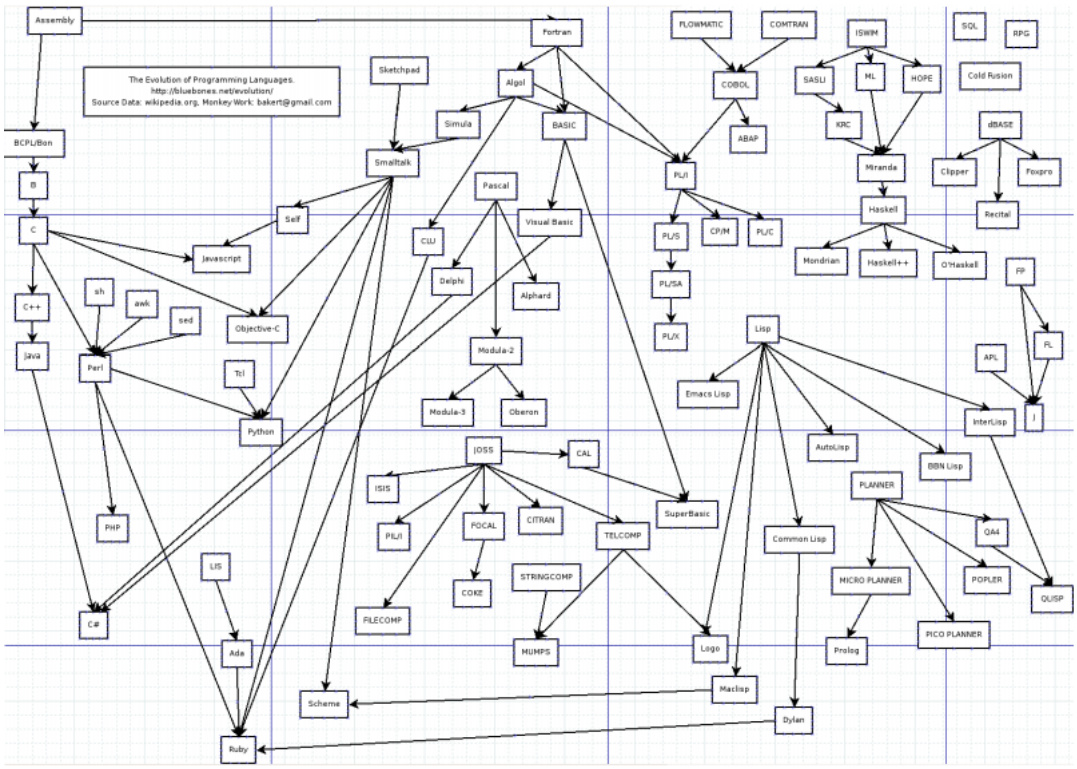
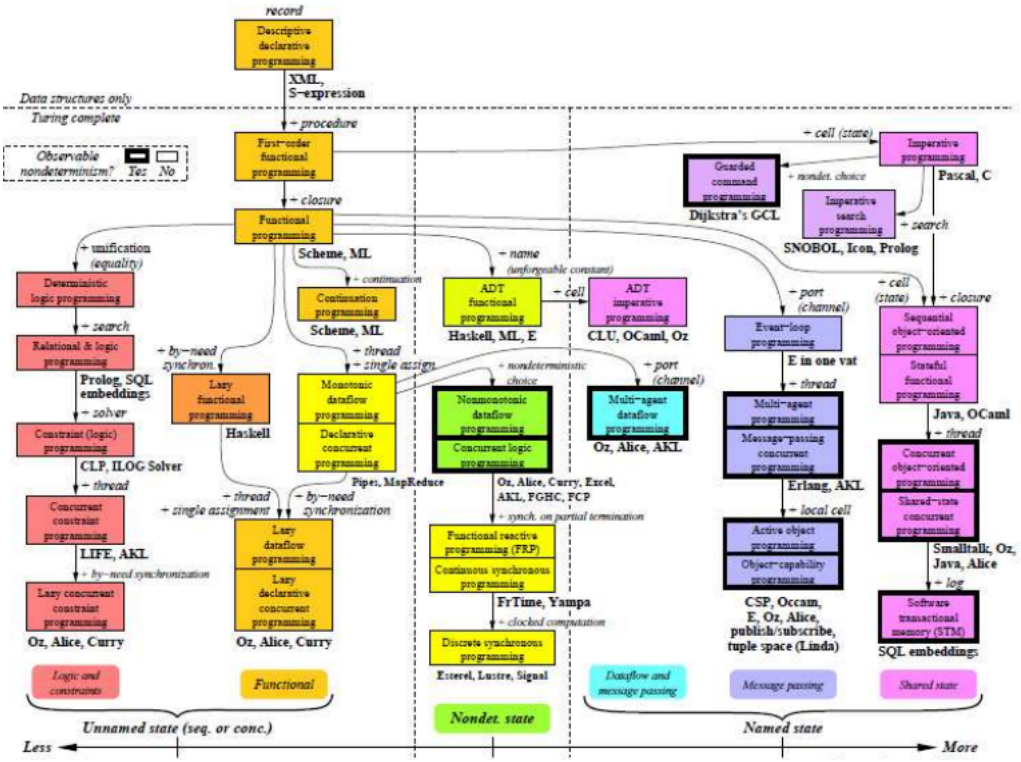
# 1.Paradigma Pemprograman

## Paradigm

Terdapat ribuan bahasa pemprograman :

1. Dalam ensiklopedia Britannica ada lebih dari 2000 bahasa
2. Sejak Mei 2006, Dari ensiklopedia computer language (Diarmuid Pigott's) di universitas Murdoch Australia terdapat 8000 Bahasa.
3. Masih banyak yang baru dibuat setiap tahunnya, beberapa bhasa pernah menjadi populer, tetapi beberapa orang profesional hanya menggunakan lusinan untuk kariernya.






Pada bahasa pemprograman tingkat rendah tidak ada abstraksi pada mesin code dan perakitannya, yang jadinya sulit untuk digunakan, tetapi memungkinkan untuk memprogram secara efisien dan dengan sedikitnya memory footprint.

Pada bahasa pemprograman tingkat tinggi, mengisolasi eksekusi dari semantik program komputer sehingga dapat menyederhanakan pengembangan program.

Paradigma pemprograman adalah hasil dari pemikiran orang tentang bagaimana program komputer harus dibuat pola yang berfungsi sebagai "Aliran pemikiran" untuk pemprogramannya.

Menurut L Flon "Tidak akan pernah ada, bahasa pemprograman yang paling sulit untuk menulis program yang buruk"

Paradigma :

- contoh sebagai pola atau model (The American Heritage Dictionary of The English Language)
- muncul sebagai hasil proses sosisal dimana orang mengembangkan ide dan menciptakan prinsip dan prakter yang mewujudkan ide-ide tersebut. (Thomas Kuhn)

## Principal Programming Paradigms

- **Imperative/procedural = Perl**

berdasarkan prosedural, setiap prosedur dapat dipanggil kapan saja selama program berlangsung, mendefinisikan setiap langkah dalam pelaksanaan tugas, serta melakukan urutan langkah algoritmik(sequence step).

manfaat : kompleksitas sedang, dapat reused code, kemampuan menjadi sangat modular dan terstruktur.

scoping merupakan teknik menjaga prosedur sangat modular. mencegah prosedur mengakses variable prosedur lain. interface yang sederhana.

Fokus dari pemrograman prosedural adalah untuk memecah tugas pemrograman.
menjadi kumpulan variabel, struktur data, dan subrutin. Penggunaan pemrograman prosedural
prosedur untuk beroperasi pada struktur data,

- **Functional = Haskell**

Penggunaan fungsi matematika. LIPS adalah bahasa pertama dari functional programming.

bahasa ML(yg dibuat 1970 oleh Robin Milner) akhirnya berkembang menjadi beberapa dialek, yang paling umum adalah sekarang Objective Caml, Standard ML, dan F #. Haskell rilis 1980.

iterasi dalam functional biasanya berupa rekursi. Tail recursion dapat diimplementasikan agar lebih efektif.

- **Object-Oriented = Prolog**

Paradigma penggunaan objek, mencakup fitur abstraksi data, enkapsulasi, modularitas, polimorphism dan ineritance.

pemrograman berorientasi objek itu adalah untuk memecah tugas pemrograman menjadi objek dengan setiap "objek" merangkum data dan metodenya sendiri (subrutin). OOP memiliki struktur data sendiri.

- Pada Class mendefiniskan karakteristik abstrak dari suatu benda.
- Pada object ialah individu dari kelas yang dibuat pada objek run-time, ia membentuk sebuah state dan behavior yang di defined dalam object's class
- Atribut atau variabel anggota.
- Method adalah subrutin yang eksklusif diasisiasikan baik dengan kelas atau dengan objek.
- method → object , class&static→ class

OO tapi prosedural : C++, C#, Java, Scala, Python.
prosedural tetapi memiliki fitur OO: VB.NET (derived from VB), Fortran 2003, Perl, COBOL 2002, PHP.
Sebagian besar OO: Oberon (Oberon-1 or Oberon-2).
Dengan dukungan tipe data yang abstrak tetapi tidak semua OO: Modula-2, Pliant, CLU.

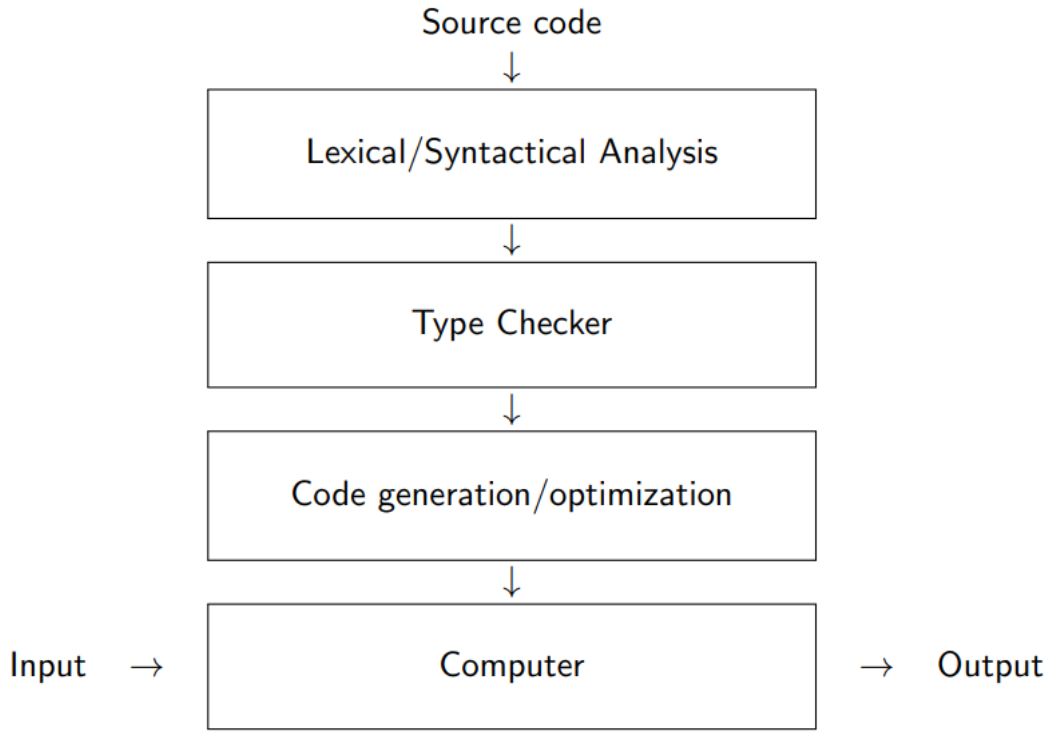
- **Concurrent = Erlang**
- **Logic**
- **Scripting**

pada kenyataannya hanya sedikit bahasa pemprograman yang benar-benar "Pure", kebanyakan ialah combine dari beberapa paradigma yang berbeda.

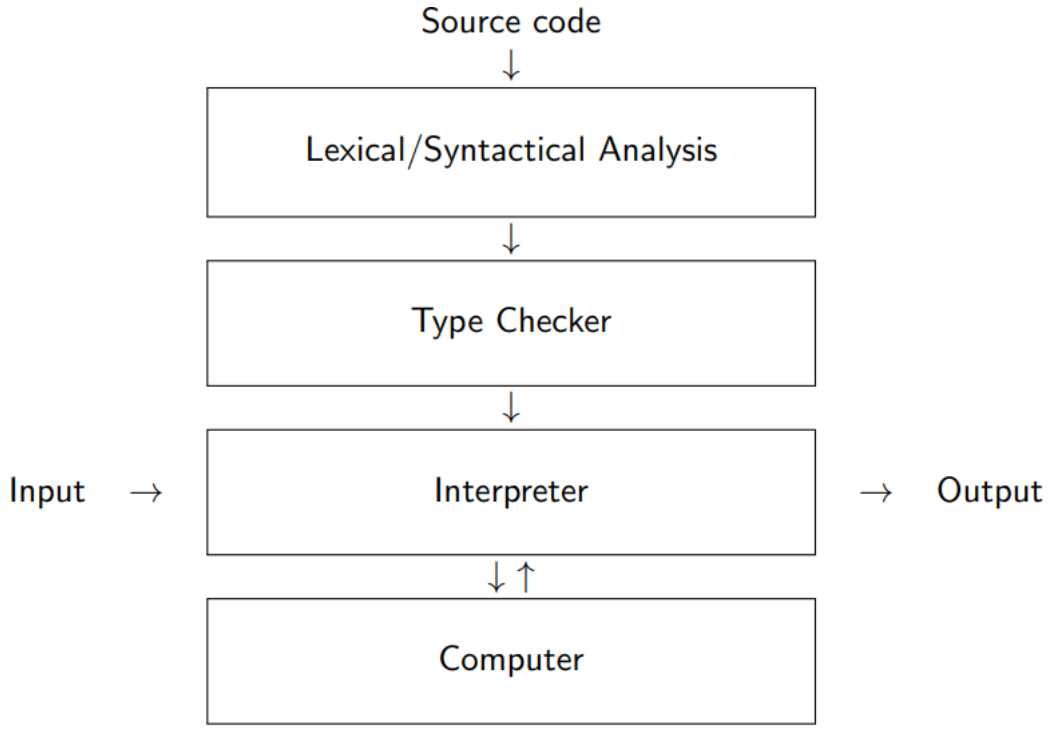
### Elements of Programming Languages

- Compiled/Interpreted
- Syntax
- Semantics
- Typing

Compiled : bahasa di terjemahkan ke dalam kode mesin yang dapat dijalankan langsung di prosessoe komputer, biasanya seluruh program diterjemahkan sebelum dijalankan




Interpreted : Bahasa diproses oleh mesin virtual tigkat yang lebih tinggi, di terjemahkan dengan cepat, yaitu oernyataannya diterjemahkan dan langsung dieksekusi.




- Syntax dalam bahasa pemprograman biasanya digambarkan dengan formalisme (tata bahasa/grammar)
- Semantiks : jauh lebih sulit didefinisikan daripada sintax.
  - Semantik operasinal : mesin abstraksi, sistem transisi
  - semantik aksiomatik : inferensi logis untuk mendefinisikan bahasa

- semantik denotasional : menerjemahkan menjadi frasa dalam bahasa lain. (formalisme matematika)

- Typing : Type membantu untuk mendesign program, sheck kebenaran, menentukan kebutuhan penyimpanan. terdapat DataType yang membantu mendefinisikan sebuah operasi.
- Type Casting : Dalam beberapa bahasa memungkinkan untuk konversi data type inplisit dilakukan oleh kompiler. Contohnya

```
double da = 3.3;
double db = 3.3;
double dc = 3.4;
int result1 = (int)da + (int)db + (int)dc; //result == 9
//or
int result2 = da + db + dc; //result == 10
```

**Static Vs Dinamic Typing**

- Static :

  +lebihsedikit rawan kesallahan

  -terkadang bersifat membatasi

- Dinamic :

  +lebih fleksibel

  -Sulit untuk di debug ketika ada kesalahan.

==========

## Latihan soal

1.Jelaskan apa yang dimaksud dengan paradigma pemrograman ? mengapa bahasa pemrograman sebaiknya dapat men-support multi paradigm ?

Paradigma pemprograman ialah pendekatan untuk memprogram komputer berdasarkan teori matematika atau sekumpulan prinsip yang koheren.Setiap paradigma men-support multi paradigma, agar bisa menyediakan kerangka kerja dimana para programmer dapat bekerja dalam berbagai gaya, secara bebas mencampurkan konstruksi dari paradigma yang berbeda, dan sebaai solusi bahwa pada beberapa tahap satu paradigma tidak mampu menyelesaikan semua masalah dengan cara termudah atau paling efisien. Misalnya, pemprograman berorientasi objek paling baik untuk masalah dengan jumlah besar abtraksi data terkait yang diatur dalam hierarki. Pemprograman logika paling baik untuk mengubah atau menavigasi struktur simbolik yang kompleks sesuai dengan aturan logis. Pemrograman sinkronis diskrit paling baik untuk masalah reaktif, yaitu masalah yang terdiri dari reaksi terhadap rangkaian peristiwa eksternal.

2. Jelaskan apa yang dimaksud dengan “Paradigm’s Kernel Language” ?

Pada setiap paradigma didefinisikan oleh sekumpulan konsep pemrograman, yang diatur ke dalam bahasa inti sederhana yang disebut Paradigm’s Kernel Language. Karena terdapat banyak sekali bahasa pemprograman, tetapi hanya sedikit paradigmanya.

3.Jelaskan apa yang dimaksud dengan *creative extension principle* ?

Sebuah konsep tidak digabungkan secara sembarangan untuk membentuk sebuah paradigma, mereka diatur menurut *creative extension principle*. Prinsip ini pertama kali didefinisikan oleh Felleisen, dalam hal ini memberi kita panduan untuk menemukan keteraturan dalam kumpulan luas kemungkinan dalam paradigma. Dalam paradigma tertentu, dapat terjadi bahwa program menjadi rumit karena alasan teknis yang tidak memiliki hubungan langsung dengan masalah spesifik yang sedang dipecahkan. Untuk menunjukkan bagaimana prinsip tersebut bekerja, asusmsikan telah memiliki paradigma pemprograman fungsional sekuensial sederhana.

4.Elemen apa saja yang menjadi bagian penting pada *Programming Concept* ?

Paradigma Pemprograman dibangun dari konsep pemprograman. Terdapat 4 elemen penting dalam konsep pemprograman, yaitu Records (rekam catatan), lexically scoped closure, independence (concurrency), and named state.

- Record : adalah struktur data, mengelompokkan referensi ke item data dengan akses yang diindeks setiap item. Record adalah dasar dari pemprograman simbolik. Pemprograman simbolik ialah mampu menghitung dengan records: membuat records baru, menguraikannya dan memeriksanya. Banyak struktur data penting seperti array,

string, tree, hash bisa diturunkan dari record. Jika digabungkan dengan closure, record bisa digunakan untuk pemprograman berbasis komponen.

- Lexically scoped closure : konsep yang sangat kuat yang ada di pusat pemprograman. Pemprograman fungsional, dimana pemprograman dengan closure adalah paradigma sentral. Dari sudutpandang implementasi, closure menggabungkan prosedure dengan referensi eksternalnya ( referensi yang digunakan pada definisinya). Dari sudutpandang programmer, closure adalah "paket kerja" yaitu sebuah program dapat berubah ke instruksi apapun ke dalam closure pada satu titik dalam program, teruskan ke titik lain dan emutuskan untuk menjalankannya pada saat itu. Hasil eksekusinya sama seperti jika instruksi dijalankan pasa saat penutupan dibuat.

Banyak kemampuan yang biasanya dikaitkan dengan paradigma tertentu didasarkan pada closure:

- Instansiasi dan kemurahan hati, biasanya terkait dengan program berorientasi
- Pemisahan masalah, niasanya terkait dengan pemprograman berorientasi aspek, bisa di sapat dengan mudah dengan menulis fungsi yang menggunakan fungsi lain sebagai argumen.
- Pemprograman berbasis komponen adalah gaya pemprograman dimana program berada disusun sebagai komponen, di mana setiap komponen dapat bergantung pada komponen lainnya.
- Independence (concurrency)

Membangun sebuah program adalah sebagai bagian dari independen. Misalnya, pertimbangkan program yang terdiri dari instruksi mengeksekusi satu demi satu. Untuk mengimplementasikan independensasi kita membutuhkan konsep pemprograman baru yang disebut konkurensi. Ketika dua bagian tidak berinteraksi sama sekali, kami mengatakan keduanya bersamaan. Bagian yang bersamaan dapat diperpanjang untuk memiliki beberapa interaksi yang terdefinisi dengan baik, yang disebut komunikasi.

Terdapat tiga tingkatan konkurensi :

- Sistem terdistribusi : sekumpulan komputer yang terhubung melalui jaringan
- Sistem operasi : perangkat lunak yang mengelola komputer.
- Aktivitas dalam satu proses Perbedaan mendasar antara proses dan utas adalah bagaimana alokasi sumber daya Konkurensi tingkat proses terkadang disebut konkurensi kompetitif: masing-masing Proses mencoba untuk memperoleh semua sumber daya sistem untuk dirinya sendiri. Kepala sistem operasiperannya adalah untuk menengahi permintaan sumber daya yang dilakukan oleh semua proses dan mengalokasikan sumber daya dengan cara yang adil. Konkurensi tingkat utas terkadang disebut konkurensi kooperatif
- Named state and modularity : state memperkenalkan abstrak gagasan dalam program, dalam program fungsional, tidak ada pengertian tentang waktu. Fungsi adalah fungsi matematika. Named state penting untuk modularity sistem. Kami mengatakan bahwa sistem (fungsi, prosedure, komponen, dll) bersifat modular jika pembaruan dapat dilakukan pada bagian dari sistem tanpa ngengubah bagian sistem lainnya. Keuntungan utama dari named state adalah program menjadi modular. Kerugian utamanya adalah program bisa menjadi tidak benar. Maka dari itu nama dari setiapnya haruslah berbeda.

5.Apa yang dimaksud deterministik dan non deterministik pada *concurrent programming*, berikan contohnya?

Nondeterministik ialah kejadian jika pada titik tertentu selama eksekusi ada pilihan apa yang harus dilakukan selanjutnya.

Salah satu masalah utama dari pemprograman concurrent iadalah nondeterministik. Eksekusi program non deterministik jika dibeberapa titik selama eksekusi ada pilihan apa yang harus dilakukan selanjutnya. Non deterministik muncul secara alami ketika ada konkurensi, sejak dua aktifitas bersamaan adalah independen, spesifikasi program tidak dapat mengatakan yang mana dieksekusi terlebih dahulu. Bagaimana cara kita menghindari efek buruk dari non deterministik dan masih memiliki konkurensi? Kita dapat menyelesaikan masalah dalam dua langkah

- Pertama membatasi nondeterministik yang dapat diamati pada bagian bagian program yang benar-benar membutuhkannya. Bagian lain seharusnya tidak memiliki nondeterminisme yang dapat diamati
- Mendefinisikan bahasa sehingga memungkinkan untuk menulis program secara bersamaan tanpa nondeterministik yang dapat diamati.

contoh nondeterminsitik terjadi jika :

C ={newCell 0}

thread c=1

thread c=2

maka dari itu hindari nondeterministik dalam bahasa yang berbarengan

Model deterministik tidak termasuk keacakan sama sekali. Semuanya terjadi secara ketat sesuai dengan rencana awal dengan akurasi tak terbatas. Tidak ada input lebih lanjut selama runtime diizinkan. Tidak ada jenis ketidaktepatan, kebisingan atau ketidakpastian, tidak ada konsep alternatif, tidak ada pertanyaan, tidak ada jawaban, tidak ada agen yang mampu membuat keputusan.

6.Dalam taksonomi paradigma pemrograman apakah yang membedakan antara *FIrst-order functional programming* dengan *functinnal programing*?

Yang membedakan antara *First-order functional programming* dengan functional programming dalam ruang lingkup taksonominya ialah pada closurenya. Closure itu sendiri ialah teknik pengimplementasian *lexically scoped name* serta sebuah rekaman yang menyimpan fungsi bersamaan dengan environment.

7.Dalam taksonomi, paradigma di bedakan kedalam *observable nondeterminism* dan *non observable nondeterminism*. *Observable nondeterminism* yang dimaksud dalam taksnonomi ini adalah apa?

*Observable nondeterminism* dalam taksonomi penanda bahwa paradigma tersebut merupakan nondeterminism yang dapat diamatiataukah tidak. Dengan demikian, *observable nondeterminism* didukung hanya jika diperlukan. Hal itu berlaku pada concurrent programming.

8.Jelaskan yang dimaksud dengan *Data Abstraction* ?

Data Abstraction adalah suatu cara untuk mengatur penggunaan struktur data secara tepat aturan yang menjamin bahwa struktur data digunakan dengan benar. Abstraksi data memiliki bagian dalam, bagian luar, dan interface diantara keduanya, semua struktur data disimpan didalam. Bagian dalam tersembunyi diluar, semua operasi pada data harus melalui interface.

9.Sebutkan apa saja keunggulan dalam menerapkan *Data Abstraction* !

- Abstraksi data ini akan selalu berfungsi dengan baik.
- Program akan lebih mudah dipahami.
- Mungkin untuk mengembangkan program yang sangat besar.

10.Konsep *abstraction* pada paradigma berorientasi *object* diimplementasikan dalam ?

Implementasikan pada OOP (Object Oriented Programming) dimana paradigma pemprograman berdasarkan konsep"objek", yang dapat berisi data dalam bentuk field (atribut), serta kode dalam bentuk fungsi/prosedur atau juga dikenal sebagai method. Konsep dari OOP ini sendiri ialah cara berpikir tentang aplikasi yang mempelajari untuk berpikir bahwa aplikasi bukan sekedar prosedur melainkan sebagai object. Object yang dimaksud disini memiliki pengertian suatu modul yang mengkombinasikan antara data dan kode program yang bekerja sama dalam program dengan melewatkan proses satu sama lain. OOP lebih memfokuskan kepada manipulasi object. OOP memiliki karakteristik seperti : Abstraction, encapsulation, inheritance, dan polymorphism. Pada konsep abstraction ini yang berorientasikan object ialah polymorphism dan inheritance