BAB I
BERPIKIR KOMPUTASIONAL

A. DIAGRAM ALUR

Diagram alur (flowchart) adalah representasi visual dari proses atau algoritma yang berfungsi memepermudah memahami,merancang, dan mengomunikasikan solusi atas suatu permasalahan dengan menggambarkan sebuah urutan langkah atau proses pada suatu sistem atau pun program. Flowchart menunjukan alur kerja atau aliran informasi dari sutu langkah ke langkah lainnya menggunakan simbol grafis dan panah yang menghubungkannya. Tujuan diagram alur adalah penyajian sebuah proses secara jelas dan mudah dipahami oleh pembaca. Penggunaan. Diagram alur kerja memungkinkan seseorang dapat memahami urutuan langkah – langkah yang perlu dilakukan dalam suatu proses dan meengidentiifikasi bagian – bagian yang mungkin memerlukan perbaikan atau optimalisasi dengan mudah. Diagram alur juga dapat mengomunikasikan proses kepada orang lain, seperti tim kerja, timpengembang, atau pemangku kepentingan lainnya. Berikut dijelaskan lebih rinci mengenai prinsip dasar dalampembuatan diagram alur simbol diagram alur, dan struktur diagram alur, serta pengaplikasikan diagram alur dalam pratik komputasional pada aplikasi Flowgorithm.

1. **Prinsip Dasar dalam Pembuatan Diagram Alur**

Berikut adalah prinsip – prinsip dasar dalam pembuatan diagram alur.

a. Aliran diagram harus jelas, umumnya mengalir dari atas ke bawah atau ke kiri ke kanan. Gunakan panah untuk menunjukkan alur proses.

b. Gunakan satu simbol untuk satu langkah proses, hindari menuliskan beberapa proses dalam satu simbol

c. Terdapat label pada percabangan, jika terdapat percabangan dalam keputusan, tandai tiap cabang dengan “Ya” atau “Tidak”agar lebih jelas.

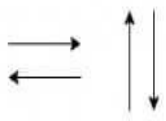
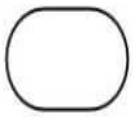
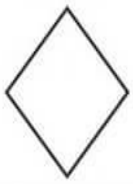
d. hindari garis yang berpotongan, pastikan garis aliran tidak tumpang tindih agar mudah dipahami. Gunakan konektor jika diagram alur terlalu panjang.

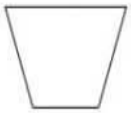
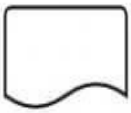
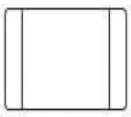
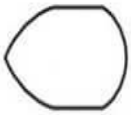
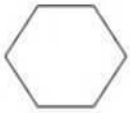
e. Gunakan nama yang jelas untuk variabel dan proses, misalnya “TotalHarga” dari pada hanya “X”. Nama proses juga harus deskriptif, misalnya “Hitung Luas Persegi” bukan hanya “Proses”.

2. **Simbol Diagram Alur**

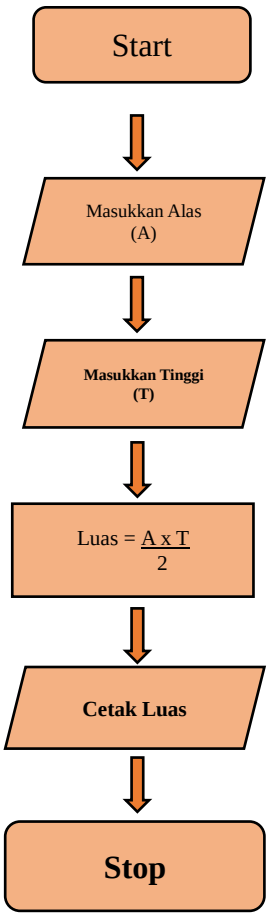
Diagram alur menggunakan berbagai simbol standar untuk merepresentasikan prosesa dalam suatu algoritma atau sistem. Simbol diagram alur digunakan secara kombinasi untuk langkah input, data, koneksi dan keputusan. Berikut adalah contohnya.

Tabel : Simbol diagram Alur

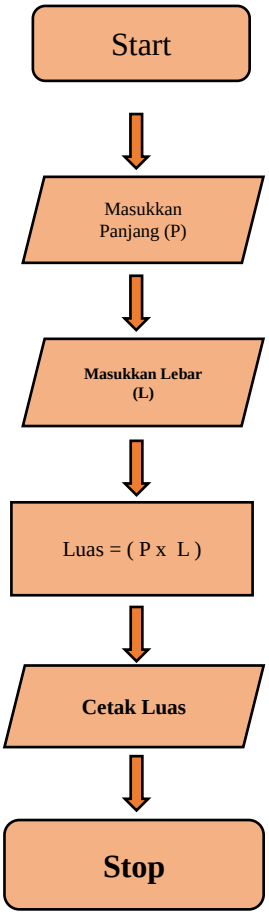
	**Flow** Simbol yang digunakan untuk menggabungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga dengan Connecting Line.
	**On-Page Reference** Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang sama.
	**Off-Page Reference** Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang berbeda.
	**Terminator** Simbol yang menyatakan awal atau akhir suatu program.
	**Process** Simbol yang menyatakan suatu proses yang dilakukan komputer.
	**Decision** Simbol yang menunjukan kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, yaitu ya dan tidak.

	**Input/output** Simbol yang menyatakan proses input atau output tanpa tergantung peralatan.
	**Manual Operation** Simbol yang menyatakan suatu proses yang tidak dilakukan oleh komputer.
	**Document** Simbol yang menyatakan bahwa input berasal dari dokumen dalam bentuk fisik, atau output yang perlu dicetak.
	**Predefine Proses** Simbol untuk pelaksanaan suatu bagian (sub-program) atau prosedure.
	**Display** Simbol yang menyatakan peralatan output yang digunakan.
	**Preparation** Simbol yang menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan nilai awal.

Berikut adalah contoh penggunaanya



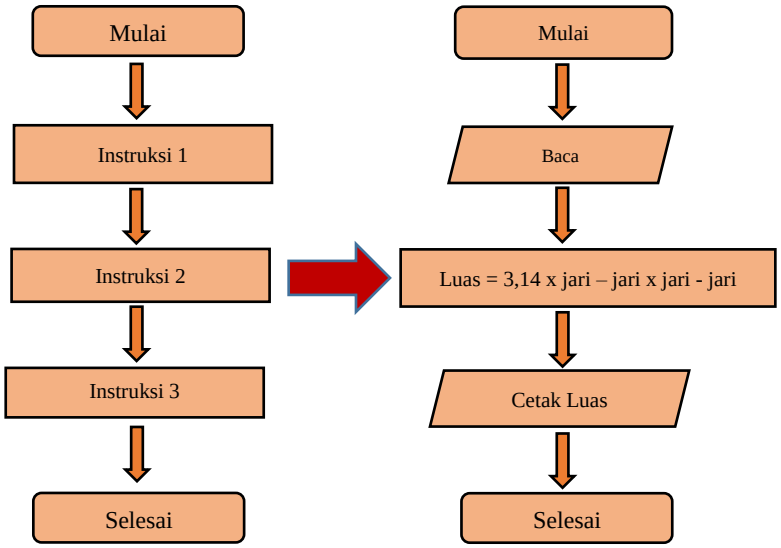
(a)



(b)

3. **Struktur Diagram Alur**

terdapat beberapa struktur diagram alur yang berfungsi merepresentasikan langkah sebuah proses, yaitu sebagai berikut.



a. Runtutan atau bertahap (*sequence*)

Runtutan adalah struktur diagram alur yang tiap langkah pelaksanaannya (instruksi) dikerjakan secara urut dari awal hingga akhir. Instruksi sebelumnya harus diselesaikan sebelum melanjutkan ke instruksi berikutnya. Instruksi pertama akan dieksekusi terlebih dahulu dalam program. Setelah instruksi pertama selesai, program akan melanjutkan ke instruksi kedua dan seterusnya hingga instruksi seluruhnya selesai.

Gambar tersebut menunjukkan diagram alur algoritma pada tahap program menghitung luas lingkaran dengan tiga langkah sebagai berikut.

1. Langkah atau instruksi 1, yaitu membaca data dari panjang jari-jari lingkaran.
2. Langkah kedua atau instruksi 2, yaitu menghitung dari luas lingkaran.
3. Langkah ketiga atau instruksi 3, yaitu mencetak luas pada layar.

Catatan: Langkah kedua tidak dapat dilakukan sebelum langkah pertama diselesaikan, begitu pula langkah ketiga tidak dapat dilakukan sebelum langkah kedua selesai.

b. Bersyarat (*conditional*)

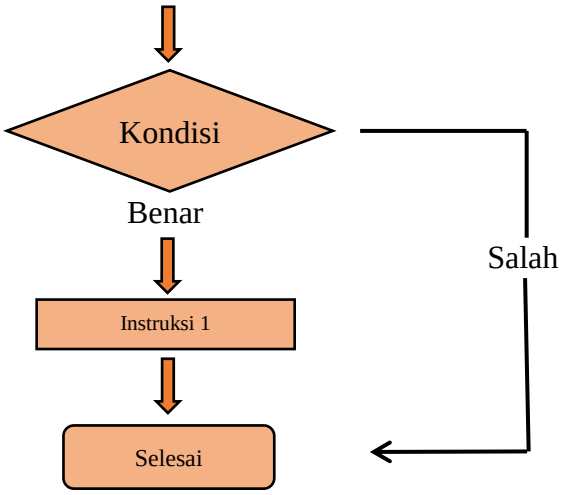
Pada algoritma, tidak semua baris instruksi dapat dieksekusi dengan cara berurutan. Terdapat kondisi tertentu ketika proses harus dilewati untuk melanjutkan ke proses lainnya. Bentuk instruksi ini disebut sebagai alur pemilihan atau percabangan. Kondisi adalah syarat yang ditetapkan dan dievaluasi sebagai "benar" atau "salah", sedangkan aksi adalah tindakan yang diambil jika kondisi tersebut dianggap "benar". Algoritma percabangan dapat menangani lebih dari satu kasus. Berikut dijelaskan mengenai percabangan satu kasus, dua kasus, dan bertingkat.

1. **Percabangan satu kasus (*if-then*)**

   Pada percabangan ini, aksi akan dilakukan pada kondisi bernilai "benar." Jika kondisi tersebut bernilai "salah" maka tidak ada tindakan yang akan dilakukan. Sebagai contoh, lampu ruangan otomatis. Jika ada orang, lampu akan menyala. Namun, jika tidak ada orang, lampu akan mati atau proses selesai.

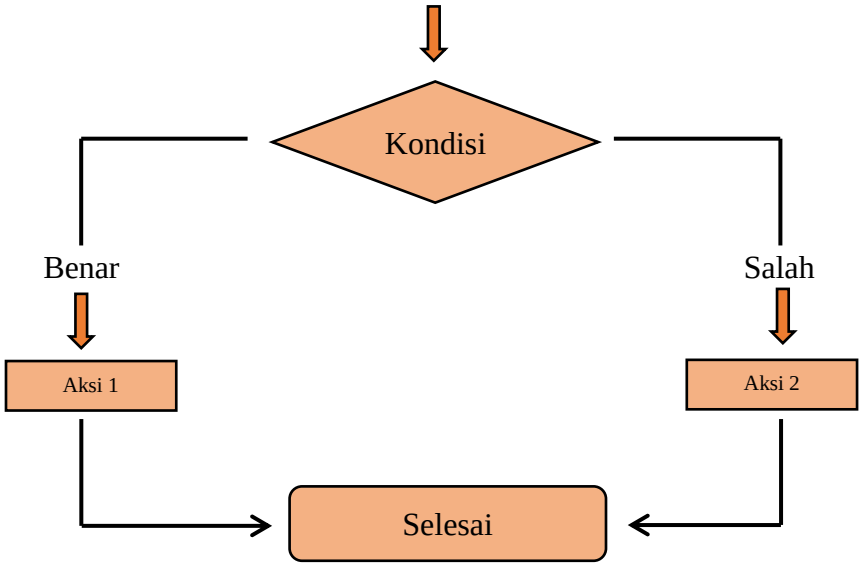
2. **Percabangan dua kasus (*if-then-else*)**

   Bentuk *if-then* yang telah dijelaskan sebelumnya hanya dapat menangani satu alternatif keadaan. Namun, pada praktiknya, sering kali terdapat lebih dari satu kondisi yang harus diperhatikan. Untuk mengatasi situasi ini, Anda dapat menggunakan struktur *if-then-else*. Sebagai contoh, kegiatan membeli telur di warung. Jika harga telur kurang dari Rp15.000,00 maka dapat membeli sebanyak 2 kg. Namun, jika harga telur lebih dari Rp15.000,00 maka hanya dapat membeli 1 kg.



Sumber: dokumen penerbit/Penerbit Erlangga

Gambar 1.4 Diagram alur struktur percabangan satu kasus.



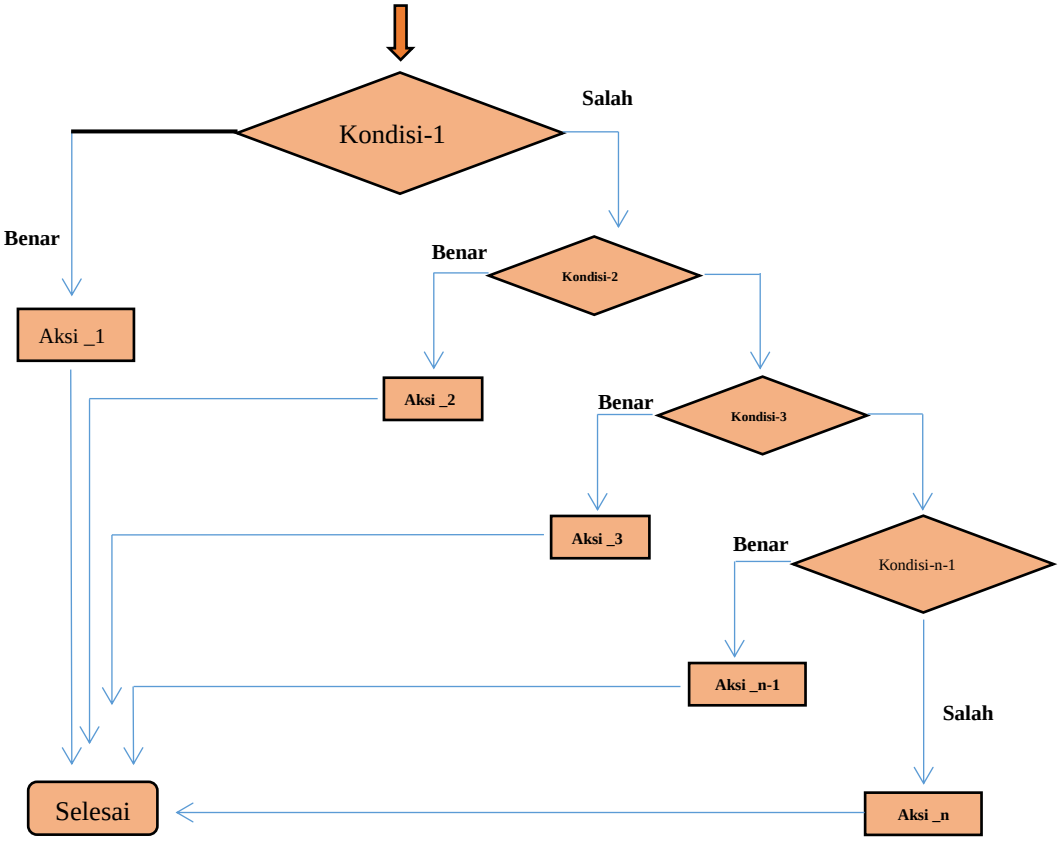
Sumber: dokumen penerbit/Penerbit Erlangga

Gambar 1.5 Diagram alur struktur percabangan dua kasus.

3. **Percabangan bertingkat (*if-then-else-if-then*)**

Bagaimana Anda akan mendefinisikan algoritma, jika diminta untuk menentukan sebuah bilangan bulat yang dimasukkan ke dalam program adalah bilangan positif, negatif, atau nol? Tentu saja, Anda tidak dapat hanya menggunakan struktur percabangan dua kasus. Pada algoritma, Anda dapat menggunakan struktur *if-then-else if-then* ketika menghadapi lebih dari dua percabangan (kondisi). Struktur ini juga dikenal sebagai "if bertingkat" atau "if bersarang" (*nested if*). Gambar berikut menunjukkan struktur percabangan bertingkat.

Sintaks:
if kondisi_1 then
Aksi_1 else if
kondisi_2 then
Aksi_2 else if
kondisi_3 then
Aksi_3 ...
else if
kondisi_n-1
Aksi_n-1 else
Aksi_n
endif



Sumber: dokumen penerbit/Penerbit Erlangga

Gambar 1.7 Diagram alur struktur percabangan bertingkat.

c. Pengulangan (*looping*)

Struktur pengulangan adalah instruksi yang digunakan untuk menjalankan serangkaian tindakan yang mengikuti pola tertentu secara berulang. Pada pemrograman, struktur ini biasa dikenal dengan istilah *looping* atau *repetition*. Secara umum, struktur pengulangan terdiri atas dua komponen utama, yaitu kondisi pengulangan dan badan pengulangan. Kondisi pengulangan adalah syarat yang harus dipenuhi untuk melakukan pengulangan dan nilainya harus "benar" atau "salah". Sementara itu, badan pengulangan adalah serangkaian aksi yang akan dieksekusi secara berulang.

Pada pemrograman, terdapat berbagai jenis struktur pengulangan yang dapat diterapkan. Beberapa jenis struktur dapat digunakan untuk menyelesaikan masalah yang serupa, tetapi pemilihan jenis pengulangan yang tepat sangat krusial karena kesalahan dalam

memilih jenis pengulangan bisa mengakibatkan hasil yang tidak akurat. Tiap bahasa pemrograman memiliki cara yang berbeda untuk mendefinisikan jenis pengulangan. Secara umum, struktur pengulangan dibagi menjadi dua kategori, yaitu sebagai berikut.

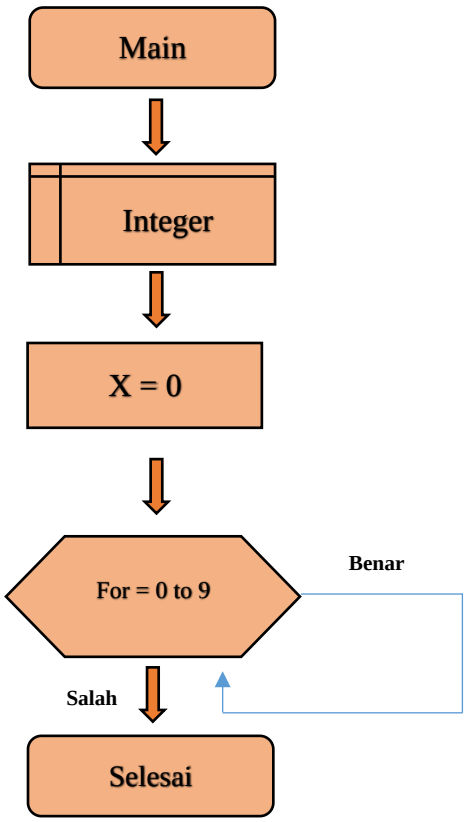
1. **Tanpa kondisi**

   Struktur ini merujuk pada pengulangan, yaitu instruksi dijalankan sejumlah n kali dengan jumlah pengulangan yang sudah diketahui sebelumnya. Pada pemrograman, kategori ini biasanya menggunakan perintah "*for*."

2. **Dengan kondisi**

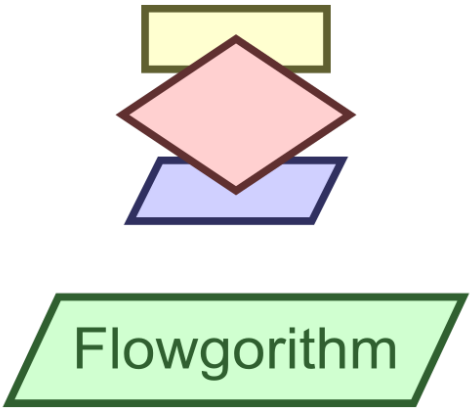
   Pada beberapa situasi, jumlah pengulangan tidak dapat diprediksi sebelumnya, tetapi pengulangan dapat dimulai atau dihentikan berdasarkan syarat tertentu. Instruksi dalam pengulangan akan terus berlangsung sampai kondisi untuk menghentikannya terpenuhi. Kategori ini biasanya diwakili oleh perintah "*while*" dan "*repeat*."

Sebagai contoh, Anda ingin membuat program yang akan selesai menghitung setelah mencapai angka 10. Untuk itu, Anda perlu membuat variabel terlebih dahulu, misalnya X. Variabel ini berfungsi menyimpan jumlah angka. Untuk tipe data angka, Anda dapat menggunakan integer. Nilai awal dari variabel X adalah 0, tetapi setiap perulangan berlangsung, variabel ini akan ditambah 1 hingga mencapai angka 10. Setelah variabel ini bernilai 10, perulangan akan berakhir dan program akan mengeluarkan *output* 10. Diagram alur untuk program ini dapat Anda lihat pada gambar di samping.



4. Mengaplikasikan Diagram Alur dalam Praktik Komputasional pada Aplikasi Flowgorithm

Mengaplikasikan diagram alur akan lebih optimal jika menggunakan aplikasi pendukung, seperti Flowgorithm. Flowgorithm adalah alat yang sangat baik dalam mengembangkan pemikiran logis, Keterampilan dasar untuk pemecahan masalah, dan pemrograman secara visual. Aplikasi ini membekali peserta didik dengan pola pikir untuk mendekati masalah secara metodis dan sistematis, baik dalam pemrograman maupun di luar pemrograman.



Sumber: commons.wikimedia.org/DevinCook
Gambar 1.10 Flowgorithm dapat menjadi aplikasi untuk melatih peserta didik dalam memahami algoritma secara visual

Hal ini sejalan dengan penggunaan diagram alur untuk menggambarkan proses secara visual sehingga memungkinkan peserta didik belajar berpikir secara logis dan terstruktur. Berpikir logis merupakan kemampuan menganalisis, memecahkan, dan menemukan solusi dari masalah secara sistematis sehingga dapat membuat keputusan berdasarkan alasan yang masuk akal. Fokus dalam berpikir logis adalah mengatur langkah-langkah dalam urutan yang benar, mengambil keputusan, dan menggunakan kondisi untuk menentukan hasil. Pemecahan masalah dilakukan dengan mengidentifikasi dan menerapkan solusi secara efektif. Berikut adalah penjelasannya.

a. Meningkatkan pemikiran logis

Melalui pendekatan visual dan interaktif, peserta didik dapat memahami cara kerja algoritma secara lebih konkret dan sistematis. Berikut adalah lima cara utama Flowgorithm dalam meningkatkan pemikiran logis.

1. **Visualisasi logika**, diagram alur memberikan gambaran yang jelas tentang alur suatu algoritma. Pengguna dapat melihat keputusan menghasilkan hasil yang berbeda, melacak kesalahan dengan mengikuti logika secara visual, serta mengidentifikasi inefisiensi dan memperbaiki proses.
2. **Memecah masalah**, Flowgorithm mendorong proses dekomposisi masalah, yaitu masalah yang rumit dipecah menjadi langkah-langkah yang dapat dikelola dengan lebih detail. Tiap blok mewakili tindakan, masukan, atau keputusan tertentu.
3. **Pengujian dan *debugging***, menjalankan diagram alur langkah demi langkah dapat membantu mengidentifikasi kesalahan logika, serta mendorong pemikiran kritis untuk menyempurnakan dan mengoptimalkan algoritma.
4. **Memahami pemikiran kondisional**, peserta didik berlatih menciptakan kondisi yang mengarahkan ke cabang yang berbeda. Merancang proses berulang dengan titik masuk dan keluar yang jelas.
5. **Mendorong pemecahan masalah terstruktur**, Flowgorithm menekankan penggunaan pernyataan yang jelas dalam mendefinisikan masalah. Hal ini dapat mempermudah proses merancang solusi secara sistematis sebelum implementasi.