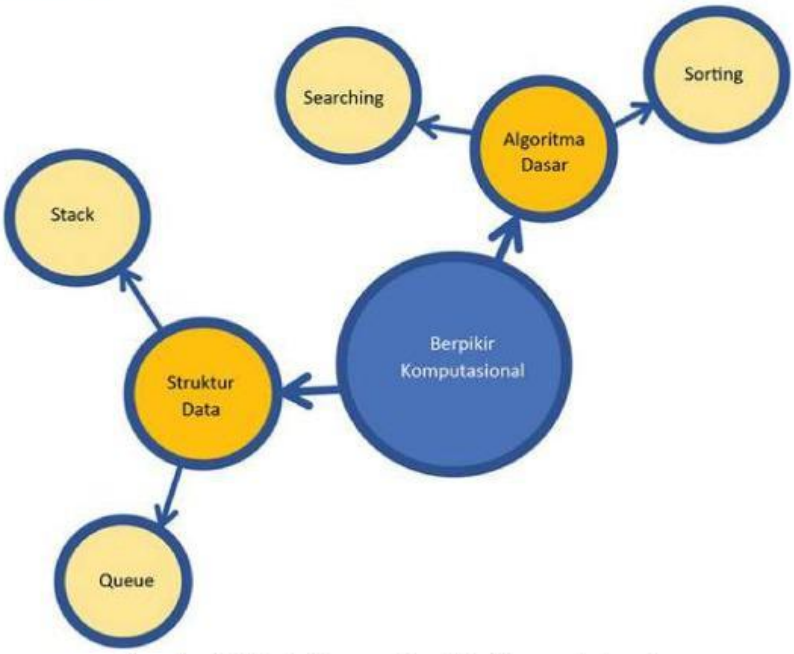
Peta Konsep



Gambar 2.1 Peta Konsep Berpikir Komputasional

A. Berpikir Komputasional (*Computational Thinking*)

Apa itu berpikir komputasional? Berpikir komputasional atau kerap disebut *Computational Thinking* adalah proses berpikir yang digunakan untuk memahami, merumuskan, dan menyelesaikan masalah dengan cara yang dapat dilakukan oleh komputer. Melalui Berpikir komputasional, kalian akan berlatih berpikir seperti seorang ilmuwan Informatika, bukan berpikir seperti komputer karena komputer adalah mesin.

Kegiatan utama dalam BK ialah penyelesaian masalah (*problem solving*), untuk menemukan solusi yang eisien, efektif, dan optimal sehingga solusinya bisa dijalankan oleh manusia maupun mesin. Dengan kata lain, kegiatan dalam BK ialah mencari strategi untuk mengatasi persoalan. Persoalan apa yang akan diselesaikan? Sebetulnya, hampir semua persoalan sehari-hari mengandung konsep komputasi sehingga bisa diselesaikan dengan bantuan mesin komputer. Sebagai contoh, robot yang bertugas melayani penjualan di restoran atau mengantar makanan dan obat untuk pasien di rumah sakit yang sudah dipakai di beberapa negara maju, sistem komputer untuk memantau perkebunan sawit yang siap panen dan sebagainya.

B. Manfaat Berpikir Komputasional (*Computational Thinking*)

1. Membantu dalam menyelesaikan suatu masalah yang besar dan kompleks dengan cara yang efektif dan efisien. Masalah yang kompleks dapat diatasi dengan efektif sehingga bertransformasi menjadi masalah yang lebih sederhana.
2. Melatih pikiran untuk berpikir secara matematis, kreatif, terstruktur, dan logis.
3. Mempermudah pengamatan terhadap masalah dan menemukan berbagai solusi yang dapat mengarah pada penyelesaian masalah secara efektif dan efisien. Semakin banyak opsi solusi yang ditemukan, semakin mungkin suatu masalah dapat diatasi dengan baik.
4. Bekerja menjadi lebih profesional dan efisien, serta lebih peka terhadap permasalahan.
5. Menciptakan inovasi tertentu maupun sistem yang lebih praktis dalam menuntaskan permasalahan.

C. Cara Berpikir Komputasional (*Computational Thinking*)

Ada 4 fondasi berpikir komputasional yang dikenal dalam ilmu Informatika, yaitu Abstraksi, Algoritma, Dekomposisi, dan Pola, yang sangat mendasar dan secara garis besar dijelaskan sebagai berikut.

1. Abstraksi, yaitu menyarikan bagian penting dari suatu permasalahan dan mengabaikan yang tidak penting sehingga memudahkan fokus kepada solusi. Abstraksi merupakan proses dalam berpikir komputasional yang difokuskan pada aspek-aspek yang memiliki relevansi dengan permasalahan yang sedang dihadapi, sementara halihal yang tidak terlalu diperlukan dalam menyelesaikan masalah dapat di abaikan.
2. Algoritma, yaitu menuliskan otomasi solusi melalui berpikir algoritmik (langkah-langkah yang terurut) untuk mencapai suatu tujuan (solusi). Jika langkah yang runtut ini diberikan ke komputer dalam bahasa yang dipahami oleh komputer, kalian akan dapat "memerintah" komputer mengerjakan langkah tersebut. Berpikir menggunakan algoritma melibatkan perencanaan dan langkah-langkah petunjuk terstruktur untuk menyelesaikan suatu masalah. Algoritma digunakan dalam berbagai proses perhitungan, pengolahan data dan otomatisasi. Meskipun umumnya terkait dengan penelitian program komputer, algoritma juga diterapkan dalam memecahkan masalah sehari-hari.

3. Dekomposisi dan formulasi persoalan sedemikian rupa sehingga dapat diselesaikan dengan cepat dan eisien serta optimal dengan menggunakan komputer sebagai alat bantu. Persoalan yang sulit apalagi besar akan menjadi komputer sebagai alat bantu. Persoalan yang sulit apalagi besar akan menjadimudah jika diselesaikan sebagian-sebagian secara sistematis. Metode dekomposisi digunakan untuk menguraikan masalah yang berskala besar dan rumit menjadi serangkaian masalah yang lebih kecil dan mudah untuk diatasi. Selain itu, dekomposisi juga memberikan kemudahan untuk menghasilkan inovasi.
4. Pengenalan pola persoalan, generalisasi serta mentransfer proses penyelesaian persoalan ke persoalan lain yang sejenis. untuk menemukan struktur atau keteraturan dalam data, sehingga dapat di peroleh informasi penting yang membantu pememahaman terhadap pola yang telah diidentifikasi. Tujuannya adalah memberikan kemampuan pada komputer untuk mendeteksi objek di lingkungan dan menentukan identitasnya, seperti pengenalan suara, wajah manusia, atau prediksi cuaca dalam kehidupan sehari-hari.

D. Karakteristik Berpikir Komputasional (*Computational Thinking*)

Dalam keterampilan berpikir komputasional juga memiliki karakteristik, diantaranya yakni:

1. Dalam berpikir komputasional, fokus terletak pada konsep daripada sekadar pemrograman komputer. Tujuannya adalah untuk memiliki pemahaman yang mendalam dalam ilmu komputer dan berpikir seolah-olah sebagai seorang ahli di dalamnya, melebihi hanya penguasaan program-program di dalam komputer.
2. Menekankan pemahaman konsep secara mendasar, bukan sekadar menghafal mekanis.
3. Menerapkan cara berpikir manusiawi, berpikir komputasional bukanlah replikasi cara komputer berpikir, melainkan penggabungan pemikiran manusiawi untuk memecahkan masalah dengan lebih efektif.
4. Ide dan bukanlah benda. Dalam konteks ini, ide dianggap lebih berharga daripada benda konkret. Berpikir komputasional melibatkan pendekatan konseptual yang diterapkan untuk mendekati, memecahkan masalah, mengelola kehidupan sehari-hari, dan berkomunikasi dengan orang lain.

5. Kemampuan menggunakan komputer atau perangkat lain sebagai alat untuk memecahkan masalah.
6. Keterampilan untuk mengorganisasi dan menganalisis data secara sistematis untuk mendapatkan wawasan yang bermanfaat menjadi hal yang sangat penting dalam konteks berpikir komputasional.
7. Kemampuan untuk merepresentasikan masalah melalui abstraksi dan model, termasuk penggunaan simulasi untuk pemahaman dan pemecahan masalah.
8. Keterampilan dalam merancang algoritma yang dapat diotomatisasi, mengguna menggunakan langkah-langkah terdefinisi dengan baik untuk memecahkan masalah.
9. Kemampuan untuk menganalisis dan mengidentifikasi serta menerapkan solusi dengan menggunakan berbagai kombinasi langkah dan sumber daya yang efisien dan efektif.
10. Kemampuan untuk menggeneralisasikan solusi dari satu masalah dan menerapkannya pada masalahmasalah serupa, mencerminkan transfer pemahaman dan konsep ke berbagai konteks

Berpikir komputasional perlu diasah dengan latihan rutin, mulai dari persoalan sederhana dan kecil. Kemudian, secara bertahap, persoalannya ditingkatkan menjadi makin besar, kompleks, dan rumit. Makin besar dan kompleks suatu persoalan, solusinya makin membutuhkan komputer agar dapat diselesaikan secara 26 Informatika SMA Kelas X eisien. Pada tingkat SD dan SMP, strategi penyelesaian persoalan belum secara khusus dirumuskan dalam bentuk algoritma. Pada tingkat SMA, kalian akan belajar bagaimana caranya agar solusi masalahnya bisa dituliskan dalam bentuk algoritma yang eisien dan siap dibuat menjadi program komputer.

Ruang permasalahan di dunia ini luas sekali, dan tentunya tak seorang pun ingin hidupnya menghadapi persoalan. Setiap bidang juga mempunyai persoalan dari sudut pandang bidang masing-masing, dan akan mengusulkan penyelesaian dengan menggunakan konsep dan prinsip keilmuan bidangnya. Kita belajar dari persoalan-persoalan yang ada dan pernah diusulkan solusinya. Oleh karena itu, belajar penyelesaian persoalan ialah belajar dari kasus-kasus dan solusinya. Namun, persoalan yang dibahas itu perlu di adaptasi dengan konteks kita. Kita perlu membentuk pola persoalan dan pola solusi dari latihan penyelesaiannya.

Karena sangat banyak, latihan persoalan perlu dipilih. Topik yang dipilih dalam berpikir komputasional untuk SMA dalam mata pelajaran Informatika merupakan persoalan-persoalan mendasar terkait kehidupan sehari-hari yang perlu dikuasai dan mengandung konsep Informatika yang dominan. Bisa saja persoalan tersebut berkaitan dengan bidang lain, tetapi kita akan fokus ke aspek informatika. Memusatkan penyelesaian persoalan dari satu sudut pandang ini merupakan berpikir kritis! Dengan mempelajari dan membahas latihan-latihan pada unit pembelajaran ini, diharapkan kalian akan mendapatkan dasar pengetahuan yang diperlukan untuk menemukan solusi-solusi yang membutuhkan program komputer. Melalui kasus yang dibahas, kalian diharapkan dapat membentuk katalog solusi, yang saat dibutuhkan, akan tinggal dipakai.

1. Pencarian (*Searching*)

Hidup adalah pencarian yang tiada henti. Mari, kita berpikir ke pengalaman "mencari" dalam kehidupan sehari-hari. Perhatikan contoh berikut:

a. Pernahkah kalian merasa kebingungan saat mencari sebuah buku di lemari buku kalian? Atau bahkan di perpustakaan? Saat kalian meminta bantuan kepada petugas perpustakaan, mengapa dia dapat menemukan buku yang kalian cari dengan waktu yang lebih singkat?
b. Suatu hari, kalian kehilangan baju seragam yang harus dipakai pada hari itu dan kalian mencarinya. Apa strategi kalian supaya baju tersebut cepat ditemukan?
c. Kalian mengingat sebuah potongan lirik lagu, tetapi tidak ingat judul lagu tersebut. Bagaimana kalian bisa menemukan lagu tersebut dengan cepat?

Apa itu mencari? Mencari adalah menemukan "sesuatu" yang bisa berupa benda, angka, konsep, informasi yang memenuhi kriteria tertentu dalam suatu ruang pencarian. Masalah pencarian sangat umum ditemukan di dalam kehidupan, termasuk dalam dunia komputasi. Ketika melakukan suatu pencarian, kalian harus menemukan suatu benda atau objek yang memenuhi kriteria tertentu dari sekumpulan benda atau objek lain.

Beberapa contoh dari masalah pencarian yang sering kalian temui ialah sebagai berikut:

a. Mencari buku dengan judul tertentu di rak buku perpustakaan
b. Mencari pakaian batik seragam kalian di lemari yang berisi semua pakaian yang kalian miliki.
c. Mencari dokumen atau web tertentu dengan mesin pencari seperti Google.

Mencari benda nyata gampang, tinggal kita lihat dan kita cocokkan dengan mata. Namun, mencari informasi atau konsep yang tidak kelihatan? Hmmmmm… Tidak mudah!

(a) (b)

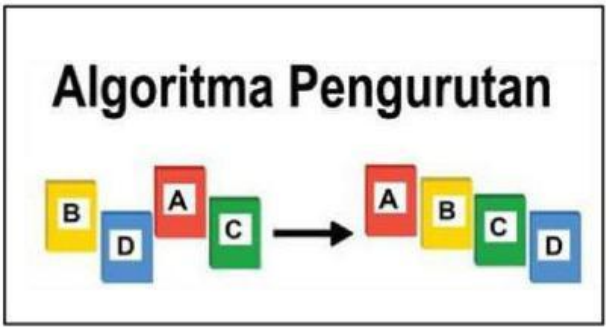
Gambar 2.2. Pencarian (a) buku di perpustakaan, (b) informasi di internet

Masalah pencarian dapat dibuat dalam bentuk yang lebih formal agar dapat diterapkan pada banyak kasus. Elemen pada masalah pencarian meliputi hal-hal berikut:

a. Sekumpulan benda atau objek.
b. Kriteria dari benda atau objek yang dicari.
c. Pengecekan benda atau objek, untuk memeriksa apakah ia memenuhi kriteria pencarian.

Pertanyaan selanjutnya ialah bagaimana strategi untuk mencari. Banyak cara yang dapat kita lakukan, misalnya: kita dapat mengambil pakaian secara acak dan mengecek apakah pakaian tersebut ialah seragam batik. Cara lain, misalnya dengan memeriksa pakaian dari yang berada paling atas ke paling bawah. Tentunya, ada banyak strategi lain yang dapat kalian gunakan. Ada strategi yang lebih baik daripada strategi yang lain, bergantung pada keadaan benda atau objek tersebut saat pencarian dilakukan. Tentunya, kita akan lebih mudah mencari suatu buku dengan judul tertentu di lemari perpustakaan yang tersusun rapi dengan aturan tertentu dibandingkan dengan mencarinya di sebuah lemari yang berantakan.

2. Pengurutan (*Sorting*)



Saat merapikan sesuatu, misalnya koleksi buku, kita menyusun buku tersebut dengan menggunakan suatu aturan. Misalnya, jika kita memiliki koleksi buku cerita berseri, kemungkinan besar kita akan menyusunnya secara berurut dari volume pertama hingga volume yang terbaru. Atau, ketika sedang berbaris, kita diminta untuk membentuk barisan berdasarkan tinggi badan. Hal-hal tersebut merupakan sebuah proses pengurutan atau sorting. Proses pengurutan akan menjadi bagian yang tidak terpisahkan dari program komputer atau aplikasi yang sering kita gunakan. Pada aktivitas ini, kita akan melihat bagaimana proses pengurutan dapat dilakukan dengan menggunakan berbagai strategi. Pelajarilah strateginya!

Pengurutan merupakan suatu permasalahan klasik pada komputasi yang dilakukan untuk mengatur agar suatu kelompok benda, objek, atau entitas diletakkan mengikuti aturan tertentu. Urutan yang paling sederhana misalnya mengurutkan angka secara terurut menaik atau menurun.

Biasanya, masalah pengurutan terdiri atas sekumpulan objek yang disusun secara acak yang harus diurutkan. Setelah itu, secara sistematis, posisi objek diperbaiki dengan melakukan pertukaran posisi dua buah objek. Hal ini dilakukan secara terus-menerus hingga semua posisi objek benar. Misal, kita memperoleh 5 buah angka acak berikut:

Kita dapat membuat angka tersebut terurut menaik dengan melakukan satu kali pertukaran, yaitu dengan menukar nilai 4 dengan nilai 3. Terdapat 2 langkah penting dalam melakukan sebuah pengurutan. Langkah pertama ialah melakukan pembandingan. Untuk melakukan pengurutan, dipastikan ada dua buah nilai yang dibandingkan. Pembandingan ini akan menghasilkan bilangan

lebih kecil dari, atau memiliki nilai sama dengan sebuah bilangan lainnya. Langkah kedua ialah melakukan penempatan bilangan setelah melakukan pembandingan. Penempatan bilangan ini dilakukan setelah didapatkan bilangan lebih besar atau lebih kecil (bergantung pada pengurutan yang digunakan).

Terdapat beberapa teknik (algoritma) untuk melakukan pengurutan seperti *bubble sort, insertion sort, quick sort, merge sort,* dan *selection sort*. Pada unit ini, hanya akan diberikan penjelasan untuk setiap tiga teknik ialah sebagai berikut:

a. Insertion Sort

Insertion Sort adalah salah satu algoritma yang digunakan untuk permasalahan pengurutan dalam list (daftar objek). Sesuai namanya, insertion sort mengurutkan sebuah list dengan cara menyisipkan elemen satu per satu sesuai dengan urutan besar kecilnya elemen hingga semua elemen menjadi list yang terurut. Misalnya, dalam kasus mengurutkan elemen list dari yang terkecil hingga terbesar (ascending), tahap pertama ialah kita akan membaca suatu elemen dengan elemen yang berdekatan. Apabila elemen yang berdekatan dengan elemen saat ini lebih kecil, elemen yang lebih kecil akan ditukar dengan elemen yang lebih besar dan dibandingkan kembali dengan elemen elemen sebelumnya yang sudah terurut. Apabila elemen saat ini sudah lebih besar dari elemen sebelumnya, iterasi berhenti. Hal ini dijalankan satu per satu hingga semua list menjadi terurut.

b. Selection Sort

Selection sort merupakan algoritma pengurutan yang juga cukup sederhana, dengan algoritma mencari (menyeleksi) bilangan terkecil/terbesar (bergantung pada urut naik atau turun) dari daftar bilangan yang belum terurut dan meletakkannya dalam daftar bilangan baru yang dijaga keterurutannya. Algoritma ini membagi daftar bilangan menjadi dua bagian, yaitu bagian terurut dan bagian yang belum terurut. Bagian yang terurut di sebelah kiri dan bagian yang belum terurut di sebelah kanan. Awalnya, semua elemen bilangan dalam daftar ialah bagian yang belum terurut, dan bagian yang terurut kosong. Berikut langkah-langkah yang terdapat pada algoritma *selection sort.*