**Transkrip Apa itu Logika Pemrograman**

Istilah logika pemrograman sering kita dengar ketika masuk dunia programming. Jika Anda masih baru mulai mengenal pemrograman dan belum tahu apa itu logika pemrograman, tenang saja karena kita akan membahasnya di sini.

Menurut Technopedia, logika pemrograman adalah sebuah kemampuan dasar yang menerapkan pemahaman operasi logika terhadap data ke dalam ilmu komputer [[1]](https://www.techopedia.com/definition/3945/programming-logic). Tidak berhenti di sana, logika pemrograman juga dapat membantu Anda untuk menyelesaikan masalah (*problem solving*), baik dalam bidang pemrograman maupun kehidupan sehari-hari. Sangat bermanfaat sekali, bukan?

Ada 3 jenis logika pemrograman yang akan kita bahas, antara lain:

1. Logika aritmatika
2. Logika perbandingan
3. Logika perulangan

Oke, mari kita ilustrasikan manfaat belajar logika pemrograman dalam kehidupan sehari-hari dengan cerita sederhana. Bayangkan di awal bulan Anda pergi ke swalayan untuk membeli kebutuhan rumah tangga. Kemudian Anda melihat sebuah plakat bertuliskan “Khusus hari ini. Diskon 20% untuk semua jenis barang yang dijual.” Tentunya hal tersebut menarik perhatian Anda karena lumayan untuk menghemat pengeluaran belanja.

Kemudian Anda lihat plakatnya lebih dekat lagi untuk mengetahui syarat dan ketentuan diskon yang tertulis kecil di pojok kanan bawah. Di sana tertulis, “Diskon berlaku dengan minimum pembelian senilai 300 ribu rupiah.”  Nah, di sini Anda dapat menerapkan salah satu logika pemrograman yaitu perbandingan. Sehingga di benak Anda kemudian muncul pemikiran berikut, “Jika jumlah harga barang yang saya beli lebih dari atau sama dengan 300 ribu, maka saya berhak mendapat diskon 20%.” Setelah berpikir demikian Anda memutuskan untuk memborong barang-barang yang dijual dan memastikan totalnya sudah lebih dari atau sama dengan 300 ribu.

Nah, cerita di atas merupakan salah satu penerapan logika perbandingan dalam kehidupan sehari-hari. Penjabaran tentang jenis logika pemrograman selengkapnya akan dibahas di dalam materi-materi selanjutnya. Sudah tidak sabar belajar logika pemrograman, bukan? Yuk, kita mulai belajar.

**Perbedaan Logika dan Algoritma Pemrograman**

Dalam video pengantar sebelumnya Anda sudah mengetahui pengertian dari logika pemrograman beserta salah satu contoh penerapannya dalam kehidupan sehari-hari. Selanjutnya ada istilah lainnya yang serupa tapi tak sama dengan logika pemrograman yaitu algoritma. Apakah Anda pernah mendengar istilah algoritma sebelumnya?

Logika dan algoritma merupakan dua hal yang berbeda tetapi saling berkaitan satu sama lain. Jika sebelumnya logika pemrograman dapat membantu Anda untuk menyelesaikan permasalahan (problem solving), algoritma bertugas untuk membuat penyelesaiannya lebih terstruktur. Dengan begitu, alur penyelesaian masalah akan lebih tertata dan eksekusinya akan lebih mudah. Sehingga dapat disimpulkan algoritma pemrograman merupakan langkah-langkah penyelesaian masalah yang terangkai secara masuk akal, urut, dan sistematis.

Ketika menyusun alur instruksi yang acak, terdapat beberapa hal yang harus diperhatikan, seperti:

* **Finiteness (Keterbatasan)**  
  Algoritma yang berjalan dan memproses setiap langkah-langkahnya memiliki sifat terbatas, sehingga ia harus berhenti ketika semua langkah-langkahnya selesai dikerjakan. Jangan membuat algoritma yang panjang dan tidak ada akhirnya karena akan bertentangan dengan sifat finiteness ini.
* **Definiteness (Kepastian)**  
  Setiap langkah algoritma harus jelas, detail, dan tidak ambigu (makna ganda). Dengan kata lain, pembaca harus mengerti apa tujuan yang dimaksud. Misalnya Anda ingin membuat algoritma untuk menghitung luas persegi. Dari sana penggunanya harus tahu kalau yang diinputkan hanya berupa angka, bukan teks.
* **Effectiveness (Efektivitas)**  
  Setiap algoritma harus dibuat secara efektif. Langkah-langkah yang ada di dalamnya juga harus sesuai kebutuhan dan tidak perlu berlebihan. Dengan langkah yang efektif, waktu yang dibutuhkan pun akan lebih singkat atau masih dalam batas wajar.
* **Input (Masukan)**  
  Algoritma pasti membutuhkan nol atau lebih masukan (input) sebelum prosesnya dimulai. Misalnya algoritma untuk menghitung luas persegi panjang. Anda harus memasukkan dua input berupa angka untuk panjang dan lebarnya.
* **Output (Keluaran)**  
  Setiap input yang diproses oleh algoritma pasti memiliki satu atau lebih keluaran (output). Keluaran adalah besaran nilai yang memiliki hubungan dengan masukan (input). Keluaran harus berupa solusi atau penyelesaian dari suatu masalah. Jika sebuah algoritma tidak menemukan solusi, setidaknya harus menunjukkan pesan eror yang jelas. Misalnya dalam algoritma menghitung luas persegi panjang, pengguna melakukan kesalahan dengan memasukkan input berupa teks. Sehingga algoritma yang Anda buat harus bisa menangani hal tersebut dengan menampilkan pesan eror dan mengatakan inputnya harus berupa angka.

Masih ingat cerita diskon 20% yang ingin Anda dapatkan di supermarket pada materi sebelumnya? Sebelumnya mari kita segarkan kembali ingatan tentang cerita diskon 20% tersebut.

Ketika Anda melihat plakat yang bertuliskan diskon 20%, seketika jiwa shopping Anda bergejolak. Setelah membaca lebih lanjut ternyata syarat mendapat diskon adalah total belanja minimal 300 ribu untuk segala jenis barang. Di sini Anda berpikir, “Jika jumlah harga barang yang saya beli lebih dari atau sama dengan 300 ribu, maka saya berhak mendapat diskon 20%.” Nah, itulah yang dimaksud dengan penerapan logika pemrograman.

Kemudian Anda memutuskan untuk belanja kebutuhan rumah tangga sekaligus produk lainnya supaya mendapat diskon 20%. Nah, untuk mendapatkan diskon tersebut kita bisa menerapkan algoritma seperti berikut.

1. Anda harus masuk ke supermarketnya terlebih dahulu.
2. Ambil keranjang atau troli belanja.
3. Pilih barang-barang yang ingin Anda beli.
4. Lihat harga yang tercantum di setiap rak tempat Anda mengambil barang.
5. Setelah semua barang sudah masuk ke dalam troli dan totalnya masih belum mencapai 300 ribu, cari barang lainnya.
6. Ketika total harga barang yang Anda beli sudah mencapai 300 ribu atau lebih dan dirasa cukup, berjalanlah menuju meja kasir.
7. Tunggu kasir menghitung kembali semua barang belanjaan Anda.
8. Keluarkan dompet Anda dan bayarkan sesuai total harga belanjaan yang sudah dihitung oleh kasir.
9. Periksa kembali struk yang diberikan oleh kasir dan pastikan sudah mendapat diskon 20%.
10. Ketika pembayaran selesai, jangan lupa ambil belanjaan Anda.
11. Keluar supermarket dan selesai.

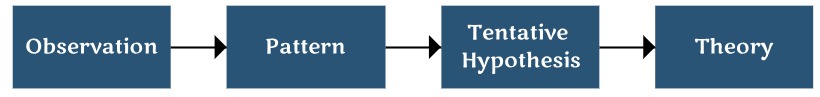
Nah, contoh urutan langkah di atas itulah yang dinamakan algoritma untuk menyelesaikan sebuah permasalahan. Sama halnya jika diterapkan pada ilmu pemrograman. Ketika Anda sudah memikirkan bagaimana logika berjalannya suatu program, tulislah dalam bentuk algoritma atau kode yang berurutan untuk mewujudkannya.

## Penalaran Logika Induktif dan Deduktif

Setelah mengetahui tentang perbedaan logika dan algoritma pemrograman, selanjutnya kita akan membahas penalaran logika induktif dan deduktif. Dua metode ini dapat kita terapkan dalam logika pemrograman.

Penasaran tentang metode induktif dan deduktif? Yuk, kita simak uraiannya di bawah ini.

### Logika Induktif



Ketika memutuskan suatu hal, Anda biasanya mengandalkan pengamatan dan menghubungkannya dengan pengalaman yang pernah terjadi. Misalnya, Pada suatu sore, Anda ingin membeli nasi goreng menggunakan sepeda motor karena warungnya lumayan jauh. Kemudian langit mulai mendung dan angin bertiup dingin. Karena akhir-akhir ini sering turun hujan terutama ketika sore hari, Anda langsung memasukkan jas hujan ke dalam jok motor dan berangkat ke warung. Nah, di sana Anda sudah menerapkan logika penalaran induktif.

Apakah Anda sudah mengetahui apa itu penalaran induktif berdasarkan cerita di atas? Menurut Stanford Encyclopedia of Philosophy, penalaran logika induktif adalah logika pendukung pembuktian [[2]](https://plato.stanford.edu/entries/logic-inductive/). Jenis penalaran ini melibatkan pembentukan generalisasi berdasarkan pengalaman, pengamatan, dan fakta. Secara sederhana, penalaran induktif merupakan metode berpikir menggunakan pengamatan dan digabungkan dengan pengalaman yang sudah Anda ketahui kebenarannya untuk mendapatkan sebuah kesimpulan.

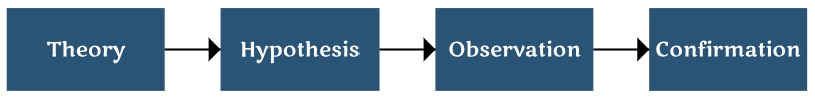
Proses penalaran induktif dimulai dari,

1. Pengamatan terhadap suatu peristiwa atau hal lainnya.
2. Mendeteksi pola dan keteraturan dari pengalaman yang telah terjadi.
3. Merumuskan beberapa hipotesis sementara yang dapat kita eksplorasi.
4. Mengambil kesimpulan.

Jika dihubungkan dengan cerita sebelumnya, proses pengamatan terjadi ketika Anda melihat langit yang mulai mendung dan angin yang bertiup dingin. Kemudian Anda mulai menghubungkannya dengan pengalaman yang telah terjadi yaitu akhir-akhir ini pada waktu sore sering turun hujan. Kemudian Anda merumuskan hipotesis “bisa saja nanti di tengah jalan tiba-tiba turun hujan” yang membawa pada kesimpulan yaitu harus membawa jas hujan.

Nah, sampai di sini kita telah tahu apa itu logika penalaran induktif. Lalu, bagaimana dengan logika penalaran deduktif? Tenang saja, kita akan membahasnya di materi berikutnya. Let’s go.

### Logika Deduktif



Menurut Chad Nilep dari Universitas Nagoya, logika penalaran deduktif adalah sistem yang tepat dan teratur bertujuan untuk memberikan dukungan pasti untuk suatu kesimpulan [[3]](http://www.ilas.nagoya-u.ac.jp/~nilep/deductive.html). Salah satu bentuk penalaran logika deduktif adalah silogisme. Apakah Anda pernah mendengar istilah silogisme sebelumnya?

Menurut KBBI, silogisme adalah bentuk, cara berpikir atau menarik simpulan yang terdiri atas premis umum, premis khusus, dan simpulan (misalnya semua manusia akan mati, si A manusia, jadi si A akan mati) [4]. Sehingga bisa dikatakan jika A = B dan B = C, maka hasil kesimpulan dari penalaran logikanya adalah A = C. Masih bingung? Perhatikan beberapa contoh silogisme berikut,

1. Semua burung memiliki bulu
2. Semua kenari adalah burung

**Kesimpulannya:** Semua kenari memiliki bulu

1. Semua hewan berbisa berbahaya.
2. Beberapa ular adalah hewan berbisa

**Kesimpulannya:** beberapa ular adalah hewan berbahaya

Selama dua premis didasarkan pada informasi yang akurat dan valid, kesimpulan yang didapatkan seringkali benar.

Proses penalaran deduktif terhadap sebuah informasi dimulai dari,

1. Premis pertama yang berisi pernyataan umum. Jika premis pertama itu valid, seharusnya juga valid untuk segala kasus yang berhubungan dengan premis tersebut.
2. Premis kedua dibuat berdasarkan premis pertama (lebih khusus). Jika premis pertama sudah valid, maka pernyataan dalam premis kedua ini harusnya juga valid.
3. Kedua premis tersebut diuji dalam berbagai skenario.
4. Menarik kesimpulan berdasarkan hasil pengujian untuk menetapkan sebuah informasi yang diuji valid atau tidak.

Mari kita hubungkan dengan salah satu contoh silogisme tentang beberapa ular adalah hewan berbahaya. Di sana premis pertama mengatakan semua hewan berbisa berbahaya. Pernyataan itu valid karena hewan berbisa dapat mengancam makhluk hidup lainnya. Kemudian premis kedua tertulis beberapa ular adalah hewan berbisa. Premis kedua juga sudah tepat karena tidak semua ular berbisa.

Kemudian kedua premis tersebut diuji dalam beberapa skenario berdasarkan pengamatan atau kondisi yang sebenarnya. Terdapat beberapa skenario misalnya sebagai berikut,

* Semua ular adalah hewan berbahaya
* Beberapa ular adalah hewan berbahaya

Untuk skenario pertama kurang tepat karena tidak semua ular berbahaya karena ada juga ular yang tidak berbisa. Sedangkan untuk skenario kedua sudah tepat yaitu beberapa ular adalah hewan berbahaya. Hal tersebut valid karena didukung oleh premis kedua yang mengatakan beberapa ular adalah hewan berbisa (tidak semua ular berbisa). Sehingga kesimpulannya beberapa ular adalah hewan berbahaya.

## Logika Pemrograman Dasar

Sebelumnya kita telah mengetahui perbedaan antara algoritma dengan logika pemrograman serta bentuk penalaran logika. Dalam bidang pemrograman, pasti Anda akan menemui sebuah kasus yang harus menerapkan logika untuk menyelesaikan sebuah permasalahan. Tahukah Anda apa saja logika pemrograman dasar tersebut? Simak uraiannya di bawah ini:

### Logika Aritmatika

Dalam ilmu pemrograman memang tidak lepas dari perhitungan matematika, seperti menerapkan operator atau operasi matematika. Sekali dua kali pasti kita akan menerapkan perhitungan dalam menyelesaikan permasalahan dalam pemrograman.

Maka dari itu Anda perlu mempelajari logika aritmatika untuk membantu pemecahan masalah terkait dengan perhitungan yang melibatkan operasi matematika seperti penjumlahan, pengurangan, dan lainnya. Untuk melakukan operasi tersebut, pasti Anda memerlukan operator yang berfungsi untuk memproses operasi matematika. Perhatikan uraian operator matematika yang dapat Anda gunakan dalam proses pemrograman.

| **Operator** | **Keterangan** |
| --- | --- |
| + | Penjumlahan |
| - | Pengurangan |
| \* | Perkalian |
| / | Pembagian |
| % | Sisa bagi atau modulus |

Nah, secara sederhana ketika kita menulis sebuah program akan familiar dengan beberapa operasi matematika dan menggunakan operator-operator seperti tabel di atas. Operasi matematika tersebut termasuk dalam logika pemrograman dasar aritmatika yang harus Anda miliki sebagai seseorang yang bekerja di bidang pemrograman.

### Logika Perbandingan

Dari namanya mungkin Anda sudah mengetahui tentang logika perbandingan. Logika ini digunakan untuk membandingkan dua hal yang mempunyai nilai. Nilai inilah yang akan dibandingkan dengan nilai lain atau kondisi yang diinginkan.

Mari kita contohkan dengan kasus yang biasa terjadi pada kehidupan sehari-hari. Pada sebuah perempatan jalan Anda melihat kondisi lampu lalu lintas yang masih merah. Kemudian Anda berpikir, “Jika lampu menyala hijau, maka saya harus jalan. Namun, jika masih merah, harus menunggu dulu” Seketika Anda langsung membandingkan kondisi lampu saat itu masih bewarna merah. Sehingga Anda memutuskan untuk menunggu hingga warna hijau. Kemudian 20 detik kemudian warna lampunya berubah menjadi hijau. Nah, saat itulah Anda mulai memacu kendaraan Anda.

Masih bingung? Tenang saja, mari kita contohkan dengan kasus berikutnya. Ketika bermain game petualangan, Anda ingin mengalahkan highscore yang sudah dicapai sebelumnya. Katakan highscore sebelumnya bernilai 500. Singkat cerita game telah selesai dan Anda mendapatkan skor 650. Dengan kondisi tersebut apakah Anda telah mengalahkan highscore sebelumnya?

Untuk menjawab pertanyaan tersebut Anda dapat membandingkan skor yang Anda dapatkan dengan highscore yang sudah ada. Berarti logika perbandingannya seperti berikut,

“Jika skor saya lebih besar dari highscore yang sudah ada, maka saya berhasil.”

“Namun, jika skor saya kurang dari highscore yang sudah ada, maka saya gagal.”

Berdasarkan logika perbandingan di atas Anda dapat menyimpulkan bahwa telah berhasil mendapat highscore. Mengapa demikian? Karena Anda mendapat skor 650 yang lebih besar dibandingkan dengan highscore sebelumnya yaitu 500.

Bagaimana menurut Anda? Penerapan logika perbandingan sangat mudah dan menyenangkan, bukan?

### Logika Perulangan

Selanjutnya kita akan membahas logika perulangan. Logika perulangan adalah proses pengulangan instruksi yang dilakukan dalam kondisi yang telah ditentukan. Dalam pemrograman, kita harus menghindari menulis kode yang sama berulang kali. Hal tersebut dilakukan supaya proses menulis kode lebih efektif dan efisien. Maka dari itu kita dapat menerapkan logika perulangan.

Contoh sederhana perulangan juga dapat kita temui di dunia nyata. Pernahkah sewaktu masa sekolah dulu Anda dihukum untuk menulis “Saya tidak akan mengulangi lagi” di papan tulis sebanyak 100 kali? Contoh tersebut jika ditulis dalam bahasa pemrograman sama seperti kita menulis statement print (“Saya tidak akan mengulangi lagi”) sebanyak 100 baris. Sangat melelahkan, bukan?

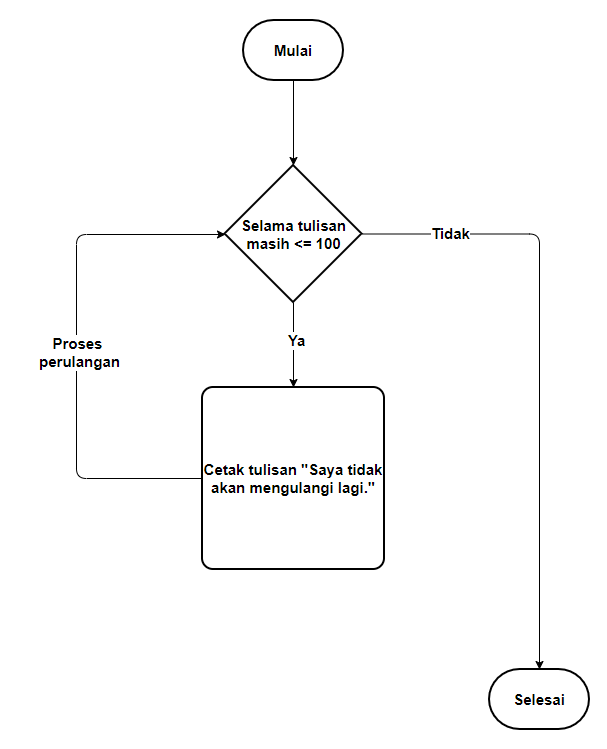
Anda dapat menggunakan logika perulangan untuk menyelesaikan permasalahan di atas. Bagian-bagian yang harus ada untuk melakukan perulangan adalah

* Kondisi yang harus terpenuhi supaya proses perulangan bisa berjalan.
* Instruksi yang harus dilakukan secara berulang.
* Sebuah variabel yang menghitung posisi perulangan saat itu.

Nah, jika dimasukkan dengan contoh kasus menulis kalimat “Saya tidak akan mengulangi lagi” selama 100 kali, maka Anda dapat membuat kondisi perulangan sebagai berikut.

“Selama tulisan yang ada kurang dari atau sama dengan 100, maka lakukan perulangan untuk mencetak instruksi “Saya tidak akan mengulangi lagi”. Setiap kali tulisan tercetak akan dihitung oleh sebuah variabel khusus, misalnya variabel counter.

Ketika variabel counter yang menyimpan posisi perulangan sudah mencapai seratus, kondisi perulangan akan berhenti. Untuk lebih jelasnya lihatlah diagram alur di bawah ini.



Itulah logika perulangan yang dapat kita terapkan di bidang pemrograman. Selain itu Anda juga dapat menggabungkan logika-logika yang sebelumnya kita bahas. Misalnya menggabungkan logika perulangan dengan perbandingan, logika perbandingan dengan aritmatika, dsb.

## Rangkuman dari Pendahuluan

Anda telah belajar banyak hal di kelas modul ini. Simak rangkuman materi yang telah dibahas di modul pendahuluan berikut ini.

Menurut Technopedia, logika pemrograman adalah sebuah kemampuan dasar yang menerapkan pemahaman operasi logika terhadap data ke dalam ilmu komputer [[1]](https://www.techopedia.com/definition/3945/programming-logic). Tidak berhenti di sana, logika pemrograman juga dapat membantu Anda untuk menyelesaikan masalah (problem solving), baik dalam bidang pemrograman maupun kehidupan sehari-hari.

Logika dan algoritma merupakan dua hal yang berbeda tetapi saling berkaitan satu sama lain. Jika sebelumnya logika pemrograman dapat membantu Anda untuk menyelesaikan permasalahan (problem solving), algoritma bertugas untuk membuat penyelesaiannya lebih terstruktur. Ketika menyusun algoritma, terdapat beberapa hal yang harus diperhatikan, seperti:

* **Finiteness (Keterbatasan)**  
  Algoritma yang berjalan dan memproses setiap langkah-langkahnya memiliki sifat terbatas, sehingga ia harus berhenti ketika semua langkah-langkahnya selesai dikerjakan.
* **Definiteness (Kepastian)**  
  Setiap langkah algoritma harus jelas, detail, dan tidak ambigu (makna ganda). Dengan kata lain, pembaca harus mengerti apa tujuan yang dimaksud.
* **Effectiveness (Efektifitas)**  
  Setiap algoritma harus dibuat secara efektif. Langkah-langkah yang ada di dalamnya juga harus sesuai kebutuhan dan tidak perlu berlebihan.
* **Input (Masukan)**  
  Algoritma pasti membutuhkan nol atau lebih masukan (input) sebelum prosesnya dimulai.
* **Output (Keluaran)**  
  Setiap input yang diproses oleh algoritma pasti memiliki satu atau lebih keluaran (output). Keluaran adalah besaran nilai yang memiliki hubungan dengan masukan (input).

Dua metode penalaran logika yang dapat diterapkan di bidang pemrograman ada 2 yaitu,

* **Induktif**  
  Menurut Stanford Encyclopedia of Philosophy, penalaran logika induktif adalah logika pendukung pembuktian [[2]](https://plato.stanford.edu/entries/logic-inductive/). Secara sederhana metode induktif adalah sebuah metode berpikir menggunakan pengamatan dan digabungkan dengan pengalaman yang sudah Anda ketahui kebenarannya untuk mendapatkan sebuah kesimpulan.
* **Deduktif**  
  Menurut Chad Nilep dari Universitas Nagoya, logika penalaran deduktif adalah sistem yang tepat dan teratur bertujuan untuk memberikan dukungan pasti untuk suatu kesimpulan [[3]](http://www.ilas.nagoya-u.ac.jp/~nilep/deductive.html). Salah satu bentuk penalaran logika deduktif adalah silogisme.  
  Menurut KBBI, silogisme adalah bentuk, cara berpikir atau menarik simpulan yang terdiri atas premis umum, premis khusus, dan simpulan [[4]](https://kbbi.kemdikbud.go.id/entri/silogisme).

Ada 3 jenis logika pemrograman yang akan kita bahas, antara lain:

1. **Logika aritmatika**  
   Logika aritmatika dapat membantu pemecahan masalah terkait dengan perhitungan yang melibatkan operasi matematika seperti penjumlahan, pengurangan, dan lainnya.
2. **Logika perbandingan**  
   Logika ini digunakan untuk membandingkan dua hal yang mempunyai nilai. Nilai inilah yang akan dibandingkan dengan nilai lain atau kondisi yang diinginkan.
3. **Logika perulangan**  
   Logika perulangan adalah proses pengulangan instruksi yang dilakukan dalam kondisi yang telah ditentukan. Dalam pemrograman, kita harus menghindari menulis kode yang sama berulang kali. Hal tersebut dilakukan supaya proses menulis kode lebih efektif dan efisien.

**Transkrip Apa itu Gerbang Logika**

Kita telah belajar banyak hal mulai dari pengertian logika pemrograman hingga macam-macam logika pemrograman dasar. Sekarang kita akan lanjut membahas gerbang logika. Pernah mendengar istilah gerbang logika sebelumnya?

Gerbang logika atau logic gate adalah sebuah rangkaian sirkuit digital yang berguna untuk memproses logika boolean (logika yang menghasilkan output benar atau salah). Gerbang logika ini menghasilkan output berupa 0 (salah) dan 1 (benar) berdasarkan input (masukan) yang kita berikan.

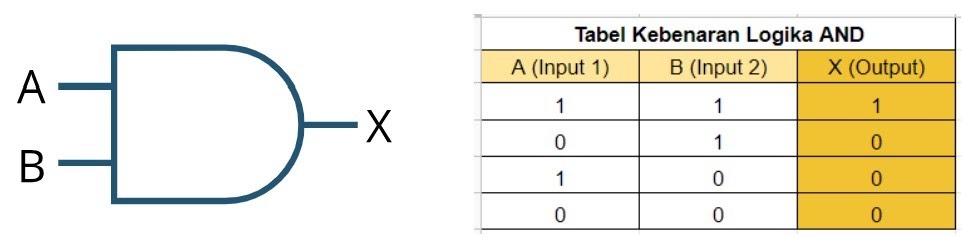
Istilah gerbang logika biasanya dihubungkan dengan sistim elektronika digital yang berhubungan dengan komponen IC (Integrated Circuit), Transistor, dan kawan-kawannya. Tenang dulu, kita tidak akan membahas elektronika di sini. Kita akan belajar tentang gerbang logika dari sisi logika boolean dalam bidang pemrogramannya.

Terdapat 7 jenis gerbang logika yang akan kita bahas antara lain:

1. AND
2. OR
3. NOT
4. NAND
5. NOR
6. XOR
7. XNOR

Wah, lumayan banyak juga jenis-jenisnya.

Gerbang logika biasanya digambarkan seperti diagram sirkuit atau bisa juga dalam bentuk tabel kebenaran. Apa itu tabel kebenaran kebenaran? Tabel kebenaran merupakan kumpulan kombinasi beberapa nilai input berdasarkan logika yang kita buat untuk menghasilkan output (keluaran) yang masuk akal. Salah satu contoh gerbang logika dan tabel kebenaran digambarkan sebagai berikut.

Dalam gerbang logika AND, jika input A dan input B bernilai 1 (True) maka output X yang dihasilkan adalah 1 (True). Jika salah satu inputnya ada yang bernilai 0 (False) maka hasilnya adalah 0 (False). Masih bingung? Berikut contoh implementasi logika AND yang dapat Anda temui di kehidupan sehari-hari.

Anda akan mendapat diskon sebesar 20% jika memiliki kartu pengguna (member card) dan minimal belanja sebesar 100 ribu. Sehingga Anda harus memenuhi dua persyaratan tersebut untuk mendapat diskon. Jika Anda hanya memiliki member card tetapi total belanja masih di bawah 100 ribu, maka Anda tidak akan mendapat diskon. Begitu pun sebaliknya.

Nah, itu tadi adalah salah satu contoh gerbang logika AND. Untuk mengetahui jenis-jenis gerbang logika yang lain, dapat Anda simak uraian lengkapnya di modul ini. Yuk, kita mulai belajar Gerbang Logika.

## Gerbang Logika AND

Gerbang Logika AND sempat kita singgung di materi video sebelumnya. Namun, kali ini kita membahas lebih detail mengenai gerbang logika AND.

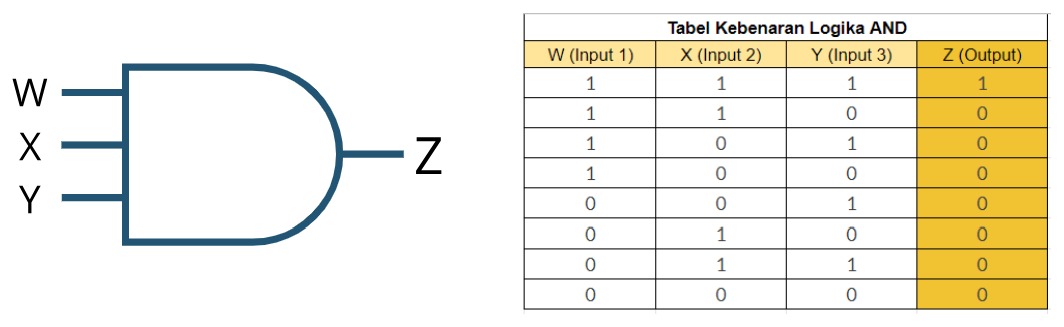
Berdasarkan simbol di atas, A dan B merupakan masukan (input) sedangkan X adalah keluaran (output). Gerbang logika AND setidaknya harus mempunyai dua masukan untuk mendapatkan satu keluaran. Jika ingin menghasilkan keluaran bernilai 1 (True), maka kedua masukan masing-masing harus bernilai 1 (True). Syarat tersebut harus terpenuhi karena jika salah satu saja bernilai 0 (False), hasilnya akan bernilai 0 (False).

Logika AND dilambangkan dengan perkalian (multiply) yang menggunakan tanda titik (“.”) atau tanpa titik karena keduanya sudah menunjukkan perkalian. Sehingga logika AND bisa diekspresikan sebagai X = A.B atau bisa juga ditulis tanpa titik X = AB.

Jika logika AND digambarkan dalam tabel kebenaran akan terlihat sebagai berikut.

Tabel kebenaran berfungsi untuk menunjukkan setiap kemungkinan benar salahnya suatu keluaran (output) yang didapatkan dari kombinasi input yang dimasukkan. Sama seperti gerbang logika, tabel kebenaran berisi angka 0 untuk menunjukkan kondisi false dan 1 untuk kondisi true.

Lalu, bisakah kita membuat 3 input dalam gerbang logika AND? Jawabannya adalah bisa. Anggap saja kita ingin menambahkan satu input dari contoh sebelumnya menjadi A, B, dan C serta 1 keluaran X. Jika digambarkan dalam gerbang logika dan tabel kebenaran menjadi sebagai berikut.



Nah, gambar di atas menunjukkan gerbang logika AND dengan 3 input dan 1 output. Sedangkan jumlah nilai yang ada di tabel kebenaran terlihat lebih banyak daripada logika AND dengan 2 input dan 1 output. Hal tersebut terjadi karena kemungkinan kombinasi yang diberikan akan lebih beragam. Tetapi kembali lagi sesuai prinsip logika AND, untuk menghasilkan output bernilai 1 (true) maka semua inputnya juga harus bernilai 1 (true). Sehingga logika AND dapat dibaca seperti berikut, “Jika input A, B, dan C bernilai 1 (true), maka hasilnya 1 (true). Salah satu saja ada input yang bernilai 0 (false) hasilnya pasti juga 0 (false).”

## Gerbang Logika OR

Setelah belajar tentang gerbang logika AND, kita akan lanjut belajar jenis gerbang logika berikutnya yaitu OR.

Berdasarkan simbol di atas, A dan B merupakan masukan (input) sedangkan X adalah keluaran (output). Gerbang logika OR setidaknya harus mempunyai dua masukan untuk mendapatkan satu keluaran. Berbeda dengan logika AND, pada logika OR jika salah satu input saja bernilai 1 (true), maka outputnya akan bernilai 1 (true).

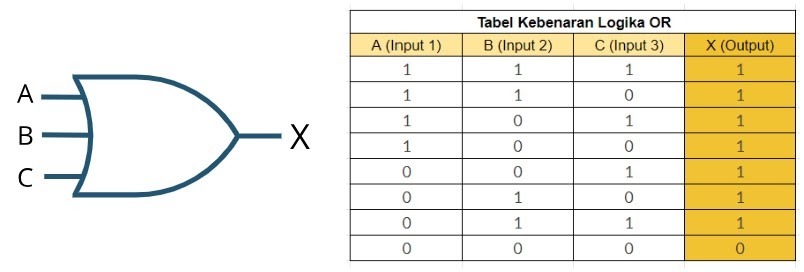
Logika OR dilambangkan dengan penjumlahan (plus) yang menggunakan tanda plus (“+”). Sehingga logika OR bisa diekspresikan sebagai X = A + B.

Jika logika OR digambarkan dalam gerbang logika dan tabel kebenaran akan terlihat sebagai berikut.

Tabel kebenaran di atas menunjukkan kemungkinan benar salahnya suatu keluaran (output) yang didapatkan dari kombinasi input yang dimasukkan. Sama seperti gerbang logika, tabel kebenaran berisi angka 0 untuk menunjukkan kondisi false dan 1 untuk kondisi true.

Dalam tabel kebenaran logika OR di atas terdapat nilai A = 0 dan B = 1 menghasilkan output X = 1. Mengapa demikian? Kembali lagi ke prinsip logika OR, jika salah satu input saja ada yang  bernilai 1 (true), maka hasil outputnya adalah 1 (true). Sedangkan jika semua inputnya bernilai 0 (false), maka hasilnya juga tetap 0 (false).

Lalu, bisakah kita membuat 3 input dalam gerbang logika OR? Jawabannya adalah bisa. Anggap saja kita ingin membuat 3 input yaitu A, B, dan C serta 1 keluaran X. Jika digambarkan dalam gerbang logika dan tabel kebenaran menjadi sebagai berikut.



Gambar di atas menunjukkan gerbang logika OR dengan 3 input dan 1 output. Jumlah kombinasi input yang ada di tabel kebenaran terlihat lebih banyak daripada logika OR yang memiliki 2 input dan 1 output. Dalam logika OR, jika salah satu input saja sudah bernilai 1 (true), maka outputnya pasti bernilai 1 (true). Sehingga logika OR dapat dibaca seperti berikut, “Jika input A atau B atau C bernilai 1 (true), maka hasilnya 1 (true).” Sedangkan jika semua input 0 (false), maka hasilnya juga akan 0 (false).

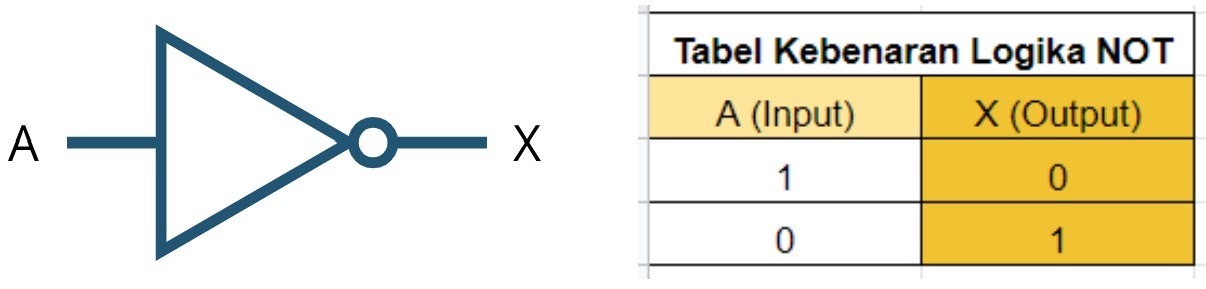
## Gerbang Logika NOT

Selanjutnya kita akan membahas gerbang logika berikutnya yaitu NOT.

Berbeda dengan dua gerbang logika sebelumnya, gerbang logika NOT hanya memerlukan 1 input untuk mendapatkan satu keluaran. Gerbang logika NOT akan menghasilkan keluaran yang berlawanan dari input. Logika NOT dilambangkan dengan tanda overline  di atas inputnya seperti berikut

Sehingga jika logika NOT dibaca menjadi seperti berikut, “Jika input W bernilai 1 (true), maka hasilnya 0 (false). Sebaliknya, jika inputnya bernilai 0 (false), maka outputnya bernilai 1 (true).”

Jika logika NOT digambarkan dalam gerbang logika dan tabel kebenaran akan terlihat sebagai berikut.



Pada gerbang logika NOT terdapat simbol lingkaran setelah segitiga yang mengarah ke output. Setiap gerbang logika yang ditambahkan lingkaran sebelum output berarti telah digabungkan dengan logika NOT, seperti NOR, NAND, dan X-NOR yang akan kita bahas di materi berikutnya.

## Gerbang Logika NAND

Masih ingat dengan gerbang logika AND yang sebelumnya telah kita bahas? Nah, kali ini kita akan membahas kebalikan atau negasi dari AND yaitu NAND.

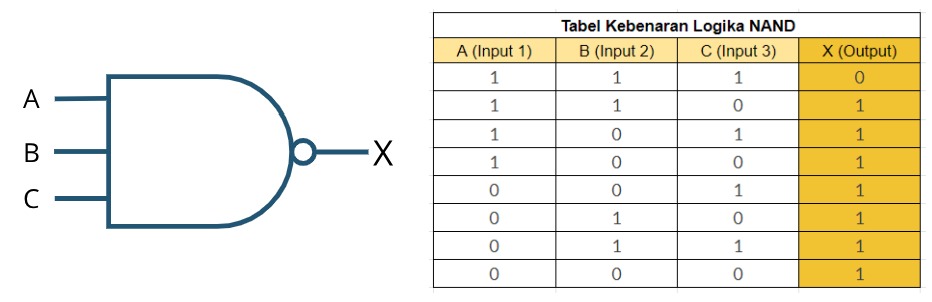
Gerbang logika NAND merupakan gabungan dari logika NOT ditambah logika AND yang menghasilkan output yang berlawanan dengan gerbang logika AND. Maka dari itu ia disebut NAND atau NOT AND. Gerbang logika NAND setidaknya harus memiliki dua masukan untuk mendapatkan satu keluaran.

Logika NAND dilambangkan dengan perkalian (multiply) yang menggunakan tanda titik (“.”) atau tanpa titik dengan tambahan tanda overline   untuk menandakan negasi. Sehingga logika NAND bisa diekspresikan sebagai

Jika logika NAND digambarkan dalam tabel kebenaran akan terlihat sebagai berikut.

Dalam tabel kebenaran logika NAND di atas terdapat nilai A = 1 dan B = 1 menghasilkan output X = 0. Mengapa demikian? Karena ini logika NOT AND, maka hasilnya adalah negasi atau kebalikan dari logika AND. Gerbang logika NAND akan menghasilkan keluaran 0 (false) jika semua masukannya bernilai 1 (true). Sedangkan jika salah satu masukannya ada yang bernilai 0 (false), maka hasil outputnya menjadi 1 (true).

Lalu, bisakah kita membuat 3 input dalam gerbang logika NAND? Jawabannya adalah bisa. Anggap saja kita ingin menambahkan satu input dari contoh sebelumnya menjadi A, B, dan C serta 1 keluaran X. Jika digambarkan dalam gerbang logika dan tabel kebenaran menjadi sebagai berikut.



Nah, gambar di atas menunjukkan gerbang logika NAND dengan 3 input dan 1 output. Jumlah kombinasi input di tabel kebenaran terlihat lebih banyak daripada logika NAND dengan 2 input dan 1 output. Untuk menghasilkan output bernilai 1 (true) maka semua inputnya juga harus bernilai 1 (true). Sehingga jika logika NAND dibaca menjadi seperti berikut, “Jika input A, B, dan C bernilai 1 (true), maka hasilnya adalah negasi dari logika AND yaitu 0 (false). Salah satu saja ada input yang bernilai 0 (false) hasilnya 1 (true).”

## Gerbang Logika NOR

Masih ingat dengan gerbang logika OR yang sebelumnya telah kita bahas? Nah, kali ini kita akan membahas kebalikan atau negasi dari OR yaitu NOR.

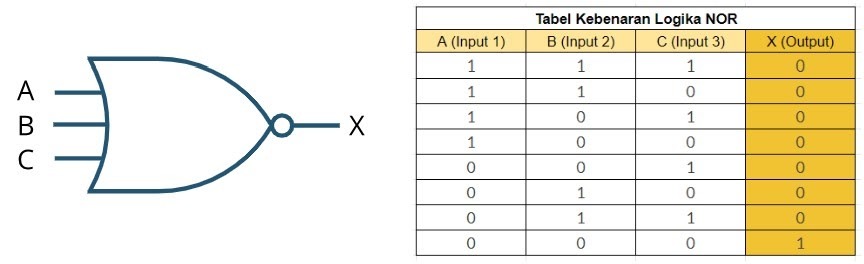
Gerbang logika NOR merupakan gabungan dari logika NOT dan logika OR yang menghasilkan output berlawanan dengan gerbang logika OR. Maka dari itu ia disebut NOR atau NOT OR. Gerbang logika NOR setidaknya harus memiliki dua masukan untuk mendapatkan satu keluaran.

Logika NOR dilambangkan dengan penjumlahan yang menggunakan **tanda plus (“+”)** dengan tambahan tanda overline  untuk menandakan negasi. Sehingga logika NOR bisa diekspresikan sebagai

Jika logika NOR digambarkan dalam gerbang logika dan tabel kebenaran akan terlihat sebagai berikut.

Dalam tabel kebenaran logika NOR di atas terdapat nilai A = 0 dan B = 1 menghasilkan output 0. Mengapa demikian? Kembali lagi ke prinsip logika NOR yang merupakan kebalikan dari OR, jika salah satu input saja ada yang bernilai 1 (true), maka hasil outputnya adalah 0 (false). Sedangkan jika semua inputnya bernilai 0 (false), maka hasilnya adalah 1 (true).

Lalu, bisakah kita membuat 3 input dalam gerbang logika NOR? Jawabannya adalah bisa. Anggap saja kita ingin membuat 3 input yaitu A, B, dan C serta 1 keluaran X. Jika digambarkan dalam gerbang logika dan tabel kebenaran menjadi sebagai berikut.



Gambar di atas menunjukkan gerbang logika NOR dengan 3 input dan 1 output. Jumlah kombinasi input yang ada di tabel kebenaran terlihat lebih banyak daripada logika OR yang memiliki 2 input dan 1 output. Dalam logika NOR yang menggunakan 3 input, jika salah satu input saja sudah bernilai 1 (true), maka outputnya pasti bernilai 0 (false). Sehingga jika logika NOR dibaca menjadi seperti berikut, “Jika input A atau B atau C bernilai 1 (true), maka hasilnya negasi dari logika OR yaitu 0 (false). Sedangkan jika semua input 0 (false), maka hasilnya adalah 1 (true).”

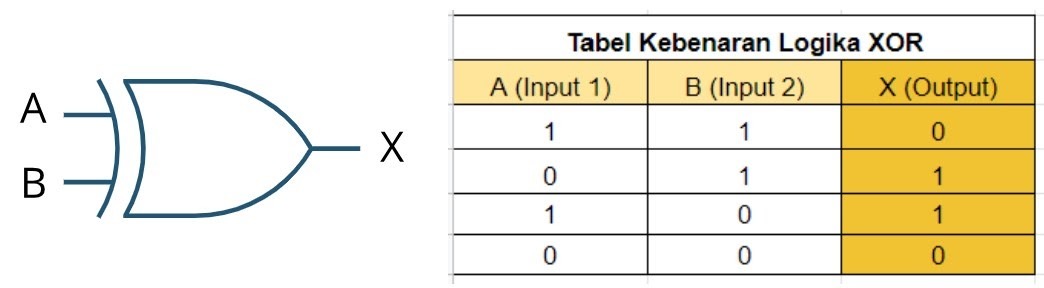
## Gerbang Logika XOR

XOR singkatan dari Exclusive OR. Jika sebelumnya dalam logika OR kita mengetahui jika salah satu input saja bernilai 1 (true), maka outputnya pasti bernilai 1 (true). Lain halnya dengan XOR yang hanya menghasilkan keluaran 1 (true) ketika kedua inputnya berbeda. Masih bingung? Oke, mari kita bahas perbedaannya di bawah ini.

Nah, dari dua gambar gerbang logika di atas kita bisa melihat perbedaan dari XOR dan OR. Gerbang logika XOR memiliki input A=1, B=1, dan menghasilkan keluaran X = 0. Hal tersebut karena logika XOR hanya akan menghasilkan nilai keluaran 1 (true) jika nilai masukannya berbeda. Sedangkan jika nilai yang dimasukkan sama akan menghasilkan nilai 0 (false).

Kemudian gerbang logika OR akan menghasilkan nilai keluaran 1 (true) jika salah satu inputnya ada yang memiliki nilai 1 (true). Walaupun inputnya A dan B sama-sama bernilai 1 akan tetap menghasilkan keluaran 1 (true). Sedangkan jika kedua inputnya 0 (false), maka hasil keluarannya juga akan 0 (false). Itulah perbedaan dari logika XOR dan OR.

Jika logika XOR digambarkan dalam gerbang logika dan tabel kebenaran akan terlihat sebagai berikut.



Gambar di atas sama seperti yang telah disinggung sebelumnya. Dalam XOR, hanya masukan yang nilainya berbeda saja yang akan menghasilkan keluaran bernilai 1 (true). Jika nilai inputnya sama, maka hasil keluarannya otomatis menjadi 0 (false). Itulah mengapa gerbang logika ini disebut Exclusive OR dilambangkan dengan tanda plus yang berada di dalam lingkaran. Sehingga logika XOR bisa diekspresikan seperti di bawah ini.

atau jika ekspresi tersebut disederhanakan menjadi seperti berikut.X =A(atas)B + AB(atas).

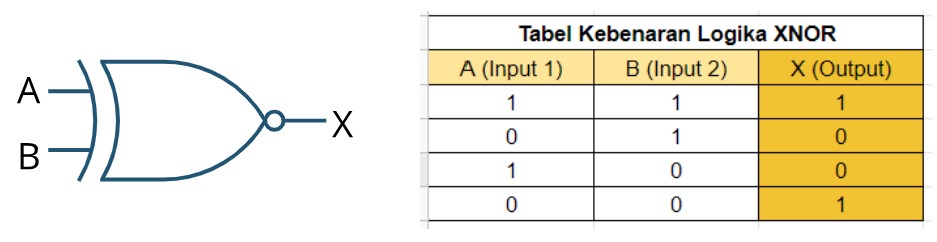
## Gerbang Logika XNOR

Setelah mengetahui berbagai jenis gerbang logika mulai dari AND hingga XOR, kita akan lanjut ke gerbang logika terakhir yaitu XNOR yang merupakan singkatan dari Exclusive NOR. Jika sebelumnya dalam logika NOR kita mengetahui jika salah satu input ada yang bernilai 1 (true), maka outputnya pasti bernilai 0 (false). Lain halnya dengan XNOR yang hanya menghasilkan keluaran 1 (true) ketika kedua nilai inputnya sama. Masih bingung? Oke, mari kita bahas perbedaannya di bawah ini.

Nah, dari dua gambar gerbang logika di atas kita bisa melihat perbedaan dari XNOR dan NOR. Gerbang logika XNOR memiliki input A=1, B=1, dan menghasilkan keluaran X = 1. Hal tersebut karena logika XNOR hanya akan menghasilkan nilai keluaran 1 (true) jika nilai masukannya sama. Namun, jika nilai yang dimasukkan berbeda akan menghasilkan nilai 0 (false).

Sedangkan gerbang logika NOR akan menghasilkan nilai keluaran 1 (true) jika nilai masukannya tidak sama. Walaupun input A dan B sama-sama bernilai 1 akan tetap menghasilkan keluaran 0 (false). Itulah perbedaan dari logika XNOR dan NOR.

Jika logika XNOR digambarkan dalam gerbang logika dan tabel kebenaran akan terlihat sebagai berikut.

Gambar di atas sama seperti yang telah disinggung sebelumnya. Dalam XNOR, hanya masukan bernilai sama yang akan menghasilkan keluaran dengan nilai 1 (true). Jika nilai inputnya berbeda, maka hasil keluarannya otomatis menjadi 0 (false). Itulah mengapa gerbang logika ini disebut Exclusive NOR yang dilambangkan dengan tanda plus di dalam lingkaran dan ditambahkan overline di atasnya. Sehingga logika XNOR bisa diekspresikan seperti di bawah ini.

atau jika ekspresi tersebut disederhanakan menjadi seperti berikutX = AB + (AB atas).

## Rangkuman Gerbang Logika

Kita telah mempelajari apa itu gerbang logika, tabel kebenaran, dan tujuh jenis gerbang logika mulai dari AND hingga XNOR. Berikut adalah rangkuman materi yang sudah kita pelajari di modul ini.

Gerbang logika atau logic gate adalah sebuah rangkaian sirkuit digital yang berguna untuk memproses logika boolean (logika yang menghasilkan output benar atau salah). Gerbang logika ini menghasilkan output berupa 0 (salah) dan 1 (benar) berdasarkan input (masukan) yang kita berikan.

Tabel kebenaran merupakan kumpulan kombinasi beberapa nilai input berdasarkan logika yang kita buat untuk menghasilkan output (keluaran) yang masuk akal dalam bentuk tabel

Terdapat 7 jenis gerbang logika, diantaranya seperti:

* **AND**  
  Gerbang logika AND setidaknya memerlukan dua input. Jika input A dan B bernilai 1 (true), maka hasil outputnya juga pasti bernilai 1 (true). Sedangkan jika salah satu input saja bernilai 0 (false), maka hasil output juga akan 0 (false).
* **OR**  
  Jika salah satu saja ada input bernilai 1 (true), maka hasil outputnya juga pasti bernilai 1 (true). Namun, jika input keduanya bernilai 0 (false), maka hasil outputnya 0 (false).
* **NOT**  
  Gerbang logika NOT hanya memerlukan 1 input untuk mendapatkan satu keluaran. Jika input A bernilai 1 (true), maka hasilnya outputnya bernilai 0 (false). Sedangkan jika inputnya bernilai 0 (false), maka hasil outputnya bernilai 1 (true).
* **NAND**  
  Jika input A dan B keduanya bernilai 1 (true), maka hasil outputnya adalah 0 (false). Sedangkan jika hanya salah satu saja inputnya ada yang bernilai 0 (false), maka hasil outputnya adalah 1 (true).
* **NOR**  
  Jika input A dan B keduanya bernilai 0 (false), maka hasil outputnya adalah 1 (true). Sedangkan jika salah satu input saja ada yang bernilai 1 (true), maka hasil outputnya adalah 0 (false).
* **XOR**  
  Jika input A atau B nilainya berbeda misalnya A=0, B=1, maka hasil outputnya akan bernilai 1 (true). Tetapi jika kedua input nilainya sama misalnya A=1 B=1 atau A=0, B=0, maka hasil outputnya bernilai 0 (false).
* **XNOR**  
  Jika input A atau B nilainya sama misalnya A=1, B=1 atau A=0, B=0, maka hasil outputnya akan bernilai 1 (true). Tetapi jika kedua input nilainya berbeda misalnya A=1 B=0 atau A=0, B=1, maka hasil outputnya bernilai 0 (false).

**Transkrip Pengenalan Computational Thinking**

Semua orang pasti pernah melakukan kesalahan. Dari kesalahan tersebut kita dapat merefleksikan sebuah pemecahan masalah dalam kehidupan sehari-hari. Seorang programmer dapat belajar dari pengalaman untuk mengasah kemampuan pemecahan masalah. Sehingga ketika menemui setiap permasalahan, seorang programmer dapat belajar dan memecahkan setiap permasalahan yang dihadapi.

Ketika seorang programmer membuat sebuah program, mereka akan mulai berpikir secara terstruktur layaknya sebuah komputer mengeksekusi setiap perintah kode secara berurutan. Begitu pula saat memecahkan sebuah permasalahan. Mereka akan mencari solusi dari permasalahan secara terstruktur dan seefisien mungkin. Proses pemecahan masalah tersebut dikenal juga dengan nama, computational thinking.

Pembahasan mengenai computational thinking bisa dikatakan masih baru. Beberapa peneliti di tahun 2011 seperti Jan Cuny, Larry Snyder, dan Wing mendefinisikan Computational thinking sebagai proses berpikir dalam perumusan masalah beserta solusinya. Sehingga solusi tersebut akan direpresentasikan dalam bentuk kegiatan secara efektif oleh agen pengolah informasi [[5]](https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why).

Dengan demikian, kita sebagai agen pengolah informasi bisa mencari solusi dari sebuah permasalah dengan lebih mudah. Computational thinking bukanlah sebuah istilah tunggal karena di dalamnya terdapat beberapa elemen penting. Elemen tersebut terdiri dari 5 metode yang dapat dilakukan dalam pemecahan sebuah masalah baik dalam pemrograman atau kehidupan sehari-hari. Kelima elemen computational thinking yang akan kita bahas di modul ini antara lain:

* Memecah permasalahan yang besar menjadi bagian kecil (decomposition).
* Pelajari pola dari setiap permasalahan (pattern recognition).
* Mengabstraksikan suatu permasalahan (abstraction).
* Susun langkah menggunakan algoritma (algorithm).
* Mengevaluasi solusi yang didapatkan (evaluation).

Penasaran tentang penjelasan masing-masing elemen tersebut? Yuk, kita mulai belajar computational thinking.

### Decomposition

Kemampuan memecah permasalahan menjadi bagian-bagian kecil merupakan hal alami yang dimiliki manusia dan sangat berguna untuk banyak hal. Bayangkan ada sebuah mesin pabrik yang besar dan rumit. Menurut Anda bagaimanakah proses pengiriman mesin tersebut dari pihak pembuat hingga sampai di pabrik tersebut? Jika langsung diangkut dalam bentuk yang besar pasti menyusahkan proses pengirimannya. Selain itu juga beresiko rusak di perjalanannya juga.

Supaya lebih mudah maka mesin tersebut dipecah menjadi beberapa bagian dan dikirimkan per bagian. Sehingga ketika sudah sampai, teknisi pabrik yang berbekal buku panduan akan merakit mesin tersebut. Sehingga dari ilustrasi di atas kita melihat masalah dari proses pengiriman mesin yang besar yang tidak memungkinkan langsung dikirim dalam bentuk utuh. Maka dari itu kita perlu memecah beberapa komponen menjadi bagian-bagian kecil untuk mempermudah proses pengiriman. Jika kita hubungkan dengan computational thinking, maka teknik pemecahan masalah menjadi bagian-bagian kecil juga bisa disebut dengan decomposition. Tujuan dari decomposition ini sendiri adalah membuatnya lebih mudah dikelola dan dikerjakan.

Kita ambil contoh sederhana lainnya yaitu membuat nasi goreng. Membuat nasi goreng bukanlah satu perintah yang langsung bisa dikerjakan. Di dalamnya harus kita pecah terlebih dahulu supaya lebih mudah dikerjakan. Berikut bagian-bagian kecil dari contoh kasus membuat nasi goreng.

* Apa saja alat yang dibutuhkan?
* Apa saja bahan-bahannya?
* Berapa lama waktu yang dibutuhkan?
* Bagaimana cara memasaknya?
* Bagaimana cara penyajiannya?

Dengan adanya tugas-tugas kecil seperti di atas mulai dari menyiapkan alat dan bahan hingga cara memasak, maka kita dapat mulai mengerjakannya satu per satu. Misalnya kita menyiapkan alat terlebih dahulu seperti kompor, penggorengan, dan spatula. Kemudian beralih ke bahan-bahannya seperti minyak goreng, sepiring nasi, telur, penyedap rasa, dan sebagainya. Dengan teknik decomposition tersebut, maka kita tidak lagi melihat sebuah permasalahan sebagai hal yang kompleks. Karena kita telah memecahnya menjadi bagian atau tugas-tugas kecil yang bisa dikerjakan.

### Pattern Recognition

Setelah kita memahami pemecahan masalah dengan teknik dekomposisi (decomposition) yang memecah permasalahan ke bagian-bagian kecil, selanjutnya kita akan mempelajari pola. Apa yang dimaksud mempelajari pola? Sebelum masuk dalam pembahasan tentang pembelajaran pola, kita akan sedikit bercerita tentang kemampuan otak manusia dalam mengingat wajah seseorang. Tahukah Anda bahwa otak manusia dapat mengingat kurang lebih 5.000 sampai 10.000 wajah? Mike Burton, seorang profesor psikologi di University of York telah melakukan studi terhadap pengenalan wajah selama bertahun-tahun dan hasil utama yang didapatkannya selalu sama. Berikut kutipan dari Mike Burton kepada kantor berita The Guardian.

“Orang-orang ternyata kurang tepat dalam mengenali wajah asli dari sebuah foto identitas, namun kita dapat mengenali teman dan kolega dalam berbagai kondisi [[6]](https://www.theguardian.com/science/2018/oct/10/how-many-faces-average-person-recognises-5000)”.

Mekanisme seseorang dapat mengenali wajah memang masih menjadi misteri. Namun, beberapa peneliti berhipotesis pengenalan tersebut bisa dari genetik, bentuk fisik wajah, lingkungan, kebiasaan, atau gabungan dari semua itu. Nah, dari sana kita dapat mengetahui bahwa terdapat ciri-ciri tertentu yang bisa membantu kita mengenali seseorang. Ciri tersebut merupakan pola yang dapat kita pelajari dan kita gunakan untuk mendapat sebuah solusi. Dalam computational thinking pengenalan pola disebut juga dengan Pattern Recognition.

Pattern Recognition merupakan teknik pemecahan masalah dengan melihat perbedaan atau persamaan pola dari berbagai permasalahan. Sehingga kita dapat memprediksi atau memproyeksikan solusi apa yang harus kita lakukan. Ketika kita dihadapkan dengan banyak permasalahan, kita harus bisa mempelajari polanya supaya dapat menyelesaikan dengan cepat. Untuk dapat memperjelas maksud dari Pattern Recognition atau mempelajari pola, mari kita ilustrasikan dengan menggambar kucing.

Bayangkan Anda mendapat tugas untuk menggambar 3 jenis kucing, yaitu kucing persia, anggora, dan kucing oranye. Kita harus mempelajari persamaan dan perbedaan setiap jenis kucing. Persamaan yang dapat diambil misalnya adalah ketiga jenis kucing di atas memiliki bulu, berkaki empat, bermata dua, dan mempunyai ekor. Persamaan tersebut merupakan ciri-ciri kucing secara umum sehingga dapat kita terapkan untuk ketiga jenis kucing di atas. Sedangkan perbedaan dari ketiga jenis kucing tersebut. Misalnya dari segi warna, panjang kaki, dan kelebatan bulu.

Setelah mengetahui perbedaan dan persamaan tersebut kita dapat mulai menggambarkan setiap jenis kucingnya dengan cepat dan tepat. Bayangkan ketika kita tidak mengetahui pola atau perbedaan dan persamaan dari ketiga jenis kucing tadi. Pasti waktu yang dibutuhkan akan semakin lama karena harus melihat kembali jenis kucing yang sedang dibuat satu per satu. Atau mungkin semisal belum mengetahui bentuk kucing sama sekali, mungkin saja digambarkan tanpa ekor atau memiliki dua kaki.

Jika tadi kita ibaratkan dengan tiga jenis kucing, maka bagaimana jika Anda mendapat tugas menggambar kucing dan singa? Tekniknya sama yaitu pelajari terlebih dahulu pola di antara keduanya, maka Anda akan lebih mudah dan cepat untuk menggambarkannya.

### Abstraction

Setelah kita mengetahui pola dari sebuah permasalahan, kita perlu mengabstraksikannya untuk mendapatkan data yang bersifat umum dan menyaring data yang tidak diperlukan. Tahukah Anda apa itu abstraksi? Abstraksi merupakan bagian ketiga dari computational thinking yang identik dengan penyaringan dan pengumpulan data yang bersifat umum. Selain itu, abstraksi juga akan mengesampingkan detail data yang bersifat khusus atau yang tidak kita perlukan untuk lebih berkonsentrasi terhadap apa yang akan dilakukan. Misalnya saat kita naik sepeda motor kita tidak perlu tahu komponen mana yang membuatnya berjalan. Yang perlu kita tahu untuk menyalakan mesin cukup dengan tombol starter yang ada di setir atau starter pedal kaki.

Masih ingat ilustrasi menggambar kucing? Dalam proses abstraksi ini kita mulai dengan mencatat karakteristik yang dimiliki kucing secara umum dan khusus. Hasil pencatatannya sebagai berikut:

* Karakteristik umum: Memiliki kaki, bulu, ekor, mata, bisa mengeong, makan ikan.
* Karakteristik khusus: Memiliki kaki pendek, ekor panjang, mata biru, mengeong dengan keras, makan ikan asin.

Untuk menggambar kucing secara umum, kita memang perlu mengetahui kalau kucing pasti memiliki kaki, ekor, bulu, dan mata karena semua kucing seperti itu. Tetapi kita tidak perlu tahu apa warna bulu atau matanya, ekornya panjang atau pendek, dan seterusnya.

Peranan proses abstraksi penting karena akan membentuk gagasan umum dari sebuah permasalahan dan kita dapat mengetahui bagaimana cara penyelesaiannya. Jika kita tidak melakukan proses abstraksi, maka bisa saja kita menyelesaikan masalah dengan cara yang kurang tepat. Misalnya jika kita tidak berpikir abstrak dalam menggambar kucing, maka kita bisa saja berpikir bahwa semua kucing memiliki bulu berwarna oranye, mata biru, dan ekor panjang. Kemudian Anda melihat tetangga Anda memiliki kucing yang berwarna abu-abu, matanya cokelat, dan ekornya pendek. Tentunya ciri-ciri kucing tetangga Anda berlawanan dengan penggambaran kucing di benak Anda. Dari sana Anda mulai mengabstraksikan kucing memang memiliki bulu tapi tidak semuanya berwarna oranye; warna matanya tidak semuanya biru; dan begitu pula ekornya tidak semuanya panjang. Dalam hal ini Anda akan mengetahui dengan lebih jelas penggambaran seekor kucing secara umum dan khusus.

Proses abstraksi cukup mudah, bukan? Masih ada lagi element computational thinking lainnya. Yuk, lanjut ke materi berikutnya.

### Algorithm

Setelah kita mendapatkan gagasan atau instruksi yang harus dikerjakan, maka kita susun menjadi langkah-langkah penyelesaian menggunakan algoritma. Masih ingat pembahasan tentang algoritma pada modul sebelumnya? Algoritma merupakan sekumpulan alur instruksi yang berurutan untuk menyelesaikan permasalahan. Sehingga kita akan merangkai sekumpulan instruksi tersebut menjadi urutan yang terstruktur, logis, dan mudah dipahami. Ketika kita memberikan algoritma yang tidak cukup efisien terhadap komputer, maka hasilnya juga pasti tidak seperti yang diharapkan. Begitupun dalam kehidupan sehari-hari, jika kita tidak menyusun langkah-langkah penyelesaian dengan perencanaan yang baik, maka hasilnya juga tidak baik.

Proses yang dimulai dari memecah masalah menjadi bagian kecil hingga abstraksi dan mengurutkannya menjadi urutan yang tepat sesuai jenis permasalahan juga bisa disebut algoritma. Algoritma yang dibuat harus memuat beberapa hal yaitu, titik awal, titik akhir, dan serangkaian instruksi yang jelas.

Ibaratkan proses penyusunan algoritma ini dengan alur kegiatan yang harus dilakukan untuk berangkat ke sekolah. Mulai dari bangun tidur, mandi, menggosok gigi, sarapan, hingga tiba di sekolah. Setiap kegiatan juga memiliki detailnya sendiri-sendiri, misal mandi pukul 6, sarapan nasi goreng, pergi ke sekolah naik sepeda, dll. Dari rangkaian kegiatan tersebut secara tidak langsung Anda telah menerapkan algoritma yang terstruktur setiap hari ketika berangkat ke sekolah.

Teknik menyusun algoritma ada 2 (dua) cara yaitu pseudocode dan flowchart. Pernah mendengar dua istilah tersebut? Flowchart atau bisa disebut dengan diagram alir merupakan bentuk penggambaran dengan pendekatan visual terkait langkah-langkah dan keputusan untuk melakukan sebuah proses, alur kerja, ataupun algoritma.

Sedangkan Pseudocode merupakan istilah dalam pemrograman untuk menuliskan sebuah sintaks, statement, algoritma, dan lainnya dalam bahasa yang bisa dipahami oleh manusia. Sederhananya, pseudocode adalah bentuk representasi dari kode kita nantinya dengan versi yang human readable, bukan computer readable.

### Evaluation

"Tanpa evaluasi diri yang tepat, kegagalan tak akan bisa dihindari." ~ John Wooden

Sama seperti quote dari John Wooden di atas, kita tidak boleh melewatkan kegiatan evaluasi. Karena salah satu penyebab kegagalan adalah kurangnya proses evaluasi yang tepat. Kita telah melalui proses pemecahan masalah menjadi bagian kecil hingga merangkainya menjadi sebuah solusi.

Proses terakhir yang dapat kita lakukan adalah melakukan evaluasi terhadap solusi yang telah kita dapatkan. Proses ini perlu dilakukan sebelum melangkah ke permasalahan atau tugas yang baru.

Bayangkan Anda membuat sebuah program yang menurut Anda sudah tidak ada eror. Karena Anda tidak mengevaluasi, kemudian seorang pengguna menemukan sebuah kendala yang membuat program tersebut terasa lambat. Ketika Anda memeriksa kembali ternyata ada beberapa kode yang harusnya perlu ditambahkan untuk membuat programnya berjalan lancar. Seandainya Anda telah melakukan evaluasi sebelum aplikasi tersebut dicoba oleh pengguna, maka pengguna tidak akan merasakan aplikasinya berjalan lambat.

Tidak hanya pada bidang pemrograman, dalam kehidupan sehari-hari pun kita harus selalu mengevaluasi solusi dari permasalahan yang kita hadapi. Sehingga jika kita merasa solusi dari permasalahan kita kurang maksimal, maka bisa kita pelajari kembali lantas dan perbaiki untuk mengatasi permasalahan serupa di masa mendatang.

Nah, dalam modul ini kita telah membahas banyak hal terkait computational thinking terkait proses pemecahan masalah. Beberapa teknik yang sudah dijelaskan secara urut mulai dari proses pemecahan masalah menjadi bagian kecil (decomposition); mempelajari pola setiap permasalahan (pattern recognition); abstraksi (abstraction); menyusun langkah-langkahnya dengan algoritma; hingga mengevaluasi dari solusi yang didapatkan. Dengan memahami proses tersebut, kita dapat membiasakan diri untuk menyelesaikan permasalahan dengan lebih efektif dan efisien baik dalam bidang pemrograman maupun kehidupan sehari-hari.

## Rangkuman dari Computational Thinking

Ketika seorang programmer membuat sebuah program, mereka akan mulai berpikir secara terstruktur layaknya sebuah komputer mengeksekusi setiap perintah kode secara berurutan. Begitu pula saat memecahkan sebuah permasalahan. Mereka akan mencari solusi dari permasalahan secara terstruktur dan seefisien mungkin. Proses tersebut dikenal juga dengan computational thinking.

Terdapat 5 elemen dalam computational thinking antara lain:

* **Decomposition**  
  Decomposition merupakan teknik pemecahan masalah yang besar menjadi bagian-bagian kecil. Tujuan dari decomposition ini sendiri adalah membuat sebuah masalah lebih mudah dikelola dan dikerjakan.
* **Pattern Recognition**  
  Pattern Recognition merupakan teknik pemecahan masalah dengan melihat perbedaan atau persamaan pola dari berbagai permasalahan. Sehingga kita dapat memprediksi atau memproyeksikan solusi apa yang harus kita lakukan.
* **Abstraction**  
  Abstraction atau abstraksi merupakan teknik pemecahan masalah dengan penyaringan dan pengumpulan data yang bersifat umum. Selain itu abstraksi juga akan mengesampingkan detail data yang bersifat khusus atau yang tidak kita perlukan untuk lebih berkonsentrasi terhadap apa yang akan dilakukan.
* **Algorithms**  
  Algoritma merupakan sekumpulan alur instruksi yang berurutan untuk menyelesaikan permasalahan. Sehingga kita akan merangkai sekumpulan instruksi tersebut menjadi urutan yang terstruktur, logis, dan mudah dipahami. Ketika kita memberikan algoritma yang tidak cukup efisien terhadap komputer, maka hasilnya juga pasti tidak seperti yang diharapkan. Begitupun dalam kehidupan sehari-hari, jika kita tidak menyusun langkah-langkah penyelesaian dengan perencanaan yang baik, maka hasilnya juga tidak baik. Teknik menyusun algoritma ada 2 (dua) cara yaitu pseudocode dan flowchart.  
    
  Flowchart atau bisa disebut dengan diagram alir merupakan bentuk penggambaran dengan pendekatan visual terkait langkah-langkah dan keputusan untuk melakukan sebuah proses, alur kerja, ataupun algoritma.  
    
  Sedangkan Pseudocode merupakan istilah dalam pemrograman untuk menuliskan sebuah sintaks, statement, algoritma, dan lainnya dalam bahasa yang bisa dipahami oleh manusia.
* **Evaluation**  
  Proses terakhir yang dapat kita lakukan adalah melakukan evaluasi terhadap solusi yang telah kita dapatkan. Proses ini perlu dilakukan sebelum melangkah ke permasalahan atau tugas yang baru.

Nah, itulah rangkuman materi yang telah kita pelajari dari sub-modul Computational Thinking. Yuk, kita lanjut ke materi berikutnya untuk melanjutkan pembelajaran di modul Pengenalan ke Logika Pemrograman.

**Transkrip Studi Kasus Penggunaan Logika Pemrograman**

Tidak terasa kita telah sampai di penghujung modul Pengenalan ke Logika Pemrograman.

Kita sudah mempelajari banyak materi seperti perbedaan logika dan algoritma dalam pemrograman, gerbang logika, computational thinking, dan lainnya.

Pada modul ini kita akan membahas contoh studi kasus tentang logika pemrograman. Tujuan dari studi kasus ini adalah untuk memberi Anda wawasan tentang bagaimana ilmu logika pemrograman dapat diterapkan dalam contoh kasus yang nyata.

Oke, studi kasus yang kita bahas kali ini adalah menghitung luas segitiga. Tentunya Anda sudah tahu bahwa studi kasus tersebut identik dengan perhitungan matematika. Maka dari itu kita dapat menerapkan logika aritmatika yang pernah dibahas di materi sebelumnya. Sehingga untuk logika dari luas segitiga adalah menggunakan rumus alas kali tinggi / 2. Nah, di sana kita sudah menggunakan operator matematika yaitu perkalian dan pembagian.

Kemudian bagaimana langkah-langkah menghitung luas segitiga mulai dari awal hingga mendapat hasil yang diinginkan? Yuk kita rancang algoritmanya.

1. **Tentukan nilai alas segitiga**  
   Di sini Anda dapat mendefinisikan nilai alas segitiga menggunakan satuan centimeter atau meter sesuai kebutuhan.
2. **Tentukan nilai tinggi segitiga**  
   Selanjutnya Anda perlu menentukan nilai tinggi dari segitiga tersebut. Sebaiknya gunakan satuan yang sama seperti nilai alas.
3. **Mulai hitung luas segitiga**  
   Nah, di sini kita akan mulai menerapkan logika aritmatika. Masukkan nilai alas dan tinggi segitiga yang sudah ditentukan. Hasil kali dari alas dan tinggi tersebut jangan lupa dibagi 2 supaya sesuai dengan rumus untuk mendapat luas segitiga.
4. **Luas segitiga sudah ditemukan dan selesai.**

Hore, kita sudah berhasil menyelesaikan studi kasus untuk menghitung luas segitiga. Dengan menerapkan logika dan algoritma, hasil yang kita peroleh dalam penyelesaian sebuah masalah akan lebih optimal.

Nah, karena sub modul ini merupakan penutup dari modul pengenalan ke logika pemrograman. Pada materi selanjutnya Anda dapat membaca kembali rangkuman materi dari keseluruhan sub-modul sebelum masuk ke ujian. Sudah tidak sabar untuk mengerjakan ujian? Yuk, kita simak dulu rangkuman materi di langkah berikutnya. Selamat Belajar.

**Daftar Referensi**

[1] Technopedia. “*Programming Logic*”. [https://www.technopedia.com/definition/3945/programming-logic](https://www.techopedia.com/definition/3945/programming-logic) (diakses pada 08 Februari 2021).

[2] Stanford Encyclopedia of Philosophy. “*Inductive Logic*”. <https://plato.stanford.edu/entries/logic-inductive/> (diakses pada 08 Februari 2021).

[3] Chad N. “*Deductive Logic*”. <http://www.ilas.nagoya-u.ac.jp/~nilep/deductive.html> (diakses pada 08 Februari 2021).

[4] KBBI Daring. “*Pengertian Silogisme*”. <https://kbbi.kemdikbud.go.id/entri/silogisme> (diakses pada 08 Februari 2021).

[5] Wing, J.M. (2011), "Research Notebook: Computational thinking -what and why?" <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why> (diakses pada 27 Oktober 2020).

[6] Ian S. (2018). "Psychologists' face off reveals humans can recognise 5000 people." theguardian.com. <https://www.theguardian.com/science/2018/oct/10/how-many-faces-average-person-recognises-5000> (diakses pada 28 Oktober 2020).