**Laporan PLC**

**Programming Language Concept – LI01 – Kelompok 3**



**Disusun Oleh:**

Audia Lavena 2001537701

Clarissa Audrey 2001539386

Kevin Hadinata 2001537784

Lorenzo R. Dorrius 2001536062

**BINA NUSANTARA UNIVERSITY**

**JAKARTA**

**2016**

**KATA PENGANTAR**

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat-Nya sehingga laporan ini dapat tersusun hingga selesai. Tidak lupa kami juga mengucapkan banyak terimakasih atas bantuan dari pihak-pihak yang terlah berkontribusi dengan memberikan kontribusi baik materi maupun pikirannya, yang antara lain adalah:

* Bina Nusantara University,
* Ibu Yanfi, selaku dosen mata kuliah Programming Language Concept,
* Orang tua dan kerabat-kerabat, yang telah memberikan kontribusi baik secara fisik atau non-fisik.

Dan harapan kami semoga laporan ini dapat menambah pengetahuan dan pengalaman bagi para pembaca. Untuk kedepannya dapat memperbaiki bentuk maupun menambah isi laporan agar menjadi lebih baik lagi.

Karena keterbatasan pengetahuan maupun pengalaman kami, kami yakin masih banyak kekurangan dalam laporan ini. Oleh karena itu kami sangat mengharapakan saran dan kritik yang membangun dari pembaca demi kesempurnaan makalah ini.

Jakarta, 10 Oktober 2016

Penyusun

**DAFTAR ISI**

JUDUL I

KATA PENGANTAR II

DAFTAR ISI III

BAB I PENDAHULUAN 1

* Latar Belakang 1
* Rumusan Masalah 1
* Tujuan Penulisan 2
* Manfaat Penulisan 2

BAB II ISI 3

* Syntax & Semantics 3
* Kriteria Syntax 3
* Fungsi Utama Syntax 3
* Elemen Syntax 3
* Aturan Syntax 4
* Semantic 5
* Contoh Semantic 5
* Semantic dalam C 6
* Error Semantic dalam C 6
* Names, Binding, and Scope 7
* Names 7
* Variables 7
* Bindings 9
* Scopes 11
* Data Types 12
* Primitive Data Type 12
* Character String Type 12
* User Defined Ordinal Type 12
* Enumeration Type 13
* Subrange Type 13
* Implementation of User-Defined Ordinal Types 13
* Array Type 13
* Record Type 14
* Tuple Type 14
* List Type 15
* Union Type 15
* Pointer and Reference Types 15
* Pointer Operations 15
* Reference Counter 16
* Type Checking 16
* Strong Typing 16
* Expression and Assignment 17
* Arithmetic Expression 17
* Operator Evaluation Order 18
* Operand Evaluation Order 20
* Overloaded Operator 21
* Type Conversion 22
* Relational and Boolean Expression 23
* Short-Circuited Evaluation 23
* Assignment Statement 24
* Control Structures Statement 26
* Counter Controlled Loop 27
* Logical Controlled Loop 27
* User-Located Loop Control Mechanism 27
* Iterasi Berdasar Data Structure 27
* Unconditional Branching Statement 28
* Guarded Command 28
* Subprograms 29
* Local Referencing Environment 29
* Nested Subprograms 29
* Metode Parsing Parameter 29
* Design Consideration 30
* Parameter that are Subprogram 30
* Overloaded Subprogram 31
* Generic Subprogram 31
* Closures 31
* Coroutines 31
* Abstract Data Types 32
* Level Abstraksi 32
* Kelebihan 33
* 3 data akses dalam C++ 33
* Constructor 33
* Destructor 33
* Encapsulation 34
* Object Oriented Programming
* Inheritance 35
* Object Oriented Concepts 35
* Dynamic Binding 36
* Exclusivity dari Object 37
* Single and Multiple Inheritance 37
* Alokasi dan Dealokasi Object 37
* Nested Class 38
* OOP pada C++ 38
* Concurrency 39
* Subprograms Level Concurrency 39
* Semaphore 41
* Monitor 42
* Exception Handling and Event Handling 43
* Event Handling 43
* Functional Programming Language 44
* LISP Interpretation 44
* Variable Global 44
* Special Form: DEFINE 45
* Output Tools 45
* Tail Reclusion 45
* Logic Programming Language 46
* Symbolic Logic 46
* Object Representation 46
* Compound Term 46
* Logika Predikat 47
* Bahasa Deklaratif 48
* Dasar Pemrograman Prolog 48
* Queries 51

BAB III PENUTUP 52

* Kesimpulan 52
* Daftar Pustaka 52

**BAB I**

**PENDAHULUAN**

* Latar Belakang

Pada dasarnya untuk membuat sebuah program dibutuhkan 3 hal, algoritma yang berfungsi sebagai dasar pemikiran atau bagaimana logika untuk membuat sebuah program tersebut, software yang merupakan wadah untuk merealisasikan program ataupun konsep yang sudah kita buat, dan yang terakhir adalah bahasa pemrograman yang akan menjadi penrantara antara bahasa manusia yang kita miliki dengan bahasa komputer karena kita akan merealisasikan program-program kita di software komputer.

Pada laporan ini kelompok kami akan membahas lebih jauh mengenai bahasa-bahasa pemrograman yang ada. Seiring berkembangnya jaman, bahasa pemrogramanpun bertambah banyak, dari yang awalnya hanya ada bahasa pemrograman C, Java, dan lain-lain, hingga sekarang yang ada banyak macamnya. Hal itu dipengaruhi oleh banyaknya gadget atau hardware dijaman sekarang ini, karena setiap adanya hardware baru maka akan ada sebuah bahasa pemrograman yang baru yang akan membuat proses pembuatan sopftware di hardware baru itu menjadi lebih mudah.

Tetapi pada dasarnya setiap bahasa pemrograman itu memiliki konsep yang sama, tidak jauh berbeda dari setiap bahasa pemrograman lainnya meski setiap bahasa pemrograman memiliki keunggulan dan kelemahan disetiap konsepnya. Nah apa aja sih konsep-konsep tersebut? Kelompok kami akan mengulas setiap konsep-konsep dari bahasa pemrograman yang akan dispesifikasikan hanya pada bahasa C.

* Rumusan Masalah
* Apa saja konsep di dalam bahasa pemrograman?
* Bagaimana penerapan setiap konsep tersebut didalam bahasa pemrograman C?
* Tujuan Penulisan
* Menjelaskan secara ringkas dan jelas apa saja konsep yang terdapat didalam bahasa pemrograman.
* Menjelaskan setiap konsep-konsep bahasa pemrograman yang terdapat didalam bahasa pemrograman C.
* Manfaat Penulisan

Laporan ini diharapkan dapat memberikan manfaat yang positif bagi pembacanya dengan memberikan ulasan-ulasan yang akan mempermudah para pembaca untuk mengerti mengenai konsep-konsep yang dimiliki oleh bahasa pemrograman terutama konsep-konsep yang terdapat di dalam bahasa pemrograman C.

**BAB II**

**ISI**

Pada bab ini, kelompok kami akan membahas secara rinci apa saja konsep-konsep yang terdapat didalam bahasa-bahasa pemrograman, dan kami juga akan menekankan pembahasan itu didalam bahasa pemrograman C.

* **Syntax & Semantics**

Syntax merupakan aturan yang mendefinisikan suatu bentuk bahasa, syntax itu tidak mengerti arti atau isi dari suatu kalimat, yang memiliki tugas untuk mengartikannya adalah semantic.

* Kriteria Syntax

Syntax memiliki beberapa kriteria yang harus dipenuhi agar dapat disebut syntax:

|  |  |  |
| --- | --- | --- |
| Readability | : | Mudah dibaca |
| Writeability | : | Mudah ditulis |
| Verifiability | : | Verifikasi Program |
| Kemudahan Translasi | : | Penerjemahan yang cepat & menghasilkan kode yang efisien |
| Pengurangan Ambiguitas | : | Dua arti atau lebih, contoh Array (pemanggilan fungsi) |

* Fungsi Utama Syntax

Syntax memiliki 2 fungsi utama, yaitu:

* Menyediakan bentuk-bentuk notasi untuk komunikasi antar programmer
* Pemroses bahasa pemrograman sehingga dapat mempermudah pembuatan suatu program
* Elemen syntax

Elemen-elemen dalam syntax:

* Himpunan Karakter
* Indentifier
* Simbol untuk operator
* Komentar
* Noise Word
* Keyword & Reserved Word
* Blank
* Delimiter & Tanda Kurung
* Ekspresi
* Aturan syntax

Dalam bahasa pemrograman C, ada beberapa syntax penting yang digunakan untuk memulai sebuah pembuatan program:

* Header “<stdio.h>”, berlaku sebagai library untukt syntax bahasa C
* Main Function ”int main ()”, berlaku sebagai plateform atau wadah
* Pengembalian nilai “return 0”

Dalam bahasa C ada tanda yang sangat penting yaitu titik koma “;” dan juga kurung kurawal “{}”. Titik koma digunakan sebagai terminator dalam sebuah statement. Sedangkan kurung kurawal itu digunakan untuk menulis isi didalam fungsi.

Seperti yang sudah dibahas sebelumnya bahwa dalam elemen syntax ada yang disebut dengan identifier, dalam bahasa C indentifier ada 3:

* Variabel
* Fungsi
* Class

Yang dimana penamaan identifier ini haris diawali dengan huruf atau underscore diikuti dengan huruf lain atau angka dan tidak boleh menggunakan symbol seperti @, $, dan %.

Dalam bahasa pemrograman ada yang disebut dengan whitespace yang artinya adalah sebuah baris kosong. Ketika kita menggunakan whitespace didalam bahasa C, compiler C akan mengabaikan setiap baris yang kosong (whitespace). Jadi apasih kegunaan whitespace ini? Whitespace ini digunakan untuk kemudahan para programmer untuk membaca program.

* Semantics

Semantic adalah cabang linguistik yang meneliti arti atau makna. Makna yang diteliti oleh semantic itu adalah makna bebas konteks, yang bersifat leksikal dan juga gramatikal.

Jenis-jenis semantic ada banyak:

* Semantic Behavioris
* Semantic Dekriptif
* Semantic Generatif
* Semantic Gramatikal
* Semantic Leksikal
* Semantic Historis
* Semantic Logika
* Semantic Struktural

Semantic memiliki beberapa ciri khusus, seperti:

* Semantic adalah kajian mengenai hubungan antara tanda dengan objek yang diacu oleh tanda tersebut
* Semantic adalah kajian mengenai maksa sebuah statement
* Contoh Semantic

Multiple Meanings (Bermakna Banyak)

Salah satu baguan dari belajar bahasa adalah mengetahui banyak makna kata-kata. Berikut adalah beberapa contoh kata dengan lebih dari satu arti:

* A water pill, at first glance could be a pill with water in it, but it is understood to be a diuretic that causes a person to lose water from his body.
* Crash, can mean auto accident, a drop in the Stock Market, to attend a party without being invited, ocean waves hitting the shore or the sound of a cymbals being struck together

Words Without Meanings (Kata Tanpa Makna)

Contohnya:

* Cleans like a white tornado
* Do you have tired blood?
* Semantics dalam C

* Error Semantic dalam C
* **Names, Binding, and Scopes**
* Names

Names atau yang berarti penamaan bahasa pemrograman C, memiliki beberapa aturan, antara lain:

* Hanya holeh menggunakan huruf (besar & kecil), angka, dan/atau huruf.
* Tidak boleh menggunakan blank (spasi).
* Karakter pertama harus berupa underscore atau huruf.
* Tidak boleh menggunakan reserved words (seperti int, break, if, dan lain-lain).
* Tidak boleh men-delcare 2 kali suatu variable dengan nama yang sama dalam 1 scope.
* Tidak dapat menggunakan konstanta yang sudah ada dalam header file.

Karena banyaknya aturan-aturan dalam penamaan tersebut maka kami memiliki beberapa tips yang kiranya akan berguna pagi kalian dalam mempelajari bahasa pemrograman C:

* Jangan menggunakan nama variable yang terlalu panjang atau yang membingungkan anda.
* Jangan menggunakan underscore sebagai karakter pertama variable.
* Nama variable itu case sensitive.
* Walaupun tidak boleh menggunakan reserved words, tetapi masih bisa menggunakan kata yang sama dengan menggunakan huruf besar (karena case sensitive).
* Variables

Variable itu adalah sebuah tempat atau memory yang digunakan sebagai tempat penyimpanan data. Variable memiliki format penulisan yang khusus:

<data type> <Variable name> = <initial value>;

Contoh:

int a;

\*keterangan:

Int, merupakan <data type> untuk nilai angka

a, merupakan nama variable yang akan digunakan untuk menyimpan data

int harga= 10000;

\*keterangan:

int, merupakan <data type>

harga, merupakan nama variable

10000, merupakan value yang ingin kita simpan di dalam variable harga

Jenis-jenis data type:

|  |  |  |
| --- | --- | --- |
| Character | (char) | Untuk karakter |
| Integer | (int) | Untuk angka (bukan desimal) |
| Floating Point | (float) | Untuk angka (desimal) |
| Double Floating Point | (double) | Rangenya dua kali dari float |
| Void | (void) | Kosong |

Jenis-jenis data modifier:

* Signed
* Unsigned
* Long
* Short



* Bindings

Binding merupakan asosiasi antara attribute dengan entity, seperti antara variable dengan tipenya atau nilai, atau antara sebuah operasi dengan symbol. Binding time adalah waktu yang dibutuhkan dalam melakukan binding. Contoh sederhananya adalah:

count = count +5;

Binding memiliki beberapa tipe, antara lain:

* Static binding, yang terjadi sebelum program berjalan, dan tetap sama selama program eksekusi.
* Dynamic binding, yang terjadi ketika program sedang berjalan dan nilainya dapat berubah selama program dieksekusi.

Selain itu binding juga memiliki storage yang tidak jauh berbeda dengan variable, storage ini dibedakan menjadi 2 tipe:

* Allocation, menambil cell dari kolom memori cell yang tersedia
* Deallocation, sel memori yang telah tidak terikat dari variabel dikembalikan ke kolam memori yang tersedia.

Kategori Binding:

* Static:
* Cell yang terikat pada memory cell sebelum eksekusi dimulai, dan tetap terikat pada memory cell yang sama sampai program eksekusi selesai.
* Kelebihan: semua addressing pada static variable bersifat langsung.
* Kekurangan: mengurangi fleksibilitas(tidak support untuk reculsive program)
* Stack Dynamic:
* Storage binding dibuat ketika deklarasi statement dijelaskan, tapi tipe nya bersifat terikat.
* Kelebihan: support dynamic storage (reculsive).
* Kekurangan: waktu akses yang lambat karena addressing tidak langsung
* Explicit Hype Dyanmic:
* 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
* Kelebihan: storage manajemen yang dinamis.
* Kekurangan: tidak efisien dan tidak handal.
* Implicit Hype Dynamic:
* Implicit heap dynamic terikat dengan heap storage ketika ada nilai yang ditugaskan
* Kelebihan: memiliki fleksibilitas yang memperbolehkan high generic code untuk ditulis ulang
* Kekurangan: run-time overhead mempertahankan semua atribut dinamis, yang akan mencakup jenisarray subscript dan range.
* Scopes

Scope atau yang dikenal sebagai ruang lingkup adalah kode sumber bersarang pada awal blok titik deklarasi dan berakhir di ujung blok. Sifat kode sumber itu akan hanya berlaku pada ruang lingkup tersebut tergantung atas penempatannya. Dalam sebuah scope ada yang disebut dengan Global Variable dan Loval Variable:

* Local Variable: variable yang didirikan di dalam function atau blok suato program, variable tersebut akan bersifat lokal hanya bisa digunakan didalam ruang lingkup itu
* Global Variable: variable yang didirikan diluar function, biasanya terletak sesudah file-header sebelum function, variable tersebut akan bersifat global, bisa digunakan dimanapun dan semua function.



* **Data Types**

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 Permasalahan 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 Type

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 Type

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

* 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.

* **Expressions and Assignment**

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.

Fungsi bahasanya menggunakan variable yang sedikit berbeda, seperti parameter function. Bahasa ini juga memiliki declaration statement yang mengikat nilai pada nama. Deklarasi ini mirip dengan assignment statement, tetapi tidak memiliki efek samping.

* 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.

* 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.

* 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.

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.

5. Assignment as an expression

Dalam bahasa C, perl, dan javascript, assignment statement akan menghasilkan hasil yang sama dengan target yang di assign. Karena itu hal ini dapat digunakan sebagai expression dan sebagai operand pada expression lainnya. Sebagai contoh :

while ((ch = getchar()) != EOF) { ... }

dapat dilihat pada kode diatas ch=getchar() dibawa, hasilnya akan digunakan sebagai nilai kondisi untuk while. Jika tidak menggunakan tanda kurung maka akan dilakukan proses perbandingan dengan !=EOF lebih dulu.

Kelemahan dari memperbolehkan assignment sebagai operand pada sebuah expression adalah memberikan efek samping dari expression yang lain.

6. Multiple assignment

Beberapa bahasa pemrograman seperti perl, ruby, dan lua memberikan sebuah fungsi yaitu multiple target, multiple source assignment. Sebagai contoh :

($first, $second, $third) = (20, 40, 60);

Semantic 20 ditempatkan pada first, semantic 40 pada second, semantic 60 pada third. Jika nilai dari 2 variabel ingin ditukar. Hal ini dapat dilakukan dengan cara

($first, $second) = ($second, $first);

Ini ditukar dengan mengubah posisi tanpa menggunakan temporary variable.

7. Assignment in functional programming language

Semua identifier digunakan pada fungsi bahasa dan beberapa digunakan untuk fungsi bahasa yang lain sebagai nama dari suatu nilai.

* Mixed Mode Assigment

Biasanya assignment statement merupakan mixed mode. Fortran , C, C++ menggunakan aturan coercion untuk mixed mode assignment untuk mixed mode expression jika mirip.

* **Control Structures Statement**

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.

* Miltiple-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: statement+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 Loop

Pengulangan ekspresi terhadap Boolean ekspresi

Sebagai contoh :

while (control\_expression)

loop body

and

do

loop body

while (control\_expression);

* User-Located Loop Control Mechanism

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 Structure

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

* **Subprograms**

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 Environment

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.

* Design Consideration

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.

* **Abstract Data Types**

Abstraksi adalah proses representasi data dan program dalam bentuk sama dengan pengertiannya, dengan menyembunyikan rincian / detil dari implementsai.

Abstraksi data adalah tingkatan pengguna dalam memandang bagaimana sebenarnya data diolah dalam sebuah system database.

Abstraksi data bertujuan untuk memudahkan pengguna dalam penggunaan data.

* Level Abstraksi

Adapun 3 level abstraksi data yaitu :

1. Level fisik (physical level)

Level fisik merupakan level terendah dalam abstraksi data, yang menunjukkan bagaimana sesungguhnya suatu data disimpan.

Contoh : sebuah data di simpan pada memori.

2. Level logic / konsep

Level logic menjelaskan data yang disimpan dalam basis data serta hubungan anatar data.

Contoh : data siswa pada tbl\_siswa.

3. Level view

Level view adalah level tertinggi dari abstraksi data yang hanya menunjukkan sebagian dari basis data / informasi data yang kemunculannya di mata user diatur oleh aplikasi end-user.

Contoh : dalam sebuah sekolah memiliki bagian keuangan, bagian perpustakaan, bagian akademik, dll.



* Kelebihan

-Struktur data dapat di modifikasi.

-User tidak perlu secara langsung mengakses objek data tersebut.

-Kompilasi yang terpisah.

* 3 data akses dalam C++

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.



* 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

-Memberikan akses kepada anggota pribadi kelas.

* **Object Oriented Programming**

Object oriented programming memiliki 3 konsep dasar :

1. Abstract data type (ADT)

2. Inheritance

3. Dynamic binding

Object oriented programming language juga mendukung paradigm menggunakan class, methods, object dan message passing.

* 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

Object oriented concepts memiliki cirri-ciri:

-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 :

oMethod name

oDestination 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 Object

-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 Object

-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

* **Concurrency**

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.

* Subprograms Level Concurrency

Task atau process atau thread adalah sebuah program unit yang dapat di eksekusi bersamaan dengan program unit yang lain.

Task berbeda dari subprogram biasa, terletak pada :

-Sebuah task dapat implicit

-Ketika program unit menjalan kan eksekusi task, maka tidak perlu dilakukan penundaan

-Ketika eksekusi task selesai, control dapat tidak kembali kepada pemanggil nya

Task dapat di bagi menjadi 2 kategori yaitu :

-Heavyweight

Task eksekusi pada ruang alamat nya sendiri

-Lightweight



Semua task berjalan pada ruang address yang sama

Jika sebuah task tidak berkomunikasi terhadap eksekusi pada sebuah program di sebut disjoint.

Synchronization adalah meknisme yang mengontrol urutan dari sebuah eksekusi task.

Terdapat 2 tipe synchronization :

-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.

Contoh semaphore :

wait(aSemaphore)

if aSemaphore’s counter > 0 then

decrement aSemaphore’s counter

else

put the caller in aSemaphore’s queue

attempt to transfer control to some ready task

(if the task ready queue is empty, deadlock occurs)

end if

release(aSemaphore)

if aSemaphore’s queue is empty (no task is waiting) then

increment aSemaphore’s counter

else

put the calling task in the task-ready queue

transfer control to a task from aSemaphore’s queue

end



* Monitor

Mengenkapsulasi shared data dan operasi nya pada akses terbatas. Monitor adalah abstract data type (ADT) untuk shared data.

* **Exception Handling and Event Handling**

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 Handling

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

Contoh pada C++ :

try {

//\*\* Code that might raise an exception

}

catch(formal parameter) {

//\*\* A handler body

}

. . .

catch(formal parameter) {

//\*\* A handler body

}

Semua bahasa pemrograman sekarang memiliki fungsi exception handling.

Salah satu nya menggunakan clause try, catch, finally clause

* **Functional Programming Language**

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).

* LISP Interpretation

Notasi lambda digunakan untuk menentukan fungsi dan definisi fungsi.

Ekspresi disusun dalam notasi Cambridge-prefix.

Contoh: (+ 3 2) maka akan menghasilkan nilai 5

Operasi aritmatika:

(+) → 0

(+ 5) →5

(+ 5 4 3 2 1) →15 // maksud nya 5+4+3+2+1 = 15

(\*)→1

(\*5) →5

(\* 1 2 3 4 5) →120 // maksud nya 1\*2\*3\*4\*5 = 120

Contoh lain: (+ (\* 5 4) (− 6 2) //maka akan menghasilkan (5 \* 4) + (6 − 2) = 24

* Variable Global

Didefinisikan dengan menggunakan fungsi define.

Contoh: (define f 120)

Evaluasi ekspresi

f→120

(+ f 5) → 125

(f) →error, karena memiliki kurung tapi tidak melakukan sebuah operasi

5 →5

#f→false

#t→true

* Special Form: DEFINE

(define warna (quote (merah kuning hijau)))

(define warna ’(merah kuning hijau))

(define x f) 120

(define x ’f) x berisi simbol f

(define warna ’ merah)

(define warna merah) error, karena merah bukanlah suatu variable yang memiliki suatu isi

Evaluasi proses dari DEFINE berbeda, parameter pertama tidak akan di evaluasi. Paramete kedua akan di evaluasi, dan terikat pada parameter pertama.

* Output Tools

Biasa nya tidak dibutuhkan, karena interpreter selalu menampilkan hasil dari fungsi yang di evaluasi pada top level (tidak nested).

Explicit input dan output bukan bagian dari fungsional programming model murni, karena input operasi merubah kondisi program dan output operasi adalah side effects.

PLT Scheme memiliki dua tools utama

-MzScheme : the core compiler, interpreter, and run-time system

-DrScheme : the programming environment

DrScheme memiliki beberapa variant. Untuk menggunakan Scheme

standar:

Pilih Module (Choose Language— Module)

Definisikan #lang scheme pada definition area.

* Tail Reclusion

Operasi disebut tail recursion jika di panggil reculsive dan pada akhir dari fungsi operasi. Fungsi Tail reculsive dapat di convert secara otomatis oleh compiler untuk iterasi dengan membuat nya cepat.

Scheme language definisi membutuhkan konversi scheme language system all tail reculsive fungsi untuk menggunakan iterasi.

* **Logic Programming Language**

Programming pada bahasa logic di ekspresikan dengan bentuk logic simbol. Menggunakan logical inferencing process untuk menghasilkan produk, bersifat declarative disbanding dengan procedural.

* Symbolic Logic

Logic dapat di gunakan untuk :

-Mengekspresikan proposisi

-Mengekspresikan hubungan antara proposisi

-Menjelaskan bagaimana proposisi baru dapat mengganggu proposisi yang lain.

Bentuk symbolic logic yang digunakan untuk symbolic logic programming dipanggil predikat kalkulus.

* Object Representation

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.

Contoh :

student(jon) // jon is student

like(seth, OSX) // seth like OSX

like(nick, windows) //nick like windows

like(jim, linux) //jim like linux

PROLOG adalah kependekan dari PROgramming in LOGic, yang berarti pemrogramanlogika. Pemrograman Prolog menggunakan bahasa deklaratif, dimana pemrogrammemberi fakta dan aturan untuk selanjutnya diselesaikan oleh Prolog secara deduktifsehingga menghasilkan suatu kesimpulan. Hal ini berbeda dengan bahasa proseduralseperti Pascal, Fortran, C, atau yang sejenis, dimana pemrogram memberi perintah ataupenugasan untuk memecahkan persoalan langkah demi langkah, sehingga sering disebutsebagai programming with assignment. Disamping itu, berbeda dengan pemrogramanfungsional, pemrograman logika ini menggunakan relasi, bukan fungsi sehingga sangat sesuai untuk implementasi sistem pakar.

* 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 seperti:

lelaki(Joko) // fakta

menikah(Joko, Tuti) // fakta

∀x ∀y [menikah(x,y) ∧ lelaki(x)] → ~ lelaki(y) // aturan

∀y ∃x [orang(y) → ibu(x,y) // aturan

Kalimat pertama menunjukkan adanya fakta bahwa Joko adalah seorang lelaki, dan kalimat kedua menyatakan bahwa Joko menikah dengan Tuti. Kalimat ketiga dan keempat menunjukkan suatu aturan atau kaidah yang umum berlaku, bahwa untuk setiap pasang orang x dan y, jika x menikah dengan y dan x adalah lelaki, maka dapat dipastikan bahwa y adalah bukan seorang lelaki. Sedangkan kalimat terakhir manyatakan bahwa untuk setiap y, ada x sehingga jika y adalah orang maka y mempunyai seorang ibu x (x ibu dari y).

Simbol predikat yang digunakan dalam kalimat-kalimat tersebut adalah lelaki, menikah, orang, dan ibu yang sering disebut sebagai relasi, sedangkan Joko dan Tuti disebut sebagai simbol konstanta.

* 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. Sebagai bukti bahwa Prolog merupakan bahasa deklaratif adalah dalam menyatakan fakta dan aturan seperti berikut:

1. Jika ingin menyatakan bahwa Prawiro adalah bapak dari Joko, maka dalam Prolog dituliskan sebagai:

bapak(prawiro, joko).

2. Jika ingin menerangkan suatu kaidah bahwa A adalah kakek dari Z maka harus dibuat dahulu logika dalam bahasa Indonesia sehingga menjadi suatu aturan seperti berikut:

A adalah kakek dari Z jika A adalah bapak dari X dan X adalah bapak Z

atau

A adalah kakek dari Z jika A adalah bapak dari X dan X adalah ibu Z

Aturan tersebut ditulis dalam Prolog sebagai:

kakek(A,Z) :- bapak(A,X), bapak(X,Z).

kakek(A,Z) :- bapak(A,X), ibu(X,Z).

* Dasar Pemrograman Prolog

Pada bagian ini akan diuraikan dasar-dasar pemrograman Prolog, aturan umum penulisan program, bagaimana melakukan dialog dengan Prolog, dan beberapa pengertian dasar yang berkaitan dengan program Prolog.

* Fact

Fakta adalah suatu kenyataan atau kebenaran yang diketahui, dan menyatakan hubungan (relasi) antara dua atau lebih obyek. Fakta dapat pula menunjukkan sifat suatu obyek.

Contoh sederhana adalah:

bapak(prawiro, joko).

merah(darah).

asin(garam).

* Rules

Aturan merupakan logika yang dirumuskan dalam bentuk relasi sebab-akibat dan hubungan implikasi. Misalnya dapat dibuat aturan bahwa jika A adalah bapak dari X dan X adalah bapak atau ibu dari Z maka dapat dipastikan bahwa A adalah kakek dari Z.

Contoh untuk ini adalah:

kakek(A,Z) :- bapak(A,X), bapak(X,Z).

kakek(A,Z) :- bapak(A,X), ibu(X,Z).

* Clause

Aturan yang ditulis ini berupa klausa (clause) dan terdiri dari kepala (kakek) dan tubuh yang dipisahkan oleh tanda :- (bapak dan ibu). Klausa adalah suatu frase (ungkapan) atau susunan kata yang di dalam Prolog dapat berupa fakta atau aturan, yang selalu diakhiri dengan tanda titik (.). Suatu tubuh klausa dapa terdiri dari beberapa sub-klausa yang dihubungkan satu sama lain menggunakan tanda koma (,) yang berarti hubungan and dan tanda titik koma (;) yang menunjukkan hubungan or. Penggabungan dalam tubuh klausa yang dirangkai dengan and disebut sebagai konjungsi, sedangkan jika dirangkai dengan or disebut disjungsi. Berikut disajikan contoh penggabungan disjungsi untuk menuliskan aturan kakek sebelumnya:

orangtua(P,Q) :- bapak(P,Q); ibu(P,Q).

kakek(A,Z) :- bapak(A,X), orangtua(X,Z).

* Relation

Istilah merah, asin, kakek, bapak, ibu, dan orangtua pada contoh fakta dan aturan sebelumnya disebut sebagai relasi. Relasi adalah tabel dengan n buah kolom dan terdiri dari beberapa baris fakta maupun aturan. Misalkan relasi append adalah sekumpulan tuple (X,Y,Z) dimana Z terdiri dari elemen X diikuti dengan Y atau Z=X+Y. Anggota relasi append terdiri dari :

([], [], [])

([a], [], [a])

([a],[b],[a,b])

([a,b],[c,d],[a,b,c,d])

sedangkan ([a],[b],[]) bukan anggota relasi append.

Secara umum, suatu relasi dinyatakan dalam bentuk aturan atau fakta sebagai berikut:

P if Q1 and Q2 and ... and Qk untuk k>= 0

Sedangkan dalam notasi EBNF dapat dituliskan sebagai:

Rule::=Term:-Term {Term}

Term::=Number|Atom|Var|Atom(Term)

Term::=Term{Term}

atau dalam Prolog ditulis sebagai:

P:-Q1,Q2,...,Qk

Fakta adalah aturan untuk k=0, artinya fakta selalu berlaku tanpa harus memenuhi kondisi tertentu, atau

Fact::=Term.

* Variables

Argumen suatu predikat dapat berupa konstanta (atom), variabel, atau obyek lain. Atom disebut juga sebagai obyek nyata, sedangkan variabel disebut obyek umum. Suatu atom, variabel, atau obyek lain dalam Prolog disebut term, sehingga argumen selalu berupa term.

Dalam Prolog terdapat dua variabel, yaitu:

1. Variabel bernama, yaitu variabel yang diberi nama seperti X, Orang, dan sebagainya

2. Variabel tak bernama (placeholder), dilambangkan dengan tanda garis bawah (\_).

Setiap term yang ditulis dengan awalan huruf kapital selalu dianggap sebagai variabel bernama dalam Prolog, sedangkan awalan dengan huruf kecil dianggap sebagai suatu relasi atau konstanta. Variabel tak bernama digunakan untuk mengabaikan nilai suatu variabel, yang berarti bisa bernilai apa saja. Berikut adalah contoh penggunaan variabel bernama dan tidak bernama.

member(X,[X|\_]).

member(X,[\_|Y]):-member(X,Y).

Di bawah ini disajikan beberapa contoh program Prolog sebagai gambaran awal pemahaman pemrograman logika:

Contoh

clauses

append([],Y,Y).

append([H|X1],Y,[H|Z1]):-append(X1,Y,Z1).

* Queries

Query atau pertanyaan digunakan untuk memperoleh jawaban dari suatu problem (secara deduktif). Dalam notasi EBNF, query didefinisikan sebagai:

Query::=Term {Term}

sedangkan dalam Prolog, query dinyatakan dalam goal. Ada dua jenis goal, yaitu internal yang dituliskan langsung di dalam tubuh program, sedangkan goal eksternal dituliskan di luar program dan diberikan pada saat program dijalankan. Berikut ini beberapa contoh goal:

Contoh 1 (goal internal)

father(Bapak,Chris), write(Bapak)

grandfather(Kakek,Chris), write(kakek)

Contoh 2 (goal internal 2)

append([a,b],[c,d],Z), write(Z), [a,b,c,d]

append(X,[c,d],[a,b,c,d]),write(X)

append([a,b],Y,[a,b,c,d]),write(Y)

**BAB III**

**PENUTUP**

* **Kesimpulan**

Dari makalah yang telah kami buat ini dapat disimpulkan bahwa konsep-konsep dalam bahasa pemrograman itu ada banyak, dan saling berterkaitan antara satu sama lain. Dan setiap bahasa pemrograman memiliki ciri khasnya sendiri masing-masing yang memberikan kelebihan dan kekurangannya mereka dibanding yang lain.

Dalam bahasa C++ memang memiliki banyak kekurangan dalam hal efisiensi dikarenakan C++ bisa dibilang adalah tetua dari para bahasa pemrograman setelah C. Tetapi C++ sangatlah cocok bagi yang baru mempelajari konsep bahasa pemrograman karena ini cukup mudah, dan tidak terlalu banyak variasi dalam syntaxnya.

* **Daftar Pustaka**
* [https://binusmaya.binus.ac.id/newStudent/#/class/resources.COMP6060/007420/1610/LEC/9239](https://binusmaya.binus.ac.id/newStudent/)
* <https://tonyteaching.wordpress.com/2009/06/06/syntax-semantics/>
* <http://www.ilmubahasainggris.com/linguistic-pengertian-syntax-dan-penjelasan-serta-contohnya/>
* <http://groups.engin.umd.umich.edu/CIS/course.des/cis400/maxim/lectures/chp4.htm>
* <http://web.eecs.utk.edu/~bvz/teaching/cs365Sp12/notes/names.html>
* <https://en.wikipedia.org/wiki/Name_binding>
* <http://searchsoa.techtarget.com/definition/data-type>
* <https://www.tutorialspoint.com/cprogramming/c_data_types.htm>
* <https://www2.southeastern.edu/Academics/Faculty/kyang/2014/Fall/CMPS401/ClassNotes/CMPS401ClassNotesChap07.pdf>
* <http://www.circuitstoday.com/control-structures-in-c-and-cpp>
* <http://groups.engin.umd.umich.edu/CIS/course.des/cis400/maxim/lectures/chp8.htm>
* <http://tibasicdev.wikidot.com/subprograms>
* <http://inst.eecs.berkeley.edu/~selfpace/studyguide/9C.sg/Output/ADTs.in.C.html>
* <https://xlinux.nist.gov/dads/HTML/abstractDataType.html>
* <http://searchsoa.techtarget.com/definition/object-oriented-programming>
* <http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html>
* <https://www.edx.org/course/object-oriented-programming-iitbombayx-cs101-2x>
* <https://www.quora.com/What-is-object-oriented-programming>
* <https://www.classes.cs.uchicago.edu/archive/2013/spring/12300-1/labs/lab6/>
* <https://www.manning.com/books/c-plus-plus-concurrency-in-action>
* <http://blog.jenkster.com/2015/12/what-is-functional-programming.html>
* <http://wiki.c2.com/?FunctionalProgrammingLanguage>
* <http://searchsoftwarequality.techtarget.com/definition/functional-programming>
* <http://blog.madhukaraphatak.com/functional-programming-in-c++/>
* <https://www.quora.com/Is-C++-is-a-functional-programming-language-or-not>
* <http://groups.engin.umd.umich.edu/CIS/course.des/cis400/maxim/lectures/chp13.htm>
* <https://chii90.wordpress.com/2013/06/26/chapter-14-exception-handling-and-event-handling/>
* <https://msdn.microsoft.com/en-us/library/4t3saedz.aspx>
* <http://www.mpprogramming.com/cpp/>