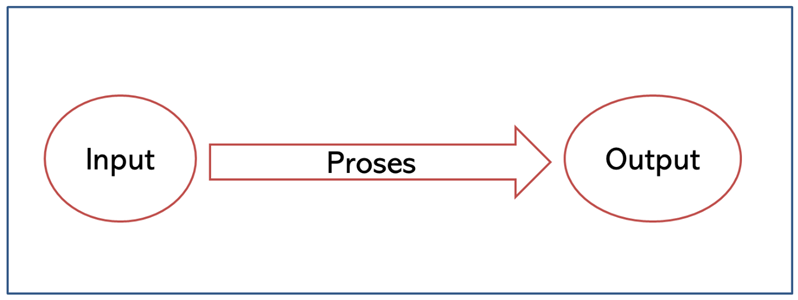
* 1. **PENGERTIAN ALOGARITMA**

Komputer hanyalah suatu alat yang diinstruksikan oleh manusia untuk menyelesaikan masalah dengan cepat, akurat, dan berulang-ulang tanpa bosan dan lelah, pandangan bahwa komputer adalah mesin yang "pintar" adalah salah. Nama "program" mengacu pada sekumpulan arahan yang dimaksudkan untuk menyelesaikan masalah tersebut. Program harus ditulis dalam bahasa yang komputer dapat memahami agar dapat digunakan. Bahasa pemrograman adalah bahasa komputer yang digunakan untuk menulis program. Algoritma adalah urutan langkah-langkah yang sistematis yang digunakan untuk menyelesaikan sebuah masalah. Di bidang pemrograman, algoritma berarti solusi, dan ketika seseorang berbicara tentang algoritma dalam bidang ini, mereka bermaksud untuk menemukan solusi untuk suatu masalah yang perlu diselesaikan menggunakan komputer. Agar algoritma dapat dipahami dan dilaksanakan oleh komputer, algoritma harus dibuat secara runut. Dalam pembuatan algoritma, analisis kasus sangat penting. Ini menunjukkan proses apa yang diperlukan untuk menyelesaikan masalah yang harus diselesaikan.

Algoritma adalah bagian integral dari teknologi dan kehidupan sehari-hari kita, membantu kita menjalani kehidupan yang lebih teratur, efisien, dan nyaman.

* Secara umum, definisi algoritma berarti: Langkah-langkah yang sistematis dan logis untuk menyelesaikan masalah (INPUT) sehingga menghasilkan penyelesaian (OUTPUT) yang diinginkan.
* Urutan Langkah Langkah yang disusun secara logis dan sistematis untuk memcahkan suatumasalah. Algoritma haruslah logis atau benar, artinya algoritma harus memberikankeluaran yang dikehendaki dari sejumlah masukan yang diberikan.



Contoh algoritma di kehidupan sehari-hari adalah **mesin penjual otomatis (vending machine)**:

**Input**:

* Pengguna memasukkan uang atau koin ke dalam mesin.
* Pengguna memilih produk yang ingin dibeli dengan menekan tombol yang sesuai.

**Proses**:

* Mesin menerima uang atau koin dan memeriksa apakah jumlahnya cukup untuk membeli produk yang dipilih.
* Jika jumlah uang cukup, mesin mengeluarkan produk yang dipilih.
* Jika jumlah uang kurang, mesin menolak transaksi dan mengembalikan uang yang dimasukkan oleh pengguna.
* Mesin juga dapat memberikan kembalian jika jumlah uang yang dimasukkan lebih besar dari harga produk.

**Output**:

* Jika uang cukup, pengguna menerima produk yang dipilih.
* Jika uang tidak cukup, pengguna menerima kembalian atau peringatan bahwa uangnya kurang.

Algoritma mesin penjual otomatis ini adalah contoh sederhana dari bagaimana input (uang/koin dan pilihan produk), proses (pengecekan uang, pemilihan produk, dan pengeluaran produk), dan output (produk atau kembalian) bekerja bersama untuk menyediakan layanan yang diinginkan oleh pengguna.

Ciri-ciri algoritma yang baik meliputi beberapa aspek penting agar algoritma tersebut dapat bekerja dengan efisien, efektif, dan mudah dipahami. Berikut adalah ciri-ciri algoritma yang baik:

1. Langkah-langkah yang jelas dan terperinci:

Algoritma harus memiliki instruksi atau langkah-langkah yang jelas dan mudah dimengerti tanpa adanya keraguan. Setiap langkah harus menggambarkan tindakan yang spesifik dan terukur.

1. Finita (Berakhir dalam waktu terbatas):

Algoritma harus memiliki titik akhir, artinya algoritma harus berakhir dalam jumlah langkah yang terbatas. Algoritma yang tidak berakhir (infinite loop) adalah algoritma yang salah atau tidak efektif.

1. Efisien:

Algoritma yang baik harus menyelesaikan masalah dengan cara yang efisien dalam hal waktu dan penggunaan sumber daya (seperti memori). Algoritma yang membutuhkan waktu yang lebih sedikit atau lebih sedikit langkah lebih baik daripada algoritma yang lebih rumit.

1. Tidak ambigu (Tidak ganda):

Setiap langkah dalam algoritma harus memiliki satu arti yang jelas. Tidak boleh ada langkah yang dapat ditafsirkan dengan cara yang berbeda. Kejelasan sangat penting agar algoritma dapat diikuti dengan tepat.

1. Input dan output yang jelas:

Algoritma harus menerima input yang jelas dan menghasilkan output yang sesuai dengan tujuan yang ingin dicapai. Proses pengolahan input dan hasil output harus terdefinisi dengan baik.

1. Generalisasi:

Algoritma yang baik dapat digunakan untuk memecahkan berbagai masalah serupa, bukan hanya untuk satu masalah khusus. Artinya, algoritma harus fleksibel dan dapat diadaptasi pada berbagai kondisi.

1. Logis dan sistematis:

Langkah-langkah dalam algoritma harus disusun secara logis dan sistematis. Proses berpikir harus mengalir dengan urutan yang benar dan tidak melompat-lompat.

1. Dapat diterapkan pada komputer (otomatisasi):

Algoritma harus dapat diterjemahkan ke dalam kode pemrograman dan dijalankan oleh komputer untuk mencapai hasil yang diinginkan.

1. Sederhana:

Algoritma yang baik tidak perlu rumit. Semakin sederhana dan mudah dipahami, semakin baik. Namun, kesederhanaan tidak berarti mengorbankan keefektifan atau fungsionalitas.

* 1. **PENYAJIAN ALGORITMA**

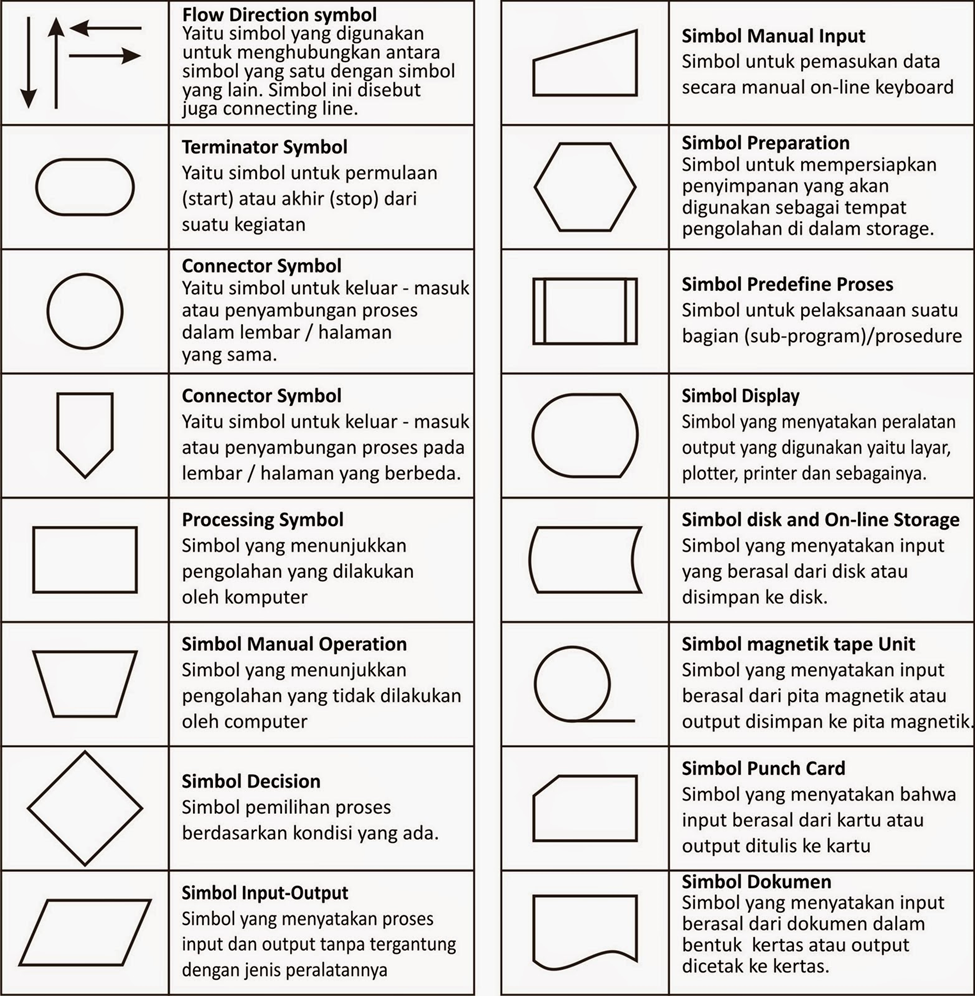
Penyajian algoritma terbagi menjadi:

* Deskriptif
* Flowchart
* Pseudocode
* Deskriptif
* Penyajian deskriptif adalah penjelasan naratif atau dalam bentuk teks tentang bagaimana algoritma berfungsi.
* Ini sering digunakan dalam bahasa manusia yang mudah dipahami dan diikuti oleh pengembang atau pembaca algoritma.
* Deskripsi langkah demi langkah algoritma diberikan dalam bahasa alami atau bahasa pemrograman yang mudah dipahami.

Contoh : misalnya algoritma menentukan bilangan terbesar dari 3 bilangan berikut ini: Algoritma Menentukan\_bilangan\_terbesar\_dari\_3\_bilangan :

* Meminta input 3 bilangan dari user, misalkan bilangan a, b, dan c.
* Apabila bilangan a lebih besar dari b maupun c, maka bilangan a merupakan bilangan terbesar
* Jika tidak (bilangan a tidak lebih besar dari b atau c) berarti bilangan a sudah pasti bukan bilangan terbesar. Kemungkinannya tinggal bilangan b atau c. Apabila bilangan b lebih besar dari c, maka b merupakan bilangan terbesar. Sebaliknya apabila bilangan b tidak lebih besar dari c, maka bilangan c merupakan yang terbesar.
* Selesai.
* Flowchart
* Flowchart adalah representasi visual dari algoritma dalam bentuk diagram yang terdiri dari simbol-simbol geometris seperti kotak, panah, dan berlian.
* Setiap simbol mewakili langkah-langkah atau keputusan dalam algoritma, dan panah menghubungkannya untuk menunjukkan alur eksekusi.
* Flowchart membantu pengguna untuk secara visual melihat aliran logika dan cabang keputusan dalam algoritma.

Berikut adalah simbol dan kegunaan pada flowchart:



Aturan dalam membuat Flowchart:

Flowchart digambarkan dari halaman atas ke bawah dan dari kiri kekanan.

* Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat di mengerti oleh pembacanya.
* Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
* Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja.
* Setiap langkah dari aktivitas harus berada pada urutan yang benar.
* Lingkup dan range dari aktifitas yang sedang digambarkan harusditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
* Gunakan simbol-simbol flowchart yang standar.
* Pseudocode

Pseudocode adalah sebuah metode penulisan menggunakan bahasa sederhana untuk mengekspresikan desain algoritma. Pseudocode bermanfaat untuk mempermudah manusia untuk memahami algoritma karena lebih mudah dipahami dibandingkan dengan bahasa pemrograman. Penulisan pseudocode tidak memiliki aturan pasti namun harus logis.

Secara umum, pseudocode akan berisi tiga bagian utama ini yang saling mendukung dalam menyusun algoritma yang jelas, mudah dipahami, dan terstruktur. Sebuah pseudocode yang baik akan menunjukkan bagaimana input diproses untuk menghasilkan output dengan jelas, serta bagaimana kontrol alur bekerja untuk mengambil keputusan atau mengulangi proses tertentu.

* 1. **TIPE DATA DAN VARIALE**

**Variable**

Variable adalah tempat penyimpanan data dalam Python. Variable dapat digunakan untuk menyimpan berbagai jenis data, seperti integer, float, string, dan boolean.

Contoh Kode Python:

nama = "Sayyid"

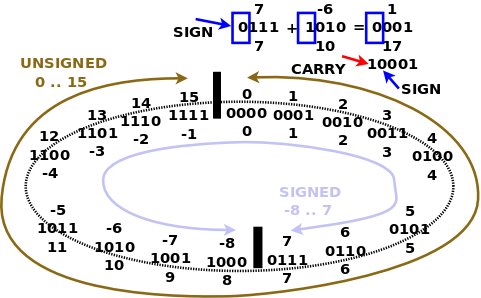
usia = 18

print("Nama saya adalah", nama, "dan usia saya adalah", usia)

Kode di atas mendeklarasikan dua variable, yaitu nama dan usia. Variable nama menyimpan data string "Bard", dan variable usia menyimpan data integer 18.

**Tipe Data Integer**

* Representasi internal : bilangan bulat dan bilangan bulat bukan negative.
* Umumnya operasi dalam operasi aritmatika dan operasi logika pada tingkat bit.



**Luas Segitiga dengan Percabangan**

Luas segitiga dapat dihitung dengan menggunakan rumus:

luas = (alas \* tinggi) / 2

Dengan menggunakan percabangan, kita dapat membuat program untuk menghitung luas segitiga dengan input dari pengguna.

# Menentukan luas segitiga

alas = float(input("Masukkan alas segitiga: "))

tinggi = float(input("Masukkan tinggi segitiga: "))

if alas <= 0 or tinggi <= 0:

print("Nilai alas atau tinggi tidak valid")

else:

luas = (alas \* tinggi) / 2

print("Luas segitiga:", luas)

Pada contoh kode di atas, kita menggunakan percabangan untuk menentukan apakah nilai alas dan tinggi segitiga valid. Jika nilai alas atau tinggi tidak valid, maka akan ditampilkan pesan kesalahan. Jika nilai alas dan tinggi valid, maka akan dihitung luas segitiga dan ditampilkan hasilnya.