Adrian Sondang Irawan

3123500038 / D3 IT B

1. **Apa itu paradigma pemrograman dan sebutkan beberapa contohnya?**

   Paradigma pemrograman adalah pendekatan, gaya, klasifikasi atau metodologi dalam penyelesaian suatu permasalahan dengan bahasa pemrograman. Ada beberapa macam paradigma pemrograman yaitu :

   a. **Pemrograman prosedural/imperatif** : Paradigma ini berfokus pada penyelesaian masalah dengan cara mengeksekusi prosedur atau langkah secara berurutan. Contoh bahasa pemrograman yang menggunakan metode ini adalah bahasa C, Pascal, BASIC dan Cobol.
   b. **Paradigma Fungsional** : Paradigma ini berfokus pada fungsi-fungsi sebagai unit utama untuk menyelesaikan masalah-masalah yang tersedia. Contoh bahasa pemrogramannya yaitu LOGO, APL, Haskell, dan Lisp
   c. **Paradigma Berbasik Objek (OOP)** : Paradgima ini memodelkan suatu program dengan kumpulan beberapa object yang memiliki atribut dan method. Contoh bahasa pemrogaman nya yaitu C++, Python, Java dan Delphi.
   d. **Paradigma declarative** : Paradigma ini menggambarkan apa yang harus dicapai daripada langkah-langkah spesifik untuk mencapainya. Ini sering menggunakan aturan, logika, atau kueri untuk menyatakan tujuan program, dan sistem komputasi menentukan cara terbaik untuk mencapainya. Contoh dari ini termasuk SQL untuk pemrograman database dan Prolog untuk pemrograman logika.
   e. **Paradigma Logical** : Paradigma ini didasari oleh konsep pendefinisian relasi antar individu yang dinyatakan sebagai predikat. Contoh bahasa pemrograman yang menggunakan paradigma ini adalah bahasa pemrograman Prolog.

2. **Apa perbedaan antara bahasa pemrograman kompilasi dan bahasa pemrograman interpretasi?**

   Definisi dari kompilasi dan interprestasi adalah

   a. Bahasa kompilasi adalah bahasa pemrograman yang membutuhkan suatu kompiler dalam menjalankkan programnya. Kompiler sendiri adalah alat yang digunakan untuk menerjemahkan bahasa tingkat tinggi ke bahasa mesin dengan cara menerjemahkan sebelum eksekusi programya. Kelebihan dari bahasa kompilasi sendiri adalah :
      - Program lebih cepat saat dijalankan karena tidak perlu terjemah code program saat dijalankan
      - Compiler dapat menghasilkan kode mesin yang lebih optimal, sehingga dapat menghasilkan program yang lebih cepat dan efisien.
      - Compiler lebih cocok untuk mengembangkan kode program yang bersifat statis, seperti bahasa pemrograman yang digunakan untuk mengembangkan sistem operasi dan perangkat lunak tingkat rendah.

   b. Bahasa interprestasi adalah bahasa pemrograman yang dijalankan dengan cara diterjemahkan dari bahasa tingkat tinggi ke bahasa mesin saat code program di jalankan. Contoh bahasa pemrogramannya yaitu PHP, Javascript, Python, dan Ruby. Kelebihan bahasa interprestasi adalah :
      - Lebih mudah untuk debugging, karena kesalahan dapat diidentifikasi dan diperbaiki saat kode program dijalankan.
      - Lebih portabel, karena dapat digunakan untuk menjalankan kode program di berbagai platform.

- Lebih mudah untuk mengembangkan kode program yang bersifat dinamis, seperti bahasa pemrograman skrip.

3. Jelaskan konsep modularitas dalam pemrograman dan mengapa itu penting?

Konsep modularitas dalam pemrograman adalah kemampuan untuk membagi program menjadi bagian lebih kecil yang independen dan dapat digunakan kembali. Bagian-bagian ini disebut modul, kelas, fungsi, prosedur atau paket tergantung paradigma pemrogramannya.

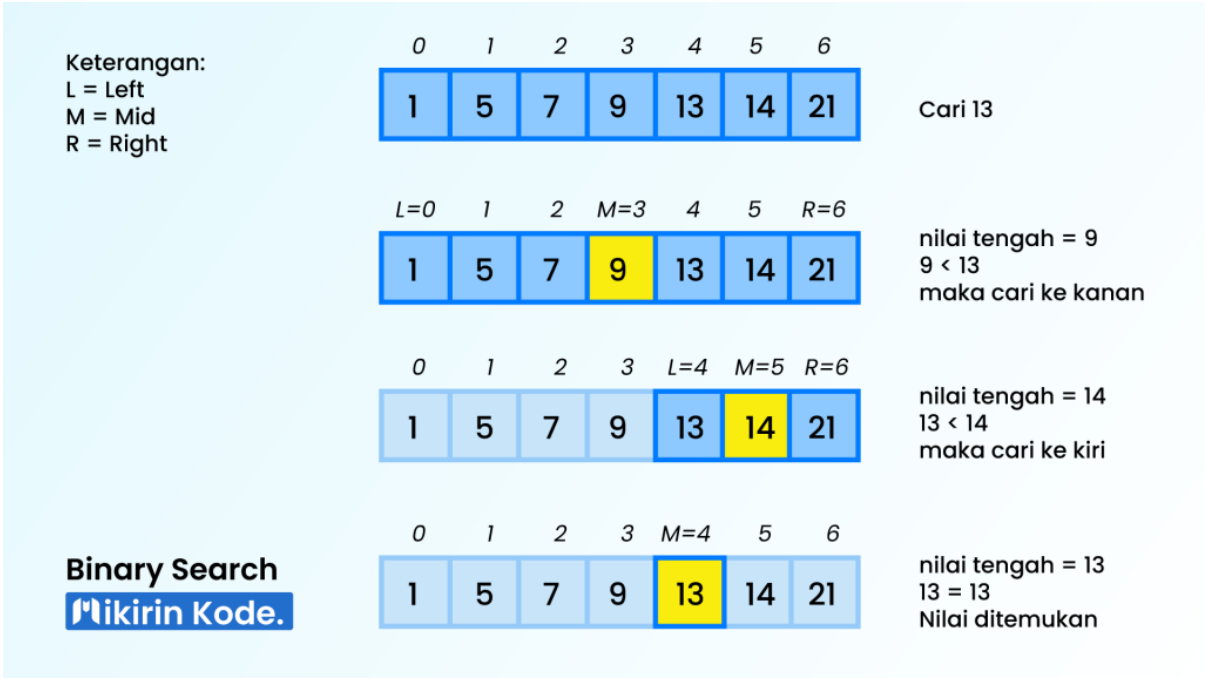
Konsep modularitas dalam pemrograman ini sangat penting karena dapat memudahkan developer saat memaintenance, developing, dan reusable code serta meningkatkan kualitas dan konsistensi program. Sehingga bila melakukan kolaborasi dengan developer lain akan memudahkan dan efektif dalam pengembangannya karena memungkinkan untuk tim bekerja secara terpisah pada modul yang berbeda.

4. Apa yang dimaksud dengan algoritma pencarian biner? Bagaimana cara kerjanya?

Algoritma binary search digunakan untuk mencari data pada list atau array yang sudah terurut. Algoritma ini bekerja dengan membandingkan nilai data yang dicari dengan nilai tengah list atau array. Jika nilai data yang dicari lebih besar dari nilai tengah, maka algoritma akan mencari pada setengah kanan list atau array, dan begitu juga sebaliknya. Berikut cara kerja binary search :

- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kanan dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kanan dengan acuan posisi data tengah akan menjadi posisi awal untuk pembagian tersebut.
- Apabila data yang dicari lebih kecil dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kiri dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kiri. Dengan acuan posisi data tengah akan menjadi posisi akhir untuk pembagian selanjutnya.
- Apabila data belum ditemukan, maka pencarian akan dilanjutkan dengan kembali membagi data menjadi dua.
- Namun apabila data bernilai sama, maka data yang dicari langsung ditemukan dan pencarian dihentikan.

Berikut ilustrasi binary search :



- Terdapat sebuah array yang menampung 7 elemen seperti ilustrasi di atas
- Nilai yang akan dicari pada array tersebut adalah 13
- Jadi karena konsep dari binary search ini adalah membagi array menjadi dua bagian, maka pertama tama kita cari nilai tengahnya dulu
- total elemen dibagi 2 yaitu 7/2 = 4.5 dan kita bulatkan jadi 4. M
- Maka elemen ke empat pada array adalah nilai tengahnya, yaitu angka 9 pada indeks ke 3
- Kemudian kita cek apakah 13 > 9 atau 13 < 9?
- 13 lebih besar dari 9, maka kemungkinan besar angka 13 berada setelah 9 atau di sebelah kanan
- Selanjutnya kita cari ke kanan dan kita dapat mengabaikan elemen yang ada di kiri
- Setelah itu kita cari lagi nilai tengahnya, didapatlah angka 14 sebagai nilai tengah
- Kita bandingkan apakah 13 > 14 atau 13 < 14?
- Ternyata 13 lebih kecil dari 14, maka selanjutnya kita cari ke kiri
- Karna tersisa 1 elemen saja, maka elemen tersebut adalah nilai tengahnya
- Setelah dicek ternyata elemen pada indeks ke-4 adalah elemen yang dicari, maka telah selesai proses pencariannya.

5. Apa perbedaan antara sintaksis dan semantik dalam konteks pemrograman?

1. Sintaksis
   - **Definisi**: Sintaksis merujuk pada aturan-aturan formal yang mengatur bagaimana kode program ditulis atau ditulis dalam bahasa pemrograman tertentu.

   - **Contoh**: Aturan-aturan sintaksis mencakup tata letak kode, penggunaan tanda baca, pemakaian kata kunci, dan struktur kode yang diperlukan. Misalnya, dalam bahasa pemrograman Python, sintaksis yang benar memerlukan penggunaan indentasi yang konsisten dan pemakaian tanda titik dua setelah struktur kontrol seperti pernyataan if atau loop.

- **Peran**: Sintaksis penting karena menentukan apakah kode program dapat dikompilasi atau diinterpretasi tanpa kesalahan. Kesalahan sintaksis mengindikasikan bahwa kode tidak mematuhi aturan sintaksis bahasa pemrograman yang dipakai.

2. Semantik
   - **Definisi**: Semantik mengacu pada arti atau makna dari kode program. Ini berkaitan dengan cara program diterjemahkan oleh komputer dan bagaimana hasilnya dijalankan.

   - **Contoh**: Semantik mencakup perilaku dan efek dari instruksi atau pernyataan dalam kode program. Misalnya, jika dalam bahasa pemrograman Java terdapat pernyataan x = y + z, semantiknya adalah bahwa nilai dari variabel y dan z akan ditambahkan dan hasilnya akan disimpan dalam variabel x. contoh lain yaitu dalam c ++ variabel "s" dideklarasikan sebagai "int s;", untuk menginisialisasi kita harus menggunakan nilai integer. Alih-alih menggunakan integer, kami telah menginisialisasi dengan "Tujuh". Deklarasi dan inisialisasi ini secara sintaksis benar tetapi secara semantik salah karena "Tujuh" tidak mewakili bentuk integer.

   - **Peran**: Semantik sangat penting karena menentukan apakah program berperilaku sesuai dengan yang diharapkan dan mencapai tujuan yang diinginkan. Kesalahan semantik dapat mengakibatkan program tidak berfungsi dengan benar atau bahkan menghasilkan perilaku yang tidak terduga.