smartphone dan tablet. Jika demikian, sistem operasi apa yang paling menarik bagi Anda? Mengembangkan perangkat lunak

untuk macOS mungkin mengharuskan Anda mempelajari berbagai bahasa yang Anda mungkin tidak perlu tahu mengembangkan aplikasi untuk Windows. Demikian juga, mengembangkan aplikasi iPhone dan iPad mungkin memerlukan keterampilan yang berbeda dari mengembangkan aplikasi Android.

3. Memahami konsep pemrograman yang berbeda

Meskipun ada banyak bahasa pemrograman yang berbeda, ada beberapa konsep dasar yang mereka miliki bersama. Beberapa konsep pemrograman dasar adalah sebagai berikut:

Variabel: Variabel adalah potongan-potongan informasi yang disimpan sehingga mereka dapat dipanggil kembali nanti. Variabel biasanya diberi nama simbolik. Salah satu contoh variabel adalah jika suatu program meminta pengguna untuk memasukkan nama mereka. Nama yang mereka masukkan dapat disimpan di bawah simbol objek yang disebut "nama". Programmer kemudian dapat menggunakan simbol "nama" untuk memanggil kembali nama input pengguna dan merujuk pengguna dengan nama mereka. Variabel atau objek yang terdiri dari karakter disebut "String".

Struktur Kontrol: Struktur Kontrol memberi tahu program bagian mana dari program yang perlu dijalankan dan dalam urutan apa. Salah satu jenis umum dari struktur kontrol sering disebut sebagai pernyataan If / Then / Else. Ini memberi tahu program bahwa jika suatu kondisi benar, maka jalankan bagian bagian berikutnya dari program. Untuk yang lainnya, kembali ke bagian lain. Misalnya, jika suatu program meminta pengguna untuk membuat kata sandi, kata sandi tersebut disimpan sebagai string. Layar kata sandi meminta pengguna untuk memasukkan kata sandi mereka. Pernyataan IF / Then / Else digunakan untuk memberi tahu program bahwa jika kata sandi yang dimasukkan sama dengan kata sandi yang

disimpan, maka jalankan sisa program. Untuk yang lainnya, tampilkan "Kata sandi Anda salah".
Struktur Data: Struktur data hanyalah cara untuk menyimpan dan mengatur data sehingga dapat digunakan secara efisien. Salah satu contoh struktur data adalah kontak di ponsel Anda. Alih-alih menyimpan kontak Anda masing-masing sebagai variabel terpisah, pemrograman Anda dapat membuat satu variabel yang disebut "Daftar" yang menyimpan semua kontak Anda.
Sintaks: Sintaks adalah cara kode yang benar dimasukkan dalam bahasa tertentu. Setiap bahasa pemrograman memiliki sintaks yang berbeda. Sintaksnya bisa berupa cara menyimpan variabel, kapan harus menggunakan simbol yang berbeda (yaitu tanda kurung (), atau tanda kurung []), penggunaan indentasi yang tepat, dan banyak lagi. Jika sintaks tidak dimasukkan dengan benar, program tidak akan dapat membaca kode dan Anda kemungkinan besar akan mendapatkan pesan kesalahan (Pranata, A. 2005).
Alat: Alat adalah hal-hal yang membantu membuat pemrograman lebih mudah. Ini bisa berupa fitur perangkat lunak yang memeriksa kode Anda dan memastikan itu benar. Ini juga bisa menjadi fitur program pra-dibuat yang dapat Anda terapkan ke dalam program Anda sendiri sehingga Anda tidak harus membangunnya sendiri.

Soal Latihan :

1. Apa Pengertian dari Logika dan Pengertian dari Algoritma.
2. Apa saja sifat dari algoritma.
3. Bagaiman algoritma yang sangat sederhana untuk Memasak Nasi Goreng.
4. Apa Perbedaan Algoritma dan Program.
5. Buatlah Pseudocode untuk mengurutkan bilanag 1 sampai 100.

BAB II
NOTASI PENULISAN ALGORITMA

Algoritme adalah rencana, proses langkah demi langkah yang logis untuk menyelesaikan masalah. Algoritma biasanya ditulis

sebagai diagram alur atau pseudocode. Kunci dari setiap tugas pemecahan masalah adalah untuk memandu proses pemikiran seseorang. Hal yang paling berguna untuk dilakukan adalah terus bertanya 'Bagaimana jika kita melakukannya dengan cara ini' Menjelajahi berbagai cara penyelesaian masalah dapat membantu menemukan cara terbaik untuk menyelesaikannya (Levitin, Anany, 2010). Untuk menuliskan lgoritma ada tiga cara yang umum digunakan :

1. Kalimat deskriptif
2. Pseudocode
3. Flowchart

A. Kalimat deskriptif

Notasi alami adalah Notasi algoritma dengan menggunakan kalimat deskriptif. Notasi deskriptif tersusun atas tiga bagian utama, yaitu :

1. Bagian Judul yaitu bagian yang terdiri atas nama algoritma, penjelasan atau spesifikasi algoritma tersebut.
2. Bagian Deklarasi yaitu bagian untuk mendefinisikan semua nama yang digunakan pada algoritma dapat berupa variabel, konstanta, tipe ataupun fungsi
3. Bagian Deskripsi yaitu bagian inti pada struktur algoritma yang berisi uraian langkah-langkah penyelesaian masalah.

Contoh penulisan algoritma dengan notasi deskriptif.

Algoritma untuk menghitung Luas_Lingkaran. Algoritma menerima masukan jari-jari lingkaran dan Menghitung luas lingkaran untuk ukuran jari-jari tertentu, serta mencetak luasnya ke piranti keluaran.

Deklarasi :

Jari_jari = real {tipe data bilangan pecahan}
Luas = real {tipe data bilangan pecahan}
PHI = 3.14

Deskripsi
1. Baca jari-jari lingkaran
2. Hitung luas lingkaran = PHI*jari-jari * jari_jari
3. Tampilkan luas lingkaran ke layar
4. Selesai

B. Pseudocode

Sebagian besar program dikembangkan menggunakan bahasa pemrograman. Bahasa-bahasa ini memiliki sintaks tertentu yang harus digunakan sehingga program akan berjalan dengan baik. Pseudocode bukan bahasa pemrograman, ini adalah cara sederhana untuk menggambarkan sekumpulan instruksi yang tidak harus menggunakan sintaksis tertentu.

Notasi pseudocode secara umum

Tidak ada set ketat notasi standar untuk kodesemu, tetapi beberapa yang paling dikenal adalah:

1. INPUT - menunjukkan pengguna akan memasukkan sesuatu.
2. OUTPUT - menunjukkan bahwa output akan muncul di layar.
3. WHILE - loop (iterasi yang memiliki kondisi di awal)
4. FOR - loop penghitungan (iterasi)
5. **REPEAT – UNTIL** - loop (iterasi) yang memiliki kondisi di akhir
6. **IF – THEN – ELSE** - keputusan (pilihan) di mana pilihan dibuat
7. Instruksi apa pun yang terjadi di dalam pilihan atau iterasi biasanya diindentasi.

C. Flowchart

Flowchart adalah diagram yang mewakili sekumpulan instruksi. Flowchart biasanya menggunakan simbol standar untuk mewakili berbagai jenis instruksi. Simbol-simbol ini

digunakan untuk membangun diagram alur dan menunjukkan solusi langkah-demi-langkah untuk masalah tersebut (Wirth, Nicholas, 2004).

Simbol Flowchart secara umum

1. Terminal Box - Start / End

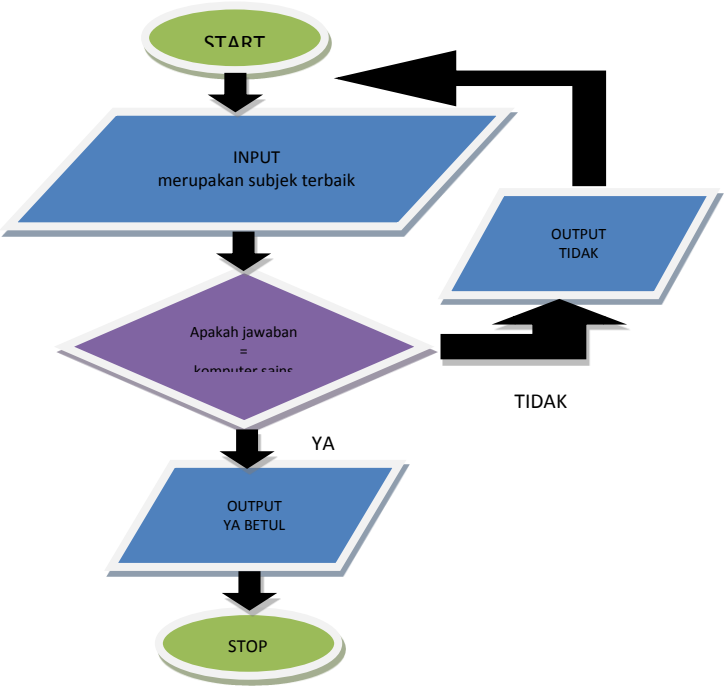
2. Input / Output

3. Proses / Instruksi

4. Decision

5. Conektor / Arah Panah

Flowchart dapat digunakan untuk merencanakan program. Merencanakan program yang bertanya kepada orang-orang apa subjek terbaik yang mereka ambil, akan terlihat seperti ini sebagai diagram alur:



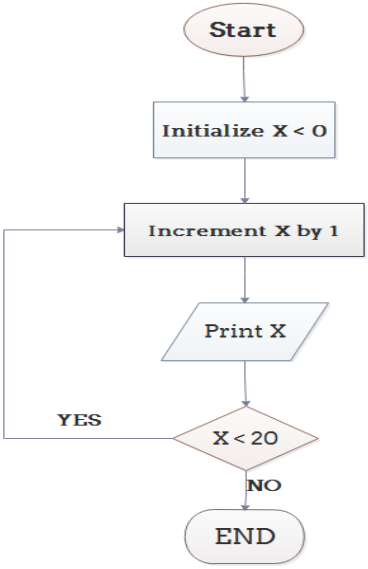
Gambar 2.1. Contoh Penggunaan Flowchart

Jadi diagram alur sering digunakan sebagai alat perencanaan program untuk secara visual mengatur proses langkah-demi-langkah dari suatu program. Berikut ini beberapa contohnya:

Contoh 1 :

Algoritma :
Langkah 1: Inisialisasi X sebagai 0,
Langkah 2: Tambah X dengan 1,
Langkah 3: Cetak X,
Langkah 4: Jika X kurang dari 20 maka kembali ke langkah 2.

Flowchart :

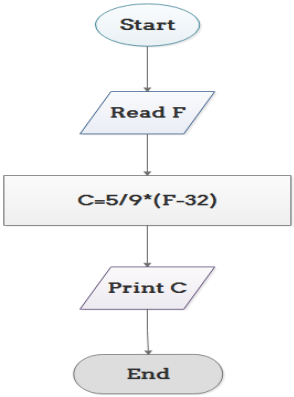


Gambar 2.2. Flowchart cetak angka 1 sampai 20

Contoh 2 :
Algoritma :

Langkah 1: Baca suhu di Fahrenheit,
Langkah 2: Hitung suhu dengan rumus C = 5/9 * (F-32),
Langkah 3: Cetak C,

Flowchart :

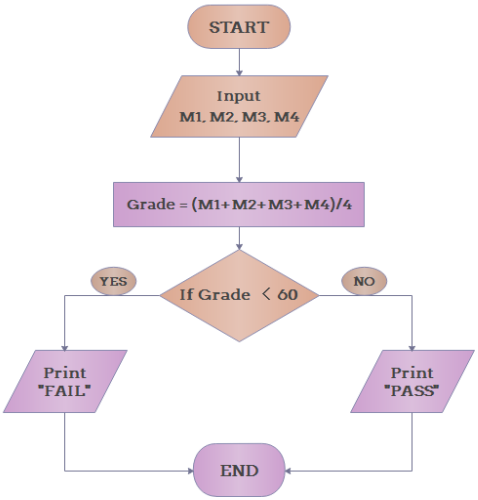


Gambar 2.3. Flowchart Konversi suhu dari Fahrenhait ke celcius

Contoh 3 :
Algoritma :

Langkah 1: Nilai input dari 4 program M1, M2, M3 dan M4,
Langkah 2: Hitung nilai rata-rata dengan rumus "Grade = (M1 + M2 + M3 + M4) / 4"
Langkah 3: Jika nilai rata-rata kurang dari 60, cetak "GAGAL", atau cetak "LULUS".

Flowchart :



Gambar 2.4. Flowchart mentukan Apakah Seorang Siswa Lulus Ujian atau Tidak

Dari penjelasan di atas maka dapat dapat disimpulkan bahwa diagram alur adalah representasi bergambar dari suatu algoritma, suatu algoritma dapat diekspresikan dan dianalisis melalui diagram alur. Algoritma menunjukkan kepada bahwa setiap langkah untuk mencapai solusi akhir, sementara diagram alur menunjukkan kepada Anda bagaimana melakukan proses dengan menghubungkan setiap langkah. Algoritma terutama menggunakan kata-kata untuk menggambarkan langkah-langkah sementara flowchart menggunakan bantuan simbol, bentuk dan panah untuk membuat proses lebih logis (Drozdek, Adam, 2001).

Soal Latihan :

1. Buatlah Flowchart dan algoritma bagaimana menampilkan bilangan genap dari 10 sampai 50.
2. Buatlah Flowchart dan algoritma bagaimana mencari bilangan ganjil dari angka 10 sampai 100.
3. Buatlah Flowchart dan algoritma bagaimana mencari bilangan prima.

BAB III
STRUKTUR DATA DALAM ALGORITMA

Struktur Data adalah cara mengumpulkan dan mengatur data sedemikian rupa sehingga kita dapat melakukan operasi pada data ini dengan cara yang efektif. Struktur Data adalah tentang memberikan elemen data dalam beberapa hubungan, untuk pengaturan dan penyimpanan yang lebih baik. Sebagai contoh, kami memiliki beberapa data yang memiliki, nama pemain "Virat" dan usia 26. Di sini "Virat" adalah tipe data String dan 26 adalah tipe data integer. Dalam bahasa yang sederhana, Struktur Data adalah struktur yang diprogram

untuk menyimpan data yang dipesan, sehingga berbagai operasi dapat dilakukan dengan mudah. Ini mewakili pengetahuan data yang akan disusun dalam memori. Ini harus dirancang dan diimplementasikan sedemikian rupa sehingga mengurangi kompleksitas dan meningkatkan efisiensi (The MIT Press, 2001)(Deitel, H. M. & Deitel P. J., 2001).

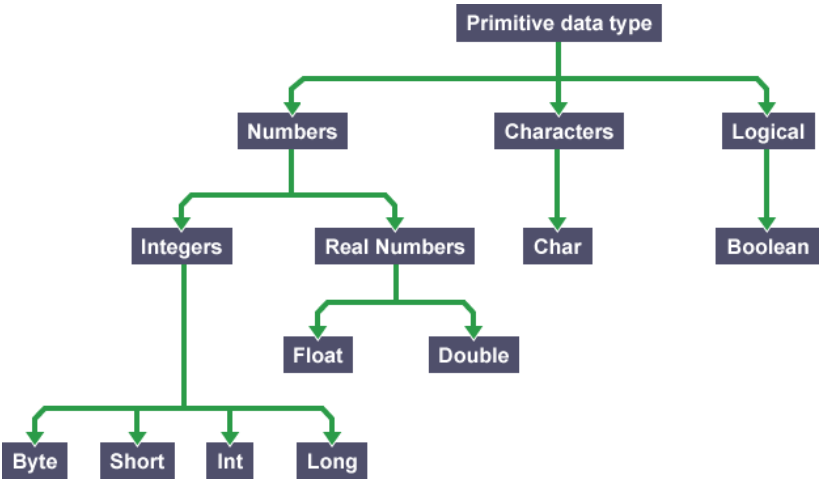
A. Tipe Data

Data diklasifikasikan ke dalam tipe, seperti seperangkat bilangan bulat (juga dikenal sebagai bilangan bulat) atau serangkaian karakter pencetakan. Berbagai jenis data direpresentasikan dalam berbagai cara di dalam komputer dan membutuhkan jumlah memori yang berbeda untuk menyimpannya. Mereka juga memiliki operasi berbeda yang dapat dilakukan pada mereka. Semua nilai yang termasuk tipe data yang sama akan direpresentasikan dengan cara yang sama. Tipe data yang paling umum didukung dalam bahasa pemrograman adalah:

Tabel 3.1 Tipe Data

Tpe Data	Contoh	Ukuran
Integer (bilangan bulat)	4, 27, 65535	1 to 8 bytes
Floating point (angka desimal)	4.2, 27.4, 5.63	4 to 8 bytes
Karakter	a, F, 3, $, £, #	1 byte
String	abc, hello world	Terbatas pada jumlah yang dapat disimpan di memori utama
Boolean	true or false	1 bit

Beberapa operasi yang dapat diterapkan pada nilai-nilai dari satu tipe data jelas tidak masuk akal ketika diterapkan pada nilai-nilai dari tipe lain. Sebagai contoh, kita dapat menghitung akar kuadrat dari bilangan bulat 4, tetapi bukan dari string "hello world". Tipe data mungkin berbeda dalam bahasa yang berbeda. Tipe data utama dikelompokkan dalam hierarki. Entah itu angka, karakter atau logis. Ada beberapa jenis nilai angka, termasuk perbedaan antara bilangan bulat dan bilangan floating-point.



Gambar 3.1 Tipe Data Primitif

Number

Komputer bekerja dengan memproses dan memanipulasi angka. Sebagian besar bahasa pemrograman membuat perbedaan antara bilangan bulat dan angka desimal. Perbedaan ini didasarkan pada bagaimana mereka diwakili di dalam mesin.

Integer adalah bilangan bulat yang direpresentasikan sebagai nilai biner. Sebagian besar bahasa pemrograman

menyediakan tipe data yang disebut 'integer', sering disebut 'int'.

Floating point numbers adalah angka yang memiliki bagian fraksional, biasanya direpresentasikan menggunakan dua komponen: angka utama dan bagian fraksional, yang masing-masing merupakan angka biner. Ini dikenal sebagai representasi floating-point. Sebagian besar bahasa pemrograman menyediakan satu atau lebih tipe data berdasarkan representasi floating-point. Mereka biasanya diberi nama seperti 'float', 'single', 'double', 'real' atau 'longreal'.

Tipe data angka dapat memengaruhi tipe perhitungan yang dapat dilakukan dalam suatu program.

Adding numbers

Suatu program perlu diberi tahu apakah ia akan bekerja dengan angka desimal atau bilangan bulat. Ekspresi aritmatika 3 + 3 melibatkan penambahan dua bilangan bulat. Hasilnya harus berupa nilai integer, jadi tipe datanya adalah integer. Ekspresi 2.5 + 3.5 melibatkan penambahan dua angka floating-point. Hasilnya adalah 6.0 dan disimpan sebagai angka floating-point.

Characters and strings

Setiap huruf, angka, dan tanda baca adalah karakter. Ada juga banyak karakter yang tidak terlihat di layar, seperti spasi, tab, dan karakter carriage-return. Sebagian besar bahasa pemrograman menyediakan tipe data yang disebut 'karakter' atau 'karakter'.

Set karakter

Setiap nilai mewakili satu karakter dari set yang telah ditentukan, seperti ASCII atau Unicode. Setiap karakter memiliki pola binernya sendiri.

String

Sebagian besar bahasa pemrograman memiliki tipe data yang

disebut string, yang digunakan untuk nilai data yang terdiri dari urutan karakter yang terurut, seperti "hello world". String dapat berisi urutan karakter apa pun, terlihat atau tidak terlihat, dan karakter dapat diulang. Jumlah karakter dalam string disebut panjangnya, dan "hello world" memiliki panjang 11 - terdiri dari 10 huruf dan 1 spasi. Biasanya ada batasan panjang maksimum string. Ada juga yang namanya string kosong, yang tidak mengandung karakter - panjang 0. String dapat berupa konstanta atau variabel. Jika konstan, biasanya ditulis sebagai urutan karakter yang dilampirkan dalam tanda kutip tunggal atau ganda, yaitu 'halo' atau "hello".

B. Konstanta dan Variabel

Dalam suatu program, nilai data dapat berupa konstanta atau variabel. Jika nilai-nilai adalah variabel mereka dapat diubah oleh program dan pengguna. Ketika suatu program dijalankan, nilai-nilai data disimpan dalam memori sementara mereka sedang dikerjakan.

Konstanta

Nilai data yang tetap sama setiap kali suatu program dieksekusi dikenal sebagai konstanta. Konstanta tidak diharapkan berubah.

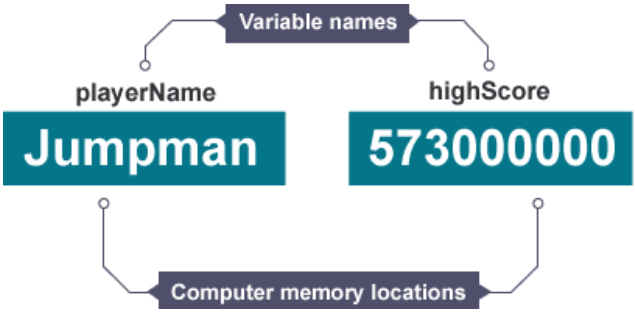
Konstanta literal adalah nilai aktual yang ditetapkan ke kode sumber. Contohnya mungkin string karakter "hello world". Nilai data "hello world" telah diperbaiki ke dalam kode. Konstanta Bernama adalah nilai-nilai di mana nama didefinisikan untuk digunakan daripada konstanta literal. Contoh dari hal ini mungkin menyatakan bahwa 'level awal' dari permainan selalu disebut sebagai 1.

Contoh konstanta dalam game :

1. satuan gravitasi
2. jumlah nyawa yang tersedia untuk pemain
3. jumlah waktu yang diizinkan untuk level dalam permainan

Variabel

Variabel adalah nilai data yang dapat berubah ketika pengguna ditanya pertanyaan, misalnya usia mereka. Variabel dapat berubah selama eksekusi program. Variabel adalah lokasi memori. Itu memiliki nama yang dikaitkan dengan lokasi itu. Lokasi memori digunakan untuk menyimpan data. Perbedaan utama ketika membandingkan konstanta ke variabel adalah bahwa nilai yang terkait dengan nama variabel dapat berubah selama eksekusi program. Misalnya 'highScore' perlu variabel untuk berubah sepanjang permainan.



Gambar 3.2 Contoh sebuah Variabel

Konten dan organisasi memori komputer tidak diperbaiki - demikian juga nilai yang ditunjukkan oleh variabel. Saat data dibaca dari variabel, konten lokasi memori disalin dan digunakan dalam perhitungan.

C. Array

Array adalah struktur data yang berisi sekelompok elemen. Biasanya elemen-elemen ini semua tipe data yang sama, seperti integer atau string. Array biasanya digunakan dalam program komputer untuk mengatur data sehingga seperangkat nilai terkait dapat dengan mudah disortir atau dicari.

Misalnya, mesin pencari dapat menggunakan array untuk menyimpan halaman Web yang ditemukan dalam pencarian yang dilakukan oleh pengguna. Saat menampilkan hasil, program akan menampilkan satu elemen array pada satu waktu. Ini dapat dilakukan untuk sejumlah nilai tertentu atau sampai semua nilai yang disimpan dalam array telah menjadi output. Sementara program dapat membuat variabel baru untuk setiap hasil yang ditemukan, menyimpan hasil dalam array adalah cara yang jauh lebih efisien untuk mengelola memori.

Sintaks untuk menyimpan dan menampilkan nilai-nilai dalam array biasanya terlihat seperti ini:

```
arrayname [0] = "Ini";
arrayname [1] = "is";
arrayname [2] = "pretty simple.";
print arrayname [0];
print arrayname [1];
print arrayname [2];
```

Perintah di atas akan mencetak tiga nilai pertama array, atau "Ini cukup sederhana." Dengan menggunakan loop "while" atau "for", programmer dapat memberi tahu program untuk menampilkan setiap nilai dalam array sampai nilai terakhir tercapai. Jadi tidak hanya array membantu mengatur memori lebih efisien, mereka membuat pekerjaan programmer lebih efisien juga.

Array 1 Dimensi

Array satu dimensi (atau array satu dimensi) adalah jenis array linier. Mengakses elemen-elemennya melibatkan sebuah subskrip tunggal yang dapat mewakili indeks baris atau kolom. Sebagai contoh, pertimbangkan deklarasi di dalam Nama Array [10]; yang menyatakan array satu dimensi dari sepuluh bilangan bulat. Di sini, array dapat menyimpan sepuluh

elemen bertipe int. Array ini memiliki indeks mulai dari nol hingga sembilan. Misalnya, ekspresi Nama Array [0] dan Nama Array [9] masing-masing adalah elemen pertama dan terakhir. Untuk vektor dengan pengalamatan linier, elemen dengan indeks i terletak di alamat B + c × i, di mana B adalah alamat basis tetap dan c konstanta tetap, kadang-kadang disebut kenaikan atau langkah alamat.
Jika indeks elemen yang valid dimulai pada 0, konstanta B hanyalah alamat dari elemen pertama array. Untuk alasan ini, bahasa pemrograman C menentukan bahwa indeks array selalu dimulai pada 0; dan banyak programmer akan memanggil elemen itu "nol" daripada "pertama".
Namun, seseorang dapat memilih indeks elemen pertama dengan pilihan yang sesuai dari alamat basis B. Sebagai contoh, jika array memiliki lima elemen, indeks 1 hingga 5, dan alamat dasar B diganti dengan B + 30c, maka indeks elemen-elemen yang sama adalah 31 hingga 35. Jika penomorannya tidak dimulai dari 0, konstanta B mungkin bukan alamat elemen apa pun.

Array Multi Dimensi

Untuk array multidimensi, elemen dengan indeks i, j akan memiliki alamat B + c · i + d · j, di mana koefisien c dan d adalah kenaikan alamat baris dan kolom, masing-masing.
Lebih umum, dalam array k-dimensional, alamat elemen dengan indeks i1, i2, ..., ik adalah

B + c1 · i1 + c2 · i2 +… + ck · ik.

Misalnya: int a [2] [3];
Ini berarti bahwa array a memiliki 2 baris dan 3 kolom, dan array tersebut bertipe integer. Di sini kita dapat menyimpan 6 elemen, mereka akan disimpan secara linear tetapi mulai dari baris pertama linier kemudian dilanjutkan dengan baris kedua.

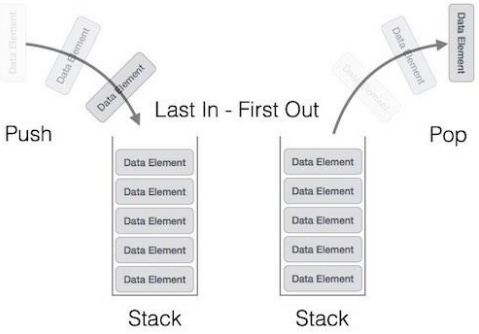
Array di atas akan disimpan sebagai a11, a12, a13, a21, a22, a23.
Rumus ini hanya membutuhkan k perkalian dan penambahan k, untuk array apa pun yang dapat ditampung dalam memori. Selain itu, jika koefisien apa pun adalah daya tetap 2, penggandaan dapat diganti dengan pergeseran bit.
Koefisien ck harus dipilih sehingga setiap indeks yang valid akan memetakan peta ke alamat elemen yang berbeda. Jika nilai hukum minimum untuk setiap indeks adalah 0, maka B adalah alamat elemen yang indeksnya nol. Seperti dalam kasus satu dimensi, indeks elemen dapat diubah dengan mengubah alamat basis B. Jadi, jika array dua dimensi memiliki baris dan kolom yang diindeks masing-masing dari 1 hingga 10 dan 1 hingga 20, maka menggantikan B dengan B + c1 - 3c2 akan menyebabkan mereka dinomori kembali dari 0 hingga 9 dan 4 hingga 23, masing-masing. Mengambil keuntungan dari fitur ini, beberapa bahasa (seperti FORTRAN 77) menetapkan bahwa indeks array dimulai pada 1, seperti dalam tradisi matematika sedangkan bahasa lain (seperti Fortran 90, Pascal dan Algol) memungkinkan pengguna memilih nilai minimum untuk setiap indeks.

D. Stack

Stack adalah struktur data linier yang mengikuti urutan tertentu di mana operasi dilakukan. Urutannya mungkin LIFO (Last In First Out) atau FILO (First In Last Out).
Tumpukan adalah Tipe Data Abstrak, yang biasa digunakan di sebagian besar bahasa pemrograman. Ini dinamai stack karena berperilaku seperti tumpukan dunia nyata, misalnya - setumpuk kartu atau tumpukan piring, dll.
Tumpukan dunia nyata memungkinkan operasi di satu ujung saja. Sebagai contoh, kita dapat menempatkan atau mengeluarkan kartu atau piring dari bagian atas tumpukan

saja. Demikian juga, Stack memungkinkan semua operasi data hanya pada satu ujung. Pada waktu tertentu, kami hanya dapat mengakses elemen atas tumpukan. Fitur ini menjadikannya struktur data LIFO. LIFO adalah singkatan dari Last-in-first-out. Di sini, elemen yang ditempatkan (dimasukkan atau ditambahkan) terakhir, diakses terlebih dahulu. Dalam terminologi stack, operasi penyisipan disebut operasi PUSH dan operasi penghapusan disebut operasi POP.

Diagram berikut menggambarkan tumpukan dan operasinya :



Gambar 3.3. Contoh Stack

Operasi tumpukan mungkin melibatkan menginisialisasi tumpukan, menggunakannya dan kemudian melakukan inisialisasi. Terlepas dari barang-barang dasar ini, tumpukan digunakan untuk dua operasi utama berikut –

push () - Mendorong (menyimpan) elemen pada stack.

pop () - Menghapus (mengakses) suatu elemen dari stack.

Ketika data di PUSH ke stack.

Untuk menggunakan tumpukan secara efisien, kita perlu memeriksa status tumpukan juga. Untuk tujuan yang sama, fungsi berikut ditambahkan ke tumpukan –

peek () - dapatkan elemen data teratas dari stack, tanpa menghapusnya.

isFull () - periksa apakah stack penuh.

isEmpty () - periksa apakah stack kosong.

Setiap saat, kami mempertahankan pointer ke data PUSHed terakhir pada stack. Karena penunjuk ini selalu mewakili bagian atas tumpukan, maka dinamai atas. Pointer atas memberikan nilai atas tumpukan tanpa benar-benar menghapusnya.

E. Queue

Antrian adalah struktur data abstrak, agak mirip dengan Stacks. Tidak seperti tumpukan, antrian terbuka di kedua ujungnya. Satu ujung selalu digunakan untuk memasukkan data (enqueue) dan yang lainnya digunakan untuk menghapus data (dequeue). Antrian mengikuti metodologi First-In-First-Out, yaitu, item data yang disimpan terlebih dahulu akan diakses terlebih dahulu.

Gambar 3.4 Ilustrasi Queue

Contoh antrian dunia nyata dapat berupa jalan satu arah satu arah, tempat kendaraan masuk terlebih dahulu, keluar terlebih dahulu. Lebih banyak contoh dunia nyata dapat dilihat sebagai antrian di jendela tiket dan halte.

Seperti yang sekarang kita pahami bahwa dalam antrian, kita mengakses kedua ujungnya untuk alasan yang berbeda. Diagram berikut yang diberikan di bawah ini mencoba menjelaskan representasi antrian sebagai struktur data.

Seperti di tumpukan, antrian juga dapat diimplementasikan menggunakan Array, Linked-list, Pointer dan Structures. Demi kesederhanaan, akan mengimplementasikan antrian menggunakan array satu dimensi.

Operasi antrian mungkin melibatkan menginisialisasi atau mendefinisikan antrian, menggunakannya, dan kemudian menghapusnya sepenuhnya dari memori. Di sini kita akan mencoba memahami operasi dasar yang terkait dengan antrian.

enqueue () - menambahkan (menyimpan) item ke antrian.

dequeue () - menghapus (mengakses) item dari antrian.

Lebih sedikit fungsi yang diperlukan untuk membuat operasi antrian yang disebutkan di atas efisien. Ini adalah

peek () - Mendapat elemen di bagian depan antrian tanpa menghapusnya.

isfull () - Memeriksa apakah antrian penuh.

isempty () - Memeriksa apakah antrian kosong.

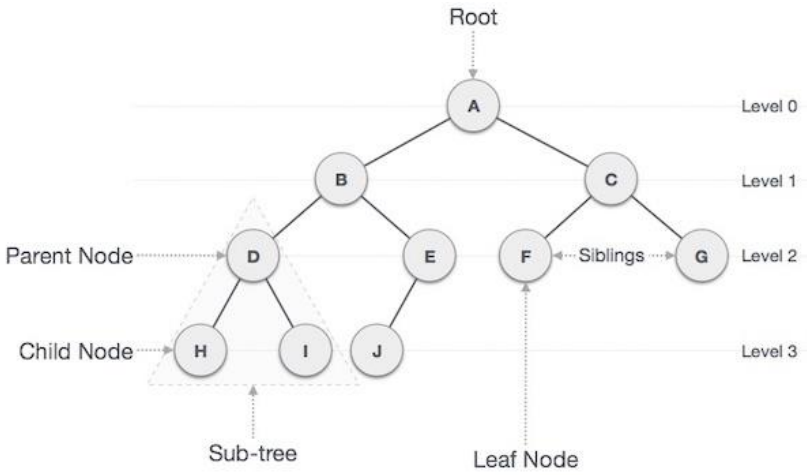
Dalam antrian, kami selalu mengeluarkan data (atau mengakses), diarahkan oleh penunjuk depan dan saat memohon (atau menyimpan) data dalam antrian, kami membantu penunjuk belakang.

F. Tree

Tree merepresentasikan node yang dihubungkan oleh edge. Kami akan membahas pohon biner atau pohon pencarian biner secara khusus. Binary Tree adalah struktur data khusus yang digunakan untuk keperluan penyimpanan data. Pohon biner memiliki kondisi khusus sehingga setiap simpul dapat memiliki maksimal dua anak. Pohon biner memiliki manfaat dari susunan yang diurutkan dan daftar yang ditautkan karena pencarian secepat di dalam susunan yang diurutkan dan operasi penyisipan atau penghapusan sama cepatnya dengan dalam daftar yang ditautkan.

Berikut ini adalah istilah penting sehubungan dengan pohon.

1. Path - Path mengacu pada urutan node di sepanjang tepi pohon.
2. Root - Node di bagian atas pohon disebut root. Hanya ada satu root per pohon dan satu path dari node root ke node apa pun.
3. **Parent** - Setiap simpul kecuali simpul akar memiliki satu tepi ke atas ke simpul yang disebut induk.
4. **Child** - Simpul di bawah simpul tertentu yang dihubungkan dengan ujungnya ke bawah disebut simpul anaknya.
5. Leaf - Node yang tidak memiliki simpul anak disebut node daun.
6. Subtree - Subtree mewakili turunan dari sebuah node.
7. **Visiting** - Mengunjungi mengacu pada memeriksa nilai suatu node ketika kontrol ada pada node.
8. Traversing - Traversing berarti melewati node dalam urutan tertentu.
9. Levels - Level dari sebuah node mewakili generasi dari sebuah node. Jika simpul root berada di level 0, maka simpul anak berikutnya adalah di level 1, cucunya ada di level 2, dan seterusnya.
10. **keys** - Kunci mewakili nilai suatu simpul berdasarkan operasi pencarian yang akan dilakukan untuk suatu simpul.

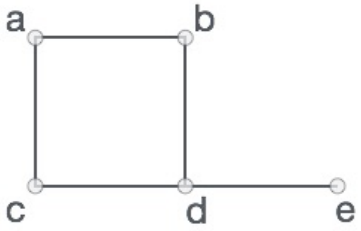


Gambar 3.5 Contoh sebuah Tree

G. Graph

Graph adalah representasi gambar dari sekumpulan objek di mana beberapa pasang objek dihubungkan oleh tautan. Objek-objek yang saling berhubungan diwakili oleh titik-titik yang disebut sebagai simpul, dan tautan yang menghubungkan simpul-simpul tersebut disebut tepi.

Secara formal, grafik adalah sepasang himpunan (V, E), di mana V adalah himpunan simpul dan E adalah himpunan tepi, menghubungkan pasangan simpul. Lihatlah grafik berikut :



Gambar 3.6 Contoh sebuah Graph

Dalam grafik di atas,
V = {a, b, c, d, e}
E = {ab, ac, bd, cd, de}

Struktur Data Grafik

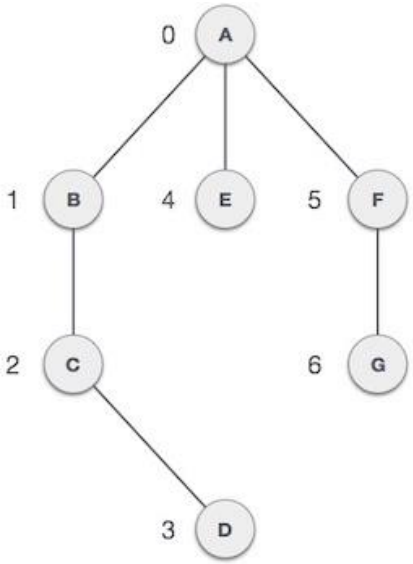
Grafik matematika dapat direpresentasikan dalam struktur data. Kita dapat merepresentasikan grafik menggunakan larik simpul dan larik tepi dua dimensi. Sebelum melangkah lebih jauh, mari kenali beberapa istilah penting.

Vertex - Setiap node grafik direpresentasikan sebagai vertex. Dalam contoh berikut, lingkaran berlabel mewakili simpul. Jadi, A ke G adalah simpul. Dapat mewakili dengan menggunakan array seperti yang ditunjukkan pada gambar berikut. Di sini A dapat diidentifikasi dengan indeks 0. B dapat diidentifikasi menggunakan indeks 1 dan seterusnya.

Edge - Tepi mewakili jalur antara dua simpul atau garis antara dua simpul. Dalam contoh berikut, garis-garis dari A ke B, B ke C, dan seterusnya mewakili tepi. Dapat menggunakan array dua dimensi untuk mewakili array seperti yang ditunjukkan pada gambar berikut. Di sini AB dapat direpresentasikan sebagai 1 pada baris 0, kolom 1, BC sebagai 1 pada baris 1, kolom 2 dan seterusnya, menjaga kombinasi lainnya sebagai 0.

Adjacency - Dua simpul atau simpul berdekatan jika mereka terhubung satu sama lain melalui tepi. Dalam contoh berikut, B berbatasan dengan A, C berbatasan dengan B, dan seterusnya.

Path - Jalur mewakili urutan tepi antara dua simpul. Dalam contoh berikut, ABCD mewakili jalur dari A ke D.



Gambar 3.7 Contoh Graph Data Structure

Soal latihan :

1. Berapa ukuran tipe data Integer (bilangan bulat) dan berikan contoh tipe data tersebut.
2. Apa yang kamu ketahui tentang Konstanta dan Variabel dan berikan contohnya.
3. Apa yang kamu ketahui Array dan berikan contah array 1 dimensi dan array 2 dimensi.
4. Bagaimana cara kerja dari Stack.
5. Apa yang kamu ketahui Queue dan bagaimana cara kerjanya

BAB IV
MENGHITUNG LUAS DAN KELILING LINGKARAN MENGGUNAKAN MATLAB

A. Permasalahan

Buatlah flowchart untuk luas dan keliling lingkaran ditampilkan ke layar.
Contoh:
Diktehaui panjang jari-jari lingkaran, dari inputan jari-jari maka akan menghasilakan Luas dan keliling lingkaran

B. Cara Penyelesaian Masalah

Inputan dengan Jari-jari lingkaran akan menghasilkan luas dan keliling lingkaran. Formula untuk menghitung luas dan keliling lingkaran adalah :
Luas lingkaran = Phi x r^2
Keliling lingkaran = 2 x phi x r
Dimana r adalah jari-jari lingkaran

C. Kebutuhan Data

1. Phi adalah konstanta, bertipe data pecahan
2. Jari-jari adalah variabel yang menampung panjang jari-jari lingkaran, bertipe data pecahan
3. Luas adalah variabel penampung hasil
4. Keliling sebagai variabel penampung hasil

Deklarasi dan inisialisasi konstanta dan variabel pada flowchart dapat dibuat sebagai berikut:

1. **Input**

Jari-jari (r) merupakan panjang jari-jari lingkaran dan disimpan dalam variabel r.

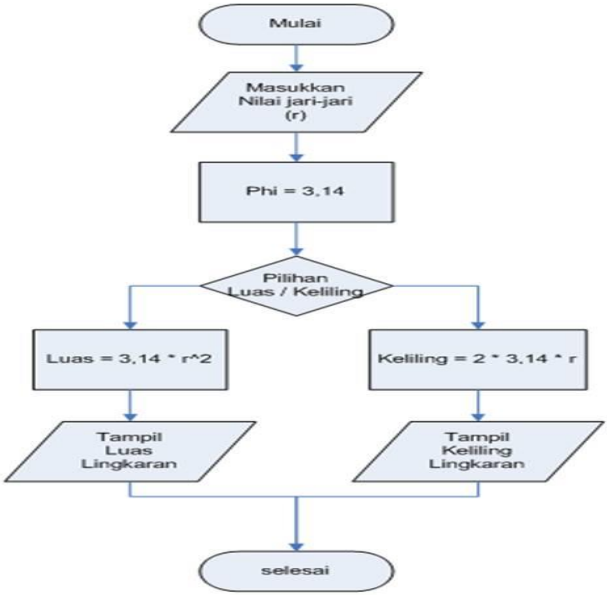
2. **Output**

Keluaran pada permasalahan ini ada dua yaitu nilai Luas lingkaran (L) dan keliling lingkaran (K).

D. Penyelesaian Proses

Ada dua kegiatan yang terjadi, yaitu perhitungan luas lingkaran dan perhitungan keliling lingkaran.

E. Flowchart Keliling dan Luas Lingkaran



Gambar 4.1 Flowchart keliling dan luas lingkaran

F. Contoh program

Carilah nilai Luas serta keliling dari sebuah lingkaran jika jari-jari lingkaran adalah 100 cm !

Cara Penyelesaian dengan menggunakan Matlab :

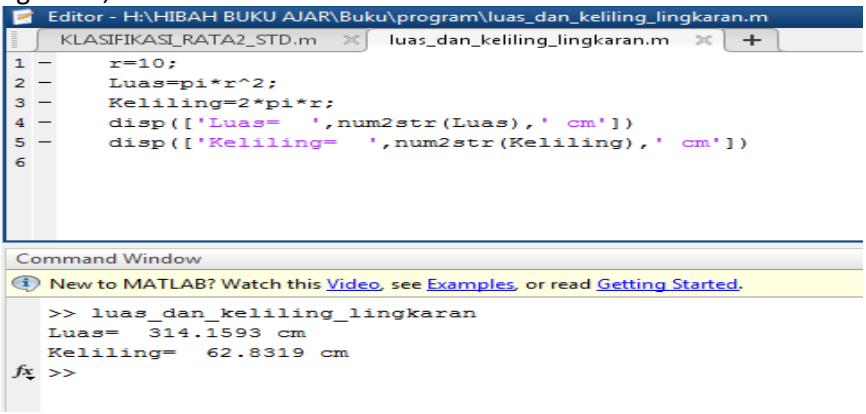
1. Pertama, buka Aplikasi Matlab di computer anda.
2. Kedua, setelah program Matlab terbuka maka pilih **File—>New—>M-File**
3. Ketiga, coba buat program di bawah ini :

```
r=100
L=pi*r^2
K=2*pi*r
disp(['Luas Lingkaran =  ',num2str(L),' cm2'])
disp(['Keliling Lingkaran =  ',num2str(K),' cm'])
```

kemudian setelah program diatas ditulis dan disimpan. Kemudian jalankan dengan memilih **Tools—>Run**, maka hasil dari program yang ditulis hasilnya dapat dilihat dibawah ini :

L= 3141,593 cm

Kg= 628,319 cm



Gambar 4.2 Hasil Program luas dan keliling lingkaran

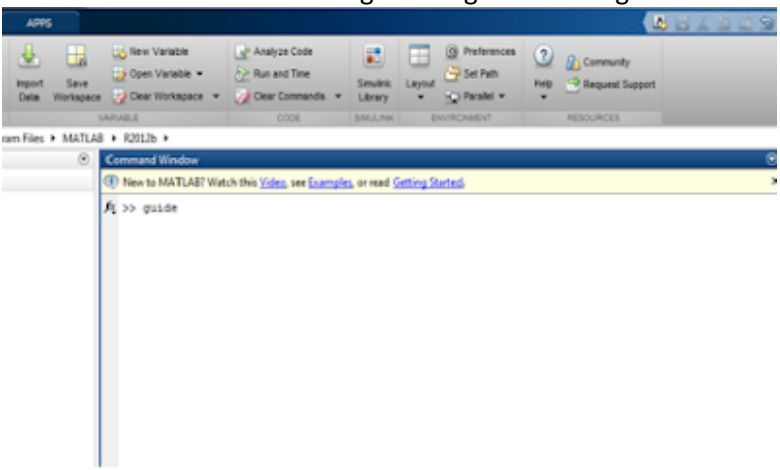
Penjelasan dari listing program di atas adalah sebagai berikut :

r=10; untuk membuat variabel r = 100

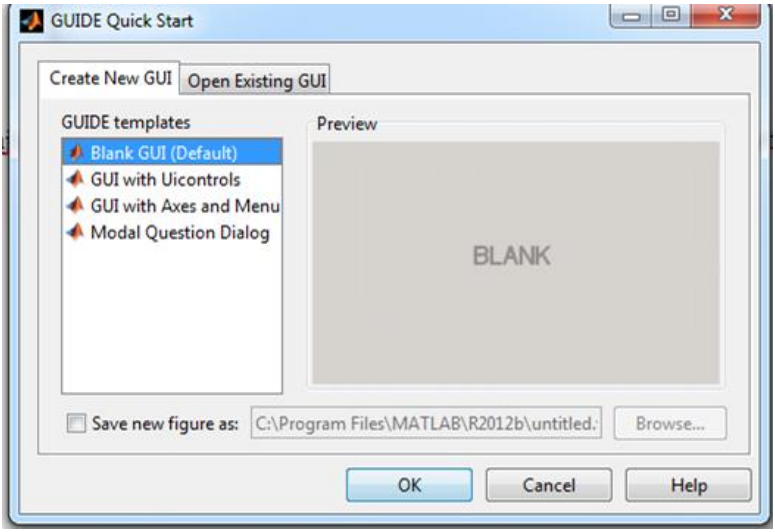
L=pi*r^2; untuk menghitung Luas Lingkaran.

K =2*pi*r; untuk menghitung keliling Lingkaran.
disp(['Luas Lingkaran = ',num2str(L),' cm2']), Untuk menampilkan Luas Lingkaran dan kata 'cm2' pada hasil yang di dapat.
disp(['Keliling Lingkaran= ',num2str(K),' cm']), Untuk menampilkan Hasil Keliling Lingkaran dan kata 'cm' pada hasil yang di dapat.

Contoh Program menggunkakan Fasilitas GUI (ANDI Yogyakarta 2000).
Melalui command matlab dengan mengetikkan: >> guide



Gambar 4.3 Tamplian Matlab dengan tulisan guide
Maka dengan mengetik guide di command prompt, maka akan muncul tampilan kotak dialog GUIDE Quick Start.



Gambar 4.4 Tamplian Matlab dengan aplikasi guide

Membuat Contoh Aplikasi GUIDE MATLAB

1. Mencari Nilai **Luas Lingkaran dan Keliling Lingkaran**

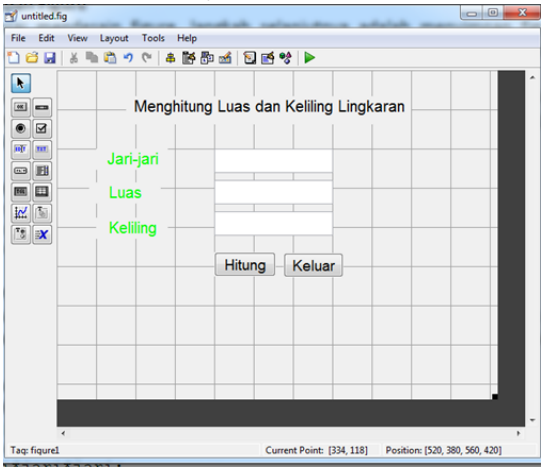
Pada contoh aplikasi yang dibuat kali ini yaitu bagaimana kita mencari Luas dan keliling lingkaran dengan menggunakan Guide pada Matlab. Langkah-langkah untuk membuat aplikasi tersebut adalah :

a. Mendesain Tampilan

Dalam mendesain Tampilan, pertama-tama komponen apa yang dibutuhkan dalam membuat aplikasi menhitung nilai Luas dan keliling Lingkaran. Setelah desain tampilan selesai, maka dalam membuat aplikasi menghitung Luas dan keliling lingkaran yang dibutuhkan komponennya adalah :

1. 3 buah edit text, untuk variable input dan dua variable output.

2. 2 tombol pushbutton/togglebutton untuk mulai melakukan proses perhitungan.
3. 4 buah static text, untuk tulisan.



Gambar 4.5 Desain Tampilan

b. Mendesain komponen yang sudah kita pilih

Untuk mendesain komponen, maka kita akan mengatur property inspector dari masing-masing komponen yang kita pilih.

Tabel 4.1 Cara Mengatur komponen Properti

KOMPONEN	PROPERTY INSPECTOR			
	FONT SIZE	FONTWEIGHT	STRING	TAG
Static text 1	12	`Bold	Menghitung Luas dan Keliling Lingkaran	text1
Static text 2	10	Normal (default)	Jar-Jari	text2
Static text 3	10	Normal (default)	Luas	text3
Static text 4	10	Normal (default)	Keliling	text4
Edit text 1	10	Normal (default)	Kosongkan	edit1
Edit text 2	10	Normal (default)	Kosongkan	edit2
Edit text 3	10	Normal (default)	Kosongkan	edit3
Pushbutton 1	10	`Bold	Hitung	btn_hitung
Pushbutton 2	10	`Bold	Edit	btn_exit

c. Menyimpan Tampilan

Proses selanjutnya adalah adalah menyimpan Tampilan. Desain tampilan yang kita buat, silahkan disimpan dalam file bernama lingkaran.fig, maka secara otomatis dibuatkan kerangka m-file dengan nama yang sama, yaitu lingkaran.m.

Fungsi yang muncul di m-file yang perlu kita edit adalah :

1. edit1

Di bawah function edit1_callback, tambahkan program menjadi berikut:

```
r=str2num(get(handles.edit1, 'String'));
handles.jari=r;
guidata(hObject, handles)
```

2. edit2 dan edit3

untuk function edit1_callback dan function edit2_callback, kita tidak perlu menambahkan kode apapun dibawahnya, karena hanya berfungsi untuk menampilkan hasil dari perhitungan.

3. btn_hitung

Di bawah function btn_hitung_callbck, tambahkan program menjadi berikut:

```
%kode ini diekskusi jika kita menekan tombol hitung
r=handles.jari;
L=pi*r*r
K=2*pi*r
%menampilkan hasil perhitungan
set(handles.edit2, 'string', L);
set(handles.edit3, 'string',K);
```

4. btn_exit

```
%untuk keluar dari apalikasi
delete(handles.figure1)
```

d. Running GUI

Proses selanjutnya adalah melakukan uji coba program yang sudah kita buat. Untuk menjalankan aplikasi yang telah dibuat dengan mengklik tombol Run dari jendela figure atau dari jendela debug m-file (tekan F5), sehingga akan muncul tampilan berikut.

Gambar 4.6 Tampilan Hasil program

Soal Latihan :

1. Bualah Flowchart dan Progam menggunakan Pemrograman Matlab untuk mencari Luas Persegi Panjang.
2. Bualah Flowchart dan Progam menggunakan Pemrograman Matlab untuk mencari Keliling Persegi Panjang.
3. Bualah Flowchart dan Progam menggunakan Pemrograman Matlab untuk mencari Luas Segitiga Siku-siku.
4. Bualah Flowchart dan Progam menggunakan Pemrograman Matlab untuk mencari Keliling Segitiga siku-siku.
5. Bualah Flowchart dan Progam menggunakan Pemrograman Matlab untuk mencari Volume Balok.

BAB V
KONVERSI SUHU MENGGUNAKAN MATLAB

A. Permasalahan

Bagaimana membuat flowchart dan program aplikasi untuk mengkonversi suhu dari derajat Celcius ke dalam derajat Reamur dan derajat Fahrenheit. Inputan Suhu dalam derajat Celcius akan dimasukkan oleh user. Output dari konversi yaitu Suhu dalam derajat Reamur dan derajat Fahrenheit akan ditampilkan ke layar komputer.

Misal :

Input : Masukkan suhu dalam derajat Celcius : 100

Output : besarnya suhu dalam derajat Reamur : 80

Output : Besarnya suhu dalam derajat Fahrenheit : 500

B. Penyelesaian Masalah

Untuk membuat flowchart dan aplikasi konversi suhu dari derajat celcius ke dalam suhu derajat reamur dan derajat fahrenheit akan dibutuhkan data berupa nilai dari suhu derajat celcius. Setelah diketahui suhu dalam derajat celcius, maka kita akan mendapatkan suhu dalam derajat reamur dan derajat Fahrenheit dengan menggunkan rumus :

R = C * 4/5 (suhu dalam derajat Reamur)
F = C * 9/5 + 32 (suhu dalam derajat celcius)

C. Data Yang Dibutuhkan

- **C** merupakan variabel bertipe data pecahan dan untuk menyimpan suhu dalam derajat Celcius.
- **R** merupakan variabel Reamur yang bertipe data pecahan untuk menyimpan suhu dalam derajat Reamur.
- **F** merupakan variabel Fahrenheit yang bertipe data pecahan untuk menyimpan suhu dalam derajat Fahrenheit.

D. Inisialisasi dan Deklarasi

Variabel C merupakan inputan dalam bentuk pecahan, sedangkan variable R dan F merupakan outputan dalam bentuk pecahan. Variable C merupakan inputan yang diinputkan oleh pengguna dan nilainya terserah dari pengguna. Sehingga apabila nilai C dimasukkan, secara otomatis nilai R dan F akan disimpan dan ditampilkan kedalam layar computer.

E. Input

Dalam aplikasi konversi suhu, inputan diberi variable C. sehingga variable C ini melambangkan inputan dari suhu dalam celcius.

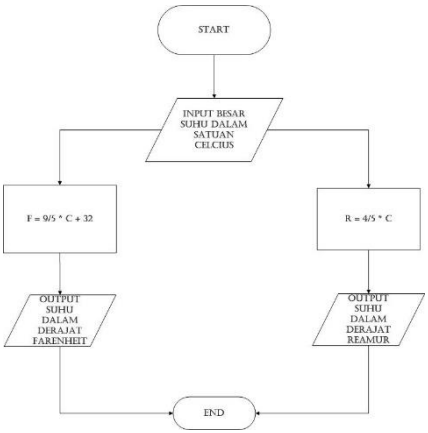
F. Output

Output dari apliaksi ini adalah reamur dan Fahrenheit yang di beri variable R dan F.

G. Proses Penyelesaian

Untuk meyelesaikan permasalahan dalam aplikasi konversi suhu dari derajat celcius ke derajat reamur dan Fahrenheit, maka dipergunakan rumus yang telah disebutkan sebelumnya.

H. Flowchart



Gambar 5.1 Flowchart konversi suhu

I. Contoh program

Tentukanlah Suhu dalam derajat Reamur dan Fahrenheit apabila diketahui suhu dalam derajat Celcius adalah 100^0 C !

Penyelesaian :

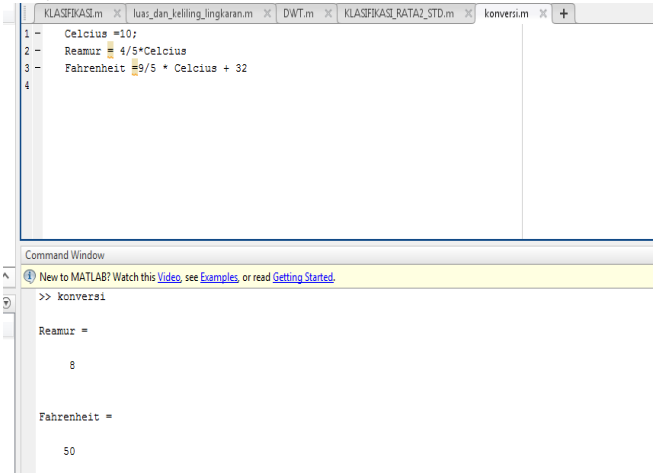
4. Jalankan Aplikasi Matlab
5. Kemudian pilih **File—>New—>M-File**
6. Kemudian Ketik listing program aplikasi konversi suhu di bawah ini :

```
C =100;
R = 4/5*C;
F =9/5 * C + 32;
disp(['Suhu dalam Reamur =  ',num2str(R),' derajat Reamur'])
disp(['Suhu dalam Fahrenheit =  ',num2str(F),' derajat Fahrenheit'])
```

Setelah selesai mengetikkan listing di atas simpan dan jalankan dengan memilih **Tools—>Run**, dan hasil yang akan di dapat adalah sebagai berikut :

R = 80;

F = 500;



Gambar 5.2 Hasil Aplikasi Program Reamur dan Fahrenheit

Penjelasan untuk listing program di atas :

C=100→ memasukkan nilai suhu divariabel C dengan Nilai 100

R= 4/5*C → Memasukkan rumus Reamur dengan variable R.

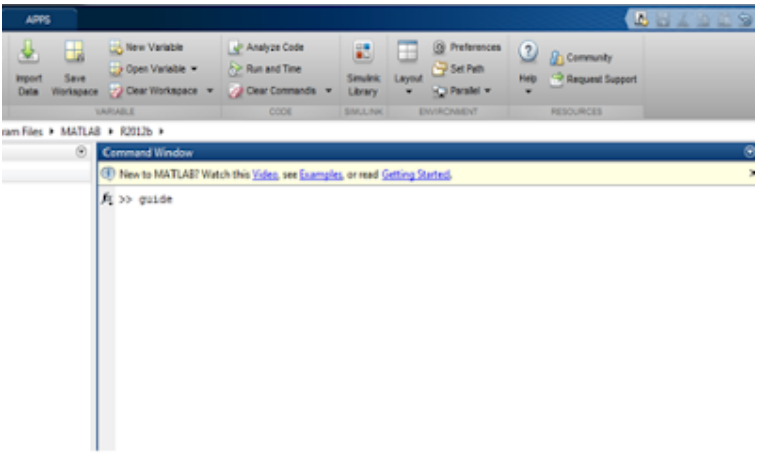
F = 9/5*C + 32 → Memasukkan rumus Fahrenheit dengan variable F.

disp([‘Suhu dalam derajat Reamur = ‘,num2str(R),’ derajat Reamur’]) → Menampilakan nilai Reamur di layar komputer.

disp([‘Suhu dalam derajat Fahrenheit = ‘,num2str(F),’ derajat Fahrenheit’]) → Menampilkan nilai suhu dalam derajat Fahrenheit.

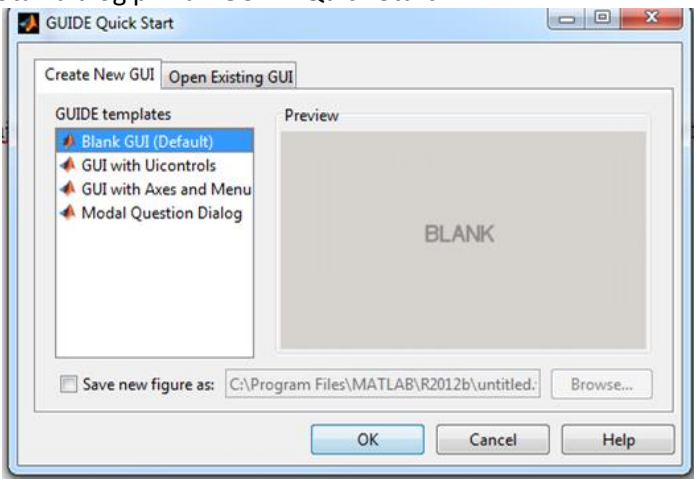
Contoh Program menggunkakan Fasilitas GUI

Melalui command matlab dengan mengetikkan: >> guide



Gambar 5.3 Tamplian Matlab dengan tulisan guide

Setelah kita menuliskan kata guide, maka akan ditampilkan kotak dialog pilihan GUIDE Quick Start.



Gambar 5.4 Tamplian Matlab dengan aplikasi guide

Membuat Contoh Aplikasi GUIDE MATLAB

1. Menghitung Konversi suhu dari Celcius ke Reamur dan Fahrenheit

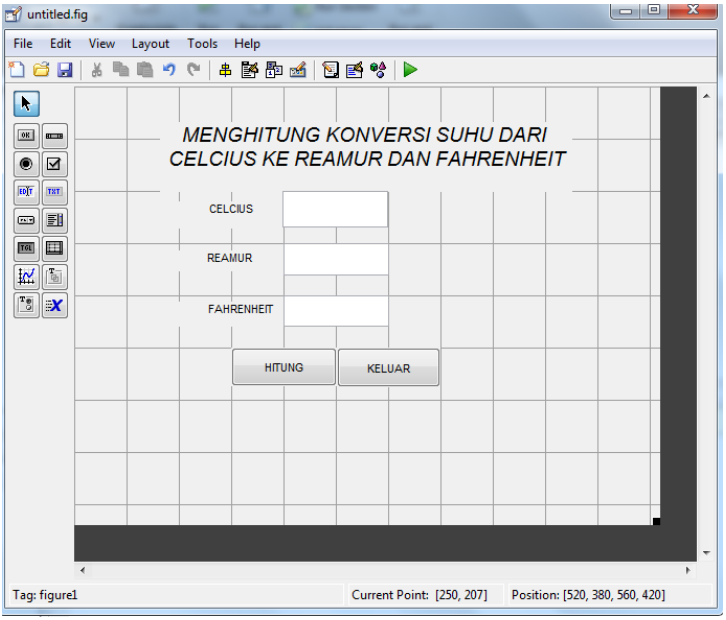
Contoh aplikasi yang dibuat adalah suhu dari Celcius ke Reamur dan Fahrenheit. Untuk bias menampilkan aplikasi konversi suhu dari derajat Celsius ke derajat reamur dan Fahrenheit, maka cara membuatnya adalah sebagai berikut :

a. Mendesain Tampilan

Dalam mendesain tampilan pada aplikasi yang akan kita buat, maka yang perlu kita pikirkan adalah komponen apa saja yang perlu kita pakai. Pada aplikasi konversi suhu dari derajat celcius ke derajat reamur dan Fahrenheit, maka komponen yang kita butuhkan adalah :

1. Tiga buah komponen edit text : untuk menampilkan nilai celcius, reamur dan Fahrenheit.
2. Dua buah komponen tombol pushbutton/togglebutton : untuk melakukan proses perhitungan.
3. empat buah komponen static text untuk membuat tulisan yang kita desain.

Gambar 5.5 merupakan hasil tampilan yang sudah didisain.



Gambar 5.5 Desain Tampilan

b. Mendesain **Layout Komponen**

Setelah semua komponen kita mendesain, maka dalam tiap komponen harus kita atur property inspector untuk keperluan yang kita gunakan.

Tabel 5.1 Cara Mengatur komponen Properti

KOMPONEN	PROPERTY INSPECTOR			
	FONT SIZE	FONT WEIGHT	STRING	TAG
Static text 1	14	Bold	Menghitung Konversi suhu dari Celcius ke Reamur dan Fahrenheit	Text 1

Static text 2	10	Normal	Celcius	Text 2
Static text 3	10	Normal	Reamur	Text 3
Static text 4	10	Normal	Fahrenheit	Text 4
Edit text 1	10	Normal		Edit1
Edit text 2	10	Normal		Edit2
Edit text 3	10	Normal		Edit3
Pushbutton 1	10	Bold	Hitung	Button_hitung
Pushbutton 2	10	Bold	Keluar	Button_keluar

C. Save Aplikasi

Aplikasi yang sudah kita buat dengan desain yang sudah ada, maka kita bias menyimpan aplikasi tersebut dengan nama suhu.m. Supaya aplikasi yang sudah kita buat desainnya bisa berjalan, maka ada fungsi-fungsi yang harus kita beri kode/program.

1. edit1_Callback untuk inputan suhu celcius
2. edit2_Callback untuk outputan suhu reamur
3. edit3_Callback untuk outputan suhu fahrenheit
4. btn_hitung_Callback untuk memproses rumus konversi
5. btn_exit_Callback untuk keluar dari aplikasi

Dibawah ini adalah kode program yang haru kita masukkan :

1. edit1 → didalam aplikasi konversi ada function edit1_callback, maka tambahkan program seperti dibawah ini:

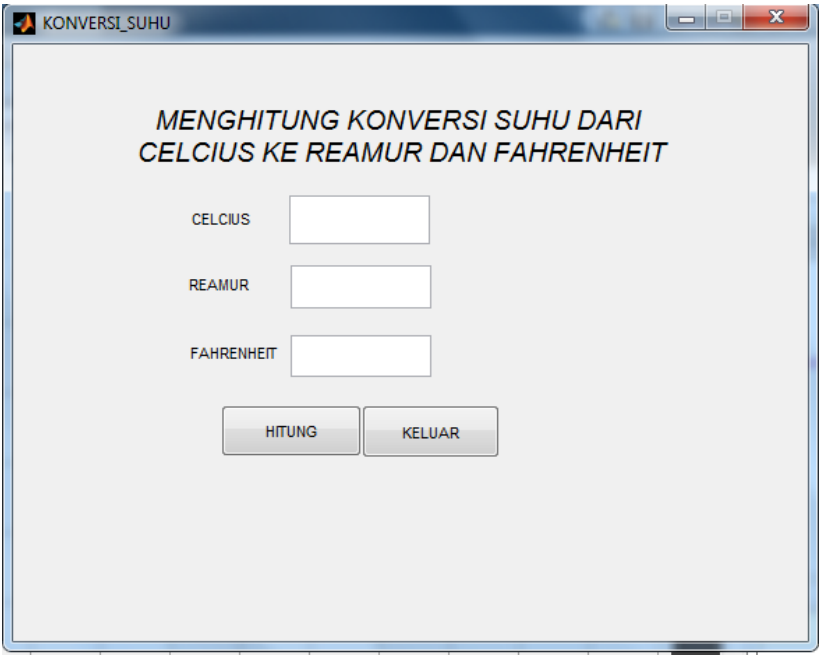
   ```
   C=str2num(get(handles.edit1, 'String'));
   handles.celcius=C;
   guidata(hObject, handles)
   ```

2. edit2 dan edit3 → didalam aplikasi konversi ada function edit1_callback dan function edit2_callback, kita tidak perlu menambahkan kode apapun dibawahnya, karena hanya berfungsi untuk menampilkan hasil dari perhitungan.

3. btn_hitung → didalam aplikasi konversi ada function btn_hitung_callbck, tambahkan program menjadi berikut:
   ```
   %kode ini diekskusi jika kita menekan tombol hitung
   C=handles.celcius;
   R=4/5*C;
   F=9/5*C + 32;
   %menampilkan hasil perhitungan
   set(handles.edit2, 'string', R);
   set(handles.edit3, 'string', F);
   ```
4. btn_exit → didalam aplikasi konversi ada
   ```
   delete(handles.figure1)
   ```

d. Menjalankan GUI

apabila kode program dalam aplikasi konversi suhu sudah kita masukkan, maka aplikasi tersebut sudah bias kita jalankan dengan mengklik tombol Run dari jendela figure. sehingga akan muncul tampilan berikut.



Gambar 5.6 Tampilan Hasil program

Soal Latihan :

1. Bila diketahui sebuah variable dengan nilai 1 sampai 100, tentukan bilangan yang habis dibagi 5, coba kerjakan dengan pemrograman Matlab.
2. Carilah konversi bilangan decimal ke heksa decimal. coba kerjakan dengan pemrograman Matlab.

BAB VI
MENAMPILKAN BILANGAN GANJIL MENGGUNAKAN MATLAB

A. Permasalahan

Bagaimana membuat sebuah flowchart dan program aplikasi yang digunakan untuk menampilkan sederetan bilangan ganjil dari angka 10 sampai angka 30 kecuali 11 dan 23!
Misal :
Keluaran : 13 15 17 19 21 25 27 29

B. Penyelesaian Masalah

Dari permasalahan yang ada di soal, maka untuk mencari keluaran bilangan ganjil antara nilai 10 sampai 30 user tidak memerlukan inputan. Dalam memgatasi soal diatas, maka dengan memasukkan batas nilai yang dijadikan dasar untuk membuat flowchart dan program aplikasi. Flowchart yang akan dibuat pertama-tama kondisi dari soal harus di berikan yaitu batas bawah dan batas atas dari soal yang dibuat. Begitu juga dengan kondisi tentang hanya bilangan ganjil saja yang akan ditampilkan, sehingga flowchart juga memfilter angka yang diperbolehkan juga harus angka ganjil. Supaya bilangan ganjil yang akan ditampilkan pada outputan, maka perlu operator Modulo yaitu sisa dari hasil bagi dengan bilangan 2. Bila hasil bagi dari soal menghasilkan sisa 0 berarti angka tersebut adalah angka genap dan kalau sisa dari hasil bagi sama dengan 1, maka angka tersebut adalah angka ganjil. Disamping bilangan ganjil, juga ada pengecekan tentang bilangan yang tidak boleh ditampilkan pada outputan yaitu bilangan 11 dan bilangan 23. Sehingga dalam pengecekan kondisi terdapat angka 11 dan 23, maka outputan tidak boleh menampilkan bilagan tersebut. Dengan demikian untuk mengatasi persoalan diatas, maka diperlukan proses perulangan.

C. Kebutuhan Data

Variable yang dibutuhkan dalam aplikasi diatas hanya satu yaitu variable bilangan.

D. Inisialisasi dan Deklarasi

Pertama tama nilai diinisialisasi dan dideklarisasikan dengan bilangan 10. Sehingga nilai 10 ini sebagai batas bawah dari aplikasi yang dibuat. Dari angka 10 ini, selanjutnya nilai akan bertambah satu demi satu untuk menambah nilai 10 menjadi 11 dan ditambah 1 terus. Pertambahan bilangan ini akan berhenti pada angka 30 yang menjadi batas atas dari soal yang diingingkan.

E. Input

Untuk aplikasi yang dibuat ini tidak memerlukan input sama sekali, baik dalam flowchart maupun program aplikasi yang akan dibuat.

F. Output

Output pada aplikasi ini yaitu nilai bilangan ganjil antara 10 sampai 30 keculai angka 11 dan 23. Maka bilangan yang akan ditampilkan yaitu bilangan 13, 15, 17, 19, 21, 25, 27, dan 29.

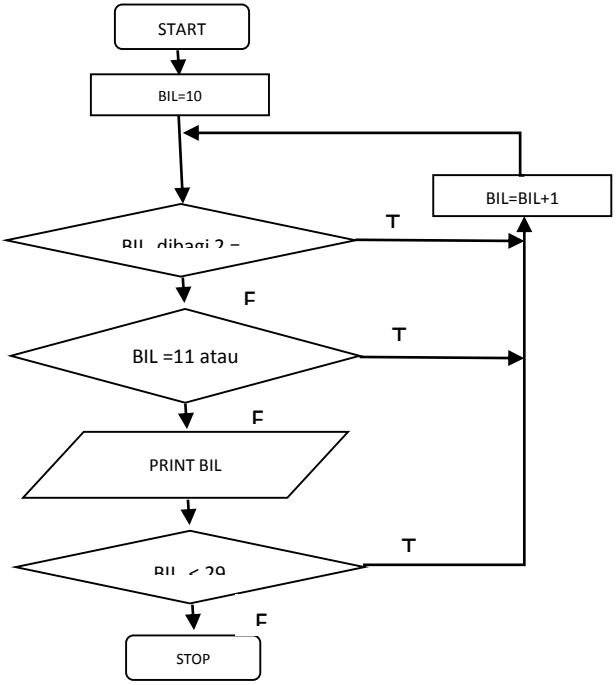
G. Proses Penyelesaian

Untuk menyelesaikan sola yang sudah dibuat, maka untuk mencari keluaran bilangan ganjil antara nilai 10 sampai 30 user tidak memerlukan inputan. Dalam memgatasi soal diatas, maka dengan memasukkan batas nilai yang dijadikan dasar untuk membuat flowchart dan program aplikasi. Flowchart yang akan dibuat pertama-tama kondisi dari soal harus di berikan yaitu batas bawah dan batas atas dari soal yang dibuat. Begitu juga dengan kondisi tentang hanya bilangan ganjil saja yang akan ditampilkan, sehingga flowchart juga memfilter angka yang diperbolehkan juga harus angka ganjil. Supaya bilangan ganjil yang akan ditampilkan pada outputan, maka perlu operator Modulo yaitu sisa dari hasil bagi dengan

bilangan 2. Bila hasil bagi dari soal menghasilkan sisa 0 berarti angka tersebut adalah angka genap dan kalau sisa dari hasil bagi sama dengan 1, maka angka tersebut adalah angka ganjil. Disamping bilangan ganjil, juga ada pengecekan tentang bilangan yang tidak boleh ditampilkan pada outputan yaitu bilangan 11 dan bilangan 23. Sehingga dalam pengecekan kondisi terdapat angka 11 dan 23, maka outputan tidak boleh menampilkan bilagan tersebut. Dengan demikian untuk mengatasi persoalan diatas, maka diperlukan proses perulangan. Sehingga output dari aplikasi yang dibuat adalah bilangan 13, 15, 17, 19, 21, 25, 27 dan 29.

.

H. Flowchart Keseluruhan



Gambar 6.1 Flowchart menampilkan bilanagn ganjil

I. Contoh program

Tampilkan bilangan ganjil dari bilanagn 10 samapai bilangan 30, kecuali bilangan 11 dan 23 !

Penyelesaian :

7. Buka Program Aplikasi Matlab
8. pilih **File—>New—>M-File**
9. Ktik listing program di bawah ini :

```
clc;
disp('program untuk menampilkan bilangan ganjil dari 10
sampai 30 kecuali 11 dan 23');
n=10;
for i=1:20
   nilai=mod(n,2);
if nilai==0
 else
  if n==11 | n==23
       else
            disp(['Bilangan ganjil= ' num2str(n)])
   end
end
      n=n+1;
end
```

Setelah selesai mengetikkan listing di atas simpan dan jalankan dengan memilih **Tools—>Run**, dan hasil yang akan di dapat adalah sebagai berikut :

```
4 -    n=10;
5 -    nilai=mod(n,2);
6 -    for i=1:20
7 -        nilai=mod(n,2);
8 -    if nilai==0
9
10 -   else
11 -      if n==11 | n==23
12
13 -      else
14 -                  disp(['Bilangan ganjil= ' num2str(n)])
15 -       end
16 -   end
```

```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

program untuk menampilkan bilangan ganjil dari 10 sampai 30 kecuali 11 dan 23
Bilangan ganjil= 13
Bilangan ganjil= 15
Bilangan ganjil= 17
Bilangan ganjil= 19
Bilangan ganjil= 21
Bilangan ganjil= 25
Bilangan ganjil= 27
Bilangan ganjil= 29
```

Gambar 6.2 Hasil Program Menampilkan bilangan Ganjil

Penjelasan untuk listing program di atas :

clc;→ digunakan untuk menghapus layar yang ada di command window.

disp('program untuk menampilkan bilangan ganjil dari 10 sampai 30 kecuali 11 dan 23'); → digunakan untuk menampilkan tulisan program untuk menampilkan bilangan ganjil dari 10 sampai 30 kecuali 11 dan 23'.

n=10; → digunakan untuk member nilai n = 10.

for i=1:20 → looping I dari 1 sampai 20

nilai=mod(n,2);→ sisa hasil bagi nilai n dengan 2

if nilai==0 → apakah variable nilai =0, kalau ya maka tidak ada statement.

Else → kalau tidak, maka ada peratnyaan lagi apakah n=11 dan n=23

if n==11 | n==23 → jika ya maka tidak ada statement.

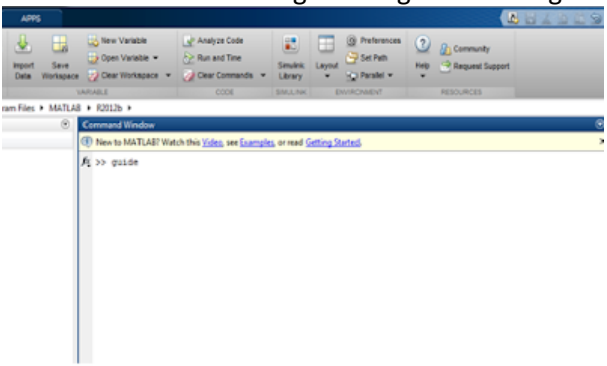
Else → jika tidak, maka akan menampilakn bilangan ganjil

disp(['Bilangan ganjil= ' num2str(n)])

```
   end
end
      n=n+1;  → nilai n ditambah dengan 1
end
```

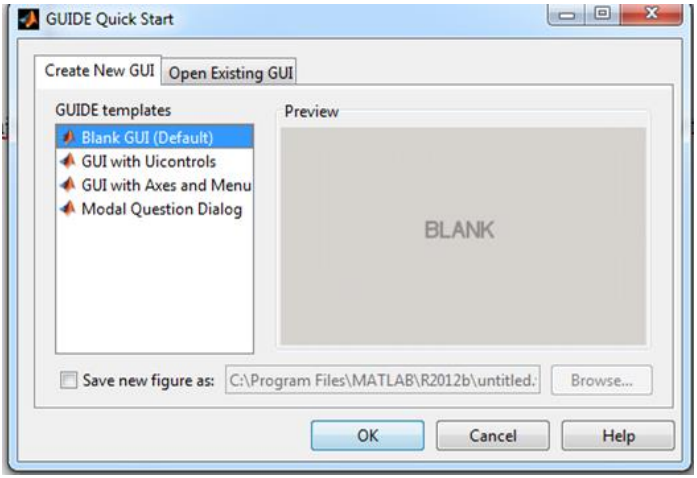
Contoh Program menggunkakan Fasilitas GUI

Melalui command matlab dengan mengetikkan: >> guide



Gambar 6.3 Tamplian Matlab dengan tulisan guide

Selanjutnya akan muncul tampilan kotak dialog pilihan GUIDE Quick Start.



Gambar 6.4 Tamplian Matlab dengan aplikasi guide

Membuat Contoh Aplikasi GUIDE MATLAB

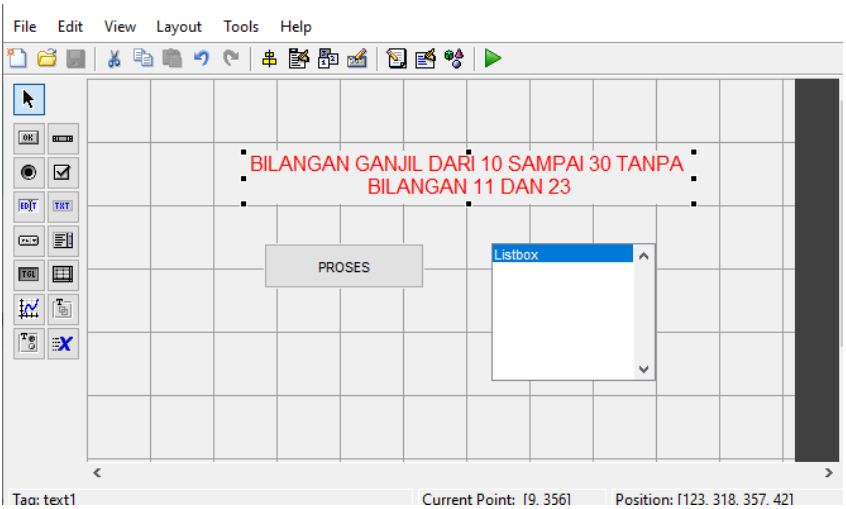
1. Menampilkan bilangan ganjil dari 10 sampai 30 kecuali angka 11 dan 23.

Contoh aplikasi yang dibuat adalah menampilakn bilangan ganjil dari 10 sampai 30 kecuali bilangan 11 dan 23. Langkah-langkah yang harus kita kerjakan adalah:

a. Mendesain Figure

Dalam mendesain Tampilan gambar, kita harus dapat membayangkan komponen apa saja yang perlu kita tampilkan. Seperti dalam membuat aplikasi menampilkan bilangan 10 sampai 30, variabel input yang dibutuhkan adalah tidak ada, kemudian variabel outputnya adalah bilangan ganjil dari 10 sampai 30 kecuali bilangan 11 dan 23. Maka kita memerlukan 1 Buah static text yang digunakan menampilkan tulisan. Kemudian kita juga memerlukan 1 tombol pushbutton/togglebutton untuk mulai melakukan proses dan 1 buah listbox untuk menampilkan bilangan putput yang kita inginkan. Kita juga bisa menambahkan komponen lain untuk memperjelas dan mempercantik desian figure yang akan kita buat.

Desainlah figure seperti pada Gambar 6.5. Gunakan 1 buah static text, 1 buah listbox, dan 1 buah pushbutton. Dalam meletakkan komponen pallete boleh tidak sesuai dengan gambar.



Gambar 6.5 Desain Tampilan

b. Mengatur Layout Komponen

Setelah kita selesai mendesain figure, aturlah masing-masing komponen menggunakan property inspector.

Tabel 5.1 Cara Mengatur komponen Properti

KOMPONEN	PROPERTY INSPECTOR			
	FONT SIZE	FONT WEIGHT	STRING	TAG
Static text 1	14	Bold	Bilangan ganjil antara 10 sampai 30 tanpa bilangan 11 dan 23	Text 1
Listbox1	10	Normal	Kosongkan	
Pushbutton 1	10	Bold	Proses	Button_Proses

C. Save Tampilan

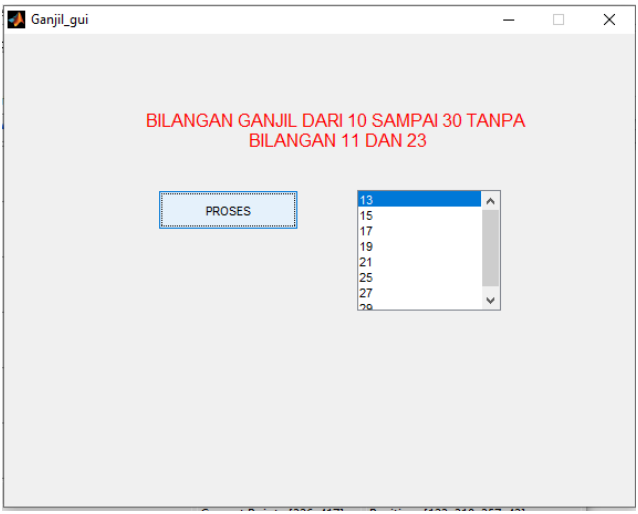
Setelah selesai mendesain figure, langkah selanjutnya adalah menyimpan Tampilan gambar, beri nama ganjil.fig, secara otomatis kita akan dibuatkan kerangka m-file dengan nama yang sama, yaitu ganjil.m.

Dari beberapa fungsi yang muncul di m-file. Kita cukup memperhatikan fungsi yang memiliki callback, yaitu :

```
function pushbutton1_Callback(hObject, eventdata, handles)
n=10;
k=0;
for i=1:20
      nilai=mod(n,2);
if nilai==0
  else
   if n==11 | n==23
      else
      hasil = num2str(n)
      k=k+1;
          listboxItems{k} = hasil;
          set(handles.listbox1, 'String', listboxItems);
           end
end
      n=n+1;
end
```

d. Menjalankan Program Ganjil di GUI

Setelah langkah-langkah diatas dijalankan, langkah terakhir adalah menjalankan aplikasi yang telah dibuat dengan mengklik tombol Run dari jendela figure atau dari jendela debug m-file (tekan F5), sehingga akan muncul tampilan berikut.



Gambar 6.6 Tampilan Hasil program

Soal Latihan :

1. Carilah bilangan genap dari 1 sampai 1000, coba kerjakan dengan menggunkan GUI pada matlab.
2. Carilah bilangan prima dari 1 sampai 1000, coba kerjakan dengan menggunkan GUI pada matlab.

BAB VII
PERCABANGAN DAN PERULANGAN MENGGUNAKAN MATLAB

Pada pemrograman Matlab semua Instruksi yang dikandung umumnya hamper sama dengan bahasa pemrograman lainnya.

Conditional IF

pernyataan else-if dapat digunakan untuk membandingkan dua angka dan memberikan opsi tergantung pada apakah kondisi tertentu terpenuhi atau tidak. Bentuk umumnya adalah:

if {condition #1}
{statement #1}
elseif {condition #2}
{statement #2}
elseif {conditions #3}
{statement #3}
else {final statement}
end

Contoh:

```
1       % menggunakan conditional if-elseif-else
2
3 -     clear; clc;
4
5       % memberi masukan
6 -     a = input('Masukkan Angka = ');
7
8       % conditional IF
9 -     if rem(a,2) ~= 0
10 -        fprintf('%d adalah bilangan Ganjil\n',a)
11 -    else
12 -        fprintf('%d adalah bilangan Genap\n',a)
13 -    end
```

Hasil dari program diatas :

```
Masukkan Angka = 167453
167453 adalah bilangan Ganjil
>>
```

Looping

Fungsi Looping atau iteratif sangat berguna dalam pemecahan masalah teknik. Contoh yang bagus adalah dalam memecahkan integral secara numerik. Dalam melakukan integrasi numerik, kami akan mencoba memperkirakan solusi analitik (solusi yang memecahkan masalah dengan menggunakan batas hingga nol atau tak terhingga) menggunakan pendekatan numerik yang akan memiliki langkah-langkah yang terbatas, meskipun besar, dalam jumlah banyak. Memecahkan integral dengan tangan akan mencakup ribuan atau jutaan perhitungan. Kita dapat menggunakan loop dalam suatu fungsi sehingga MATLAB akan menyelesaikan jutaan perhitungan secara otomatis dengan satu perintah. Pertama kita harus terbiasa menggunakan loop ini. Instruksi yang bias digunakan adalah sebagai berikut :

For A = c:inkremen/dekremen:b

End

Statemen

Supaya lebih memahami proses dari penggunaan For, ada contoh berikut ini :

```
1   % menggunakan loop for
2
3   clear; clc;
4
5   % kuadrat bilangan
6   n = input('Banyak bilangan = ');
7
8   disp('Hasil Kuadrat');
9   disp('==============');
10
11  for a=1:n
12      hasil = a*a;
13      fprintf('%d x %d \t = %d\n',a,a,hasil)
14  end
```

Hasil dari program diatas adalah sebagai berikut :

```
Banyak bilangan = 9
Hasil Kuadrat
=============
1 x 1    = 1
2 x 2    = 4
3 x 3    = 9
4 x 4    = 16
5 x 5    = 25
6 x 6    = 36
7 x 7    = 49
8 x 8    = 64
9 x 9    = 81
>>
```

A. Permasalahan

Bagaimana membuat kalkulator sederhana, sehingga dapat melakukan operasi penjumlahan (+), pengurangan (-), Perkalian (*) dan Pembagian (/).

Misal :

Bilangan 1 → Masukkan bilangan 1= 25
Bilangan 2 → Masukkan bilangan 2 = 5
Operasi → Masukkan operator = /
Hasil Output : Hasilnya dari 25 / 5 adalah 5

B. Cara Penyelesaian Masalah

Untuk membuat aplikasi kalkulator sederhana, maka ada dua buah inputan yaitu inputan 1 dan inputan 2. Kemudian ada sebuah operator untuk memproses kedua inputan, proses tersebut bisa penjumlahan, pengurangan, perkalian atau pembagian. Jika kedua inputan tadi diproses dengan penjumlahan, maka inputan 1 akan ditambahkan dengan inputan 2 begitu juga dengan operator yang lain.

C. Input:

Masukan pada aplikasi kalkulator sederhana ini yaitu :

- Inputan 1 → Input bilangan pertama berupa angka (integer)
- Inputan 2 → Input bilangan kedua berupa angka (integer)

D. Output:

Keluaran pada aplikasi kalkulator sederhana ini yaitu :
- Hasil perhitungan dari kedua input bilangan.

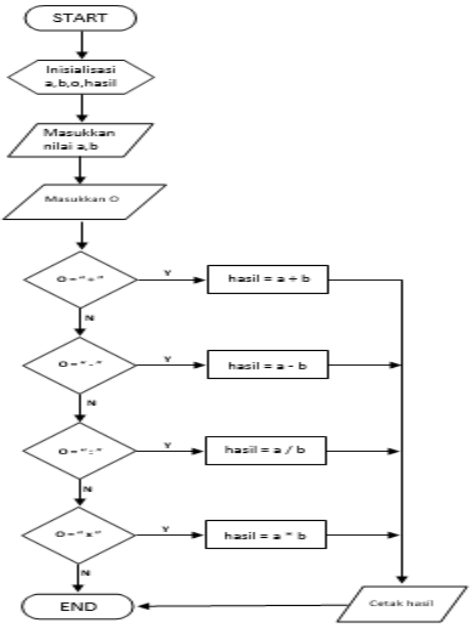
E. Data yang Dibutuhkan

Inputan 1 → bilangan 1 yang berupa Variabel bertipe integer
Inputan 2 → bilangan 2 berupa Variabel bertipe integer
Operator → operator berupa Variabel bertipe char
Output → hasil yang berupa Variabel bertipe float

F. Logika Pemrograman:

- Untuk Operasi percabangan bias menggunakan **'IF ... THEN ... ELSE ...'**
- Untuk operator penjumlahan → bisa menggunakan operator '+'
- Untuk operator penguranagn → bisa menggunakan operator '-'
- Untuk operator perkalian → bisa menggunakan operator '*'
- Untuk operator pembagian → bisa menggunakan operator '/'

G. Flowchart Keseluruhan:



Gambar 7.1 Flowchart Kalkulator Sederhana

H. Contoh program

Buatlah Kalkulator sederhana dengan operator +, -, x dan /

Penyelesaian :

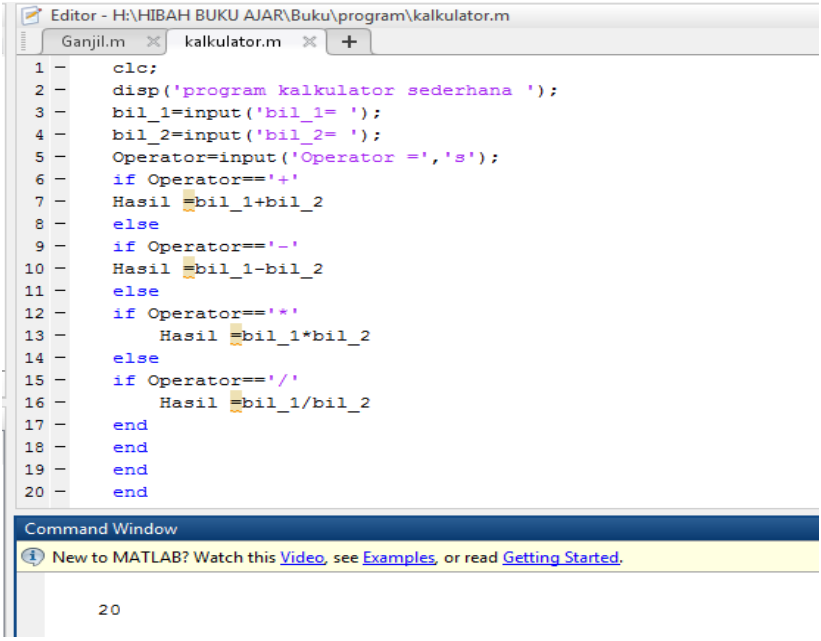
Buka Program Aplikasi Matlab

pilih **File—>New—>M-File**

Ktik listing program di bawah ini :

```
clc;
disp('program kalkulator sederhana ');
bil_1=input('bil_1= ');
bil_2=input('bil_2= ');
Operator=input('Operator =','s');
if Operator=='+'
Hasil =bil_1+bil_2
else
if Operator=='-'
Hasil =bil_1-bil_2
else
if Operator=='*'
    Hasil =bil_1*bil_2
else
if Operator=='/'
    Hasil =bil_1/bil_2
end
end
end
end
```

Setelah selesai mengetikkan listing di atas simpan dan jalankan dengan memilih **Tools—>Run**, dan hasil yang akan di dapat adalah sebagai berikut :



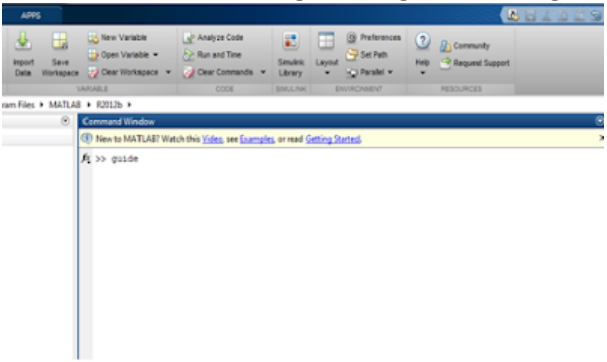
Gambar 7.2 Hasil Program kalkulator sederhana

Penjelasan untuk listing program di atas :

1. clc;→ digunakan untuk menghapus layar yang ada di command window.
2. disp('program kalkulator sederhana '); → digunakan untuk menampilkan tulisan program kalkulator sederhana.
3. bil_1=input('bil_1= ');→ masukan untuk bil 1
4. bil_2=input('bil_2= '); → masukan untuk bil 2
5. Operator=input('Operator =','s'); → masukan untuk operator +, -, * atau /
6. if Operator=='+' → pernyataan apakah operatornya +, kalau ya, maka akan mengerjakan dibawahnya, jika tidak maka ke if Operator=='-'

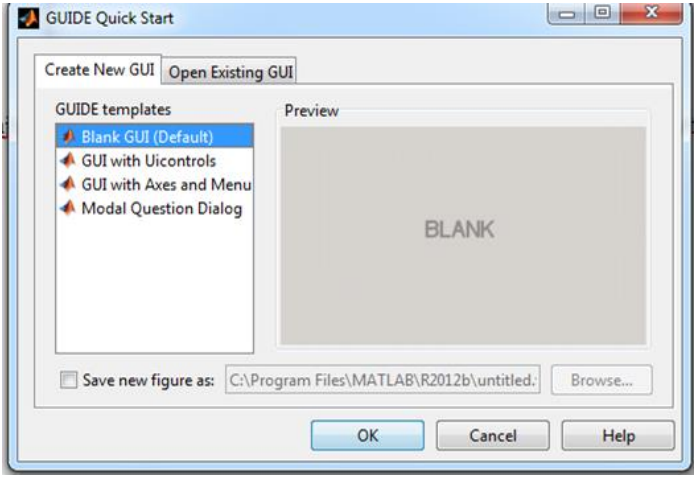
7. dan seterusnya

Contoh Program menggunkakan Fasilitas GUI
Melalui command matlab dengan mengetikkan: >> guide



Gambar 7.3 Tamplian Matlab dengan tulisan guide

Selanjutnya akan muncul tampilan kotak dialog pilihan GUIDE Quick Start.



Gambar 7.4 Tamplian Matlab dengan aplikasi guide

Membuat Contoh Aplikasi GUIDE MATLAB

1. Membuat kalkulator sederhana**.**

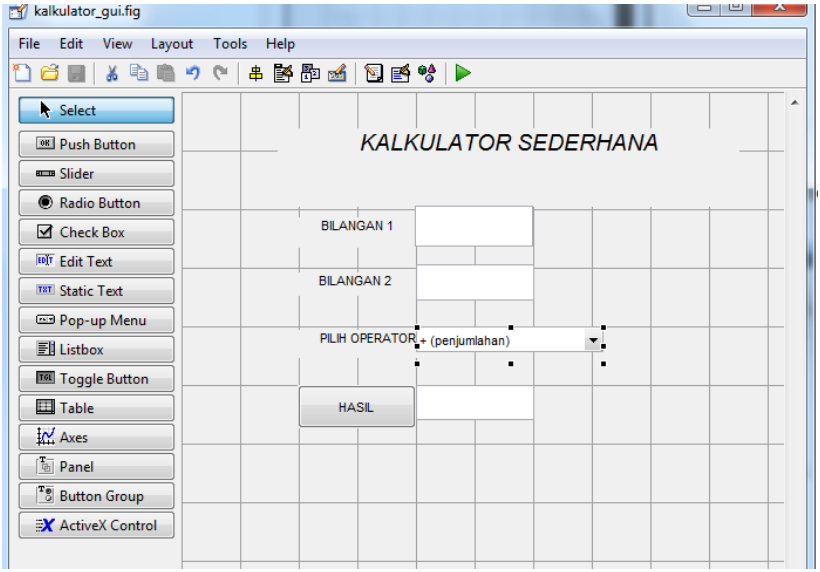
Berikut ini adalah proses yang dikerjakan untuk memulai membuat sebuah program :

a. Mengatur **Tampilan Gambar**

Untuk membuat sebuah tampilan, maka komponen apa saja yang kita butuhkan untuk membuat aplikasi kalkulator sederhana. Variabel input yang dibutuhkan adalah inputan 1 dan inputan 2, kemudian variabel outputnya adalah bilangan hasil dari proses penjumlahan atau penguranagn atau perkalian atau pembagian. Untuk membuat aplikasi kalkulator sederhana komponen yang dibutuhkan adalah :

1. Tiga buah edit text → digunakan untuk inputan 1, inputan 2 dan keluaran atau hasil.
2. satu tombol pushbutton/togglebutton → digunakan untuk mulai melakukan proses perhitungan.
3. Satu buah listbox → digunakan untuk operator.

Desain tampilan yang terlihat pada Gambar 7.5



Gambar 7.5 Desain Tampilan

b. Layout Komponen

Setelah kita selesai mendesain figure, aturlah masing-masing komponen menggunakan property inspector.

Tabel 7.1 Cara Mengatur komponen Properti

No	Nama	Property	Nilai
1	Edit Text	FontSize	12
		String	<kosongkan>
		Tag	edit1
2	Edit Text	FontSize	12
		String	<kosongkan>
		Tag	edit2

3	Edit Text	FontSize	12
		String	<kosongkan>
		Tag	edit3
4	Pushbutton	FontSize	12
		String	Hasil
		Tag	pushbutton1
5	Popupmenu	FontSize	12
		String	+ (penjumlahan)– (pengurangan) x (perkalian) : (pembagian)
		Tag	popupmenu1
6	Static Text	FontSize	12
		String	Masukkan angka pertama
		Tag	text2
7	Static Text	FontSize	12
		String	Masukkan angka kedua
		Tag	text3
8	Static Text	FontSize	12
		String	Pilih Operator
		Tag	text4
9	Static Text	FontSize	14
		String	Program Kalkulator Sederhana
		Tag	text1

C. Save Tampilan

Untuk menyimpan aplikasi yang sudah dibuat, proses selanjutnya yaitu menyimpan aplikasi tersebut dengan nama kalkulator_sederhana.m.

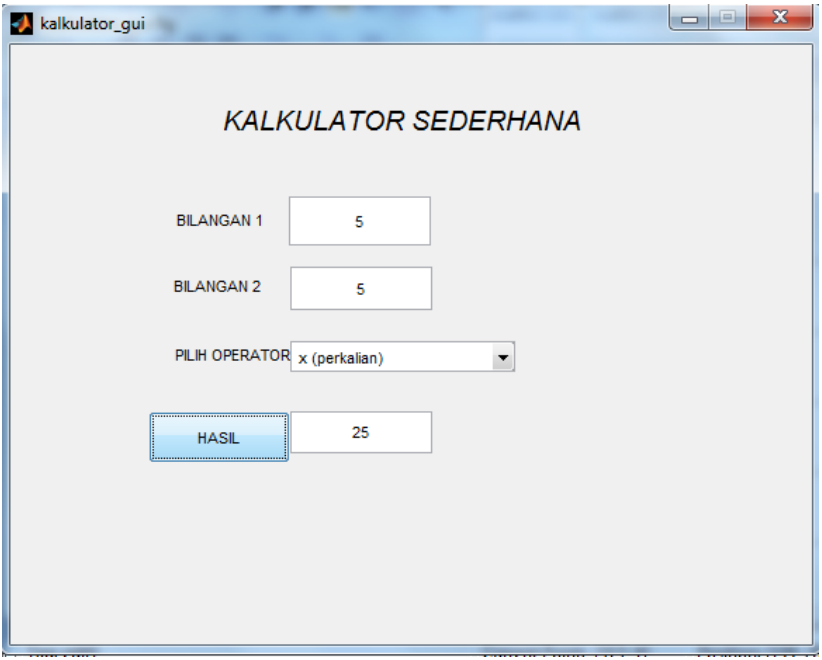
Fungsi yang digunakan untuk membuat aplikasi kalkulator sederhana ini, diantaranya adalah :

1. btn_hasil_Callback.

```
bil_pertama = str2double(get(handles.edit1,'String'));
bil_kedua = str2double(get(handles.edit2,'String'));
OP = get(handles.popupmenu1,'Value');
if OP == 1
   hasil = bil_pertama + bil_kedua;
elseif OP == 2
   hasil = bil_pertama - bil_kedua;
elseif OP == 3
   hasil = bil_pertama * bil_kedua;
elseif OP == 4
   hasil = bil_pertama / bil_kedua;
end
set(handles.edit3,'String',num2str(hasil))
```

d. Menjalankan GUI

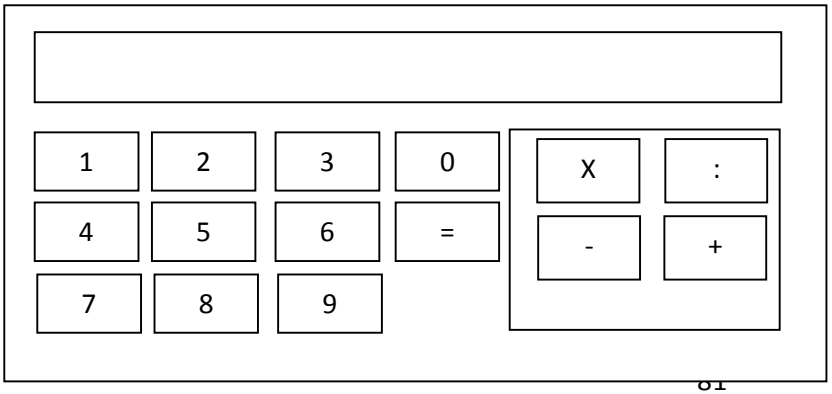
Untuk menjalankan aplikasi yang sudah kita buat, maka kita klik tombol RUN atau menekan tombol F5.



Gambar 7.6 Tampilan Hasil program

Soal Latihan :

Buatlah Kalkulator dengan menggunkan tombol seperti gambar di bawah in.



BAB VIII
TUMPUKAN (STACK) MENGGUNAKAN MATLAB

Menyusun data dari banyak variabel menjadi variabel tunggal

Syntax

S = stack(U,vars)

S = stack(U,vars,Name,Value)

[S,iu] = stack(___)

Penjelasan :

S = stack (U, vars) mengubah tabel atau jadwal, U, menjadi tabel atau jadwal yang setara, S, yang ditumpuk. Fungsi stack menumpuk banyak variabel dari U, yang ditentukan oleh vars, menjadi variabel tunggal di S. Secara umum, S berisi lebih sedikit variabel, tetapi lebih banyak baris, daripada U.

Argumen keluaran, S, berisi variabel kategorikal baru untuk menunjukkan dari mana variabel dalam U data yang ditumpuk di setiap baris berasal. stack mereplikasi data dari variabel dalam U yang tidak ditumpuk.

Jika U adalah sebuah tabel, maka Anda tidak dapat menumpuk nama baris.

Jika U adalah jadwal, maka Anda tidak dapat menumpuk waktu baris.

S = stack (U, vars, Name, Value) mengonversi tabel, U, dengan opsi tambahan yang ditentukan oleh satu Nama atau lebih, argumen pasangan nilai.

Misalnya, Anda bisa menentukan nama variabel untuk variabel baru dan yang ditumpuk di U.

[S, iu] = stack (___) juga mengembalikan vektor indeks, iu, yang menunjukkan korespondensi antara baris dalam S dan baris di U. Anda dapat menggunakan argumen input sebelumnya.

Contoh :
Buat tabel yang berisi skor tes dari tiga tes terpisah. Tabel ini dalam format unstacked.

```
angka1 = [13;27;87;85];
angka2 = [19;37;92;87];
angka3 = [15;42;89;99];

U = table(angka1,angka2,angka3)
```

Hasil :

U=*4×3 table*

angka1	angka2	angka3
13	19	15
27	37	42
87	92	89
85	87	99

Tabel ini berisi empat baris dan tiga variabel. stack skor tes menjadi satu variabel.

```
S = stack(U,1:3)
```

Hasil :

S=12×2 table

Angka1_angkat2_angka3_Indikator angka1_angka2_angka3	
Angka1	93
Angka2	89
Angka3	95
Angka1	57
Angka2	77

Angka3	62
Angka1	87
Angka2	92
Angka3	89
Angka1	85
Angka2	87
Angka3	99

S berisi dua belas baris dan dua variabel. S dalam format bertumpuk.

Variabel kategorikal, Test1_Test2_Test3_Indicator, mengidentifikasi tes mana yang sesuai dengan skor dalam variabel data yang ditumpuk, Test1_Test2_Test3.

Variable stack dan menentukan Nama Variabel

Buat jadwal yang menunjukkan jumlah salju di tiga kota dari lima badai yang berbeda. Tentukan tanggal badai sebagai nilai datetime dan gunakan mereka sebagai waktu baris dari jadwal U. Tentukan array nomor badai, Badai, sebagai array kategorikal karena ada satu set tetap nomor badai dalam jadwal ini.

```
St = categorical([1;2;3;4;5]);
Tgl = datetime({'2015-12-15';'2016-01-12';'2017-11-
23';'2018-02-17';'2019-02-15'});
Na = [10;5;13;0;47];
Bo = [28;9;21;5;52];
Wor = [36;10;16;3;65];

U = timetable(Tgl,St,Na,Bo,Wor)
```

Hasil :

U=5×4 timetable

Date	St	Na	Bo	Wor

15-Dec-2015	1	20	18	26
12-Jan-2016	2	5	9	10
13-Jan-2017	3	13	21	16
17-Feb-2018	4	0	5	3
15-Feb-2019	5	17	12	15

Variabel St dan Tgl berisi data yang konstan di setiap lokasi. Variabel stack Na, Bo, dan Wor menjadi satu variabel. Beri nama variabel yang berisi data yang ditumpuk, Snowf, dan beri nama variabel indikator baru, Tow.

```
S = stack(U,{'Na','Bo','Wor'},...
        'NewDataVariableName','Snowfall',...
        'IndexVariableName','Tow')
```

Hasil :

S=15×3 timetable

Tgl	St	Tow	Snowf
15-Dec-2015	1	Na	20
15-Dec-2015	1	Bo	18
15-Dec-2015	1	Wor	26
12-Jan-2016	2	Na	5
12-Jan-2016	2	Bo	9
12-Jan-2016	2	Wor	10
13-Jan-2017	3	Na	13
13-Jan-2017	3	Bo	21
13-Jan-2017	3	Wor	16
17-Feb-2018	4	Na	0
17-Feb-2018	4	Bo	5
17-Feb-2018	4	Wor	3

15-Feb-2019	5	Na	17
15-Feb-2019	5	Bo	12
15-Feb-2019	5	Wor	15

A. Permasalahan

Bagaimana cara untuk membuat flowchart dan aplikasi program untuk membalik kalimat.

Misal :

Masukan : Input kalimat = “MARI BELAJAR MATLAB”
Keluaran : “BALTAM RAJAEB IRAM”

B. Penyelesaian Masalah

Untuk membuat program aplikasi dengan membalik kalimat dapat dilakukan dengan :

1. Penggunaan array.
2. Penggunaan stack.
   Stack →Pengolahan data yaitu data yang terakir dimasukkan akan diambil duluan seperti Last In First Out (LIFO).

C. Masukan

Data Input yang dibutuhkan dalam aplikasi adalah :

- beruapa data Kalimat dalam bentuk string

D. Keluaran

Apliaklsi Program membalik kalimat akan menghasilkan output :

- Kalimat yang sudah dibalik dalam bentuk string

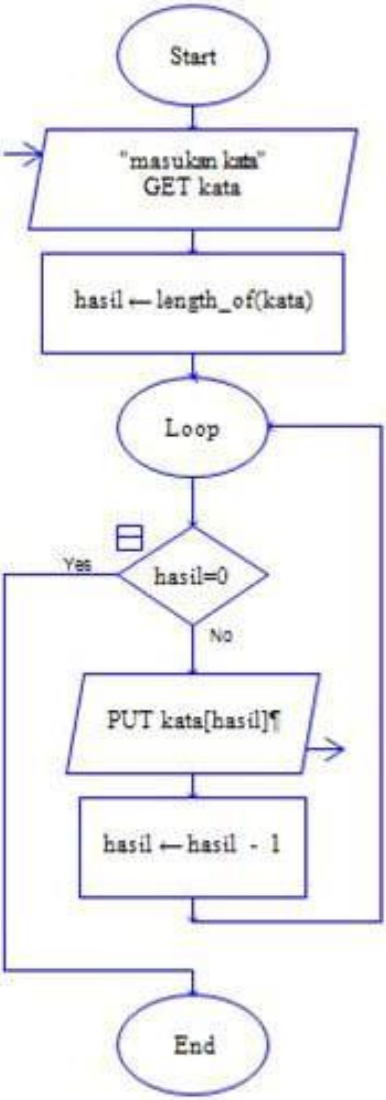
E. Kebutuhan Data

- sebuah kalimat yang mempunyai Variabel bertipe string (array of character).

variabel ***kalimat***

F. Flowchart

Penyelesaian menggunakan Flowchart dengan array


Start
"masukan kata"
GET kata
hasil ← length_of(kata)
Loop
Yes
hasil=0
No
PUT kata[hasil]¶
hasil ← hasil - 1
End

Gambar 1. Flowchart membalik kalimat

G. Pemrograman

```
>> a = 'mari belajar matlab'
>>b = length(a)
>> d = a(b:-1:1)
```

Soal Latihan :

Buatlah sebuah flowchart dan program untuk membalik sebuah kalimat " Mari belajar bahasa pemrograman dengan menggunakan Matlab".

BAB IX
KONVERSI BILANGAN MENGGUNAKAN MATLAB

A. Permasalahan

Bagaimana membuat flowchart dan Program aplikasi konversi bilangan biner ke bilanagn decimal.
Misal :
Inputan biner : 10110111
Output desimal : 55

B. Penyelesaian Masalah

Untuk mengkonversikan bilangan biner menjadi bilangan decimal, maka bilangan biner yang akan dikonversikan tersebut dikalikan dengan bilanagn 2 yang dipangkatkan :
$b1 * 2^{i-1} + b2 * 2^{i-2} + ... + bi-2 * 2^2 + bi-1 * 2^1 + bi * 2^0$
untuk soal diatas, maka bisa diselesaikan menurut aturan diatas :
Inputan biner : 00110111
$1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$
= 0*128 + 0*64 + 1*32 + 1*16 + 0*8 + 1*4 + 0*2 + 1*1
= 128 + 0 + 32 + 16 + 0 + 4 + 0 + 1
= 181

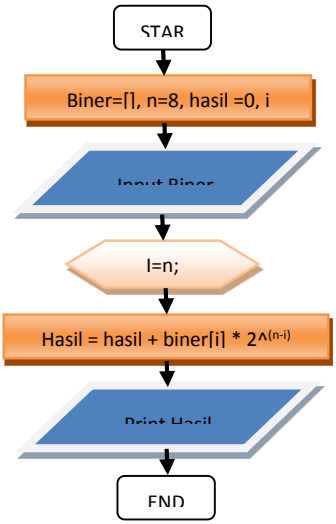
C. Data Yang Dibutuhkan

Banyaknya data elemen yang digunakan pada contoh diatas sebanyak 8 data elemen (i=8).

D. Penyelesaian

Untuk menghitung konversi bilangan biner, maka dilakukan perhitungan *bi* * 2^0 dulu, yaitu mulai dari kanan kita (kalau kita menulis di buku). Dari perkalian 2^0 kemudian perkalian 2^1 dan seterusnya. Sehingga diperlukan proses looping yang berlawanan arah untuk proses perkaliannya, kemudian hasilnya diakumulasikan pada hasil secara keseluruhan.

E. Flowchart



Gambar 1. Flowchart Konversi biner ke decimal

F. Program

```
clear all;
close all;
clc;
biner=input('Nilai Biner = ','s');
biner2=fliplr(num2str(biner));
desm=0;
```

```
for m=1:length(biner2)
temp=str2double(biner2(m));
desm=desm+temp*(2^(m-1));
end
disp('Biner');
disp(biner);
disp('Desimal');
disp(desm);
```

G. Permasalahan konversi bilangan desimal ke bilangan biner

Bagaimana membuat flowchart dan program aplikasi untuk konversi bilangan desimal ke bilangan biner.
Misal :
Inputan untuk bilangan desimal : 55
Output hasil Bilangan biner : 110111

H. Penyelesaian Masalah

Untuk mengkonversi dari bilangan desimal ke bilangan biner, maka diperlukan langkah-langkah sebagi berikut :

1. Bagilah angka dengan 2
2. Dapatkan hasil bagi integer untuk iterasi berikutnya.
3. Dapatkan sisanya untuk digit biner.
4. Ulangi langkah-langkah sampai hasil bagi sama dengan 0.

Contoh 1
Konversi 1310 ke biner :

Dibagi dengan 2	Hasil Bagi	Sisa	Bit #
13/2	6	1	0

6/2	3	0	1
3/2	1	1	2
1/2	0	1	3

Sehingga $13_{10} = 1101_2$

Contoh 2 :

Konversi 174_{10} ke Biner :

Dibagi dengan 2	Hasil Bagi	Sisa	Bit #
174/2	87	0	0
87/2	43	1	1
43/2	21	1	2
21/2	10	1	3
10/2	5	0	4
5/2	2	1	5
2/2	1	0	6
1/2	0	1	7

Sehingga $174_{10} = 10101110_2$

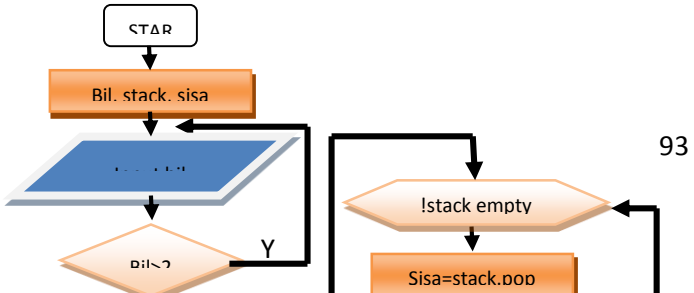
I. Data Yang Dibutuhkan

Input bilangan → variabel dari nilai bilangan

Outputan sisa → variabel dari nilai sisa bagi dengan 2

Stack → untuk penampung sisa bagi.

J. Flowchart



Gambar 2. Flowchart Konversi decimal ke biner

K. Program

```
clear all;
close all;
clc;
% Konversi nilai desimal ke biner
desimal=input('Nilai desimal = ','s');
bit=dec2bin(desimal);
% Mengkondisikan jumlah anggota vektor agar sesuai
ttemp=[zeros(1,3-rem(length(bit),3)) bit];
temp=fliplr(ttemp);
for m=1:3:length(temp)
k=0;
tempp=0;
for n=m:m+2
tempp=tempp+temp(n)*(2^k);
k=k+1;
end
```

```
end
disp('Desimal');
disp(desimal);
disp('Biner');
disp(ttemp);
```

Soal Latihan :

1. Buatlah sebuah flowchart dan Program untuk mengkonversi bilangan decimal ke bilangan Oktal.
2. Buatlah sebuah flowchart dan Program untuk mengkonversi bilangan Oktal ke bilangan Desimal.
3. Buatlah sebuah flowchart dan Program untuk mengkonversi bilangan decimal ke bilangan Heksa desimal.
4. Buatlah sebuah flowchart dan Program untuk mengkonversi bilangan Heksa decimal ke bilangan Desimal.

BAB X
OPERASI MATRIKS MENGGUNAKAN MATLAB

A. Notasi Matrix

Matriks r × c adalah susunan elemen persegi panjang dengan baris r dan kolom c. Matriks r × c dikatakan berorde r × c. Matriks biasanya dilambangkan dengan huruf kapital yang dicetak dengan font tebal (misalnya, A, B, X). Elemen matriks diwakili oleh huruf kecil dengan subskrip ganda (misalnya,,, bjk, xjk). Misalnya, dalam matriks X, $x13$ adalah elemen di baris pertama dan kolom ketiga. A 4 × 3 matriks X ditunjukkan di bawah ini.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \end{bmatrix}$$

Matriks dengan satu baris disebut vektor baris dan matriks dengan satu kolom disebut vektor kolom. Vektor biasanya diwakili oleh huruf kecil yang dicetak dengan font tebal (misalnya, a, b, x). Elemen vektor diwakili oleh huruf kecil dengan satu subskrip (misalnya, aj, bj, xj). Vektor kolom 3 × 1 y dan vektor baris 1 × 4 h ditampilkan di bawah ini.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\mathbf{h} = [h_1 \;\; h_2 \;\; h_3 \;\; h_4].$$

Misalnya, penambahan dua matriks akan direpresentasikan dengan X = Y + W.

>> X = Y + W;

Baris perintah titik koma di akhir baris merupakan perintah yang digunakan untuk menghilangkan output pada layar monitor. Bila dalam perintah tidak menggunakan tanda titik koma pada akhir perintah, maka akan menghasilkan konten operasi. Pada baris sebelumnya, ini berarti bahwa konten A akan ditampilkan di layar jika tanda koma dihilangkan. Kami juga menggunakan notasi monospace untuk nama-nama fungsi MATLAB, seperti membaca, menulis, plot, dan sebagainya. Ketika sebuah matriks secara spesifik merujuk pada suatu gambar, kita akan menggunakan simbol-simbol seperti f dan g, dalam upaya untuk menjadi setepat mungkin dengan notasi yang digunakan dalam buku Digital Image Processing.

Simbol matriks, A, dipahami berarti :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix}.$$

Dengan kata lain, matriks adalah larik simbol, terdiri dari baris M dan kolom N. Ukuran array seperti itu dengan notasi M x N. Jika M = 1, kami menyebut matriks 1 x N sebagai vektor baris. Jika N = 1, kita menyebut matriks M x 1 sebagai vektor kolom. Jika M dan N = 1, maka A dinyatakan skalar. Apabila ada Notasi B (k;l) berarti elemen baris k dan kolom i dari B. Misalnya, B (5; 3) adalah elemen yang terletak di baris kelima dan kolom ketiga dari array. Perhatikan bahwa elemen pertama dari tuple (i; j) mengacu pada deretan array dan yang kedua ke kolom.

Contoh :

A = [1 2 3 ; 4 5 6 ; 7 8 9]

Maka hasil pada Matlab adalah :

A =

1 2 3

4 5 6

7 8 9

B = [1 2 3 , 4 5 6 , 7 8 9]

Maka hasil pada Matlab adalah :

B =

1 2 3 4 5 6 7 8 9

B. Matrix Identitas

Matriks dapat direpresentasikan dengan menulis baris individual, dipisahkan oleh titik koma (penggunaan titik koma

ini berbeda dengan penggunaan karakter yang sama di akhir baris).
Misalnya :
>> B = [456;abc;123]
menciptakan matriks 3 £ 3 di MATLAB di mana baris pertama memiliki elemen 4, 5, 6, baris kedua memiliki elemen a, b, c, dan seterusnya (perhatikan penggunaan tanda kurung []). Sebaliknya, jika kita menulis baris yang sama tanpa titik koma, kita akan mendapatkannya
>> B = [456;abc;123]
B =
4 5 6
abc
123
operator titik dua (:) merupakan salah satu operator indeks MATLAB yang paling kuat.
Misalnya :
>> 5:10
menghasilkan deretan bilangan bulat
5 6 7 8 9 10
>> 1:25
ans =
 Columns 1 through 15
   1   2   3   4   5   6   7   8   9 10 11 12 13 14
15
 Columns 16 through 25
  16 17 18 19 20 21 22 23 24 25
>> 10:20
ans =
  10 11 12 13 14 15 16 17 18 19 20

C. Penggabungan Matrix

Matriks gabungan adalah proses bergabung dengan matriks kecil untuk membuat matriks yang lebih besar. Operator gabungan adalah sepasang tanda kurung siku, [].

```
>> A = [456;123;246]
A =
  456
  123
  246
```

Penggabungkan tiga baris (dipisahkan oleh titik koma) untuk membuat matriks 3 x 3. Perhatikan bahwa elemen individual dipisahkan oleh spasi. Memisahkan elemen dengan koma akan menghasilkan hasil yang sama. Perhatikan juga urutan unsur-unsurnya. Kelompok tiga elemen sebelum titik koma pertama membentuk baris pertama, kelompok berikutnya membentuk baris kedua, dan seterusnya. Jelas jumlah elemen antara titik koma harus sama. Konsep ini berlaku juga ketika elemen itu sendiri adalah matriks. Sebagai contoh, perhatikan matriks 2 x 2

```
>> B=[45;89]
B =
   45
   89
```

Pernyataan :

```
>> C = [B B;B+4 B-1]
```

di mana B + 4 dan B-1 menunjukkan menambahkan 4 dan mengurangi 1 dari semua elemen B, masing-masing, menghasilkan hasilnya

```
C =
45   45
89   89
49   44
```

```
93    88
```
Contoh Lainnya :
```
>> D =[2*B 2*B;B+2 B-2]

D =
   90   90
  178  178
   47   43
   91   87

>> E =[5*B-5 5*B;B+20 B-20]
E =
  220  225
  440  445
   65   25
  109   69
```

D. Menghapus Row dan colom pada matrix

Anda dapat menghapus seluruh baris atau kolom dari sebuah matriks dengan menetapkan kumpulan tanda kurung siku [] kosong ke baris atau kolom tersebut. Pada dasarnya, [] menunjukkan larik kosong. Misalnya, mari kita hapus baris keempat dari C.

```
>> C(:,2)=[]
C =
45
89
49
 93
```

Pusat 2 x 2 dari matriks C asli dapat diekstraksi dengan menghapus baris pertama dan pertama dan kolom pertama dan terakhir dari array. Ini dicapai dalam dua langkah:

```
>> C(1:3:4,:)=[]
```

yang menghasilkan
C =
89
49
>> E(1:2:4,:)=[]

E =
  440 445
  109   69

E. Sifat-Sifat Matriks

Bagian ini kami memperkenalkan beberapa fungsi yang berguna untuk tujuan ini. Kami mengasumsikan matriks umum B ukuran M x N. Untuk contoh numerik kami menggunakan matriks B = [1 2 3;4 5 6;7 8 9].

>> B = [1 2 3;4 5 6;7 8 9]
B =
   1   2   3
   4   5   6
   7   8   9

Fungsi maks dan min Jika B adalah vektor (mis., Baik M = 1 atau N = 1), maks (B) mengembalikan nilai elemen terbesar dalam vektor. Jika A adalah sebuah matriks, maks (B) mengembalikan vektor baris yang mengandung elemen maksimum dari setiap kolom. Ini berarti bahwa, jika B adalah sebuah matriks, kami menggunakan.
>> max(max(B))
untuk mendapatkan nilai elemen terbesar di B. Misalnya,
>> max(B)
ans =
7   8   9

```
>> max(max(B))
ans =
9
```
Fungsi [V, R] = maks (B) mengembalikan nilai maksimum V di setiap kolom B, dan dalam R indeks baris yang sesuai dari nilai-nilai tersebut.

Misalkan,
```
>> [V,R] = max(B)
V =
   7    8    9
R =
   3    3    3
```

Fungsi min berperilaku dengan cara yang sama, kecuali bahwa fungsi mencari nilai minimum. Gunakan matriks yang sama seperti pada contoh sebelumnya,
```
>> min(B)
ans =
 1    2    3
>> min(min(B))
ans =
1
>> [V,R] = min(B)
V =
   1    2    3
R =
   1    1    1
```

Functions size, length, **and** ndims

Jika B adalah M x N matrix, ukuran fungsi (B) mengembalikan vektor baris dengan komponen M dan N. Jika D adalah vektor baris atau kolom, panjang (D) mengembalikan jumlah elemen dalam vektor. Perhatikan bahwa panjang (D) sama dengan maks (ukuran (D)). Akhirnya, fungsi ndims (B) mengembalikan jumlah dimensi array B (mis., 2 untuk sebuah matriks dan 1 untuk sebuah vektor). Dalam contoh berikut ini kami menggunakan matriks B yang sudah dikeringkan sebelumnya:

```
>> size(B)
ans =
3 3
>> S = size(B)
S =
3 3
>> D = B(2,:)
D =
3 9 4
>> size(D)
ans =
1 3
>> length(D)
ans =
3
>> ndims(B)
ans =
2
```

Function whos

Cara lain untuk mendapatkan informasi tentang properti matriks adalah dengan menggunakan fungsi whos. Misalnya, menerapkan fungsi ini ke matriks B dalam paragraf sebelumnya berikan

```
>>whos B
```

Name	Size	Bytes	Class	Attributes
B	3x3	72	double	

F. **Addition pada Matriks**

Penjumlahan matriks :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

Misalkan A + B :

$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ dan $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

Kode program yang dibuat adalah :

```
>>A = [1 2; 3 4]; B = [5 6; 7 8];
>> A = [1 2; 3 4]
A =
    1    2
    3    4

>> B = [5 6; 7 8]
B =
    5    6
    7    8

>> A+B
ans =
    6    8
   10   12

>> 2*A+2*B
ans =
   12   16
```

```
   20   24

>> 2*A-B+2*B
ans =
    7   10
   13   16
```

G. Pengurangan (Subtraction) pada Matriks

Pengurangan dalam Matriks :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$

Misalkan **C-D :**

$$C = \begin{bmatrix} 7 & 29 \\ 0 & 1 \end{bmatrix} \text{ dan } D = \begin{bmatrix} 6 & 4 \\ 11 & 9 \end{bmatrix}$$

Kode Programnya sebagai berikut :

```
>> C = [7 29; 0 1]; D = [6 4; 11 9];

>> C = [7 29; 0 1]
C =
    7   29
    0    1

>> D = [6 4; 11 9]
D =
    6    4
   11    9

>> C-D
ans =
    1   25
  -11   -8

>> 5*C-2*D
```

```
ans =
   23   137
  -22   -13
```

ukuran matriks harus sama baik pada penjumlahan atau pengurangan matriks.

H. Multiplication (Perkalian) pada Matriks

1. Multiplication (Perkalian) pada Matriks dengan skalar

$$c\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} = \begin{bmatrix} ca_{1,1} & ca_{1,2} \\ ca_{2,1} & ca_{2,2} \end{bmatrix}$$

Contoh :

$$a = 7 \text{ dan } B = \begin{bmatrix} 3 & 6 \\ -3 & 8 \\ 0 & 9 \end{bmatrix}$$

Kode Programnya :

```
>>a = 7; B = [3 6; -3 8; 0 9];
>> a = 7
a =
    7

>> B = [3 6; -3 8; 0 9]
B =
    3    6
   -3    8
    0    9

>> a*B
ans =
   21   42
  -21   56
```

```
    0   63

>> 2*a*2*B
ans =
   84  168
  -84  224
    0  252
```

2. Multiplication (Perkalian) pada Matriks dengan matriks Matriks A dengan ukuran s **x t** dan matriks B dengan ukuran t **x w**, apabila matrik A dan Matrik B dikalikan maka menghasilkan matriks AB berukuran s **x w**.

$$AB = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} g & h \\ i & j \\ k & l \end{bmatrix} = \begin{bmatrix} (ag+bi+ck) & (ah+bj+cl) \\ (dg+ei+fk) & (dh+dj+fl) \end{bmatrix}$$

Contoh :

$$A = \begin{bmatrix} 12 & 16 & 28 \\ 43 & 78 & 45 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 72 & 67 \\ 97 & 23 \\ 12 & 45 \end{bmatrix}$$

Kode Programnya adalah :

```
>> A=[12 16 28; 43 78 45]; B=[72 67; 97 23; 12 45];
>> A=[12 16 28; 43 78 45]
A =
   12   16   28
   43   78   45

>> B=[72 67; 97 23; 12 45]
B =
   72   67
   97   23
   12   45

>> A*B
```

```
Ans =
     2752       2432
    11202       6700

>> 2*A*2*B
ans =
    11008       9728
    44808      26800
```

Pada Matlab untuk perkalian matriks A dan matrik B, maka ukuran baris A harus sama dengan ukuran kolom B.

I. Pembagian pada Matriks

1. Matriks Identitas

Matriks identitas adalah matriks persegi yang memiliki 1 di sepanjang diagonal utama dan 0 untuk semua entri lainnya. Matriks ini sering ditulis hanya sebagai I, dan bersifat khusus karena berfungsi seperti 1 dalam perkalian matriks. Dalam pelajaran ini, kita akan melihat properti ini dan beberapa ide penting lainnya yang terkait dengan matriks identitas.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

2. Invers Matriks

Invers dari matriks A adalah matriks yang jika dikalikan dengan A menghasilkan identitas. Notasi untuk matriks invers ini adalah A^{-1}

$$AA^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A^{-1}A = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Didalam Matlab untuk mencari sebuah invers fungsi yang digunakan adalah syntax **inv(variabel).**

$$A = \begin{bmatrix} 3 & 5 \\ 1 & 2 \end{bmatrix}$$

```
>> A=[3 5; 1 2]
A =
3 5
1 2

>> B=inv(A)
B =
2.0000 -5.0000
-1.0000 3.0000
```

3. Determinan Matriks

syntax **det(variabel)** merupakan syntak yang digunakan untuk menghitung determinan suatu matriks berukuran mxn. Sebagai contoh bila kita akan menghitung determinan matriks A yang berukuran **5 x 5.**

```
>> A = [1 2 3 4 5; 6 7 8 9 1; 1 2 3 3 4; 1 2 6 7 8; 1 4 7 9 8]
A =
1 2 3 4 5
6 7 8 9 1
1 2 3 3 4
1 2 6 7 8
1 4 7 9 8

>> det_A = det(A)
det_A =
114
```

Pada Matlab ada 2 jenis pembagian yaitu right division (/) dan left division(\).

1. Right Division

Sebagai contoh apabila ada matriks **A** akan dibagi dengan matriks **B** maka menghasilkan matriks **C**, sehingga rumus yang digunakan adalah:

$$C = AB^{-1}$$

$$C = A/B$$

Sebagai contoh, bila diketahui,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ dan } \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$$

Maka akan nilai C dapat dihitung :

```
>> A=[1 2; 3 4];B = [6 7; 8 9];
>> A=[1 2; 3 4]
A =
   1   2
   3   4
>> B = [6 7; 8 9]
B =
   6   7
   8   9

>> A/B
ans =
3.5000 -2.5000
2.5000 -1.5000
```

2. Left Division

Sebagai contoh apabila ada matriks **D** dibagi dengan matriks **E** maka menghasilkan matriks **F**, sehingga rumus yang digunakan adalah :

$$F = D^{-1}E$$

$$F = D \setminus E$$

Sebagai contoh, bila diketahui,

$D = \begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix}$ dan $\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$

Sehingga akan dihasilkan nilai **F** :

```
>> D = [1 1; 3 2]; E=[4 5; 7 8];
>> D = [1 1; 3 2]
D =
   1   1
   3   2

>> E=[4 5; 7 8]
E =
   4   5
   7   8

>> D\E
ans =
-1 -2
5 7
```

Soal Latihan :

1. Diketahui matriks A dan B, Carilah Nilai nilai A+B dengan menggunakan Matlab.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

2. Diketahui matriks C dan D, kemudian carilah nilai C-D dengan menggunakan Matlab.

$$C = \begin{bmatrix} 7 & 29 \\ 0 & 1 \end{bmatrix} \text{ dan } D = \begin{bmatrix} 6 & 4 \\ 11 & 9 \end{bmatrix}$$

3. Diketahui matriks A dan B, hitunglah hasil kalinya dengan menggunakan Matlab.

$$A = \begin{bmatrix} 12 & 16 & 28 \\ 43 & 78 & 45 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 72 & 67 \\ 97 & 23 \\ 12 & 45 \end{bmatrix}$$

4. Carilah invers matriks A yang didefinisikan sebagai berikut.

$$A = \begin{bmatrix} 3 & 5 \\ 1 & 2 \end{bmatrix}$$

5. Carilah hasil bagi dari matriks A dibagi dengan matriks B.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ dan } \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$$

DAFTAR PUSTAKA

Getting Started With Matlab, Version 6 The MathWorks.Inc. 2002.

ANDI Yogyakarta 2000, “Matlab bahasa komputasi teknis”.

Munir, R. 1999. Algoritma dan Pemrograman Dalam Bahasa Pascal dan C. Bandung: Informatika.

Kadir, A dan Heriyanto, 2005, ”Algoritma Pemrograman Menggunakan C++”. Yogyakarta: Penerbit Andi.

Pranata, A. 2005, Algoritma dan Pemrograman. Yogyakarta: Penerbit Graha Ilmu.

Levitin, Anany, 2010, “Pengantar Desain dan Analisis Algoritma”, Edisi 2, Buku 1, Penerbit Salemba Infotek.

Wirth, Nicholas, 2004, “Algorithms and Data Structures”, Oberon version: August 2004, Prentice Hall.

The MIT Press, 2001, “Introduction to Algorithms” Second Edtion, McGraw-Hill.

Drozdek, Adam, 2001, “Data Structures and Algorithms in C++”, 2nd Edition, Brooks/Cole Thomson Learning.

Deitel, H. M. & Deitel P. J., 2001, “C: How to Program”. Third Edition. Pearson Education Prentice Hall. Prentice-Hall, New Jersey.

BIODATA PENULIS

Dr.Hindarto, S.Kom, MT. dilahirkan di Surabaya, 30 Juli 1973. Pada tahun 1995, penulis mendapatkan gelar Diploma dari Politeknik Negeri Surabaya dan melanjutkan jenjang Sarjana di prodi Informatika FakultasTeknik Umsida. Penulis melanjutkan magister Teknik Elektro ITS dengan program beasiswa dari DIKTI. Tahun 2007, penulis secara resmi mendakatkan gelar M.T. Penulis melanjutkan doctoral di Jaringan Cerdas Multimedia Teknik Elektro ITS dengan program beasiswa dari DIKTI. Tahun 2016, penulis secara resmi mendakatkan gelar Dr. Penulis mengawali karirnya sebagai Dosen di prodi Informatika Universitas Muhaammdiyah Sidoarjo pada Tahun 2004. Beberapa penelitian yang pernah dilakukan oleh penulis adalah tentang Deteksi Sinyal Jantung dan Deteksi Sinyal EEG.

Ade Eviyanti, S.Kom, M.Kom

dilahirkan di Jakarta, 24 Juni 1978. Pada tahun 2003, penulis mendapatkan gelar Sarjana di prodi Informatika FakultasTeknik Umsida. Penulis melanjutkan magister Informatika di STTS Suranaya. Tahun 2019, penulis secara resmi mendakatkan gelar M.Kom. Penulis mengawali karirnya sebagai Dosen di prodi Informatika Universitas Muhaammdiyah Sidoarjo pada Tahun 2006. Beberapa penelitian yang pernah dilakukan oleh penulis adalah tentang Deteksi Sinyal Jantung dan Deteksi Sinyal EEG.