

INTRODUCTION

SISTEM KOMPUTER DAN PEMROGRAMAN

Dalam memprogramkan suatu aplikasi kita memerlukan suatu prinsip – prinsip yang perlu diketahui sehingga dalam memprogramkan suatu aplikasi menjadi lebih sederhana dan terarah. Ada beberapa langkah yang harus dilakukan dalam membuat program yaitu :

1. Menentukan tujuan yang ingin dicapai dari program yang dibuat

Dalam membuat program tentu kita perlu tahu apa yang ingin dicapai dari pembuatan program tersebut. Dengan kita mengetahui tujuan tersebut, kita dapat dengan mudah membuat alur proses dari program sehingga kesalahan – kesalahan yang ada.

2. Menentukan bahasa apa yang akan dipakai

Setelah kita menentukan tujuan dari program yang akan dibuat maka langkah selanjutnya adalah menentukan bahasa apa yang akan dipakai. Mengapa ini penting ? Hal ini penting dikarenakan masing – masing bahasa memiliki kekurangannya masing – masing. Maka dari itu pemilihan bahasa yang tepat akan mempermudah dalam membuat program. Ada beberapa bahasa yang terkenal dan mudah untuk dipakai :

- C / C++
 - Pascal
 - Basic
 - Java
 - Delphi
 - Ladder
 - PHP
 - HTML
 - ASP
 - Javascript
 - MYSQL
- } WEB BASE

Bahasa yang biasa digunakan dalam membuat program adalah C dan Java. Hal ini dikarenakan banyak yang mendukung dalam penggunaannya serta library yang banyak sehingga mudah untuk membuat program dengan bahasa tersebut. Untuk web base, bahasa yang biasa digunakan ialah HTML, PHP dan javascript. Untuk ladder biasanya digunakan dalam pemrograman PLC

3. Menentukan IDE yang akan dipakai

Setelah kita menentukan bahasa yang akan dipakai maka selanjutnya kita memilih IDE yang akan dipakai. IDE(Integrated Development Environment) adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat

lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

Sebuah IDE, atau secara bebas dapat diterjemahkan sebagai Lingkungan Pengembangan Terpadu, setidaknya memiliki fasilitas:

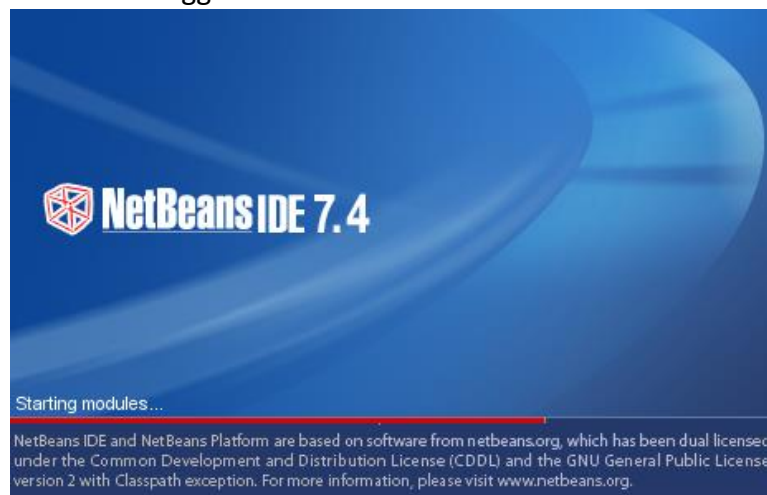
- **Editor**, yaitu fasilitas untuk menuliskan kode sumber dari perangkat lunak.
- **Compiler**, yaitu fasilitas untuk mengecek sintaks dari kode sumber kemudian mengubah dalam bentuk binari yang sesuai dengan bahasa mesin.
- **Linker**, yaitu fasilitas untuk menyatukan data binari yang beberapa kode sumber yang dihasilkan compiler sehingga data-data binari tersebut menjadi satu kesatuan dan menjadi suatu program komputer yang siap dieksekusi.
- **Debugger**, yaitu fasilitas untuk mengetes jalannya program, untuk mencari *bug*/kesalahan yang terdapat dalam program.

Sampai tahap tertentu IDE modern dapat membantu memberikan saran yang mempercepat penulisan. Pada saat penulisan kode, IDE juga dapat menunjukkan bagian-bagian yang jelas mengandung kesalahan atau keraguan.

Berikut ini adalah daftar IDE yang biasa dipakai dalam industri teknologi informasi:

Bahasa	Komersial	Free/Opensource
Basic	MS Visual Basic	
C / C++	MS Visual C++	DEV C++
Delphi	Borland Delphi	
Java	JCreator Pro	Netbeans, Eclipse
Pascal	Turbo Pascal, Lazarus	

Untuk praktikum kita menggunakan **NetBeans 7.4** untuk **Java**



Untuk NetBeans dapat didownload melalui link berikut :

<https://netbeans.org/downloads/index.html>

Pilih **JAVA SE**

NetBeans IDE 7.4 Download

7.3.1 | 7.4 | 8.0 Beta | Development | Archive

Email address (optional):

Subscribe to newsletters: ☒ Monthly ☐ Weekly

☒ NetBeans can contact me at this address

IDE Language: English Platform: Windows

Note: Greyed out technologies are not supported for this platform.

Supported technologies *

- ☒ NetBeans Platform SDK
- ☒ Java SE
- ☒ Java FX
- ☒ Java EE
- ☒ Java ME
- ☒ HTML5
- ☒ Java Card™ 3 Connected
- ☒ C/C++
- ☒ Groovy
- ☒ PHP

Bundled servers

- ☒ GlassFish Server Open Source Edition 4.0
- ☒ Apache Tomcat 7.0.41

NetBeans IDE Download Bundles

Java SE	Java EE	C/C++	HTML5 & PHP	All
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Pilih ini ←

Download

Free, 84 MB

Download

Free, 185 MB

Download

Free, 59 MB

Download

Free, 60 MB

Download

Free, 204 MB

Setelah itu klik Download dan jalankan installernya. Untuk beberapa plugin memerlukan koneksi internet saat melakukan instalasi.

Kita menggunakan NetBeans dikarenakan penggunaannya yang mudah dan gratis sehingga cocok untuk digunakan pada akademik (**Legal**).

Untuk Bahasa **C** kita menggunakan **DEV C++**.



Untuk DEV C++ dapat di download melalui link berikut :

http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe

Setelah itu download akan segera muncul dan jalankan installernya.

Kita menggunakan DEV C++ dikarenakan penggunaannya yang mudah dan gratis sehingga cocok untuk digunakan pada akademik (**Legal**).






4. Menentukan metode – metode yang akan dipakai dalam membuat program





Setelah kita mengetahui IDE yang akan dipakai maka langkah selanjutnya ialah menentukan metode yang akan dipakai dalam pemrograman. Metode – metode yang dimaksud ialah misalkan kita ingin melakukan sort terhadap suatu array maka kita perlu tahu metode yang digunakan untuk melakukan sort tersebut atau kita ingin mengkalikan matriks maka kita perlu tahu bagaimana perkalian matriks itu.

5. Membuat flowchart

Flowchart adalah gambar yang merepresentasikan deskripsi proses yang terjadi mulai dari awal proses sampai dengan goal yang dicapai.

Macam – macam bentuk pada flowchart serta fungsinya :

SIMBOL	NAMA	FUNGSI
	TERMINATOR	Permulaan/akhir program
	GARIS ALIR (FLOW LINE)	Arah aliran program
	PREPARATION	Proses inialisasi/ pemberian harga awal
	PROSES	Proses perhitungan/ proses pengolahan data
	INPUT/OUTPUT DATA	Proses input/output data, parameter, informasi

	PREDEFINED PROCESS (SUB PROGRAM)	Permulaan sub program/ proses menjalankan sub program
	DECISION	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	ON PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	OFF PAGE CONNECTOR	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Dalam membuat flowchart ada hal yang perlu diperhatikan :

1. Flowchart digambarkan dari halaman **atas** ke **bawah** dan dari **kiri** ke **kanan**.
2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja, misalkan **Melakukan penggandaan diri**.
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.
6. Lingkup dan range dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
7. Gunakan simbol-simbol flowchart yang standar.

Contoh :

Misalkan kita ingin mengetahui apakah suatu bilangan genap atau ganjil.

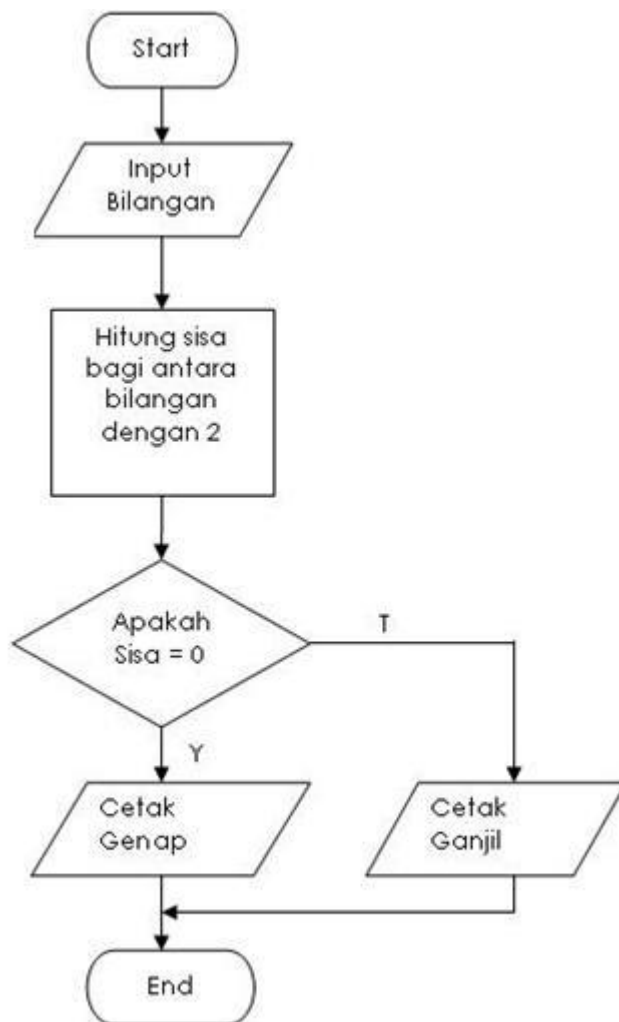
Solusi :

Untuk mengetahui apakah suatu bilangan disebut genap atau ganjil maka kita harus mengetahui **sifat dari genap ataupun ganjil** (Metode penentuan genap atau ganjil).

Untuk sifat genap ialah $2n$ maka **bilangan genap** adalah **bilangan yang habis dibagi 2**.

Maka dengan kita mengetahui sisa bagi dari suatu bilangan maka kita bisa menentukan apakah bilangan tersebut genap atau ganjil. Jika **sisa bagi = 0** maka **genap selain itu ganjil**.

Flowchart :



Untuk membuat flowchart dapat menggunakan beberapa aplikasi seperti : **Microsoft Visio**

6. Variabel, Operator dan Literal

Variabel adalah **lokasi pada memori** dimana **suatu nilai** dapat **disimpan**. Variable memiliki nama, tipe dan nilai. **Sebelum digunakan dideklarasikan terlebih dahulu** (biasa diawal program).

Ada 7 macam variabel, yaitu :

1) Variabel kelas

Variabel yang dideklarasikan didalam kelas dan diberi modifier statik.

2) Variabel Instans

Variabel yang dideklarasikan didalam kelas tetapi tidak memiliki modifier statik

3) Komponen Array

Array adalah deretan variabel yang bertipe sama dan dianggap sebagai suatu kelompok. Meskipun merupakan satu kelompok namun masing – masing anggotanya memiliki indeks yang menjelaskan alamat dari array. Masing – masing dari alamat tersebut bisa diisi nilainya dan dapat diakses secara individu.

4) Parameter Method

Variabel ini berfungsi sebagai argumen input. Variable ini memiliki life time hanya selama interpreter memanggil dan mengeksekusi method tersebut. Setelah proses eksekusi selesai, variabel ini akan dihapus.

5) Parameter Kontruktor

Adalah nilai argumen yang dilewatkan pada konstruktor

6) Paramater Exception Handler

Adalah variable yang muncul setiap kali exception terjadi dan di tangkap oleh **catch** dan **try-catch**

7) Variabel lokal dalam blok

Adalah variabel yang dideklarasikan secara lokal untuk suatu blok (antara {...})

Setiap variabel memiliki tipe data. Tipe data dibagi menjadi 2 kategori :

1) Tipe data sederhana

Merupakan tipe data dasar yang tidak diturunkan dari tipe lain. Tipe ini merupakan tipe primitif. Ada 8 tipe primitif :

Keyword	Deskripsi	Size / Format
Bilangan bulat		
Byte	Byte-length integer	8 bit two's complement
Short	Short integer	16 bit two's complement
Int	Integer	32 bit two's complement
Long	Long integer	64 bit two's complement
Bilangan real		
Float	Single-precision floating point	32 bit IEEE 754
Double	Double-precision floating point	64 bit IEEE 754
Non Numerik		
Char	Karakter Tunggal	16 bit Unicode character
Boolean	Nilai Boolean	True or False

2) Tipe data komposit

Tipe data ini disusun dari tipe data sederhana atau tipe komposit lain yang telah ada. Tipe ini antara lain String, array, class dan interface. Untuk **string pada java** merupakan **satu kelas tersendiri yang mengatur semua fungsi** sedangkan pada bahasa pemrograman lain **string masuk ke dalam tipe data primitif**.

Konversi dan Casting Tipe data

Konversi maksudnya adalah mengubah tipe data dari suatu ekspresi menjadi tipe data yang lain. Hal ini penting untuk diketahui agar nantinya kita dapat dengan mudah memanipulasi program kita agar didapatkan hasil yang lebih baik. Ada dua tipe konversi yaitu :

1) Konversi Otomatis

Yang dimaksudkan konversi otomatis ialah mengubah tipe data dari suatu data ke tipe data lain dengan kategori yang sama. Ada tiga kategori yaitu bilangan

bulat, bilangan real dan non numerik. Namun perlu diperhatikan size dari masing – masing tipe data. Dimana masing – masing tipe data tersebut mempunyai ukuran maksimum masing – masing. Contoh konversi dari tipe data **byte** ke tipe data **integer**. Namun jika dari integer ke byte akan mengalami masalah ketika angka integer sudah melebihi kapasitas tipe data byte.

2) Casting / Konversi Paksa

Yang dimaksud casting adalah mengubah dari suatu tipe data dari suatu data ke tipe data lain dengan kategori yang berbeda. Misalkan dari tipe data **integer** ke tipe data **double**

Tipe Asal	Tipe Tujuan
Byte	Short, char, int, long, float, double
Short	Int, long, float, double
Char	Int, long, float, double
Int	Long, float, double
Long	Float, double
Float	Double

Operator

Operator adalah karakter khusus yang memerintahkan compiler untuk melakukan operasi terhadap sejumlah operand. Operand berupa variabel atau besaran literal.

Berdasarkan jumlah operand yang dipasangkan terhadapnya, operator terbagi menjadi beberapa jenis, yakni :

1) Operator unary

Operator ini hanya membutuhkan 1 operand saja

Operand ini terbagi atas :

- Operator prefiks : Operator yang ditempatkan sebelum operand
- Operator postfiks : Operator yang ditempatkan setelah operand

2) Operator binary infiks

Operator yang membutuhkan 2 operand, operator berada diantara 2 operand

3) Operator ternary

Membutuhkan 3 operand

4 macam operator dasar yaitu :

1) Operator aritmatika

Operator	Pemakaian	Deskripsi
+	Op1 + Op2	Menjumlahkan Op1 dan Op2
-	Op1 - Op2	Mengurangi Op1 dengan Op2
*	Op1 * Op2	Mengalikan Op1 dengan Op2
/	Op1 / Op2	Membagi Op1 dengan Op2
%	Op1 % Op2	Menghitung sisa bagi dari Op1/Op2

* Op = Operand / variabel hitung

Terdapat beberapa bentuk sederhana dari operator aritmatika ini jika variabel hasil merupakan operand :

$a = a + 4 \rightarrow a += 4$

$a = a * 4 \rightarrow a *= 4$

$a = a \% 4 \rightarrow a \% = 4$

$a = a + 1 \rightarrow a += 1$

$a = a - 1 \rightarrow a -= 1$

Operand yang dapat digunakan adalah variabel dengan tipe data **byte, short, int, long, float dan double**

Operasi aritmatika + dan – memiliki fungsi lain yaitu :

Operator	Pemakaian	Deskripsi
+	+Op	Mempromosikan Op menjadi tipe int jika dia bertipe byte, short atau char
-	-Op	Menegatifkan Op

Operasi aritmatika yang lain :

Operator	Pemakaian	Deskripsi
++	Op++ atau ++Op	Menambahkan Op dengan 1
--	Op-- atau --Op	Mengurangi Op dengan 1

*** Op = Operand / variabel hitung**

Perbedaan antara ++/-- didepan operand dan ++/-- dibelakan operand adalah jika operator ini berada didepan maka **Op sebelum** dimasukan terlebih dahulu ke **Op sekarang** setelah itu dijumlah/dikurangi dengan 1. Sedangkan jika operator ini berada dibelakang maka nilai **Op sebelum** akan dijumlah/dikurangi terlebih dahulu setelah itu nilai ini dimasukan ke **Op sekarang**.

2) Operator relasi

Operator ini membandingkan dua nilai operand yang dipasangkan kepadanya. Nilai yang dihasilkan oleh operator ini bertipe data boolean (True atau False)

Operator	Pemakaian	Menghasilkan TRUE jika
>	Op1 > Op2	Op1 lebih besar dari Op2
>=	Op1 >= Op2	Op1 lebih besar dari atau sama dengan Op2
<	Op1 < Op2	Op1 lebih kecil dari Op2
<=	Op1 <= Op2	Op1 lebih kecil dari atau sama dengan Op2
==	Op1 == Op2	Op1 dan Op2 bernilai sama
!=	Op1 != Op2	Op1 dan Op2 tidak bernilai sama

*** Op = Operand / variabel hitung**

3) Operator logika boolean

Operator yang digunakan untuk operand yang bertipe boolean. Hasil dari Operator ini adalah boolean.

Operator	Pemakaian	Deskripsi
&	Op1 & Op2	Evaluation AND
	Op1 Op2	Evaluation OR
^	Op1 ^ Op2	Evaluation XOR
&&	Op1 && Op2	Logical AND
	Op1 Op2	Logical OR
!	! Op	Negation
==	Op1 == Op2	Sama dengan
!=	Op1 != Op2	Tidak sama dengan

* **Op = Operand / variabel hitung**

Perbedaan antara operator evaluation dengan logical adalah evaluation melakukan evaluasi kedua sisi untuk menentukan hasil sedangkan operator logical mengabaikan evaluasi sisi sebelah kanan jika hasil telah didapat dari sisi sebelah kiri.

4) Operator bitwise

1. NOT ~
2. AND &
3. OR |
4. XOR ^
5. Shift Kiri <<
6. Shift Kanan >>

Literal

Literal adalah istilah dalam bahasa pemrograman yang berarti apa yang anda ketik itulah yang anda dapat. Jika anda mengetik angka 4 maka yang dikeluarkan oleh program adalah angka 4. Literal dibagi menjadi 4 yaitu :

1. Literal Numerik

Literal numerik adalah literal yang berupa bilangan. Ada 2 yaitu bertipe bilangan bulat dan bertipe bilangan real.

2. Literal Karakter

Literal karakter adalah literal yang berupa satu karakter. Karakter disini adalah 16 bit Unicode. Literal karakter direpresentasikan di antara tanda petik tunggal (**Contoh : 'a','x'**). Beberapa karakter tidak dapat ditampilkan langsung namun diwakilkan oleh gabungan karakter lain yang disebut **special character**.

Berikut tabel special character :

Special Karakter	Deskripsi
\n	Membuat baris baru (ENTER)
\t	Tabulasi (TAB)
\b	Backspace
\r	Carriage Return

\f	Form feed
\\	Backslash
\'	Petik tunggal
\"	Petik ganda
\ddd	Oktal
\xdd	Hexa decimal
\udddd	Unicode karakter

3. Literal String

Literal string adalah literal yang berupa gabungan karakter. Literal string direpresentasikan diantara tanda petik ganda (**Contoh : “Saya Budi”**).

4. Literal Boolean

Literal boolean adalah literal untuk ungkapan logika. Terdiri dari dua nilai yaitu **TRUE** dan **FALSE**.

7. Membuat program sesuai dengan flowchart yang ada

Setelah kita membuat flowchart maka yang harus dilakukan adalah memulai programming. Untuk itu kita perlu mempelajari cara penggunaan IDE dan bahasa – bahasa pemrograman pada tahap selanjutnya.

8. Memperbaiki kesalahan atau error

Masing – masing IDE memiliki caranya sendiri untuk memberi tahu kepada user letak kesalahan penulisan. Kesalahan yang biasa terjadi adalah :

1) Syntax Error

Kesalahan yang paling sering ditemukan pada saat membuat program adalah kesalahan sintaks atau Syntax Error, dimana perintah atau statemen yang diketikkan menyalahi aturan pengkodean yang dimiliki oleh bahasa pemrograman yang Anda gunakan.

Sebuah bahasa pemrograman memiliki aturan pengkodean tersendiri yang harus dipatuhi, sebagai contoh pada bahasa pemrograman Pascal/Delphi, setiap statemen diwajibkan diakhiri dengan tanda titik koma (;). Jika Anda tidak menuliskannya, program akan menampilkan pesan Syntax Error pada saat dijalankan.

Setiap bahasa pemrograman memiliki keyword, yaitu perintah-perintah baku yang digunakan. Sebagai contoh, keyword yang umum adalah kondisi if, perulangan for atau while, penulisan fungsi dan lambang aritmatika seperti modulus, pangkat, dan lain-lain. Kesalahan penulisan keyword juga merupakan Syntax Error.

Kesalahan penulisan parameter pada sebuah function/procedure juga termasuk dalam kategori Syntax Error, misalnya jika function yang Anda gunakan memerlukan parameter sementara Anda lupa menuliskan parameter tersebut.

Syntax Error merupakan jenis kesalahan yang paling sering ditemui, tetapi juga pada umumnya paling mudah untuk ditanggulangi. Syntax Error cukup mudah diketahui dan diperbaiki jika bahasa pemrograman yang Anda gunakan menunjukkan baris kesalahan dengan tepat, dan menampilkan pesan kesalahan yang benar.

Pada beberapa bahasa pemrograman, disediakan fasilitas Auto Syntax Check, dimana muncul sebuah pesan peringatan ketika Anda mengetikkan sintaks yang salah.

2) **Run-time Error**

Jenis kesalahan Run-time Error terjadi ketika kode program melakukan sesuatu yang tidak dimungkinkan. Contohnya pada saat sebuah aplikasi mencoba mengakses file yang tidak ada, atau terjadi kesalahan alokasi memory.

Terkadang Run-time Error terjadi karena berbagai aspek dan tidak selalu karena kesalahan pemrograman, sebagai contoh jika Anda sengaja menghapus beberapa file penting yang digunakan oleh suatu aplikasi, maka terdapat kemungkinan akan terjadi Run-time Error pada saat aplikasi tersebut dijalankan.

Walaupun demikian, pencegahan semaksimal mungkin dengan memberikan validasi dan pesan yang user-friendly saat terjadi kesalahan pada aplikasi, akan sangat membantu untuk mengetahui mengapa aplikasi tidak berjalan dengan semestinya.

3) **Logical Error**

Logical Error merupakan jenis kesalahan yang cukup sulit untuk ditemukan penyebabnya. Karena aplikasi yang mengandung Logical Error berjalan tanpa pesan kesalahan, tetapi mengeluarkan hasil yang tidak diharapkan, misalnya jika aplikasi Anda menghasilkan perhitungan yang salah.

Logical Error baru dapat diketahui setelah Anda melakukan testing dan meninjau hasilnya. Logical Error dapat diperbaiki dengan memeriksa alur program dan nilai variabel yang dihasilkan.

Bagaimana cara mencegah dan memperbaiki error

Error adalah sesuatu yang tidak diinginkan. Apa yang harus dilakukan supaya terhindar dari error ? berikut tips dalam mengatasi dan mencegah error yang ada :

1) **Selalu Deklarasikan Variabel**

Syntax Error, bahkan Logical Error, mungkin terjadi jika terdapat penulisan variabel yang salah. Sebaiknya Anda mendeklarasikan variabel yang Anda

gunakan walaupun bisa jadi bahasa pemrograman yang Anda gunakan mengijinkan untuk tidak melakukan deklarasi variabel.

Visual Basic merupakan salah satu bahasa pemrograman yang mengijinkan penggunaan variabel tanpa deklarasi, walaupun demikian disarankan Anda menggunakan deklarasi variabel. Hal tersebut akan memperkecil kesalahan penulisan variabel.

Masih dengan contoh Visual Basic, Anda dapat menambahkan perintah Option Explicit pada program untuk mencegah kesalahan tulis pada variabel, jika terdapat variabel yang belum dideklarasikan, maka Visual Basic akan menampilkan pesan kesalahan.

Anda sebaiknya memiliki suatu skema standard untuk pemberian nama variabel dan konsisten dengan penggunaannya. Contohnya berikan nama variabel diawali dengan huruf s jika bertipe data string, misalnya sResult, sTemp, dan lain-lain.

Pada Visual Basic maupun beberapa bahasa pemrograman lain yang berorientasi object, kita dapat mendeklarasikan variabel dengan tipe data object. Terdapat berbagai jenis macam object yang dikenal, dan sebaiknya Anda menuliskannya dengan lengkap object yang dimaksud. Misalnya object ListBox, Label, dan lain-lain.

2) **Gunakan Variabel Lokal**

Sangat disarankan agar Anda selalu menggunakan variabel lokal. Salah satu manfaatnya adalah jika terjadi kesalahan program (terutama Logical Error), maka penyebab kesalahan dan solusinya akan lebih mudah ditemukan. Hal ini dikarenakan variabel lokal memiliki ruang lingkup penggunaan yang lebih kecil dibandingkan variabel global, yang dapat diakses oleh procedure yang mana saja.

Penggunaan variabel global, sering menimbulkan kerancuan dan memperbesar kemungkinan terjadinya kesalahan tanpa disadari.

3) **Kenali Jenis error**

Error yang timbul pada sebuah aplikasi memiliki karakteristik. Karena itu selalu baca dan perhatikan baik-baik pesan kesalahan yang timbul. Beberapa jenis bug berdasarkan karakteristiknya adalah sebagai berikut:

[1] **Divide By Zero**

Jika pada sebuah pembagian, pembagi bernilai 0, maka program akan terhenti dan mengalami error.

[2] **Infinite Loop**

Pengertian loop adalah perulangan, yang sering digunakan dalam

teknik pemrograman. Penggunaan loop yang salah dapat mengakibatkan program menjalankan sebuah procedure tanpa akhir.

[3] **Arithmetic overflow or Underflow**

Overflow terjadi saat sebuah perhitungan menghasilkan nilai yang lebih besar daripada nilai yang dapat ditampung oleh media/variabel penyimpanan. Sementara underflow merupakan kebalikannya. Pada perhitungan aritmatik, hal ini sering ditemukan dan menjadi masalah.

[4] **Exceeding Array Bounds**

Array merupakan variabel berdimensi yang memiliki indeks. Saat program mengakses indeks diluar array yang ditentukan, maka akan mengakibatkan error.

[5] **Access Violation**

Hal yang terjadi saat sebuah proses mencoba melewati batas yang diijinkan oleh sistem. Misalnya menulis sebuah nilai pada alamat memory, segmen, atau media yang diproteksi.

[6] **Memory Leak**

Penggunaan memory yang tidak diinginkan, dapat terjadi karena program gagal melepaskan memory yang sudah tidak digunakan.

[7] **Stack Overflow or Underflow**

Stack merupakan struktur data dengan prinsip LIFO (Last In First Out), pada program Anda dapat mengimplementasikan logika stack untuk suatu tujuan. Tetapi jika stack melebihi atau dibawah nilai yang diijinkan oleh program, maka akan timbul kesalahan Stack Overflow/Underflow.

[8] **Buffer Overflow**

Buffer merupakan tempat penyimpanan sementara dalam teknik pemrograman. Buffer Overflow terjadi jika Anda menyimpan terlalu banyak data yang tidak dapat ditampung oleh buffer yang disediakan.

[9] **Deadlock**

Merupakan suatu kondisi dimana dua atau lebih proses saling menunggu satu sama lain untuk menyelesaikan prosesnya, dan tidak satupun dari proses tersebut yang selesai. Problem deadlock sering ditemukan pada multiprocessing.

[10] **Off By One Error**

Merupakan istilah untuk menggambarkan perulangan yang terlalu banyak atau terlalu sedikit. Misalnya perulangan yang dikehendaki adalah lima kali, tetapi kenyataan yang terjadi aplikasi mengulang proses tersebut sebanyak empat kali atau enam kali. Kesalahan ini pada umumnya terjadi karena kesalahan logika penulisan kode pada proses perulangan.

4) **Berikan Komentar**

Jangan kuatir kode program Anda dipenuhi oleh komentar. Karena akan lebih

mudah bagi Anda untuk mempelajari lagi kode-kode program yang pernah Anda buat dengan membaca komentar.

Dengan mengerti kode program dengan baik, maka akan menjadi lebih mudah jika pada suatu saat terdapat Logical Error yang membutuhkan analisa ulang kode program.

5) Penutup dan pembuka

Tanda (harus ditutup dengan) .

Tanda { harus ditutup dengan } .

Tanda [harus ditutup dengan] .

Tanda < harus ditutup dengan > .

6) Validasi

Tidak semua orang mematuhi aturan yang Anda terapkan pada aplikasi, karena itu Anda harus melakukan validasi untuk data yang dimasukkan oleh pengguna. Misalnya pada suatu form pendaftaran, Anda sebaiknya melakukan validasi untuk input yang tidak boleh kosong (mandatory/required fields), melakukan pembatasan karakter, dan validasi huruf/angka yang diperlukan.

9. Memaksimalkan penggunaan variabel dan syntax

Melakukan optimasi terhadap penggunaan variabel dan syntax dapat mempercepat proses yang ada serta menghemat memori komputer dalam memproses data.

PERTEMUAN 1

BAHASA JAVA DAN C

BAHASA JAVA

Java adalah bahasa pemrograman tingkat tinggi yang berorientasi objek, diedarkan oleh Sun Microsystems¹ pada awal tahun 1996. Sejarah awal Java berawal di tahun 1991 ketika satu group insinyur-insinyur Sun, yang dipimpin oleh Patrick Naughton dan James Gosling, ingin mendesain sebuah bahasa pemrograman komputer yang berukuran kecil yang dapat digunakan untuk peralatan elektronika konsumen seperti switchboxes TV kabel.

Dikarenakan peralatan-peralatan ini menggunakan konsumsi daya dan memory yang rendah, maka bahasa pemrograman tersebut harus berukuran sangat kecil. Juga karena setiap vendor menggunakan CPUs (Central Processing Unit) yang berbeda, maka bahasa tersebut harus bersifat multiplatform, tidak terikat hanya pada satu arsitektur (Architecture Neutral). Proyek ini diberi nama Green Project.

Karena harus bersifat Architecture Neutral, maka Green Project menggunakan Virtual Machine (atau dikenal dengan Java Virtual Machine) yang berasal dari model implementasi bahasa Pascal di awal-awal perkembangan PC. Dikarenakan insinyur-insinyur Sun berlatar belakang Unix², jadi mereka mendasari bahasa pemrograman mereka dengan C++ dari pada Pascal. Secara khusus mereka membuat bahasa mereka berorientasi obyek (object oriented), bukan berorientasi prosedur (procedural oriented) seperti model bahasa Pascal. Bahasa pemrograman tersebut dinamakan Oak, kemudian diubah menjadi Java.

Karena pada awalnya ditujukan untuk pemrograman device kecil, Java memiliki karakteristik berukuran kecil, efisien, dan portable untuk berbagai hardware. Perkembangannya sempat terhenti karena tidak ada yang tertarik dan tidak memiliki pasar seperti yang diramalkan. Ketika teknologi internet berkembang, Java diarahkan untuk menjadi bahasa pemrograman internet karena fitur-fitur Java seperti Architecture Neutral, real time, reliable dan secure sangat sesuai untuk pengembangan internet.

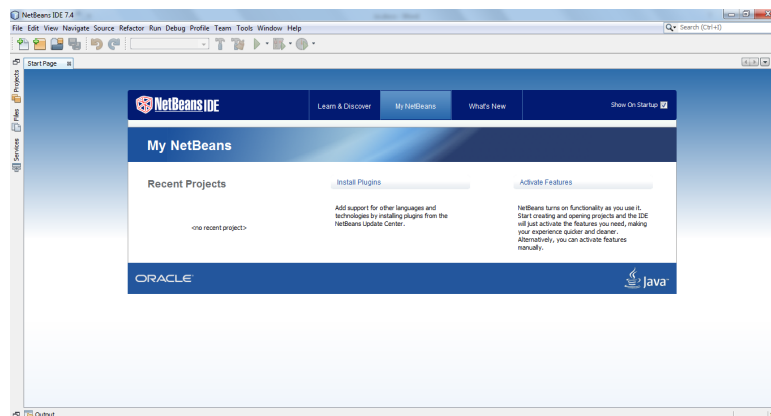
NETBEANS

Sebelum kita mempelajari bagaimana bahasa java itu, kita perlu mengenal IDE yang akan kita gunakan yaitu Netbeans 7.4. NetBeans dimulai pada tahun 1996 sebagai Xelfi (word bermain di Delphi), Java IDE proyek mahasiswa di bawah bimbingan Fakultas Matematika dan Fisika di Universitas Charles di Praha. Pada tahun 1997 Roman Stanek membentuk perusahaan sekitar proyek tersebut dan menghasilkan versi komersial NetBeans IDE hingga kemudian dibeli oleh Sun Microsystems pada tahun 1999. Sun open-source IDE NetBeans

pada bulan Juni tahun berikutnya. Sejak itu, komunitas NetBeans terus berkembang. Pada tahun 2010, Sun (dan dengan demikian NetBeans) diakui oleh Oracle.

NetBeans IDE adalah sebuah lingkungan pengembangan open source yang terintegrasi. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (Java SE (termasuk JavaFX), Java ME, web, EJB, dan aplikasi mobile) di luar kotak. Di antara fitur-fitur lainnya adalah Ant berbasis proyek sistem, dukungan Maven, refactorings, kontrol versi (CVS mendukung, Subversion, Mercurial dan ClearCase).

Modularitas: Semua fungsi IDE disediakan oleh modul. Setiap modul menyediakan fungsi yang didefinisikan dengan baik, seperti dukungan untuk bahasa Java, editing, atau dukungan untuk sistem versi CVS, dan SVN. NetBeans memuat semua modul yang dibutuhkan untuk pengembangan Java dalam sekali download, memungkinkan pengguna untuk mulai bekerja segera. Modul juga memungkinkan NetBeans untuk diperpanjang. Fitur-fitur baru, seperti dukungan untuk bahasa pemrograman lain, dapat ditambahkan dengan menginstal modul tambahan. Misalnya, Sun Studio, Sun Java Studio Enterprise, dan Sun Java Studio Creator dari Sun Microsystems semua didasarkan pada NetBeans IDE.



BEKERJA DENGAN NETBEANS 7.4

Ada beberapa langkah utama untuk bekerja dengan Netbeans 7.4 :

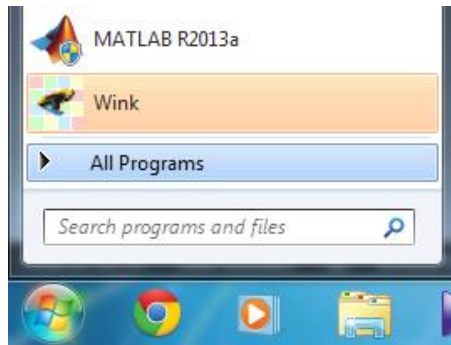
1. Menjalankan Netbeans 7.4

Untuk menjalankan aplikasi Netbeans 7.4, kita dapat mengaksesnya lewat start menu.

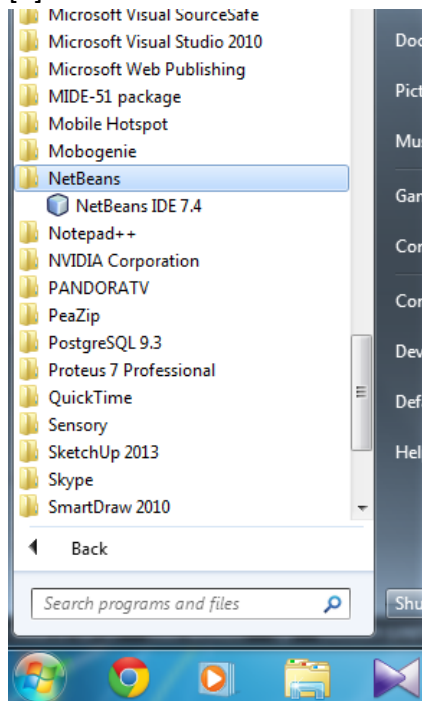
[1] Klik tombol start.



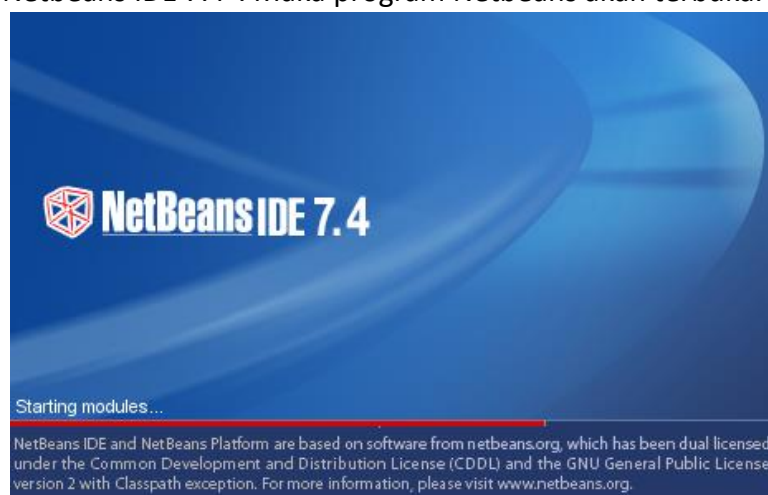
[2] Lalu klik All Programs.



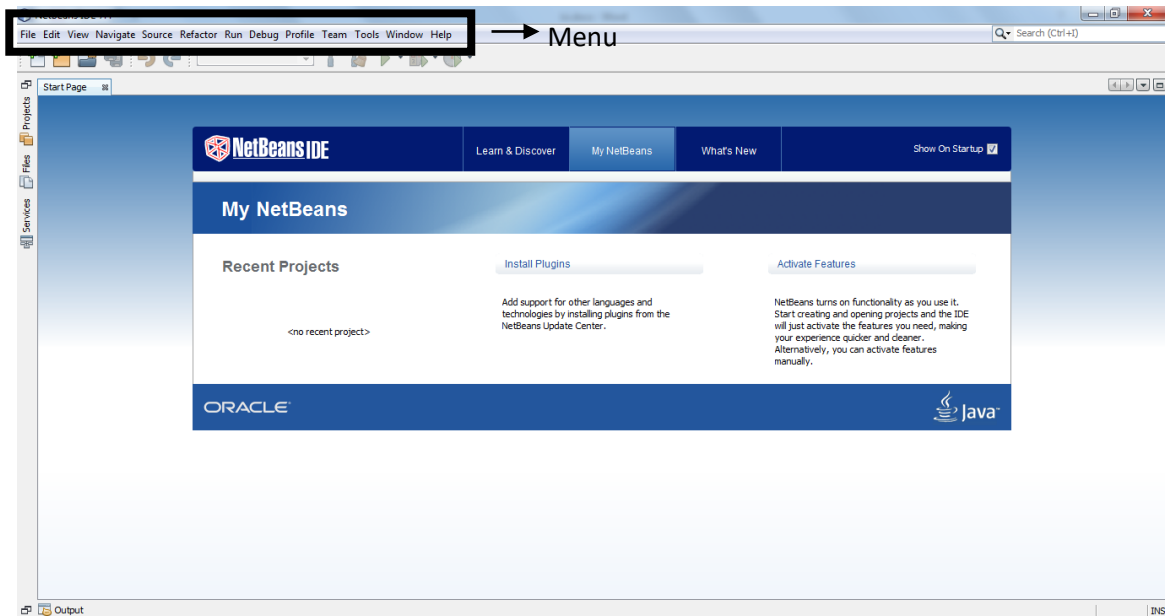
[3] Cari folder bernama Netbeans dan klik.



[4] Lalu pilih “Netbeans IDE 7.4”. Maka program Netbeans akan terbuka.



2. Tampilan utama



3. Contoh program sederhana

```
/*
To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package latihan1;

/**
 *
 * @author
 */

public class Latihan1 {

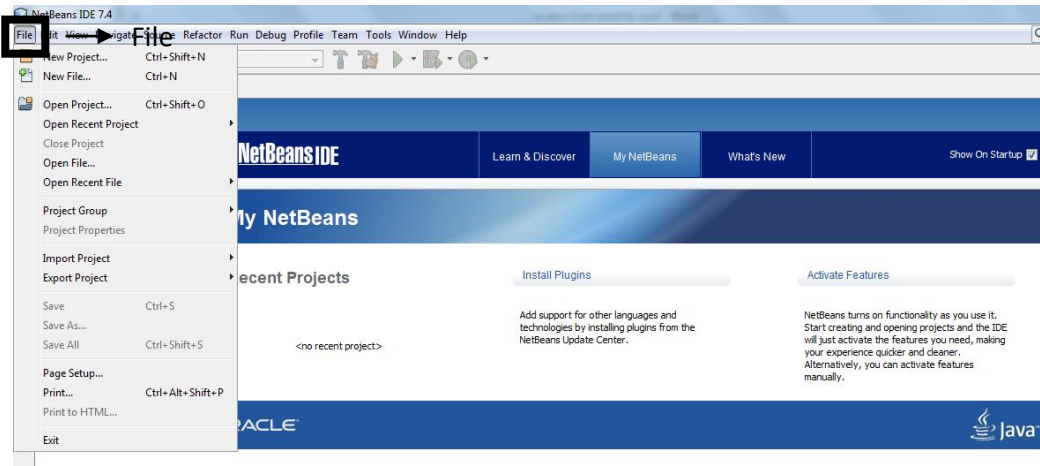
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("Hello World !!!!");
    }

}
```

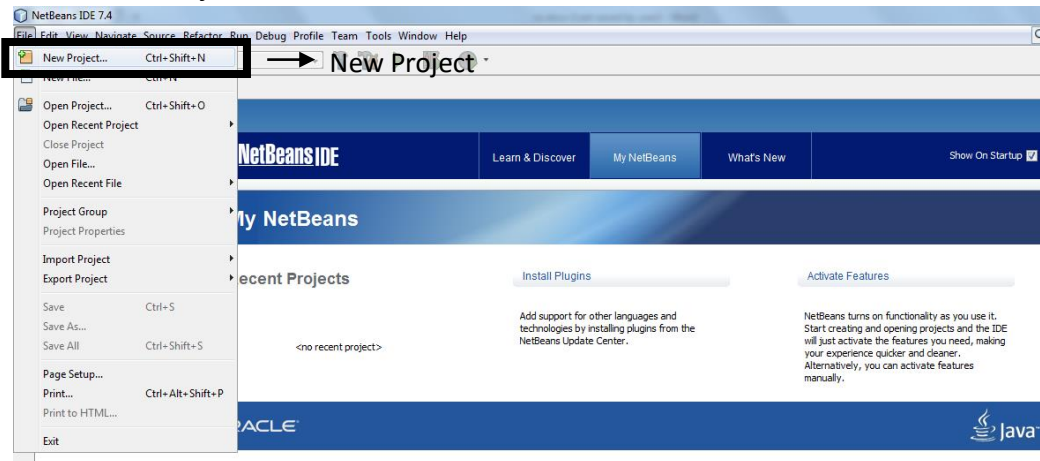
4. Membuat project baru

Berikut langkah – langkah untuk membuat project baru :

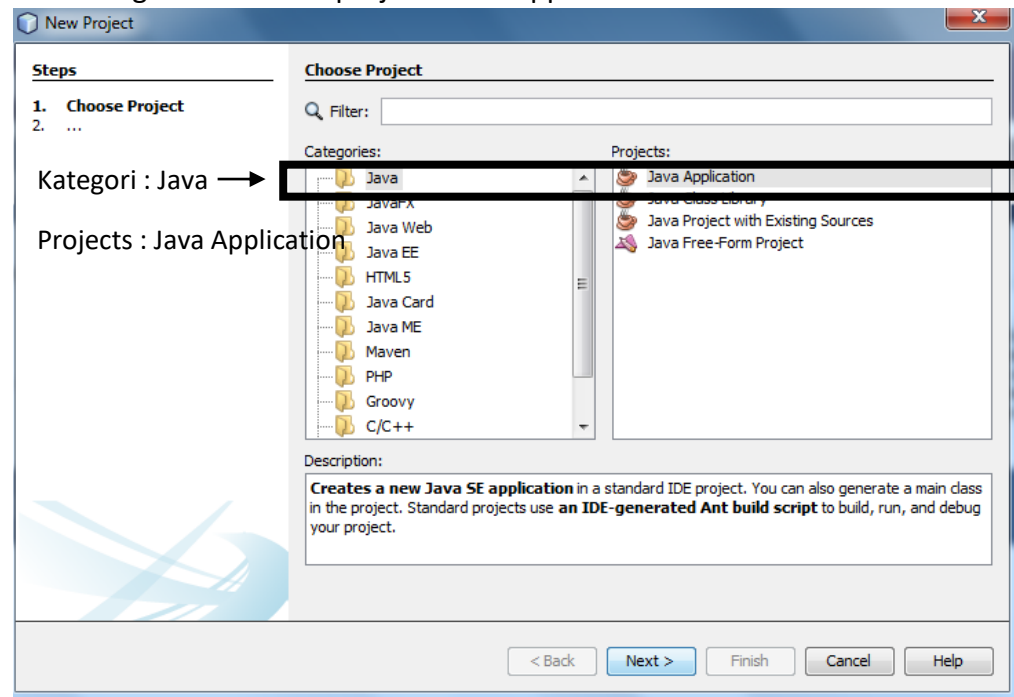
[1] Klik menu “File”



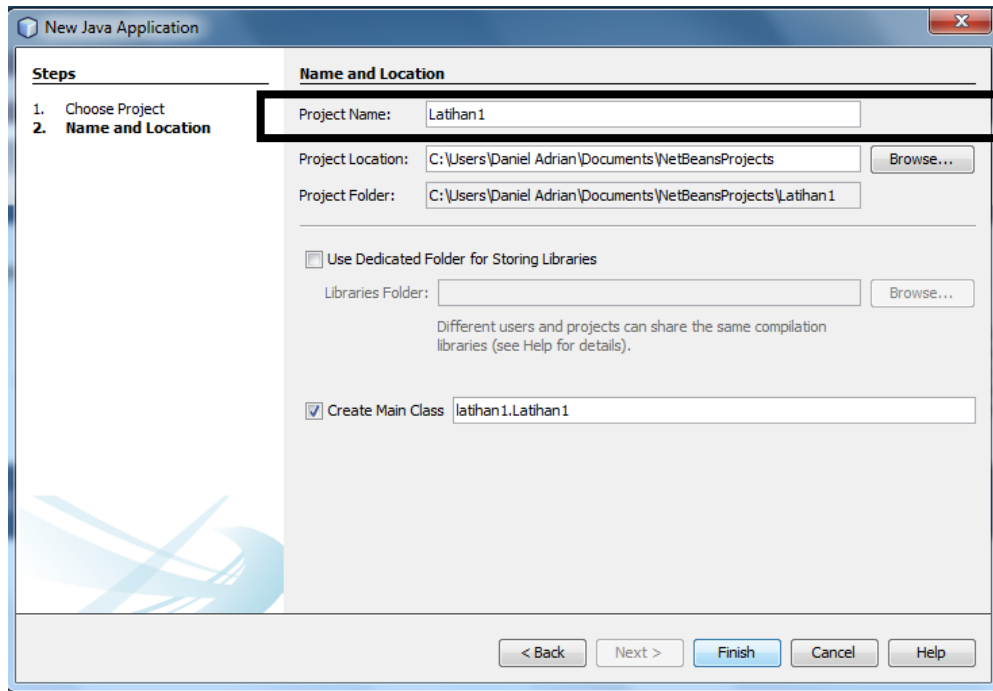
[2] Pilih “New Project”



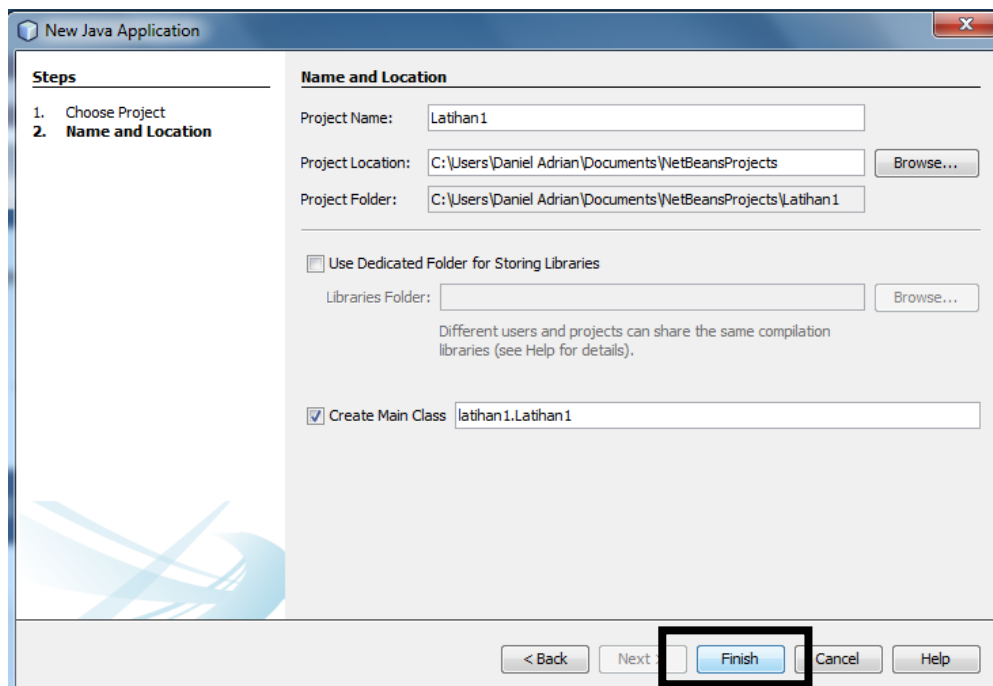
[3] Pilih kategori “Java” dan project “Java Application”



[4] Ketik nama project yang akan dibuat



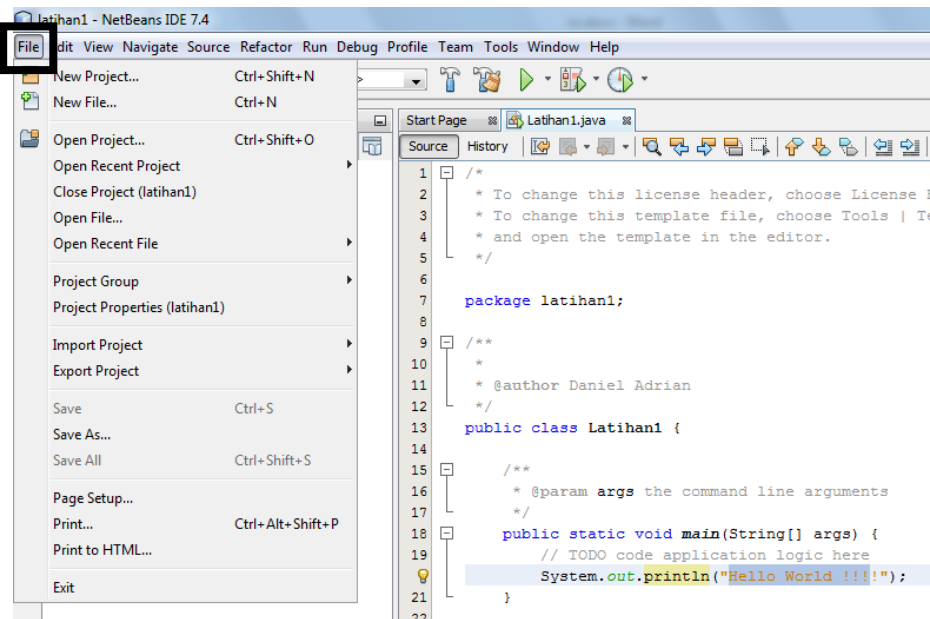
[5] Klik Finish



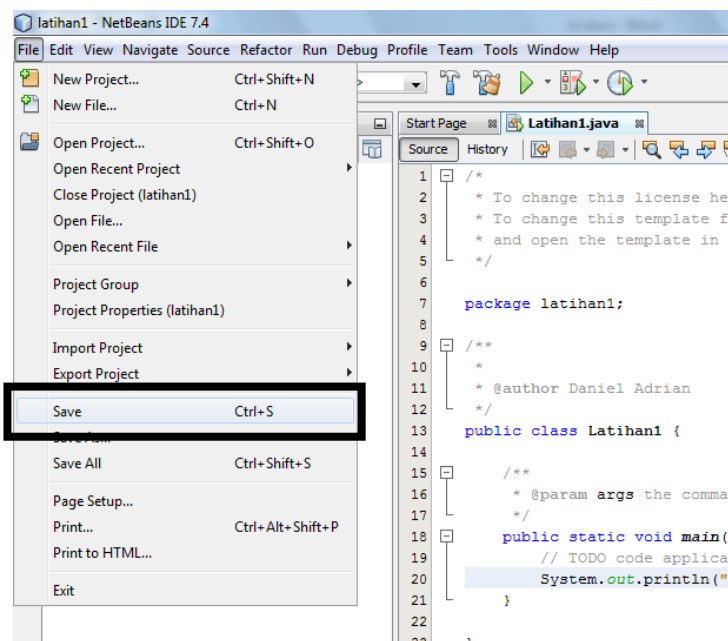
5. Menyimpan project

Berikut langkah – langkah untuk menyimpan project :

[1] Klik “File”



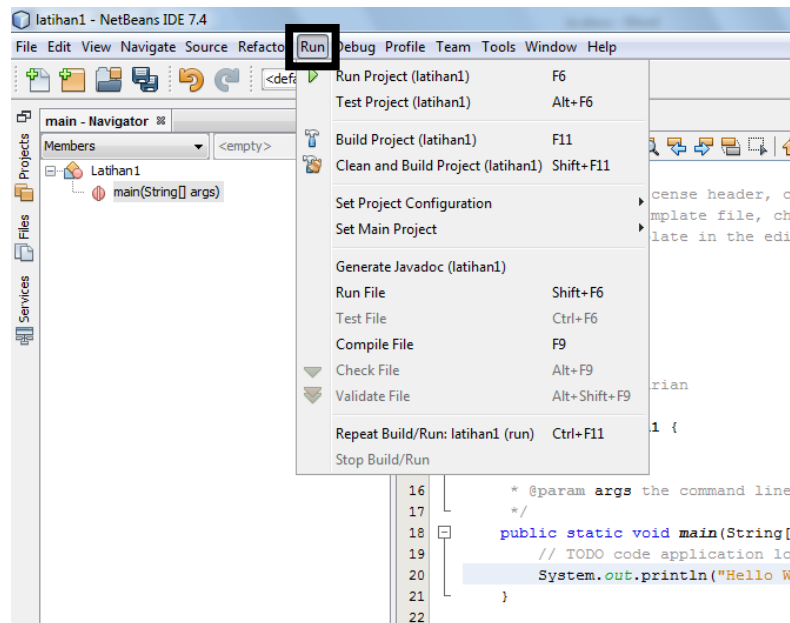
[2] Klik “Save”



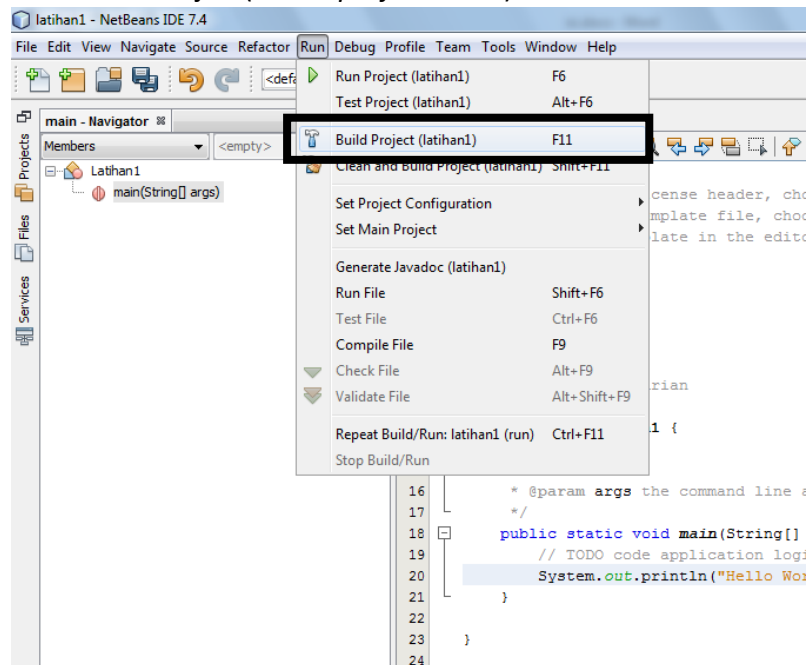
6. Build Project

Berikut langkah – langkah untuk menyusun project :

[1] Klik “Run”



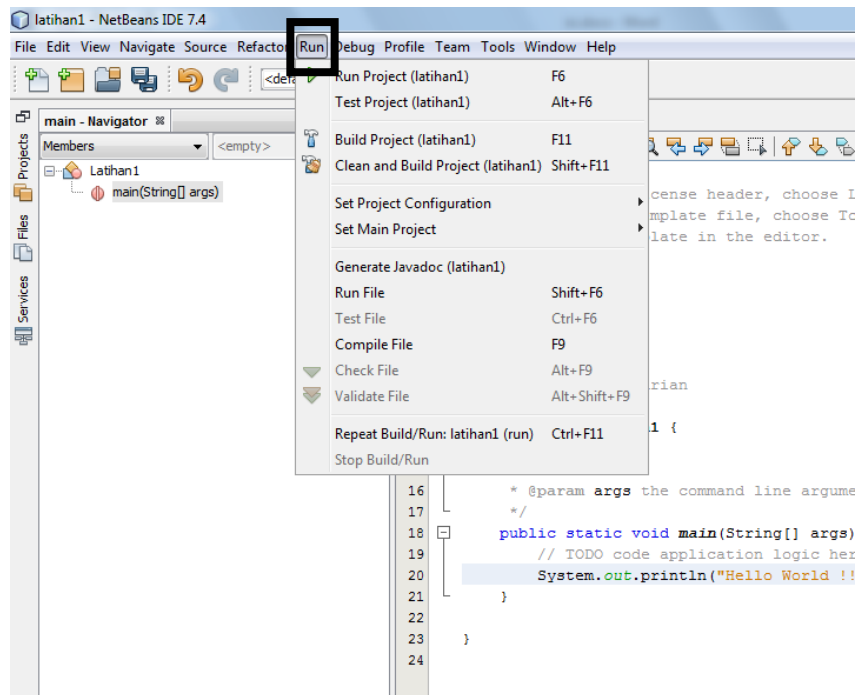
[2] Klik “Build Project(Nama project anda)”



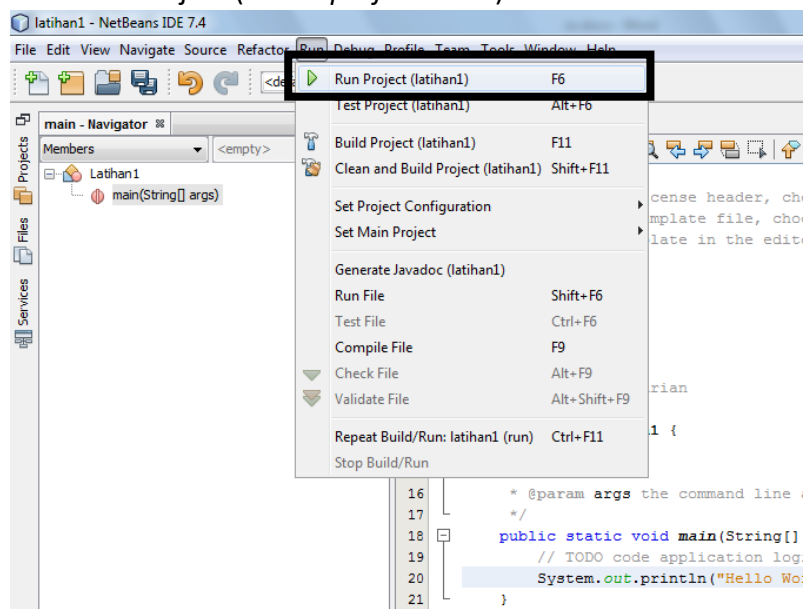
7. Menjalankan project

Berikut langkah – langkah untuk menjalankan project :

[1] Klik “Run”

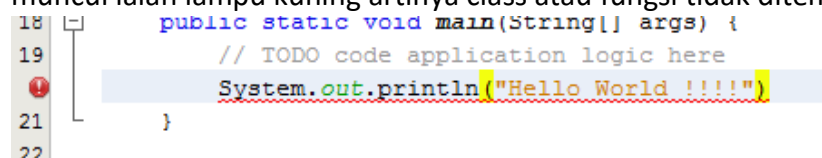


[2] Klik “Run Project (Nama project anda)”



8. Mendeteksi error

Kita mengetahui adanya error dengan cara melihat pada line number. Jika terdapat bola merah berarti dalam penulisan program mengalami kesalahan. Jika lambang yang muncul ialah lampu kuning artinya class atau fungsi tidak ditemukan.



Icon Bola merah


```

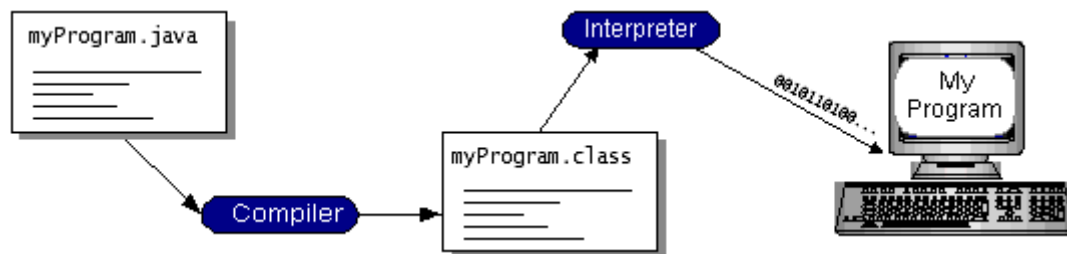
17  */
18  public static void main(String[] args) {
19      // TODO code application logic here
20      System.out.println("Hello World !!!!");
21  }
22

```

Icon Lampu kuning

Perkenalan Java Language

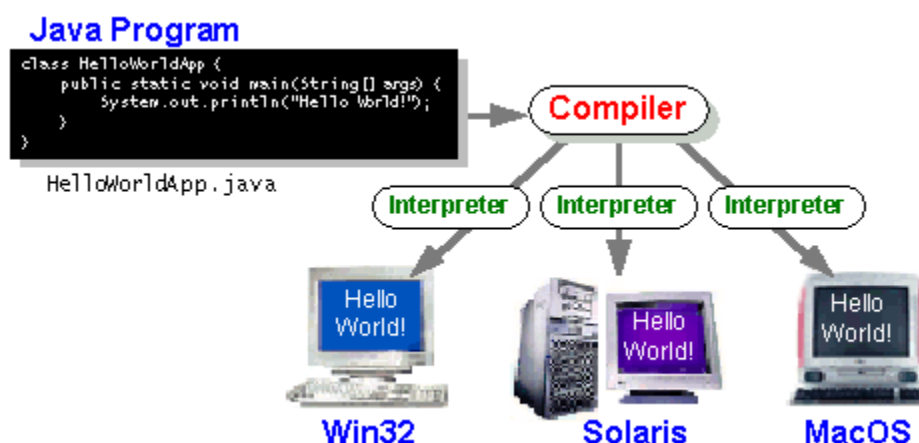
Berbeda dengan bahasa pemrograman yang lain yang hanya perlu di-compile atau hanya di-interpret sudah dapat menjalankan program di komputer, Java membutuhkan kedua hal tersebut (compile dan interpret) baru dapat menjalankan program yang telah dibuat.



Java Virtual Machine (JVM)

Kumpulan kode-kode program yang dibuat terlebih dahulu di-compile dan akan menghasilkan *Java Bytecode* (file *.class* yang akan memuat *Java Bytecode* yang dihasilkan tersebut)

JVM merupakan mesin virtual yang berfungsi menerjemahkan *Java Bytecode* tadi ke bahasa yang dimengerti oleh sistem operasi yang bersangkutan. Dengan adanya *Java Bytecode*, para programmer merasa sangat tertolong karena hanya perlu menulis programnya satu kali saja tapi dapat digunakan di berbagai mesin yang berbeda sepanjang mesin tersebut memiliki sebuah JVM ("write once, run anywhere"). Dengan hal ini lah Java dikatakan bahasa pemrograman yang multiplatform.



Beberapa aturan dasar yang perlu diperhatikan sebelum mulai menulis program Java:

- Setiap memulai untuk menulis program Java harus menggunakan keyword **class**.
- Penggunaan tanda kurung (seperti { }, (), []) harus selalu berpasangan. Penggunaan tanda – tanda kurung tersebut memiliki peran masing – masing yang akan saudara pelajari nantinya.
- Java bersifat *case sensitif*, artinya setiap huruf besar dan huruf kecil akan dibedakan oleh Java.
- Penggunaan tanda ; (titik koma) untuk mengakhiri setiap perintah, pendeklarasian serta hal – hal lain.
- Setiap program yang ditulis harus disimpan dengan nama**java** (..... diisi sesuai dengan nama **class**). Dan setelah di-compile akan menghasilkan**class**. Contoh: file **dkp.java** setelah di-compile akan menghasilkan **dkp.class**. Setiap program **.java** minimal menghasilkan satu file **.class**.

Ketik program → simpan (...**java**) → compile (**javac****java**) → execute (**java****class**) → hasil output

Java Development Kit tidak menyertakan sebuah tool untuk menuliskan program Java kita. Oleh karena itu, untuk menuliskan program Kita harus menggunakan editing tool yang ada seperti Notepad, Textpad, JCreator dsb. Atau kita dapat menggunakan IDE yang lebih canggih seperti Netbeans, JBuilder, Eclipse, BlueJ, dsb tergantung pada kebutuhan kita. Untuk yang menggunakan Notepad terlebih dahulu melakukan setting path pada komputer saudara karena akan melakukan proses compile dan execute melalui command prompt.

Struktur bahasa java

Komentar

Komentar adalah baris program yang tidak ikut di proses atau di kerjakan sebagai suatu perintah oleh compiler atau interpreter. Baris komentar hanya berfungsi sebagai tag atau tanda keterangan tentang baris atau Blok(kumpulan) perintah di bawahnya. Contoh dari program di atas adalah:

//komentar

Komentar di atas menggunakan // sebagai penanda komentar yang berarti satu baris setelah tanda tersebut akan di anggap sebagai komentar yang panjang anda dapat menggunakan tanda/* dan di akhiri tanda */

contoh:

```
/* ini baris komentar hingga 2  
Baris, ini yang pertama  
Ini baris yang kedua*/
```

Pada java anda dapat juga memberikan komentar yang akan di anggap sebagai Javadoc commets yang menggunakan tanda `/**` dan di akhiri `*/` komentar ini di gunakan untuk momberikan dokumentasi tentang class, data dan method yang di gunakan.

Blok

Tanda brace (kurung kurawal) di dalam program yang membentuk sekelompok (satu blok) perintah atau komponen lain dalam program digunakan untuk membentuk sebuah struktur pada program seperti class atau method.

Contoh:

```
Public class latihan1  
{ //awal blok class
```

```
} //akhir dari blok class
```

Class

Setiap program java setidaknya harus memiliki sebuah class karena pada java class adalah struktur program yang paling mendasar. Untuk melakukan pemograman dengan menggunakan bahasa java anda harus mengerti dasar-dasar pemograman berorientasi object.Dan mampu membuat class dan menggunakannya di dalam pemograman .

Format class :

```
Modifier class nama_class{  
}
```

Contoh:

```
Public class latihan1{  
}
```

```
Private class latihan1{  
}
```

Main method

Seperti yang telah disebutkan diatas bahwa tiap class harus memiliki method , dan class utama dalam program juga harus memiliki method utama yang disebut juga main method. main method ini fungsinya mengontrol seluruh alur dari program sewaktu menjalankan tugasnya.

Contoh :

```
Public static void main (String args [] ) {  
}
```

Method / Fungsi

Sebuah class harus memiliki setidaknya sebuah method. Class utama dalam program java harus memiliki method utama .Jadi method harus ada di dalam class dan tidak dapat berdiri

sendiri seperti sebuah fungsi diluar class.

Format penulisan nya:

```
Modifier tipe_data_balikan nama_method (parameter){  
}
```

Contoh :

```
Public void latihan1(){  
}
```

Kata Kunci / Reserved Word

Kata yang sudah memiliki arti tersendiri atau khusus bagi interpreter dan compiler java untuk diterjemahkan menjadi perintah kepada computer untuk mengerjakan sesuatu.

Contoh : *Public, Static, Void, Privat, Protected*

Statement/Syntax

Statement mempresentasikan sebuah aksi atau sebuah urutan aksi dan di akhiri tanda (;) titik koma.

Contoh:

```
System.out.println("HELLO WORLD !!!!");
```

Menggunakan Java Library (Java API):

Java mempunyai library yang merupakan kumpulan dari class yang dapat langsung digunakan, tanpa harus dideklarasikan secara eksplisit terlebih dahulu. Sebelum menggunakannya, terlebih dahulu harus di-import ke kode program kita dengan menggunakan keyword **import**. Contoh:

```
import java.util.*;  
import javax.swing.JOptionPane;
```

Tanda * menunjukkan seluruh class yang terkandung di dalamnya.

Ada ratusan class di Java API, untuk lebih detailnya dapat dilihat pada dokumentasi yang disediakan di situs Java. Atau dapat melihat pada file *src.zip* yang ada di folder instalasi Java. Secara default (tanpa meng-import apa – apa) di kode program yang kita tulis telah terintegrasi library *java.lang.**.

TUGAS

Salinlah program dibawah ini dan berikan pengamatan tentang :

- [1] Output Program ?
- [2] Alur program dan nyatakan dalam flowchart ?
- [3] Variabel yang dipakai pada program dibawah ini ?
- [4] Adakah penggunaan komentar pada program ini ?
- [5] Java API apa saja yang di pakai ?

Program :

```
package latihan1;
```

```
/**
```

```
*  
* @author lab komputer  
*/  
public class Latihan1 {  
  
    /**  
    * @param args the command line arguments  
    */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        double x=10;  
        double y=5;  
        double z ;  
        z=x+y;  
        System.out.println("Hasil 1 : "+z);  
        z=x*y;  
        System.out.println("Hasil 2 : "+z);  
        z=x-y;  
        System.out.println("Hasil 3 : "+z);  
        z=x/y;  
        System.out.println("Hasil 4 : "+z);  
    }  
}
```

Buatlah sebuah program sederhana berdasarkan pengetahuan yang anda tahu setelah melakukan pengamatan.

BAHASA C

C merupakan bahasa rakitan yang dibuat pada tahun 1972 oleh Dennis Ritchie untuk Sistem Operasi Unix di Bell Telephone Laboratories. Meskipun C dibuat untuk memprogram sistem dan jaringan komputer namun bahasa ini juga sering digunakan dalam mengembangkan software aplikasi. C juga banyak dipakai oleh berbagai jenis platform sistem operasi dan arsitektur komputer, bahkan terdapat beberapa compiler yang sangat populer telah tersedia. C secara luar biasa memengaruhi bahasa populer lainnya, terutama C++ yang merupakan ekstensi dari C.

BEKERJA DENGAN DEV C++

Ada beberapa langkah utama untuk bekerja dengan DEV C++ :

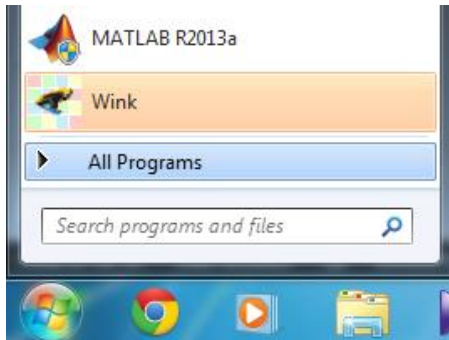
1. Menjalankan DEV C++

Untuk menjalankan aplikasi Netbeans 7.4, kita dapat mengaksesnya lewat start menu.

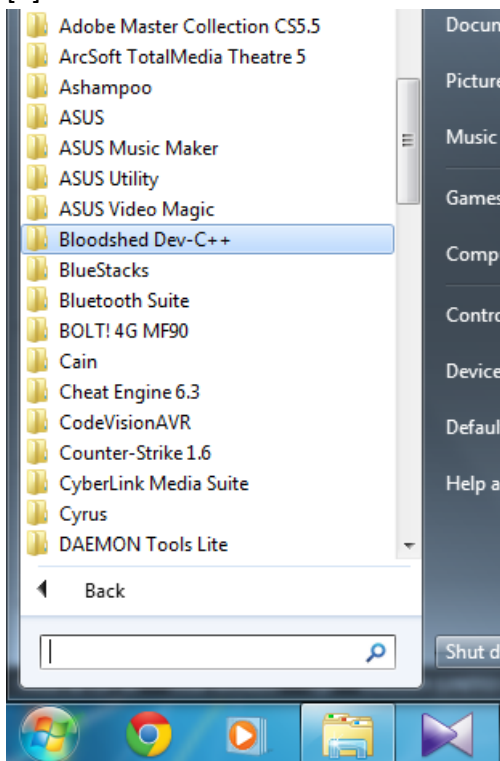
[1] Klik tombol start.



[2] Lalu klik All Programs.



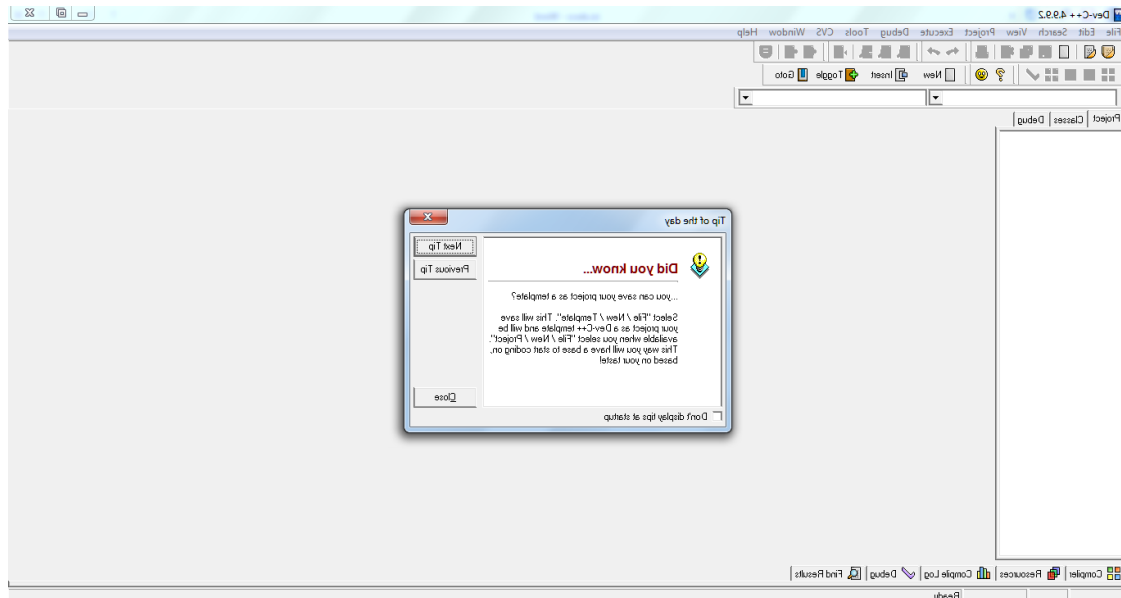
[3] Cari folder bernama Bloodshed DEV C++ dan klik.



[4] Klik DEV-c++ dan DEV C++ akan terbuka



2. Tampilan utama



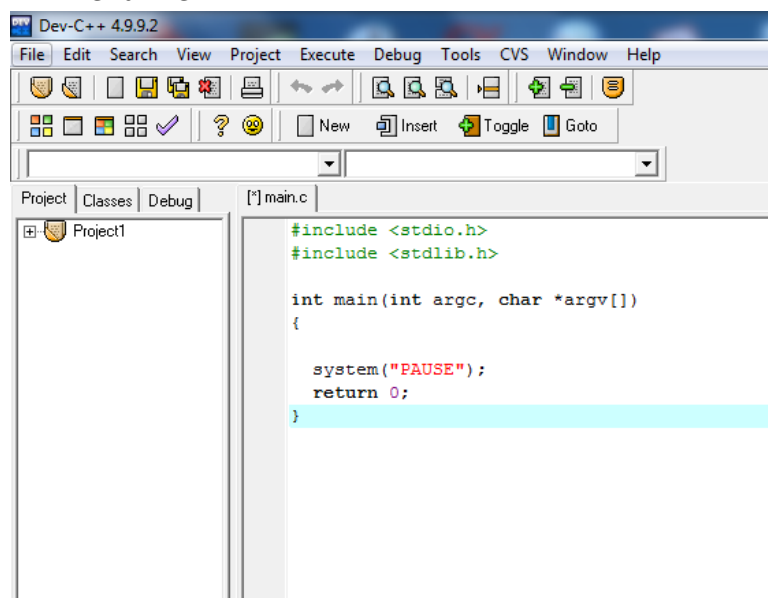
3. Contoh program sederhana

```
#include <stdio.h>
#include <stdlib.h>

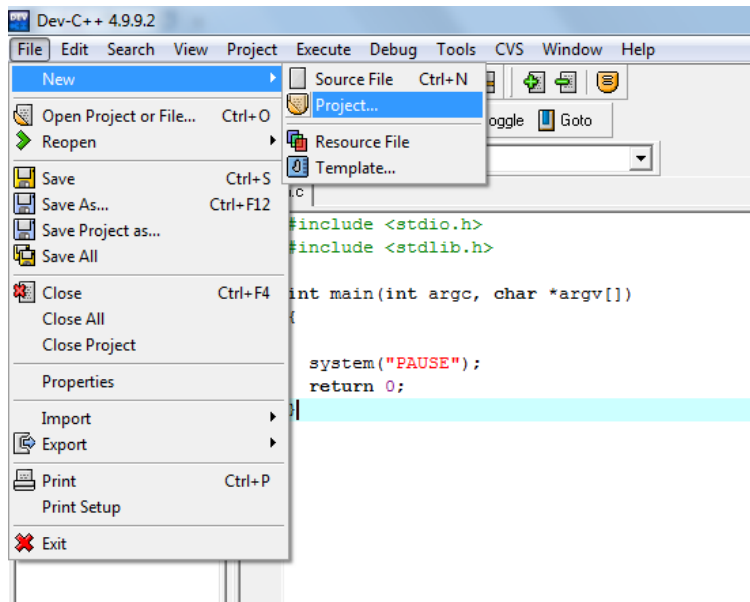
int main(int argc, char *argv[])
{
    printf("HELLO WORLD !!!!");
    system("PAUSE");
    return 0;
}
```

4. Membuat project baru

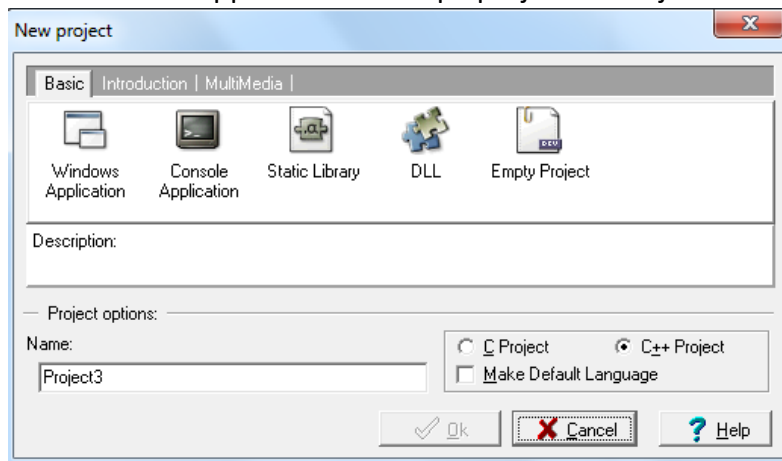
[1] Klik menu File



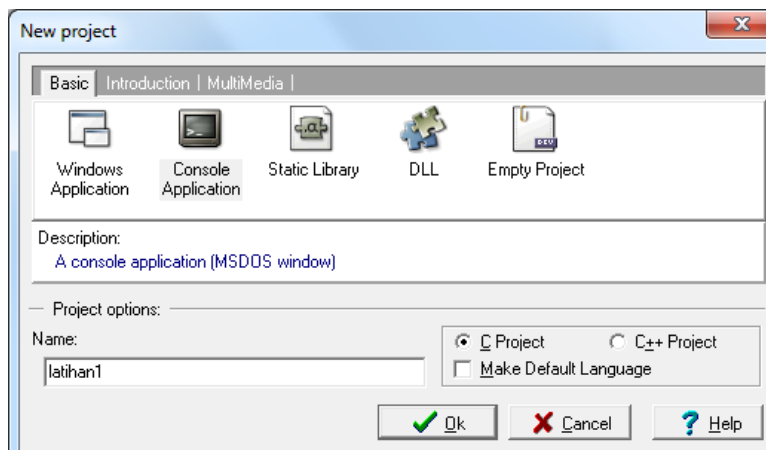
[2] Pilih New -> Project..



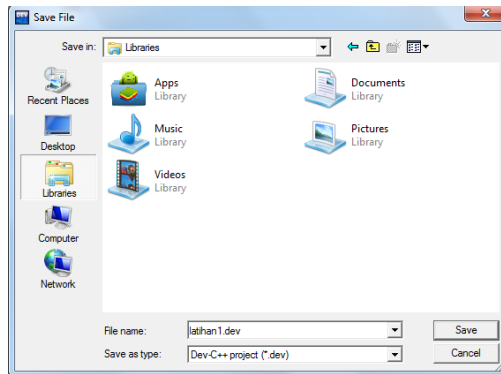
[3] Pilih "Console Application" dan tipe project "C Project" serta isi nama project



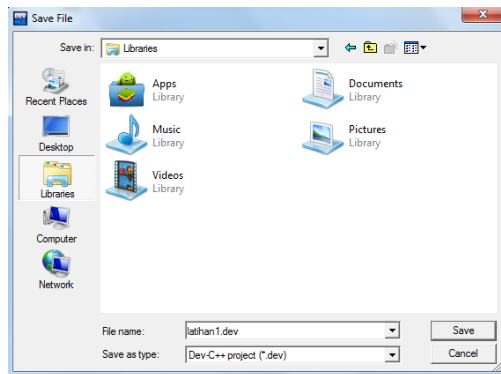
[4] Klik Ok



[5] Maka akan muncul window save file. Pilih letak kita ingin menyimpan project

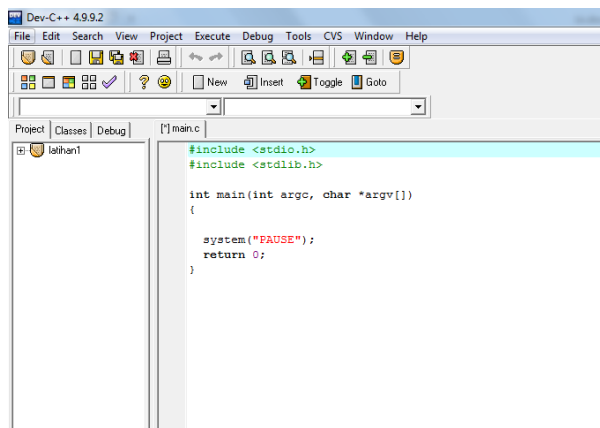


[6] Klik Save

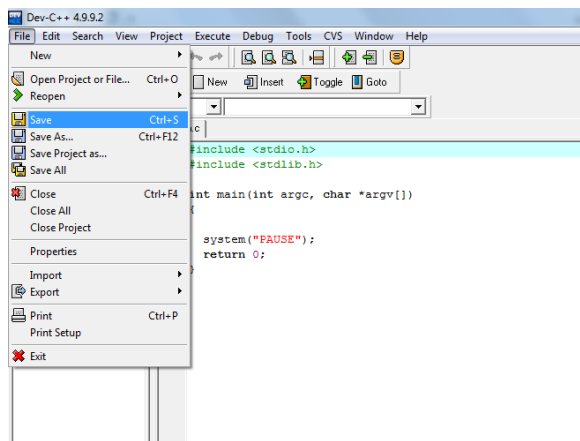


5. Menyimpan project

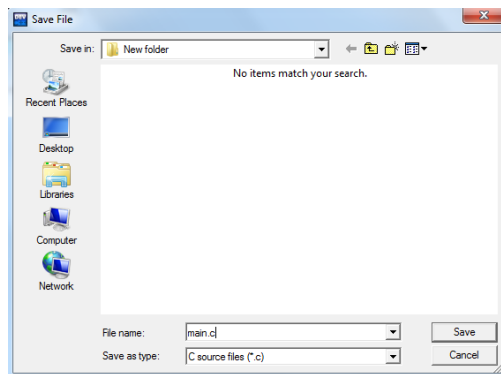
[1] Klik "File"



[2] Klik "save"

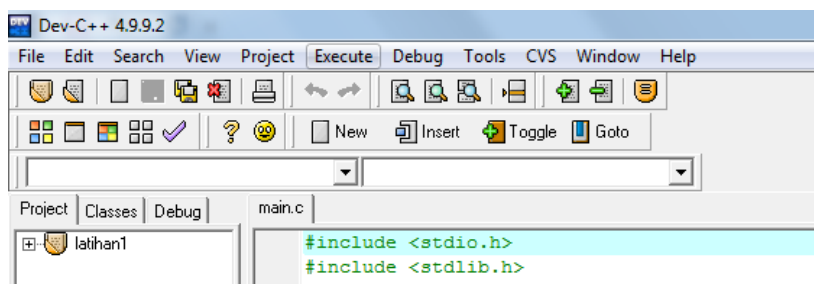


- [3] Jika anda belum melakukan save pada main.c maka akan muncul dialog box lalu klik “Save”

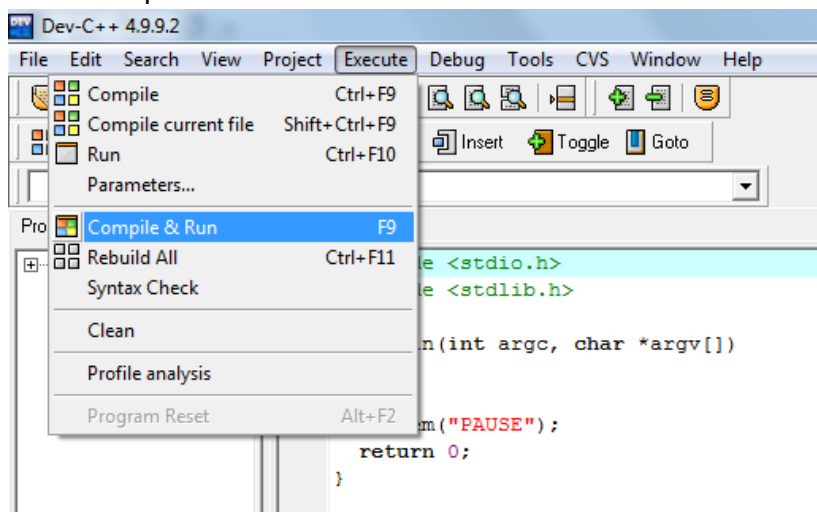


6. Build dan menjalankan project

- [1] Klik “Execute”



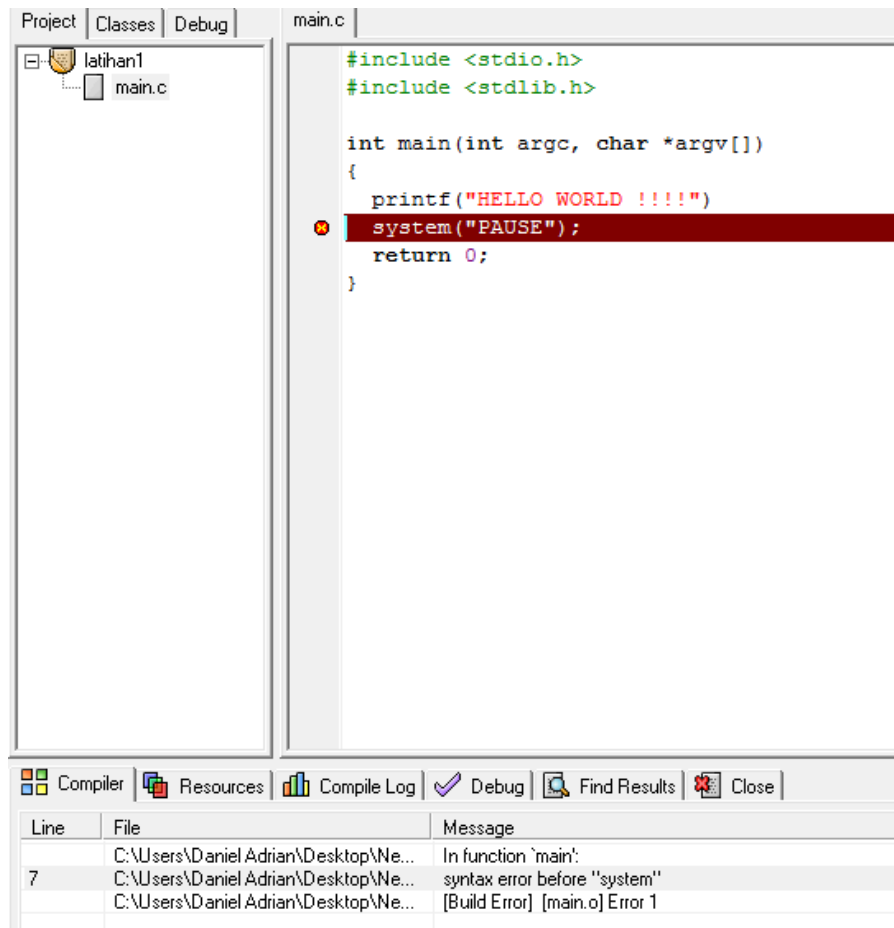
- [2] Pilih “Compile & Run”



7. Mendeteksi error

Error pada DEV C++ dapat di lihat setelah di kompilasi. Error dapat dideteksi dengan melihat tanda silang. Error bisa terjadi karena line tersebut menyebabkan error, line sebelumnya menyebabkan error ataupun adanya kurang penutup pada kurung.

Contoh :



STRUKTUR BAHASA C

Secara umum struktur program Bahasa C/C++ terdiri atas 2 (dua) bagian besar yaitu :

- Bagian Deklarasi/Definisi dan
- Program Utama

Bagian Deklarasi/Definisi

Hal-hal yang didefinisikan pada Bagian Deklarasi/Definisi antara lain :

- Preprocessor Directive (Kompilasi Bercabang akan dibahas pada C++ tingkat lanjut)
- Penyertaan Header (include)
- Pendefinisian Type Data Baru
- Pendefinisian Konstanta
- Pendefinisian Function/Procedure

Penyertaan Header (include)

Header adalah sebuah program dengan ekstensi .H (baca: Titik H) yang berisi kumpulan function/procedure yang berguna untuk pengolahan data.

Syntax pendefinisian penyertaan header :

```
#include "<"<namaheader>".h">"
```

Contoh :

Di dalam DEV C++ terdapat sebuah file header yang bernama "**stdio.h**" yang berisikan kumpulan function/procedure yang berguna untuk pengolahan standar input/output. Untuk menyertakan file "**stdio.h**" tersebut di dalam program adalah sebagai berikut:

```
#include <stdio.h>
```

Program Utama

Di dalam bahasa C standar mutlak harus menggunakan sebuah function/ procedure program utama yang diberi nama "**main**" dengan syntax pendefinisian sebagai berikut:

```
{void | <typevariabel>} main()
```

```
{    statement1;

    [statement1;]

    ...

}
```

Contoh program sederhana dapat dilihat di bawah ini:

```
#include <stdio.h>
```

```
void main()
```

```
{    printf("HELLO WORLD !!!!");

}
```

TUGAS

Salinlah program dibawah ini dan berikan pengamatan tentang :

- [1] Output Program ?
- [2] Alur program dan nyatakan dalam flowchart ?
- [3] Variabel yang dipakai pada program dibawah ini ?
- [4] Adakah penggunaan komentar pada program ini ?
- [5] Header apa saja yang di pakai ?

Program :

```
#include <stdio.h>
#include <stdlib.h>
```


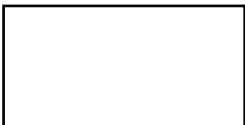
```
int main(int argc, char *argv[])
```

```
{
    double x=15;
    double y=6;
    double z=0;
    int a,b;
    a=x;
    b=y;
    z=x+y;
    printf("Hasil 1 : %lf\n",z);
    z=a%b;
    printf("Hasil 2 : %lf\n",z);
    z=x-y;
    printf("Hasil 3 : %lf\n",z);
    z=x*y;
    printf("Hasil 4 : %lf\n",z);
    system("PAUSE");
    return 0;
}
```

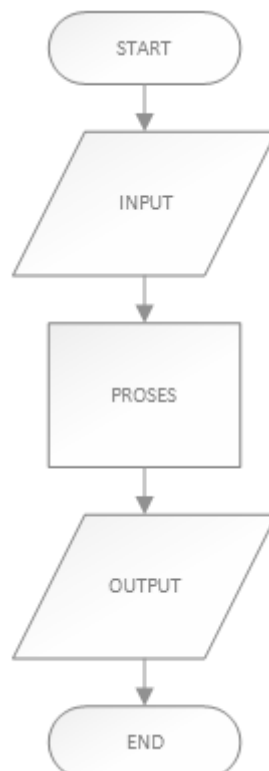
Buatlah sebuah program sederhana berdasarkan pengetahuan yang anda tahu setelah melakukan pengamatan.

PERTEMUAN 2

INPUT, PROSES DAN OUTPUT

Nama	Simbol flowchart
INPUT/OUTPUT	
PROSES	

Input adalah data yang akan diolah. Data – data ini akan diproses sehingga menjadi suatu output yang diinginkan. Proses yang dilakukan untuk masing – masing program berbeda – beda. Secara garis besar flowchart input-proses-output sebagai berikut :



BAHASA JAVA

INPUT

Untuk melakukan input pada java ialah menggunakan JAVA API yang bernama JOptionPane. Syntax yang dipakai untuk menerima input ialah :

```
String a = JOptionPane.showInputDialog("Masukan Input :");
```

Contoh :

```
package latihan1;
import javax.swing.JOptionPane;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String a = JOptionPane.showInputDialog("Masukan Input :");
    }

}
```

PROSES

Input yang kita masukan dapat berupa angka maupun karakter huruf. Jika input yang diterima kita perlu melakukan casting sehingga input yang dalam bentuk string dapat berubah menjadi format angka. Ada dua tipe bilangan yaitu real dan bulat dalam java.

Untuk merubah dari string menjadi integer menggunakan : Integer.parseInt()

Untuk merubah dari string menjadi double menggunakan : Double.parseDouble()

Setelah diubah ke bentuk bilangan maka angka yang dimasukan dapat diolah menggunakan operasi misalkan tambah (+). Untuk karakter mempunyai cara tersendiri untuk mengolahnya namun untuk kali ini belum akan dibahas.

Contoh :

```
package latihan1;
/**
 *
 * @author Daniel Adrian
```

```
*/  
public class Latihan1 {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
int x=10;  
int y=4;  
int z;  
z=x+y;  
    }  
  
}
```

OUTPUT

Untuk menampilkan output dapat menggunakan dua cara yaitu dengan `JOptionPane.showMessageDialog()` dan dengan `System.out.print()`.

Contoh 1:

```
package latihan1;  
/**  
 *  
 * @author Daniel Adrian  
 */  
public class Latihan1 {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
int x=10;  
  
System.out.print("-"+x+"-");  
    }  
  
}
```


Contoh 2 :

```
package latihan1;
import javax.swing.JOptionPane;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int x=10;
        JOptionPane.showMessageDialog(null,"-"+x+"-");

    }
}
```

Contoh program gabungan :

```
package latihan1;
import javax.swing.JOptionPane;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
//Deklarasi
        int x;
        int z;
        int y;
        //
//input
        y=10;
        String a = JOptionPane.showInputDialog("Masukan Input :");
        //
//Proses
        x=Integer.parseInt(a);
```

```
z=x+y;
//
//Output
JOptionPane.showMessageDialog(null,"-"+z+"-");
System.out.println("-"+z+"-");
//
}
```

Bahasa C

INPUT

Untuk melakukan input pada bahasa C kita menggunakan syntax “**scanf**”. Untuk melakukan input pada bahasa C kita perlu untuk mengetahui tipe data dari data yang di input. Ada beberapa tipe data yang dapat diinput

Tipe data	%
Integer	%d
Double	%lf
String (kata tanpa spasi menggunakan array)	%s
Char (hanya satu huruf)	%c

Contoh penggunaan :

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int x;
    double y;
    char z;
    printf("Masukan nilai integer : ");
    scanf("%d",&x);
    printf("Masukan nilai double : ");
    scanf("%lf",&y);
    printf("Masukan nilai char : ");
    scanf("%s",&z);
    system("PAUSE");
    return 0;
}
```

PROSES

Untuk melakukan proses perhitungan pada bahasa C memerlukan tipe data bilangan yaitu integer dan double.

Contoh :

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int x=10;
    int y=20;
    int z;
    z=x+y;
    system("PAUSE");
    return 0;
}
```

OUTPUT

Untuk menampilkan output menggunakan syntax **"printf"**. Jika kita ingin memasukan suatu variable kita perlu mengetahui tipe data disesuaikan seperti tabel di atas pada penjelasan input.

Format :

- printf("Out");
- printf("%d",a); -> untuk integer
- printf("%lf",b); -> untuk double

Contoh :

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int x=10;
    double y=20;
    printf("x = %d\n",x);
    printf("y = %lf\n",y);
    system("PAUSE");
    return 0;
}
```

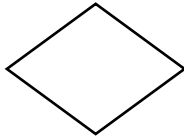
Contoh program gabungan :

```
#include <stdio.h>
#include <stdlib.h>

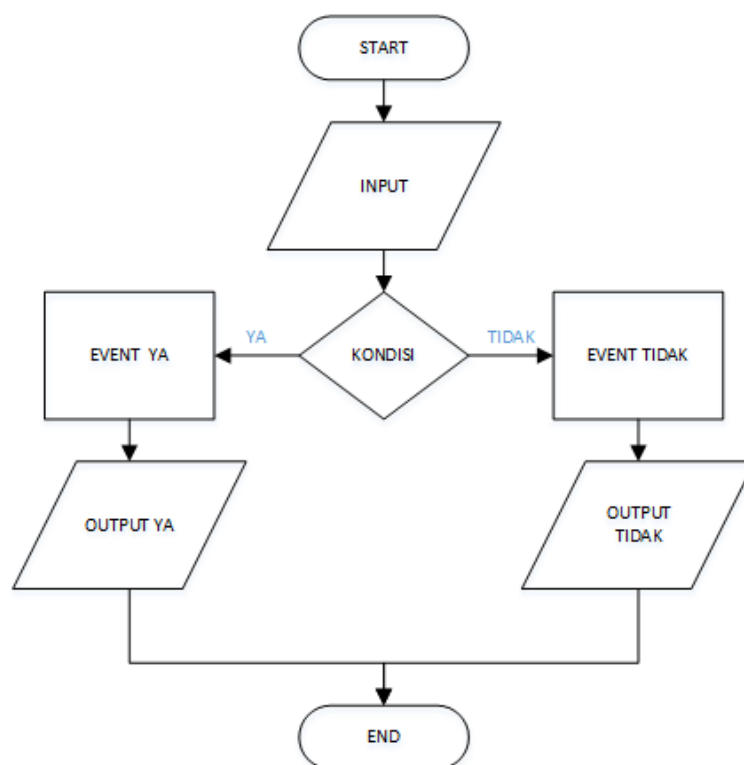
int main(int argc, char *argv[])
{
    //DEKLARASI
    int x;
    int y;
    int z;
    double a;
    double b;
    double c;
    //INPUT
    printf("Masukan nilai x : ");
    scanf("%d",&x);
    printf("Masukan nilai y : ");
    scanf("%d",&y);
    printf("Masukan nilai a : ");
    scanf("%lf",&a);
    printf("Masukan nilai b : ");
    scanf("%lf",&b);
    //PROSES
    z=x+y;
    c=a+b;\
    //OUTPUT
    printf("x = %d\n",x);
    printf("y = %d\n",y);
    printf("z = %d\n",z);
    printf("-----\n");
    printf("a = %lf\n",a);
    printf("b = %lf\n",b);
    printf("c = %lf\n",c);
    system("PAUSE");
    return 0;
}
```

PERTEMUAN 2

BRANCHING DAN MATH OPERATION

Nama	Simbol flowchart
BRANCHING	

Branching atau percabangan adalah suatu kemampuan yang dibuat agar program dapat memilih event tertentu dengan kondisi tertentu. Setiap percabangan memiliki jawaban ya dan tidak. Ya dan tidak ini merupakan tanda jika syarat kondisi terpenuhi maka event/output yang akan dijalankan ialah ya dan jika tidak sesuai dengan kondisi maka yang akan dijalankan ialah tidak. Branching merupakan salah satu bagian dari proses yaitu proses memilih. Dalam kehidupan sehari – hari kita sering melakukan branching misalkan jika hari ini mendung maka saya akan membawa payung, jika tidak saya tidak akan membawa payung. Jika kita melihat ada dua kondisi berbeda yaitu mendung dan tidak mendung. Dan kondisi inilah yang membuat kita menentukan apakah kita ingin membawa payung atau tidak. Secara garis besar flowchart proses branching sebagai berikut :



Beberapa macam Branching:

1. if

Bentuk umum :

```
if (kondisi) {  
    Event  
}
```

Event akan dieksekusi jika memenuhi kondisi yang ada.

Contoh :

BAHASA JAVA

```
package latihan1;  
/**  
 *  
 * @author Daniel Adrian  
 */  
public class Latihan1 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        int x=10;  
        if(x==10){  
            System.out.println("TRUE");  
        }  
    }  
}
```

BAHASA C

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(int argc, char *argv[])  
{  
    int x=10;  
    if(x==10){  
        printf("True");  
    }  
    system("PAUSE");  
    return 0;  
}
```

2. if – else

Berbeda dengan bentuk if, bentuk ini ditambah dengan kondisi else untuk menjelaskan statement yang tidak sesuai dengan kondisi-kondisi boolean yang ada.

Bentuk umum :

```
if (kondisi) {  
    Event 1  
}  
  
else {  
    Event 2  
}
```

Event 1 akan dieksekusi jika memenuhi kondisi yang ada. Selain itu maka Event 2 akan dieksekusi.

Contoh :

BAHASA JAVA

```
package latihan1;  
  
/**  
 *  
 * @author Daniel Adrian  
 */  
  
public class Latihan1 {  
    /**  
     * @param args the command line arguments  
     */  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        int x=10;  
        if(x==10){  
            System.out.println("TRUE");  
        }else{  
            System.out.println("FALSE");  
        }  
    }  
}
```

BAHASA C

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int x=10;
    if(x==10){
        printf("True");
    }else{
        printf("False");
    }
    system("PAUSE");
    return 0;
}
```

3. if – else if

Berbeda dengan bentuk if – else, bentuk ini ditambah dengan kondisi 2 untuk menjalankan statement yang tidak sesuai dengan kondisi 1 yang ada.

Bentuk umum :

```
if (kondisi 1) {
    Event 1
}
else if(kondisi 2) {
    Event 2
}
```

Event 1 akan dieksekusi jika memenuhi kondisi 1. Jika memenuhi kondisi 2 maka Event 2 akan dieksekusi.

BAHASA JAVA

```
package latihan1;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {
    /**
     * @param args the command line arguments
     */
}
```



```
public static void main(String[] args) {  
    // TODO code application logic here  
    int x=10;  
    if(x==10){  
        System.out.println("TRUE");  
    }else if(x==20){  
        System.out.println("FALSE");  
    }  
    }  
}
```

BAHASA C

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(int argc, char *argv[])  
{  
    int x=10;  
    if(x==10){  
        printf("x=10");  
    }else if(x==20){  
        printf("x=20");  
    }  
    system("PAUSE");  
    return 0;  
}
```

4. switch

Bentuk Umum :

```
switch(kondisi syarat) {  
    case 0: .....;  
        break;  
    case 1: .....;  
        break;  
    case 2: .....;  
        break;  
    default: .....;  
}
```

BAHASA JAVA

```
package latihan1;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {
    public static void main(String[] args) {
        // TODO code application logic here
        int x=10;
        switch(x){
            case 10:
                System.out.println("x=10");
                break;
            case 20:
                System.out.println("x=20");
                break;
            default:
                System.out.println("Tidak ditemukan");
        }
    }
}
```

BAHASA C

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int x=10;
    switch(x){
        case 10:
            printf("x=10");
            break;
        case 20:
            printf("x=20");
        default:
            printf("Tidak ditemukan");
    }
    printf("\n");
    system("PAUSE");
    return 0;
}
```

Contoh Program :

BAHASA JAVA

Contoh 1 :

```
package latihan1;

import javax.swing.JOptionPane;

/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int x;
        int y;
        int z;
        String a=JOptionPane.showInputDialog("Masukan nilai x :");
        String b=JOptionPane.showInputDialog("Masukan nilai y :");
        x=Integer.parseInt(a);
        y=Integer.parseInt(b);
        z=x+y;
        if(z<=10){
            System.out.println("Kurang dari 10");
        }else if(z<=20){
            System.out.println("Lebih dari 10 dan kurang dari 20");
        }else{
            System.out.println("Lebih dari 20");
        }
    }
}
```

Contoh 2 :

```
package latihan1;

import javax.swing.JOptionPane;

/**
 *
```

```
* @author Daniel Adrian
*/
public class Latihan1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
int list;
        String rumah = JOptionPane.showInputDialog("Tipe rumah :\n1. Sederhana\nsekali\n2. Sederhana\n3. Biasa\n4. Mewah\nMasukan tipe rumah :");
        list = Integer.parseInt(rumah);
        switch(list) {
            case 1: rumah = "sederhana sekali";
                    break;
            case 2 : rumah = "sederhana";
                    break;
            case 3 : rumah = "biasa";
                    break;
            case 4: rumah = "mewah";
                    break;
            default: rumah = "tidak ada";
        }
        System.out.println("Tipe rumah yang anda pilih "+rumah);

    }

}
```

BAHASA C

Contoh 1 :

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int x;
    int y;
    int z;
    printf("Masukan nilai x : ");
    scanf("%d",&x);
    printf("Masukan nilai y : ");
    scanf("%d",&y);
    z=x+y;
```

```
printf("z=x+y\nz=%d\n",z);
if(z<=10){
printf("Maka kurang dari 10\n");
}else if(z<=20){
printf("Maka lebih dari 10 dan kurang dari 20\n");
}else{
printf("Maka lebih besar dari 20\n");
}
system("PAUSE");
return 0;
}
```

Contoh 2 :

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{

    int list;

    printf("Tipe rumah :\n1. Sederhana sekali\n2. Sederhana\n3. Biasa\n4. Mewah\n");
    printf("Masukan tipe rumah :");
    scanf("%d",&list);
    printf("Tipe rumah yang anda pilih ");

        switch(list) {
            case 1:printf("sederhana sekali\n");
                    break;
            case 2 :printf("sederhana\n");
                    break;
            case 3 :printf("biasa\n");
                    break;
            case 4:printf("mewah\n");
                    break;
            default:printf("tidak ada\n");
        }

    system("PAUSE");
    return 0;
}
```

Skill Improvement

- [1] Buatlah sebuah program yang bertujuan untuk mengukur jumlah sks yang boleh diterima mahasiswa sesuai dengan IP yang diterimanya. Untuk syarat dari IP bebas dengan kriteria IP merupakan bilangan double. (Java)
- [2] Buatlah sebuah menu restoran sederhana yang memberikan harga dari makanan tersebut saat makanan tersebut dipilih. Jumlah makanan minimal 4. Setiap makanan memiliki harga yang berbeda. (C)
- [3] Buatlah sebuah kalkulator sederhana. Input yang dimasukan berupa angka bertipe data double. Jumlah input angka adalah dua. Program ini bertujuan untuk melakukan kalkulasi terhadap input angka pertama dengan input angka kedua dengan sebuah operator. Operator tersebut dipilih oleh user. Operator yang tersedia '+', '-', 'x', '/'. (Java)

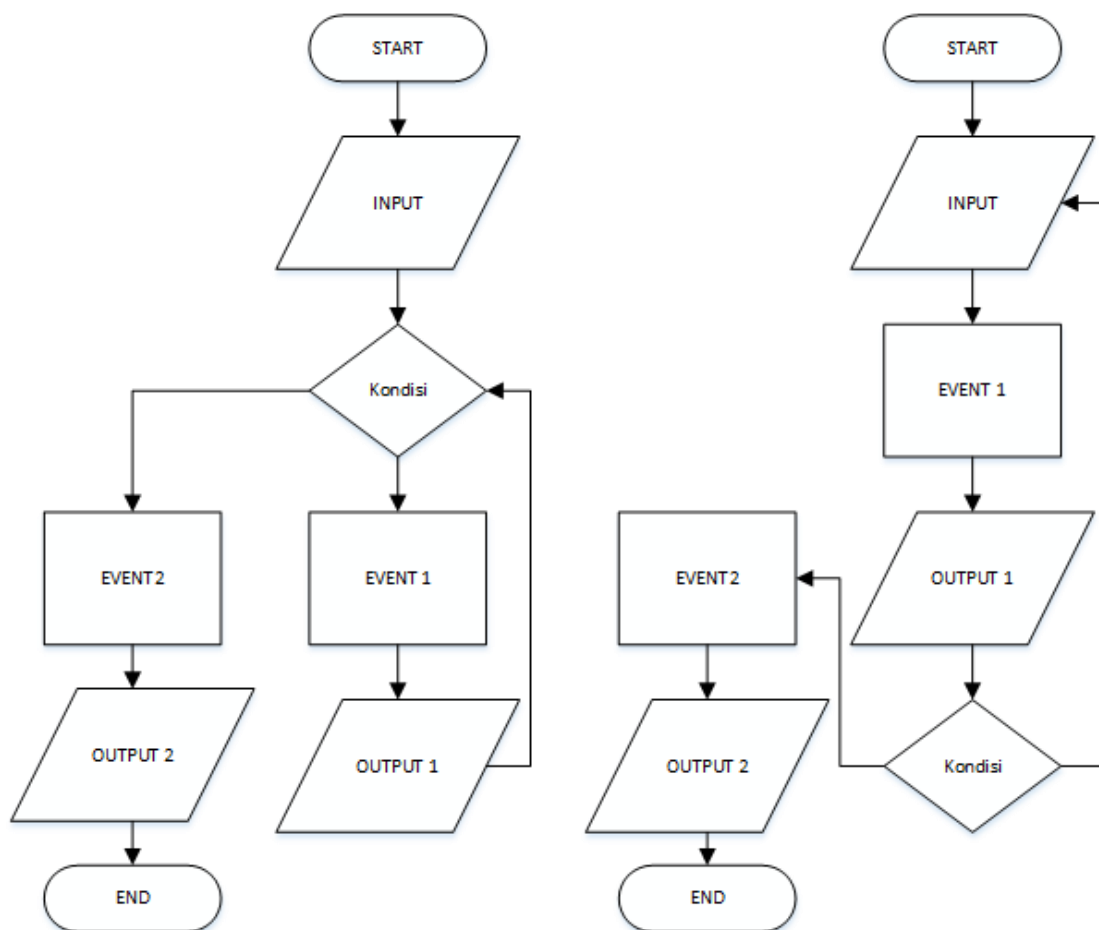
MATH OPERATION

PERTEMUAN 4

LOOPING

Nama	Simbol flowchart
LOOPING	

Looping dipakai untuk mengulang suatu event yang memiliki karakteristik yang sama. Dengan menggunakan looping kita dapat menghemat baris program untuk kondisi yang sama. Contoh perulangan yang ada ialah deret. Berikut adalah penggunaan looping pada flowchart :



Ada 3 macam looping:

1. for

Bentuk umum :

for(....;....;....)

{.....};

BAHASA JAVA

```
package latihan1;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int i;
        for(i=0;i<20;i++){
            System.out.print(i + " ");
        }
    }
}
```

BAHASA C

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i;
    for(i=0;i<20;i++){
        printf("%d ",i);
    }
    printf("\n");
    system("PAUSE");
    return 0;
}
```


2. while

Bentuk umum :

while (... ..)

{.....};

BAHASA JAVA

```
package latihan1;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int i;
        i=1;
        while(i<20){
            System.out.print(i + " ");
            i++;
        }
    }
}
```

BAHASA C

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i;
    i=1;
    while(i<20){
        printf("%d ",i);
        i++;
    }
    printf("\n");
    system("PAUSE");
    return 0;
}
```

3. do-while

Bentuk umum :

```
do{.....  
    .....};
```

```
while{.....};
```

BAHASA JAVA

```
package latihan1;  
/**  
 *  
 * @author Daniel Adrian  
 */  
public class Latihan1 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        int i;  
        i=1;  
        do{  
            System.out.print(i+" ");  
            i++;  
        }while(i<20);  
    }  
}
```

BAHASA C

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(int argc, char *argv[])  
{  
    int i;  
    i=1;  
    do{  
        printf("%d ",i);  
        i++;  
    }while(i<20);  
    system("PAUSE");  
    return 0;  
}
```

Skill Improvement

- [1] Buatlah sebuah deret bilangan ganjil dari 0 sampai 100
- [2] Buatlah deret bilangan genap turun dari 100 samapi 0
- [3] Buatlah deret bilangan ganjil dari 0 sampai 100 dimana setiap 10 angka diganti dengan bilangan genap.
- [4] Buatlah bentuk berikut :

* * * * * *
* * * * * *
* * * * * * * * *
* * * * * * * * *

PERTEMUAN 5

ARRAY

Array adalah kelompok variable dengan tipe sejenis dan dinyatakan dengan nama yang sama. Yang dimaksud dengan tipe yang sejenis adalah kita dapat membuat suatu array dengan tipe data tertentu, misalnya array dengan tipe *integer* atau bertipe *double*.

BAHASA JAVA

Bentuk umum :

ArrayType **VariableName** [];

VariableName = new *ArrayType* [banyak data];

Contoh :

```
package latihan1;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int arr[];
        arr = new int[4];
        arr[0] = 10;
        arr[1] = 20;
        arr[2] = 30;
        arr[3] = 40;
        System.out.println(arr[2]);
    }
}
```

BAHASA C

Bentuk umum :

ArrayType **VariableName**[banyak data];

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i[4];
```

```
i[0]=10;
i[1]=20;
i[2]=30;
i[3]=40;
printf("%d\n", i[2]);
system("PAUSE");
return 0;
}
```

Array Multidimensi

Yang dimaksud dengan Array Multidimensi adalah array yang dapat dibuat dalam susunan [] (1 Dimensi), [] [] (2 Dimensi), [] [] [] (3 Dimensi), dsb. Contoh nya seperti matriks.

BAHASA JAVA

Bentuk umum :

ArrayType **VariableName** [] [];

VariableName = new *ArrayType* [banyak data] [banyak data]...;

```
package latihan1;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        double m[][];
        m = new double [3][3];
        m[0][0]=1;
        m[1][1]=1;
        m[2][2]=1;
        System.out.println(m[0][0]+" "+m[0][1]+" "+m[0][2]);
        System.out.println(m[1][0]+" "+m[1][1]+" "+m[1][2]);
        System.out.println(m[2][0]+" "+m[2][1]+" "+m[2][2]);

    }
}
```

BAHASA C

Bentuk umum :

ArrayType **VariableName** [banyak data][banyak data]....;

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    double m[3][3];
    int i,j;
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            m[i][j]=0;
        }
    }

    m[0][0]=1;
    m[1][1]=1;
    m[2][2]=1;
    printf("%f %f %f\n",m[0][0],m[0][1],m[0][2]);
    printf("%f %f %f\n",m[1][0],m[1][1],m[1][2]);
    printf("%f %f %f\n",m[2][0],m[2][1],m[2][2]);

    system("PAUSE");
    return 0;
}
```

Skill Improvement

- [1] Buatlah sebuah array satu dimensi yang berfungsi untuk menyimpan hasil dari suatu angket untuk seorang dosen. Pilihan dari jawaban angket tersebut disimpan pada array sesuai dengan nomor soal. Untuk pilihan jawabannya ada terdapat 3 jenis yaitu baik, sedang dan buruk. Pada akhir dari angket maka hasil dari angket akan ditampilkan dengan JOptionPane.showMessageDialog. (Java)
- [2] Buatlah dua buah array multidimensi 3 x 3 yang berfungsi menyimpan angka angka bertipe double. Array tersebut berfungsi sebagai matriks. Program ini bertujuan untuk menjumlahkan antara matriks A dan matriks B. (C)
- [3] Buatlah sebuah kalkulator pintar menggunakan array dimana array tersebut berfungsi untuk menyimpan angka yang ingin di kalkulasikan. Jumlah banyaknya data array bisa mencapai 20 angka yang artinya banyaknya angka yang bisa dikalkulasikan diantara 2 sampai 20. Input angka dilakukan pertama sampai angka akhir. Setelah itu dilakukan pemilihan operator antara angka pertama dengan angka kedua. Berikutnya angka pertama dan kedua dengan angka ketiga sampai seterusnya. Operator yang bisa dilakukan '+', '-', 'x', '/'. Hasil dari program menampilkan hasil dari kalkulasi program. (Java)
- [4] Buatlah bentuk - bentuk tersebut dengan array :

```
*
* *
* * *
```

* * *

* *

*

*

* * *

* * * * *

*

* * *

* * * * *

PERTEMUAN 6

STRUKTUR DATA

Dalam kita membuat program, kita menggunakan variabel untuk menyimpan suatu nilai. Namun variabel tersebut terpecah – pecah dengan berbagai nama. Untuk itu dalam program kita dapat mengelompokkan suatu data kedalam sebuah struktur yang sudah didefine. Suatu struktur ini memiliki beberapa variabel yang menyimpan suatu data yang hubungan dengan data lain dalam satu kelompok. Hal ini dinamakan struktur data.

BAHASA JAVA

Untuk bahasa java struktur didefine dengan menggunakan class baru yang dapat diakses. Bentuk dari class adalah :

```
Class Nama_class{  
/*Deklarasi variabel*/  
  
Public Nama_fungsi{  
  
}  
  
}
```

Contoh :

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */  
  
package latclass;  
import javax.swing.JOptionPane;  
/**  
 *  
 * @author Daniel Adrian  
 */  
public class LatClass {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here
```



```
    fungsi func = new fungsi();
    int x;
    String a = JOptionPane.showInputDialog("Masukan Input :");
    x=Integer.parseInt(a);
    func.masuk(x);
    func.display();
}

class fungsi{
private int data;
public fungsi(){
data=0;
}
public void masuk(int d){
data=d;
}
public void display(){
System.out.println("Data : "+data);
}
}
```

BAHASA C

Bentuk dari struktur adalah :

```
struct Nama_struktur{
```

```
Tipedata Nama_variabel ;
```

```
};
```

```
struct Nama_struktur group_struktur;
```

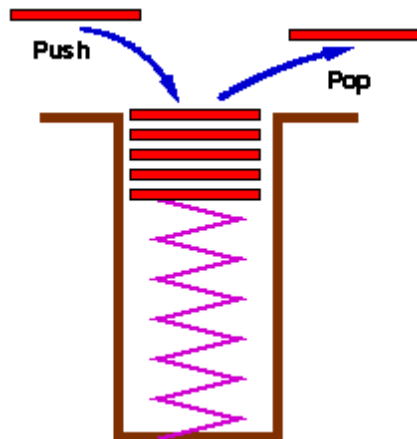
Contoh :

```
#include <stdio.h>
#include <stdlib.h>
#define MaxS 10
struct Struktur{
    int isi[MaxS];
};
struct Struktur struktur;
int main(int argc, char *argv[])
{
    struktur.isi[0]=1;
    printf("Data pada struktur.isi[0] = %d \n",struktur.isi[0]);
    system("PAUSE");
}
```

```
return 0;  
}
```

STACK

Stack atau Tumpukan adalah suatu struktur data yang penting dalam pemrograman. Stack bersifat LIFO (Last In First Out) dimana data yang terakhir masuk ke dalam stack akan menjadi data pertama yang dikeluarkan dari stack.



BAHASA JAVA

Untuk Java kita perlu untuk membuat class baru untuk dapat menggunakan stack. Ada beberapa fungsi yang bisa dipakai oleh stack :

- Init stack
Melakukan inisialisasi class stack dan komponennya :

```
public Stack(){  
    full = false;  
    empty = true;  
    pos = 0;  
} //end of constructor
```

- PUSH
Memasukan data kedalam stack :

```
public void push(int data){  
    if(!isFull()){  
        item[pos++] = data;  
        empty = false;  
        if(pos == max_data) full = true;  
        System.out.println("Data sudah ditambahkan");  
    }  
}
```

```
else{
System.out.println("Stack sudah penuh");
}
} //end of push method
```

- POP

Mengeluarkan data dari dalam stack :

```
//method pop
public int pop(){
int x = 0;
if(!isEmpty()){
x = item[--pos];
full = false;

System.out.println("Data yang di POP adalah : " + item[pos]);
System.out.println("");
item[pos]=0;

if(pos==0)empty = true;
else{
System.out.println("Stack Kosong!");
}
}else{
System.out.println("Stack Masih Kosong!\n");
}
return(x);
} //end of pop method
```

- Display

Menampilkan data pada stack :

```
public void Display(){
System.out.println("Isi Stack Adalah : ");

//printing list item
for(int i=0; i<pos; i++){
System.out.print(item[i]+" ");
}

System.out.println("\n");
} //end of Display
```

- isFull

Melakukan pengecekan apakah stack full :

```
public boolean isFull(){
    return(full);
} //end of isFull method
```

- isEmpty

Melakukan pengecekan apakah stack kosong :

```
public boolean isEmpty(){
    return(empty);
} //end of isEmpty method
```

Berikut program keseluruhan :

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package latihan;
import java.util.Scanner;
/**
 *
 * @author Daniel Adrian
 */
public class Latihan {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int pilihan;
        int data;
        Stack result = new Stack();
        do{
            //Displaying Menu
            System.out.println("1. PUSH Item");
            System.out.println("2. POP Item");
            System.out.println("3. Lihat Isi Data");
            System.out.println("0. Keluar");

            Scanner input = new Scanner(System.in);
```

```
System.out.println("Masukkan Pilihan :");
pilihan = input.nextInt();

//condition for choice
if(pilihan==1){
System.out.println("Data yang ditambahkan :");
data = input.nextInt();
result.push(data);
}
else if(pilihan==2){
result.pop();
}
else if(pilihan==3){
result.Display();
}
else if(pilihan==0){
System.exit(0);
}
else{
System.out.println("Pilihan Tidak Ada!!");
} //end of condition

} while(pilihan!=0); //end looping

}
}

class Stack {
private boolean empty,full;
private int pos;//menunjukkan tempat kosong
private int max_data = 10;
private final int item[] = new int[max_data];

public Stack(){
full = false;
empty = true;
pos = 0;
} //end of constructor

//method isFull
public boolean isFull(){
return(full);
} //end of isFull method

//method isEmpty
public boolean isEmpty(){
```

```
return(empty);
} //end of isEmpty method

//method push
public void push(int data){
    if(!isFull()){
        item[pos++] = data;
        empty = false;
        if(pos == max_data) full = true;
        System.out.println("Data sudah ditambahkan");
    }
    else{
        System.out.println("Stack sudah penuh");
    }
} //end of push method

//method pop
public int pop(){
    int x = 0;
    if(!isEmpty()){
        x = item[--pos];
        full = false;

        System.out.println("Data yang di POP adalah : " + item[pos]);
        System.out.println("");
        item[pos]=0;

        if(pos==0) empty = true;
        else{
            System.out.println("Stack Kosong!");
        }
    }else{
        System.out.println("Stack Masih Kosong!\n");
    }
    return(x);
} //end of pop method

//method Display
public void Display(){
    System.out.println("Isi Stack Adalah : ");

    //printing list item
    for(int i=0; i<pos; i++){
        System.out.print(item[i]+" ");
    }

    System.out.println("\n");
```

```
}//end of Display  
  
}
```

BAHASA C

Suatu stack pada C dapat dideklarasikan dengan cara :

```
#define MaxS n  
Struct Stack  
{  
  TypeData Isi[MaxS] ;  
  TypeData Top;  
};  
struct Stack stack;
```

Dimana :

MaxS = n = jumlah stack yang dapat masuk

Isi = nilai yang terkandung dalam sebuah stack

Top = data teratas yang memiliki value

Ada beberapa function yang digunakan pada Stack :

- **Inisialisasi Stack**

Sebelum Stack dapat dioperasikan; terlebih dahulu diinisialisasikan dengan memberi harga Top = 0.

```
void init(){  
    stack.Top=0;  
}
```

- **PUSH function**

Fungsi PUSH untuk memasukan elemen ke Stack . Jika Stack direpresentasikan dengan menggunakan Array, maka elemen array yang dimasukkan, tidak boleh melebihi ukuran stack. Maka dengan demikian harus ada nama sebagai penunjuk posisi elemen puncak (Top) dari stack dan penunujuk isi maksimum dari stack (Max). Operasi PUSH dapat dilakukan jika stack belum mencapai Max. Berikut ini adalah fungsi PUSH untuk memasukkan elemen ke Stack :

```
void push(int data){  
    if(stack.Top<MaxS){  
        stack.Top++;  
        stack.isi[stack.Top]=data;  
    }else{  
        printf("Data stack penuh...");  
    }
```

```
}  
}
```

- **POP function**

Operasi POP mengambil atau menghapus elemen paling puncak dari Stack. Setiap kali operasi POP dilakukan, maka petunjuk Top akan berkurang satu. Operasi POP dapat dilakukan jika Stack tidak kosong. Berikut ini adalah fungsi POP untuk menghapus elemen pada Stack :

```
void pop(){  
    if(stack.Top!=0){  
        stack.isi[stack.Top]=0;  
        stack.Top--;  
    }else{  
        printf("Data stack kosong...");  
    }  
}
```

- **Mencetak Stack**

Isi suatu stack dapat dicetak dengan menggunakan fungsi berikut :

```
void cetak(){  
    int i=0;  
    if(stack.Top!=0){  
        for(i=1;i<=stack.Top;i++){  
            printf("Data ke %d = %d\n",i,stack.isi[i]);  
        }  
    }else{  
        printf("Data stack kosong...");  
    }  
}
```

- **Full check**

Operasi FULL adalah operasi yang akan digunakan untuk menentukan apakah stack dalam keadaan kosong atau tidak. Suatu Stack dikatakan penuh jika S.Top bernilai MaxS (S.Top=MaxS).

```
void fullcheck(){  
    if(stack.Top<MaxS){  
        printf("Data stack berisi %d dari %d",stack.Top,MaxS);  
    }else{  
        printf("Data stack penuh...");  
    }  
}
```

- **Empty check**

Berfungsi untuk menentukan apakah stack dalam keadaan kosong atau tidak. Suatu Stack dikatakan kosong jika S.Top bernilai nol (S.Top = 0). Berikut adalah fungsinya :

```
void emptycheck(){
```



```
        if(stack.Top!=0){
            printf("Data stack berisi %d dari %d",stack.Top,MaxS);
        }else{
            printf("Data stack kosong...");
        }
    }
```

- **Clear function**

Operasi yang berfungsi untuk mengosongkan Stack. Suatu kondisi bahwa stack dalam keadaan kosong jika Top = 0.

```
void clear(){
    stack.Top=0;
}
```

Program gabungan untuk stack :

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#define MaxS 10
struct Stack{
    int isi[MaxS+1];
    int Top;
};
struct Stack stack;
void init(){
    stack.Top=0;
}

void Spush(int data){
    if(stack.Top<MaxS){
        stack.Top++;
        stack.isi[stack.Top]=data;
        printf("Data %d berhasil dimasukan...\n",data);
    }else{
        printf("Data stack penuh...\n");
    }
    system("PAUSE");
}
```

```
}

void Spop(){
if(stack.Top!=0){
stack.isi[stack.Top]=0;
stack.Top--;
printf("Data teratas berhasil di pop...\n");
}else{
printf("Data stack kosong...\n");
}
system("PAUSE");
}

void cetak(){
int i=0;
if(stack.Top!=0){
for(i=1;i<=stack.Top;i++){
printf("Data ke %d = %d\n",i,stack.isi[i]);
}
}else{
printf("Data stack kosong...\n");
}
system("PAUSE");
}

void fullcheck(){
if(stack.Top<MaxS){
printf("Data stack berisi %d dari %d\n",stack.Top,MaxS);
}else{
printf("Data stack penuh...\n");
}
system("PAUSE");
}
```

```
void emptycheck(){
if(stack.Top!=0){
printf("Data stack berisi %d dari %d\n",stack.Top,MaxS);
}else{
printf("Data stack kosong...\n");
}
system("PAUSE");
}

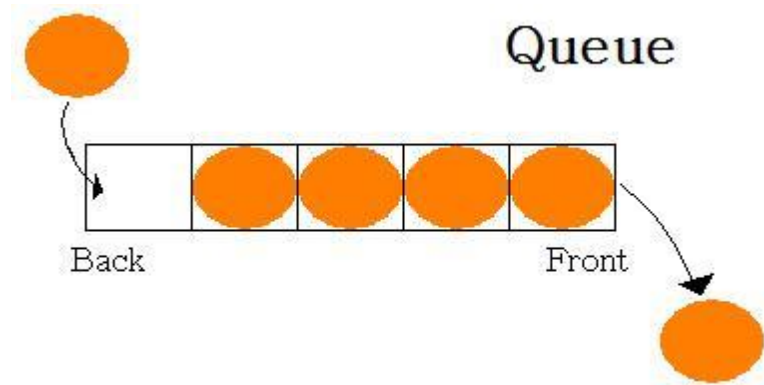
void Sclear(){
printf("Data berhasil di hapus...\n");
stack.Top=0;
system("PAUSE");
}

int main(int argc, char *argv[])
{
int a;
init();
while(1){
system("cls");
printf("Menu Stack :\n1. PUSH\n2. POP\n3. CETAK STACK\n4. FULL CHECK\n5. EMPTY
CHECK\n6. CLEAR\nPilih fungsi yang diinginkan : ");
scanf("%d",&a);
switch(a){
case 1:
system("cls");
printf("Fungsi PUSH\nMasukan data : ");
scanf("%d",&a);
Spush(a);
break;
case 2:
system("cls");
```

```
Spop();
break;
case 3:
system("cls");
cetak();
break;
case 4:
system("cls");
fullcheck();
break;
case 5:
system("cls");
emptycheck();
break;
case 6:
system("cls");
Sclear();
break;
default:
system("cls");
printf("Pilihan anda tidak terdapat dalam daftar.....\n");
system("PAUSE");
}
}
system("PAUSE");
return 0;
}
```

QUEQUE

Queue merupakan sekumpulan elemen dengan penyisipan dan penghapusan elemen yang dilakukan dari sisi/gerbang yang berbeda. Penyisipan dilakukan dari gerbang belakang (back) dan penghapusan dilakukan dari gerbang depan. Queue in bersifat FIFO (Firs In First Out), dengan elemen pertama masuk, akan keluar yang pertama juga. Kalau pada Stack dikenal dengan menggunakan prinsip LIFO (Last In First Out).



BAHASA JAVA

Ada 5 fungsi pada queue yaitu:

- Inisialisasi stack

Sebelum queue dapat dioperasikan; terlebih dahulu diinisialisasikan dengan memberi harga `pos=max_data` :

```
public queue(){
    pos=max_data;
}
```

- PUSH

Memasukan antrian ke paling depan :

```
public void push(){
    int i;
    if(pos==0){
        System.out.println("Antrian Penuh....");
    }else{
        String a = JOptionPane.showInputDialog("PUSH\nMasukan data : ");
        i=Integer.parseInt(a);
        item[pos-1]=i;
        pos--;
        System.out.println("Data berhasil dimasukan....");
    }
}
```

- POP

Mengeluarkan antrian terdepan :

```
public void pop(){
    int i;
    if(pos==max_data){
        System.out.println("Antrian kosong....");
    }else{
        for(i=max_data;i>0;i++){
            item[i]=item[i-1];
            item[i-1]=0;
        }
        pos++;
        System.out.println("Data berhasil di pop...");
    }
}
```

```
}  
}
```

- Cetak

Menampilkan antrian :

```
public void cetak(){  
    int j=1;  
    if(pos==max_data){  
        System.out.println("Antrian kosong....");  
    }else{  
  
        for(int i=max_data;i>pos;i--){  
            System.out.println("Data antrian ke "+j+" = "+item[i]);  
            j++;  
        }  
    }  
}
```

- isFull

Melakukan check apakah antrian sudah penuh :

```
public void isFull(){  
    if(pos==0){  
        System.out.println("Antrian Penuh....");  
    }else{  
        System.out.println("Antrian belum penuh....");  
    }  
  
}
```

- isEmpty

Melakukan check apakah antrian sedang kosong :

```
public void isEmpty(){  
    if(pos==max_data){  
        System.out.println("Antrian kosong....");  
    }else{  
        System.out.println("Antrian tidak kosong....");  
    }  
}
```

Contoh Program gabungan :

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */  
  
package latque;  
import javax.swing.JOptionPane;
```

```
/**
 *
 * @author Daniel Adrian
 */
public class Latque {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        queue result = new queue();
        int pilihan;
        while(true){
            String a = JOptionPane.showInputDialog("Menu queue \n1. PUSH\n2.POP\n3. Cetak\n4.
Full check\n5. Empty check\nMasukan pilihan anda :");
            pilihan = Integer.parseInt(a);
            switch(pilihan){
                case 1:
                    result.push();
                    break;
                case 2:
                    result.pop();
                    break;
                case 3:
                    result.cetak();
                    break;
                case 4:
                    result.isFull();
                    break;
                case 5:
                    result.isEmpty();
                    break;
                default:

            }
        }
    }

    class queue{
        private int pos;//menunjukkan tempat kosong
        private int max_data = 10;
        private final int item[] = new int[max_data+1];
        public queue(){
            pos=max_data;
        }
        public void push(){
```

```
int i;
if(pos==0){
System.out.println("Antrian Penuh....");
}else{
String a = JOptionPane.showInputDialog("PUSH\nMasukan data : ");
i=Integer.parseInt(a);
item[pos-1]=i;
pos--;
System.out.println("Data berhasil dimasukan....");
}
}

public void pop(){
int i;
if(pos==max_data){
System.out.println("Antrian kosong....");
}else{
for(i=max_data;i>0;i++){
item[i]=item[i-1];
item[i-1]=0;
}
pos++;
System.out.println("Data berhasil di pop...");
}
}

public void cetak(){
int j=1;
if(pos==max_data){
System.out.println("Antrian kosong....");
}else{
for(int i=max_data;i>pos;i--){
System.out.println("Data antrian ke "+j+" = "+item[i]);
j++;
}
}
}

public void isFull(){
if(pos==0){
System.out.println("Antrian Penuh....");
}else{
System.out.println("Antrian belum penuh....");
}
}

public void isEmpty(){
if(pos==max_data){
```



```
System.out.println("Antrian kosong....");
}else{
System.out.println("Antrian tidak kosong....");
}
}
}
```

BAHASA C

Ada 5 fungsi yang terdapat pada Queue yaitu:

1. INIT

Sebelum queue dapat dioperasikan; terlebih dahulu diinisialisasikan dengan memberi harga queue.Back=MaxQ;

```
void init(){
queue.Back=MaxQ;
}
```

2. PUSH

Memasukkan data ke antrian terdepan. Fungsi :

```
void Qpush(){
int a;
if(queue.Back==0){
printf("Antrian penuh...\n");
}else{
printf("Fungsi PUSH\nMasukan data : ");
scanf("%d",&a);
queue.isi[queue.Back-1]=a;
queue.Back--;
printf("Data berhasil dimasukan...\n");
}
system("PAUSE");
}
```

3. POP

Mengeluarkan antrian terdepan. fungsi :

```
void Qpop(){
int i;
if(queue.Back==MaxQ){
printf("Antrian kosong...\n");
}else{
for(i=MaxQ;i>0;i--){
queue.isi[i]=queue.isi[i-1];
queue.isi[i-1]=0;
}
queue.Back++;
printf("Data berhasil di pop...\n");
}
system("PAUSE");
}
```

4. CETAK

Untuk mencetak data pada antrian :

```
void cetak(){
    int i,j=1;
    if(queue.Back==MaxQ){
        printf("Antrian kosong...\n");
    }else{
        for(i=MaxQ;i>queue.Back;i--){
            printf("Data antrian ke %d = %d\n",j,queue.isi[i]);
            j++;
        }
    }
    system("PAUSE");
}
```

5. FULL CHECK

Untuk melakukan check isi dari antrian apakah full atau tidak :

```
void fullcheck(){
    if(queue.Back==0){
        printf("Antrian penuh...\n");
    }else{
        printf("Antrian belum penuh...\n");
    }

    system("PAUSE");
}
```

6. EMPTY CHECK

Untuk melakukan check apakah antrian kosong atau tidak :

```
void emptycheck(){
    if(queue.Back==MaxQ){
        printf("Antrian kosong...\n");
    }else{
        printf("Antrian tidak kosong...\n");
    }
    system("PAUSE");
}=-
```

7. CLEAR

Untuk melakukan pengosongan pada antrian :

```
void Qclear(){
    printf("Antrian berhasil di clear...\n");
    queue.Back=MaxQ;
    system("PAUSE");
}
```

Program Gabungan :

```
#include <stdio.h>
#include <stdlib.h>
#define MaxQ 2
struct Queue{
    int isi[MaxQ+1];
    int Back;
};
struct Queue queue;
void init(){
    queue.Back=MaxQ;
}

void Qpush(){
    int a;
    if(queue.Back==0){
        printf("Antrian penuh...\n");
    }else{
        printf("Fungsi PUSH\nMasukan data : ");
        scanf("%d",&a);
        queue.isi[queue.Back-1]=a;
        queue.Back--;
        printf("Data berhasil dimasukan...\n");
    }
    system("PAUSE");
}

void Qpop(){
    int i;
    if(queue.Back==MaxQ){
        printf("Antrian kosong...\n");
    }else{
        for(i=MaxQ;i>0;i--){
            queue.isi[i]=queue.isi[i-1];
            queue.isi[i-1]=0;
        }
        queue.Back++;
        printf("Data berhasil di pop...\n");
    }
    system("PAUSE");
}

void cetak(){
    int i,j=1;
    if(queue.Back==MaxQ){
        printf("Antrian kosong...\n");
    }else{
        for(i=MaxQ;i>queue.Back;i--){
            printf("Data antrian ke %d = %d\n",j,queue.isi[i]);
            j++;
        }
    }
}
```

```
system("PAUSE");
}

void fullcheck(){
if(queue.Back==0){
printf("Antrian penuh...\n");
}else{
printf("Antrian belum penuh...\n");
}

system("PAUSE");
}

void emptycheck(){
if(queue.Back==MaxQ){
printf("Antrian kosong...\n");
}else{
printf("Antrian tidak kosong...\n");
}
system("PAUSE");
}

void Qclear(){
printf("Antrian berhasil di clear...\n");
queue.Back=MaxQ;
system("PAUSE");
}

int main(int argc, char *argv[])
{
int a;
init();
while(1){
system("cls");
printf("Menu Queue : \n1. PUSH\n2. POP\n3. CETAK QueQue\n4. FULL CHECK\n5. EMPTY CHECK\n6. CLEAR\nPilih fungsi yang diinginkan : ");
scanf("%d",&a);
switch(a){
case 1:
system("cls");
Qpush();
break;
case 2:
system("cls");
Qpop();
break;
case 3:
system("cls");
cetak();
break;
case 4:
```

```
system("cls");
fullcheck();
break;
case 5:
system("cls");
emptycheck();
break;
case 6:
system("cls");
Qclear();
break;
default:
system("cls");
printf("Pilihan anda tidak terdapat dalam daftar.....\n");
system("PAUSE");
}
}

system("PAUSE");
return 0;
}
```

PERTEMUAN 7

REKURSIF

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri. Fungsi ini akan terus berjalan sampai kondisi berhenti terpenuhi, oleh karena itu dalam sebuah fungsi rekursif perlu terdapat 2 blok penting, yaitu blok yang menjadi titik berhenti dari sebuah proses rekursi dan blok yang memanggil dirinya sendiri.

Perbedaan dan Persamaan Antara Rekursif dan Iteratif

Persamaan :

- Sama-sama merupakan bentuk perulangan.
- Dilakukan pengecekan kondisi terlebih dahulu sebelum mengulang.

Perbedaan :

- Iteratif menggunakan FOR, WHILE, DO-WHILE sedangkan rekursif hanya menggunakan IF.
- Iteratif dapat berjalan pada program yang terdiri dari prosedur (Tidak terdapat fungsi) sedangkan rekursif merupakan fungsi.

Salah satu aplikasi rekursif biasanya digunakan dalam menghitung perpangkatan dua buah bilangan, factorial dari suatu bilangan bulat, menentukan suku pada deret fibonanci, Tower Hanoi dll.

BAHASA JAVA

Contoh program factorial :

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package factorial;
import javax.swing.JOptionPane;
/**
 *
 * @author Daniel Adrian
 */
public class Factorial {
    public static void main(String[] args) {
        System.out.println("Faktorial ");
        String a=JOptionPane.showInputDialog("Masukan angka :");
        int x=Integer.parseInt(a);
```

```
        System.out.println(x+"!");
        System.out.println("hasil : "+faktorial(x));
        System.out.println("-----");
    }

    static int faktorial(int x){
        if(x==0){
            return 1;
        }else{
            return x*faktorial(x-1);
        }
    }
}
```

BAHASA C

Contoh program fibonanci :

```
#include <stdio.h>
#include <stdlib.h>

void fibonanci(int now,int last,int batas){
    int a;
    a=now+last;
    printf("%d ",a);
    if(a<batas){
        last=now;
        now=a;
        fibonanci(now,last,batas);
    }else{
        return 1;
    }
}

int main(int argc, char *argv[])
{
    int batas=300;
    int now=1;
    int last=1;
    printf("Fibonanci :\n1 1 ");
    fibonanci(now,last,batas);
    system("PAUSE");
    return 0;
}
```

