**LAPORAN PROGRAMMING LANGUAGE CONCEPT**

**C**

****

**Kelompok 1**

**Alvin Saputra/2001553415**

**Martinus Idris/2001549746**

**Klemens Litano/2001547526**

**Christian Viali/2001547596**

**Vincent/2001547583**

Pertemuan 1 (Introducing)

* Tujuan belajar konsep bahasa pemograman :
  + Peningkatan kemampuan untuk mengekspresikan ide-ide
  + Peningkatan latar belakang untuk memilih bahasa yang sesuai
  + Peningkatan kemampuan untuk belajar bahasa baru
  + Pemahaman yang lebih baik signifikansi penerapan
  + Lebih baik menggunakan bahasa yang sudah dikenal
  + Keseluruhan kemajuan dalam komputer
* Kriteria penilaian bahasa :
  + Readability : kemudahan yang program dapat dibaca dan dipahami
  + Writability : kemudahan yang bahasa dapat digunakan untuk membuat program
  + Reliability : sesuai dengan spesifikasi (yaitu, melakukan dengan spesifikasinya)
  + Cost : total biaya seluruhnya.
* Kategori bahasa :
  + Imperative
  + Functional
  + Logic
  + Markup/programming hybrid

Pertemuan 2 (Describing Syntax and Semantics)

* Syntax adalah aturan menulis 'kalimat' agar mampu dimengerti dengan benar oleh bahasa pemrograman.
* Semantik (dari Bahasa Yunani: semantikos, memberikan tanda, penting, dari kata sema, tanda) adalah cabang linguistik yang mempelajari arti/makna yang terkandung pada suatu bahasa, kode, atau jenis representasi lain.
* Masalah utama dalam mendescribe syntax :
  + Kesalahan kalimat
  + Kesalahan bahasa
  + Kesalahan tanda
  + Kesalahan deskripsi tanda

Pertemuan 3 (Names, Bindings, and Scopes)

* Names dalam bahasa C : no limit, and all are significant, case sensitive
* Dalam C / C dan Pascal, konstan, variabel, dan bindings prosedur dan fungsi yang berurutan. Untuk memberikan definisi saling rekursif dari fungsi dan prosedur, C / C dan Pascal menyediakan untuk pemisahan tanda tangan dari sebuah fungsi atau prosedur dari tubuh dengan cara prototipe fungsi & deklarasi ke depan sehingga sehingga saling definisi rekursif dapat dibangun .
* C, C ++, PHP, dan Python mendukung struktur program yang terdiri dari urutan definisi fungsi dalam file
* C dan C ++ memiliki keduanya deklarasi (hanya atribut) dan definisi (atribut dan penyimpanan)

**Session 4**

**Type data**

Tipe data adalah suatu pengenal yang terdapat pada suatu program yag ahrus ada dan sangat penting untuk tanda pengenal.

Type data yang terdapat dalam bahasa c ada 5 sebagai berikut:

* Integer
* Char
* Float
* Double
* Long

**Integer**

Pada type data ini integer dapat menampung data data yang bersifat numberic tetapi hanya dapat menyimpan bilangan bulat.

**Char**

Char adalah type data yang di gunakan untukmenyimpan data yang bersifat character,type data char ini hanya dapat menampung 1 huruf,akan tetapi char dapat menampung lebih dari 1 huruf jika di beri array sebagai contoh:

Char kata[100];

**Float**

Pada type data ini float dapat menampung data data yang bersifat numberic tetapi berbeda denagn integer yang hanya dapat menyimpan bilangan bulat type data float juga dapat menyimpan bilangan desimal.

**long**

Pada type data ini long dapat menampung data data yang bersifat numberic tetapi hanya dapat menyimpan bilangan bulat.perbedaan double dengan integer adalah long dapat menyimpan data dengan skala lebih besar di bandingkan dengan integer.

**Double**

Tipe data double sama dengan float, bedanya float menampung data dalam skala kecil sedangkan double dalam sekala besar.

**ARRAYS**

Array adalah suatu skala yang disediakan si programmer baik untuk variable maupun untuk selection statement contohnya pada data types char jika tidak di beri array hanya dapat menampung 1 huruf saja,jika sudah di beri array type data tersebut dapat menampung beberapa jumlah huruf sesuai dengan array yang di sediakan.

Contoh:

Char coba[100];

Maka variable coba dapat menampung huruf sebanyak 100.

**Session 5**

**Ekspresi aritmatika**

Ekspresi aritmatika adalah suatu ekspresi atau suatu perintah atau suatu statement matematika yang berguna untuk melakukan kalkulasi atau sebagainya yang berhubungan dengan hitung menghitung angka ,dan biasanya ekspresi aritmatika ini juga sering di gunakan untuk suatu selection statement.

|  |  |
| --- | --- |
| **Simbol** | **Operator** |
| + | Penjumlahan |
| - | Pengurangan |
| \* | Perkalian |
| / | Pembagian |
| % | Modulo |

== sama dengan

< lebih kecil

<= lebih kecil sama dengan

> lebih besar

>= lebih besar sama dengan

++ penambahan suatu nilai

-- pengurangan suatu nilai

Logika yang biasanya terdapat pada suatu selection statement antara lain,<,>,<=,>= contohnya sebagai berikut

If(int g<=90){

Statement;

}

Else(){

Statement ;

}

Dan ada pula yang di gunakan untuk looping atau pengulangan contohnya:

For(int g=0;g<=29;g++)

{

Statement;

}

**Session 6**

**Selection statement**

Selection statement adalah statement yang di gunakan untuk memberikan pilihan untuk si pengguna program akan diarahkan kemana program tersebut karena dalam suatu code atau program sangat di perlukan selection ini karena selection dapat memberikan berbagai variasi pada suatu program agar lebih menarik.

selection ini dapat di bagi menjadi 2 jenis yaitu:

1. Two way selection
2. Multiple way selection

**Two way selection**

Two way selection adalah statement atau suatu perintah yang di gunakan untuk menyeleksi suatu statement dengan 2 pilihan seperti contoh if else,contoh:

If(i<=2>){

Statement;

}

Else{

Statement;

}

**multiple selection adalah statement atau suatu perintah yang di gunakan untuk menyeleksi suatu statement dengan banyak pilihan seperti else if , switch case;**

**contoh:**

* Else if

If(i<=2>){

Statement;

}

Elseif{2<i<4)

Statement;

}

Else (){

Statement;

**}**

**Switch case**

printf(" Masukan pilihan Anda : ");

scanf("%d", &N);

switch(N)

{

case 1 :

printf(" -> Anda memilih program A\n");

break;

case 2 :

printf(" -> Anda memilih program B\n");

break;

}

}

SESSION 7

SUBPROGRAMS

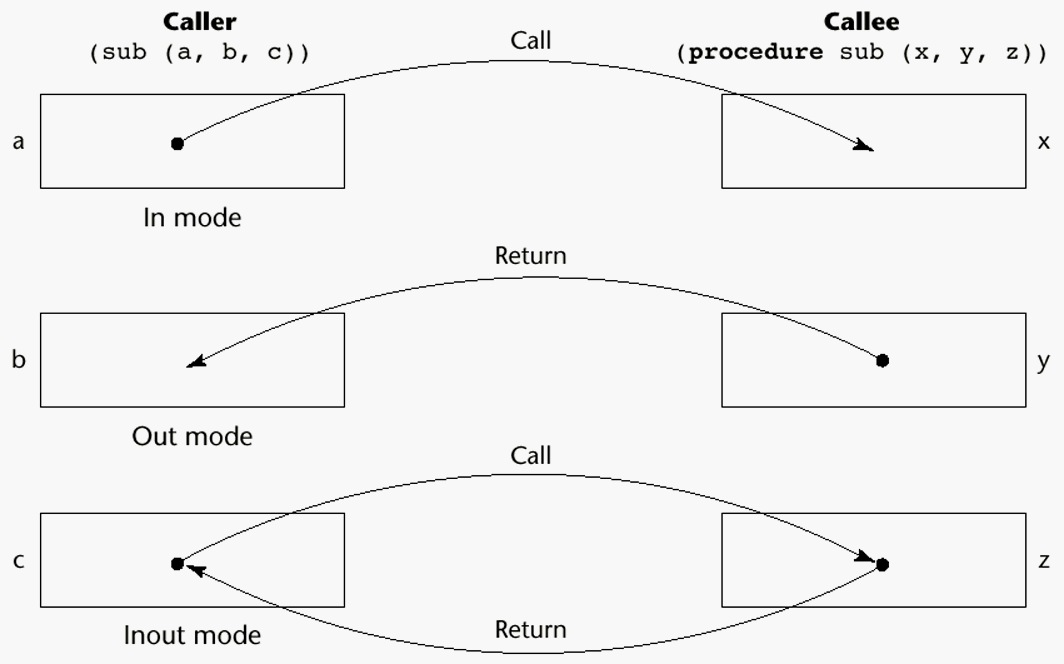
Sub Program adalah kumpulan perintah yang independent (berdiri sendiri) dan dijalankan saat dipanggil oleh main program atau subprogram lain

Ada 2 jenis Subprogram yaitu *Function* dan *Procedure.* FUNCTION adalah subprogram yang nama functionnya merupakan variable penampung hasil (*return value*)dari function yang bersangkutan. PROCEDURE adalah subprogram yang dijalankan dengan statement CALL.

Dalam function maupun procedure kita dapat menggunakan *local variable.* Local variable bisa berubah – ubah (dynamic) atau static yang berarti tidak dapat diubah. Namun ada beberapa keuntungan dan kerugian dalam menggunakan local variable. Pertama, keuntungan dalam menggunkan variabel *stack-dynamic* (dinamis) adalah dapat digunakan dalam penggunakan fungsi rekursif (fungsi yang memanggil dirinya sendiri) Co. dalam perhitungan Faktorial, keuntungan selanjutnya adalah penyimpanan data dalam menggunkan *stack-variable* dibagi dalam setiap subprogram, jadi hemat storage. Lalu kerugian dalam menggunkan *stack-dynamic variable* adalah alokasi memory banyak, jadi membutuhkan waktu untuk inisialisasi, lalu rawan terjadinya error penempatan alamat jika ternyata alamat yang akan ditempati sudah tersedia. Dan kerugian terakhir adalah subprogram tidak dapat menyimpan variable yang didalam fungsinya telah diubah – ubah. Kedua, keuntungan dan kerugian dalam menggunakan  *local variable static*(statis) adalah kebalikan dari setiap keuntungan dan kerugian dalam menggunakan *local variable stack-dynamic*.

Ada juga dalam subprograms yang disebut *Semantic Models of Parameter Passing.* Terdapat 3 model passing yaitu In mode, Out mode, dan Inout mode.

*MODELS OF PARAMETER PASSING*



SESSION 8

ABSTRACT DATA TYPE

*Abstract Data Type* (ADT) atau dalam bahasa Indonesia adalah Tipe Data Abstrak (TDA) merupakan model matematika yang merujuk pada sejumlah bentuk struktur data yang memiliki kegunaan atau perilaku yang serupa, atau suatu tipe data dari suatu bahasa pemograman yang memiliki sematik yang serupa. Merupakan hal yang sangat penting untuk mengenali bahwa operasi-operasi yang akan dimanipulasi data pada objek yang bersangkutan termuat dalam spesifikasi ADT. Sebagai contoh, ADT HIMPUNAN didefinisikan sebagai koleksi data yang diakses oleh operasi-operasi himpunan seperti penggabungan (UNION), irisan (INTERSECTION), dan selisih antar-himpunan (SET DIFFERENCE).

1. Implementasi dari ADT harus menyediakan cara tertentu untuk merepresentasikan unsur tipe data (seperti matrix) dan cara untuk mengimplementasikan operasi -operasi matrix. Secara tipikal, kita akan mendeskripsikan operasi-operasi pada ADT dengan **algoritma** (logika berfikir) tertentu. Algoritma ini biasanya berupa urutan instruksi yang menspesifikasi secara tepat bagaimana operasi-operasi akan dilakukan/dieksekusi oleh komputer.
2. Kita sekarang akan membahas lebih konkret tentang apa itu ADT. Pada dasarnya, ADT adalah tipe data tertentu yang didefinisikan oleh pemrogram untuk kemudahan pemrograman serta untuk mengakomodasi tipe-tipe data yang tidak secara spesifik diakomodasi oleh bahasa pemrograman yang digunakan. Maka, secara informal dapat dinyatakan bahwa ADT adalah :
3. Tipe data abstrak ADT pertama kali ditemukan oleh para ilmuan komputer utuk memisahkan struktur penyimpanan dari perilaku tipe data yang abstrak seperti misalnya, Tumpukan(Stack) serta antrian(Queue). Seperti kita duga, pemrogram tidak perlu tahu bagaimana Tumpukan(Stack) perubahan inplementasi ADT tidak mengubah program yang menggunakannya secara keseluruhan, dengan catatan bahwa interface ADT tersebut dengan ‘dunia luar’ tetap dipertahankan.
4. Pemakaian dan pembuatan ADT dapat dilakukan secara terpisah. yang perlu dibicarakan antara pembuat dan pengguna ADT adalah *interface* ADT yang bersangkutan.
5. ADT merupakan sarana pengembangan sistem yang bersifat *modular*, memungkinkan suatu sistem dikembangkan oleh beberapa orang anggota tim kerja dimana masing-masing anggota tim bisa melakukan bagiannya sendiri-sendiri dengan tetap mempertahankan keterpaduannya dengan anggota tim yang lain.

Kesimpulannya adalah ADT dapat dibentuk sebuah tipe data baru yang menyimpan tipe data lainnya.

Contoh ADT dalam bahasa C ++:

#include <iostream>

struct mahasiswa{ //membuat tipe data baru yang membungkus nama, NIM, umur

char nama[100];

char NIM[100];

int umur[100];

}Mahasiswa[100]; //nama variable

int main(){

Mahasiswa.nama = “Klemens”; //menambahkan data mahasiswa baru

Mahasiswa.NIM = “2001547526”;

Mahasiswa.umur = 19;

getchar();

return 0;

}

SESSION 9

OBJECT-ORIENTED PROGRAMMING (OOP)

OOP(*Object Oriented Programming*) adalah suatu metode pemograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari.

Ada 3 konsep dalam menggunkana OOP yaitu Encapsulation, Inheritance, dan Polymorphism. Konsep pertama yaitu Encapsulation yang diambil dari kata capsule(dibungkus), petanda dalam menggunakan encapsulation adalah adanya setter dan getter. Access modifier dalam menggunakan Encapsulation ada 3 yaitu public, private, protected, dan no modifier. Public dapat diakses oleh class manapun sedangkan Private hanya bias diakses oleh class itu sendiri. Lalu Protected dapat diakses class nya sendiri atau class turunannya. No modifier berarti tipe data yang tidak memiliki modifier sama sekali int, float, double, char dll.

Lalu yang kedua ada Inhertitance yang artinya turunan, dianalagikan seperti ini,. Jika seorang ibu yang memiliki kemampuan bernyanyi maka keturunannya nanti akan mendapatkan kemampuan bernyanyi pula, namun anak ini juga dapat memiliki kemampuan lain seperti bermain music. Inheritance merupakan penurunan dari class induk ke class anaknya dan class anaknya ini dapar diisi dengan statement lainnya

Dan konsep terakhir dalam OOP adalah Polymorphism, diambil dari kata Poly yaitu artinya banyak bentuk. Maka artinya dalam menyelesaikan suatu tugas dalam diselesaikan dengan banyak bentuk class

Pertemuan 10 (Concurrency)

Concurency merupakan landasan umum perancangan sistem operasi.

Proses-proses disebut concurrency jika proses-proses (lebih dari satu proses) berada pada saat yang sama.

Proses-proses yang mengalami kongkuren dapat berdiri sendiri (independen) atau dapat saling berinteraksi, sehingga membutuhkan sinkronisasi atau koordinasi proses yang baik. Untuk penanganan kongkuren, bahasa pemograman saat ini telah memiliki mekanisme kongkurensi dimana dalam penerapannya perlu dukungan sistem operasi dimana bahasa berada.

Pertemuan 11 (Event Handling dan Exception Handling)

Exception handling adalah suatu mekanisme penanganan eror yang mungkin terjadi dalam suatu program. Misalnya sebuah program yang memiliki operasi pembagian, dimana suatu ketika user menginputkan bilangan pembagi atau denumerator 0, maka program tersebut harus memiliki suatu exception handling untuk kasus tersebut. Selain contoh diatas exception handling disini juga berguna saat program kita membutuhkan akses ke CD ROM, USB, Audio&Video File, Internet dll, sehingga apabila alat alat yang kita butuhkan dalam program kita tersebut tiba tiba terputus aksesnya maka program kita tidak secara tiba tiba mati dan agar program kita bisa terus berjalan.

Event handler adalah sebuah method yang dapat mendeteksi suatu kejadian yang terjadi pada sebuah form, contoh ketika sebuah button di Click maka program akan melakukan sebuah eksekusi perintah yang ada pada event handlernya.

Pertemuan 12 – Functional Programming Language

Functional Programming Language dibagi menjadi 2, yaitu Imperative dan Functional.

Imperative memiliki eksekusi yang lebih efisien, namun memiliki semantik dan syntax yang kompleks. Dan perancangan sistem operasi lebih didesain untuk para programmer.

Sementara Functional memiliki syntaks dan semantik yang simpel namun kurang efisien eksekusi nya. Dan program nya dapat dijadikan concurrent.

Pertemuan 13 – Logical Programming Language

Logical Programming Language dapat diartikan dengan logika matematika dalam bahasa pemograman yang tiap logikanya diwakilkan dengan berbagai simbol. Dan lebih memfokuskan pada hasil dibandingkan prosedurnya.

Contoh, Preposisi yg artinya premis yang mungkin benar dan mungkin salah. Berbagai simbol untuk mewakili logika matematika yg dapat digunakan sbg logika pemograman *predicate calculus.*

Contoh daru logika matematika tsb adalah negasi, implikasi, konjungsi ekuivalen dan disjungsi