[Home](http://tkj.smkmutu.com/) » [Pemrograman Dasar](http://tkj.smkmutu.com/search/label/Pemrograman%20Dasar) » Materi Pemrograman Dasar Kelas X

[Materi Pemrograman Dasar Kelas X](http://tkj.smkmutu.com/2015/08/materi-pemrograman-dasar-kelas-x.html)

By [Fiqo Al Fatih](https://plus.google.com/111680269061132869542)[00.40](http://tkj.smkmutu.com/2015/08/materi-pemrograman-dasar-kelas-x.html)[No comments](http://tkj.smkmutu.com/2015/08/materi-pemrograman-dasar-kelas-x.html#comment-form)

[](http://3.bp.blogspot.com/-b57w4wfh3F8/VcxJ3LQ-A_I/AAAAAAAAXGc/GEE6szEiais/s1600/Algoritma%2Bdan%2BPemrograman%2BVisual%2BC%252B%252B.jpg)

[**Algoritma dan Pemrograman**](https://andikafisma.wordpress.com/algoritma-dan-pemrograman/)

**1.  Apakah Itu Algoritma**

Ditinjau dari asal-usul katanya, kata Algoritma sendiri mempunyai sejarah yang aneh. Orang hanya menemukan kata *algorism*yang berarti proses menghitung dengan angka arab. Anda dikatakan *algorist*jika Anda menghitung menggunakan angka arab. Para ahli bahasa berusaha menemukan asal kata ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal kata tersebut yang berasal dari nama penulis buku arab yang terkenal yaitu Abu Ja’far Muhammad Ibnu Musa Al-Khuwarizmi. Al-Khuwarizmi dibaca orang barat menjadi *Algorism.*Al-Khuwarizmi menulis buku yang berjudul *Kitab Al Jabar Wal-Muqabala*yang artinya “Buku pemugaran dan pengurangan” (*The book of* *restoration and reduction*). Dari judul buku itu kita juga memperoleh akar kata “Aljabar” (*Algebra*). Perubahan kata dari *algorism*menjadi*algorithm* muncul karena kata *algorism*sering dikelirukan dengan *arithmetic*, sehingga akhiran *–sm*berubah menjadi *–thm.*Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya. Dalam bahasa Indonesia, kata *algorithm*diserap menjadi *algoritma*.

**2.  Definisi Algoritma**

*“Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis”.*Kata *logis*merupakan kata kuncidalam algoritma. Langkah-langkah dalam algoritma harus logis dan harusdapat ditentukan bernilai salah atau benar.Dalam beberapa konteks, algoritma adalah spesifikasi urutan langkah untukmelakukan pekerjaan tertentu. Pertimbangan dalam pemilihan algoritmaadalah, pertama, algoritma haruslah benar. Artinya algoritma akan memberikankeluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidakpeduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik.

Pertimbangan kedua yang harus diperhatikan adalah kita harus mengetahui seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil (hasil yang hanya berupa pendekatan). Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

Ketiga adalah efisiensi algoritma. Efisiensi algoritma dapat ditinjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar (paling mendekati), tetapi jika kita harus menunggu berjam-jam untuk mendapatkan keluarannya, algoritma tersebut biasanya tidak akan dipakai, setiap orang menginginkan keluaran yang cepat. Begitu juga dengan memori, semakin besar memori yang terpakai maka semakin buruklah algoritma tersebut. Dalam kenyataannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya kita mengharapkan keluaran yang sama. Jika terjadi demikian, carilah algoritma yang paling efisien dan cepat.

**3.  Beda Algoritma dan Program**

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman. Beberapa pakar memberi formula bahwa :

**Program = Algoritma + Bahasa (Struktur Data)**

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya.

Pembuatan algoritma mempunyai banyak keuntungan di antaranya :

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

Beberapa hal yang perlu diperhatikan dalam membuat algoritma :

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

a.  Pendeklarasian variabel

Untuk mengetahui dibutuhkannya pendeklarasian variabel dalam penggunaan bahasa pemrograman apabila    tidak semua bahasa pemrograman membutuhkannya.

b.  Pemilihan tipe data

Apabila bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel maka perlu hal ini dipertimbangkan pada saat pemilihan tipe data.

c.  Pemakaian instruksi-instruksi

Beberapa instruksi mempunyai kegunaan yang sama tetapi masing-masing memiliki kelebihan dan kekurangan yang berbeda.

d.  Aturan sintaksis

Pada saat menuliskan program kita terikat dengan aturan sintaksis dalam bahasa pemrograman yang akan digunakan.

e.  Tampilan hasil

Pada saat membuat algoritma kita tidak memikirkan tampilan hasil yang akan disajikan. Hal-hal teknis ini diperhatikan ketika mengkonversikannya menjadi program.

f.  Cara pengoperasian *compiler*atau *interpreter*.

Bahasa pemrograman yang digunakan termasuk dalam kelompok *compiler*atau *interpreter*.

**4.  Algoritma Merupakan Jantung Ilmu Informatika**

Algoritma adalah jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang mengarah ke dalam terminologi algoritma. Namun, jangan beranggapan algoritma selalu identik dengan ilmu komputer saja. Dalam kehidupan sehari-hari pun banyak terdapat proses yang dinyatakan dalam suatu algoritma. Cara-cara membuat kue atau masakan yang dinyatakan dalam suatu resep juga dapat disebut sebagai algoritma. Pada setiap resep selalu ada urutan langkah-langkah membuat masakan. Bila langkah-langkahnya tidak logis, tidak dapat dihasilkan masakan yang diinginkan. Ibu-ibu yang mencoba suatu resep masakan akan membaca satu per satu langkah-langkah pembuatannya lalu ia mengerjakan proses sesuai yang ia baca. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses (*processor*). Pemroses tersebut dapat berupa manusia, komputer, robot atau alat-alat elektronik lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau “mengeksekusi” algoritma yang menjabarkan proses tersebut.

Algoritma adalah deskripsi dari suatu pola tingkah laku yang dinyatakan secara primitif yaitu aksi-aksi yang didefenisikan sebelumnya dan diberi nama, dan diasumsikan sebelumnya bahwa aksi-aksi tersebut dapat kerjakan sehingga dapat menyebabkan kejadian.

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

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

**5.  Mekanisme Pelaksanaan Algoritma oleh Pemroses**

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

Kata “algoritma” dan “program” seringkali dipertukarkan dalam penggunaannya. Misalnya ada orang yang berkata seperti ini: “program pengurutan data menggunakan algoritma *selection sort*”. Atau pertanyaan seperti ini: “bagaimana algoritma dan program menggambarkan grafik tersebut?”. Jika Anda sudah memahami pengertian algoritma yang sudah disebutkan sebelum ini, Anda dapat membedakan arti kata algoritma dan program. Algoritma adalah langkah-langkah penyelesaikan masalah, sedangkan program adalah realisasi algoritma dalam bahasa pemrograman. Program ditulis dalam salah satu bahasa pemrograman dan kegiatan membuat program disebut**pemrograman**(*programming*). Orang yang menulis program disebut **pemrogram**(*programmer).*Tiap-tiap langkah di dalam program disebut **pernyataan**atau **instruksi.**Jadi, program tersusun atas sederetan instruksi. Bila suatu instruksi dilaksanakan, maka operasi-operasi yang bersesuaian dengan instruksi tersebut dikerjakan komputer.

Secara garis besar komputer tersusun atas empat komponen utama yaitu, piranti masukan, piranti keluaran, unit pemroses utama, dan memori. Unit pemroses utama (*Central Processing Unit – CPU)*adalah “otak” komputer, yang berfungsi mengerjakan operasi-operasi dasar seperti operasi perbandingan, operasi perhitungan, operasi membaca, dan operasi menulis. Memori adalah komponen yang berfungsi menyimpan atau mengingatingat.

Yang disimpan di dalam memori adalah program (berisi operasi-operasi yang akan dikerjakan oleh CPU) dan data atau informasi (sesuatu yang diolah oleh operasi-operasi). Piranti masukan dan keluaran (*I/O* *devices)*adalah alat yang memasukkan data atau program ke dalam memori, dan alat yang digunakan komputer untuk mengkomunikasikan hasil-hasil aktivitasnya. Contoh piranti masukan antara lain, papan kunci (*keyboard)*, pemindai (*scanner)*, dan cakram (*disk).*Contoh piranti keluaran adalah, layar peraga (*monitor)*, pencetak (*printer)*, dan cakram.

Mekanisme kerja keempat komponen di atas dapat dijelaskan sebagai berikut. Mula-mula program dimasukkan ke dalam memori komputer. Ketika program dilaksanakan (*execute)*, setiap instruksi yang telah tersimpan di dalam memori dikirim ke CPU. CPU mengerjakan operasioperasi yang bersesuaian dengan instruksi tersebut. Bila suatu operasi memerlukan data, data dibaca dari piranti masukan, disimpan di dalam memori lalu dikirim ke CPU untuk operasi yang memerlukannya tadi. Bila proses menghasilkan keluaran atau informasi, keluaran disimpan ke dalam memori, lalu memori menuliskan keluaran tadi ke piranti keluaran (misalnya dengan menampilkannya di layar monitor).

**6.  Belajar Memprogram dan Belajar Bahasa Pemrograman**

Belajar memprogram tidak sama dengan belajar bahasa pemrograman. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami. Sedangkan belajar bahasa pemrograman berarti belajar memakai suatu bahasa aturan-aturan tata bahasanya, pernyataan-pernyataannya, tata cara pengoperasian *compiler-*nya, dan memanfaatkan pernyataan-pernyataan tersebut untuk membuat program yang ditulis hanya dalam bahasa itu saja. Sampai saat ini terdapat puluhan bahasa pemrogram, antara lain bahasa rakitan (*assembly), Fortran, Cobol,* *Ada, PL/I, Algol, Pascal, C, C++, Basic, Prolog, LISP, PRG*, bahasabahasa simulasi *seperti CSMP, Simscript, GPSS, Dinamo*. Berdasarkan terapannya, bahasa pemrograman dapat digolongkan atas dua kelompok besar :

* + Bahasa pemrograman bertujuan khusus. Yang termasuk kelompok ini adalah *Cobol*(untuk terapan bisnis dan administrasi). *Fortran*(terapan komputasi ilmiah), bahasa rakitan (terapan pemrograman mesin), *Prolog*(terapan kecerdasan buatan), bahasa-bahasa simulasi, dan sebagainya.
  + Bahasa perograman bertujuan umum, yang dapat digunakan untuk berbagai aplikasi. Yang termasuk kelompok ini adalah bahasa *Pascal, Basic*dan *C*. Tentu saja pembagian ini tidak kaku. Bahasabahasabertujuan khusus tidak berarti tidak bisa digunakan untuk aplikasi lain. *Cobol*misalnya, dapat juga digunakan untuk terapan ilmiah, hanya saja kemampuannya terbatas. Yang jelas, bahasabahasa pemrograman yang berbeda dikembangkan untuk bermacam-macam terapan yang berbeda pula.

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

* + Bahasa tingkat rendah. Bahasa jenis ini dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (*translator*). Contohnya adalah bahasa mesin. CPU mengambil instruksi dari memori, langsung mengerti dan langsung mengerjakan operasinya. Bahasa tingkat rendah bersifat primitif, sangat sederhana, orientasinya lebih dekat ke mesin, dan sulit dipahami manusia. Sedangkan bahasa rakitan dimasukkan ke dalam kelompok ini karena alasan notasi yang dipakai dalam bahasa ini lebih dekat ke mesin, meskipun untuk melaksanakan instruksinya masih perlu penerjemahan ke dalam bahasa mesin.
  + Bahasa tingkat tinggi, yang membuat pemrograman lebih mudah dipahami, lebih “manusiawi”, dan berorientasi ke bahasa manusia (bahasa Inggris). Hanya saja, program dalam bahasa tingkat tinggi tidak dapat langsung dilaksanakan oleh komputer. Ia perlu diterjemahkan terlebih dahulu oleh sebuah *translator bahasa*(yang disebut kompilator atau *compiler)*ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU. Contoh bahasa tingkat tinggi adalah *Pascal, PL/I, Ada, Cobol, Basic, Fortran, C, C++,*dan sebagainya.

Bahasa pemrograman bisa juga dikelompokkan berdasarkan pada tujuan dan fungsinya. Di antaranya adalah :

**7.  Menilai Sebuah Algoritma**

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

* + Tingkat kepercayaannya tinggi (*realibility*). Hasil yang diperoleh dari proses harus berakurasi tinggi dan benar.
  + Pemrosesan yang efisien (*cost*rendah). Proses harus diselesaikan secepat mungkin dan frekuensi kalkulasi yang sependek mungkin.
  + Sifatnya general. Bukan sesuatu yang hanya untuk menyelesaikan satu kasus saja, tapi juga untuk kasus lain yang lebih general.
  + Bisa dikembangkan (*expandable*). Haruslah sesuatu yang dapat kita kembangkan lebih jauh berdasarkan perubahan *requirement*yang ada.
  + Mudah dimengerti. Siapapun yang melihat, dia akan bisa memahami algoritma Anda. Susah dimengertinya suatu program akan membuat susah di-*maintenance*(kelola).
  + Portabilitas yang tinggi (*portability*). Bisa dengan mudah diimplementasikan di berbagai *platform*komputer.
  + *Precise*(tepat, betul, teliti). Setiap instruksi harus ditulis dengan seksama dan tidak ada keragu-raguan, dengan demikian setiap instruksi harus dinyatakan secara eksplisit dan tidak ada bagian yang dihilangkan karena pemroses dianggap sudah mengerti. Setiap langkah harus jelas dan pasti.

Contoh :   Tambahkan 1 atau 2 pada x.

Instruksi di atas terdapat keraguan.

* + Jumlah langkah atau instruksi berhingga dan tertentu. Artinya, untuk kasus yang sama banyaknya, langkah harus tetap dan tertentu meskipun datanya berbeda.
  + Efektif. Tidak boleh ada instruksi yang tidak mungkin dikerjakan oleh pemroses yang akan menjalankannya.

Contoh :   Hitung akar 2 dengan presisi sempurna.

Instruksi di atas tidak efektif, agar efektif instruksi tersebut diubah.

Misal : Hitung akar 2 sampai lima digit di belakang koma.

* + Harus *terminate*. Jalannya algoritma harus ada kriteria berhenti. Pertanyaannya adalah apakah bila jumlah instruksinya berhingga maka pasti *terminate*?
  + *Output*yang dihasilkan tepat. Jika langkah-langkah algoritmanya logis dan diikuti dengan seksama maka dihasilkan *output*yang diinginkan.

Sedangkan kriteria Algoritma menurut Donald E. Knuth adalah :

* 1. Input: algoritma dapat memiliki nol atau lebih inputan dari luar.
  2. Output: algoritma harus memiliki minimal satu buah output keluaran.
  3. *Definiteness*(pasti): algoritma memiliki instruksi-instruksi yang jelas dan tidak ambigu.
  4. *Finiteness*(ada batas): algoritma harus memiliki titik berhenti (stopping role).
  5. *Effectiveness*(tepat dan efisien): algoritma sebisa mungkin harus dapat dilaksanakan dan efektif. Contoh instruksi yang tidak efektif adalah: A = A + 0 atau A = A \* 1

Namun ada beberapa program yang memang dirancang untuk unterminatable : contoh Sistem Operasi.

**8.  Penyajian Algoritma**

Penyajian algoritma secara garis besar bisa dalam 2 bentuk penyajian yaitu tulisan dan gambar. Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu (misalnya bahasa Indonesia atau bahasa Inggris) dan *pseudocode*. *Pseudocode*adalah kode yang mirip dengan kode pemrograman yang sebenarnya seperti Pascal, atau C, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram. Sedangkan algoritma disajikan dengan gambar, misalnya dengan *flowchart*. Secara umum, *pseudocode*mengekspresikan ide-ide secara informal dalam proses penyusunan algoritma. Salah satu cara untuk menghasilkan kode pseudo adalah dengan meregangkan aturan-aturan bahasa formal yang dengannya versi akhir dari algoritma akan diekspresikan. Pendekatan ini umumnya digunakan ketika bahasa pemrograman yang akan digunakan telah diketahui sejak awal.

*Flowchart*merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta pernyataannya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan antara proses digambarkan dengan garis penghubung. Dengan menggunakan *flowchart*akan memudahkan kita untuk melakukan pengecekan bagian-bagian yang terlupakan dalam analisis masalah. Di

samping itu *flowchart*juga berguna sebagai fasilitas untuk berkomunikasi antara pemrogram yang bekerja dalam tim suatu proyek.

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

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

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

**Kaidah-Kaidah Umum Pembuatan *Flowchart*Program**

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

* 1. *Input*,
  2. Proses pengolahan dan
  3. *Output*

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

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

Walaupun tidak ada kaidah-kaidah yang baku dalam penyusunan *flowchart*, namun ada beberapa anjuran :

* 1. Hindari pengulangan proses yang tidak perlu dan logika yang berbelit sehingga jalannya proses menjadi singkat.

Jalannya proses digambarkan dari atas ke bawah dan diberikan tanda panah untuk

* 1. memperjelas.
  2. Sebuah *flowchart*diawali dari satu titik START dan diakhiri dengan END.

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

Untuk memahami lebih dalam mengenai *flowchart*ini, akan diambil sebuah kasus sederhana.

***Kasus :***Buatlah sebuah rancangan program dengan menggunakan *flowchart*, mencari luas persegi panjang.

***Solusi :***Perumusan untuk mencari luas persegi panjang adalah :

***L*= *p . l***

di mana, ***L***adalah Luas persegi panjang, ***p***adalah panjang persegi, dan ***l*** adalah lebar persegi.