**MAKALAH**

**ALGORITMA STRUKTUR DATA**



DISUSUN OLEH :

Nama : L Hafidl Alkhair

NIM : 2023903430060

Kelas : TRKJ 1.C

Jurusan : Teknologi Informasi dan Komputer

Program Studi : Teknologi Rekayasa Komputer Jaringan

Dosen Pengajar : Indrawati, SST. MT



**PROGRAM STUDI TEKNOLOGI REKAYASA KOMPUTER JARINGAN**

**JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER**

**POLITEKNIK NEGERI LHOKSEUMAWE**

**TAHUN 2022/2023**

**DAFTAR ISI**

DAFTAR ISI 2

BAB I 3

Pendahuluan 3

Latar Belakang 3

Rumusan Masalah 4

Tujuan penulisan 5

Manfaat penulisan 6

BAB II 8

Pembahasan 8

Sejarah algoritma 8

Sejarah pemograman 9

Penggunaan algoritma 11

Penyajian algoritma 12

Pseudocode 12

Flowchart 20

Cara Kerja Algoritma Pemrograman 26

Fungsi Algoritma Pemrograman 27

Jenis-jenis Algoritma Pemrograman 27

Contoh Algoritma Pemrograman 30

Struktur data 32

Jenis Struktur Data Linear 33

Jenis Struktur Data non-Linear 36

Mengamankan Data 38

Bab III 40

Penutup 40

Kesimpulan 40

Daftar Pustaka 41

**BAB 1**

**PENDAHULUAN**

**Latar Belakang**

Algoritma dan struktur data adalah fondasi dasar dalam pemrograman komputer. Mereka membantu pengembang perangkat lunak merancang, mengembangkan, dan memahami cara mengatasi masalah dalam lingkungan komputer.

**Algoritma**

Algoritma adalah panduan atau resep yang mengatur bagaimana suatu tugas atau proses harus dilakukan.

Beberapa karakteristik penting dari algoritma meliputi : **input, Langkah-Langkah, Output, Terdefenisi dengan jelas, Terbatas dalam waktu.** Algoritma digunakan di berbagai bidang, termasuk pemrograman komputer, matematika, ilmu data, kecerdasan buatan, dan banyak lagi. Mereka adalah fondasi dari pemecahan masalah komputasi dan digunakan untuk mengembangkan perangkat lunak, mengoptimalkan proses bisnis, dan melakukan analisis data, di antara banyak aplikasi lainnya. Algoritma dapat digunakan untuk menyelesaikan berbagai jenis masalah, dari yang sederhana seperti pengurutan bilangan hingga yang kompleks seperti pemodelan perubahan iklim.

**Stuktur Data**

Struktur data adalah cara data disimpan, diorganisasi, dan diakses dalam komputer atau sistem komputasi. Ini adalah konsep fundamental dalam ilmu komputer yang memungkinkan kita untuk menyusun dan mengelola data dengan efisien. Struktur data memberikan kerangka kerja yang memungkinkan operasi-operasi tertentu dilakukan pada data dengan cepat dan efisien.

Beberapa karakteristik penting dari struktur data meliputi: **Organisasi Data, Efesiensi, Akses data, Operasi tertentu, Tipe dan Abstrak.** Struktur data memainkan peran penting dalam pengembangan perangkat lunak, analisis data, kecerdasan buatan, grafika komputer, basis data, dan banyak bidang lainnya di dalam ilmu komputer. Pemilihan struktur data yang tepat adalah kunci untuk mengoptimalkan kinerja dan efisiensi aplikasi komputer.

RUMUSAN MASALH

Rumusan masalah terkait algoritma dan struktur data dapat di rumuskan sebagai berikut:

1. Bagaimana pemilihan algoritma dan struktur data yang tepat dapat memengaruhi kinerja aplikasi komputer dalam hal waktu pemrosesan dan penggunaan memori?
2. Bagaimana kita dapat mengoptimalkan algoritma dan struktur data untuk meningkatkan efisiensi dalam pencarian informasi dalam basis data besar?
3. Bagaimana kita dapat membandingkan dan mengevaluasi efisiensi berbagai algoritma pengurutan dalam hal waktu pemrosesan dan kompleksitasnya?
4. Bagaimana algoritma dan struktur data digunakan dalam pemrosesan citra medis untuk mendeteksi dan mendiagnosis penyakit?
5. Bagaimana algoritma genetika dan struktur data yang sesuai dapat digunakan untuk menyelesaikan masalah optimisasi dalam berbagai bidang, seperti perencanaan rute atau pengelolaan sumber daya?
6. Bagaimana kompleksitas algoritma berubah ketika diterapkan pada data besar, dan bagaimana kita dapat mengatasi tantangan ini dalam analisis data besar?
7. Apa tantangan dan manfaat migrasi dari struktur data konvensional ke struktur data terdistribusi dalam skenario perusahaan atau data pusat?
8. Bagaimana algoritma machine learning dapat digunakan bersamaan dengan struktur data yang efisien untuk menganalisis dan mendapatkan wawasan dari data besar?
9. Bagaimana kita dapat memilih dan merancang struktur data yang cocok untuk mengelola dan menganalisis data dari perangkat IoT yang sangat beragam?
10. Bagaimana kita dapat mengamankan data dalam berbagai struktur data agar melindungi privasi pengguna dan mencegah akses yang tidak sah?

**Tujuan Penulisan**

Tujuan penulisan algoritma dan struktur data adalah sebagai berikut:

1. Pemecahan Masalah: Tujuan utama dari penulisan algoritma adalah untuk merancang serangkaian langkah yang dapat menyelesaikan suatu masalah atau tugas tertentu. Algoritma adalah panduan yang membantu dalam mengatasi masalah yang beragam, dari yang sederhana hingga yang sangat kompleks.
2. Efisiensi Kinerja: Algoritma dan struktur data dirancang untuk meningkatkan efisiensi kinerja aplikasi komputer. Dengan memilih algoritma yang tepat dan struktur data yang sesuai, kita dapat mengoptimalkan penggunaan sumber daya seperti waktu pemrosesan dan penggunaan memori.
3. Skalabilitas: Salah satu tujuan penting adalah memastikan bahwa algoritma dan struktur data dapat berskala dengan baik ketika jumlah data atau beban kerja meningkat. Ini penting untuk menghindari penurunan kinerja saat aplikasi berhadapan dengan data yang lebih besar.
4. Kemudahan Pemeliharaan: Algoritma dan struktur data yang baik juga dirancang dengan tujuan untuk memudahkan pemeliharaan kode. Mereka harus mudah dimengerti dan dikelola, sehingga pengembang dapat memperbaikinya atau memodifikasinya dengan efisien.
5. Kereabilitasan (Reliability): Algoritma dan struktur data yang andal adalah yang dapat diandalkan dalam berbagai situasi. Mereka harus memberikan hasil yang benar dan konsisten dalam semua kondisi.
6. Penghematan Sumber Daya: Algoritma dan struktur data yang efisien membantu menghemat sumber daya seperti daya komputasi dan memori. Hal ini menjadi semakin penting dalam lingkungan dengan sumber daya terbatas seperti perangkat mobile atau Internet of Things (IoT).
7. Kemudahan Pengembangan: Dengan algoritma dan struktur data yang baik, pengembang perangkat lunak dapat mengembangkan aplikasi dengan lebih cepat dan lebih mudah, karena mereka memiliki kerangka kerja yang kuat untuk bekerja.
8. Ketahanan Terhadap Serangan: Algoritma dan struktur data juga harus dirancang dengan tujuan keamanan. Mereka harus tahan terhadap serangan seperti serangan pemecahan kode atau serangan keamanan lainnya.
9. Kemudahan Pengujian: Algoritma dan struktur data yang baik juga memudahkan pengujian aplikasi. Pengujian dapat dilakukan dengan lebih baik dan lebih komprehensif jika algoritma dan struktur data telah dirancang dengan baik.
10. Inovasi Teknologi: Dalam beberapa kasus, penulisan algoritma dan struktur data bertujuan untuk mengembangkan teknologi baru atau menghadapi tantangan baru yang muncul dalam teknologi informasi.

**Manfaat penulisan**

Penulisan algoritma dan struktur data memberikan beberapa manfaat yang signifikan, antara lain :

1. Pemecahan Masalah: Algoritma membantu dalam pemecahan masalah. Mereka adalah panduan langkah-demi-langkah yang dapat digunakan untuk merancang solusi untuk berbagai masalah, dari yang sederhana hingga yang kompleks.
2. Kinerja yang Lebih Baik: Pemilihan algoritma dan struktur data yang tepat dapat meningkatkan kinerja aplikasi. Ini menghasilkan waktu pemrosesan yang lebih cepat dan penggunaan memori yang lebih efisien.
3. Optimalisasi Sumber Daya: Penulisan algoritma dan struktur data yang efisien membantu menghemat sumber daya seperti daya komputasi, memori, dan bandwidth. Ini sangat penting dalam lingkungan dengan sumber daya terbatas.
4. Skalabilitas: Algoritma dan struktur data yang baik dapat berskala dengan baik saat jumlah data atau beban kerja meningkat. Ini memastikan bahwa aplikasi tetap berkinerja baik saat menghadapi pertumbuhan.
5. Pemeliharaan yang Mudah: Kode yang didasarkan pada algoritma dan struktur data yang baik umumnya lebih mudah dipelihara. Pengembang dapat dengan cepat memahami dan memodifikasi kode tersebut.
6. Kemudahan Pengujian: Algoritma dan struktur data yang baik memudahkan pengujian aplikasi. Pengujian dapat dilakukan dengan lebih baik dan lebih komprehensif.
7. Kemampuan Inovasi: Penulisan algoritma dan struktur data juga memungkinkan pengembang untuk berinovasi dan menciptakan solusi yang lebih baik untuk masalah tertentu.
8. Ketahanan Terhadap Serangan: Dalam bidang keamanan, penggunaan algoritma dan struktur data yang benar dapat membantu melindungi aplikasi dari serangan dan kerentanan keamanan.
9. Pengembangan yang Lebih Cepat: Dengan algoritma dan struktur data yang baik, pengembang dapat mengembangkan aplikasi lebih cepat karena mereka memiliki kerangka kerja yang kuat untuk bekerja.
10. Penggunaan yang Luas: Algoritma dan struktur data digunakan di hampir semua aspek ilmu komputer dan teknologi informasi, termasuk pemrograman, analisis data, kecerdasan buatan, dan banyak lagi. Menguasai konsep-konsep ini membuka peluang luas di berbagai bidang.
11. Peningkatan Keahlian: Penulisan algoritma dan struktur data membantu pengembang meningkatkan pemahaman mereka tentang komputasi dan pemrograman. Hal ini mendukung pengembangan keterampilan dan karir mereka.

**BAB II**

**PEMBAHASAN**

Algoritma yang lebih efisien akan memerlukan waktu eksekusi yang lebih singkat untuk menyelesaikan tugas tertentu. Pemilihan algoritma dengan kompleksitas waktu yang lebih rendah dapat mempercepat aplikasi secara keseluruhan. Beberapa masalah dapat diselesaikan dengan berbagai algoritma yang berbeda. Memilih algoritma yang paling sesuai dengan jenis masalah yang dihadapi dapat mengurangi waktu pemrosesan yang dibutuhkan.

Struktur data yang tepat dapat menghemat penggunaan memori. Misalnya, menggunakan struktur data yang memori-efisien seperti array dapat mengurangi overhead penggunaan memori dibandingkan dengan struktur data yang tidak efisien. Dalam kasus struktur data yang menggunakan alokasi memori dinamis (seperti linked list), pemilihan yang baik dalam mengelola memori (misalnya, dengan menghindari kebocoran memori atau fragmentasi) dapat meminimalkan penggunaan memori yang tidak perlu.

Sejarah algoritma

Konsep dasar algoritma sudah ada pada zaman kuno. Matematikawan Yunani, Euclid, dikenal karena Algoritma Euclidean yang digunakan untuk menghitung Faktor Persekutuan Terbesar (GCD) dari dua bilangan bulat.

Matematikawan Persia, Al-Khwarizmi, memberikan kontribusi besar dengan mengembangkan metode-algoritma dalam bukunya "Al-Kitab al-Mukhtasar fi Hisab al-Jabr wal-Muqabala," yang menjadi dasar kata "aljabar" (algebra). Pada Abad Pertengahan, algoritma matematika dan ilmu pengetahuan lebih berkembang. Fibonacci memperkenalkan deret Fibonacci yang digunakan dalam berbagai algoritma, dan algoritma aritmetika India dibawa ke Eropa oleh Leonardo dari Pisa (Fibonacci). Penemuan perkakas mekanik, seperti abakus, juga berperan dalam pengembangan algoritma perhitungan.

Charles Babbage dianggap sebagai bapak komputer modern. Dia merancang mesin perbedaan dan mesin analitik yang memiliki elemen-elemen algoritma modern seperti perulangan dan percabangan.

Ada juga perkembangan teori algoritma selama abad ke-20, termasuk kerja Alan Turing tentang mesin Turing dan Alonzo Church dengan Lambda calculus, yang merupakan landasan untuk pemahaman modern tentang algoritma dan komputasi.

Era komputer modern membawa berbagai jenis algoritma yang digunakan dalam pemrosesan data dan komputasi. Algol-60 adalah salah satu bahasa pemrograman awal yang mendefinisikan algoritma dalam bentuk yang lebih dekat dengan bahasa pemrograman modern.

Perkembangan dalam algoritma berbagai bidang seperti kecerdasan buatan, pemrosesan citra, dan pemelajaran mesin telah mengubah cara kita berinteraksi dengan teknologi.

Abad ke-21 menyaksikan kemajuan pesat dalam algoritma, terutama dalam bidang seperti deep learning yang menggantikan pembelajaran mesin konvensional dan telah membawa kemajuan besar dalam pemrosesan citra, pengenalan suara, dan pemahaman bahasa alami.

Algoritma kriptografi terus berkembang untuk melindungi keamanan data di era digital, dan algoritma optimasi digunakan dalam berbagai aplikasi, dari pengiriman logistik hingga pemodelan keuangan.

Sejarah pemograman

Sejarah pemrograman komputer adalah perjalanan panjang yang dimulai sejak konsep komputasi modern pertama kali muncul hingga perkembangan pemrograman yang kita kenal saat ini. Berikut adalah gambaran umum tentang sejarah pemrograman komputer:

Konsep Awal Pemrograman (Abad ke-19 dan Awal Abad ke-20): Sebelum komputer modern ada, ada beberapa konsep awal yang menjadi dasar pemrograman. Matematikawan seperti Ada Lovelace memainkan peran penting dalam mengembangkan instruksi langkah demi langkah untuk mesin komputasi seperti mesin analitik Charles Babbage, dianggap sebagai komputer mekanis pertama.

Komputer Generasi Pertama (1940-an): Pada tahun 1940-an, komputer pertama seperti ENIAC (Electronic Numerical Integrator and Computer) dan UNIVAC (Universal Automatic Computer) muncul. Pemrograman pada komputer-komputer ini dilakukan dengan menghubungkan kabel dan mengatur saklar fisik. Ini adalah era pemrograman dalam bahasa mesin.

Munculnya Bahasa Tingkat Tinggi (Awal 1950-an): Pada awal 1950-an, bahasa pemrograman tingkat tinggi pertama seperti Fortran (Formula Translation), LISP (LISt Processing), dan COBOL (Common Business-Oriented Language) muncul. Bahasa-bahasa ini memungkinkan programmer untuk menulis kode yang lebih abstrak daripada bahasa mesin, membuat pemrograman lebih efisien.

Pengembangan Bahasa Pemrograman (1950-an hingga 1960-an): Selama periode ini, banyak bahasa pemrograman baru muncul, termasuk ALGOL, BASIC, dan FORTRAN II. Bahasa ALGOL menjadi dasar untuk banyak bahasa pemrograman modern, dan BASIC menjadi populer sebagai bahasa pemrograman untuk pemula.

Masa Revolusi Mikrokomputer (1970-an hingga 1980-an): Munculnya mikrokomputer dan PC pada tahun 1970-an membawa pemrograman ke tingkat yang lebih luas. Bahasa seperti C dan Pascal mulai mendominasi dalam pengembangan perangkat lunak.

Masa Internet dan Web (Akhir 1980-an hingga 1990-an): Perkembangan World Wide Web pada tahun 1990 oleh Tim Berners-Lee di CERN membawa revolusi baru dalam pemrograman. Bahasa pemrograman seperti Perl, Java, dan JavaScript muncul dan mendukung pengembangan web.

Era Bahasa Pemrograman Modern (2000-an hingga Sekarang): Era ini ditandai dengan kemunculan bahasa pemrograman yang sangat kuat dan produktif seperti Python, Ruby, dan JavaScript (Node.js). Terdapat pula perkembangan dalam paradigma pemrograman seperti pemrograman berorientasi objek, pemrograman fungsional, dan kecerdasan buatan (AI) yang semakin menonjol.

Pemrograman di Era Mobile dan Cloud (2010-an hingga Sekarang): Pemrograman aplikasi seluler dan komputasi awan (cloud computing) menjadi fokus utama dalam pemrograman modern. Bahasa seperti Swift untuk iOS dan Kotlin untuk Android semakin populer, sementara infrastruktur awan seperti AWS, Azure, dan Google Cloud memberikan fleksibilitas yang besar dalam pengembangan aplikasi.

Kecerdasan Buatan dan Data Besar: Kecerdasan buatan dan analisis data besar (big data) adalah tren terbaru dalam pemrograman. Bahasa seperti Python dan perpustakaan seperti TensorFlow dan PyTorch digunakan untuk mengembangkan aplikasi AI dan analisis data.

Sejarah pemrograman komputer adalah cerita tentang evolusi alat-alat dan paradigma yang digunakan oleh para pemrogram untuk mengembangkan perangkat lunak yang semakin canggih dan kompleks. Perkembangan ini telah membawa dampak besar pada berbagai aspek kehidupan kita dan terus berlanjut hingga saat ini.

Penggunaan Algoritma

Pemilihan yang tepat dari algoritma dan struktur data memastikan bahwa aplikasi dapat berkinerja baik bahkan dengan volume data yang besar. Algoritma dengan kompleksitas waktu rendah dan struktur data yang efisien membantu mengatasi masalah ini.

Beberapa struktur data, seperti hash table, dapat memiliki overhead yang signifikan terutama ketika data menjadi sangat besar. Pemilihan struktur data yang tepat untuk kasus penggunaan tertentu penting untuk menghindari overhead yang tidak perlu.

Sejarah algoritma mencerminkan perkembangan pemikiran manusia dalam pemrosesan informasi dan komputasi. Algoritma yang kita gunakan sekarang adalah hasil dari perjalanan panjang melalui waktu dan terus berkembang dengan pesat seiring dengan perkembangan teknologi dan pengetahuan kita tentang dunia komputasi.

Penyajian Algoritma

Sebagai awalan, Anda bisa belajar dua cara penyajian algoritma programming, yaitu:

1. Pseudocode

Pseudocode adalah cara penulisan program secara informal dengan kaidah sendiri, daripada menggunakan aturan yang ditetapkan bahasa pemrograman. Tujuannya yaitu agar alur logika yang ditulis lebih mudah dipahami manusia.

Di bawah ini adalah contoh pseudocode untuk menentukan bilangan terbesar dari tiga inputan angka:

Tujuan Pseudocode

Tujuan dari penggunaan pseudocode iala untuk mempermudah manusia dalam meyelesaikan suatu permasalahan yang dihadapi.

Sebab apabila kita bandingkan pseudocode dengan bahasa pemograman, pseudocode akan lebih mudah untuk dipahami.

Ciri Pseudocode

Berikut ini adalah ciri dari pseudocode.

1. Pseudocode adalah notasi atau tanda bagaimana cara menyelesaikan masalah dengan sistematis dan runut.
2. Pseudocode digunakan untuk menulis algoritma.
3. Pseudocode berisisikan serangkaian proses untuk menyelesaikan permasalahan.
4. Bahasa yang digunakan lebih ringkas serta mudah dipahami.
5. Tidak memiliki aturan baku dalam menuliskan pseudocode.

Struktur Pseudocode

Judul

{Berisi Judul Algoritma}

Deskripsi

{Berisi Deklarasi Variabel atau Konstantan}

Implementasi

{Berisi Inti Algoritma}

Penulisan sebuah pseudocode harus diawali dengan judul, kemudian di isi dengan deskripsi dimana mencakup variabel dan konstanta. Kemudian pada bagian akhir memuat implementasi yang mana bagian inti.

Notasi Pseudocode

Berikut ini adalah notasi yang digunakan dalam pseudocode.

1. Bentuk

Pernyataan

X <—- Y

keterangan :

X : diberi nilai

Y : Memberi Nilai

Ex : Hasil <—- Bilangan Mod 2

1. Bentuk Percabangan

* if kondisi then

pernyataan

* if kondisi 1 then

pernyataan 1

else

pernyataan 2

* if kondisi 1 then

pernyataan 1

else if kodisi 2 then

pernyataan 2

else if kondisi n then

pernyataan

else

pernyataan else

1. Bentuk Perulangan

* for (persyaratan) do

pernyataan for

* while (persyaratan)

pernyataan while

* Repeat

pernyataan repeat

Until (persyratan)

Fungsi Pseudocode

Pseudocode memiliki fungsi untuk mempermudah penulisan dan pemetaan dari sebuah algoritma. Sebab sebelum dapat membuat dan menjalankan sebuah program, terlebih dahulu haruslah memiliki susunan atau pemecahan masalahnya yang disebut dengan algoritma.

Melalui penulisan pseudocode juga dapat diketahui serumit apa program yang akan dibuat. Selain itu pseudocode memiliki fungsi lain yaitu sebagai alat dokumentasi.

Cara Menulis Pseudocode

Secara umum penulisan pseudocode terbagi menjadi 3 bagian sebagai berikut.

1. Bagian judul – Bagian judul senantiasa diawali oleh kata “program” kemudian diikuti oleh nama algoritma. Pada umumnya nama algoritma senantiasa terdiri dari satu kata, apabila lebih dari satu kata penulisan disatukan. Artinya jika terdiri lebih dua kata sapasi ditiadakan.
2. Bagian deklarasi – Bagian ini digunakan untuk mendefinisikan atau mendeklarasikan variabel yang dimiliki oleh algoritma. Dalam pemograman komputer sendiri memiliki beberapa variabel, seperti bilangan bulat, pecahan, desimal dan lain sebaginya.
3. Bagian isi – Bagian isi dapat dikatakan bagian utama, dimana jalannya sebah algoritma. Terdiri dari sekumpulan perintah algoritma, perintahnya pun bisa berupa runtutan, kondisional ataupun perulangan. Isi memili penulisan format, seperti yang telah saya singguh pada bagian sebelumnya, untuk lebih jelasnya pada bagian contoh pseudocode.

Contoh Pseudocode

Berikut ini kami sajikan kumpulan contoh dari pseudocode, yang tentu kalian dapat menganalisis dan mempelajarinya.

1. Contoh Pseudocode Menghitung Luas Segitiga

Terdapat hal yang perlu kita perhatikan sebelum kita menuliskan pseudocode menghitung luas segitiga, yaitu kita harus tahu terlebih dahulu mengenai rumus menghitung luas segitiga.

Setelah kita mengetahui rumus menghitung luas segitiga, selanjutnya kita harus mengetahui alas dan tinggi dari sebuah segitiga yang akan kita hitung.

Contoh soal

Diketahui sebuah segitiga memiliki ukuran sebagai berikut :

* Alas 25
* Tinggi 30

Hitunglah luas segitiga tersebut dengan menggunakan psuedocode.

program hitung\_luas\_segitiga

deklarasi

var luas,alas,tinggi:integer;

algoritma:

alas <– 25;

tinggi <– 30;

luas <– 1/2 \* alas \* tinggi

write(luas)

1. Contoh Pseudocode Menghitung Luas Lingkaran

Terlebih dahulu kita harus mengetahui rumus mengenai lingkaran, adapun rumusnya adalah sebagai berikut.

Gambar di atas sekilas menyerupai contoh grafik lingkaran, padahal bukan ya. Gambar di atas adalah lingkaran yang akan kita hitung luasnya. Seperti yang sama-sama kita ketahui rumus lingkaran ialah phi\*r.

Contoh soal

Sebuah lingkaran diketahui memiliki jari-jari 30, maka tuliskan pseudocode untuk menghitung luas lingkaran.

program hitung\_luas\_lingkaran

deklarasi

var phi : float;

var r,luas:integer;

algoritma:

phi <– 3.14;

read(r); {diinput user}

luas <– phi \* r \*r;

write(luas);

1. Contoh Pseudocode Menghitung Luas Trapesium

Berikut adalah rumus yang digunakan untuk menghitung luas trapesium.

1. Contoh Pseudocode Menghitung Luas Persegi Panjang

Sebelum kita menuliskan algoritma dalam bentuk pseudocode, kita harus mengetahui rumus yang digunakan dalam menghitung persegi panjang sebagai berikut.

Contoh soal

Tentukan pseudocode luas persegi panjang yang mana lebar dan panjangnya menggunakan inputan.

program hitung\_luas\_segi\_panjang

deklarasi

var panjang,lebar,luas:integer;

algoritma:

read(panjang);

read(lebar);

luas <– panjang \* lebar;

write(luas);

1. Contoh Pseudocode Menentukan Bilangan Prima

Berikut ini adalah contoh dari penulisan algoritma pseudocode dalam menentukan bilangan prima dan bukan bilangan prima.

Bilangan prima sendiri ialah suatu bilangan yang habis dibagi 1 dan bilangan lebih besar dari 2 8 tidak mempunyai kelipatan.

Program\_menentukan\_bilangan\_prima

Deklarasi (kamus)

Ulang,Jumb,Sisa,Bilangan : Interger

Hasil : String

Deskripsisi

Read (Bilangan)

For (ulang = 1 to ulang <=bil.step1)

sisa if (sisa=0) THEN

jUmB<-jumB+1

Else

jumb2) THEN

Hasil <-(“Bukan bil prima”)

Else

Hasil <-(“Bilangan Prima”)

Write (Hasil)

Flowchart

Flowchart adalah diagram yang menampilkan proses dan langkah pengambilan keputusan dalam suatu program. Tujuan flowchart adalah menyederhanakan rangkaian prosedur serta mengurangi risiko salah tafsir.

Salah satu contoh flowchart dalam programming adalah sebagai berikut, untuk menunjukkan alur kerja program dari awal hingga akhir dalam menghitung luas persegi panjang:

Fungsi flowchart

Fungsi utama dari flowchart adalah memberi gambaran jalannya sebuah program dari satu proses ke proses lainnya. Sehingga, alur program menjadi mudah dipahami oleh semua orang. Selain itu, fungsi lain dari flowchart adalah untuk menyederhanakan rangkaian prosedur agar memudahkan pemahaman terhadap informasi tersebut.

Jenis flowchart

Flowchart sendiri terdiri dari lima jenis, masing-masing jenis memiliki karakteristik dalam penggunaanya. Berikut adalah jenis-jenisnya:

1. Flowchart dokumen

Pertama ada flowchart dokumen (document flowchart) atau bisa juga disebut dengan paperwork flowchart. Flowchart dokumen berfungsi untuk menelusuri alur form dari satu bagian ke bagian yang lain, termasuk bagaimana laporan diproses, dicatat, dan disimpan.

1. Flowchart program

Selanjutnya kita akan membahas flowchart program. Flowchart ini menggambarkan secara rinci prosedur dari proses program. Flowchart program terdiri dari dua macam, antara lain: flowchart logika program (program logic flowchart) dan flowchart program komputer terinci (detailed computer program flowchart).

1. Flowchart proses

Flowchart proses adalah cara penggambaran rekayasa industrial dengan cara merinci dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

1. Flowchart sistem

Yang keempat ada flowchart sistem. Flowchart sistem adalah flowchart yang menampilkan tahapan atau proses kerja yang sedang berlangsung di dalam sistem secara menyeluruh. Selain itu flowchart sistem juga menguraikan urutan dari setiap prosedur yang ada di dalam sistem.

1. Flowchart skematik

Terakhir ada flowchart skematik. Flowchart ini menampilkan alur prosedur suatu sistem, hampir sama dengan flowchart sistem. Namun, ada perbedaan dalam penggunaan simbol-simbol dalam menggambarkan alur. Selain simbol-simbol, flowchart skematik juga menggunakan gambar-gambar komputer serta peralatan lainnya untuk mempermudah dalam pembacaan flowchart untuk orang awam.

Simbol flowchart

Pada dasarnya simbol-simbol dalam flowchart memiliki arti yang berbeda-beda. Berikut adalah simbol-simbol yang sering digunakan dalam proses pembuatan flowchart.

Flow

Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain atau menyatakan jalannya arus suatu proses. Simbol ini disebut juga connecting line.

One page reference

Yaitu simbol untuk keluar – masuk atau penyambungan proses dalam lembar / halaman yang sama.

Off Page reference

Yaitu simbol untuk keluar – masuk atau penyambungan proses pada lembar / halaman yang berbeda.

Terminator

Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan.

Process

Simbol yang menunjukkan pengolahan yang dilakukan oleh computer.

Decistion

Simbol pemilihan proses berdasarkan kondisi yang ada.

Input/output

Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya

Manual operation

Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer

Document

Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

Predifine Proses

Simbol untuk pelaksanaan suatu bagian (sub-program)/procedure

Display

Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.

Preparition

Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.

Simbol-simbol di atas memiliki jenis dan fungsi yang berbeda-beda. Ada yang berfungsi untuk menghubungkan satu simbol dengan simbol lainnya seperti simbol flow, on-page dan off-page reference. Selain itu ada juga simbol yang berfungsi untuk menunjukan suatu proses yang sedang berjalan, dan yang terakhir terdapat simbol yang berfungsi untuk memasukan input dan menampilkan output.

Cara Kerja Algoritma Pemrograman

Di dalam pemrograman, algoritma bekerja dengan mengandalkan tiga aspek utama, yakni input → proses → output.

Ketika pengguna memasukkan data atau perintah, komputer akan menganalisis dan mengeksekusinya sesuai alur algoritma untuk menghasilkan output yang diharapkan. Sebagai contoh, ketika melakukan pencarian di Google, langkah logika algoritma pemrogramannya adalah:

Anda menuliskan query pencarian → sistem memproses perintah Anda → sistem menampilkan hasil pencarian yang relevan terhadap perintah Anda.

Dalam praktiknya, algoritma pemrograman terdiri dari tiga macam konstruksi, yaitu algoritma sekuensial (linear sequence), algoritma percabangan (conditional), dan algoritma perulangan (looping).

* Algoritma Sekuensial – Algoritma dengan prosedur yang berjalan dari satu proses ke proses berikutnya untuk mencapai hasil akhir. Contohnya, merebus air mentah hingga air matang.
* Algoritma Percabangan – Algoritma yang menjalankan keputusan berdasarkan kondisi tertentu dengan dua atau lebih percabangan. Misalnya, jika nilai ujian lebih dari 75 maka hasilnya lulus ujian.
* Algoritma Pengulangan – Algoritma yang menjalankan urutan perintah berulang-ulang hingga beberapa kali. Contohnya, menampilkan bilangan kelipatan dua mulai dari 0 hingga 50.

Fungsi Algoritma Pemrograman

Algoritma pemrograman berfungsi untuk menyelesaikan masalah yang menjadi tujuan pembuatan program. Namun tak cuma itu, fungsi algoritma pemrograman juga:

1. Mengotomatiskan solusi permasalahan yang ingin diselesaikan program.
2. Menyederhanakan alur kerja dan logika program untuk memudahkan programmer menulis kode (coding).
3. Mencegah penulisan kode program yang sama secara berulang.
4. Memudahkan penelusuran masalah apabila terdapat bug atau program error.
5. Meringankan pekerjaan programmer saat mengupdate atau menambah fitur-fitur baru di program.

Jenis-jenis Algoritma Pemrograman

Beberapa jenis algoritma yang wajib diketahui programmer adalah sebagai berikut:

1. Algoritma Brute Force

Algoritma brute force adalah jenis algoritma yang mencoba semua kemungkinan untuk menyelesaikan masalah. Dengan kata lain, ia akan melakukan percobaan satu persatu hingga menemukan solusi yang tepat.

Contoh penggunaan algoritma Brute Force misalnya untuk memecahkan 4 digit kode PIN dari angka 0 hingga 9. Setidaknya, butuh hingga 10.000 percobaan untuk menemukan kombinasi angka yang tepat.

1. Algoritma Recursive

Jenis algoritma yang kedua adalah algoritma recursive. Pada algoritma pemrograman ini, masalah diselesaikan sedikit demi sedikit dengan cara membaginya ke dalam beberapa kondisi yang serupa.

Lebih jauh, algoritma recursive masih dibagi lagi menjadi empat tipe spesifik, yaitu:

* Algoritma Divide and Conquer – Membagi masalah menjadi dua bagian. Yang pertama yaitu masalah itu sendiri dan yang kedua adalah metode pemecahannya.
* Algoritma Dinamis – Menggunakan teknik memoisasi, yaitu menyimpan hasil pemecahan masalah ke memori untuk selanjutnya digunakan lagi di masa mendatang.
* Algoritma Greedy – Bertolak belakang dengan algoritma dinamis, algoritma greedy justru tidak akan mempertimbangkan hasil pemecahan masalah sebelumnya untuk mengambil keputusan.
* Algoritma Backtracking – Menyelesaikan masalah secara bertahap sambil mengeliminasi solusi yang ternyata tidak memecahkan masalah tersebut.

1. Algoritma Randomized

Sesuai namanya, algoritma randomized memanfaatkan penggunaan nomor secara acak untuk menentukan apa yang harus dilakukan selanjutnya. Tujuannya yaitu untuk mengurangi kerumitan di dalam pemrograman.

Contoh dari algoritma randomized adalah menentukan pivot selanjutnya menggunakan nomor yang muncul secara acak.

1. Algoritma Sorting

Bisa ditebak, algoritma sorting adalah jenis yang berfungsi untuk mengurutkan data berdasarkan kondisi tertentu, misalnya dari abjad A sampai Z atau jumlah angka besar ke kecil.

Contoh penerapan algoritma sorting antara lain bubble sort, merge sort, insertion sort, dan selection sort.

1. Algoritma Searching

Algoritma searching adalah tipe algoritma yang digunakan untuk mencari suatu data, baik yang sudah disortir maupun belum, menggunakan query yang spesifik. Contoh pemanfaatan algoritma searching yaitu binary search dan linear search.

1. Algoritma Hashing

Algoritma hashing sebenarnya mirip dengan algoritma searching. Bedanya, hashing tak hanya mencari data, tapi juga mencocokkan query pencarian dengan kunci ID yang sudah ditetapkan.

Contoh pemanfaatan algoritma hashing yaitu untuk verifikasi password ketika ingin masuk ke akun tertentu.

Contoh Algoritma Pemrograman

Ini dia lima contoh algoritma pemrograman di dalam kehidupan sehari-hari:

1. Menjumlahkan dua angka
2. Mencari angka terbesar
3. Mengepel lantai
4. Menulis pesan
5. Membuka whatsapp di web
6. Algoritma Menjumlahkan Dua Angka

Di dalam pemrograman, algoritma untuk menjumlahkan dua angka adalah sebagai berikut:

Langkah 1: Masukkan angka pertama.

Langkah 2: Masukkan angka kedua.

Langkah 3: Komputer menjumlahkan angka pertama dan kedua.

Langkah 4: Komputer menampilkan hasil penjumlahan.

1. Algoritma Mencari Angka Terbesar

Untuk mencari tahu angka terbesar dari dua pilihan, maka perhitungan algoritma bahasa pemrograman yang digunakan adalah:

Langkah 1: Masukkan angka pertama.

Langkah 2: Masukkan angka kedua.

Langkah 3: Buat kondisi pertama. Jika angka pertama lebih besar dari angka kedua, maka tampilkan angka pertama.

Langkah 4: Buat kondisi kedua. Jika angka kedua lebih besar dari angka pertama, maka tampilkan angka kedua.

Langkah 5: Komputer menampilkan angka terbesar sesuai kondisi yang sesuai.

1. Algoritma Mengepel Lantai

Contoh penerapan algoritma program untuk mengepel lantai secara sistematis yaitu:

Langkah 1: Lantai basah.

Langkah 2: Ambil alat pel.

Langkah 3: Pel lantai menggunakan alat pel.

Langkah 4: Lantai kering.

1. Algoritma Menulis Pesan

Berikut ini contoh penggunaan algoritma untuk menulis sebuah pesan singkat:

Langkah 1: Ambil handphone.

Langkah 2: Buka aplikasi pesan.

Langkah 3: Tulis pesan singkat.

Langkah 4: Pilih nomor handphone tujuan.

Langkah 5: Kirim pesan singkat.

Langkah 6: Handphone menampilkan notifikasi pesan terkirim.

1. Membuka whatsapp di web

Berikut ini contoh penggunaan algoritma membuka whatsapp di web

Langkah 1: Masuk ke chrome/google

Langkah 2: ketik wa web di pencarian

Langkah 3: klik whatsapp web

Langkah 4: jangan lupa untuk memindai barcode yang ada di google/chrome

Menggunakan handphone

Langkah 5: dan whatsapp sudah bisa dimainkan di laptop atau pc.

Struktur Data

Apa itu Struktur Data?

Struktur data adalah cara menyimpan, mengambil, dan menyusun data. Struktur data memungkinkan data yang ada di komputer lebih mudah diakses dan diperbarui.

Fondasi setiap program komputer adalah data dan algoritma. Dengan algoritma, program komputer bisa menggunakan data secara efektif. Sementara struktur data ibarat kontainer yang menyimpan banyak data dengan layout tertentu. Tiap jenis struktur data memiliki layout masing-masing.

Dikarenakan memori komputer terbatas, layout (tata letak) data harus diatur dengan baik. Oleh sebab itu, penyimpanan data membutuhkan “formasi” khusus agar data lebih rapi dan tidak menghabiskan banyak memori.

Jenis Struktur Data dan Contoh Struktur Data

Klasifikasi struktur data terdiri dari linear dan non- linear. Masing-masing klasifikasi dibagi lagi menjadi beberapa jenis, antara lain:

Jenis Struktur Data Linear

Jenis struktur data linear dibagi dalam dua jenis, yaitu struktur data statis dan dinamis.

1. Array (Statis)

Tipe Array mempunyai kumpulan elemen yang jaraknya saling berdekatan. Array dapat membantu seseorang untuk mencari data secara acak hanya menggunakan indeksnya.

Kapasitas elemen yang dapat dialokasikan pada tipe array bersifat statis. Jika Anda ingin menyisipkan elemen baru ke array, maka harus membuat array baru dengan ukuran yang lebih besar. Sebaliknya, jika ingin menghapus elemen tertentu, Anda harus membuat array baru dengan ukuran yang lebih kecil.

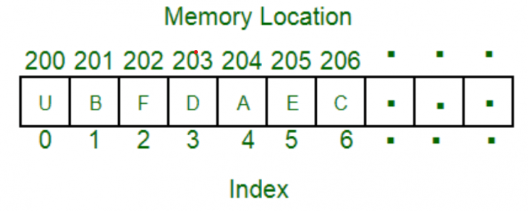
Selain itu, array juga memungkinkan Anda untuk menyimpan beberapa data dengan jenis yang sama dalam satu nama. Tipe array biasa digunakan untuk membangun struktur data, seperti vektor dan matriks.

Keunggulan tipe array:

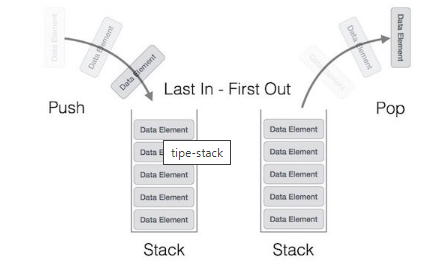
* Bisa digunakan sebagai implementasi tipe lainnya, seperti queue dan stack.
* Proses pencarian data bisa dilakukan lebih cepat.

Kekurangan tipe array:

* Penambahan dan pengurangan data membutuhkan waktu yang lebih lama karena tipe array menampung data secara berurutan.



1. Stack (Dinamis)



Stack adalah tipe struktur data yang linier dan mengikuti urutan tertentu. Adapun urutan yang digunakan adalah LIFO (Last In First Out) atau FILO (First In Last Out). Kedua istilah tersebut artinya sama, yaitu data yang terakhir masuk akan menjadi data yang keluar pertama kali. Sebaliknya, data yang pertama masuk akan menjadi data yang keluar terakhir.

Keunggulan tipe stack:

* Dapat mengelola data secara efisien.
* Bisa membersihkan objek secara otomatis.
* Dapat mengontrol memori dengan mandiri.

Kekurangan tipe stack:

* Kapasitas memori yang sangat terbatas.
* Kemungkinan terjadi overflow ketika jumlah objek terlalu banyak.
* Tidak dapat mengakses data secara acak.

1. Queue (Dinamis)



Queue adalah tipe struktur data linear yang mengikuti urutan tertentu, yaitu FIFO (First In First Out). Jadi, data yang masuk pertama kali adalah data yang pertama kali diambil. Analogi sederhana yang menggambarkan tipe ini adalah orang yang sedang mengantre. Siapa yang datang pertama, itulah yang dilayani terlebih dulu.

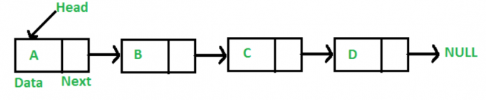
Keunggulan tipe queue:

* Data yang masuk akan dilayani sesuai urutannya.
* Proses antrean data lebih cepat dan optimal.
* Menangani beberapa tipe data sekaligus.

Kekurangan tipe queue:

* Jika waktu pelayanan habis, maka data yang terakhir masuk tidak bisa dilayani.
* Proses yang rumit saat harus menambah atau menghapus elemen dari tengah.
* Butuh waktu lama untuk mencari antrean.

1. Linked List (Dinamis)



Linked list adalah struktur data yang terdiri dari urutan data linier yang dihubungkan satu sama lain. Saat menggunakan tipe linked list, Anda harus mengakses data secara manual. Hal ini karena Anda tidak bisa mencari data dengan sistem acak.

Tipe linked list terbagi menjadi tiga jenis, yaitu singly linked list, doubly linked list, dan circular linked list. Ketiganya dapat dibedakan dari proses traversal atau proses kunjungan ke setiap node dalam satu waktu.

Keunggulan tipe linked list:

* Ukuran lebih dinamis.
* Alokasi penggunaan memori bisa Anda sesuaikan dengan kebutuhan.
* Penambahan atau pengurangan data lebih cepat.

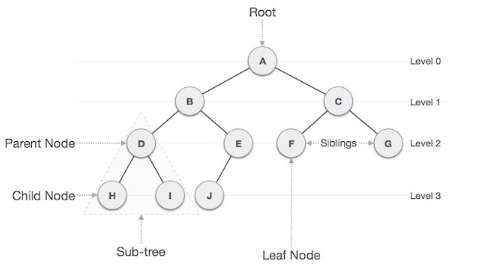
Kekurangan tipe linked list:

* Menguras memori yang lebih besar.
* Tidak bisa kembali ke node sebelumnya (reverse traversal), kecuali pada jenis doubly linked list.
* Proses traversal lebih panjang karena tidak langsung mengakses data dengan indeks.

Jenis Struktur Data Non-linear

Struktur data non-linear adalah struktur data yang tempat elemen-elemennya tidak berurutan atau tidak linear.

1. Tree



Tree adalah tipe struktur data yang memiliki bentuk seperti pohon. Tipe tree efisien untuk menyimpan data secara hierarkis karena disusun dalam berbagai level. Jadi, tipe ini sering dianggap sebagai kumpulan node yang saling dihubungkan.

Setiap node bisa berisi beberapa data atau link dari node lainnya. Beberapa istilah yang ada pada tipe tree antara lain:

* Root: node yang berada di paling atas..
* Child node: Turunan dari setiap node.
* Parent node: node yang berisi sub-node.
* Siblings: node yang berasal dari parent node yang sama.
* Leaf node: node yang tidak memiliki turunan lagi

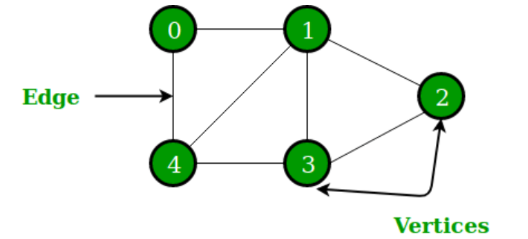
Keunggulan tipe tree:

* Proses mencari data bisa dilakukan dengan cepat.

Kekurangan tipe tree:

* Membutuhkan waktu yang lebih lama untuk memasukkan data karena harus menyesuaikan dengan urutan nilainya.

1. Graph



Graph adalah tipe yang berisi beberapa node yang saling terhubung. Node pada tipe graph disebut sebagai simpul. Jadi, setiap garis akan saling menghubungkan dua simpul. Biasanya tipe ini digunakan untuk menunjukkan jaringan tertentu. Contohnya seperti jaringan telepon.

Tipe graph terbagi menjadi dua jenis jenis, yaitu directed graph dan undirected graph. Directed graph artinya setiap garis akan terhubung ke semua simpul. Sedangkan undirected graph artinya tidak semua simpul akan terhubung dengan garis. Jika sebuah simpul tidak terhubung dengan simpul lainnya, maka disebut dengan isolated vertex.

Keunggulan tipe graph:

* Dapat membantu memeriksa hubungan antar node dengan cepat.

Cocok digunakan untuk grafik yang tidak mengandung banyak node.

Kekurangan tipe graph:

* Membutuhkan waktu lama untuk memodifikasi data.

Mengamankan Data

Mengamankan data dalam berbagai struktur data adalah langkah penting untuk melindungi privasi pengguna dan mencegah akses yang tidak sah. Berikut adalah beberapa langkah dan praktik umum untuk meningkatkan keamanan data:

1. Enkripsi Data
2. Manajemen Akses

* Terapkan kontrol akses yang ketat untuk memastikan hanya pengguna yang berwenang yang memiliki akses ke data.
* Berlakukan konsep kebutuhan-prinsip terhadap akses data, yaitu setiap pengguna hanya diberi akses ke data yang benar-benar diperlukan untuk pekerjaan mereka.

1. Otentikasi yang Kuat

* (MFA) untuk memastikan bahwa hanya orang yang sah yang dapat Menerapkan otentikasi dua faktor (2FA) atau otentikasi multi-faktor mengakses data.
* Memaksa pengguna untuk menggunakan kata sandi yang kuat yang sulit untuk ditebak.

1. Audit Log

* Membuat log audit yang mencatat setiap tindakan yang terkait dengan data, termasuk siapa yang mengaksesnya, kapan, dan apa yang dilakukan.
* Memeriksa log audit secara berkala untuk mendeteksi aktivitas yang mencurigakan atau akses yang tidak sah.

1. Pemutusan Akses

* Memiliki mekanisme untuk secara cepat memutuskan akses pengguna yang tidak sah atau ketika ada ancaman keamanan yang diidentifikasi.

1. Pembaruan Perangkat Lunak dan Keamanan

* Menjaga sistem dan perangkat lunak yang digunakan terbaru dengan menerapkan pembaruan keamanan yang dikeluarkan oleh penyedia perangkat lunak.
* Melakukan pemindaian kelemahan keamanan secara berkala.

1. Proteksi Fisik dan Fisik Logis

* Mengamankan perangkat keras dan infrastruktur fisik tempat data disimpan.
* Menggunakan firewall, jaringan terisolasi, dan kontrol akses fisik untuk mencegah akses yang tidak sah.

1. Pelatihan Karyawan

* Memberikan pelatihan kepada karyawan tentang praktik keamanan data yang baik, seperti tidak membagikan kata sandi, mengenali serangan phishing, dan melaporkan insiden keamanan.

1. Kepatuhan Hukum dan Peraturan

* Mematuhi peraturan keamanan data yang berlaku, seperti Regulasi Umum Perlindungan Data (GDPR) di Uni Eropa atau Health Insurance Portability and Accountability Act (HIPAA) di Amerika Serikat.

1. Penilaian Keamanan Berkala

* Melakukan penilaian keamanan berkala dan audit keamanan oleh pihak eksternal atau internal untuk mengevaluasi dan memperbaiki kelemahan keamanan.

1. Pemulihan Data dan Rencana Darurat

* Membuat rencana pemulihan bencana dan darurat untuk memastikan data dapat dipulihkan jika terjadi gangguan atau kerusakan.
* Melindungi privasi pengguna dan mencegah akses yang tidak sah memerlukan upaya yang berkelanjutan dan perhatian terus-menerus terhadap keamanan data. Pendekatan yang komprehensif untuk keamanan data adalah kunci untuk menjaga data tetap aman.

**BAB III**

**PENUTUP**

Kesimpulan

Dalam kesimpulannya, berikut adalah beberapa poin penting tentang algotitma dan struktur data

1. Algoritma adalah urutan langkah-langkah logis yang sistematis dan terstruktur yang dirancang untuk menyelesaikan suatu masalah atau tugas tertentu dalam komputasi atau matematika.
2. Struktur data adalah cara data disimpan, diorganisasi, dan diakses dalam komputer. Struktur data yang tepat dapat meningkatkan efisiensi dalam operasi data, mengurangi penggunaan memori yang tidak perlu, dan mendukung pengembangan perangkat lunak yang mudah dipelihara.
3. Algoritma terdiri dari serangkaian langkah-langkah logis yang harus diikuti oleh komputer untuk mencapai hasil yang diinginkan. Setiap langkah memiliki tujuan tertentu.
4. Algoritma harus dapat dibaca, dipahami, dan direview oleh orang lain. Kode yang dihasilkan dari algoritma harus mudah dipelihara.
5. Pemilihan struktur data yang tepat sangat penting dalam pengembangan perangkat lunak.
6. Pemahaman yang baik tentang berbagai jenis struktur data dan kapan harus menggunakannya adalah keterampilan penting dalam pemrograman.
7. Kadang-kadang, algoritma perlu dioptimalkan untuk meningkatkan kinerja atau mengatasi masalah dalam waktu yang lebih singkat. Ini melibatkan analisis kompleksitas algoritma.

**DAFTAR PUSTAKA**

https://www.niagahoster.co.id/blog/algoritma-pemrograman/

https://www.dicoding.com/blog/flowchart-adalah/

https://legblitz.blogspot.com/2016/09/pengertian-simbol-jenis-jenis-flowchart.html

https://revou.co/kosakata/struktur-data#apa-itu-struktur-data

https://www.pinhome.id/blog/contoh-pseudocode/