**Kelompok 5**

* **Fernando Christyanto**
* **Devin Dwitama Loputra**
* **Roberto Ewaldo**
* **Daniel**
* **Kevin Hermawan**
* **Joseph MP**

**PERTEMUAN 1**

Programming domain terbagi menjadi

* Scientific applications
* Business applications
* Artificial intelligence
* Systems programming
* Web Software

Language evaluation criteria terbagi menjadi

* **Readability**: the ease with which programs can be read and understood
* **Writability**: the ease with which a language can be used to create programs
* **Reliability**: conformance to specifications (i.e., performs to its specifications)
* **Cost**: the ultimate total cost

Kita juga mempelajari metode metode implementasi yaitu Compilation, Pure implementation, dan Hybrid implementation system

Compilation adalah code diterjemahkan menjadi Bahasa mesin. Pengerjaannya lambat namun eksekusinya cepat

Fase fase dalam compilation :

* + lexical analysis: converts characters in the source program into lexical units
  + syntax analysis: transforms lexical units into *parse trees* which represent the syntactic structure of program
  + Semantics analysis: generate intermediate code
  + code generation: machine code is generated

Pure implementation adalah program diterjemahkan kedalam Bahasa lain dengan program interpreter. Ini kurang efisien.

Hybrid implementation system berada ditengah tengah antara Compilation dan Pure Implementation

**PERTEMUAN 2**

DEFINISI :

* Bahasa pemrograman merupakan notasi formal.
* Mempunyai 2 komponen utama, yaitu Syntax dan Semantic.
* Syntax : Kumpulan aturan formal yang menspesifikasikan komposisi suatu program yang terdiri dari huruf, angka, dan karakter lain.
* Semantic : Mendefinisikan arti dari program yang benar secara syntax dari bahasa tersebut.

Kriteria desain bahasa pemrograman

* Abstraction : Kemampuan untuk mengkonstruksi suatu pola yang berulang-ulang.
* Simplycity : Kesederhanaan, semakin sedikit konsep untuk dapat dimengerti, semakin baik.
* Regularity : Keteraturan, semakin sedikit melanggar aturan semakin baik.
* Konsistensi : Ketetapan suatu alur atau bentuk bahasa.
* Translation : Penerjemah yang cepat dan menghasilkan kode tujuan yang efisien.

1. **SYNTAX**

* Kumpulan aturan yang mendefinisikan suatu bentuk bahasa.
* Syntax mendefinisikan bagaimana suatu kalimat dibentuk sebagai barisan/urutan dari pemilihan suatu kata dasar.

Kriteria Syntax :

* Readability
* Writeability
* Verifiability
* Kemudahan Translasi
* Pengurangan Ambiguitas

Lexical     : menspesifikasikan kumpulan karakter yang terdapat dalam alphabet dari bahasa dan cara supaya karakter-karakter tersebut dikombinasikan ke dalam kata-kata yang valid dan diterima.

Syntactic  : elemen-elemen yang mengatur bentuk suatu statement-statement dalam bahasa pemrograman.

**Elemen-elemen syntactic :**

* Himpunan Karakter
* Identifier
* Simbol untuk operator
* Keyword dan Reserved Word
* Komentar
* Blank (spasi)
* Delimiter dan Tanda Kurung
* Ekspresi

**2. SEMANTIC**

* Mendefinisikan arti dari program yang benar secara syntax dari bahasa tersebut.
* Tingkatan Translasi
* Translasi dibagi dua :

1. Analisa input program sumber.

**Analisa Lexical**

* Tahap dasar dari translasi, mengerjakan pengelompokkan urut-urutan karakter ke dalam komponen pokok: identifier, delimiter, symbol operator, angka, keyword, blank, komentar, dst.

**Analisa Syntactic**

* Biasa disebut parsing.
* Struktur program yang lebih besar diidentifikasi : statement, deklarasi, ekspresi, dll.

**Analisa Semantic**

* Pusat dari tahapan translasi.
* Struktur syntatic hasil dari syntatic analyzer diproses.
* Menghasilkan suatu kode objek yang executable sederhana.
* Akan dimanipulasi oleh tahap optimasi sampai jadi kode executable.

2. Synthesis program objek yang executable

Terdiri dari beberapa tahapan :

**Optimasi**

* Menghilangkan inefisiensi kode.
* Mengubah kode yang tidak efisien menjadi kode yang efisien.

**Penghasil Kode**

* Dibentuk sebagai bahasa Assembly, kode mesin atau program obyek lainnya.

**Linking dan Loading**

* Bersifat optional.
* Menggabungkan potongan-potongan kode yang dihasilkan dari translasi terpisah ke dalam program final executable yang utuh.

**3.GRAMMAR**

Suatu kumpulan aturan (production) yang menentukan urut-urutan karakter.

* Suatu formal grammar adalah grammar biasa yang ditentukan dengan menggunakan notasi yang ketat.
* Ada dua kelas grammar yang berguna untuk teknologi compiler :

1. **Context free grammar**
2. **BN Form**

**PERTEMUAN 3**

Bab ini memperkenalkan masalah semantik mendasar variabel. Ini dimulai dengan menggambarkan sifat nama dan kata-kata khusus dalam bahasa programming. Ada atribut dari variable, yaitu termasuk jenis, alamat dan nilai. Kemudian di diskusikan, termasuk issue tentang alias. Hal yang paling penting dari konsep binding dan binding diperkenalkan berikutnya, termasuk cara binding yang berbeda untuk atribut variabel dan bagaimana mereka mendefinisikan empat kategori yang berbeda dari variabel. Setelah itu, dua aturan scoping yang sangat berbeda untuk nama, statis dan dinamis, semuanya dijelaskan bersama dengan konsep lingkungan referensi dari sebuah pernyataan. Akhirnya konstanta bernama dan inisialisasi variabel akan dibahas.

Bahasa Imperative adalah abtsraksi dari arsitektur von neuman. Variable dapat di karakteristkkan dengan kumpulan properties, atau attribute, dan yang paling penting adalah tipe dari konsep bahasa pemrograman nya.

Names adalah sebuah string karakter yang digunakan untuk mengidentifikasi  beberapa entity dalam sebuah program. Sebuah names dapat mengandung kalimat, digit, atau pun underscore karakter (\_).

**Special Words**

Special words dalam bahasa pemrograman digunakan untuk membuat program lebih dapat dibaca dengan menamai aksi yang akan dilaksanakan. Digunakan juga untuk memisahkan antara statement dan program.

Keyword adalah kata dalam sebuah bahasa pemrograman yang special dan berada pada context tertentu saja.

Reserved word adalah special kata dari sebuah bahasa pemrograman yang tidak dapat digunakan sebagai nama. Dalam reserved word ada problema sebagai contoh :

Integer Real

Real Integer

Statement diatas mendeklarasikan program variable **real** itu adalah tipe **integer** dan  variable **integer** itu adalah tipe **real**

**Variable**

Variable adalah abstraksi dari memory cell computer atau kumpulan cell.

Variable attribute terdiri dari :

-          Name = nama dari sebuah variable, hamper setiap variable memiliki nama.

-          Address = address dari sebuah variable dalam mesin memory dimana dia di asosiasikan.

Asosiasi ini tidak se-simple yang di perkirakan, bisa saja variable yang sama di asosiasikan dengan alamat yang berbeda dan pada waktu yang berbeda pada sebuah program.

Aliases adalah ketika satu nama variable dapat mengakses lokasi memory yang sama. kelemahan aliasing adalah memperbolehkan variable untuk mengubah nilai berdasarkan tugas dari variable yang lain.

-          Type = variable menentukan range nilai dari variable yang di simpan dan set operasi yang mendefinisikan nilai dari type. Contoh : tipe int pada java memiliki range -2147483648 sampai 2147483647.

-          Value = variable content dari memory cell atau cell asosiasi dengan variable.

**Binding**

Binding adalah asosiasi antara attribute dan entity, seperti antara variable dengan tipe nya atau nilai, atau antara sebuah operasi dan symbol. Binding time adalah aktu yang dibutuhkan dalam melakukan binding.

Sebagai contoh :

count = count + 5;

tipe count  binding dengan waktu compile.

Binding dapat ditentukan tipe nya.  Static binding yaitu terjadi sebelum program berjalan, dan tetap sama selama program eksekusi.  Dynamic binding yaitu binding terjadi ketika program sedang berjalan dan nilai dapat berubah selama program dieksekusi.

Storage binding and lifetime

Ada 2 tipe storage binding yaitu :

Allocation : mengambil cell dari kolam memori cell yang tersedia

Deallocation : sel memori yang telah tidak terikat dari variabel dikembali ke kolam memori yang tersedia.

Lifetime variable dimulai ketika cell spesifik terikat, dan berakhir ketika tidak terikat dari cell itu.

Terdapat 4 kategori type binding yaitu :

1.       Static

Static = cell yang terikat pada memory cell sebelum eksekusi dimulai, dan tetap terikat pada memory cell yang sama sampai program eksekusi selesai.

(+) semua addressing pada static variable bersifat langsung.

(­-) mengurangi fleksibilitas(tidak support untuk reculsive program)

2.       Stack dynamic

Stack dynamic variable = storage binding dibuat ketika deklarasi statement dijelaskan, tapi tipe nya bersifat terikat.

(+)support dynamic storage(reculsive)

(-)waktu akses yang lambat karena addressing tidak langsung

**3.**       **Explicit heap dynamic**

Explicit heap dynamic = abstrak memory cell yang di alokasi dan dealokasi dengan instruksi eksplicit run-time yang ditulis oleh programmer. Variable nya hanya bisa dialokasikan oleh pointer atau reference variable. Sebagai contoh :

**int**\*intnode; // Create a pointer

intnode = **new int;**// Create the heap-dynamic variable

. . .

**delete**intnode; // Deallocate the heap-dynamic variable

// to which intnode points

Explicit heap dynamic variable pada tipe int dibuat oleh operator **new**. Variable ini bisa di referensikan melalui pointer , intnode. Nantinya variable di deallocation dengan **delete**operator.

(+) storage manajemen yang dinamis.

(-) tidak efisien dan tidak handal.

4.     **Implicit heap dynamic**

Implicit heap dynamic terikat dengan heap storage ketika ada nilai yang ditugaskan.

Sebagai contoh:

highs = [74, 84, 86, 90, 71];

variable bernama highs sudah di deklarasi di sebelumnya, dan sekarang digunakan pada array 5 data.

(+) memiliki fleksibilitas yang memperbolehkan high generic code untuk ditulis ulang.

(-)run-time overhead mempertahankan semua atribut dinamis, yang akan mencakup jenisarray subscript dan range.

**Static scope**

Static scope adalah metode bindung nama ke nonlocal variable.

Ada 2 kategori static scope :

-          Dimana subprogram bisa di nested, yang membuat nested scope

-          Program tidak dapat di nested,  nested scope hanya dapat dibuat oleh nested class dan blocks.

**function**big() {

**function**sub1() {

**var**x = 7;

sub2();

}

**function**sub2() {

**var**y = x;

}

**var**x = 3;

sub1();

}

Pada potongan kode diatas variable x pada sub adalah mengambil x yang di deklarasikan pada prosedur big. In benar karena pencarian x dimulai pada prosedur reference yang terjadi pada sub2, big, ketika dekalarsi x ditemukan.

**Blocks**

Block adalah metode membuat static scope dalam sebuah program. Sebagai contoh :

**void**sub() {

**int**count;

. . .

**while**(. . .) {

**int**count;

count++;

. . .

}

. . .

}

Referensi count pada while adalah loop local count. Dalam kasus ini, count tersembunyi pada while loop.

**Evaluasi static scooping**

Static scoping dapat bekerja pada banyak situasi.

Masalah : - terlalu banyak akses

-    Program awal di hancurkan dan local variable biasa nya menjadi global

-    Subprogram gravitate menjadi global dari pada nested

**Dynamic scope**

Referensi pada variable terhubung melalui deklarasi dengan mencari kembali melalui rantai dari subprogram yang memaksa panggil eksekusi poin

**function** big() {

**function** sub1()

**var** x = 7;

**function** sub2() {

**var** y = x;

                      }

**var** x = 3;

                   }

Static scopping  : x pada sub2 ke x pada big

Dynamic scoping : x pada sub2 ke x paada sub1

**PERTEMUAN 4**

Data Type

Data type menjelaskan kumpulan objek data dan sebuah set operasi yang telah ditetapkan pada objek tersebut. Descriptor adalah kumpulan atribut dari sebuah variabel. Salah satu permaslahanan desain untuk semua tipe data adalah operasi apa yang didefinisikan dan bagaimana mereka ditentukan?.

Primitive Data Type

Primitive data type adalah tipe data yang berbentuk primitive, dan pasti ada dalam setiap bahasa pemrograman.

Contoh dari primitive data type :

- Integer

- Floating Point

- Complex

- Decimal

- Boolean

- Character

Character String Type

Character String Type adalah satu tipe di mana nilai-nilai terdiri dari urutan

karakter. Karakter string yang konstan digunakan untuk label output, dan input

dan output dari semua jenis data yang sering dilakukan dalam hal string.

Adapun Premasalahan pada character string type :

o Apakah tipe primitif atau hanya jenis khusus array?

o Haruskah panjang string statis atau dinamis?

Tipe – tipe operasi :

o Assignment and copying

o Comparison (=, >, etc.)

o Catenation

o Substring reference

o Pattern matching

User Defined Ordinal Type

User defined ordinal type Adalah kisaran nilai yang mungkin dapat dengan mudah terkait dengan himpunan bilangan bulat positif.

3 tipe data user defined primitive :

o Integer

o Char

o Boolean

Enumeration type

Enumeration type adalah semua nilai yang mungkin, dimana dinamai konstan, dan memberikan, atau, disebutkan dalam definisi. Fungsinya untuk memudahkan dalam tipe data grup.

Contoh :

enum days {Mon, Tue, Wed, Thu, Fri, Sat, Sun};

Subrange Type

Subrange type adalaah Sebuah subsequence berdekatan memerintahkan tipe ordinal

Implementation of User-Defined Ordinal Types

Enumeration type diimplementasikan sebagai Integer

Subrange types diimplementasikan seperti type parent dengan kode dimasukkan (oleh kompiler) untuk membatasi tugas untuk subrange variabel

Array Type

array adalah agregat homogen elemen data di mana seorang individu elemen diidentifikasi oleh posisinya dalam agregat, relatif terhadap elemen pertama.

Subscript Binding and Array Categories

- Static = rentang subscript yang statis terikat

dan alokasi penyimpanan statis (dilakukan sebelum run time).

- Fixed stack-dynamic = rentang subscript yang statis

terikat, namun alokasi dilakukan pada saat deklarasi elaborasi selama

eksekusi.

- Stack-dynamic = rentang subscript dan

alokasi penyimpanan dinamis terikat pada waktu elaborasi.

- Fixed heap-dynamic = rentang subscript dan storage binding keduanya tetap, setelah penyimpanan dialokasikan.

- Heap-dynamic = pengikatan rentang subscript dan

alokasi penyimpanan dinamis dan dapat berubah beberapa kali selama

seumur hidup array.

Heterogeneous Arrays

Heterogeneous array adalah satu di mana unsur-unsur tidak perlu dari jenis yang sama.

Rectangular array adalah array multidimensional di mana semua baris memiliki

jumlah yang sama elemen dan semua kolom memiliki jumlah yang sama dari

elemen. Array Rectangular bermodelkan meja persegi panjang.

jagged array adalah di mana panjang dari baris tidak perlu menjadi

sama. Sebagai contoh, sebuah matriks jagged dapat terdiri dari tiga baris, satu dengan 5 elemen, satu dengan 7 elemen, dan satu dengan 12 elemen.

Associative Arrays

Associative Arrays adalah koleksi tak berurut elemen data yang

diindeks oleh jumlah yang sama nilai-nilai yang disebut kunci.

Record Types

record adalah jumlah elemen data di mana unsur-unsur individu

diidentifikasi oleh nama dan diakses melalui offset dari awal

struktur. Record adalah agregat mungkin heterogen elemen data di mana unsur-unsur individu diidentifikasi dengan nama

Tuple Types

Tuple adalah adalah tipe data yang mirip dengan catatan, kecuali bahwa unsur-unsur yang tidak bernama .

Digunakan dalam Python, ML, dan F # untuk memungkinkan fungsi untuk kembali beberapa nilai

List Type

Lists pertama kali didukung dalam bahasa pemrograman fungsional LISP. Mereka selalu menjadi bagian dari bahasa-bahasa fungsional, tetapi dalam beberapa tahun terakhir mereka telah menemukan cara mereka ke dalam beberapa bahasa imperatif.

Lists dalam Skema dan LISP dipisahkan oleh tanda kurung dan

unsur tidak dipisahkan oleh tanda baca apapun. Sebagai contoh,

(A B C D)

Nested lists memiliki bentuk yang sama, sehingga kita bisa memiliki

(A (B C) D)

Union Type

Union adalah jenis variabel yang dapat menyimpan nilai jenis yang berbeda pada waktu yang berbeda selama eksekusi program. Sebagai contoh kebutuhan untuk jenis serikat, mempertimbangkan tabel konstanta untuk compiler, yang digunakan untuk menyimpan konstanta ditemukan dalam program yang dikompilasi.

Pointer and Reference Types

Sebuah variabel tipe pointer memiliki berbagai nilai-nilai yang terdiri dari alamat memori dan nilai khusus, nil. Memliki kemampuan untuk indirect addressing . memberikan cara untuk me manage dynamic memory. Pointer dapat digunakan sebagai akses lokasi di area penyimpanan dinamis(Heap)

Pointer Operations

Terdiri dari 2 operasi :

Assignment = digunakan untuk menetapkan nilai variabel pointer untuk beberapa alamat berguna

Deferencing = menghasilkan nilai yang disimpan di lokasi yang diwakili oleh nilai pointer ini

Problem Pada Pointer

Dangling pointers = Sebuah pointer menunjuk ke variabel tumpukan-dinamis yang telah deallocated

Lost heap-dynamic variable = Sebuah variabel heap-dinamis dialokasikan yang tidak lagi dapat diakses oleh program pengguna

Dangling Pointer Problem

Tombstone = extra heap cell yang merupakan pointer ke heap dynamic variable

- pointer poin variabel yang sebenarnya hanya pada tombstone

- Ketika variabel heap-dynamic de-dialokasikan, tombstone yang tersisa di set ke nil

- Mahal dalam ruang dan waktu

Locks-and-keys = nilai pointer direpresentasikan sebagai pasangan

- Heap dynamic variable direpresentasikan sebagai variabel ditambah sel untuk nilai kunci bilangan bulat

- Ketika heap dynamic variabel yang dialokasikan, nilai lock dibuat dan ditempatkan dalam lock cell dan key cell pointer

Reference Counter

maintain counter di setiap cell yang menyimpan jumlah pointer yang saat ini menunjuk pada cell

kekurangan : ruang yang dibutuhkan, waktu eksekusi yang dibutuhkan, komplikasi untuk sel terhubung sirkuler

keuntungan :intrinsik tambahan, sehingga penundaan yang signifikan dalam pelaksanaan aplikasi dihindari

Type Checking

Type checking adalah kegiatan memastikan bahwa operan dari operator adalah dari jenis yang compatible .

Compatible type adalah salah satu yang legal bagi operator, atau diperbolehkan di bawah aturan bahasa yang akan dikonversi secara implisit, dengan kode compiler yang dihasilkan, untuk tipe legal. Konversi otomatis ini disebut coercion.

Type error adalah aplikasi operator untuk operan dari tipe yang tidak layak

Strong Typing

Strong Typing adalah jika jenis kesalahan selalu terdeteksi. Ini mensyaratkan bahwa semua jenis operan dapat ditentukan, baik pada waktu kompilasi atau run time. Pentingnya String typing terletak pada kemampuannya untuk mendeteksi semua penyalahgunaan variabel yang menghasilkan jenis kesalahan. Strong typing juga memungkinkan deteksi, pada waktu berjalan, kegunaan dari jenis yang salah

nilai dalam variabel yang dapat menyimpan nilai lebih dari satu jenis.

**PERTEMUAN 5**

Expression adalah dasar dari komputasi dalam menentukan perhitungan dalam sebuah bahasa pemrograman salah satu bentuk nya adalah BNF. Ini sangat penting bagi programmer untuk memahami syntax dan semantics dari bahasa pemrograman yang digunakan.

Untuk memahami evaluation expression penting untuk mengenal urutan dari operator dan operand evaluation. Urutan dari evaluation operator diketahui dari associativity dan hukum dari bahasa pemrograman. Meski terkadang nilai dari expression bergantung padanya, urutan operasi operand pada sebuah expression biasanya tidak dijelaskan oleh designer program. Ini memberikan si pembuat untuk memilih urutan, yang membawanya pada kemungkinan dari hasil program yang berbeda dari implementasi. Jadi semua bahasa imperative memiliki konsep variable yang nilai nya dapat berubah selama eksekusi berjalan.

Inti dari bahasa pemrograman imperative adalah peran dominan dari pernyataan penugasan(assignment statement). Tujuan dari statement ini untuk menyebabkan efek samping pada perubahan nilai variable, atau state dari sebuah program.

**Arithmetic expression**

Arithmetic expression di dapat dari konvensi yang berevolusi dari matematika. Arithmetic expression terdiri dari operator(karakter khusus untuk memproses operand), operand(angka/karakter), parentheses(tanda kurung), dan function call. Operator itu sndiri bisa unary(punya single operand), binary(two operand), atau ternary(three operand). Dalam bahasa pemrograman binary operator itu adalah infix, yang berarti muncul diantara operand. Kecuali pada perl, memiliki beberapa operator yang prefix, yang artinya muncul sebelum operand.

Tujuan dari arithmetic expression adalah menentukan arithmetic computation. Implementasi dari komputasi ini harus memiliki 2 aksi yaitu fetching operand atau mengambil operand, biasanya pada memory, dan melakukan eksekusi arithmetic operation pada operand.

**Operator Evaluation Order**

Operator evaluation order berfungsi sebagai menentukan urutan dari evaluasi proses operator.

Ada 3 jenis yaitu :

1.      Precedence

Precedence menggunakan nilai dari sebuah expression bergantung pada bagian yang di evaluasi dari operator pada sebuah expression. Sebagai contoh :

            A+B\*C

Jika nilai a, b, dan c adalah 3, 4 , 5 jika kita lakukan perhitungan dari kiri ke kanan maka akan menghasilkan nilai 35. Tetapi jika melakukan perhitungan dari kanan ke  kiri maka akan menghasilkan nilai 23. Dari pada bingung menggunakan proses dari kiri ke kanan atau kanan ke kiri maka dilihat lah operand nya yang mana yang harus lebih dulu.

      Operation precedence rules adalah untuk expression evaluation menentukan urutan operator mana yang harus diproses terlebih dulu. Dalam bahasa ini exponent adalah nilai tertinggi, lalu perkalian dan pembagian pada level kekuatan yang sama, lalu diikuti oleh pertambahan dan pengurangan pada level yang sama.

      Banyak bahasa menggunakn unary version dari pertambahan dan pengurangan. Pertambahan unary(unary addition) biasa di sebut identity operator, karena biasanya tidak memiliki asosiasi operasi dan tidak memberikan efek pada sebuah operand.

2.      Associativity

Associativity adalah proses dimana jika level nya (operator) sama, maka akan dilakukan perhitungan biasa dari kiri ke kanan(kecuali \*\* dibaca dari kanan ke kiri). Misal :

A-    B+C-D

Maka akan dilakukan dari A-B lalu dilanjutkan +C dan ­­–D

Sebuah operator dapat memiliki left associativity ataupun right associativity, bermaksud jika 2 operator berdekatan, maka left operator didahulukan baru kekanan secara berurutan. Associativity biasanya diproses dari kiri ke kanan kecuali exponent operator (ketika diberikan) terkadang dari kanan ke kiri.

Jika pada kasus ruby seperti :

A \*\* B \*\* C

Akan diproses dari kanan ke kiri

3.      Parentheses

Programmer dapat mengubah urutan dan aturan asosiasi dengan menempaatkan tanda kurung pada expression. Bagian yang di tanda kurungkan lebih diutamakan di banding yang tidak menggunakan tanda kurung.  Dilihat dari penggunaan tanda kurung missal :

(A+B)\*C

Proses pertambahan akan dilakukan pertama, secara matematis ini normal. Proses ini dilakukan karena A+B menggunakan tanda kurung, sehingga lebih diprioritas kan di banding dengan perkalian. Maka proses akan dilakukan adalah A+B setelah itu hasil nya akan  \*C. proses dapat berlangsung dari kiri ke kanan atau kanan ke kiri bergantung pada prioritas dari tanda kurung.

     Programmer akan melakukan identifikasi urutan evaluasi menggunakan parentheses. Pendekatan ini lebih simple karena si pembuat atau pembaca program membutuhkan untuk mengingat precedence dan aturan associativity, kekurangan dari schema ini adalah membuat penulisan program lebih melelahkan, dan pengkompromian dari kode data.

4.      Ruby expression

Ruby adalah bahasa murni dari bahasa object oriented, yang berarti setiap nilai data, termasuk literal adalah objek. Ruby mendukung fungsi arithmetic dan logical dari sebuah operasi yang di include pada bahasa C. yang membedakan antara ruby dan C adalah pada area expression dimana semua arithmetic, relational dan assignment operator, dan juga indexing, array shift, dan bitwise logic operator.

      Hasil dari implementasi operator ini bisa melakukan system overridden pada program aplikasi ini. Jadi operator ini bisa di jelaskan kembali

5.      Conditional expression

If-then-else statement dapat digunakan untuk membuat expresi yg diproses berdasarkan kondisi. Sebagai contoh :

if (count == 0)

average = 0;

else

average = sum / count;

kondisi diatas akan menyesuaikan dengan expression yang ada. Jika count bernilai 0 maka nilai dari average akan bernilai 0, dan jika count tidak 0 maka nilai dari average adalah hasil pembagian antara sum dan count.

     Conditional expression dapat digunakan dimana saja dalam sebuah program, dimana expression yang lain dapat digunakan. Dalam bahasa C , Conditional expression juga terdapat pada perl, javascript, dan ruby.

**Operand Evaluation Order**

Operand evaluation order = desain karakteristik adalah urutan evaluasi dari operand. Variable di expression di evaluasi dengan mengambil nilai-nilai dari memori. Jika operand bersifat parenthesized maka semua operator harus dievaluasi sebelum nilai dapat digunakan sebagai operand, Jika tidak dilakukan maka operand dari operator akan memiliki side effect, maka urutan evaluasi dari operand akan tidak relevan.

1.      Side effect

Sebuah fungsi yang biasa nya disebut sebagai functional side effect, terjadi ketika perubahan fungsi dari salah satu parameter atau variable global.

Contoh :

int a = 5;

int fun1()

{

a = 17;

return 3;

}

void main()

{

a = a + fun1();

}

Pada kode diatas yang awal nya nilai a adalah 5 dengan fungsi fun1 nilai global dari a berubah menjadi 17, dan nilai a pada main program akan menjadi penjumlahan dari a dari variable global dan fungsi fun1().

            Untuk menghentikan efek samping dari parameter 2 arah ini dibutuhkan sub program.

2.      Referensi transparan dan efek samping

Referensi transparan terjadi jika ada 2 expression dalam sebuah program memiliki nilai yang sama dapat diganti antara satu dan yang lainnya dimana saja dalam sebuah program, tanpa mempengaruhi aksi dari sebuah program.

Contoh :

result1 = (fun(a) + b) / (fun(a) - c);

temp = fun(a);

result2 = (temp + b) / (temp - c);

dapat di lihat dari program di atas bahwa fungsi fun tidak akan memiliki efek samping pada program, nilai dari result1 dan result2 akan bernilai sama, karena expression yang digunakan adalah ekuivalen.

Keuntungan dari referensi transparan program adalah semantic pada program lebih mudah untuk dipelajari dan dipahami dibandingkan dengan semantic pada program yang tidak menggunakan referensi transparan. Dengan menggunakan referensi transparan akan membuat fungsi menjadi ekuivalen seperti fungsi matematika, dalam hal pemahaman.

**Overloaded Operator**

Penggunaan sebuah operator untuk lebih dari satu tujuan disebut operator overloading. Contoh berbahaya dari overloading operator adalah ampersand (&) misal :

x= &y;

dapat menyebabkan alamat dari y ditempatkan pada x. ada masalah dengan penggunaan multiple ampersand. Pertama penggunaan symbol yang sama untuk 2 operasi yang berbeda dapat menyebabkan kerugian pada proses pembacaan. Kedua keying error sederhana dapat meninggalkan operand pertama untuk bitwise dan operasi dapat bepergian tanpa diketahui oleh compiler, karena diinterpretasikan sebagai address operator.

**Type Conversion**

Tipe conversion bertipe narrowing atau widening. Conversion narrowing mengubah nilai ke tipe yang tidak dapat disimpan bahkan semua pendekatan dari tipe original. Sebagai contoh mengubah tipe data double menjadi float dalam java adalah narrowing conversion, karena range dari double lebih besar dari float. Widening conversion mengubah nilai ke tipe yang dapat dimasukkan ke pendekatan dari semua nilai tipe original. Sebagai contoh mengubah tipe data int menjadi float pada java disebut widening conversion. Widening conversion sering disebut sebagai metode aman, maksudnya adalah besarnya nilai yang dikonversi akan di pertahankan. Narrowing conversion tidak selalu aman terkadang besar nya nilai yang dikonversi dapat berubah ketika dalam proses. Tipe konversi dapat implicit atau explicit 2 sub seksi dari konversi adalah :

1.      Coercion in Expression

Salah satu bentuk decision desain tentang arithmetic expression apakah operator boleh memiliki operand dari tipe yang berbeda. Bahasa yang memperbolehkan expression seperti itu , biasa disebut mixed-mode expression, harus menentukan konvensi untuk konversi tipe operand implicit karena computer tidak memiliki binary operations yang mengambil operand dari tipe data yang berbeda. Coercion di definisikan sebagai tipe implicit konversi yang di inisialisasi oleh compiler.

Ketika 2 operand dari sebuah operator tidak memiliki tipe yang sama dan legal dalam sebuah bahasa pemrogaman, compiler harus memilih diantara nya untuk dipaksa. Memiliki kelemahan dalam deteksi error dari sebuah program.

2.      Explicit type conversion

Banyak bahasa memberikan fungsi untuk melakukan konversi explicit, narrowing dan widening. Dalam beberapa kasus pesan peringatan muncul ketika explicit narrowing conversion menghasilakn perubahan signifikan pada objek yang di konversikan.

3.      Error in expression

Error dapat muncul ketika melakukan evaluasi dari expresi. Jika bahasa membutuhkan type checking, mau static atau dynamic, maka operand type error tidak akan muncul. Error akan muncul jika operand dipaksakan pada expression yang telah di jalankan.

Error yang biasa muncul ketika hasil dari operasi tidak dapat di representasikan pada memory  yang harus di simpan. Ini biasa disebut overflow atau underflow, berdasarkan hasil nya terlalu besar atau terlalu kecil.

**Relational  and Boolean expression**

Relational expression adalah operator yang melakukan perbandingan nilai dengan 2 operand. Relational expression memiliki 2 operand dan 1 operator. Nilai dari relational expression adalah Boolean, kecuali ketika Boolean tidak dimasukkan kedalam fungsi bahasa. Relational. Relational expression biasanya overloaded untuk berbagai macam tipe. Operasi yang menentukan benar atau salah nya sebuah relational expression bergantung pada tipe operand.

Boolean expression memiliki Boolean variable, Boolean constant, relational expression dan Boolean operator. Operator biasanya dimasukkan dengan fungsi pemanggil AND, OR, dan NOT operation, dan terkadang untuk exclusive OR dan equivalence. Boolean operator biasanya hanya memanggil Boolean operand dan menghasilkan Boolean values. Dalam matematika Boolean algebra, atau biasa lebih dikenal dengan OR dan AND harus didahulukan. Yang diutamakan dari arithmetic, relational, dan Boolean operator pada bahasa C adalah :

**Short-circuited evaluation**

Short-circuited evaluation adalah expression dimana hasil dari salah satu nya ditentukan tidak melalui hasil evaluasi dari semua operand dan operator. Sebagai contoh :

(13 \* a) \* (b / 13 - 1)

Nilai dari (b / 13 - 1) bersifat independen. Jika nilai dari a adalah 0, maka tidak perlu melakukan operasi perhitungan dari (b / 13 - 1). Tapi  dalam operasi arithmetic shortcut ini terkadang tidak mudah di deteksi ketika melakukan eksekusi.

**Assignment Statement**

Assignment statement adalah bagian tengah dari susunan bahasa imperative. Dia memberikan mekanisme yang user dapat secara dinamis dapat mengubah binding dari nilai variable. Assignment ada beberapa tipe yaitu :

1.      Simple assignment

Hampir semua bahasa pemrograman menggunakan tanda equal untuk assignment operator. Semua ini harus menggunakan sesuatu yang berbeda dari tanda equal untuk equality relational operator untuk menghindari ketidakjelasan dengan assignment operator.

2.      Conditional Target

Tipe assignment yang akan menyesuaikan dengan syarat yang dibutuhkan dari kode yang ada. Contoh :

3.      Compound assignment operator

Compound assignment operator adalah metode untuk menentukan apa saja yang di butuhkan dalam sebuah assignment. Bentuk dari assignment yang dapat disingkat dengan teknik ini memiliki tujuan variable dan muncul sebagai operand pertama dari expression pada sisi kanan.

4.      Unary assignment operator

Menggabungkan increment dan decrement operasi pada assignment. Operator ++ sebagai increment dan operator – –  sebagai decrement, bisa digunakan sebagai expression atau bentuk berdiri sendiri single operator statement. Dapat muncul sebagai prefix operator maupun sebagai postfix operator.

**PERTEMUAN 6**

Control structure (Kontrol struktur) adalah statement kontrol dan statement yang eksekusi nya di atur  oleh kontrol.

Statement yang menyediakan jenis-jenis  kemampuan sering di sebut dengan kontrol statement.

Selection statement (Statement Seleksi) memberikan arti pada pemilihan antara 2 atau lebih dari jalan eksekusi.

Kategori :

-          **Two-way selector**

Sebagai contoh :

**if**control\_expression

then clause

else clause

o   Nesting selector

**if**(sum == 0)

**if**(count == 0)

result = 0;

**else**

result = 1;

statement diatas dapat di interpretasikan dengan 2 cara, berdasar pada fungsi else di samakan dengan fungsi then atau yang kedua.

-          **Multiple-way selector**

Multiple selection statement memperbolehkan seleksi satu dari banyak nomor dari statement / statement group.

Contoh :

**switch**(expression)

{

**case**constant\_expression1:statement1;

. . .

**case**constantn: statement\_n;

[**default**: statementn+1]

}

**Counter controlled loop**

Counting iterative control statement memiliki variable yang di sebut dengan loop variable, yang count value nya di simpan. Termasuk menjelaskan initial dan terminal nilai dari loop variable, dan perbedaan antara sequential loop variable value, yang biasa di sebut stepsize. Initial, terminal, dan stepsize merukapak sepsifikasi dari loop yang di sebut loop parameter.

**Logical controlled loops**

Pengulangan ekspresi terhadap Boolean ekspresi

Sebagai contoh :

**while**(control\_expression)

loop body

and

**do**

loop body

**while**(control\_expression);

**User-Located Loop Control Mechanisms**

Terkadang, fungsi ini emudah kan programmer untuk memilih lokasi dari kontrol loop dari pada menggunakan sistem top atau bottom dari sebuah loop body.

Sebagai contoh :

**while**(sum < 1000) {

getnext(value);

**if**(value < 0)

**break**;

sum += value;

}

Nilai negatif akan memberhentikan fungsi loop diatas.

**Iterasi berdasar data struktur**

Kontrol mekanisme biasa di sebut iterator yang mengembalikan element berikutnya dari sebuah urutan.

Contoh :

      Do Count = 1, 9, 2

Pada statement diatas nilai 1 sebagai nilai awal dari count, dan nilai 9 sebagai nilai akhir, dan stepsize antar nilai adalah 2. Maka fungsi internal, iterator(iterasi), harus di panggil untuk setiap iterasi untuk menghitung nilai selanjut nya dari count (dengan menambah 2 sebagai nilai akhir dari count, pada contoh tadi).

Unconditional branching statement

Unconditional branching statement mentransfer kontrol eksekusi ke lokasi spesifik dari sebuah program. Memiliki masalah pada goto logic yaitu tanpa restriksi pada penggunaan akan membuat program menjadi sulit dibaca, yang dapat menyebabkan tidak handal dan mahal perawatan.

Guarded Command

Guarded command adalah setiap baris dari seleksi statement yang mengandung Boolean ekspresi dan statement atau statement sequence. Sebagai contoh :

**if**i = 0 -> sum := sum + i

[] i > j -> sum := sum + j

[] j > i -> sum := sum + i

**PERTEMUAN 7**

Subprogram adalah pondasi block dari sebuah program yang memiliki bentuk terpentitng dalam konsep design bahasa pemrograman.

Memiliki 2 dasar dari abstraksi fasilitas:

1.      Process abstraction

2.      Data abstraction

Dasar dari sebuah subprogram adalah :

-         Memiliki sebuah subprogram pada setiap single entry point

-         Pemanggilan nya tergantung selama eksekusi dari subprogram yang di panggil

-         Control selalu mengembalikan ke pemanggil ketika memanggil menghentikan eksekusi subprogram

**Local referencing, environments**

Local variable dapat stack dynamic

Keuntungan :

-         Support recursion

-         Storage untuk local di share antara sesame subprogram

Kerugian :

-         Alokasi / de-alokasi, waktu inisialisasi

-         Indirect addressing

-         Subprogram tidak bisa history sensitive

Local variable static

Keuntungan dan kerugian dari static adalah kebalikan dari stack dynamic

**Nested subprogram**

Nested subprogram adalah subprogram yang diambil sesuai dengan keperluan, jadi tidak perlu menjalankan semua fungsi dari program untuk memanggil subprogram.

**Metode parsing parameter**

Formal parameter memiliki 3 model semantic yang berbeda yaitu :

1.      In mode

2.      Out mode

3.      In-out mode

Implementasi model parsing parameter :

1.      Pass by value

2.      Pass by result

3.      Pass by value result

4.      Pass by reference

5.      Pass by name

Implementasi parsing parameter method

Dalam banyak bahasa komunikasi parameter mengambil tempat saat melakukan run-time stack. Pass by reference adalah implementasi tersimple dan hanya untuk address yang di tempatkan pada stack.

**Designing considerations**

Ada 2 pertimbangan yang di masukkan dalam parameter passing method yaitu :

-         Efisiensi

-         One-way / two-way data transfer

2 metode diatas memiliki konflik :

-         Programming yang baik menyarankan access terbatas pada sebuah variable, yang berarti one-way setiap berhasil.

-         Pass by reference lebih efisien untuk pass struktur pada ukuran yang signifikan

**Parameter that are subprogram**

Masalahnya ada pada referensi lingkungan untuk mengeksekusi subprogram passed harus digunakan yaitu :

1.      Shallow binding

Pemanggilan statement memberlakukan melewati subprogram

Sangat natural untuk dynamic scoped

2.      Deep binding

Lingkungan dari definisi dari passed subprogram

Paling natural untuk static scoped languange

3.      Ad hoc binding

Lingkungan dari statement yang di panggil yang passed subprogram

**Overloaded subprogram**

Subprogram yang memiliki nama yang sama dengan subprogram lain pada lingkungan referensi yang sama.

**Generic subprogram**

 Mangambil parameter dari tipe yang berbeda aktivasi. Overloaded subprogram memberikan ad hoc polymorphism. Subtype polymorphism adlaah variable tipe T dapat akses semua objek tipe T atau semua tipe turunan dari T.

**Closures**

Subprogram dan reference environment dimana di definisikan.

-         Reference environment dibutuhkan jika subprogram dapat di panggil dari mana arbitrary place pada subprogram.

-         Bahasa static scope yang tidak mengijinkan nested subprogram tidak membutuhkan closures

-         Hanya dibutuhkan jika subprogram mengakses variable pada nesting scopes dan dapat dipanggil dari mana saja.

**Coroutines**

Coroutine adalah subprogram yang memiliki multiple entries dan mengontrol nya untuk mereka sendiri, di support langsung pada Lua. Symmetric control adalah pemanggil dan yang di panggil coroutines dalam ukuran yang sama. couroutine call disebut dengan resume. Coroutines memberikan quasi-concurrent execution dari program unit

**PERTEMUAN 8**

Abstraksi adalah proses representasi data dan program dalam bentuk sama dengan pengertiannya, dengan menyembunyikan rincian / detil dari implementasi. **Kelebihan dari melakukan abstraksi data adalah :**

-          Struktur data dapat di modifikasi.

-          User tidak perlu secara langsung mengakses objek data tersebut.

-          Kompilasi yang terpisah.

**Dalam C++ memiliki 3 tipe data akses yaitu :**

1.      Private (hidden entities)

Digambar kan dengan simbol (+) entities nya hanya dapat di akses oleh kelas itu saja.

2.      Public (interface entities)

Digambar kan dengan simbol (−) entities nya hanya dapat di akses oleh semua nya.

3.      Protected (inheritance)

Digambar kan dengan simbol (#) entities nya hanya dapat di akses oleh kelas itu saja dan kelas turunan nya.

**Constructor**

Constructor berfungsi sebagai inisialisasi sebuah data dari suatu instance. Dalam class harus dibuat.

**Destructor**

Destructor berfungsi sebagai penghancur sebuah data dari suatu constructor. Dalam class harus dibuat jika tidak di buat maka memori akan penuh karena data tidak pernah di hancurkan.

Java

Kelebihan :

- Multiplatform à memiliki JVM (java virtual machine).

- Bergaya C++.

- Pengumpulan sampah otomatis à tidak perlu membuat destructor.

Kekurangan :

- Tulis sekali, jalan dimana saja à swt-awt bridge tidak berjalan pada MAC OS X.

- Mudah di decompile.

- Memakan memori banyak.

Encapsulation

Enkapsulasi adalah sebuah konsep dalam penyembunyian sebuah informasi atau sebuah grup dari subprogram yang terhubung secala logis melalui sebuah unit yang dapat di compile secara terpisah.

Contoh : class

Enkapsulasi pada C

- File berisi satu atau lebih subprogram dapa melakukan kompilasi sendiri

- Interface ditempatkan pada header file

- Memiliki masalah pada hubungan antara header dan implementasi asosiasi

Enkapsulasi pada C++

- Dapat mendefinisikan header dan kode file

- Kelas di gunakan pada enkapsulasi :

a. Class sebagai interface

b. Definisi anggota di definisikan pada file terpisah

**PERTEMUAN 9**

Object oriented programming memiliki 3 konsep dasar :

1. Abstract data type (ADT)

2. Inheritance

3. Dynamic binding

Inheritance

Inheritance adalah pewarisan sifat dari superclass ke subclass, yang mewarisi attribute dan method yang masih sejenis beserta attribute dan method tambahan yang merupakan sifat khusus subclass.

Sebagai contoh class sapi adalah hewan bertulang belakang / vertebrata yang merupakan mamalia pemakan tumbuhan / herbivora.

Object oriented concepts

- ADT biasa nya di sebut class

- Isi dari class biasa nya disebut objek

- Class yang menurunkan ke subclass

- Subprogram yang melakukan definisi operasi pada objek disebut dengan method

- Pemanggilan terhadap method di sebut messages

- Kumpulan dari method pada sebuah objek disebut dengan message protocol atau message interface

- Messages memiliki 2 bagian :

o Method name

o Destination object

- Dalam kasus simple, class mewarisi semua entities dari parent nya

- Inheritance dapat di komplikasikan dengan akses control terhadap enkapsulasi entities

o Class dapat menyembunyikan entities dari subclass nya

o Class dapat menyembunyikan entities dari client nya

o Class dapat menyembunyikan entities dari client nya tetapi memperbolehkan subclass mengakses nya

- Sebuah class dapat memodifikasi method dari class turunan nya

o Method yang baru melakukan override terhadap method yang lama

o Method pada parent di overridden

- 3 cara membedakan dari class parent nya:

1. Class parent dapat mendefinisikan beberapa variable atau method nya untuk memiliki private akses, yang berarti mereka tidak akan terlihat oleh kelas turunan nya.

2. Subclass dapat menambahkan variable atau method untuk yang di wariskan dari parent.

3. Subclass dapat memodifikasi sikap dari satu atau banyak method yang diwariskan.

- Terdapat 2 jenis variable pada sebuah class :

1. Class variable – satu / class

2. Instance variable – satu / objek

- Terdapat 2 jenis method pada sebuah class :

1. Class method – menerima pesan kepada class.

2. Instance method – menerima pesan kepada objek.

- Single vs multiple inheritance

- Kekurangan dari inheritance pada penggunaan ulang :

o Membuat ketergantungan antar class yang membuat sulit di maintenance

Dynamic Binding

- Polymorphic variable dapat di definisikan pada kelas yang dapat melakukan referensi objek kelas dan objek dari turunan nya

- Ketika hierarki kelas mengandung kelas yang override method biasa disebut dengan polymorphic variable, binding ke method akan dynamic

- Mengizinkan system software lebih mudah di perpanjang ketika dikembangkan dan maintenance

- Abstrak method adalah salah satu yang tidak dimiliki pada definisi (hanya mendefinisikan protocol)

- Abstrak kelas memiliki setidak nya satu virtual method

- Abstrak kelas tidak dapat di instansikan

Exclusivity dari objek

- Semua adalah objek

o Keuntungan : elegant dan purity

o Kekurangan : lambat pada operasi objek yang simple

- Menambah objek untuk menyempurnakan typing system

o Kelebihan : cepat operasi simple objek

o Kekurangan : hasil membingungkan type system (2 tipe entities)

- Memasukkan gaya imperative typing sistem untuk primitive tetapi membuat semua nya menjadi objek

o Kelebihan : cepat operasi pada simple objek, secara relative system typing kecil

o Kekurangan : masih membingungkan memiliki 2 type system

Jika kelas turunan adalah kelas parent. Maka objek pada kelas turunan harus bersifat sama dengan objek pada kelas parent

Subclass dapat menambah variable dan method dan override method turunan dengan cara compatible

Single dan multiple inheritance

- Multiple inheritance mengizinkan kelas baru untuk di warisi dari 2 atau lebih kelas

- Kekurangan multiple inheritance :

o Bahasa dan implementasi yang kompleks

o Potensial tidak efisien – dynamic binding memiliki biaya lebih disbanding dengan multiple inheritance

- Kelebihan :

o Terkadang berharga dan berguna

Alokasi dan dealokasi objek

- Jika bersifat ADT, maka dapat di alokasikan dari semua tempat

o Alokasi dari run-time stack

o Secara explicit membuat pada heap

- Jika heap-dynamic referensi dapat melalui pointer atau referensi variable

o Memudahkan assignment – melakukan referensi ulang secara implicit

- Jika objek stack dynamic, akan memiliki masalah pada subtype – object slicing

Nested class

- Jika kelas baru dibutuhkan oleh satu kelas, maka tidak ada alasan untuk mendefinisikan agar dapat terlihat oleh kelas lain nya

OOP pada C++

- Berevolusi dari C dan SIMULA 67

- OOP yang paling sering digunakan

- Memiliki mixed typing system

- Constructor dan destructor

- Elaborate akses control kepada kelas entities

Inheritance

- Kelas tidak perlu menjadi subclass dari kelas lain

- Akses control terhadap

o Private (hanya terlihat oleh kelas sendiri dan friend kelas itu, tidak mengizinkan subclass menjadi subtype)

o Public (terlihat oleh subclass dan semua nya)

o Protected (terlihat pada kelas dan subclass, tetapi tidak pada client nya)

Dynamic binding

- Method dapat didefinisikan dengan virtual, yang berarti dapat dipanggil melalui polymorphic variable dan secara dynamic terikat pada message

**PERTEMUAN 10**

Concurrency pada sebuah mesin dapat terjadi pada 4 buah level :

- Instruksi level (eksekusi 2 atau lebih instruksi mesin serentak)

- Statement level (eksekusi 2 atau lebih statement bahasa tingkat tinggi serentak)

- Unit level (eksekusi 2 atau lebih subprogram unit serentak)

- Program level (eksekusi 2 atau lebih program serentak)

Ada 2 kategori concurrent unit control yaitu :

- Physical concurrency

Memiliki lebih dari 1 processor, beberapa program unit dari program yang sama di eksekusi serentak.

- Logical concurrency

Relaxation konsep dari concurrency yang membolehkan programmer dan aplikasi software memberikan concurrency asli.

Task terbagi menjadi 2:

* + Heavyweight = mengeksekusi tempat alamat mereka sendiri
  + Lightweight = semua berjalan dalam satu alamat, lebih efisien.

Cooperation synchronization

Dibutuhkan antara task A dan task B ketika task A harus menunggu task B untuk menyelesaikan aktivitas spesifik sebelum task A dapat melanjutkan eksekusi.

- Competition synchronization

Dibutuhkan antara 2 task ketika keduanya membutuhkan resource yang tidak dapat digunakan secara serentak.

Scheduler berfungsi memberikan synchronisasi ke mekanisme eksekusi task yang terlambat.

Task eksekusi control di atur oleh program bernama scheduler.

Macam-macam task eksekusi states :

- New (dibuat tapi belum dijalankan)

- Ready (siap untuk dijalankan tapi belum berjalan)

- Running (task yang sedang berjalan / eksekusi)

- Blocked (task yang diblok ketika sedng berjalan / interrupted oleh sesuatu)

- Dead (task yang tidak lagi aktif)

Liveness = karakteristik dari program unit yang mungkin / tidak mungkin memiliki sequential code (unit akan menyelesaikan eksekusinya)

Deadlock = semua task pada concurrent environment kehilangan liveness nya

Semaphore

Semaphore adalah data struktur mengandung counter dan queue untuk menyimpan task descriptor (data struktur yang menyimpan semua informasi relevan tentang state eksekusi dari sebuah task) / mekanisme simple yang dapat digunakan untuk memberikan syncronisasi task.

**PERTEMUAN 11**

Exception handling adalah special process yang mungkin dibutuhkan ketika sebuah exception ditemukan. Process ini dapat dilaksanakan dengan unit kode atau segment yang disebut exception handler.

Event adalah notifikasi bahwa suatu spesifik telah terjadi, seperti mouse click pada tombol grafis.

**PERTEMUAN 12**

Functional programming adalah Program serba fungsi, yang artinya setiap persoalan diselesaikan dengan menggunakan fungsi.

Functional programming sendiri mulai dikembangkan tahun 1960an, dimotivasi oleh peneliti bidang artificial intelligence, symbolic computation, theorem proving, rule-based system, dan NLP.

Bahasa fungsional pertama adalah Lisp (McCarthy, 1960).

Memodelkan masalah komputasi sebagai suatu fungsi matematika, yang mempunyai input (domain) dan hasil atau output (range).

**PERTEMUAN 13**

Program dalam logic language di ekspresikan dalam symbolic logic.

Symbolic logic terdiri dari :

* + Express proposition
  + Express relation between propositions
  + Describe how new propositions can be inferred from other propositions

Symbolic logic yang digunakan untuk logic programming dinamakan predicate calculus

Objek representasi

Objek pada proposisi biasa di representasikan dengan syarat simple : konstan atau variable.

Konstan : symbol yang merepresentasikan objek.

Variable : symbol yang merepresentasikan objek lain pada waktu lain

Compound term

Atomic proposisi : mengandung compound term

Compound term = satu elemen pada hubungan matematika, ditulis seperti fungsi matematika (mapping).

Functor = fungsi symbol yang menamai hubungan.

PROLOG adalah kependekan dari PROgramming in LOGic, yang berarti pemrograman

logika. Pemrograman Prolog menggunakan bahasa deklaratif, dimana pemrogram

memberi fakta dan aturan untuk selanjutnya diselesaikan oleh Prolog secara deduktif

sehingga menghasilkan suatu kesimpulan.

LOGIKA PREDIKAT

Logika predikat (kalkulus predikat) merupakan bagian dari komputasi logika yang juga

mencakup aljabar Boole (logika proposisional), dimana fakta dan aturan dinyatakan

melalui predikat

BAHASA DEKLARATIF

Seperti yang dijelaskan sebelumnya bahwa pokok perbedaan Prolog dari bahasa lain

adalah karena bersifat deskriptif atau deklaratif, sedang bahasa lain umumnya bersifat

prosedural atau imperatif.