



# MODUL PELATIHAN

## PEMROGRAMAN WEB



**UPTD BALAI LATIHAN KERJA  
DINAS TENAGA KERJA DAN TRANSMIGRASI KABUPATEN KARAWANG**  
Jl. Surotokunto Km.6 Warung Bambu Kec. Karawang Timur Kab. Karawang  
2023



**BUKU INFORMASI**  
**MENGGUNAKAN STRUKTUR DATA**  
**J.620100.004.01**

KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

## DAFTAR ISI

DAFTAR ISI -----	2
BAB I PENDAHULUAN -----	4
A. Tujuan Umum -----	4
B. Tujuan Khusus -----	4
BAB II MENGIDENTIFIKASI KONSEP DATA DAN STRUKTUR DATA -----	5
A. Pengetahuan yang Diperlukan dalam Mengidentifikasi Konsep Data dan Struktur Data -----	5
1. Pengertian dan Macam-macam Konsep Data dan Struktur Data Serta Pengidentifikasianya Sesuai dengan Konteks Permasalahan -----	5
2. Kelebihan dan Kekurangan Alternatif Struktur Data untuk Konteks Permasalahan yang Diselesaikan -----	16
B. Keterampilan yang Diperlukan dalam Mengidentifikasi Konsep Data dan Struktur Data -----	17
C. Sikap Kerja dalam Mengidentifikasi Konsep Data dan Struktur Data -----	18
BAB III MENERAPKAN STRUKTUR DATA DAN AKSES TERHADAP STRUKTUR DATA TERSEBUT -----	19
A. Pengetahuan yang Diperlukan dalam Menerapkan Struktur Data dan Akses Terhadap Struktur Data Tersebut -----	19
1. Pengimplementasian Struktur Data Sesuai dengan Bahasa Pemrograman yang Akan Dipergunakan -----	19
2. Akses Terhadap Data dalam Algoritma yang Efisien -----	24
B. Keterampilan yang Diperlukan dalam Menerapkan Struktur Data dan Akses Terhadap Struktur Data Tersebut -----	64
C. Sikap Kerja yang Diperlukan dalam Menerapkan Struktur Data dan Akses Terhadap Struktur Data Tersebut -----	64
DAFTAR PUSTAKA -----	65
A. Dasar Perundang-undangan -----	65
B. Buku Referensi -----	65
C. Referensi Lainnya -----	65

DAFTAR PERALATAN/MESIN DAN BAHAN -----	67
A. Daftar Peralatan/Mesin-----	67
B. Daftar Bahan-----	67
DAFTAR PENYUSUN -----	68

## **BAB I**

### **PENDAHULUAN**

#### **A. Tujuan Umum**

Setelah mempelajari modul ini peserta latih diharapkan mampu Menggunakan Struktur Data.

#### **B. Tujuan Khusus**

Adapun tujuan mempelajari unit kompetensi melalui buku informasi Menggunakan Struktur Data ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut:

1. Mengidentifikasi konsep data dan struktur data yang meliputi kegiatan mengidentifikasi konsep data dan struktur data sesuai dengan konteks permasalahan, serta membandingkan kelebihan dan kekurangan alternatif struktur data untuk konteks permasalahan yang diselesaikan;
2. Menerapkan struktur data dan akses terhadap struktur data tersebut yang meliputi kegiatan mengimplementasikan struktur data sesuai dengan bahasa pemrograman yang akan dipergunakan, menyatakan akses terhadap data dalam algoritma yang efisiensi sesuai bahasa pemrograman yang akan dipakai.

## **BAB II**

### **MENGIDENTIFIKASI KONSEP DATA DAN STRUKTUR DATA**

#### **A. Pengetahuan yang Diperlukan dalam Mengidentifikasi Konsep Data dan Struktur Data**

1. Pengertian dan macam-macam konsep data dan struktur data serta pengidentifikasianya sesuai dengan konteks permasalahan.

Secara etimologi (asal kata) definisi data merupakan bentuk jamak dari *datum* yang dalam bahasa Latin berarti pernyataan atau nilai dari suatu kenyataan atau dengan kata lain data adalah representasi dari fakta dunia nyata. Pernyataaan atau nilai ini berasal atau didapat dari proses pengukuran atau pengamatan.

Berikut ini adalah istilah – istilah umum dalam konsep data dan struktur data:

- Variabel adalah lokasi di memori yang kita siapkan dan kita beri nama khas untuk menampung suatu nilai/ tipe data. Sedangkan konstanta adalah suatu harga yang diberikan pada sebuah variabel.
- Pointer adalah variabel yang berisi alamat dari suatu lokasi struktur data. Pointer digunakan untuk menyatakan secara eksplisit alamat tersebut pada waktu dilakukan pengoperasiannya. Manipulasi dapat dilakukan pada alamat maupun struktur data tersebut.
- Tipe data adalah jenis data yang mampu ditangani oleh suatu bahasa pemrograman pada komputer.
- Struktur data adalah cara penyimpanan, pengorganisasian, dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat diinterpretasikan dan operasi-operasi spesifik dapat dilaksanakan pada data tersebut (dapat digunakan secara efisien).

Struktur data berupa tata letak data yang berisi kolom-kolom data, baik kolom yang tampak oleh pengguna (*user*) ataupun kolom yang hanya digunakan untuk

keperluan pemrograman yang tidak tampak oleh pengguna. Setiap baris dari kumpulan kolom-kolom tersebut dinamakan catatan (*record*).

Pemakaian struktur data yang tepat sesuai dengan konteks permasalahan didalam proses pemrograman akan menghasilkan algoritma yang lebih jelas dan tepat, sehingga menjadikan program secara keseluruhan lebih efisien dan sederhana.

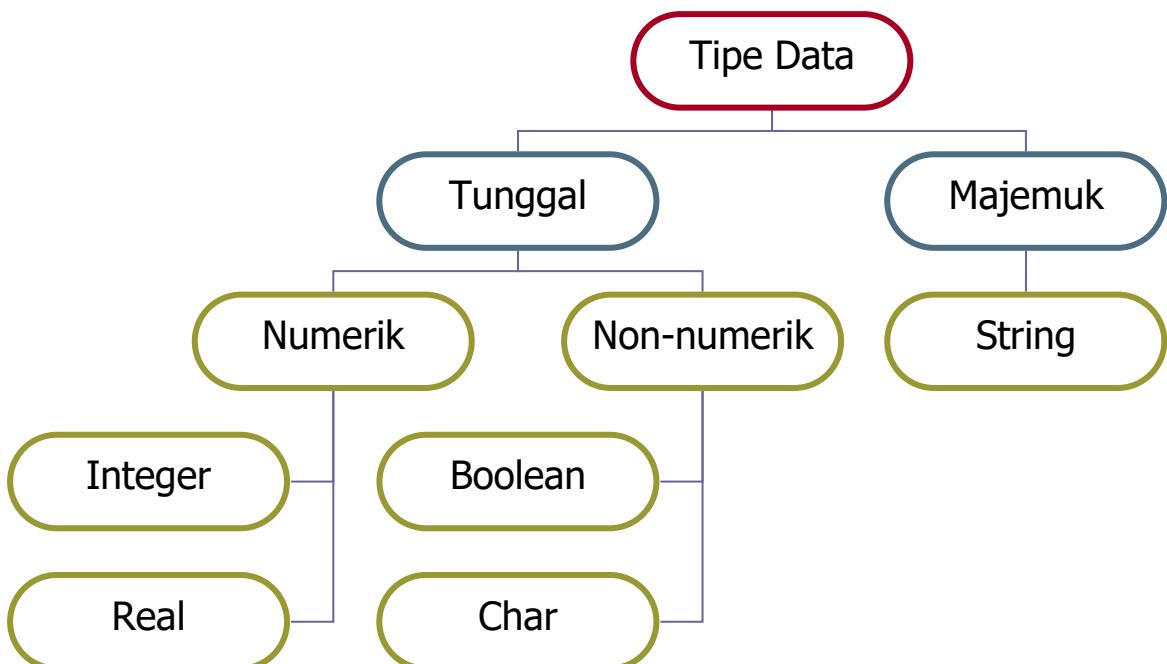


Diagram Ilustrasi Tipe Data

Secara garis besar tipe data dapat dikategorikan menjadi:

1. Tipe data tunggal/ sederhana

Tipe data tunggal adalah tipe data yang hanya mampu menyimpan satu nilai pada setiap satu variabel. Tipe data ini merupakan tipe data dasar yang sering digunakan program, yang termasuk dalam tipe data ini adalah antara lain, namun tidak terbatas pada:

### 1.1 Numerik yang dapat berupa tipe data:

- a. *Integer* (int), merupakan tipe data bilangan bulat yang dapat terdiri dari:

Tipe Data	Ukuran Tempat	Range Nilai
<b>byte</b>	1 byte	0 hingga 255
<b>shortint</b>	1 byte	-28 hingga 127
<b>integer</b>	2 byte	-32768 hingga 32767
<b>word</b>	2 byte	0 hingga 65535
<b>longint</b>	4 byte	2147483648 hingga 2147483647

- b. Riil/ *Real* / *Float* (*floating point*)

Jika Anda bermaksud menyimpan informasi angka dengan format pecahan atau angka desimal, *floating point* adalah jenis tipe data yang sesuai. Angka "65,00" secara teknis merupakan nilai bertipe *floating point* meskipun nilai tersebut juga dapat direpresentasikan sebagai nilai "65". Jenis tipe data ini membutuhkan ukuran memori yang lebih besar dibandingkan jenis tipe data angka *integer*.

### 1.2 Non-numerik yang dapat berupa tipe data:

- a. *Boolean*

*Boolean* merupakan tipe data logika yang hanya bernilai TRUE (benar) dan FALSE (salah). Tipe data ini membutuhkan memori paling kecil bila dibandingkan dengan tipe data lain karena merupakan jenis yang paling sederhana, tetapi cukup sering digunakan dalam setiap pemrograman.

- b. Karakter/ *Character* (char)

Karakter merupakan tipe data yang menyimpan hanya satu digit karakter, karena ukuran satu digit itu 1 *byte* (1 *byte* = 8 bit). Karakter ditulis dengan diapit tanda petik tunggal dan dapat berupa huruf, angka, dan tanda baca atau tanda khusus.

## 2. Tipe data majemuk

### 1. *String*

Tipe data *string* ini memungkinkan variabel menyimpan informasi majemuk berupa untaian karakter, seperti kata atau kalimat. Ciri khas dari tipe data string adalah nilainya di apit oleh tanda *quote* atau petik ganda. Atau dengan tanda *grave accent/backticks* (`), tanda ini umumnya terletak di sebelah kiri tombol 1 pada keyboard komputer. Keistimewaan *String* yang dideklarasikan menggunakan *backticks* adalah membuat semua karakter didalamnya dianggap sebagai *String*.

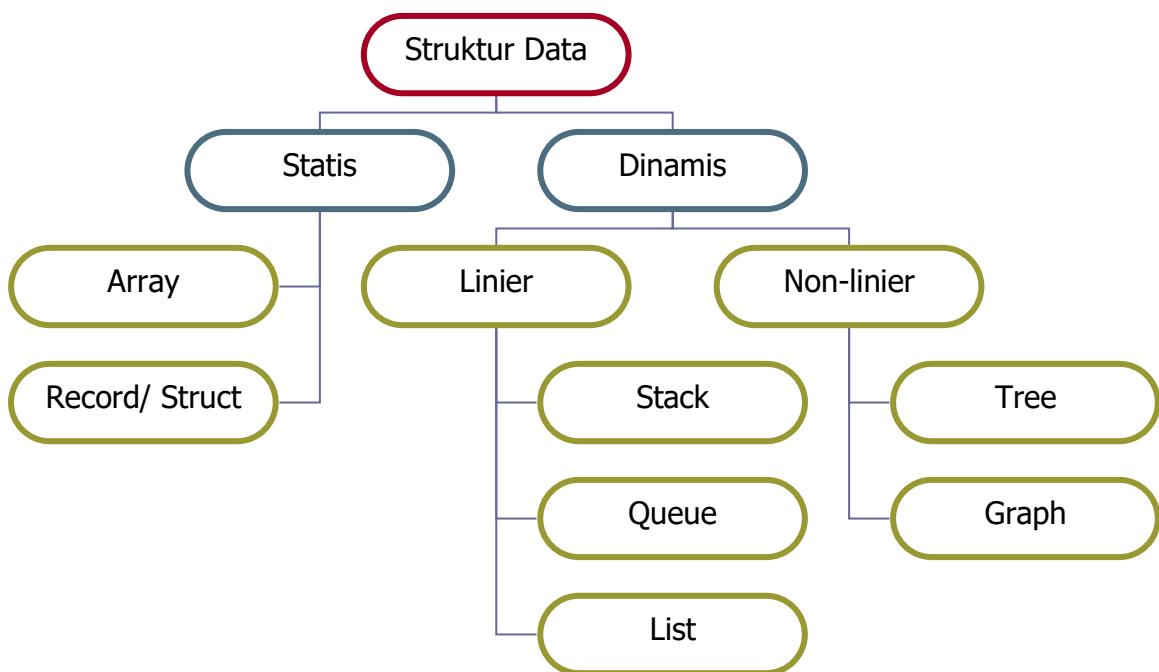


Diagram Ilustrasi Struktur Data

Struktur data yang sering dipakai dalam pemrograman, secara garis besar dapat dikategorikan menjadi:

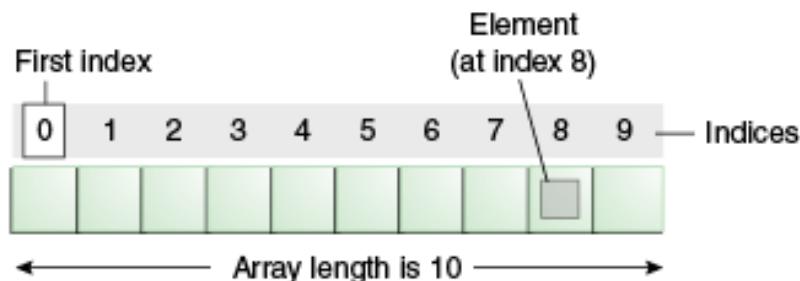
#### 1. Struktur data statis (*fixed-size*)

Struktur data ini besarnya sudah ditentukan dan tidak dapat berubah, yang termasuk dalam kategori ini adalah antara lain, namun tidak terbatas pada:

##### 1.1 Larik/ *Array*

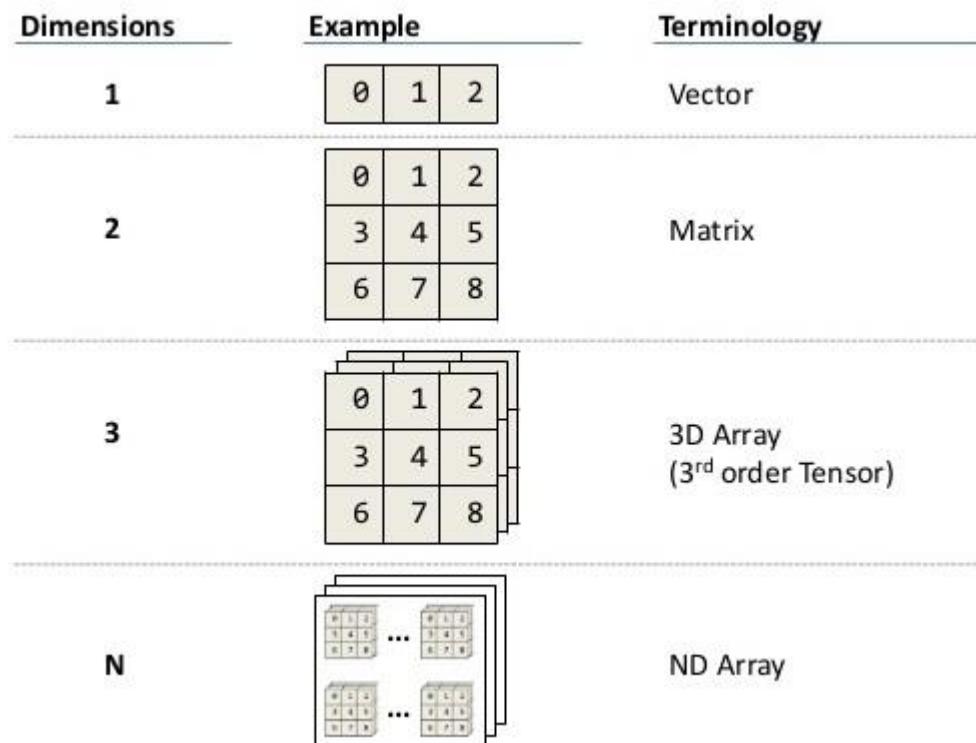
Jenis tipe data *Array* sering disebut juga sebagai tipe data larik. Tipe data ini pada dasarnya merupakan kumpulan sejumlah variabel bertipe data sama (*homogen*) dengan ukuran tertentu (*fixed-size*), yang tersusun

secara beruntun dan disimpan menggunakan sebuah nama yang sama. Nilai-nilai data pada suatu *Array* disebut dengan elemen-elemen *Array*. Letak urutan dari suatu *Array* ditunjukkan atau diakses melalui indeks, dimana indeks pertama adalah 0 (bukan 1).



Gambar Iustrasi *Array* 1 Dimensi

*Array* multidimensi yaitu *Array* yang terdiri dari beberapa indeks. Contoh, *Array* 2 dimensi adalah *Array* yang mempunyai 2 indeks, *Array* 3 dimensi adalah *Array* yang mempunyai 3 indeks.



Gambar Iustrasi *Array* dan *Array* Multidimensi

## 1.2 Struktur/ *Record* (*structure*/ *Struct*)

Istilah *Record* biasa dikenal pada bahasa pemrograman Pascal atau Delphi sedangkan *Struct* lebih dikenal di C/C++. Tipe data *Record*/ *Struct* menampung tipe data yang berbeda-beda (heterogen) atau dapat dikatakan bahwa *Record*/ *Struct* merupakan kumpulan elemen data yang digabungkan menjadi satu kesatuan data dengan ukuran tertentu (*fixed-size*). Masing-masing elemen data tersebut dinamakan *field* atau elemen struktur. *Field* tersebut bisa memiliki tipe data yang sama ataupun berbeda, meskipun *field* tersebut dalam satu kesatuan tetapi tetap bisa diakses secara individu melalui *identifier* atau nama variabel.

## 2. Struktur data dinamis (*dynamic*)

Sering kali kita memerlukan data yang dapat diatur sesuai kebutuhan. Struktur data yang demikian disebut struktur data dinamis.

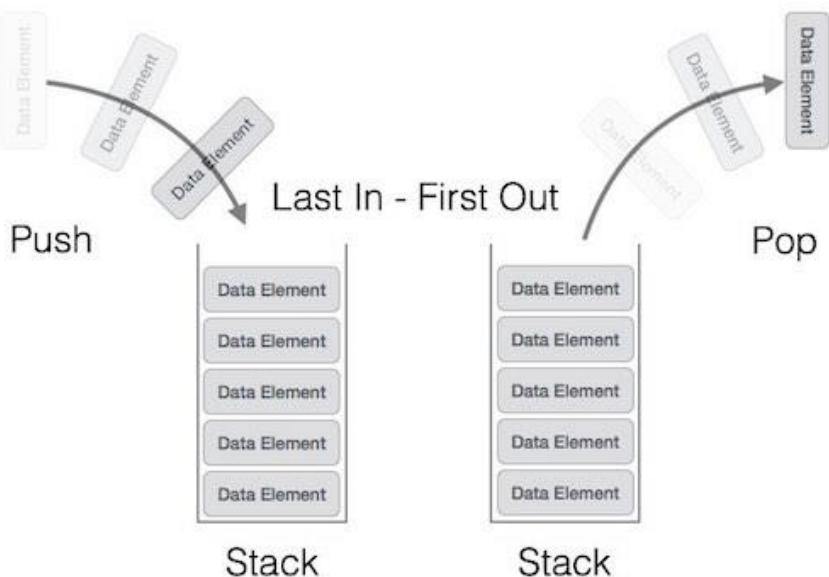
Implementasi dari struktur data dinamis dapat berupa, namun tidak terbatas pada:

### 2.1 Linier

Struktur data linier adalah bentuk struktur data yang berisi kumpulan komponen-komponen/ data yang tersusun secara sekuensial, saling bersambung, dan dinamis membentuk satu garis linier. Bila komponen-komponen ditambahkan (atau dikurangi), maka struktur-struktur tersebut berkembang (atau menyusut).

#### 1.1.1 Tumpukan/ *Stack*

*Stack* adalah sekumpulan data yang organisasi atau strukturnya berupa tumpukan. Elemen-elemen baru atau biasa disebut simpul (node) dapat diletakkan maupun diambil dari sebuah tumpukan hanya dari bagian akhir (atas). Oleh sebab itu, maka tumpukan mengacu pada struktur LIFO (*Last In First Out*), yaitu elemen yang terakhir masuk adalah yang awal dapat dikeluarkan atau kadang juga disebut struktur FILO (*First In Last Out*).



Gambar Ilustrasi Struktur Data *Stack*

### 1.1.2 Antrian/ *Queue*

Struktur data *Queue* adalah suatu bentuk khusus dari list linier dengan operasi penyisipan (*Insertion*) hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (*Rear*) dan operasi penghapusan (*Deletion*) hanya diperbolehkan pada sisi lainnya yang disebut sisi depan (*Front*). Prinsip *Queue* biasa disebut dengan struktur FIFO (*First In First Out*), dimana elemen yang pertama masuk adalah yang awal dapat dikeluarkan.



Gambar Ilustrasi Struktur Data Queue

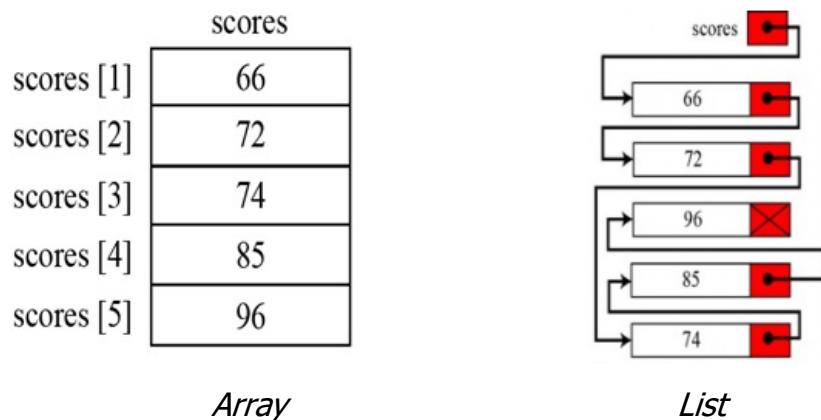
### 1.1.3 *List/ Linked List*

Daftar bertaut atau kadang-kadang disebut dengan senarai bertaut atau senarai berantai dalam ilmu komputer merupakan sebuah struktur data yang digunakan untuk menyimpan sejumlah obyek

data biasanya secara terurut sehingga memungkinkan penambahan, pengurangan, dan pencarian atas elemen data yang tersimpan dalam daftar dilakukan secara lebih efektif. Pada praktiknya sebuah struktur data memiliki elemen yang digunakan untuk saling menyimpan rujukan antara satu dengan lainnya sehingga membentuk sebuah daftar abstrak, tiap-tiap elemen yang terdapat pada daftar abstrak ini seringkali disebut sebagai node. Karena mekanisme rujukan yang saling terkait inilah disebut sebagai daftar berantai/ *Linked List/ List*.

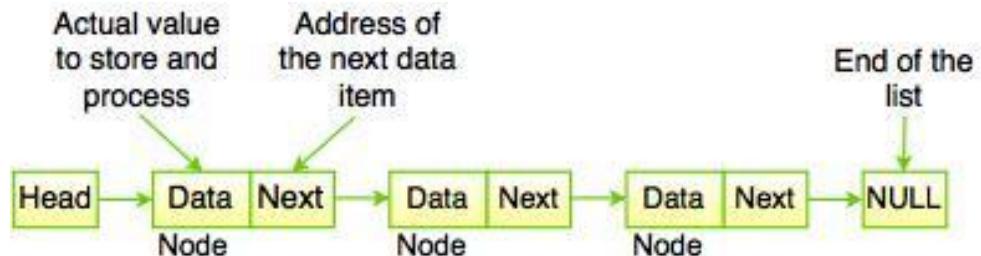
Walau sepintas terkesan List mirip dengan Array, namun terdapat perbedaan diantara keduanya, berikut diantaranya:

<b>Array</b>	<b>List</b>
Alokasi memori statis	Alokasi memori dinamis menyesuaikan dengan banyak data
Setiap elemen berdiri sendiri, tidak ada keterkaitan antar elemen, sehingga pengaksesan dapat dilakukan secara random dan lebih cepat.	Setiap elemen saling berhubungan dimana alamat/ lokasi dari setiap elemen disimpan dalam bagian dari elemen sebelumnya, sehingga untuk pengaksesan harus dilakukan secara berurut.



Gambar Ilustrasi *Array* dan *List*

*List* linier secara sederhana dapat diilustrasikan sebagai berikut:



Gambar Ilustrasi *List* Sederhana

a. *Single Linked List*

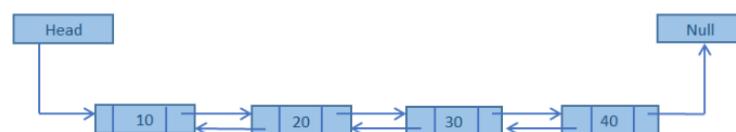
Single Linked List adalah sebuah linked list yang menggunakan sebuah variable pointer saja untuk menyimpan banyak data dengan menggunakan metode *linked list*, suatu daftar isi yang saling berhubungan dengan susunan yang saling berurutan, sehingga perpindahan/ lintasan hanya dapat terjadi satu arah.



Gambar Ilustrasi *Single Linked List*

b. *Double Linked List*

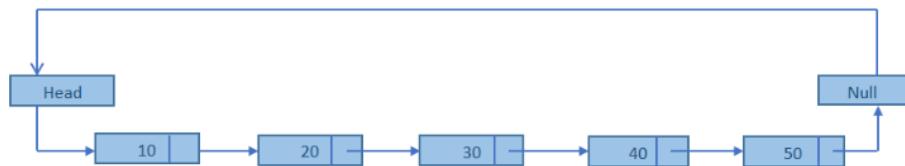
Double Linked List adalah file pointernya terdiri dari dua buah dan dua arah, yaitu *next* (berikutnya) dan *prev* (sebelumnya), sehingga dapat melintasi/ pergerakan dapat terjadi dari dua arah.



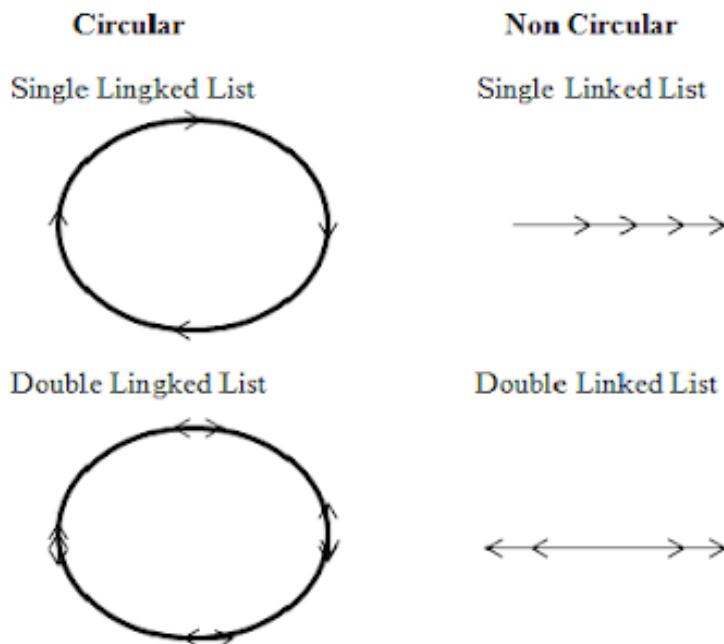
Gambar Ilustrasi Double Linked List

c. *Circular Linked List*

Pada circular linked list, node pertama dan terakhir berdampingan/ terhubung.



Gambar Ilustrasi *Circular Linked List*



Gambar Ilustrasi *List Circular* dan *Non-Circular*

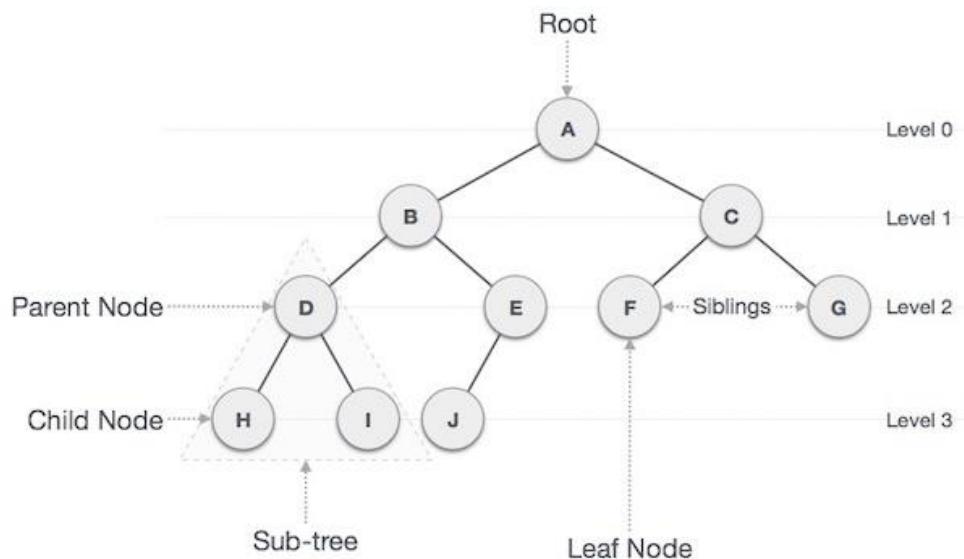
## 2.2 Non-linier/ tidak linier

Struktur data Non-linier adalah bentuk struktur data yang tidak memenuhi prinsip superposisi dimana komponennya tidak tersusun secara berurut.

### 2.2.1 Pohon biner/ *Tree*

Merupakan salah satu bentuk struktur data tidak linier yang menggambarkan hubungan yang bersifat hirarkis yang tidak linier

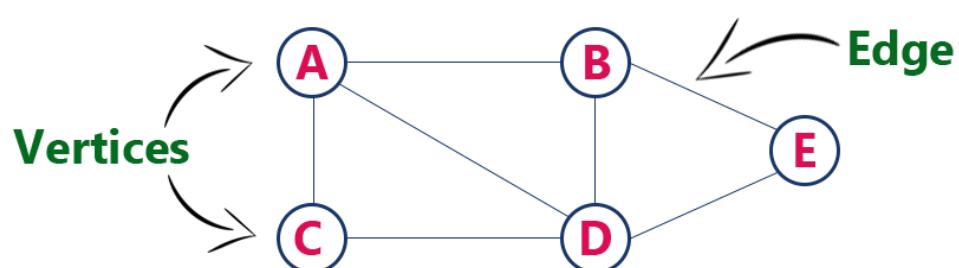
(hubungan *one to many*) antara elemen-elemen, dimana node-node saling terhubung satu sama lain dalam suatu kesatuan yang membentuk layaknya struktur sebuah pohon yang terdiri dari satu simpul khusus yang disebut akar (*root*) sedangkan sisanya membentuk bagian-bagian yang saling tak berhubungan satu sama lain (*subtree*).



Gambar Ilustrasi Struktur Data *Tree*

### 2.2.2 Graf/ Graph

Graf adalah kumpulan node (simpul) di dalam bidang dua dimensi yang dihubungkan dengan sekumpulan garis (sisi). Graf dapat digunakan untuk merepresentasikan obyek-obyek diskrit dan hubungan antara obyek-obyek tersebut. Representasi visual dari graf adalah dengan menyatakan obyek sebagai node, bulatan atau titik (*vertex*), sedangkan hubungan antara obyek dinyatakan dengan garis (*edge*).



Gambar Ilustrasi Graf

Setelah konsep data dan struktur data diuraikan di atas, kita akan dapat menentukan tipe data dan struktur dapat yang tepat yang sesuai dengan konteks permasalahan yang akan dikerjakan dengan mempertimbangkan sifat-sifat dari setiap data dan struktur data.

2. Kelebihan dan kekurangan alternatif struktur data untuk konteks permasalahan yang diselesaikan.

Setelah penjabaran panjang di atas, berikut dapat disimpulkan kelebihan/ keunggulan dan kekurangan/ kelemahan struktur data yang umum digunakan.

Pengetahuan yang tepat tentang kelebihan dan kekurangan setiap tipe akan membantu kita memilih tipe data yang tepat ketika membuat program, sehingga program dan algoritma selain dapat berjalan juga efisien dalam penggunaan sumber daya yang ada, seperti penggunaan memori yang nantinya akan menentukan kecepatan piranti lunak yang dihasilkan.

<b>Struktur Data</b>	<b>Kelebihan</b>	<b>Kekurangan</b>
Larik/ Array	<ul style="list-style-type: none"><li>• data disusun secara beruntun</li><li>• sangat cocok untuk pengaksesan acak</li><li>• terdapat <i>array</i> dengan dimensi lebih dari 1 yang berguna dalam operasi matriks</li><li>• jika berada di suatu lokasi elemen, maka sangat mudah menelusuri ke elemen-elemen tetangga, baik elemen pendahulu atau elemen penerus</li><li>• jika elemen-elemen array adalah nilai-nilai independen dan seluruhnya terjaga, maka</li></ul>	<ul style="list-style-type: none"><li>• besar penyimpanan <i>fixed-size</i>, sehingga akan sulit diubah ukurannya di waktu eksekusi</li><li>• hanya dapat menyimpan tipe data homogen</li><li>• tidak efisien dalam penggunaan memori</li><li>• membutuhkan banyak waktu komputasi</li><li>• karena sifatnya yang statis, maka pada beberapa operasi tidak dapat diaplikasikan</li></ul>

	penggunaan memorinya akan sangat efisien	
Struktur Data	Kelebihan	Kekurangan
<i>Struct/ Record</i>	<ul style="list-style-type: none"> <li>• dapat diakses secara individu</li> <li>• dapat menampung data heterogen</li> </ul>	<ul style="list-style-type: none"> <li>• besar penyimpanan <i>fixed-size</i></li> </ul>
<i>Tumpukan/ Stack</i>	<ul style="list-style-type: none"> <li>• ukurannya dinamis</li> <li>• data tersusun secara sekuensial</li> </ul>	<ul style="list-style-type: none"> <li>• data hanya dapat diakses secara LIFO/ FILO</li> </ul>
<i>Antrian/ Queue</i>	<ul style="list-style-type: none"> <li>• ukurannya dinamis</li> <li>• data tersusun secara sekuensial</li> </ul>	<ul style="list-style-type: none"> <li>• penambahan dan penghapusan hanya dapat dilakukan dari 1 sisi</li> <li>• data hanya dapat diakses secara FIFO</li> </ul>
<i>List/ Link List</i>	<ul style="list-style-type: none"> <li>• ukurannya dinamis</li> <li>• data tersusun secara sekuensial</li> <li>• memungkinkan penambahan, pengurangan, dan pencarian atas elemen data yang tersimpan dalam daftar dilakukan secara lebih efektif</li> </ul>	<ul style="list-style-type: none"> <li>• pemakaian memori lebih besar</li> <li>• pengaksesan tidak dapat dilakukan secara random</li> </ul>
<i>Pohon biner/ Tree</i>	<ul style="list-style-type: none"> <li>• ukurannya dinamis</li> <li>• data tersusun secara sekuensial</li> <li>• berguna untuk menyimpan data dengan hubungan hirarki</li> </ul>	<ul style="list-style-type: none"> <li>• penghapusan sangat rumit</li> </ul>
<i>Graf/ graph</i>	<ul style="list-style-type: none"> <li>• dapat merepresentasikan obyek-obyek diskrit</li> </ul>	<ul style="list-style-type: none"> <li>• beberapa algoritma akan sangat lambat dan rumit</li> </ul>

## B. Keterampilan yang Diperlukan dalam Mengidentifikasi Konsep Data dan Struktur Data

1. Mengidentifikasi konsep data dan struktur data sesuai dengan konteks permasalahan.
2. Membandingkan kelebihan dan kekurangan alternatif struktur data.

### C. Sikap kerja

Harus bersikap secara:

1. Analitis dan teliti dalam mengidentifikasi konsep data dan struktur data sesuai dengan konteks permasalahan.
2. Analitis dan teliti dalam membandingkan kelebihan dan kekurangan alternatif struktur data.

## **BAB III**

### **MENERAPKAN STRUKTUR DATA DAN AKSES TERHADAP STRUKTUR DATA TERSEBUT**

#### **A. Pengetahuan yang Diperlukan dalam Menerapkan Struktur Data dan Akses Terhadap Struktur Data Tersebut**

1. Pengimplementasian struktur data sesuai dengan bahasa pemrograman yang akan dipergunakan.

Dalam modul ini akan diberikan referensi berupa pengimplementasian struktur data dengan menggunakan bahasa pemrograman C untuk mempermudah pemahaman.

##### **1.1 *Array***

tipe\_data nama\_var\_array [ukuran];  
tipe\_data : menyatakan jenis tipe data elemen *array*  
(int, char, float, dll)  
nama\_var\_array : menyatakan nama variabel yang dipakai  
ukuran : menunjukkan jumlah maksimal elemen *array*

Contoh :

```
int nilai[6];
int nilai[6] = {8,7,5,6,4,3};
```

atau dapat disederhanakan menjadi :

```
int nilai[] = {8,7,5,6,4,3};
```

Keterangan berdasarkan contoh diatas berarti kita memblok tempat di memori komputer sebanyak 6 tempat dengan indeks dari 0-5, dimana indeks ke-0 bernilai 8, ke-1 bernilai 7, dst, dan semua elemennya bertipe data *integer*.

Pendeklarasian array 2 dimensi:

```
tipe_data nama_var_array [batas_baris][batas_kolom];
```

### 1.2 *Record/ Struct*

Pendeklarasian struktur:

```
struct nama_struktur
{
    type1 element1;
    type2 element2;
    .
    .
};

struct tipe_struct nama_variabel;
```

atau

```
typedef struct
{
    type1 element1;
    type2 element2;
    type3 element3;
    .
    .
} nama_struct;
tipe_struct nama_variable;
```

### 1.3 Tumpukan/ *Stack*

Proses yang dapat dilakukan pada *stack* adalah :

- PUSH: untuk memasukkan data ke dalam Stack

Langkah yang diperlukan

- cek apakah Top < N
- bila ya, tambahkan top dengan 1
- isikan data ke stack

- POP: mengeluarkan (delete) data dari Stack

Langkah yang diperlukan :

- cek apakah Top masih > 0
- bila ya, copy data ke suatu variabel

- kurangkan Top dengan 1

#### 1.4 Antrian/ Queue

Prinsip:

FIFO (First In First Out)

atau

FCFS (First Come First Serve)

Ada 2 macam pointer, yaitu: F(Front) dan R(Rear)

- Untuk pengambilan data menggunakan pointer F sedang untuk pemasukkan data menggunakan pointer R
- Bila kondisi kosong  $F=0$  dan  $R=0$  sedang kondisi penuh  $R=N$  maka syarat antrian adalah  $F \leq R$

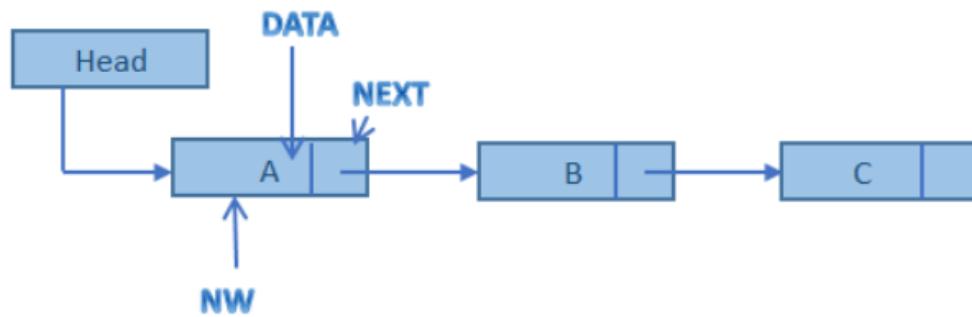
Proses yang dapat dilakukan adalah:

- a. INSERT, untuk memasukkan ke antrian;
- b. DELETE, untuk mengeluarkan data dari antrian.

#### 1.5 List/ Linked List

List atau bisa juga dikenal dengan sebutan Linked List dapat diimplementasikan sebagai berikut:

```
struct node
{
    int data;
    struct node *next;
};
```



Gambar Ilustrasi Linked List

### 1.6 Tree (pohon biner)

B-tree adalah generalisasi dari *binary search tree* dimana node dapat memiliki lebih dari dua turunan (*children*). Syarat-syarat B-tree dengan urutan  $m$  adalah sebagai berikut:

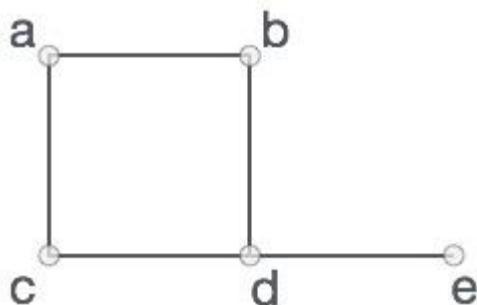
- a. Root memiliki paling sedikit 2 turunan (Child)
- b. Setiap node sebagian besar memiliki  $m$  Children
- c. Setiap node yang bukan Leaf dan Root memiliki setidaknya  $m/2$  Child
- d. Setiap node yang bukan Leaf dengan  $k$  children terdiri dari  $k-1$  Key
- e. Semua Leaf muncul di level yang sama dan membawa informasi

#### Operasi pada B-tree

1. B-TREE CREATE
2. B-TREE INSERT
3. B-TREE DELETE
4. B-TREE SEARCH

### 1.7 Graf/ Graph

Secara umum, graf adalah pasangan  $(V,E)$  dimana  $V$  adalah Vertices/ Vertex dan  $E$  adalah Edge yang menghubungkan dua vertices. Walau kita tidak akan membahas terlalu dalam perihal implementasi graf dalam modul ini, namun berikut akan dijelaskan secara singkat pengenalan dasar graf.



Gambar Ilustrasi Graf 1

Pada graf di atas,

$$V = \{a, b, c, d, e\}$$

$$E = \{ab, ac, bd, cd, de\}$$

Berikut beberapa istilah umum dalam graf yang perlu kita ketahui:

1. Vertex

Setiap node dalam graf adalah berupa Vertex. Pada ilustrasi di bawah, lingkaran A hingga G adalah Vertex, dimana A memiliki indeks 0, B berindeks 1, dan seterusnya.

2. Edge

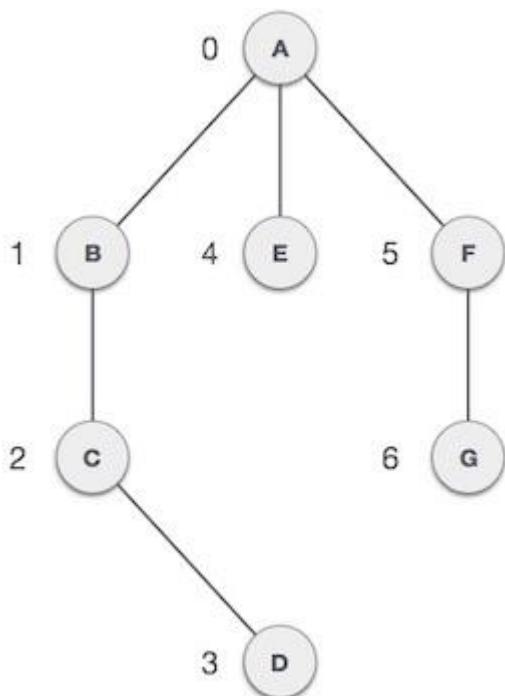
Edge mewakili lintasan antara dua Vertex, dalam ilustrasi di atas, garis yang menghubungkan A dan B, B dan C, dan seterusnya adalah Edge.

3. Adjacency

Dua node atau Vertex dikatakan berdampingan/ Adjacent bila mereka saling dihubungkan oleh Edge. Pada contoh ilustrasi di atas, B dikatakan Adjacent terhadap A, C Adjacent terhadap B, dan seterusnya.

4. Path

Path mewakili urutan dari Edge antara dua Vertex. Pada contoh di atas ini, ABCD adalah Path A ke D.



Gambar Ilustrasi Graf 2

Berikut adalah operasi dasar yang utama pada graf antara lain:

1. Add Vertex

Menambahkan Vertex pada graf.

2. Add Edge

Penambahan Edge antara dua Vertex pada graf.

3. Display Vertex

Menampilkan Vertex graf.

2. Akses terhadap data dalam algoritma yang efisien.

Secara umum, operasi dasar yang biasanya dilakukan dalam struktur data adalah:

Insertion	Menambahkan (menyisipkan) data elemen baru
Deletion	Penghapusan data elemen
Searching	Pencarian elemen dalam struktur data
Traversal	Pemrosesan data elemen
Sorting	Pengurutan data elemen dengan urutan tertentu
Merging	Menggabungkan elemen dari data struktur sejenis menjadi struktur baru dengan jenis yang sama

## 2.1 *Array*

Untuk mengakses elemen array dapat dilakukan dengan cara:

nama\_var\_array [indeks];

Berikut contoh penerapan Array dengan bahasa pemrograman C/C++ dimana terdapat pendeklarasian nilai Array dan pengaksesannya. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

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

void main (){
    //variabel y dideklarasikan sebagai array
    int n, y [] = {9, 8, 12, 6, 7};

    //array
    // [0]=>9, [1]=>8, [2]=>12, [3]=>6, [4]=>7,
    //tampilkan array
    cout<<"Data array: ";
    for(n=0;n<5;n++){
        cout<<y[n]<<" ";
    }
    cout<<endl;

    cout<<"Masukan indeks array yang ingin ditampilkan (1-5): ";
    cin>>n;

    //tampilkan nilai array dengan indeks ke-n
    cout<<"Nilai array ke-<<n<<" : "<<y[n-1];
    getch();
}
```

*Output* program:

```
Data array: 9 8 12 6 7
Masukan indeks array yang ingin ditampilkan (1-5): 5
Nilai array ke-5 : 7
```

## 2.2 Struktur (*structure/ struct*)/ *record*

Pengaksesan *Struct* dapat dilakukan dengan cara berikut:

```
nama_var_struct.nama_var_elemen;
```

Berikut contoh implementasi penggunaan *Struct* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

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

main()
{
    cout<<"Menghitung upah perhari"<<endl;
    cout<<"(1 jam kerja normal dihargai 1000 per jam"<<endl;
    cout<<"lebih dari 8jam dianggap lembur"<<endl;
    cout<<"dan dihargai 2000 per jam)"<<endl<<endl;

    struct jam
    {
        int jam,lembur,nilai;
    }jm;

    cout<<"masukkan jam kerja : ";
    cin>>jm.jam;
    if(jm.jam<=8)
        jm.nilai= jm.jam*1000;
    else if(jm.jam>8){
        jm.lembur=jm.jam-8;
        jm.nilai= (8*1000)+(jm.lembur*2000);
    }
    cout<<"Upah = "<<jm.nilai;
    getch();
}
```

*Output* program:

```
Menghitung upah perhari
(1 jam kerja normal dihargai 1000 per jam
lebih dari 8jam dianggap lembur dan dihargai 2000 per jam)

masukkan jam kerja : 10
Upah = 12000
```

## 2.3 Tumpukan/ Stack

Berikut contoh penerapan *Stack* dengan bahasa pemrograman C/C++ dimana terdapat pendeklarasian nilai *Stack* dan pengaksesannya berupa fungsi menambah data *Stack* (PUSH) dan menghapus data *Stack* (POP). Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
#define size 5

int stack[size];
int top = 0;

void push(){
    int n;
    printf("\n\nMasukan data stack: ");
    scanf("%d",&n);
    if (top==size){
        printf("\n\nStack Penuh ");
    }else{
        stack[top] = n;
        top = top+1;
    }
}

void pop(){
    int item;
    if (top==0){
        printf("\n\nStack Kosong");
    }else{
        item=stack[top-1];
        printf("\n\nData yang dihapus adalah %d", item);
        top--;
    }
}

void display(){
    int i;
    printf("\n\nData stack: ");
    if (top > 0){
        for(i=top-1; i>=0; i--)
            printf("\n %d", stack[i]);
    }else{
        printf("Kosong");
    }
}
```

```
void main(){
    char ch,chl;
    ch ='y';
    chl='y';
    //top=0;
    clrscr();
    while(ch!=='n'){
        push();
        if(top == size){
            printf("\n\nStack penuh");
            break;
        }else{
            printf("\n\nIngin menambah data ke stack? (y/n) ");
            ch=getch();
        }
    }
    display();
    while(chl!=='n'){
        if(top == 0){
            break;
        }else{
            printf("\n\nIngin menghapus data stack? (y/n) ");
            chl=getch();
            if(chl == 'y'){
                pop();
            }
        }
    }
    display();
    getch();
}
```

*Output program:*

```
Masukan data stack: 1

Ingin menambah data ke stack? (y/n)

Masukan data stack: 2

Ingin menambah data ke stack? (y/n)

Masukan data stack: 3

Ingin menambah data ke stack? (y/n)

Masukan data stack: 4
```

```
Ingin menambah data ke stack? (y/n)  
Masukan data stack: 5  
  
Stack penuh  
  
Data stack:  
5  
4  
3  
2  
1
```

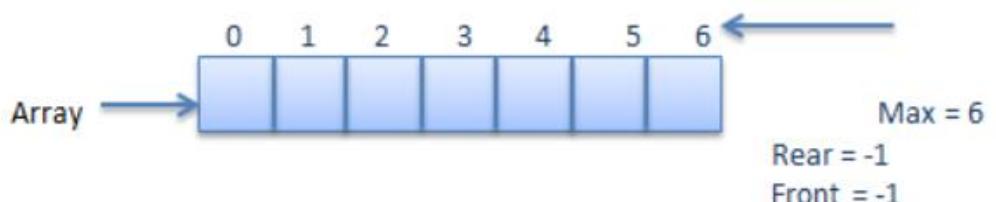
```
Ingin menghapus data stack? (y/n)  
Data yang dihapus adalah 5  
  
Ingin menghapus data stack? (y/n)  
Data yang dihapus adalah 4  
  
Ingin menghapus data stack? (y/n)  
  
Data stack:  
3  
2  
1
```

## 2.4 Antrian/ Queue

Operasi yang dapat dilakukan di struktur Queue:

### 2.4.1 initialize()

Inisialisasi Queue dengan menambahkan nilai -1 Rear dan Front



Gambar Ilustrasi operasi initialize()

### 2.4.2 enqueue()

Penambahan/ insert elemen ke Rear

#### 2.4.3 dequeue()

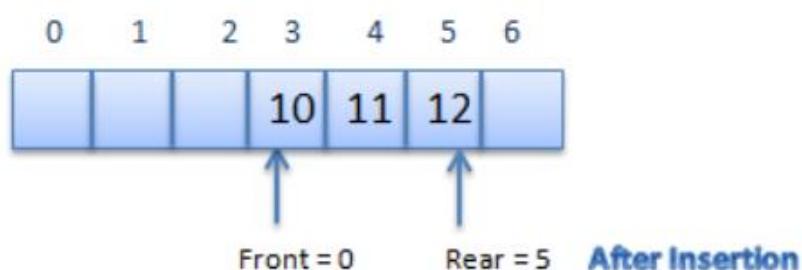
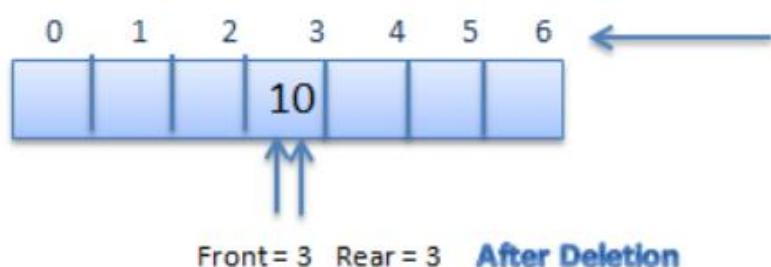
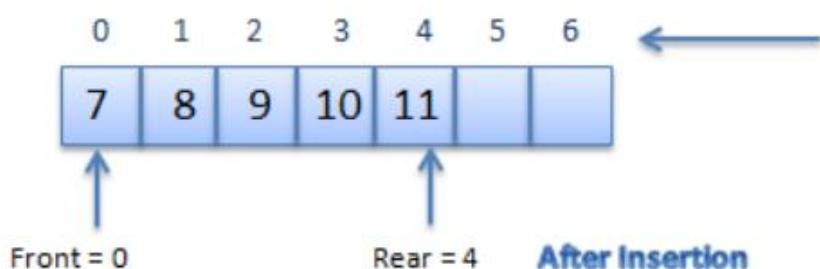
Penghapusan elemen depan dan mengembalikan sama

#### 2.4.4 empty()

Operasi ini akan mengembalikan nilai TRUE(1) jika Queue kosong dan mengembalikan nilai FALSE(0) jika Queue tidak kosong.

#### 2.4.5 full()

Operasi ini akan mengembalikan nilai TRUE(1) jika Queue penuh dan mengembalikan nilai FALSE(0) jika Queue tidak penuh.



Gambar Ilustrasi Operasi pada Queue

Implementasi pada Queue:

1. Antrian statis (static queue) menggunakan Array
2. Antrian dinamis (dynamic queue) menggunakan Linked List

Struktur *Queue* statis

```
#define max 5
struct queue
{
    int data[max];
    int front, rear ;
}
```

Berikut contoh implementasi *Queue dengan Array* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```
#include<stdio>
#include<conio>
#include<stdlib>
#define SIZE 5

int front=-1;
int rear=-1;
int q[SIZE];

void insert();
void del();
void display();

void main(){
    int choice;
    clrscr();
    do{
        clrscr();
        printf("\t Menu");
        printf("\n 1. Tambah Data");
        printf("\n 2. Hapus Data");
        printf("\n 3. Tampilkan Data");
        printf("\n 4. Keluar");
        printf("\n Masukan pilihan Anda: ");

        scanf("%d", &choice);
        switch(choice){
            case 1:
                insert();
                display();
                getch();
                break;
            case 2:
                del();
                display();

                break;
            case 3:
                display();
                getch();
                break;
            case 4:
                printf("Keluar....!!!!");
                getch();
                exit(0);
        }
    }
    while(choice!=4);
}
```

```
void insert(){
    int no;
    printf("\nMasukan Angka: ");
    scanf("%d",&no);
    if(rear < SIZE-1){
        q[++rear] = no;
        if(front == -1)
            front=0;
        // front=front+1;
    }else{
        printf("\nQueue Overflow");
    }
}

void del(){
    if(front == -1){
        printf("\nQueue Underflow");
        return;
    }else{
        printf("\nDeleted Item:-->%d\n",q[front]);
    }

    if(front==rear){
        front=-1;
        rear=-1;
    }else{
        front=front+1;
    }
}

void display(){
    int i;
    if(front == -1){
        printf("\nQueue Kosong....");
        return;
    }
    for(i=front;i<=rear;i++)
        printf("\t%d",q[i]);
}
```

*Output* program:

```
        Menu
1. Tambah Data
2. Hapus Data
3. Tampilkan Data
4. Keluar
Masukan pilihan Anda: 3
      1       2       3       4       5
```

```
        Menu
1. Tambah Data
2. Hapus Data
3. Tampilkan Data
4. Keluar
Masukan pilihan Anda: 2

Deleted Item:-->1
      2       3       4       5
```

Berikut adalah pendeklarasian struktur *Queue* yang dinamis:

```
struct link
{
    int info;
    struct link *next;
}*front,*rear;
```

Berikut contoh implementasinya dalam bahasa pemrograman C/C++ dimana terdapat akses data berupa *Insert* data dan *Delete* data, serta menampilkan data tersebut. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```
/*Implementasi Queue dengan Linked List */
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.h>

struct link{
    int info;
    struct link *next;
}*front,*rear;

void insert_q(int no);
int delete_q();
void display();

void main(){
    int ch,no;
    rear=NULL;
    front=NULL;
    clrscr();

    while(1){
        printf("\nQueue Dinamis");
        printf("\n1->Tambah Data");
        printf("\n2->Hapus Data");
        printf("\n3->Tampilkan Data");
        printf("\n4->Keluar");
        printf("\nMasukan pilihan : ");
        scanf("%d",&ch);
        switch(ch){

            case 1:
            {
                printf("Masukan Data Queue\n");
                scanf("%d",&no);
                insert_q(no);
                printf("\nQueue Setelah Penambahan Data");
                display();
                break;
            }
            case 2:
            {
                no=delete_q();
                printf("\nElemen yang dihapus = %d",no);
                printf("\nQueue Setelah Penghapusan Data");
                display();
                break;
            }
            case 3:
            {
                printf("\nQueue : ");
                display();
                break;
            }
        }
    }
}
```

```

        case 4:
    {
        printf("Keluar....!!!!");
        getch();
        exit(0);
        break;
    }
    default :
        printf("\nAnda Memasukan Pilihan yang Salah...!!!! " );
    }
}

void insert_q(int no){
    struct link *newl;
    newl=(struct link*)malloc(sizeof(struct link));
    newl->info=no;
    newl->next=NULL;
    if(rear==NULL||front==NULL){
        front=newl;
    }else{
        rear->next=newl;
    }
    rear=newl;
}

int delete_q(){
    struct link *t;
    int no;
    if(front==NULL||rear==NULL){
        printf("\nQueue Kosong!");
        getch();
        return 0;
    }else{
        t=front;
        no=t->info;
        front=front->next;
        free(t);
        return 0;
    }
}

void display(){
    struct link *t;
    t=front;
    if(front==NULL||rear==NULL){
        printf("\nQueue Kosong");
        getch();
        exit(0);
    }
    while(t!=NULL){
        printf("\n %d",t->info);
        t=t->next;
    }
}

```

```
Queue Dinamis
1->Tambah Data
2->Hapus Data
3->Tampilkan Data
4->Keluar
Masukan pilihan : 1
Masukan Data Queue
1

Queue Setelah Penambahan Data
1
Queue Dinamis
1->Tambah Data
2->Hapus Data
3->Tampilkan Data
4->Keluar
Masukan pilihan : 1
Masukan Data Queue
2

Queue Setelah Penambahan Data
1
2
Queue Dinamis
1->Tambah Data
2->Hapus Data
3->Tampilkan Data
4->Keluar
Masukan pilihan : 1
Masukan Data Queue
3

Queue Setelah Penambahan Data
1
2
3
```

```
Queue Dinamis
1->Tambah Data
2->Hapus Data
3->Tampilkan Data
4->Keluar
Masukan pilihan : 2

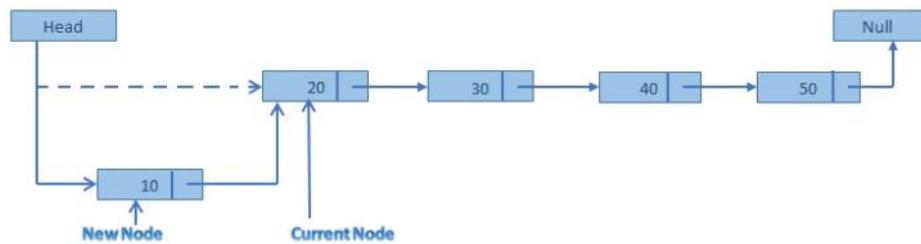
Elemen yang dihapus = 0
Queue Setelah Penghapusan Data
2
3
Queue Dinamis
1->Tambah Data
2->Hapus Data
3->Tampilkan Data
4->Keluar
Masukan pilihan :
```

## 2.5 List/ Linked list

### 1.5.1 Insertion

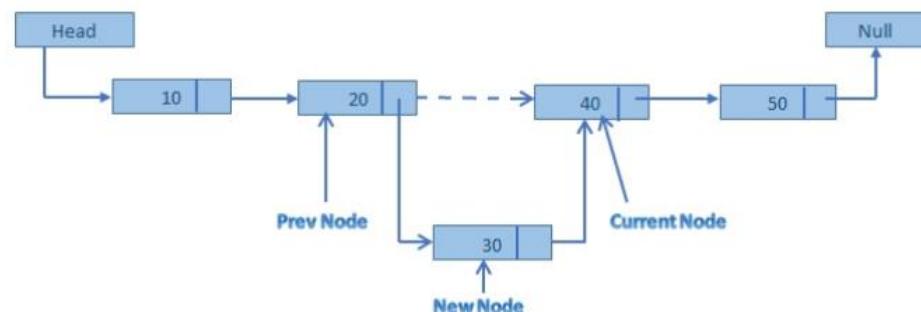
Ada 3 macamsituasi insertion/ penambahan/ penyisipan di list, yaitu:

- di depan/ awal list



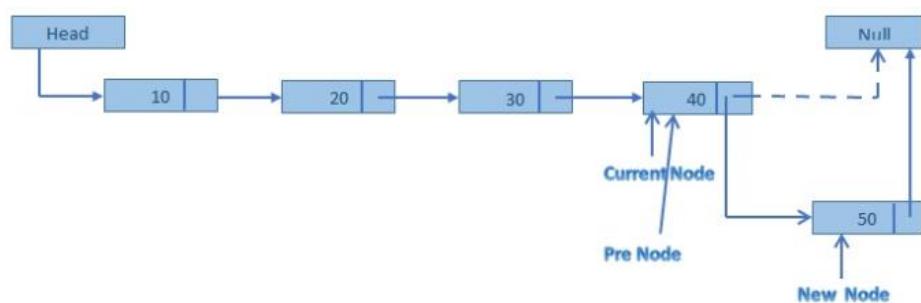
Gambar Ilustrasi Insert di awal List

- di tengah list



Gambar Ilustrasi Insert di tengah List/ di posisi tertentu

- di belakang/ akhir list



Gambar Ilustrasi Insert di akhir List

Prosedur Insert:

1. Ambil nilai yang ingin ditambahkan/ di-Insert dan posisinya
2. Buat node baru dengan malloc()
3. Insert data dalam data filed node
4. Tambahkan node baru ini ke posisi yang diinginkan dengan menunjukkan lokasi/ indeks di List
5. Ulangi langkah nomor 1 hingga mendapatkan nilai yang cukup untuk ditambahkan ke List

Menambahkan node

```
void insert(node *ptr, int data)
{
    /* Mencari node terakhir*/
    while(ptr->next!=NULL)
    {
        ptr = ptr -> next;
    }

    /* Alokasi memori untuk node baru dan deklarasi nilai pada node tersebut*/

    ptr->next = (node *)malloc(sizeof(node));
    ptr = ptr->next;
    ptr->data = data;
    ptr->next = NULL;
}
```

Berikut contoh penerapan *Insert* di lokasi tertentu pada *List Linier* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```

#include<stdio.h>
#include<malloc.h>
#include<conio.h>

typedef struct Node{
    int info ;
    struct Node *next;
} node;

void createsig(node**,int);
void insertAtloc(node **,int,int,int);
void display(node *);

void main(){
    int ch, item, pos,loc,i;
    node *start ;
    start = NULL;
    clrscr();
    printf("Masukan Jumlah Node: ");
    scanf("%d",&i);
    createsig(&start,i);
    printf("\nList : ");
    display(start);
    printf("\nMasukan Urutan Lokasi Penyisipan : ");
    scanf("%d",&loc);
    printf("\n\nMasukan Angka yang ingin Disisipkan : ");
    scanf("%d",&item);
    insertAtloc(&start,item,loc,i);
    printf("\nList Baru Setelah Penyisipan : ");
    display(start);
    getch();
}

void createsig(node **start,int i){
    int item ,k=1;
    while(i){
        node *ptr,*last;
        printf("\nMasukan Angka untuk Node ke- %d : ",k);
        scanf("%d",&item);
        ptr=(node*)malloc(sizeof(node));
        ptr->info=item;
        ptr->next=NULL;
        if (*start==NULL){
            *start = ptr ;
        }else{
            last = *start;
            while(last->next != NULL){
                last=last->next;
            }
            last->next = ptr ;
        }
        i--;
        k++;
    }
}

```

```
void insertAtloc(node **start,int item , int i,int k ){
    node *ptr,*loc,*last;
    int n=1 ;
    i=i-1;
    ptr=(node*)malloc(sizeof(node));
    ptr->info=item;
    loc = *start ;
    if (*start==NULL){
        ptr->next = NULL ;
        *start = ptr ;
    }else if(i<=k){
        while(n != i){
            loc=loc->next;
            n++;
        }
        ptr->next = loc->next ;
        loc->next = ptr ;
    }else{
        last = *start;
        while(last->next != NULL){
            last=last->next;
        }
        last->next = ptr ;
    }
}

void display(node *start){
    while(start !=NULL){
        printf("\t %d",start->info);
        start = start->next;
    }
}
```

*Output program:*

```
Masukan Jumlah Node: 3
Masukan Angka untuk Node ke- 1 : 1
Masukan Angka untuk Node ke- 2 : 2
Masukan Angka untuk Node ke- 3 : 3
List : 1 2 3
Masukan Urutan Lokasi Penyisipan : 2

Masukan Angka yang ingin Disisipkan : 9
List Baru Setelah Penyisipan : 1 9 2 3
```

### 2.5.2 Deletion

Operasi penghapusan juga sama dengan Insertion, bisa terjadi di awal, tengah, dan akhir List.

Algoritma Delete (di awal List)

DELETE AT BEG(INFO,NEXT,START)

1. IF(START=NULL)

2. ASSIGN PTR = START

3. ASSIGN TEMP = INFO[PTR]

4. ASSIGN START = NEXT[PTR]

5. FREE(PTR)

6. RETURN(TEMP)

Berikut fungsi delete di awal List

```
void deleteatbeg(node **start)
{
    node *ptr;
    int temp;
    ptr = *start ;
    temp = ptr->info;
    *start = ptr->next ;
    free(ptr);
    printf("\nDeleted item is %d : \n",temp);
}
```

Berikut contoh penerapannya dalam bahasa pemrograman C/C++.

Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```

#include<stdio.h>
#include<conio.h>
#include<malloc.h>

typedef struct nodetype{
    int info;
    struct nodetype *next;
} node ;

void createlist(node **,int );
void deleteatbeg(node **);
void display(node *);

void main(){
    node *start;
    int item,n,i;
    start=NULL;
    clrscr();
    printf("\nMasukan Jumlah Node : ");
    scanf("%d",&n);
    for(i= 0 ;i< n ;i++){
        printf("\nMasukan Angka untuk Node ke- %d : ",i+1);
        scanf("%d",&item);
        createlist(&start,item) ;
    }
    printf("\nData :\n");
    display(start);
    printf("\n \nTekan key Apapun untuk Menghapus Node Pertama");
    getch();
    deleteatbeg(&start);
    printf("\nData Setelah Penghapusan :\n");
    display(start);
    getch();
}

void createlist(node **start,int item){
    node *ptr,*last;
    ptr =(node*)malloc(sizeof(node));
    ptr->info = item ;
    ptr->next = NULL;
    if(*start == NULL)
        *start = ptr ;
    else{
        last = *start ;
        while(last->next !=NULL){
            last = last->next;
        }
        last->next = ptr ;
    }
}

```

```
void deleteatbeg(node **start){  
    node *ptr;  
    int temp;  
    ptr = *start ;  
    temp = ptr->info;  
    *start = ptr->next ;  
    free(ptr);  
    printf("\n \nNode yang Dihapus : %d\n",temp);  
}  
  
void display(node *start){  
    int n = 0;  
    while(start !=NULL){  
        printf("\t %d",start->info);  
        n++;  
        start = start->next;  
    }  
    printf("\nJumlah Node : %d",n);  
}
```

*Output program:*

```
Masukan Jumlah Node : 5  
  
Masukan Angka untuk Node ke- 1 : 1  
  
Masukan Angka untuk Node ke- 2 : 2  
  
Masukan Angka untuk Node ke- 3 : 3  
  
Masukan Angka untuk Node ke- 4 : 4  
  
Masukan Angka untuk Node ke- 5 : 5  
  
Data :  
      1      2      3      4      5  
Jumlah Node : 5  
  
Tekan key Apapun untuk Menghapus Node Pertama  
  
Node yang Dihapus : 1  
  
Data Setelah Penghapusan :  
      2      3      4      5  
Jumlah Node : 4
```

### 2.5.3 Search

Pencarian sekuensial adalah paling umum digunakan dalam struktur List.

#### Algoritma Search

1. Inisialisasi pointer dengan awal List
2. Bandingkan nilai kunci (KEY) dengan nilai node saat ini
  - Bila sama (TRUE) maka selesai
  - Bila tidak (FALSE) maka lanjutkan langkah nomor 3 berikut
3. Pindahkan pointer ke poin node selanjutnya di List dan lakukan langkah nomor 2 hingga selesai

Berikut contoh penerapan fungsi pencarian (*Search*) dalam *List* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```

#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node{
    int data;
    struct node *next;
}
first, *nw;
int search(int);

void main(){
    int no,i,item,pos;
    clrscr();
    first.next=NULL;
    nw=&first;
    printf("Masukan Jumlah Node yang Ingin Dibuat Linked List: ");
    scanf("%d",&no);
    printf("\n");
    for(i=0;i< no;i++){
        nw->next=(struct node *)malloc(sizeof(struct node));
        printf("Masukan Angka Untuk Node ke- %d: ",i+1);
        scanf("%d",&nw->data);
        nw=nw->next;
    }
    nw->next=NULL;
    printf("\nLinked List :\n");
    nw=&first;
    while(nw->next!=NULL){
        printf("%d\t",nw->data);
        nw=nw->next;
    }
    printf("\n");
    printf("\nMasukan Angka yang Ingin Dicari: ");
    scanf("%d",&item);
    pos=search(item);

    if(pos<=no)
        printf("\nAngka Yang Dicari ada di Node ke- %d",pos);
    else
        printf("\nMaaf, Angka yang Dicari Tidak Ditemukan.");
    getch();
}

int search(int item){
    int count=1;
    nw=&first;
    while(nw->next!=NULL){
        if(nw->data==item)
            break;
        else
            count++;
        nw=nw->next;
    }
    return count;
}

```

*Output program:*

```
Masukan Jumlah Node yang Ingin Dibuat Linked List: 5
Masukan Angka Untuk Node ke- 1: 99
Masukan Angka Untuk Node ke- 2: 88
Masukan Angka Untuk Node ke- 3: 77
Masukan Angka Untuk Node ke- 4: 33
Masukan Angka Untuk Node ke- 5: 22

Linked List :
99      88      77      33      22

Masukan Angka yang Ingin Dicari: 77
Angka Yang Dicari ada di Node ke- 3
```

#### 2.5.4 Sort

Berikut contoh penerapan pengurutan data dalam *List* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node{
    int data;
    struct node *next;
};

void main()
{
    int i;
    int num ;
    struct node *first, *nw, *pre, *newl, *count;
    clrscr();
    printf("\nMasukan jumlah node yang ingin dibuat: ");
    scanf("%d", &num );
    first->next = NULL;
    nw = first;

    for (i = 0; i < num ; i++)
    {
        nw->next = (struct node* ) malloc(sizeof(struct node));
        nw = nw->next;
        printf("\nMasukan node ke- %d: ", i+1);
        scanf("%d", &nw->data);
        nw->next = NULL;
    }

    newl = first;
    for( ; newl->next != NULL; newl = newl->next)
    {
        for(count = newl->next; count != NULL; count = count->next)
        {
            if(newl->data > count->data)
            {
                int temp = newl->data;
                newl->data = count->data;
                count->data = temp;
            }
        }
    }
    nw = first->next;
    printf("\nData Setelah Pengurutan (Sorting):\n");
    while (nw)
    {
        printf("%d\t", nw->data);
        nw = nw->next;
    }
    getch();
}
```

*Output program:*

```
Masukan jumlah node yang ingin dibuat: 5
Masukan node ke- 1: 8
Masukan node ke- 2: 4
Masukan node ke- 3: 2
Masukan node ke- 4: 6
Masukan node ke- 5: 1
Data Setelah Pengurutan (Sorting):
1      2      4      6      8
```

## 2.6 Tree (pohon biner)

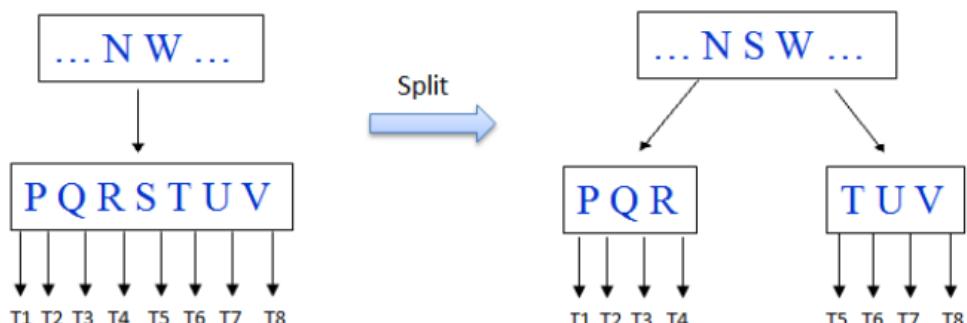
### 2.6.1 B-TREE CREATE

Untuk membuat Tree yang tidak kosong, maka peratam-tama harus membuat Tree yang kosong, lalu Insert node, berikut algoritmanya:

```
B-TREE-CREATE(T)
x ← ALLOCATE-NODE()
leaf[x] ← TRUE
n[x] ← 0
DISK-WRITE(x)
root[T] ← x
```

### 2.6.2 B-TREE INSERT

Splitting adalah tahapan fundamental dalam insert.



Gambar Ilustrasi Splitting

### Algoritma Insertion

```
B-TREE-INSERT(T, k)
r ← root[T]
if n[r] = 2t - 1
then s ← ALLOCATE-NODE()
root[T] ← s
leaf[s] ← FALSE
n[s] ← 0
c1[s] ← r
B-TREE-SPLIT-CHILD(s, 1, r)
B-TREE-INSERT-NONFULL(s, k)
else B-TREE-INSERT-NONFULL(r, k)
```

```
B-TREE-INSERT-NONFULL(x, k)
i ← n[x]
if leaf[x]
then while i ≥ 1 and k < keyi[x]
do keyi+1[x] ← keyi[x]
i ← i - 1
keyi+1[x] ← k
n[x] ← n[x] + 1
DISK-WRITE(x)
else while i ≥ 1 and k < keyi[x]
do i ← i - 1
i ← i + 1
DISK-READ(ci[x])
if n[ci[x]] = 2t - 1
then B-TREE-SPLIT-CHILD(x, i, ci[x])
if k > keyi[x]
then i ← i + 1
B-TREE-INSERT-NONFULL(ci[x], k)
```

Berikut contoh ilustrasi insertion

key = 1,12,8,2,25,6,14,28,17,7,52,16,48,68,3,26,29,53,55,45,67.

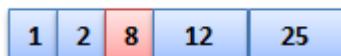
Order = 5

Prosedur untuk menambah key ke B-tree:

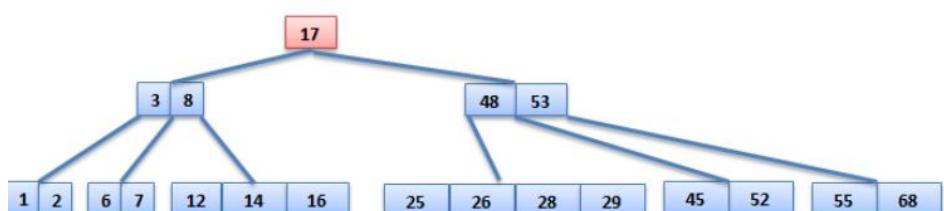
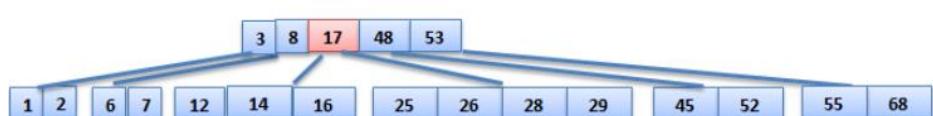
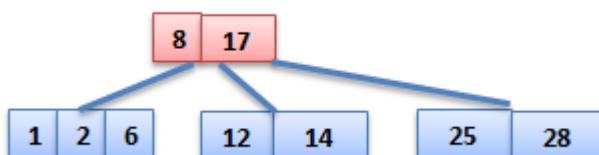
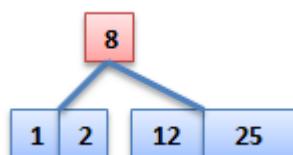
1. Tambah Key pertama sebagai node Root



2. Tambah Key berikutnya secara berurutan hingga node penuh

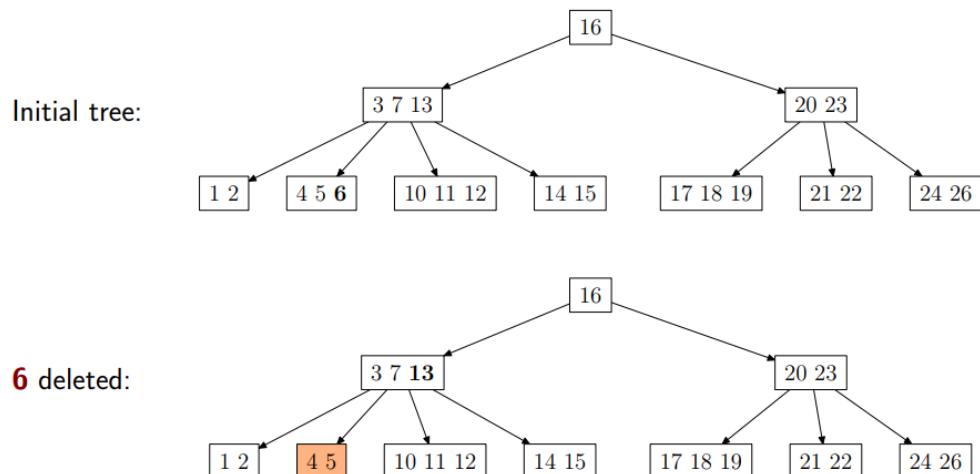


3. Bila node sudah penuh dilakukan operasi Split



### 2.6.3 B-TREE DELETE

Dalam penghapusan data pada B-Tree terdapat beberapa kasus, namun dalam modul ini hanya akan dibahas bila penghapusan data adalah data pada Leaf.



Gambar Ilustrasi B-TREE DELETE

### 2.6.4 B-TREE SEARCH

Berikut algoritma pencarian pada B-Tree

(dengan  $x$  adalah pointer node root dan  $k$  adalah nilai yang ingin dicari pada tree tersebut)

```

function B-Tree-Search(x, k) returns (y, i) such that
keyi [y] = k or nil
i < 1
while i ≤ n[x] and k > keyi [x]
do i ← i + 1
if i ≤ n[x] and k = keyi [x]
    then return (x, i)
if leaf[x]
    then return nil
else Disk-Read(ci [x])
    return B-Tree-Search(ci [x], k)

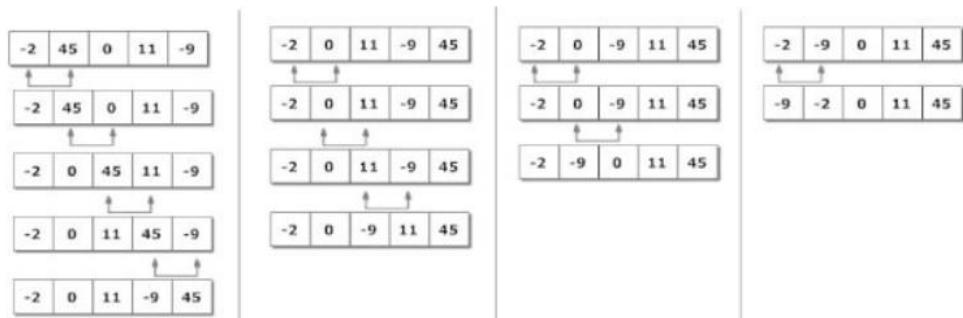
```

## 2.7 Operasi Sort

Operasi pengurutan data adalah salah satu operasi yang cukup penting dalam struktur data/ pemrograman. Berikut beberapa metode *sorting* yang umum digunakan:

### 2.7.1 Bubble Sort

Bubble sort atau disebut juga *sinking sort* adalah jenis algoritma pengurutan data sederhana yang bekerja dengan melakukan langkah-langkah pengurutan secara berulang-ulang, membandingkan setiap pasangan data yang saling berdampingan dan menukar tempat mereka bila urutannya salah.



Gambar Ilustrasi Langkah-langkah Proses Bubble Sort

Berikut contoh penerapan *bubble sort* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diujii menggunakan aplikasi Borland C/C++ Versi 5.02.

```
#include <constrea.h>

int main(){
    int data[10];
    int i, j, k, tmp, jumlah=0;
    cout<<"PROGRAM PENGURUTAN BILANGAN DENGAN BUBBLE SORT\n\n";
    cout<<"Masukkan jumlah bilangan : "; cin>>k;
    for(i=0; i<k; i++){
        cout<<"Masukkan Angka ke "<<(i+1)<<" : ";
        cin>>data[i];
        if(data[i]%2==0){jumlah+=data[i];}
    }
    cout<<"\nData sebelum diurutkan: "<<endl;
    for(i=0; i<k; i++){
        cout<<data[i]<<" ";
    }
    cout<<endl;

    for( i=0;i<k;i++){
        for(j=i+1;j<k;j++){
            if(data[i]>data[j]){
                tmp=data[i];
                data[i]=data[j];
                data[j]=tmp;
            }
        }
    }
    cout<<"\nData setelah diurutkan: "<<endl;
    for(i=0; i<k; i++){
        cout<<data[i]<<" ";
    }
    getch();
}
```

*Output program:*

```
PROGRAM PENGURUTAN BILANGAN DENGAN BUBBLE SORT

Masukkan jumlah bilangan : 5
Masukkan Angka ke 1 : 7
Masukkan Angka ke 2 : 6
Masukkan Angka ke 3 : 2
Masukkan Angka ke 4 : 9
Masukkan Angka ke 5 : 1

Data sebelum diurutkan:
7 6 2 9 1

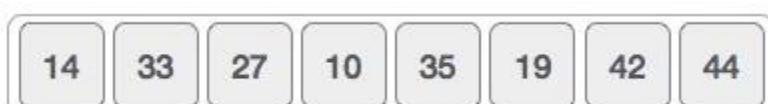
Data setelah diurutkan:
1 2 6 7 9
```

### 2.7.2 *Insertion Sort*

*Insertion sort* adalah algoritma sederhana yang membangun *Array* final terurut satu-persatu. Dimana untuk *List* besar akan lebih tidak efisien dibandingkan algoritma pengurutan lain seperti *Quick* atau *Merge Sort*.

Untuk lebih mudah memahami, berikut ilustrasi tahapan/ cara kerja *Insertion Sort*:

Tahap 1



Tahap 2



Tahap 3



Tahap 4



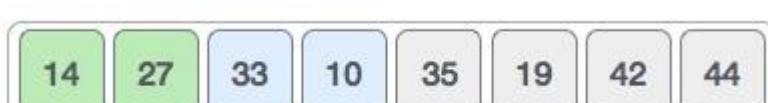
Tahap 5



Tahap 6



Tahap 7



Tahap 8



Tahap 9



Tahap 10



Tahap 11



Tahap 12



Tahap 13



Lalu akan mengulang tahapan 1-13 hingga seluruh *List* urut.

Algoritma *Insertion Sort*

```
INSERTION-SORT(A)
for j = 2 to A.length
key = A[j]
// Insert A[j] into the sorted sequence A[1...j-1]
i = j - 1
while i > 0 and A[i] > key
A[i+1] = A[i]
i = i - 1
A[i+1] = key
```

Berikut contoh penerapan *Insertion Sort* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```
#include<stdio.h>
#include<conio.h>
void main(){
    int A[20],n,Temp,i,j;
    clrscr();
    printf("\n\t\t-----INSERTION SORT-----\n\n");
    printf("\nMasukan jumlah data: ");
    scanf("%d",&n);
    printf("\nMasukan Angka:\n");
    for(i=0; i < n;i++){
        scanf("%d", &A[i]);
    }
    for(i=1; i< n; i++){
        Temp = A[i];
        j = i-1;
        while(Temp < A[j] && j>=0){
            A[j+1] = A[j];
            j = j-1;
        }
        A[j+1] = Temp;
    }
    printf("\n\t\t-----DATA HASIL INSERTION SORT-----\n");
    printf("Data Dengan Urutan Ascending: ");
    for(i=0; i < n; i++){
        printf("\t%d", A[i]);
    }
    getch();
}
```

*Output* program:

```
-----INSERTION SORT-----

Masukan jumlah data: 5
Masukan Angka:
7
3
4
9
1
-----DATA HASIL INSERTION SORT-----

Data Dengan Urutan Ascending: 1 3 4 7 9
```

### 2.7.3 *Quick Sort*

*Quick sort* adalah algoritma pengurutan sederhana menggunakan prosedur *divide-and-conquer* yang berulang. Algoritma ini adalah yang paling cepat dibandingkan algoritma *comparison-based* lainnya, juga dikenal sebagai *Partition-exchange Sort*.

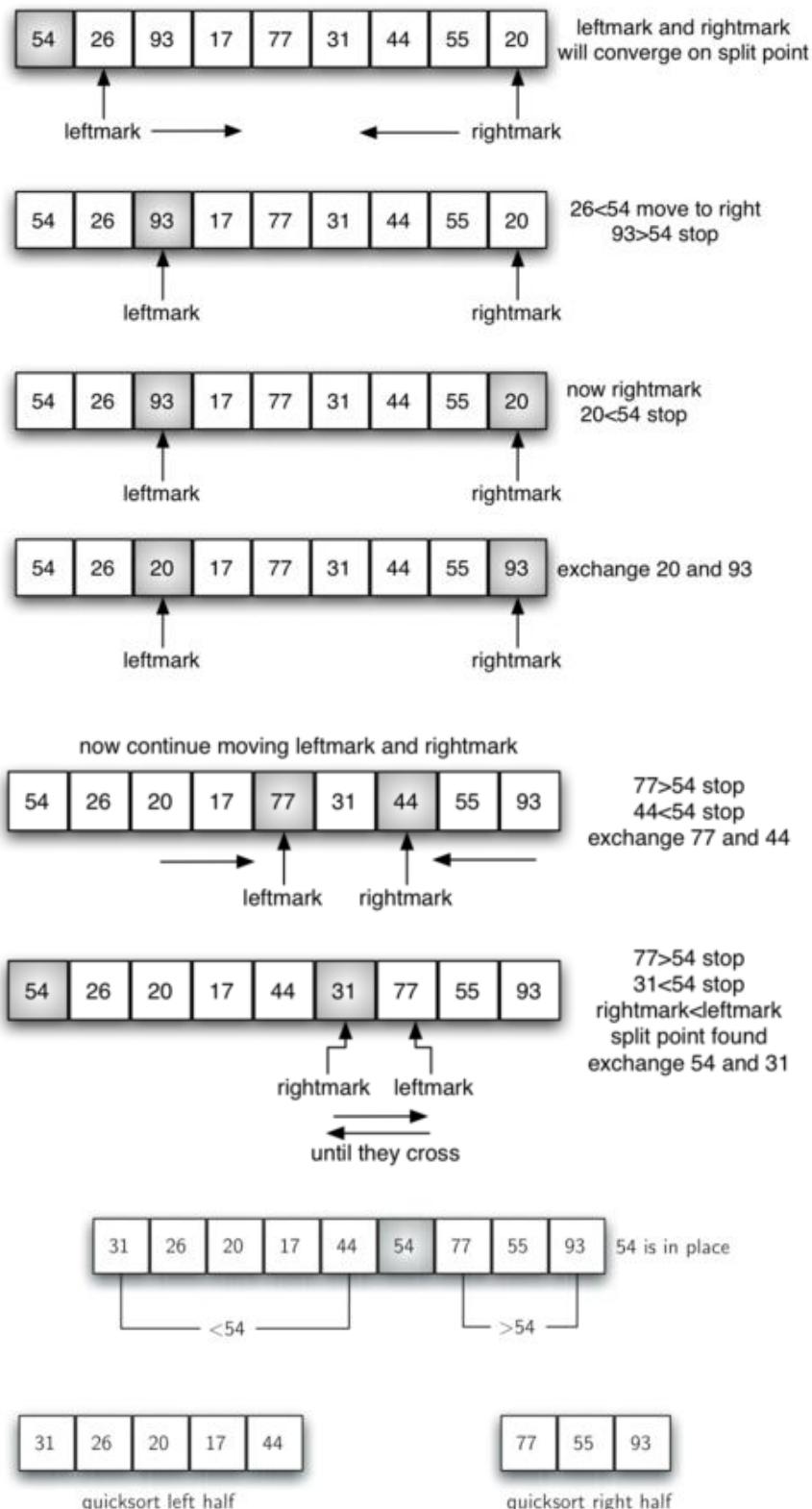
Algoritma

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q - 1)
    QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] <= x
        i = i + 1
    exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Langkah-langkah *Quick Sort* dapat diilustrasikan sebagai berikut:





Gambar Ilustrasi Langkah-langkah Quick Sort

Berikut contoh penerapan *Quick Sort* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

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

void quicksort(int arr[], int lb, int ub);

void main(){
    int arr[20], n, i;
    clrscr();
    printf("\n\t\t-----QUICK SORT-----\n\n");
    printf("Masukan jumlah data:");
    scanf("%d",&n);
    printf("Masukan data yang ingin diurutkan:\n");
    for(i=0;i < n;i++)
        scanf("%d",&arr[i]);
    quicksort(arr, 0, n-1);
    printf("\n\t\t-----DATA SETELAH QUICK SORT-----\n\n");
    printf("Data yang sudah diurutkan:");
    for(i = 0; i < n; i++)

        printf("\t%d ",arr[i]);
    getch();
}

void quicksort(int arr[], int lb, int ub){
    int pivot, i, j, temp;
    if(lb < ub){
        pivot = lb;
        i = lb;
        j = ub;
        while(i < j){
            while(arr[i] <= arr[pivot] && i <= ub)
                i++;
            while(arr[j] > arr[pivot] && j >= lb)
                j--;
            if(i < j){
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        temp = arr[j];
        arr[j] = arr[pivot];
        arr[pivot] = temp;
        quicksort(arr, lb, j-1);
        quicksort(arr, j+1, ub);
    }
}
```

*Output program:*

```
-----QUICK SORT-----  
Masukan jumlah data:3  
Masukan data yang ingin diurutkan:  
6  
2  
9  
-----DATA SETELAH QUICK SORT-----  
Data yang sudah diurutkan: 2 6 9
```

#### 2.7.4 Merge Sort

*Merge Sort* merupakan algoritma pengurutan *comparison-based*.

Berikut dijelaskan tahapan *Merge Sort* dengan ilustrasi contoh pada Array yang belum urut sebagai berikut:

Tahap 1



Tahap 2



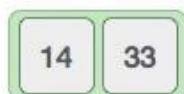
Tahap 3



Tahap 4



Tahap 5



Tahap 6



Tahap 7



Berikut contoh penerapan *Merge Sort* dengan bahasa pemrograman C/C++. Kode program berikut dibuat, dijalankan, dan diuji menggunakan aplikasi Borland C/C++ Versi 5.02.

```
#include <stdio.h>
#include <constrea.h>
#define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
int b[10];

void merging(int low, int mid, int high) {
    int l1, l2, i;
    for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
        if(a[l1] <= a[l2])
            b[i] = a[l1++];
        else
            b[i] = a[l2++];
    }
    while(l1 <= mid)
        b[i++] = a[l1++];

    while(l2 <= high)
        b[i++] = a[l2++];

    for(i = low; i <= high; i++)
        a[i] = b[i];
}

void sort(int low, int high){
    int mid;
    if(low < high){
        mid = (low + high) / 2;
        sort(low, mid);
        sort(mid+1, high);
        merging(low, mid, high);
    }else{
        return;
    }
}

int main(){
    int i;
    printf("Data Sebelum Diurutkan\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);

    sort(0, max);
    printf("\nData Setelah Diurutkan\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);

    getch();
}
```

*Output program:*

```
Data Sebelum Diurutkan
10 14 19 26 27 31 33 35 42 44 0
Data Setelah Diurutkan
0 10 14 19 26 27 31 33 35 42 44
```

**B. Keterampilan yang Diperlukan dalam Menerapkan Struktur Data dan Akses Terhadap Struktur Data Tersebut**

1. Mengimplementasikan struktur data.
2. Membuat akses terhadap data dalam algoritma yang efisien.

**C. Sikap kerja**

Harus bersikap secara:

1. Analitis dan teliti dalam mengimplementasikan struktur data.
2. Analitis dan teliti dalam membuat akses terhadap data dalam algoritma yang efisien.



## BUKU INFORMASI

# MENGIMPLEMENTASIKAN *USER INTERFACE* J.620100.005.01



KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

Modul Pelatihan Berbasis Kompetensi Bidang Pekerjaan Domestik	Kode Modul J.62010.033.02
<b>DAFTAR ISI</b>	
<b>BAB I PENDAHULUAN.....</b>	
A. Tujuan Umum .....	4
B. Tujuan Khusus .....	4
<b>BAB II MENGIDENTIFIKASI RANCANGAN USER INTERFACE .....</b>	
A. Pengetahuan yang diperlukan dalam mengidentifikasi rancangan user interface. ....	5
1 Rancangan user interface diidentifikasi sesuai kebutuhan. ....	5
2 Komponen user interface dialog diidentifikasi sesuai konteks rancangan proses. ....	16
3 Urutan dari akses komponen user interface dialog dijelaskan. ....	26
4 Simulasi (mock-up) dari aplikasi yang akan dikembangkan dibuat. ....	32
B. Keterampilan yang diperlukan dalam mengidentifikasi rancangan user interface. ....	38
C. Sikap Kerja yang diperlukan dalam mengidentifikasi rancangan user interface. ....	38
<b>BAB III MELAKUKAN IMPLEMENTASI RANCANGAN USER INTERFACE .....</b>	
A. Pengetahuan yang diperlukan dalam melakukan implementasi rancangan user interface. ....	39
1 Menu program sesuai dengan rancangan program diterapkan .....	39
2 Penempatan user interface dialog diatur secara sekuensial. ....	50
B. Keterampilan yang diperlukan dalam melakukan implementasi rancangan user interface. ....	57
1 Mampu menerapkan menu program sesuai rancangan .....	57
2 Mampu menempatkan user interface sekuensial .....	57
C. Sikap Kerja yang diperlukan dalam melakukan implementasi rancangan user interface. ....	57
1 Harus cermat dalam mempersiapkan dokumentasi uji cob melakukan implementasi rancangan user interface. ....	57
2 Harus cermat memilih tools yang digunakan .....	57

Modul Pelatihan Berbasis Kompetensi Bidang Pekerjaan Domestik	Kode Modul J.62010.033.02
------------------------------------------------------------------	------------------------------

3 Harus teliti dan taat asas .....	57
DAFTAR PUSTAKA.....	58
DAFTAR PERALATAN/MESIN DAN BAHAN.....	59
DAFTAR PENYUSUN MODUL .....	60

Modul Pelatihan Berbasis Kompetensi Bidang Pekerjaan Domestik	Kode Modul J.62010.033.02
<b>BAB I</b>	
<b>PENDAHULUAN</b>	
<p><b>A. Tujuan Umum</b></p> <p>Setelah mempelajari modul ini peserta latih diharapkan mampu menimplementasikan user interface dengan benar.</p> <p><b>B. Tujuan Khusus</b></p> <p>Adapun tujuan mempelajari unit kompetensi melalui buku informasi menimplementasikan user interface ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut: (<i>diambil dari elemen dan KUK</i>)</p> <ol style="list-style-type: none"><li>1. Mengidentifikasi rancangan user interface.</li><li>2. Melakukan implementasi rancangan user interface.</li></ol>	

## BAB II

### BAB II MENGIDENTIFIKASI RANCANGAN USER INTERFACE

#### A. Pengetahuan yang diperlukan dalam mengidentifikasi rancangan user interface.

##### 1 Rancangan user interface diidentifikasi sesuai kebutuhan.

###### a. Konsep User Interface

###### UI dan UX

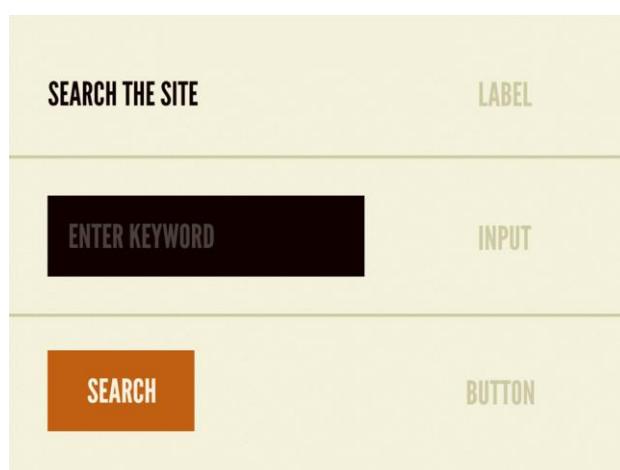
UI adalah metode bagaimana sebuah produk terlihat dan estetis, sedangkan UX adalah bagaimana sebuah produk terasa dan sudah memenuhi kebutuhan pengguna

###### Memahami layout dan prinsip desain atomic (*atomic design principal*)

- Atom: Materi dasar dalam interface.

Jika atom adalah blok bangunan dasar materi, maka atom antarmuka /interface berfungsi sebagai blok bangunan dasar yang terdiri dari semua antarmuka pengguna. Atom-atom ini termasuk elemen HTML dasar seperti label formulir, masukan, tombol, dan lainnya yang tidak dapat dipecah-2 lebih lanjut tanpa menghentikan fungsinya.

Contoh atom: Gambar di bawah ini terdiri dari 3 atom (*label, input, dan button*)

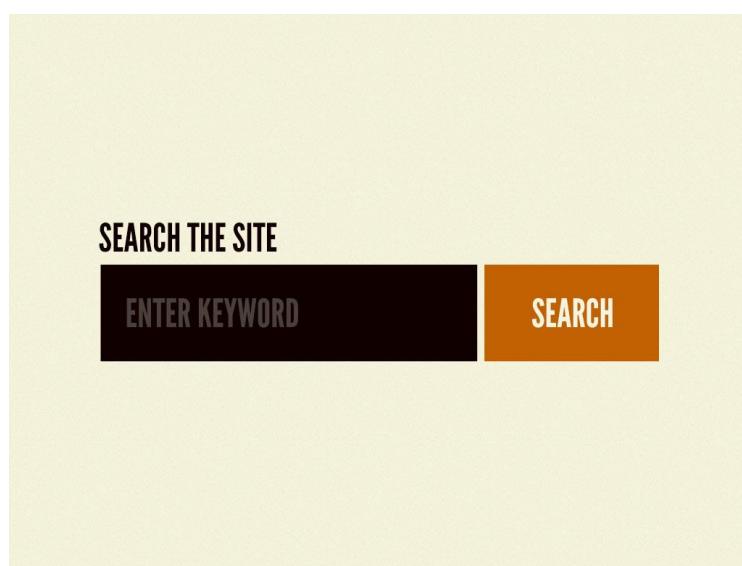


Gambar 1. Atom Label, Input dan Button

- Molekul

Dalam kimia, molekul adalah kelompok atom terikat bersama yang mempunyai sifat baru yang berbeda. Misalnya, molekul air dan molekul hidrogen peroksida memiliki sifat uniknya sendiri dan berperilaku cukup berbeda, meskipun mereka terdiri dari unsur atom yang sama (hidrogen dan oksigen).

Dalam terminologi interface, molekul adalah kelompok elemen UI yang relatif sederhana berfungsi bersama sebagai satu unit. Misalnya, label formulir, masukan pencarian, dan tombol dapat bergabung bersama untuk membuat molekul bentuk pencarian. Contoh molekul pencarian berikut.



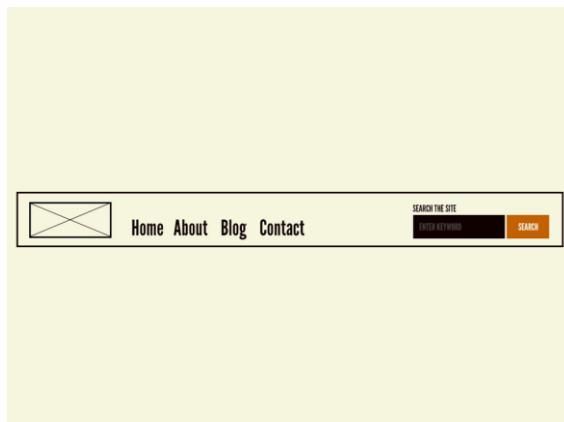
Gambar 2. Molekul

- Organisme

Organisme adalah komponen UI yang relatif kompleks yang terdiri dari kelompok molekul dan / atau atom dan / atau organisme lain. Organisme ini membentuk bagian berbeda dari suatu antarmuka.

Lihat kembali molekul bentuk pencarian di atas. Sebuah formulir pencarian sering dapat ditemukan di header dari banyak web, jadi mari kita masukkan molekul pencarian ke dalam konteks organisme header.

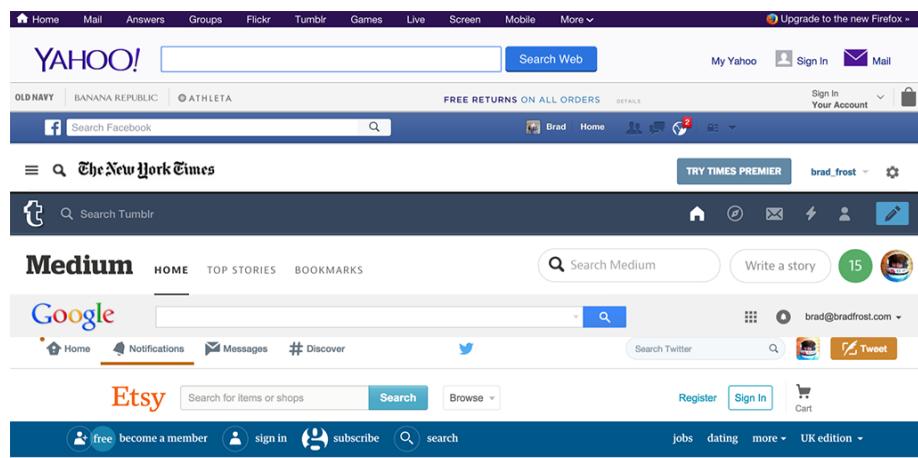
Contoh organisme seperti gambar interface di bawah. Organisme header ini terdiri dari molekul form pencarian, atom logo, dan molekul navigasi utama.



Gambar 3. Organisme Navigasi

Header membentuk bagian yang berdiri sendiri dari sebuah antarmuka, meskipun berisi beberapa bagian antarmuka yang lebih kecil dengan sifat dan fungsi masing-masing yang unik.

Organisme dapat terdiri dari jenis molekul yang sama atau berbeda. Organisme header mungkin terdiri dari elemen yang berbeda seperti gambar logo, daftar navigasi utama, dan formulir pencarian. Kita bisa melihat berbagai jenis organisme ini di hampir setiap situs web yang kami kunjungi. Contoh berikut.



Gambar 4. Contoh berbagai organisme dalam situs web

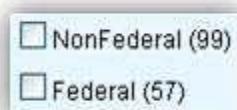
## b. Elemen Antarmuka Pengguna / User Interface

Saat merancang antarmuka, sebaiknya konsisten dan dapat diprediksi dalam pilihan elemen antarmuka. Disadari atau tidak, pengguna telah menjadi akrab dengan unsur-unsur yang berperilaku dengan cara tertentu, sehingga memilih untuk mengadopsi elemen-elemen tersebut dengan sesuai akan membantu penyelesaian tugas, efisiensi, dan kepuasan.

Elemen antarmuka di antaranya termasuk namaun tidak terbatas pada:

- **Kontrol Input:** kotak centang, tombol radio, daftar dropdown, kotak daftar, tombol, matikan, bidang teks, bidang tanggal
- **Komponen Navigasi:** *breadcrumb, slider, kolom pencarian, pagination, slider, tag, ikon*
- **Komponen Informasi:** tooltips, ikon, bilah kemajuan, pemberitahuan, kotak pesan, jendela modal
- **Wadah:** accordion
- **Kontrol Input,** terdiri dari:

*Checkbox (kotak centang).* Kotak centang memungkinkan pengguna untuk memilih satu atau lebih opsi dari satu set/himpunan. Biasanya yang terbaik adalah menyajikan kotak centang dalam daftar vertikal. Lebih dari satu kolom fisibel juga jika daftar cukup panjang sehingga mungkin memerlukan pengguliran atau jika perbandingan persyaratan mungkin diperlukan. Contoh kotak centang:



*Radiobutton (Tombol radio),* Tombol radio digunakan untuk memungkinkan pengguna memilih satu item dalam satu waktu. Contoh tombol radio

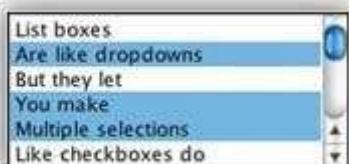


*Dropdown List (Daftar dropdown)*, Daftar dropdown memungkinkan pengguna untuk memilih satu item pada satu waktu, mirip dengan tombol radio, tetapi lebih kompak memungkinkan sehingga kita untuk menghemat ruang. Pertimbangkan untuk menambahkan teks ke bidang, seperti 'Pilih satu' untuk membantu pengguna mengenali tindakan yang diperlukan.

Contoh dropdown

Find your state or... Go

*ListBox (Daftar kotak)*, Daftar kotak, seperti kotak centang, memungkinkan pengguna untuk memilih beberapa item sekaligus, tetapi lebih kompak dan dapat mendukung daftar opsi yang lebih panjang jika diperlukan. Contoh kotak daftar



*Button (Tombol)*, Suatu tombol menunjukkan tindakan saat disentuh dan biasanya diberi label menggunakan teks, ikon, atau keduanya. Contoh tombol

Send Post Tweet

*Dropdown Button (Tombol Dropdown)*, Tombol dropdown terdiri dari tombol yang ketika diklik menampilkan daftar drop-down dari item-item yang saling eksklusif. Contoh tombol tarik-turun



*Toggle Button (Tombol toggle)* memungkinkan pengguna untuk mengubah pengaturan antara dua negara bagian. Mereka paling efektif ketika negara on / off secara visual berbeda. Contoh toggles



*Text field (Bidang teks)* Bidang teks memungkinkan pengguna memasukkan teks. Hal ini dapat memungkinkan satu baris atau beberapa baris teks. Contoh bidang teks

Text input fields let you input text

*Date and time Picker (Pemilih tanggal dan waktu)* Pemilih tanggal memungkinkan pengguna memilih tanggal dan / atau waktu. Dengan menggunakan pemilih, informasi secara konsisten diformat dan dimasukkan ke dalam sistem.



Gambar 5. Pemilih tanggal dan waktu

### C. Komponen Navigasi

*Searching field (Bidang Pencarian)* Kotak pencarian memungkinkan pengguna memasukkan kata kunci atau frasa (permintaan) dan mengirimkannya untuk mencari indeks dengan maksud untuk mendapatkan kembali hasil yang paling relevan. Bidang pencarian biasanya adalah kotak teks satu baris dan sering disertai dengan tombol pencarian. Contoh kotak pencarian dengan berbagai fungsi



Gambar 6. Kotak pencarian dengan berbagai fungsi

*Breadcrumb, Breadcrumbs* memungkinkan pengguna untuk mengidentifikasi lokasi mereka saat ini di dalam sistem dengan memberikan jejak yang dapat diklik pada halaman yang sedang berjalan untuk menavigasi. Contoh

## breadcrumb

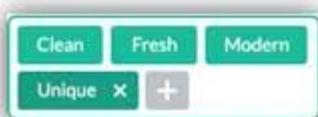
Home > Folder Index Page > Page You're On

*Penanda Halaman (Paginasi/Pagination)*, membagi konten di antara halaman, dan memungkinkan pengguna untuk melewati halaman atau mengirimkan request melalui konten. Contoh paginasi



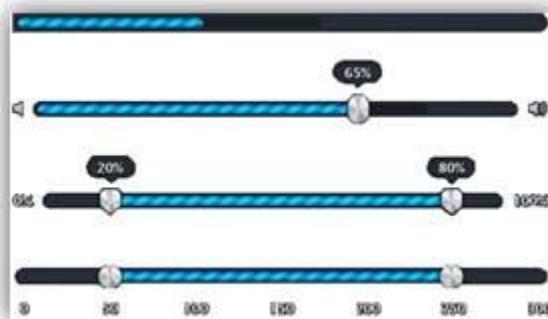
Gambar 7. Paginasi

*Tag (Tags)* memungkinkan pengguna menemukan konten dalam kategori yang sama. Beberapa sistem penandaan juga memungkinkan pengguna untuk menerapkan tag mereka sendiri ke konten dengan memasukkannya ke dalam sistem. Contoh tag



Gambar 8. Tag

*Slider*, slider, juga dikenal sebagai track bar, memungkinkan pengguna untuk mengatur atau menyesuaikan nilai. Ketika pengguna mengubah nilai, itu tidak mengubah format antarmuka atau info lain di layar. Contoh slider



Gambar 9. Slider

*Icon (Ikon)*, Ikon adalah gambar sederhana yang berfungsi sebagai simbol intuitif yang digunakan untuk membantu pengguna menavigasi sistem. Biasanya, ikon di-hyperlink. Contoh ikon



Gambar 10. Contoh berbagai ikon

*Gambar Carousel (Image carousels)* memungkinkan pengguna untuk menelusuri serangkaian item dan membuat pilihan satu jika mereka memilihnya. Biasanya, gambar tersebut hyperlink. Contoh korsel gambar

Komponen Informasi

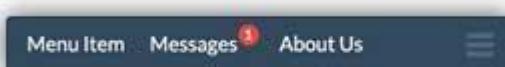


Gambar 11. Gambar Carousel

#### d. Komponen Informasi (Information Component)

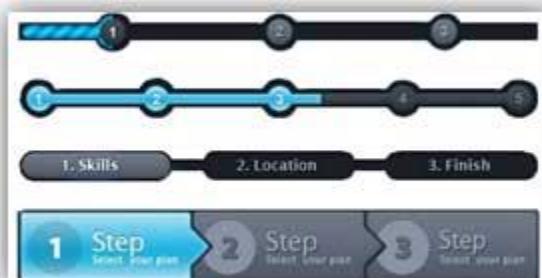
*Pemberitahuan (Notification/Notifikasi),* Pemberitahuan adalah pesan pembaruan yang mengumumkan sesuatu yang baru bagi pengguna untuk dilihat. Notifikasi biasanya digunakan untuk menunjukkan item seperti, penyelesaian tugas yang berhasil, atau pesan kesalahan atau peringatan.

Contoh pemberitahuan



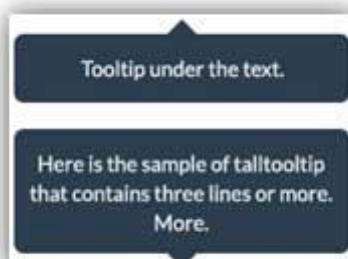
Gambar 12. Komponen informasi

*Bilah Kemajuan (Progress Bars),* Sebuah bilah kemajuan menunjukkan di mana pengguna berada saat mereka maju melalui serangkaian langkah dalam suatu proses. Biasanya, bilah kemajuan tidak dapat diklik. Contoh bilah kemajuan



Gambar 13. Progress bar

*Tips Kakas (Tools Tips)* Sebuah tooltip memungkinkan pengguna untuk melihat petunjuk ketika mereka membawa lebih dari satu item yang menunjukkan nama atau tujuan dari item tersebut. Contoh kakas tip



Gambar 14. Tips tools

Kotak Pesan (Message Box), Kotak pesan adalah jendela kecil yang memberikan informasi kepada pengguna dan mengharuskan mereka untuk mengambil tindakan sebelum dapat bergerak maju. Contoh kotak pesan



Gambar 15. Message box

Modal Window (pop-up), Jendela modal mengharuskan pengguna untuk berinteraksi dengannya dalam beberapa cara sebelum mereka dapat kembali ke sistem. Contoh jendela modal



Gambar 16. Modal Window

### e. Wadah (Container)

Akordeon adalah daftar item yang ditumpuk secara vertikal yang memanfaatkan fungsi pertunjukan / sembunyikan. Ketika label diklik, ia memperluas bagian yang menunjukkan konten di dalamnya. Ada dapat memiliki satu atau lebih item yang ditampilkan sekaligus dan mungkin memiliki status default yang mengungkapkan satu atau beberapa bagian tanpa pengguna mengklik . Contoh akordeon



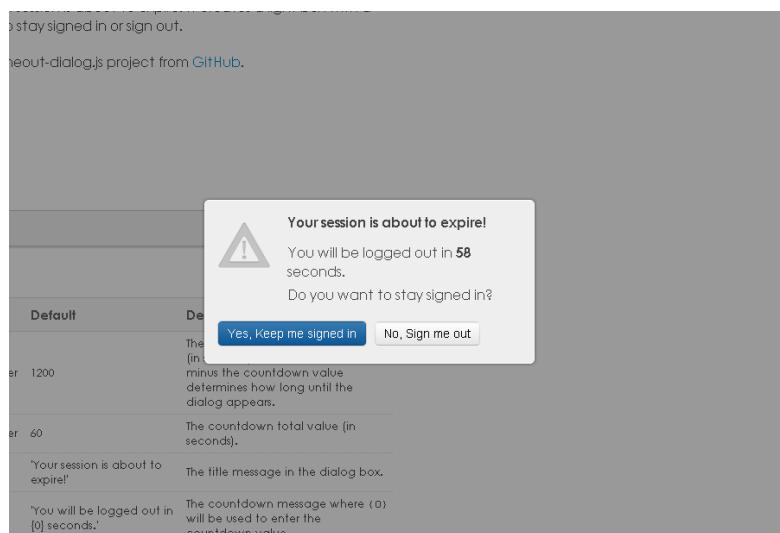
Gambar 17. Container

## 2 Komponen user interface dialog diidentifikasi sesuai konteks rancangan proses.

### Dialog

Dialog adalah overlay yang mengharuskan pengguna untuk berinteraksi dengannya dan dirancang untuk mendapatkan tanggapan dari pengguna.

Dialog menginformasikan pengguna tentang informasi penting, mengharuskan pengguna untuk membuat keputusan, atau melibatkan banyak tugas. Dalam aplikasi, di web dan bahkan pada dialog perangkat mobilie semakin banyak digunakan untuk mengarahkan perhatian pengguna ke tugas tertentu, tanpa keluar dari layar saat kita sedang aktif. Berikut contoh dialog *session expired*



Gambar 18. Kotak dialog

Berikut panduan dalam mengimplementasikan dialog

#### a. Mengurangi Gangguan/Interruption

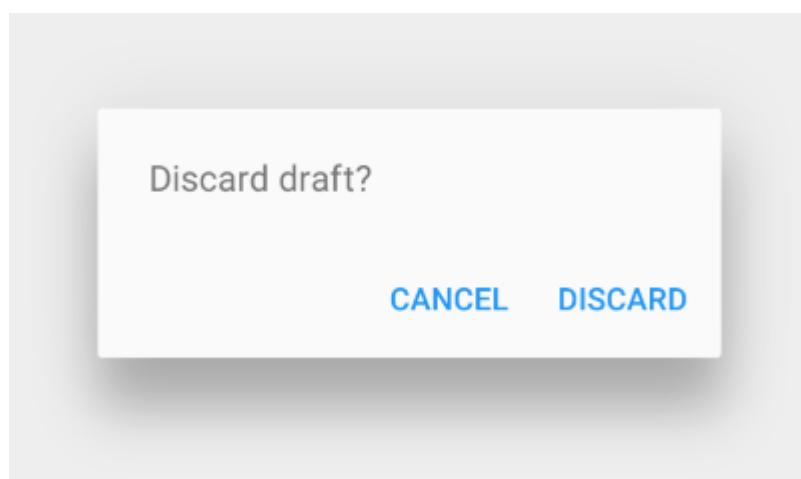
Gunakan dialog dengan hemat karena sehingga tidak terlalu mengganggu.

Penampilan dialog yang tiba-tiba memaksa pengguna untuk menghentikan

tugas mereka saat ini dan fokus pada konten dialog. Pengguna harus berurusan dengan dialog sebelum melanjutkan dan tidak lagi dapat mengakses halaman saat ini. Terkadang ini adalah hal yang baik, seperti ketika pengguna harus mengonfirmasi tindakan penting, tetapi sebagian besar waktunya tidak perlu dan cukup sering sangat mengganggu.

### **Kebutuhan Konfirmasi**

Sangat masuk akal untuk menggunakan dialog dalam situasi di mana kita ingin agar pengguna berinteraksi sebelum melanjutkan aktivitasnya, atau ketika dampak dari kesalahan yang serius.



Gambar 19. Dialog konfirmasi

### **Hindari DenganTiba-tiba Membuka Dialog**

Tiba-tiba membuka dialog tanpa pengguna melakukan sesuatu adalah ide yang kurang baik. Banyak situs membombardir pengunjung dengan berlangganan kotak seperti contoh di bawah ini.



Gambar 20. Dialog tiba-tiba, kurang baik untuk disain UI

Dialog harus selalu terbuka saat pengguna melakukan (atau melakukan) sesuatu. Sesuatu itu mungkin mengklik tombol, mengikuti tautan atau memilih opsi.

### **Ringkasan**

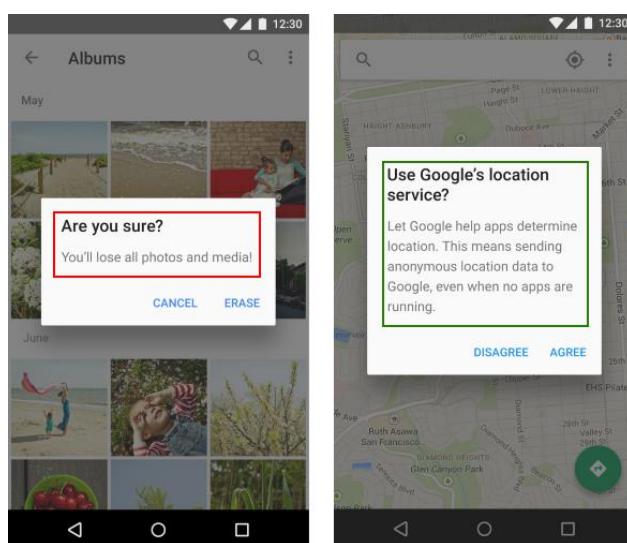
Tidak setiap pilihan, pengaturan, atau detail baik untuk diberikan gangguan berupa dialog box. Alternatif untuk dialog adalah menu atau perluasan sebaris, yang keduanya mempertahankan konteks saat ini. Jangan hanya membuka dialog. Buat tampilan dialog dapat diprediksi untuk pengguna.

#### **b. Sesuaikan arti Dialog dengan Konisi Sebenarnya**

Dialog harus berbicara dengan bahasa pengguna (gunakan kata, frasa dan konsep yang dikenal pengguna), daripada istilah sistem khusus.

### **Hapus Pertanyaan dan Opsi**

Kita harus menggunakan pertanyaan atau pernyataan yang jelas dengan penjelasan di bidang konten, seperti "Hapus penyimpanan kita?" Atau "Hapus akun kita?" Secara umum, kita harus menghindari permintaan maaf, ambiguitas, atau pertanyaan, seperti "Peringatan!" atau "Apakah kamu yakin?"

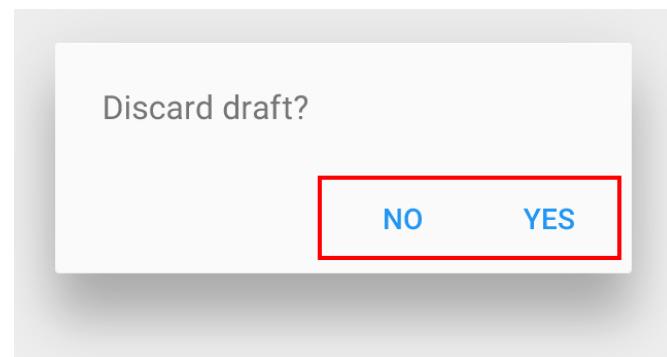


*Dialog kiri menimbulkan pertanyaan ambigu dan ruang lingkup dampaknya tidak jelas. Dialog kanan menimbulkan pertanyaan spesifik, menggambarkan dampak bagi pengguna dan memberikan tindakan yang jelas.*

Hindari menyajikan kepada pengguna berupa opsi ambigu atau tidak jelas.

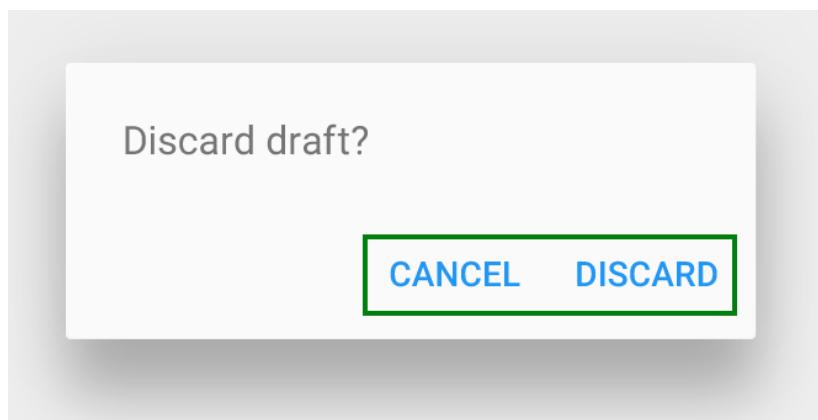
Kita harus menggunakan hanya opsi yang jelas. Dalam banyak kasus, pengguna harus dapat memahami pilihan berdasarkan judul dan teks tombol saja.

Contoh buruk: Teks tindakan menjawab pertanyaan "Tidak", tetapi tidak menyarankan apa yang akan terjadi sesudahnya.



Gambar 21. Dialog konfirmasi yang buruk

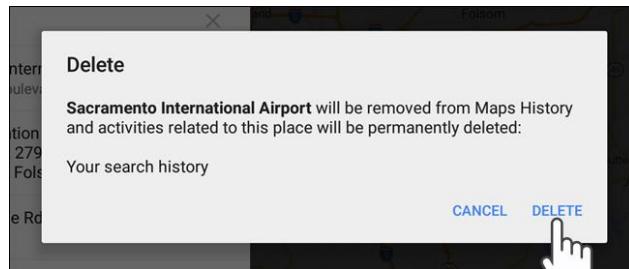
Contoh yang bagus: Teks tindakan afirmatif "Buang"/"Discard" dengan jelas menunjukkan hasil keputusan.



Gambar 22. Dialog konfirmasi yang baik, Dialog sistem di Android. Sumber: Desain Material.

### Berikan Informasi Penting

Penting bahwa dialog tidak mengaburkan informasi yang mungkin berguna bagi pengguna. Misalnya, dialog yang meminta pengguna untuk mengonfirmasi penghapusan beberapa item harus mencantumkan item yang dihapus.

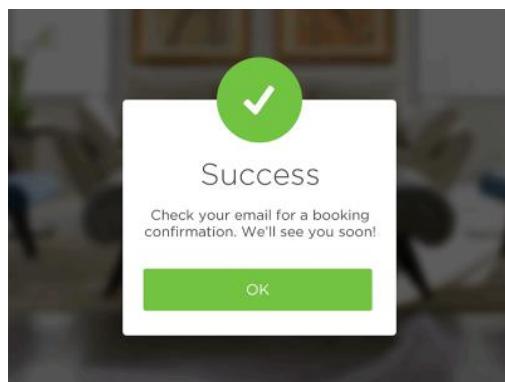


Gambar 23. Dialog ini dengan singkat menguraikan tentang dampaknya.

Hindari menggunakan tindakan "Pelajari lebih lanjut" untuk mengakses dokumentasi bantuan; ekspansi in-line dalam dialog harus digunakan sebagai gantinya. Jika informasi lebih luas diperlukan, sediakan sebelum memasuki dialog.

### Menawarkan Umpan Balik Informatif

Ketika suatu proses selesai, ingatlah untuk menampilkan pesan pemberitahuan (atau umpan balik visual). Beri tahu pengguna bahwa dia telah melakukan semua yang diperlukan.



Gambar 24. Contoh Sukses setelah melakukan tindakan.

### Ringkasan

Gunakan pertanyaan dan opsi yang jelas dalam dialog.

Desain sialog untuk menghasilkan penutupan.

Beri tahu pengguna setelah tindakan.

### c. Upayakan untuk Minimalis

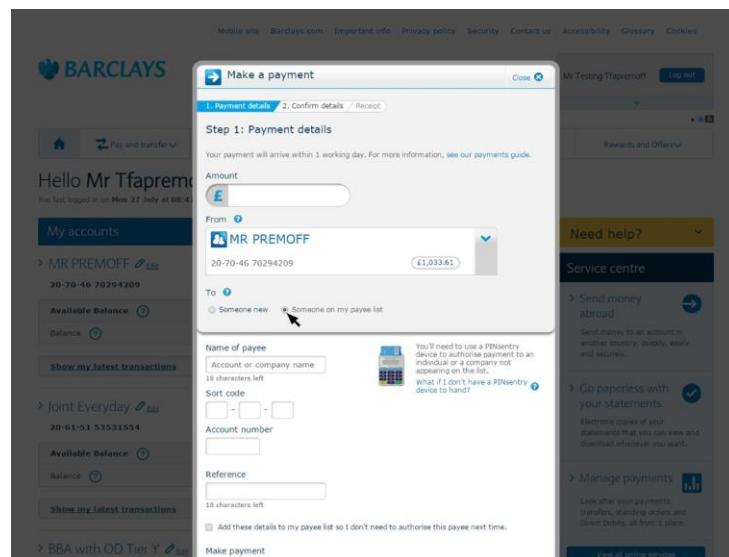
Kita tidak boleh mencoba menjelaskan terlalu banyak komponen ke dalam

dialog. Tetap bersih dan sederhana (ikuti prinsip KISS). Tetapi minimalis tidak berarti terbatas. Semua informasi harus bernilai dan relevan.

### Jumlah Elemen dan Opsi

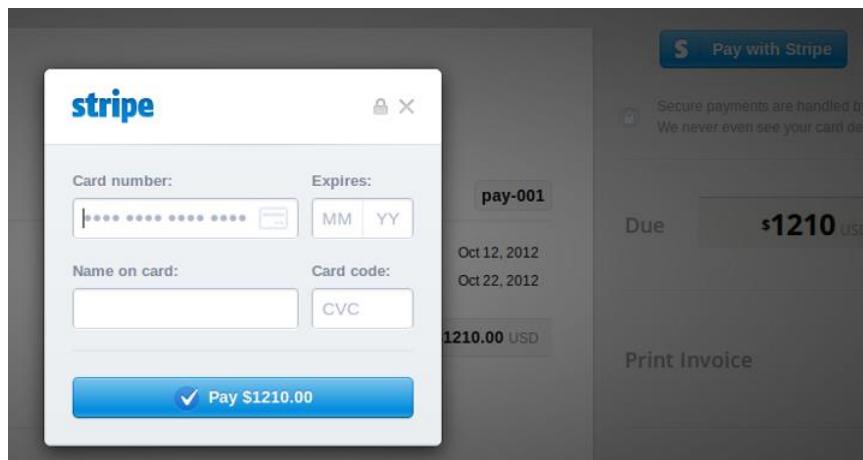
Dialog tidak boleh muncul sebagian di layar. Kita tidak harus menggunakan dialog yang berisi konten gulir.

**Contoh Buruk:** Dialog pemrosesan pembayaran Barclays Bank memiliki banyak opsi dan elemen, bagian dari opsi tersebut hanya tersedia menggunakan pengguliran (terutama untuk perangkat seluler yang biasanya memiliki layar-estat yang relatif kecil).



Gambar 25. Dialog konfirmasi yang buruk,  
Sumber: Barclays

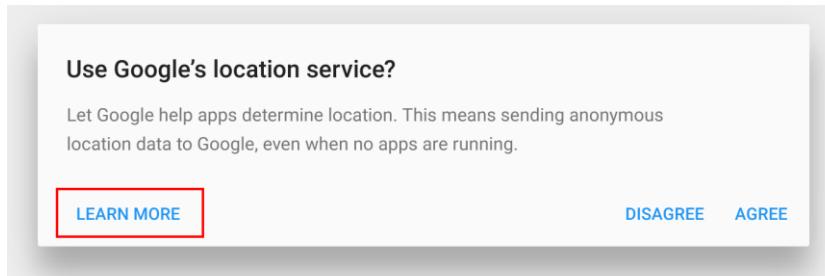
**Contoh Bagus:** Stripe menggunakan dialog yang sederhana dan cerdas dengan hanya informasi penting yang terlihat bagus baik di desktop maupun di layar ponsel.



Gambar 26. Dialog konfirmasi yang baik

### Jumlah Aksi

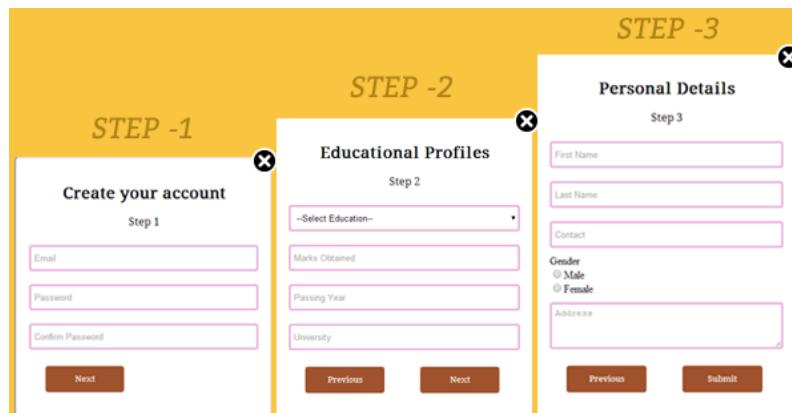
Dialog tidak boleh mencakup lebih dari dua tindakan. Tindakan ketiga, seperti "Pelajari lebih lanjut," menjauhi dialog, berpotensi meninggalkan tugas yang belum selesai.



Gambar 27. Tindakan "Learn more" menjauhkan kita dari dialog ini, membiarkannya dalam keadaan tak tentu.

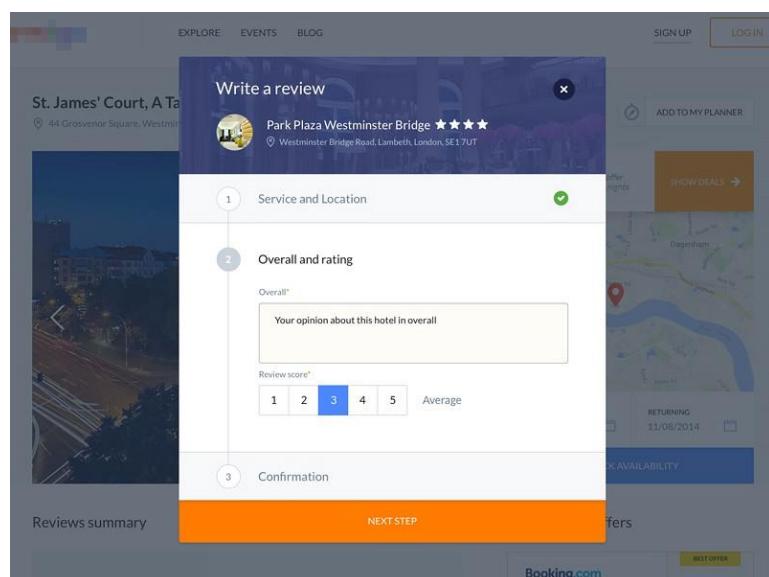
### Jangan Sertakan Beberapa Langkah di Dialog

Memecah tugas yang rumit menjadi beberapa langkah adalah ide yang bagus, tetapi itu juga secara umum merupakan tanda bahwa ada sesuatu yang terlalu rumit untuk meminta pengguna menyelesaikan dalam batasan dialog.



Gambar 28. Dialog yang buruk, menyertakan beberapa tahapan dalam satu dialog

Jika sebuah interaksi cukup kompleks untuk memerlukan beberapa langkah (seperti pada contoh di bawah), maka sebaiknya dijadikan sebagai halaman tersendiri.



Gambar 29. Jendela dialog dengan beberapa langkah. Sumber: Dribbble.

## Ringkasan

- Jika kita menemukan bahwa kita mencoba menjelaskan banyak elemen ke dalam dialog, itu berarti dialog bukanlah solusi desain terbaik.
- Sederhanakan dialog dengan menghapus elemen atau konten yang tidak perlu yang tidak mendukung tugas pengguna.
- Cobalah untuk menghindari dialog dengan beberapa langkah.

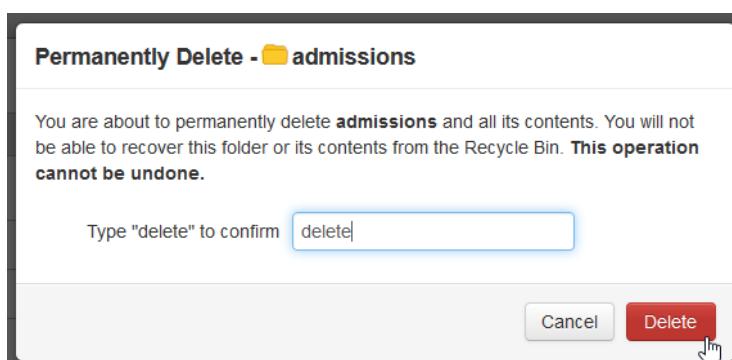
### d. Pilih Jenis Dialog yang Tepat

Dialog diperlukan dalam dua tipe utama. Tipe pertama adalah pop up "pay attention dialog yang memaksa pengguna untuk berinteraksi dengan sistem sebelum melanjutkan. Dialog modal biasanya digunakan untuk proses pemblokiran diskrit, di mana:

- Konteks sekitarnya tidak diperlukan untuk memutuskan tindakan apa yang harus diambil.
- Membutuhkan tindakan 'terima' atau 'batal' untuk menutupnya. Tidak menutup ketika area di luar itu diklik.
- Tidak dapat diterima bagi kemajuan pengguna untuk dibiarkan dalam keadaan setengah jadi.

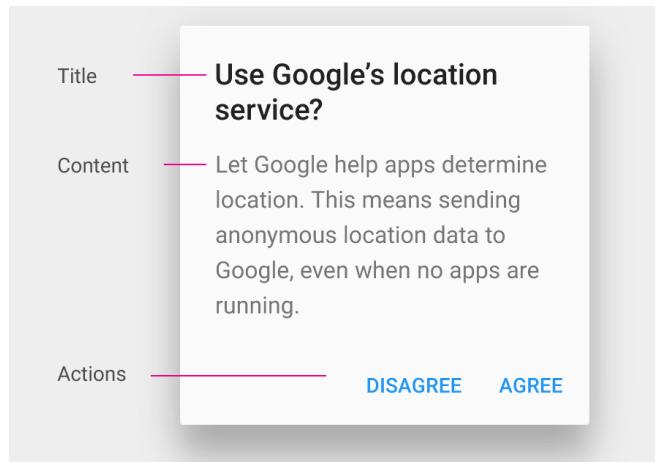
Tipe kedua adalah dialog non-modal yang memungkinkan pengguna untuk mengklik atau ketuk di luar mereka untuk memecat mereka.

Kita harus menggunakan dialog modal (jenis pertama) hanya untuk interaksi yang sangat penting (misalnya, hapus akun, setujui persyaratan dan ketentuan).



Gambar 30. Dialog modal: Pengguna harus mengkonfirmasi tindakan penghapusan dengan benar-benar mengetik delete.

Juga sistem seluler (native) dialog adalah modal dan biasanya mengikuti elemen-elemen dasar berikut: - konten, tindakan, dan judul.

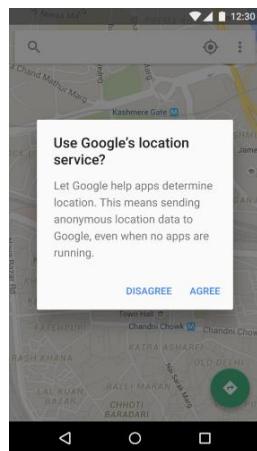


Gambar 31. Jendela dialog Android modal.

## e. Konsistensi Visual

### Latar Belakang Di Balik Dialog

Saat membuka dialog, penting bahwa halaman di belakang sedikit gelap. Hal ini mengindikasikan dua aktivitas. Pertama-tama ia menarik perhatian pada hamparan dan kedua memungkinkan pengguna mengetahui bahwa halaman tersebut saat ini tidak aktif.



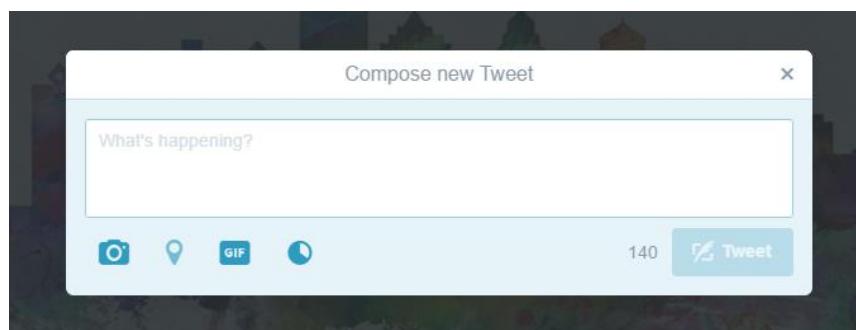
Gambar 32. Dialog modal Android.

Hati-hati dengan *tinting*. Jika kita menjadikan terlalu gelap, pengguna tidak akan lagi dapat melihat halaman di latar belakang. Jika kita membuatnya terlalu terang, pengguna mungkin berpikir bahwa halaman tersebut masih aktif dan bahkan mungkin tidak melihat dialog di tempat pertama.

## Pilihan Hapus Tutup (Close Open)

Harus ada opsi tutup untuk dialog di sudut kanan atas. Banyak dialog memiliki tombol 'x' di sudut jendela yang digunakan pengguna untuk keluar dari jendela. Namun, tombol 'x' ini bukan jalur keluar yang cukup mudah bagi rata-rata pengguna. Biasanya membutuhkan lebih banyak waktu dan upaya untuk mengeklik 'x' karena ukurannya lebih kecil dan pengguna harus melihat dan mengeklik (mengetuk).

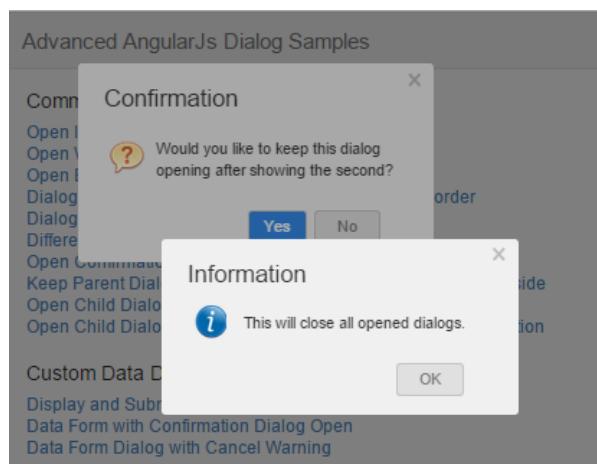
Sebaiknya izinkan pengguna untuk keluar dari jendela non-modal ketika mereka mengeklik area latar belakang di luarnya.



Gambar 33. Twitter menggunakan opsi dekat 'x' dan mengeklik latar belakang untuk keluar.

#### Hindari Dialog menampilkan Dialog

Harus dihindari perancangan dialog menampilkan dialog sederhana tambahan, karena mereka meningkatkan kompleksitas situs



Gambar 34. Contoh Buruk: Dialog dalam dialog

### 3 Urutan dari akses komponen user interface dialog dijelaskan. Panduan merancang akses dialog

### a. Umum

- Jangan gunakan kotak dialog yang dapat discroll. Jangan gunakan kotak dialog yang mengharuskan penggunaan bilah gulir (scrollbar) untuk dilihat sepenuhnya selama penggunaan normal. Mendesain ulang kotak dialog sebagai gantinya. Pertimbangkan untuk menggunakan pengungkapan atau tab progresif.
- Jangan gunakan bilah menu atau bilah status. Sebagai gantinya, berikan akses ke perintah dan status secara langsung pada kotak dialog itu sendiri, atau dengan menggunakan menu konteks pada kontrol yang relevan.  
Pengecualian: Bilah menu dapat diterima ketika kotak dialog digunakan untuk mengimplementasikan jendela utama (seperti utilitas).
- Jika kotak dialog membutuhkan perhatian segera dan program tidak aktif, tekan tombol taskbar tiga kali untuk menarik perhatian, dan biarkan disorot. Jangan lakukan hal lain: jangan pulihkan atau aktifkan jendela dan jangan memainkan efek suara apa pun. Sebagai gantinya, hormati pilihan status jendela pengguna dan biarkan pengguna mengaktifkan jendela saat siap.

### b. Kotak dialog modal

- Gunakan sesekali untuk tugas-tugas yang penting dan jarang, yang membutuhkan penyelesaian sebelum melanjutkan.
- Gunakan model commit yang tertunda sehingga perubahan tidak berlaku sampai komitmen secara eksplisit.
- Implementasikan menggunakan dialog tugas kapan pun untuk mencapai tampilan yang konsisten..

### c. Kotak dialog tanpa bingkai

- Gunakan untuk tugas yang sering, berulang, dan yang sedang berlangsung.

- Gunakan model komit langsung sehingga perubahan segera berlaku.
- Untuk dialog modeless, gunakan tombol perintah Tutup secara eksplisit pada dialog untuk menutup jendela. Untuk keduanya, gunakan tombol Tutup pada bilah judul untuk menutup jendela.
- Pertimbangkan membuat model kotak dialog *dockable*. Dialog modeless Dockable memungkinkan penempatan lebih fleksibel.

#### **d. Interaksi**

- Ketika ditampilkan, kotak dialog yang dimulai pengguna harus selalu mengambil fokus input. Kotak dialog yang diprakarsai program tidak boleh mengambil fokus input karena pengguna mungkin berinteraksi dengan jendela lain. Interaksi semacam itu salah arah di kotak dialog mungkin memiliki konsekuensi yang tidak diinginkan.
- Tetapkan fokus input awal ke kontrol yang paling mungkin untuk berinteraksi dengan pengguna pertama, yang biasanya (tetapi tidak selalu) kontrol interaktif pertama. Hindari menetapkan fokus input awal ke tautan Bantuan.
- Untuk navigasi keyboard, urutan tab harus mengalir dalam urutan logis, umumnya dari kiri ke kanan, dari atas ke bawah. Biasanya urutan tab mengikuti perintah membaca, tetapi pertimbangkan untuk membuat pengecualian ini:
  - Masukkan kontrol yang paling umum digunakan sebelumnya di urutan tab.
  - Sertakan tautan Bantuan di bagian bawah kotak dialog, setelah tombol commit dalam urutan tab.

Saat mengasign requirement, asumsikan bahwa pengguna menampilkan kotak dialog untuk tujuan yang dimaksudkan; jadi, misalnya, pengguna menampilkan dialog pilihan untuk membuat pilihan, bukan untuk meninjau dan mengklik Batal.

- Menekan tombol Esc selalu menutup kotak dialog yang aktif. Ini berlaku untuk kotak dialog dengan Batal atau Tutup, dan bahkan jika Batal telah diubah namanya menjadi Tutup karena hasilnya tidak dapat dibatalkan lagi.

### e. Kunci akses/Access Key

- Bila memungkinkan, berikan kunci akses unik ke semua kontrol interaktif atau label mereka. Kotak teks hanya-readable adalah kontrol interaktif (karena pengguna dapat menggulirnya dan menyalin teks) sehingga mereka mendapat manfaat dari kunci akses. Jangan menetapkan kunci akses ke:
  - Tombol OK, Batal, dan Tutup. Enter dan Esc digunakan untuk kunci akses mereka. Namun, selalu tetapkan kunci akses ke kontrol yang berarti OK atau Batal, tetapi memiliki label yang berbeda.
  - Group label. Biasanya, kontrol individu dalam grup diberi kunci akses, sehingga label grup tidak memerlukannya. Namun, jika ada kekurangan kunci akses, tetapkan kunci akses ke label grup dan bukan kontrol individu.
  - Tombol Bantuan Generik, yang diakses dengan F1.
  - Label Tautan. Sering kali ada terlalu banyak tautan untuk menetapkan kunci akses yang unik, dan garis bawah yang sering digunakan untuk menandai tautan menyembunyikan kunci jalur akses. Akses tautan dengan tombol Tab saja.
  - Nama tab. Tab diklik menggunakan Ctrl + Tab dan Ctrl + Shift + Tab.
  - Tombol telusuri berlabel "...". Tombol Browse ini tidak dapat diberi kunci akses secara unik.
  - Kontrol tanpa label, seperti kontrol putaran, tombol perintah grafis, dan kontrol pengungkapan progresif tanpa label.
  - Teks atau label statis non-label untuk kontrol yang tidak interaktif, seperti bilah kemajuan.
- Bila memungkinkan, berikan kunci akses untuk perintah yang umum digunakan sesuai dengan Tugas Kunci Akses Standar. Meskipun tugas kunci akses yang konsisten tidak selalu mungkin, kunci akses tentu lebih disukai terutama untuk kotak dialog yang sering digunakan.

- Tetapkan akses tombol komit terlebih dahulu untuk memastikan bahwa mereka memiliki penugasan kunci standar (*standard key assignment*). Jika tidak ada penugasan kunci standar, gunakan huruf pertama dari kata pertama. Misalnya, tombol akses untuk tombol Ya dan Tidak Komit harus selalu "Y" dan "N", terlepas dari kontrol lainnya di kotak dialog.
- Untuk membuat kunci akses mudah ditemukan, tetapkan kunci akses ke karakter yang muncul di awal label, idealnya karakter pertama, bahkan jika ada kata kunci yang muncul nanti di label.
- Lebih disarankan gunakan karakter dengan lebar dan tinggi seimbang, seperti w, m, dan huruf besar.
- Lebih disarankan konsonan atau vokal yang khas, seperti "x" di Exit.
- hindari penggunaan karakter yang ber-garis bawah sulit untuk dilihat, seperti (dari yang paling bermasalah hingga yang paling tidak bermasalah):
  - Huruf yang hanya selebar satu piksel, seperti i dan l.
  - Huruf turunan, seperti g, j, p, q, dan y.
  - Huruf di samping huruf dengan keterurutan.

#### f. Dialog progress:

Dialog yang digunakan untuk menunjukkan kemajuan suatu proses. Untuk tugas yang berjalan lama, asumsikan bahwa pengguna akan melakukan sesuatu yang lain saat tugas selesai. Rancang tugas untuk berjalan tanpa pengawasan.

- Sajikan kepada pengguna dengan kotak dialog umpan balik kemajuan jika operasi membutuhkan waktu lebih lama dari lima detik untuk menyelesaikan, bersama dengan perintah untuk membatalkan atau menghentikan operasi.
  - Pengecualian: Untuk alur wizard dan alur tugas, gunakan dialog modal untuk kemajuan hanya jika tugas tetap pada halaman yang sama (dibandingkan dengan memajukan ke halaman lain) dan pengguna tidak dapat melakukan apa pun saat menunggu. Jika tidak, gunakan halaman kemajuan atau

kemajuan di tempat.

- Jika operasi adalah tugas yang berjalan lama (lebih dari 30 detik) dan dapat dilakukan di latar belakang, gunakan dialog progress modeless sehingga pengguna dapat terus menggunakan program kita saat menunggu.

*Modelles progress dialog:*

- Miliki tombol Minimalkan di bilah judul.
- Ditampilkan di bilah tugas.
- Menerapkan dialog kemajuan modeless sehingga tugas/aktivitas terus berjalan bahkan jika pemilik jendela ditutup.
- Berikan tombol perintah untuk menghentikan operasi jika dibutuhkan lebih dari beberapa detik untuk selesai, atau memiliki potensi untuk tidak pernah selesai. Label tombol Batal jika membatalkan mengembalikan lingkungan ke keadaan sebelumnya (tanpa meninggalkan efek samping); jika tidak, beri label pada tombol Berhenti untuk menunjukkan bahwa itu meninggalkan operasi yang selesai sebagian utuh. kita dapat mengubah label tombol dari Batal menjadi Berhenti di tengah operasi, jika pada titik tertentu tidak mungkin mengembalikan lingkungan ke keadaan sebelumnya.
- Berikan tombol perintah untuk menjeda operasi jika dibutuhkan lebih dari beberapa menit untuk diselesaikan, dan ini merusak kemampuan pengguna untuk menyelesaikan pekerjaan. Melakukannya tidak memaksa pengguna untuk memilih antara menyelesaikan tugas dan menyelesaikan pekerjaan mereka.
- Kumpulkan sebanyak mungkin informasi sebelum memulai tugas.
- Jika masalah yang dapat dideteksi terdeteksi, minta pengguna mengatasi semua masalah yang ditemukan di akhir tugas. Jika itu tidak praktis, minta pengguna mengatasi masalah saat terjadi.
- Jangan tinggalkan tugas sebagai hasil dari kesalahan yang dapat dipulihkan.
- Tunjukkan masalah dengan memutar bilah kemajuan menjadi merah.
- Jika hasilnya jelas bagi pengguna, tutup dialog progres secara

otomatis pada penyelesaian yang berhasil. Jika tidak, gunakan umpan balik hanya untuk melaporkan masalah:

- Untuk menampilkan umpan balik sederhana, tampilkan umpan balik dalam dialog progres, dan ubah tombol Batal ke Tutup.
- Untuk menampilkan umpan balik terperinci, tutup kotak dialog progres dan tampilkan dialog informasi.
- Jangan gunakan pemberitahuan untuk umpan balik penyelesaian. Gunakan dialog kemajuan atau pemberitahuan keberhasilan aksi, tetapi tidak keduanya.

### **Waktu yang tersisa**

- Gunakan format waktu berikut. Mulai dengan yang pertama dari format berikut di mana unit waktu terbesar tidak nol, lalu ubah ke format berikutnya setelah unit waktu terbesar menjadi nol.

### **Untuk bilah kemajuan:**

- Jika informasi terkait ditampilkan dalam format titik dua:
- Sisa waktu: h jam, m menit
- Sisa waktu: m menit, detik
- Sisa waktu: detik s

## **4 Simulasi (mock-up) dari aplikasi yang akan dikembangkan dibuat.**

Mockup UI membantu menjawab pertanyaan-pertanyaan visual yang krusial.

Tips di bawah menyajikan panduan perancangan mock-up web / blog.

### **a. Pertama Sketsa Ide**

Sketsa merupakan aktivitas yang cepat, mudah, dan bebas risiko.

Sebelum melangkah lebih jauh, ke dalam piksel dan dimensi kotak, membuat sketsa pikiran kita dapat menjadi draf pertama yang membantu untuk mengatur ide-ide kita selanjutnya. Sketsa bisa dibuat dengan manual (di atas kertas) atau menggunakan prangkat lunak pembuat sketsa.



Gambar 35. Contoh sketsa UI

Lagi pula, lebih mudah menguji konsep dalam sketsa daripada dalam format digital. Kita bahkan dapat membuat sketsa beberapa komposisi yang berbeda dan memutuskan yang terbaik setelah semuanya disusun bersama - hanya membutuhkan sedikit waktu dan beberapa kertas.

### b. Mulai dengan Layar Ponsel

Tidak masalah apakah kita mendesain maket, wireframes, atau prototipe - selalu dirancang pada layar terkecil terlebih dahulu. Meskipun pendekatan seluler pertama adalah praktik terbaik desain responsif umum, namun kurang relevan saat merancang maket.

Dimulai dengan versi terkecil dan kemudian di-scale up skala desainnya untuk pertimbangan hanya yang penting terlebih dahulu, kemudian tambahkan konten sekunder dalam versi yang lebih baru.

Alternatif lain adalah mendesain dengan banyak kebebasan dan kemudian menghapus elemen saat kita menurunkan skala, yang sering mengarah pada komplikasi dan kemunduran. Desain seluler pertama memadatkan komposisi yang paling merepotkan terlebih dahulu, membuatnya lebih mudah untuk mempertahankan pengalaman yang konsisten di layar yang lebih besar

sesudahnya.

### c. Gunakan Perangkat Wireframing & Prototyping yang Kompatibel

Mockup hanya satu tautan dalam rantai proses desain yang lebih besar. Efektivitasnya tergantung pada kedua wireframe sebelumnya dan prototipe sesudahnya. Dan ketiganya bekerja paling baik ketika didigitalkan.

Desainer cenderung memilih perangkat lunak yang dibuat untuk visual daripada desain, perangkat lunak seperti Photoshop atau Sketch. Ini memungkinkan mereka lebih banyak pilihan untuk desain grafis, tetapi dapat menyebabkan komplikasi ketika mentransfer ke perangkat lunak prototyping.

Demikian juga, wireframes dan prototipe memiliki perangkat lunak mereka dengan lebih banyak pilihan untuk interaktivitas, pola UI, pustaka yang dapat disesuaikan, dan elemen pintar.

Beberapa wireframing dan prototyping tools memungkinkan pengguna untuk langsung mengintegrasikan Photoshop dan dokumen Sketch, membantu desainer menambahkan elemen interaktif ke dokumen-dokumen ini sekali statis hanya sebagai menyeret dan menjatuhkan.

### d. Berkomitmen Pada Perangkat Lunak Desain UI Yang sudah dipilihan

Misalnay dalam perdebatan antara *Sketch* dan *Photoshop*, keduanya berpihak dan tetap tertanam pada pendapat mereka masing-masing. Masalahnya adalah, satu pihak tidak secara inheren lebih baik dari yang lain – hal ini tergantung pada selera perancang.

Ini tidak berarti bahwa Sketch dan Photoshop adalah sama. Sketch

menawarkan banyak fitur desain yang berbeda dari Photoshop, sementara Photoshop memiliki kemampuan visual yang lebih maju.

Jika kita belum memiliki preferensi, lihat dan pertimbangkan menggunakan keduanya dan lihat mana yang berhasil untuk kita. Peluang kita akan merasa lebih nyaman dengan satu atau yang lain.

#### e. Tinjau Keberhasilan Visual Lainnya

Terkadang cara terbaik untuk belajar adalah dengan hanya mengamati.

Kiya hampir dapat secara intuitif mengatakan situs dan produk mana yang memiliki "itu," faktor X, tetapi dapatkah kita memberi tahu mengapa? Pelajari contoh lain dari kesuksesan visual dan lihat apakah kita dapat menentukan apa yang mereka lakukan dengan benar, dan bagaimana kita dapat menggunakan taktik untuk situs kita sendiri.

Beberapa galeri-galeri bisa dijadikan acuan, yang sebagian besar memperbarui setiap hari:

- [awwwards \(<https://www.awwwards.com/>\)](https://www.awwwards.com/)
- [Dribbble \(<https://dribbble.com/tags/mockup>\)](https://dribbble.com/tags/mockup)
- [Site Inspire \(<http://www.siteinspire.com/>\)](http://www.siteinspire.com/)
- [CSS Design Awards \(<https://www.cssdesignawards.com/>\)](https://www.cssdesignawards.com/)

#### f. Hapus Unsur yang Tidak Diperlukan

Secara visual, antarmuka yang berantakan tidak efektif. Anatr muka seperti ini mengalihkan perhatian, belum lagi mereka secara negatif mempengaruhi pemahaman, keterlihatan, dan keterbacaan. Untuk alasan ini, pertahankan jumlah elemen hingga seminimal mungkin.

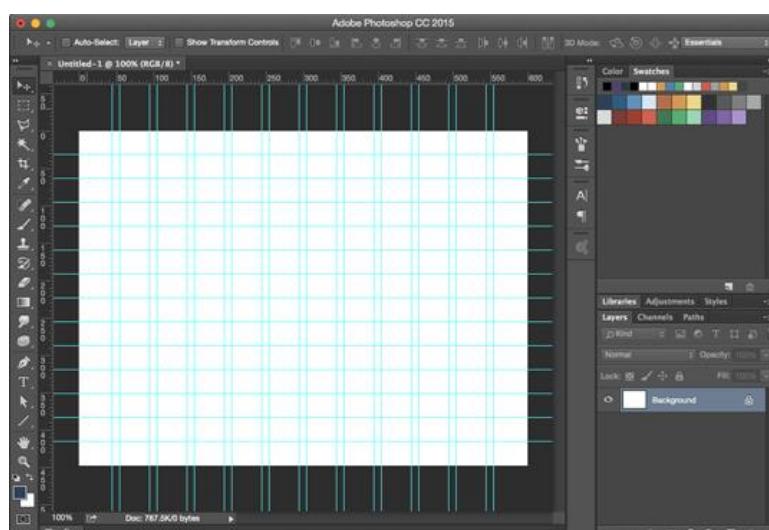
Unsur yang lebih sedikit berarti yang kita pilih untuk tetap lebih dominan. Bahkan jika kita tidak menggunakan gaya minimalis, elemen yang tidak perlu masih mengaburkan konten utama, sehingga melemahkan tujuan kita yang paling penting. Jika tidak perlu ke UX, singkirkan.

Dalam *The Guide to Interactive Wireframing*, profesor desain Tom Green menguraikan prosedur yang berfokus pada pengguna untuk memutuskan apa yang tetap dan apa yang terjadi:

- Inventarisasi konten dari semua elemen prospektif di halaman
- Prioritaskan elemen-elemen ini dan hapus elemen-elemen yang tidak diperlukan
- Sesuaikan elemen yang tersisa menjadi hierarki visual berdasarkan kepentingan
- Buat blok konten berdasarkan kategori elemen
- Tambahkan dan kurangi blok ke dalam tata letak sesuai prioritas
- Desain "Sculpt" pada setiap iterasi hingga mulai menyerupai objek antarmuka

### **g. Menerapkan Sistem Grid**

Seperti instrumen kerja lainnya, sistem grid berevolusi untuk memfasilitasi tugas sehari-hari desainer digital. Sementara beberapa masih menolak untuk menggunakannya, mereka tetap bermanfaat, atau mereka tidak akan ada. Sebuah sistem grid yang terorganisir memungkinkan desain secara tepat mengukur keberpihakan, ruang putih, dan hierarki konten ke pixel.



Gambar 36. Contoh perancangan UI berbasis Grid

Sementara grid horizontal yang paling banyak digunakan, grid vertikal (baseline) masih bisa berguna, terutama dengan tipografi.

### **h. Manfaatkan Elemen dan Ikon UI Gratis**

Masing-masing gaya setiap tombol, ikon, dan grafik dapat memakan waktu lebih banyak, jika tidak lebih, sebagai mockup itu sendiri. Manfaatkan kit UI gratis dan koleksi, seperti kit ikon media sosial gratis, untuk menghemat waktu.



Gambar 37. Contoh elemen ikon gratis

Banyak pihak menawarkan kit gratis untuk publisitas, untuk membangun komunitas, atau bahkan keluar dari kebaikan hati mereka. Kit ini sering dilapisi untuk penyesuaian warna atau tematik yang mudah, sehingga kita dapat mempersonalisasikannya ke proyek kita.

Ada banyak peralatan untuk Photoshop dan Sketch, dan kita juga dapat menemukan sumber daya tambahan untuk aplikasi seluler - seperti perangkat Android Lollipop ini - dengan pondasi yang sudah dibangun sebelumnya.

### **i. Gunakan Vektor**

Gambar tampilan grafis raster dengan set piksel tetap (seperti foto dan video), sedangkan vektor mengubah jumlah piksel berdasarkan resolusi dan ukuran layar - mereka dapat diskalakan tanpa kehilangan kualitas. Kapanpun memungkinkan, gunakan grafik vektor.

Skala grafik vektor dengan cepat, beradaptasi dengan definisi tinggi, layar

Modul Pelatihan Berbasis Kompetensi Bidang Pekerjaan Domestik	Kode Modul J.62010.033.02
<p>retina pada dua atau tiga kali ukuran. Photoshop dan Sketsa dirancang dengan pemikiran ini. Bentuk dan jalur khusus Photoshop secara otomatis menjadi vektor, dan Sketch menawarkan dukungan vektor secara default.</p> <p>B. Keterampilan yang diperlukan dalam mengidentifikasi rancangan user interface.</p> <ol style="list-style-type: none"><li>1. Mengidentifikasi rancangan user interface yang tepat</li><li>2. Mengidentifikasi komponen user interface dialog sesuai konteks rancangan proses</li><li>3. Menjelaskan urutan akses komponen user interface dialog</li><li>4. Membuat simulasi dari aplikasi yang akan dikembangkan</li></ol> <p>C. Sikap Kerja yang diperlukan dalam mengidentifikasi rancangan user interface.</p> <ol style="list-style-type: none"><li>1. Harus cermat dalam menentukan kebutuhan uji coba dalam pengembangan</li><li>2. Harus cermat memilih tools yang digunakan</li><li>3. Harus teliti dan taat atas</li><li>4. Harus tekun</li></ol>	
Judul Modul Mengimplementasikan <i>User Interface</i> Buku Informasi	Versi: 2018
Halaman: 38 dari 60	

## **BAB III**

### **MELAKUKAN IMPLEMENTASI RANCANGAN USER INTERFACE**

A. Pengetahuan yang diperlukan dalam melakukan implementasi rancangan user interface.

#### **1 Menu program sesuai dengan rancangan program diterapkan**

Pada bagian ini dijelaskan bagaimana mengimplementasikan UI menggunakan CSS.

##### **a. CSS**

CSS (Cascading Style Sheets) digunakan untuk membuat style dan lay out halaman web - misalnya, untuk mengubah font, warna, ukuran dan jarak dari konten ita, membaginya menjadi beberapa kolom, atau menambahkan animasi dan fitur dekoratif lainnya.

#### **Bagaimana cara CSS mempengaruhi HTML?**

Browser web menerapkan Rule CSS ke dokumen untuk mempengaruhi bagaimana disain user interface ditampilkan. Aturan/Rule CSS dibentuk dari:

- Satu set properti, yang memiliki nilai ditetapkan untuk memperbarui bagaimana konten HTML ditampilkan, misalnya kita ingin elemen menjadi 50% dari elemen induknya, dan latar belakangnya menjadi merah.
- Selektor, pemilih elemen yang ingin diterapkan pada nilai properti yang diperbarui. Sebagai contoh, kita ingin menerapkan aturan CSS ke semua paragraf di dokumen HTML.

Seperangkat aturan CSS yang terdapat dalam stylesheet menentukan tampilan halaman web. Kita akan belajar lebih banyak tentang seperti apa sintaks CSS dalam artikel berikutnya dari modul - CSS Syntax.

#### **Contoh CSS**

Deskripsi di atas mungkin tidak terlalu memberikan pengertian yang memadai, jadi mari kita perjelas dengan menyajikan contoh berikut.

Pertama-tama, misalnya ambil dokumen HTML sederhana, yang berisi <h1> dan <p> (perhatikan bahwa stylesheet diterapkan ke HTML menggunakan elemen <link>):

```
1 | <!DOCTYPE html>
2 | <html>
3 |   <head>
4 |     <meta charset="utf-8">
5 |     <title>My CSS experiment</title>
6 |     <link rel="stylesheet" href="style.css">
7 |   </head>
8 |   <body>
9 |     <h1>Hello World!</h1>
10 |     <p>This is my first CSS example</p>
11 |   </body>
12 | </html>
```

Sekarang perhatikan contoh CSS sederhana yang hanya terdiri dari dua aturan/rule sebagai berikut.

```
1 | h1 {
2 |   color: blue;
3 |   background-color: yellow;
4 |   border: 1px solid black;
5 |
6 | }
7 | p {
8 |   color: red;
9 | }
```

Rule pertama dimulai dengan selector **h1**, yang berarti hal ini akan menerapkan nilai property pada elemen <h1>.

Aturan ini berisi tiga properti dan nilainya (masing-masing pasangan properti / nilai disebut deklarasi):

- Yang pertama mengatur warna teks menjadi biru.
- Yang kedua mengatur warna latar belakang menjadi kuning.
- Yang ketiga menempatkan perbatasan di sekitar header lebar 1 piksel, padat (tidak putus-putus, atau putus-putus, dll.), Dan berwarna hitam.

Aturan kedua dimulai dengan pemilih p, yang artinya akan menerapkan nilai propertinya ke elemen <p>. Ini berisi satu deklarasi, yang mengatur warna teks menjadi merah.

Di browser web, kode di atas akan menghasilkan output berikut:

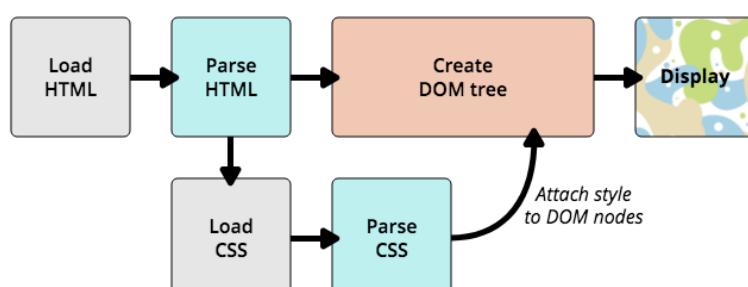


## Cara kerja CSS

Ketika browser menampilkan dokumen, browser harus menggabungkan konten dokumen dengan informasi tentang stylenya. Hal ini akan memproses dokumen dalam dua tahap:

- 1 Browser mengubah HTML dan CSS menjadi DOM (*Document Object Model*). DOM mewakili dokumen dalam memori komputer. Ini menggabungkan konten dokumen dengan gayanya.
- 2 Peramban/Browser menampilkan isi DOM.

Perhatikan gambar berikut:



Gambar 38. Cara kerja CSS

## DOM

DOM memiliki struktur seperti pohon. Setiap elemen, atribut dan potongan teks dalam bahasa markup menjadi simpul DOM dalam struktur pohon. Node ditentukan oleh hubungannya dengan node DOM lainnya. Beberapa elemen adalah orang tua dari simpul anak, dan simpul anak memiliki saudara kandung.

Memahami DOM membantu Anda merancang, men-debug, dan memelihara CSS karena DOM adalah tempat CSS dan konten dokumen Anda bertemu.

## Representasi DOM

Berikut contoh untuk melihat bagaimana DOM dan CSS bekerja bersama.

Misalnya kita asumsikan kode HTML berikut:

```
1 <p>
2   Let's use:
3     <span>Cascading</span>
4     <span>Style</span>
5     <span>Sheets</span>
6   </p>
```

Di DOM, node yang terkait dengan elemen `<p>` di atas adalah induk. Anak-anaknya adalah simpul teks dan simpul yang terkait dengan elemen `<span>`. Node SPAN juga orang tua, dengan simpul teks sebagai anak-anak mereka:

```
1 P
2 └ "Let's use:"
3   └ SPAN
4     └ "Cascading"
5     └ SPAN
6       └ "Style"
7       └ SPAN
8         └ "Sheets"
```

Seperti potongan gambar di atas lah, cara browser menafsirkan cuplikan HTML sebelumnya —membuat pohon DOM di atas dan kemudian menampilkannya di peramban seperti ini:

Let's use: Cascading Style Sheets

[Open in CodePen](#)

[Open in JSFiddle](#)

Gambar 39. Contoh tampilan browser

## Menerapkan CSS ke DOM

Katakanlah kita menambahkan beberapa CSS ke dokumen, untuk mengurnya. Sekali lagi, HTML adalah sebagai berikut:

```
1 | <p>
2 |   Let's use:
3 |   <span>Cascading</span>
4 |   <span>Style</span>
5 |   <span>Sheets</span>
6 | </p>
```

Jika kita menerapkan CSS untuknya, sbb:

```
1 | span {
2 |   border: 1px solid black;
3 |   background-color: lime;
4 | }
```

Browser akan mengurai/parse HTML dan membuat DOM darinya, lalu menguraikan CSS. Karena satu-satunya aturan yang tersedia di CSS memiliki *span selection*, itu akan menerapkan aturan itu untuk masing-masing dari tiga *span*. Output yang diperbarui adalah sebagai berikut:

Let's use: Cascading Style Sheets

[Open in CodePen](#)

[Open in JSFiddle](#)

## Menerapkan CSS Anda ke HTML

Ada tiga cara berbeda untuk menerapkan CSS ke dokumen HTML yang biasanya ditemui, beberapa lebih bermanfaat daripada yang lain. Di sini kita akan meninjau masing-masing secara singkat.

- External Stylesheet

Stylesheet eksternal berlaku ketika kita memiliki CSS yang ditulis dalam file terpisah dengan ekstensi .css, dan kita mereferensikannya dari elemen HTML <link>. File HTML terlihat seperti ini:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>My CSS experiment</title>
6          <link rel="stylesheet" href="style.css">
7      </head>
8      <body>
9          <h1>Hello World!</h1>
10         <p>This is my first CSS example</p>
11     </body>
12 </html>
```

dan file CSS nya adalah :

```
1  h1 {
2      color: blue;
3      background-color: yellow;
4      border: 1px solid black;
5  }
6
7  p {
8      color: red;
9  }
```

- o Internal stylesheet

Stylesheet internal ketika tidak menggunakan file CSS eksternal, tetapi menempatkan CSS di dalam elemen <style>, yang terdapat di dalam kepala HTML. Jadi HTML akan terlihat seperti ini:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>My CSS experiment</title>
6      <style>
7        h1 {
8          color: blue;
9          background-color: yellow;
10         border: 1px solid black;
11       }
12
13       p {
14         color: red;
15       }
16     </style>
17   </head>
18   <body>
19     <h1>Hello World!</h1>
20     <p>This is my first CSS example</p>
21   </body>
22 </html>
```

- **Inline Styles**

Gaya sebaris adalah deklarasi CSS yang hanya memengaruhi satu elemen, yang terdapat dalam atribut style:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>My CSS experiment</title>
6    </head>
7    <body>
8      <h1 style="color: blue; background-color: yellow; border: 1px solid black;">Hello World!
9      <p style="color: red;">This is my first CSS example</p>
10     </body>
11 </html>
```

## b. Sintak CSS

Pada tingkat paling dasar, CSS terdiri dari dua blok bangunan:

**Properti/Property:** Pengenal yang dapat dibaca manusia yang menunjukkan fitur gaya (misalnya font, lebar, warna latar belakang) yang ingin diubah.

**Nilai/Value:** Setiap properti yang ditentukan diberi nilai, yang menunjukkan bagaimana kita ingin mengubah fitur gaya tersebut (misalnya font apa yang ingin kita ubah, lebar atau warna latar belakang yang dikehendaki.)

Properti yang dipasangkan dengan nilai disebut deklarasi CSS. Deklarasi CSS dimasukkan dalam Blok Deklarasi CSS. Dan akhirnya, blok deklarasi CSS dipasangkan dengan pemilih untuk menghasilkan CSS Rulesets (atau CSS Rules).

Sebelum terlalu mendalam dalam teori dan penjelasan tertulis, mari kita lihat contoh konkret berikut:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>My CSS experiment</title>
6      <link rel="stylesheet" href="style.css">
7    </head>
8    <body>
9      <h1>Hello World!</h1>
10     <p>This is my first CSS example</p>
11
12     <ul>
13       <li>This is</li>
14       <li>a list</li>
15     </ul>
16   </body>
17 </html>
```

dengan CSS filenya adalah:

```
1  h1 {
2    colour: blue;
3    background-color: yellow;
4    border: 1px solid black;
5  }
6
7  p {
8    color: red;
9  }
10
11 p, li {
12   text-decoration: underline;
13 }
```

Kombinasi keduanya akan menghasilkan

**Hello World!**

[This is my first CSS example](#)

- [This is](#)
- [a list](#)

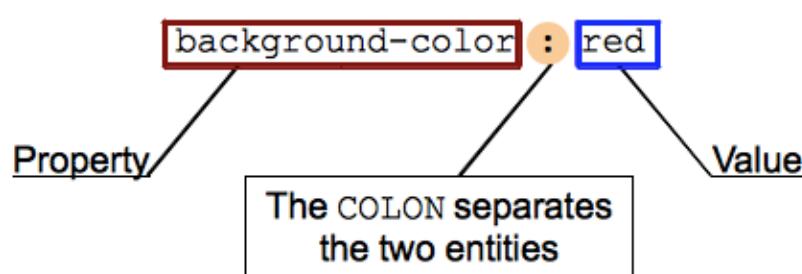
[Open in CodePen](#)

[Open in JSFiddle](#)

## Deklarasi CSS

Mengatur properti CSS ke nilai tertentu adalah fungsi inti dari bahasa CSS. Mesin CSS menghitung deklarasi yang berlaku untuk setiap elemen halaman agar dapat meletakkannya dengan tepat dan memberinya style. Yang penting untuk diingat adalah bahwa baik properti maupun nilai bersifat case-sensitive dalam CSS. Properti dan nilai dalam setiap pasangan dipisahkan oleh titik dua (:).

A CSS declaration :



Terdapat lebih dari 300 properti berbeda dalam CSS dan hampir tak terbatas jumlah nilai yang berbeda. Tidak semua pasangan properti dan nilai diizinkan; setiap properti memiliki daftar spesifik nilai valid yang ditentukan untuknya. Silakan pelajari referensi untuk memahami lebih rinci.

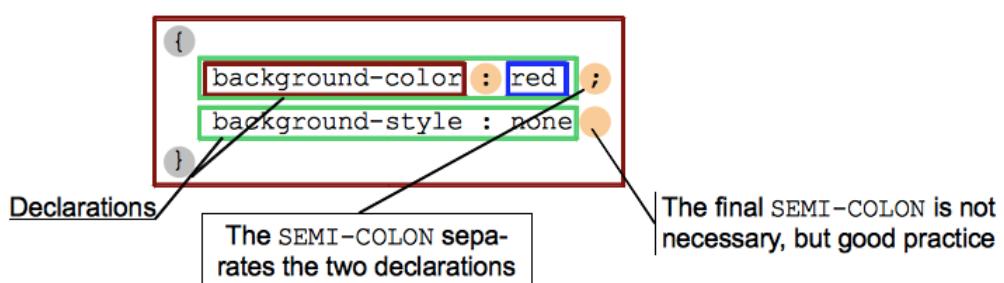
Dalam contoh di atas, ada lima deklarasi CSS terpisah. Dapatkah anda mengidentifikasi deklarasi yang tidak valid (salah atau salah) dan mencari tahu mengapa itu tidak valid?

### Deklarasi Blok CSS

Deklarasi dikelompokkan dalam blok, dengan setiap set deklarasi diapit oleh kurung kurawal buka, ({}) dan penutup (}).

Setiap deklarasi yang terkandung di dalam blok deklarasi harus dipisahkan oleh semi-kolon (;), jika tidak, kode tidak akan berfungsi (atau setidaknya akan memberikan hasil yang tidak diharapkan.) Deklarasi terakhir dari blok tidak perlu dihentikan oleh semi-kolon, meskipun sering dianggap gaya yang baik untuk melakukannya karena mencegah lupa untuk menambahkannya ketika memperpanjang blok dengan deklarasi lain.

A CSS declarations block:



### CSS selectors and rules

Berikut membahas bagaimana memberi tahu blok deklarasi, elemen mana yang harus diterapkan. Ini dilakukan dengan mengawali setiap blok deklarasi dengan pemilih - pola yang cocok dengan beberapa elemen di halaman. Deklarasi terkait akan diterapkan hanya untuk elemen tersebut. Selector dan blok deklarasi disebut seperangkat aturan/rule set, atau sering hanya sekadar aturan/rule.

Selector bisa menjadi sangat rumit - AnKitada dapat membuat aturan mencocokkan beberapa elemen dengan menyertakan beberapa selector

yang dipisahkan oleh koma (grup,) dan selector dapat dirangkai bersama-sama, misalnya kita ingin memilih elemen apa pun dengan kelas "bla", tetapi hanya jika berada di dalam elemen <article>, dan hanya ketika sedang digerakkan oleh pointer mouse. Pelajari lebih rinci di referensi-referensi yang membahas Selectors.

Sebuah elemen dapat dicocokkan oleh beberapa selector, oleh karena itu beberapa aturan dapat mengatur properti yang diberikan beberapa kali. CSS mendefinisikan yang mana yang didahulukan dari yang lain dan harus diterapkan: ini disebut algoritma kaskade, dan hal ini lebih rinci bisa dipelajari lebih banyak tentang cara kerjanya di bab Cascade dan pewarisan.

### Statement CSS

Rule CSS adalah blok bangunan utama dari style sheet - blok paling umum yang akan kita lihat di CSS. Namun ada jenis blok lain yang terkadang ditemukan - aturan CSS adalah salah satu jenis pernyataan CSS. Jenis lainnya adalah sebagai berikut:

- **At-rules** digunakan dalam CSS untuk menyampaikan metadata, informasi kondisional, atau informasi deskriptif lainnya. Rule ini mulai dengan tanda (@), diikuti oleh pengidentifikasi untuk mengatakan apa jenis aturan itu, maka blok sintaks semacam itu, diakhiri dengan titik koma (;). Setiap jenis at-rule, didefinisikan oleh identifier, akan memiliki sintaks dan semantik internalnya sendiri. Contohnya termasuk:
  - @charset and @import (metadata)
  - @media or @document (conditional information, also called nested statements, see below.)
  - @font-face (descriptive information)

Contoh sintaks yang spesifik seperti di bawah:

```
1 | @import 'custom.css';
```

at-rule ini mengimport file CSS lain ke current CSS.

**Nested statements** merupakan subset of at-rule yang spesifik, suatu sintak yang merupakan blok bersarang dari CSS rules yang hanya akan diaplikasikan pada dokumen jika suatu kondisi spesifik tertentu terpenuhi.

- @media konten at-rule diaplikasi jika dan hanya jika device yang menjalankan browser sesuai dengan kondisi ekspresinya
- @supports konten at-rule yang diaplikasikan jika dan hanya jika browser mendukung fitur yang diperiksa;
- @document konten at-rule yang diaplikasi jika dan hanya jika halaman saat ini sesuai dengan beberapa kondisi.

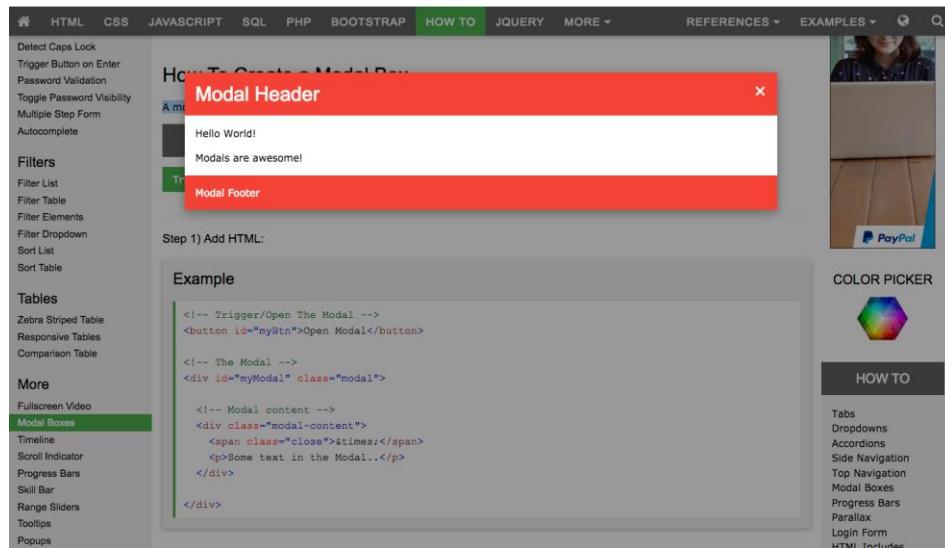
Contoh sintaks spesifiknya sebagai berikut:

```
1 | @media (min-width: 801px) {  
2 |   body {  
3 |     margin: 0 auto;  
4 |     width: 800px;  
5 |   }  
6 | }
```

## 2 Penempatan user interface dialog diatur secara sekuensial.

Pada bagian ini dijelaskan bagaimana membuat dialog Box menggunakan HTMl, CSS dan JavaScript.

Suatu window modal adalah sebuah kotak dialog (*dialog box/popup window*) yang ditampilkan di atas/di depan halaman yang aktif saat ini. Contoh berikut memperjelas ilustrasi ini.



### a. Tahap #1, Tambahkan HTML:

Contoh:

```
<!-- Trigger/Open The Modal -->
<button id="myBtn">Open Modal</button>

<!-- The Modal -->
<div id="myModal" class="modal">

    <!-- Modal content -->
    <div class="modal-content">
        <span class="close">&times;</span>
        <p>Some text in the Modal..</p>
    </div>

</div>
```

### b. Tahap #2, Tambahkan CSS:

Contoh:

```
/* The Modal (background) */
.modal {
    display: none; /* Hidden by default */
    position: fixed; /* Stay in place */
    z-index: 1; /* Sit on top */
    left: 0;
    top: 0;
    width: 100%; /* Full width */
    height: 100%; /* Full height */
    overflow: auto; /* Enable scroll if needed */
    background-color: rgb(0,0,0); /* Fallback color */
    background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
```

```
}

/* Modal Content/Box */
.modal-content {
    background-color: #fefefe;
    margin: 15% auto; /* 15% from the top and centered */
    padding: 20px;
    border: 1px solid #888;
    width: 80%; /* Could be more or less, depending on screen size */
}

/* The Close Button */
.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}
```

**c. Tahap #3, Tambahkan JavaScript:**

Contoh:

```
// Get the modal
var modal = document.getElementById('myModal');

// Get the button that opens the modal
var btn = document.getElementById("myBtn");

// Get the <span> element that closes the modal
var span = document.getElementsByClassName("close")[0];

// When the user clicks on the button, open the modal
btn.onclick = function() {
    modal.style.display = "block";
}

// When the user clicks on <span> (x), close the modal
span.onclick = function() {
    modal.style.display = "none";
}
```

```
// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
```

**d. Tahap #4, Tambahkan *Header* dan *Footer*:**

Contoh:

Tambahkan kelas untuk modal-header, modal-body dan modal-footer:

```
<!-- Modal content -->
<div class="modal-content">
    <div class="modal-header">
        <span class="close">&times;</span>
        <h2>Modal Header</h2>
    </div>
    <div class="modal-body">
        <p>Some text in the Modal Body</p>
        <p>Some other text...</p>
    </div>
    <div class="modal-footer">
        <h3>Modal Footer</h3>
    </div>
</div>
```

Update Style *modal header*, *body* dan *footer*, dan tambahkan *animation (slide in the modal)*:

```
/* Modal Header */
.modal-header {
    padding: 2px 16px;
    background-color: #5cb85c;
    color: white;
}

/* Modal Body */
.modal-body {padding: 2px 16px;}

/* Modal Footer */
.modal-footer {
    padding: 2px 16px;
    background-color: #5cb85c;
    color: white;
}

/* Modal Content */
.modal-content {
```

```
position: relative;
background-color: #fefefe;
margin: auto;
padding: 0;
border: 1px solid #888;
width: 80%;
box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2),0 6px 20px 0
rgba(0,0,0,0.19);
animation-name: animatetop;
animation-duration: 0.4s
}

/* Add Animation */
@keyframes animatetop {
    from {top: -300px; opacity: 0}
    to {top: 0; opacity: 1}
}
```

Kombinasi akhir dari tahapan-tahapn tersebut adalah:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}

/* The Modal (background) */
.modal {
    display: none; /* Hidden by default */
    position: fixed; /* Stay in place */
    z-index: 1; /* Sit on top */
    padding-top: 100px; /* Location of the box */
    left: 0;
    top: 0;
    width: 100%; /* Full width */
    height: 100%; /* Full height */
    overflow: auto; /* Enable scroll if needed */
    background-color: rgb(0,0,0); /* Fallback color */
    background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
}

/* Modal Content */
.modal-content {
    position: relative;
    background-color: #fefefe;
    margin: auto;
    padding: 0;
    border: 1px solid #888;
    width: 80%;
    box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2),0 6px 20px 0 rgba(0,0,0,0.19);
    -webkit-animation-name: animatetop;
    -webkit-animation-duration: 0.4s;
    animation-name: animatetop;
```

```

        animation-duration: 0.4s
    }

    /* Add Animation */
    @-webkit-keyframes animatetop {
        from {top:-300px; opacity:0}
        to {top:0; opacity:1}
    }

    @keyframes animatetop {
        from {top:-300px; opacity:0}
        to {top:0; opacity:1}
    }

    /* The Close Button */
    .close {
        color: white;
        float: right;
        font-size: 28px;
        font-weight: bold;
    }

    .close:hover,
    .close:focus {
        color: #000;
        text-decoration: none;
        cursor: pointer;
    }

    .modal-header {
        padding: 2px 16px;
        background-color: #5cb85c;
        color: white;
    }

    .modal-body {padding: 2px 16px;}

    .modal-footer {
        padding: 2px 16px;
        background-color: #5cb85c;
        color: white;
    }
</style>
</head>
<body>

<h2>Animated Modal with Header and Footer</h2>

<!-- Trigger/Open The Modal -->
<button id="myBtn">Open Modal</button>

<!-- The Modal -->
<div id="myModal" class="modal">

    <!-- Modal content -->
    <div class="modal-content">
        <div class="modal-header">
            <span class="close">&times;</span>

```

```
<h2>Modal Header</h2>
</div>
<div class="modal-body">
    <p>Some text in the Modal Body</p>
    <p>Some other text...</p>
</div>
<div class="modal-footer">
    <h3>Modal Footer</h3>
</div>
</div>

</div>

<script>
// Get the modal
var modal = document.getElementById('myModal');

// Get the button that opens the modal
var btn = document.getElementById("myBtn");

// Get the <span> element that closes the modal
var span = document.getElementsByClassName("close")[0];

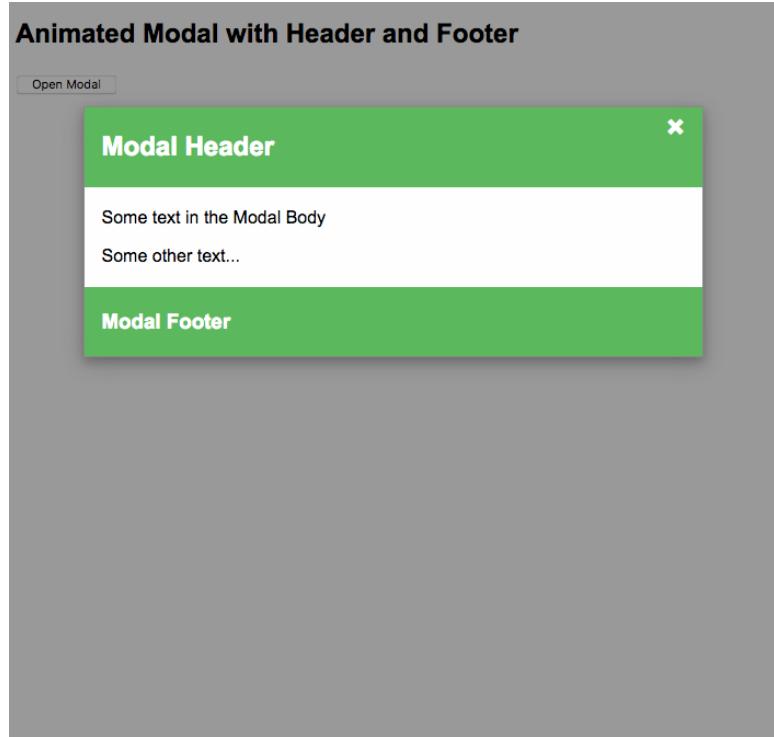
// When the user clicks the button, open the modal
btn.onclick = function() {
    modal.style.display = "block";
}

// When the user clicks on <span> (x), close the modal
span.onclick = function() {
    modal.style.display = "none";
}

// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
</script>

</body>
</html>
```

Dan contoh hasil dialog box adalah:



**B. Keterampilan yang diperlukan dalam melakukan implementasi rancangan user interface.**

- 1 Mampu menerapkan menu program sesuai rancangan
- 2 Mampu menempatkan user interface sekuensial

**C. Sikap Kerja yang diperlukan dalam melakukan implementasi rancangan user interface.**

- 1 Harus cermat dalam mempersiapkan dokumentasi uji cob melakukan implementasi rancangan user interface.
- 2 Harus cermat memilih tools yang digunakan
- 3 Harus teliti dan taat atas



KEMENTERIAN  
KETENAGAKERJAAN  
REPUBLIK INDONESIA



# MATERI PELATIHAN BERBASIS KOMPETENSI

MENERAPKAN PERINTAH EKSEKUSI BAHASA  
PEMROGRAMAN BERBASIS TEKS,  
GRAFIK DAN MULTIMEDIA  
J.620100.010.02

KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA  
Jl. Jenderal Gatot Subroto Kav. 51 Lt.VI A Telp. (021) 5262782. Jakarta Selatan

## DAFTAR ISI

Daftar Isi .....	2
Kata Pengantar .....	3
A. Pendahuluan .....	4
B. Panduan Penggunaan Modul .....	4
C. Daftar Ikon .....	6
D. Bacaan Referensi .....	6
E. Pengantar Teori .....	7
F. Langkah Kerja .....	84
G. Implementasi Unit Kompetensi .....	95
1. Elemen Kompetensi 1 .....	95
1. 1. Referensi 1.1 .....	95
1. 2. Aktifitas 1.2 .....	95
2. Elemen Kompetensi 2 .....	96
2.1. Referensi 2.1 .....	96
2.2. Aktivitas 2.2 .....	96
2.3. Aktivitas 2.3 .....	96
2.4. Aktivitas 2.4 .....	96
2.5. Aktivitas 2.5 .....	96
2.6. Video Youtube 2.6 .....	96
3. Elemen Kompetensi 3 .....	97
3.1 . Referensi 3.1 .....	97
3.2 . Aktivitas 3.2 .....	97
3.3 . Aktivitas 3.3 .....	97
3.4 . Aktivitas 3.4 .....	97
3.5 . Aktivitas 3.5 .....	97
3.6 . Aktivitas 3.6 .....	97
H. Lampiran .....	
1) Kamus Istilah .....	99
2) Referensi .....	103
3) Unit Kompetensi .....	105
4) Daftar Nama Penyusun .....	108

Assalamu'alaikum Wr. Wb.

Dengan memanjatkan puji syukur kehadirat Allah SWT buku Materi Pelatihan Berbasis Kompetensi dengan judul **"Menerapkan Perintah Eksekusi Bahasa Pemrograman berbasis Teks, Grafik, dan Multimedia (J.620100.010.12)"** dapat tersusun dengan baik dan menjadi media pembelajaran untuk mentransformasikan pengetahuan, keterampilan dan sikap kerja kepada peserta pelatihan.

Penyusunan Materi Pelatihan Berbasis Kompetensi merupakan hasil identifikasi silabus, capaian unit kompetensi, kriteria capaian yang lalu dituangkan ke dalam pokok pembahasan sebagaimana ditentukan dalam pedoman penyusunan materi pelatihan berbasis kompetensi.

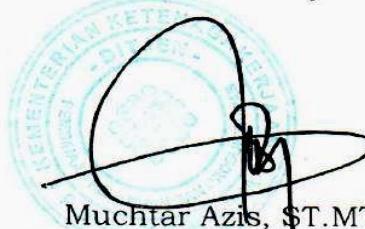
Materi pelatihan berbasis kompetensi diformulasikan menjadi 2 (dua) buku, yakni buku Materi dan buku Asesmen (penilaian) yang tidak terpisahkan dalam penggunaannya. Materi pelatihan ini menjadi salah satu bahan pengajaran kepada peserta pelatihan agar pelaksanaan pelatihan dapat dilakukan secara efektif dan efisien.

Kami berharap materi ini dapat meningkatkan kemampuan aplikatif bagi peserta pelatihan dan instruktur serta dapat dikembangkan lebih lanjut. Semoga Tuhan Yang Maha Esa memberikan tuntunan kepada kita semua dalam melakukan berbagai upaya untuk menunjang proses pelaksanaan pelatihan berbasis kompetensi guna menghasilkan tenaga kerja yang kompeten dan berdaya saing tinggi.

Wassalamu'alaikum Wr. Wb

Jakarta, September 2020

Plt. Direktur  
Bina Standardisasi Kompetensi dan  
Pelatihan Kerja



Muchtar Azis, ST.MT  
NIP. 19680505 199703 1 002

## A. PENDAHULUAN

Tuntutan pembelajaran berbasis kompetensi menjadi sangat penting dalam meningkatkan kualitas Sumber Daya Manusia (SDM) yang kompeten, sesuai dengan tuntutan kebutuhan pasar kerja. Selaras dengan tuntutan tersebut, maka dibutuhkan mekanisme pelatihan yang lebih praktis, aplikatif, serta dapat menarik dilaksanakan sehingga memotivasi para peserta dalam melaksanakan pelatihan yang diberikan. Seiring dengan mudahnya teknologi digunakan, maka materi pelatihan dapat disajikan dengan berbagai media pembelajaran sehingga dapat diakses secara offline dan online.

Materi pelatihan ini terdiri dari buku Panduan Materi Pelatihan dan buku Panduan Asesmen. Serta dilengkapi dengan materi yang bersifat soft copy seperti materi presentasi dan video.

## B. PANDUAN PENGGUNAAN MODUL

Beberapa ketentuan panduan penggunaan materi yang harus diperhatikan adalah sebagai berikut:

1. Materi ini dapat dijadikan rujukan untuk pelaksanaan PBK dengan penggunaannya dapat dikembangkan dan dikontekstualisasikan sesuai dengan kebutuhan, materi ini terdiri dari:
  - a. Bacaan Referensi
  - b. Pengantar Teori
  - c. Langkah Kerja
  - d. Implementasi Unit kompetensi
  - e. Lampiran :
    - 1) Kamus istilah
    - 2) Daftar referensi
    - 3) Unit kompetensi
    - 4) Daftar penyusun

2. Slide *powerpoint* dan video merupakan kelengkapan yang dapat dijadikan referensi bagi para instruktur.
3. Peran instruktur terkait dengan penggunaan modul, antara lain:
  - a. Instruktur dapat menggunakan modul dengan referensi video dan *powerpoint* yang terlampir dalam modul sebagai referensi, diharapkan dapat mengembangkan bahan yang disesuaikan dengan BLK masing-masing
  - b. Proses pembelajaran dapat disampaikan dengan menggunakan berbagai sumber yang menguatkan peserta pelatihan, baik melalui tahapan persiapan, pelaksanaan di kelas, praktik, melakukan investigasi, menganalisa, mendiskusikan, tugas kelompok, presentasi, serta menonton video.
  - c. Keseluruhan materi yang tersedia sebagai referensi dalam buku ini dapat menjadi bahan dan gagasan untuk dikembangkan oleh instruktur dalam memperkaya materi pelatihan yang akan dilaksanakan.
4. Buku penilaian menjadi kesatuan, namun disajikan dalam paket buku penilaian secara terpisah. Buku penilaian dapat berupa soal tertulis, panduan wawancara, serta instruksi demonstrasi yang akan dilaksanakan sesuai dengan proses penilaian yang dilaksanakan.
5. Referensi merupakan referensi yang menjadi acuan dalam penyusunan buku panduan pelatihan ini.
6. Lampiran merupakan bagian yang berisikan lembar kerja serta bahan yang dapat digunakan sebagai berkas kelengkapan pelatihan.

### C. DAFTAR IKON

Daftar ikon yang dapat digunakan dalam buku ini, antara lain:

Ikon	Keterangan
 Pemeriksaan	Ikon ini memiliki arti anda diminta untuk mencari atau menemui seseorang untuk mendapatkan informasi

 Aktivitas	Icon ini memiliki arti anda diminta untuk menuliskan/mencatat,melengkapi,latihan/aktivitas (bermain peran, presentasi) dan mencatatkan dalam lembar kerja pada buku/media lain sesuai instruksi
 Referensi material/manual	Icon ini memiliki arti anda harus melihat pada aturan atau kebijakan yang berlaku dan prosedur-prosedur atau materi pelatihan/ sumber informasi lain untuk dapat melengkapi latihan/ aktivitas ini.
 Berpikir	Icon ini memiliki arti ambil waktu untuk Anda dapat berpikir/ menganalisa informasi dan catat gagasan-gagasan yang anda miliki.
 Komunikasi/ Diskusi	Icon ini memiliki arti berbicara/ berdiskusi lah dengan rekan anda untuk gagasan yang anda miliki.
 Membaca	Icon ini memiliki arti pilihlah bacaan yang dibutuhkan sesuai dengan kebutuhan materi pelatihan.
 Video/Youtube	Icon ini memiliki arti pilihlah video/youtube yang dibutuhkan dalam materi pelatihan.

#### D. BACAAN REFERENSI



Membaca secara lengkap :

- Buku Pengantar Teknologi Informasi Edisi Revisi, Bab 6 (Perangkat Lunak Aplikasi) dan Bab 7 (Perangkat Lunak Sistem), Penulis Abdul Kadir & Terra Ch. Tri wahyuni, Penerbit Andi.

**MENERAPKAN PERINTAH EKSEKUSI BAHASA PEMROGRAMAN  
BERBASIS TEKS, GRAFIK DAN MULTIMEDIA**

Sebuah perangkat komputer tidak akan dapat digunakan jika tidak ada perangkat lunak (software) di dalamnya. Tanpa perangkat lunak, komputer hanyalah sekedar mesin yang tidak dapat melakukan tugas yang dikehendaki pemakainya. Perangkat lunak adalah salah satu perangkat utama dari sebuah sistem komputer. Dua perangkat lainnya adalah perangkat keras (*hardware*) dan pengguna (*brainware*). Perangkat lunak sesungguhnya adalah program. Program adalah deretan instruksi yang digunakan untuk mengendalikan komputer sehingga komputer dapat melakukan tindakan sesuai yang dikehendaki pembuatnya.

Berdasarkan fungsinya, perangkat lunak dapat dibagi menjadi dua golongan, yaitu perangkat lunak aplikasi dan perangkat lunak sistem

- Perangkat lunak aplikasi (application software) adalah program yang biasa digunakan pemakai untuk melakukan tugas-tugas yang spesifik, misalnya untuk membuat dokumen, memanipulasi foto, atau membuat laporan keuangan.
- Perangkat lunak sistem (system software, kadangkalah disebut perangkat lunak pendukung atau support software) adalah program yang digunakan untuk mengontrol sumber daya komputer seperti CPU dan peranti masukan/keluaran. Perangkat lunak sistem yang saat ini banyak digunakan adalah sistem operasi seperti Windows dan Linux.

Seorang pembuat perangkat lunak menuliskan instruksi program untuk kemudian akan dieksekusi atau dijalankan pada perangkat komputer. Agar dapat menuliskan instruksi program tersebut, maka seorang pembuat perangkat lunak atau dikenal dengan nama programmer harus memahami dan menguasai bahasa program tersebut.

Setelah dibuat dengan menggunakan bahasa pemrograman tertentu, kode sumber atau *source code* perlu diterjemahkan menjadi kode mesin yang berupa deretan angka 0 dan 1. Ini diperlukan karena perangkat komputer hanya mengingat kode biner. Berkas yang berisi kode mesin inilah yang kemudian akan terinstal pada komputer tersebut.

Pada umumnya, program dituliskan dengan menggunakan bahasa

pemrograman yang mudah dipahami oleh manusia. Dan biasanya menggunakan kata-kata bahasa Inggris, misalnya IF untuk menyatakan “jika” dan AND untuk menyatakan “dan”

Agar bahasa pemrograman yang dituliskan tersebut dapat dijalankan oleh perangkat komputer, maka diperlukan sebuah penerjemah bahasa pemrograman. Penerjemah bahasa pemrograman dibedakan menjadi *assembler*, kompiler dan interpreter. Adapun perbedaan dari 3 hal tersebut adalah:

1. Assembler adalah program yang digunakan untuk menerjemahkan kode sumber dalam bahasa rakitan (Assembly)
2. Kompiler (*compiler*) adalah program penerjemah yang mengkonversi semua kode sumber selain dalam bahasa rakitan menjadi kode objek. Hasilnya yang berupa kode objek inilah yang bisa dijalankan oleh komputer.
3. Interpreter adalah program yang menerjemahkan satu persatu instruksi dalam kode sumber dan kemudian segera menjalankan instruksi yang diterjemahkan tersebut.

#### **A. Klasifikasi dan Jenis – Jenis Bahasa Pemrograman**

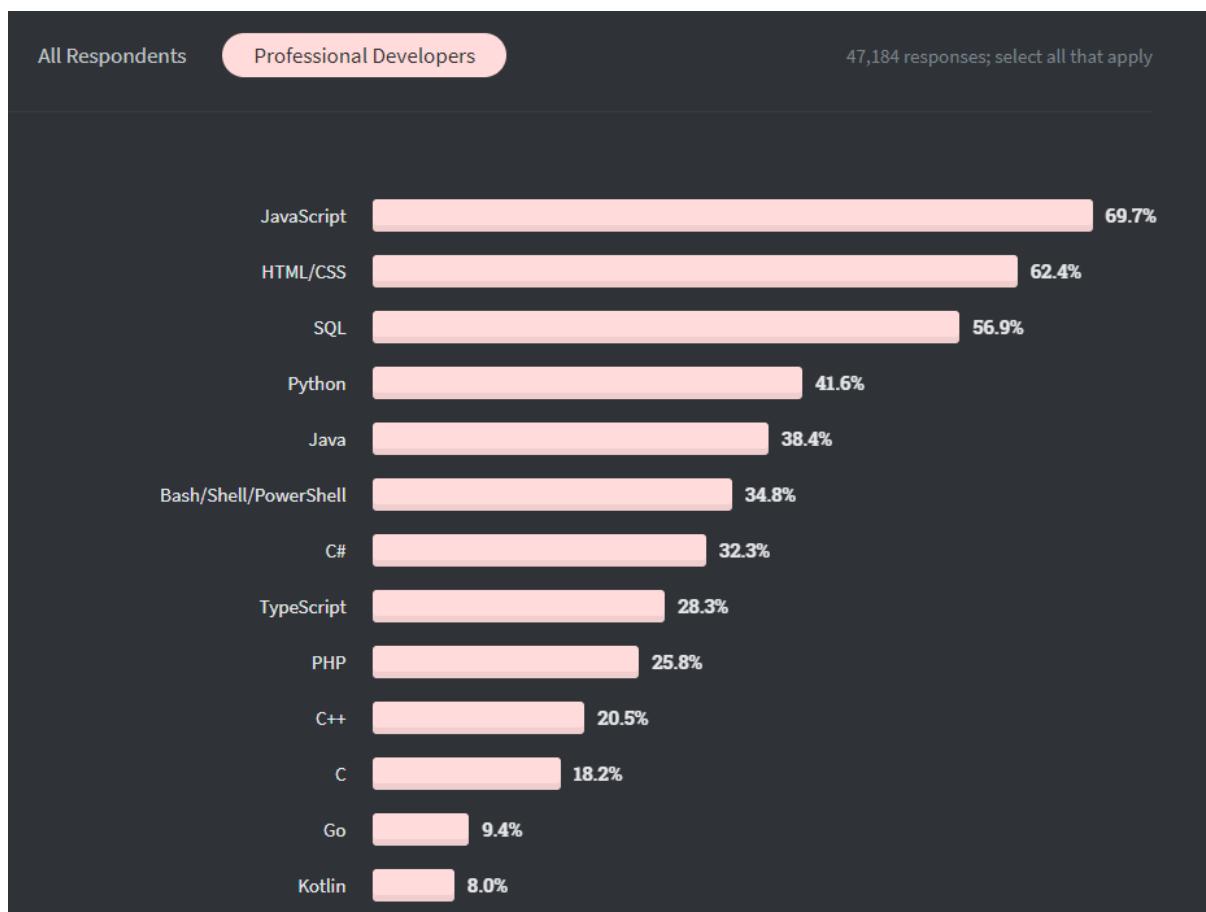
Menurut tingkat kedekatannya dengan mesin komputer, bahasa pemrograman terdiri dari:

- Bahasa Mesin, yaitu bahasa yang digunakan untuk memberikan instruksi kepada komputer dengan memakai kode bahasa biner, contohnya 01100101100110
- Bahasa Tingkat Rendah, atau dikenal dengan istilah bahasa rakitan (Assembly), yaitu bahasa yang digunakan untuk memberikan instruksi kepada komputer memakai kode-kode singkat (kode mnemonic), contohnya adalah: MOV, SUB, CMP, JMP, JGE, JL, LOOP, dsb.
- Bahasa Tingkat Menengah, yaitu bahasa komputer yang memakai campuran instruksi dalam kata-kata bahasa manusia dan instruksi yang bersifat simbolik, contohnya {}, ?, <<, >>, &&, || dan sebagainya.
- Bahasa Tingkat Tinggi, yaitu bahasa komputer yang memakai instruksi berasal dari unsur kata-kata bahasa manusia, contohnya begin, end, if, for, while, and, or dan sebagainya.

Sedangkan menurut generasinya, bahasa pemrograman diklasifikasikan menjadi lima generasi, yaitu:

- Bahasa Generasi Pertama yaitu bahasa mesin
- Bahasa Generasi Kedua yaitu bahasa rakitan
- Bahasa Generasi Ketiga yaitu high-level language
- Bahasa Generasi Keempat yaitu 4 GL (fourth-generation language)
- Bahasa Generasi Kelima yaitu Programming Language Based Object Oriented & Web Development.

Adapun bahasa pemrograman yang paling populer berdasarkan survey yang dilakukan oleh salah satu web yang digunakan para developers seluruh dunia untuk saling berkolaborasi dalam pemecahan masalah yang ditemui didalam dunia pemrograman yaitu stackoverflow.com, diantaranya adalah:



Sumber: <https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers>

Dari sumber di atas, kita akan coba bahas beberapa di antaranya, yaitu:

## 1. JavaScript

JavaScript diciptakan oleh Brendan Eich (perusahaan Netscape) dengan nama awal berupa LiveScript. JavaScript adalah bahasa pemrograman yang biasa diletakkan bersama kode HTML untuk menentukan satu tindakan. Seperti pada sebuah halaman web, JavaScript dapat membuat halaman web tersebut tampil secara dinamis, misalkan menampilkan perubahan setiap detik pada sebuah jam.

Selain sebagai bahasa pemrograman dari sisi klien pada halaman web, saat ini JavaScript juga dapat digunakan pada sisi server. Bahkan JavaScript dapat digunakan untuk membuat aplikasi berbasis mobile yang dapat dijalankan pada sistem operasi Android maupun IOS.

Contoh *source code* yang dituliskan dengan JavaScript:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Callbacks</h2>

<p>Do a calculation and then display the result.</p>

<p id="demo"></p>

<script>
function myDisplayer(some) {
    document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
    let sum = num1 + num2;
    myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
</script>

</body>
</html>
```

Sumber :

[https://www.w3schools.com/js/tryit.asp?filename=tryjs\\_callback4](https://www.w3schools.com/js/tryit.asp?filename=tryjs_callback4)

## 2. Python

Python adalah bahasa pemrograman yang dibuat oleh Guido van Rossum dan rilis pada tahun 1991. Python banyak digunakan untuk membuat berbagai macam program aplikasi yaitu, CLI (Command Line Interface), Program GUI (Desktop0, Aplikasi Mobile, Web, IoT, Game, Program untuk Hacking dan sebagainya. Contoh *source code* yang dituliskan dengan bahasa pemrograman python adalah sebagai berikut:

```
def my_function():
    print("Hello from a function")

my_function()
```

Sumber :

[https://www.w3schools.com/python/trypython.asp?filename=demo\\_function](https://www.w3schools.com/python/trypython.asp?filename=demo_function)

### 3. Java

Java adalah sebuah bahasa pemrograman yang dibuat pada tahun 1995 oleh James Gosling saat masih bergabung di Sun Microsystems, yang saat ini merupakan bagian dari Oracle. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin level bawah yang minimal. Bahasa pemrograman Java dapat digunakan untuk:

- Aplikasi mobile (khususnya aplikasi android)
- Aplikasi Desktop
- Aplikasi Web
- Server Web dan aplikasi server
- Games
- Koneksi database

Contoh *source code* yang dituliskan dengan bahasa pemrograman Java adalah sebagai berikut:

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Sumber:

[https://www.w3schools.com/java/tryjava.asp?filename=demo\\_helloworld](https://www.w3schools.com/java/tryjava.asp?filename=demo_helloworld)

### 4. TypeScript

TypeScript adalah sebuah bahasa pemrograman yang dikembangkan dan dimaintenance oleh Microsoft. TypeScript dipublish pada Oktober 2012. TypeScript adalah turunan dari JavaScript, tetapi memiliki syntax yang lebih strict atau ketat. Selain itu TypeScript juga menambahkan pilihan static typing di dalam bahasa pemrogramannya. Dikarenakan

TypeScript adalah turunan dari JavaScript, maka program JavaScript yang tersedia juga valid terhadap program-program TypeScript. Contoh penulisan TypeScript adalah sebagai berikut:

```
// How code flows inside our JavaScript files can affect
// the types throughout our programs.

const users = [{ name: "Ahmed" }, { name: "Gemma" }, { name: "Jon" }];

// We're going to look to see if we can find a user named "jon".
const jon = users.find((u) => u.name === "jon");

// In the above case, 'find' could fail. In that case we
// don't have an object. This creates the type:
//
//   { name: string } | undefined
//
// If you hover your mouse over the three following uses of 'jon' below,
// you'll see how the types change depending on where the word is located:

if (jon) {
| jon;
} else {
| jon;
}
```

Sumber :

<https://www.typescriptlang.org/play?strictNullChecks=true&q=424#example/code-flow>

## 5. C#

C# atau C sharp adalah bahasa pemrograman yang dikembangkan sekitar tahun 2000 oleh Microsoft sebagai bagian pengembangan .NET yang kemudian diapproved sebagai sebuah standar internasional oleh Ecma dengan nomer ECMA-334 pada tahun 2002. C# di desain oleh Anders Hejlsberg. Dengan bahasa pemrograman C#, kita dapat membuat diantaranya:

- Aplikasi client berbasis windows
- Komponen dan pustaka windows
- Service windows
- Aplikasi Web
- Web Service dan Web API
- Aplikasi Mobile untuk IOS dan Android

Contoh source code yang dituliskan dengan C# adalah sebagai berikut:

```
using System;
namespace CSharpStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            // Define .NET Strings
            // String of characters
            System.String authorName = "Mahesh Chand";

            // String made of an Integer
            System.String age = "33";

            // String made of a double
            System.String numberString = "33.23";

            // Write to Console.
            Console.WriteLine("Name: {0}", authorName);
            Console.WriteLine("Age: {0}", age);
            Console.WriteLine("Number: {0}", numberString);
            Console.ReadKey();
        }
    }
}
```

Sumber : <https://www.c-sharpcorner.com/article/what-is-c-sharp/>

## 6. PHP

PHP adalah bahasa pemrograman yang dibangun khusus untuk pengembangan web. PHP dibuat oleh seorang programmer bernama Rasmus Lerdorf pada tahun 1994. PHP awalnya adalah singkatan dari Personal Home Page, tetapi saat ini adalah kepanjangan dari PHP: Hypertext Preprocessor. Source code PHP diproses pada sebuah web server oleh sebuah interpreter PHP. Interpreter standar PHP yang didanai oleh Zend Engine adalah sebuah software gratis yang dirilis dibawah lisensi PHP. PHP telah digunakan secara luas dan bisa dideploy pada sebagian besar web server pada hampir semua sistem operasi dan platform, secara gratis. Contoh penulisan source code PHP adalah sebagai berikut:

```

<?php

class foo {
    var $bar = 'I am bar.';
    var $arr = array('I am A.', 'I am B.', 'I am C.');
    var $r   = 'I am r.';
}

$foo = new foo();
$bar = 'bar';
$baz = array('foo', 'bar', 'baz', 'quux');
echo $foo->$bar . "\n";
echo $foo->{$baz[1]} . "\n";

$start = 'b';
$end   = 'ar';
echo $foo->{$start . $end} . "\n";

$arr = 'arr';
echo $foo->{$arr[1]} . "\n";

?>

```

Sumber:

<https://www.php.net/manual/en/language.variables.variable.php>

## 7. C++

C++ adalah sebuah bahasa pemrograman general-purpose yang dibuat oleh Bjarne Stroustrup sebagai sebuah ekstensi atau kelanjutan dari bahasa pemrograman C. Pada tahun 1979 Bjarne Stroustrup memulainya dengan nama “C with Classes”, karena merupakan superset dari bahasa C. “C with Classes” menambahkan konsep Object-oriented di dalamnya. Pada tahun 1983, namanya berganti menjadi C++. Bahasa C++ telah berkembang secara signifikan dari waktu ke waktu, dan C++ modern telah memiliki object-oriented, generic da fitur functional di dalam tambahan pada fasilitas untuk manipulasi low-level memory. Ini hampir selalu diimplementasikan sebagai sebuah bahasa compile. Banyak vendor menyediakan kompiler C++, termasuk Free Software Foundation, LLVM, Microsoft, Intel, Oracle dan IBM, sehingga tersedia di dalam banyak platform. C++ dapat digunakan untuk membuat sistem operasi, browser, games dan masih banyak yang lainnya. Adapun contoh *source code* yang dituliskan dengan bahasa pemrograman C++

adalah sebagai berikut:

```
#include <iostream>

int main()

{
    std::cout << "Welcome to C++";

    return 0;
}
```

f

Sumber: <https://qwords.com/blog/contoh-program-c/>

## 8. C

Bahasa pemrograman C adalah sebuah bahasa pemrograman general-purpose yang dibuat pada tahun 1972 oleh Dennis Ritchie untuk sistem operasi Unix di Bell Telephone Laboratories. C banyak dipakai oleh berbagai jenis platform sistem operasi dan arsitektur komputer. Selain itu C juga dibuat untuk jaringan komputer dan pengembangan software aplikasi. Banyak bahasa pemrograman yang merupakan turunan dari bahasa C, contohnya C++. Contoh *Source code* yang dituliskan dengan bahasa C adalah sebagai berikut

```
#include <stdio.h>
int main(void) {
    printf("Hello, World!\n");
    return 0;
}
```

Sumber: [https://id.wikipedia.org/wiki/C\\_\(bahasa\\_pemrograman\)](https://id.wikipedia.org/wiki/C_(bahasa_pemrograman))

## 9. Shell

Shell adalah sebuah program yang menjembatani user (pengguna) dengan sistem operasi dalam hal ini kernel (inti sistem operasi). Umumnya shell menyediakan prompt sebagai user interface, tempat di

mana user mengetikkan perintah-perintah yang diinginkan. Bahasa pemrograman yang digunakan untuk menuliskan perintah-perintah tersebut adalah Shell script. Sebuah shell script di desain untuk dijalankan pada Unix Shell. Operasi yang dapat dilakukan shell scripts termasuk di antaranya adalah manipulasi file, eksekusi program dan menampilkan text. Sebuah script yang digunakan untuk mensest up environment, menjalankan program, dan melakukan beberapa pembersihan yang penting, pencatatan (logging) dan semacamnya dikenal dengan nama sebuah **wrapper**.

Perangkat Unix standar, sed dan awk menyediakan kemampuan ekstra untuk programming shell. Perl juga bisa di sertakan di dalam shell scripts seperti halnya bahasa script lainnya seperti Tcl. Contoh penulisan source code shell script adalah sebagai berikut:

```
#!/bin/sh
printf 'compiling...\n'
cc -c foo.c
cc -c bar.c
cc -c qux.c
cc -o myprog foo.o bar.o qux.o
printf 'done.\n'
```

Sumber: [https://en.wikipedia.org/wiki/Shell\\_script](https://en.wikipedia.org/wiki/Shell_script)

## 10. Kotlin

Kotlin adalah sebuah bahasa pemrograman yang cross platform serta general purpose. Kotlin dirancang untuk beroperasi penuh dengan bahasa pemrograman Java. Versi Java Virtual Machine (JVM) dari pustaka standar Kotlin bergantung pada kelas pustaka Java. Tetapi type inference memungkinkan sintaksnya menjadi lebih ringkas. Kotlin secara utama menargetkan JVM, tetapi juga mengkompilasinya ke JavaScript.

Kotlin diluncurkan pada tahun 2011 oleh JetBrains. Nama Kotlin berasal dari nama pulau Kotlin dekat St. Petersburg. Kotlin v1.0 dirilis pada 15 Februari 2016. Versi ini dianggap sebagai rilis stabil pertama yang resmi. Pada konferensi Google I/O tahun 2017, Google mengumumkan dukungan yang sangat penting untuk Kotlin di Android. Pada 7 Mei 2019, Google mengumumkan bahwa bahasa pemrograman Kotlin kini menjadi bahasa pilihan bagi para pengembang aplikasi

Android. Kotlin dapat digunakan pada banyak pengembangan aplikasi, baik sebagai server-side, client-side web dan Android. Contoh penulisan *source code* untuk bahasa pemrograman Kotlin adalah sebagai berikut:

```
fun foo(  
    bar: Int = 0,  
    baz: Int = 1,  
    qux: () -> Unit,  
) { /*...*/ }  
  
foo(1) { println("hello") }      // Uses the default value baz = 1  
foo(qux = { println("hello") }) // Uses both default values bar = 0 and baz = 1  
foo { println("hello") }        // Uses both default values bar = 0 and baz = 1
```

Sumber: <https://kotlinlang.org/docs/reference/functions.html>

## B. Pengertian Running, Debugging, Compiling, Interpreting dan Pembuatan Executable file

*Source code* yang telah dituliskan dalam bahasa pemrogramannya masing-masing memerlukan beberapa tahap lagi agar *script code* tersebut dapat dipahami oleh perangkat komputer, dan dapat dijalankan sesuai dengan maksud dari instruksi yang dituliskan di dalam *source code* tersebut. Beberapa bahasa pemrograman memiliki cara yang berbeda dalam prosesnya tersebut. Dalam hal ini ada dua cara bagaimana sebuah *source code* sebuah bahasa pemrograman diproses agar dapat dipahami oleh komputer yang menjalankannya, yaitu:

### 1. Compiling

Proses compiling adalah proses mengubah *source code* yang telah kita tuliskan di dalam text editor ataupun pada perangkat IDE (Integrated Development Environment) ke dalam bentuk program dalam bahasa assembly untuk kemudian dapat dijalankan pada komputer target. Untuk proses compiling ini membutuhkan sebuah perangkat lunak yang dinamakan compiler. Pada proses ini compiler akan menghasilkan file yang perangkat lunak yang siap dijalankan pada komputer sesuai sistem operasi yang diperlukan. Jika dicompiling untuk target komputer dengan sistem operasi Windows maka hasilnya adalah Executable file dengan ekstensi .exe, dan jika di compiling untuk target komputer dengan sistem operasi Ubuntu (Linux) akan menghasilkan Executable file dengan ekstensi .deb. Adapun Executable file ini adalah file yang dihasilkan dari proses compiling dan siap untuk diinstall atau

dieksekusi pada komputer dengan platform sistem operasi yang sesuai. Adapun dari 10 bahasa pemrograman yang telah dijelaskan di atas, bahasa-bahasa pemrograman yang membutuhkan proses compiling untuk dapat dijalankan adalah:

- Python
- Java
- C#
- PHP

Berbeda dengan hasil compiling dari bahasa pemrograman lainnya, pada PHP yang dihasilkan dari proses compiling bukanlah executable code/file, tetapi dalam bentuk serangkaian kode operasi yang umumnya dikenal sebagai “opcodes”. Yang kemudian akan diproses oleh interpreter.

- C++
- C
- Kotlin

## 2. Interpreting

Berbeda dengan compiling, pada proses interpreting setiap instruksi pada *source code* diterjemahkan secara langsung tanpa terlebih dahulu menyusunnya menjadi program bahasa mesin. Perangkat lunak yang melakukan tugas interpreting dinamakan Interpreter. Interpreter secara umum menggunakan salah satu strategi berikut untuk menjalankan program yaitu:

- a) Mengeksekusi *source code* secara langsung, atau
- b) Menerjemahkannya ke dalam serangkaian p-code kemudian mengekuskusinya, atau
- c) Mengeksekusi kode yang telah dikompilasi sebelumnya oleh compiler yang merupakan bagian dari sistem penerjemahan.

Dari 10 bahasa pemrograman yang telah dijelaskan sebelumnya, bahasa-bahasa pemrograman yang membutuhkan proses interpreting adalah :

- JavaScript
- Python
- TypeScript
- PHP
- Shell

Secara garis besar, perbedaan Compiler dan Interpreter adalah sebagai berikut :

Compiler	Interpreter
Memindai seluruh program dan menerjemahkannya secara keseluruhan ke dalam kode mesin.	Menerjemahkan satu pernyataan satu per satu.
Dibutuhkan banyak waktu untuk menganalisis kode sumber namun keseluruhan waktu eksekusi relatif lebih cepat.	Dibutuhkan sedikit waktu untuk menganalisis kode sumber namun keseluruhan waktu eksekusi lebih lambat.
Menghasilkan kode objek menengah yang membutuhkan tautan lebih lanjut, sehingga membutuhkan lebih banyak memori.	Tidak ada kode objek antara yang dihasilkan, maka memori menjadi efisien.
Ini menghasilkan pesan kesalahan hanya setelah memindai seluruh program.	Terus menerjemahkan program sampai kesalahan pertama terpenuhi, dalam hal ini berhenti. Oleh karena itu debugging mudah dilakukan.
Bahasa pemrograman seperti C, C ++ menggunakan kompiler.	Bahasa pemrograman seperti Python, Ruby menggunakan interpreter.

Sumber: [https://medium.com/@larasn/\\_mengenal-compiler-dan-interpreter-30610c6df554](https://medium.com/@larasn/_mengenal-compiler-dan-interpreter-30610c6df554)

Selain istilah compiling dan interpreting, dalam proses pembuatan perangkat lunak, juga dikenal istilah *running* dan *debugging*. *Running* adalah istilah yang digunakan ketika user memerintahkan komputer untuk mengeksekusi sebuah *source code* atau program. Ketika *source code* di-run, maka jika diperlukan proses compiling, maka *source code* tersebut akan dicompile lalu dijalankan sehingga terlihat hasil program tersebut. Dalam penggunaannya, pengguna akan men-*run* program dan sistem akan mengeksekusi program tersebut.

Sedangkan *debugging* adalah proses untuk mencari dan menyelesaikan bugs (masalah yang menyebabkan kesalahan operasi) di dalam program komputer, perangkat lunak ataupun sistem. Banyak bahasa pemrograman dan tools pengembangan software juga menyediakan program untuk membantu di dalam proses *debugging*, program tersebut dikenal dengan nama *debuggers*.

### C. Perangkat Aplikasi dan Cara Melakukan Eksekusi Source Code

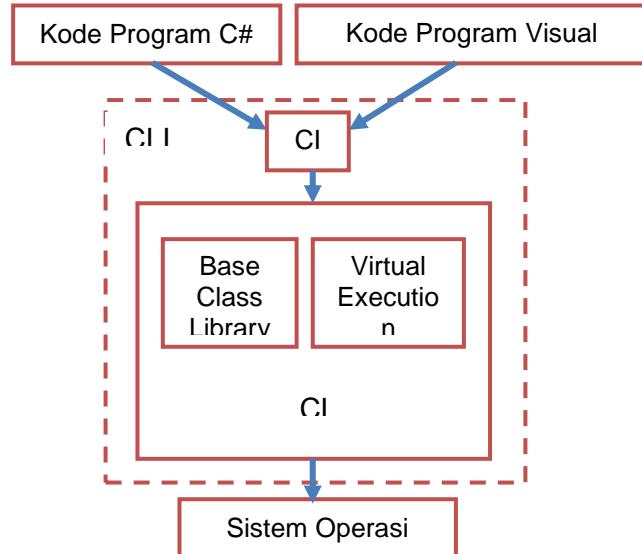
Setiap bahasa pemrograman memerlukan perangkat aplikasi yang berbeda untuk melakukan eksekusi kode programnya. Begitu pula cara melakukan eksekusi kode program untuk setiap bahasa pemrograman tersebut. Selain itu, perangkat aplikasi dan cara melakukan eksekusi juga akan berbeda bila bahasa pemrograman tersebut digunakan untuk tujuan

yang berbeda. Pada modul ini akan mengambil beberapa contoh bahasa pemrograman dan beberapa contoh kasus penggunaan bahasa pemrograman terkait perangkat aplikasi yang dibutuhkan serta cara melakukan eksekusi kode programnya. Contoh bahasa pemrograman tersebut adalah Kotlin, C#, PHP dan JavaScript.

## **1. C# sebagai Bahasa Pemrograman Aplikasi Desktop**

Bahasa Pemrograman C# adalah bahasa pemrograman berorientasi objek yang digunakan untuk mengembangkan aplikasi yang berjalan di ekosistem *.NET Framework* yaitu perangkat lunak yang digunakan untuk mengembangkan sistem operasi *Windows* atau *Web Service* berbasis *Windows*. Karena dapat dijalankan pada ekosistem *.NET Framework*, Bahasa Pemrograman C# disebut juga dengan *C#.NET*. Perangkat lunak ini ter-install secara *default* pada *Windows 7* ke atas. Sehingga, Bahasa Pemrograman C# umumnya digunakan untuk mengembangkan aplikasi desktop berbasis *Windows*. C# juga digunakan sebagai bahasa *scripting* untuk perangkat *Game Engine Unity*.

Di dalam ekosistem *.NET Framework* terdapat *Runtime Environment* yang disebut *Common Language Runtime (CLR)*. Sama halnya dengan *JRE (Java Runtime Environment)* pada ekosistem *Java*, *CLR* berperan untuk menjalankan kode program yang dilalui oleh *.NET Framework*. Sebelum dijalankan pada *CLR*, kode program yang melalui *.NET Framework* akan dikonversi ke dalam *Common Intermediate Languege (CIL)* yang dapat langsung dijalankan oleh *CLR*. *CIL* dan *CLR* terangkum dalam *Common Language Infrastructure (CLI)* yang memungkinkan *.NET Framework* untuk menjalankan kode program dari beberapa bahasa pemrograman lainnya seperti *Visual Basic*, *F#*, dan *Python*.

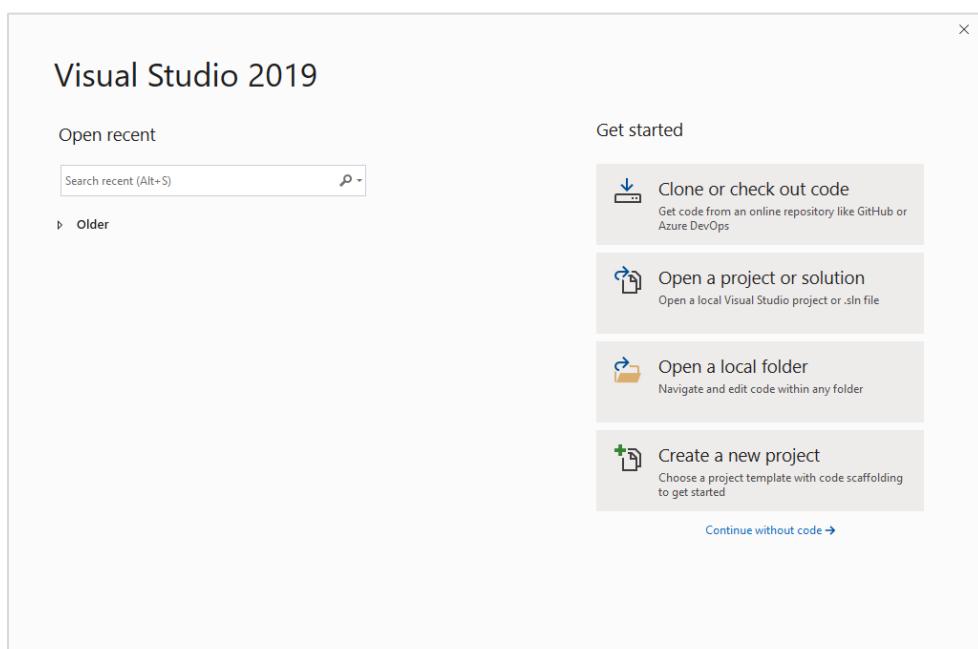


Gambar 1.1 Ilustrasi eksekusi C# dengan ekosistem .NET Framework

Perangkat lunak *IDE Visual Studio* dapat digunakan untuk membuat dan menjalankan aplikasi *desktop* berbasis *Windows* yang ditulis menggunakan Bahasa Pemrograman C# dengan ekosistem .NET Framework. Umumnya, aplikasi *desktop* yang dikembangkan adalah aplikasi *Console* yang berjalan di dalam *Command Line* dan aplikasi *Windows Form* berupa aplikasi yang memiliki *GUI* berupa *window*.

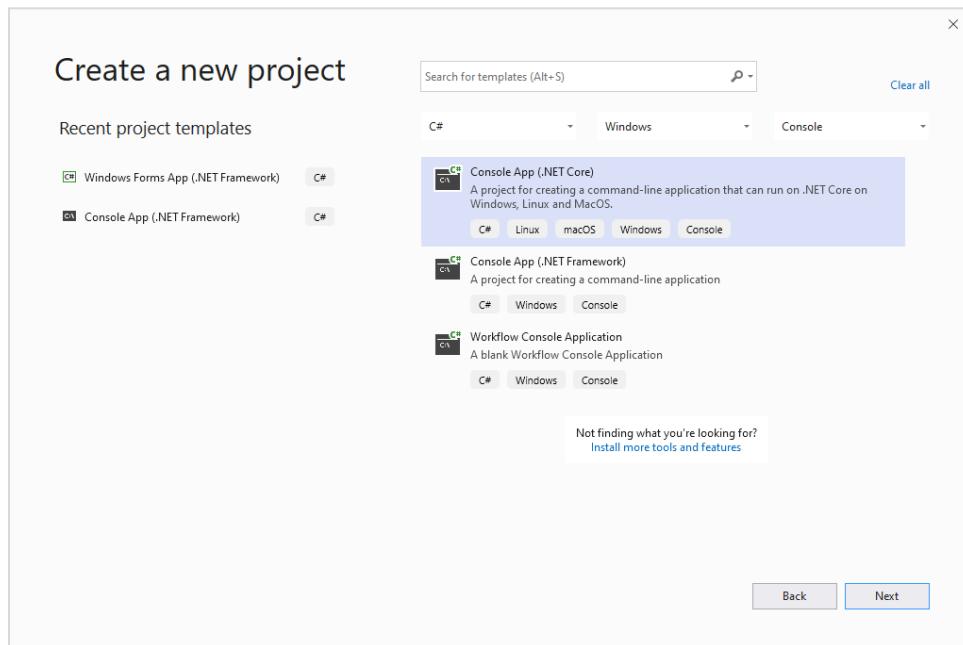
Berikut adalah langkah-langkah untuk membuat *project* baru aplikasi *Console* dengan Bahasa Pemrograman C# pada *Visual Studio* (pada modul ini menggunakan *Visual Studio 2019 Community Edition*).

- 1) Jalankan *Visual Studio* maka muncul *window* berikut. Pilih ‘Create a new project’ untuk memulai membuat *project* baru.



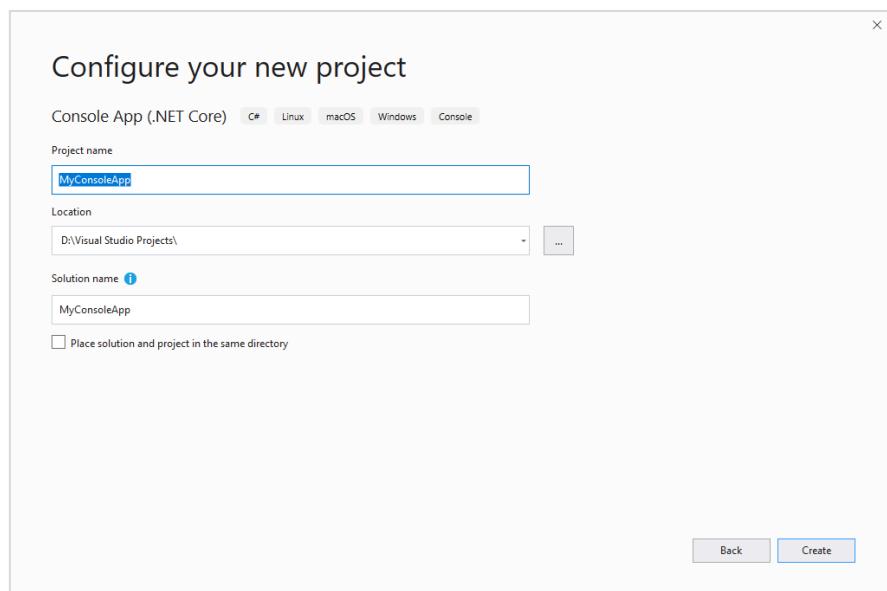
Gambar 1.2 Window awal Visual Studio

- 2) Pilih jenis aplikasi yang akan dikembangkan yaitu ‘Console App’. Pastikan urutan pilihan untuk tiga *dropdown* adalah ‘C#’, ‘Windows’, dan ‘Console’. Kombinasi pilihan ini digunakan untuk membuat *project* aplikasi *Console* yang berjalan di sistem operasi *Windows* yang dikembangkan dengan Bahasa Pemrograman C#. Selanjutnya tekan tombol ‘Next’.



Gambar 1.3 Memilih tipe project aplikasi Console

- 3) Lakukan konfigurasi *project*. Konfigurasi yang dapat dilakukan di antaranya:
- Mengubah nama *project*
  - Mengubah lokasi penyimpanan *project*
  - Mengubah nama *solution*

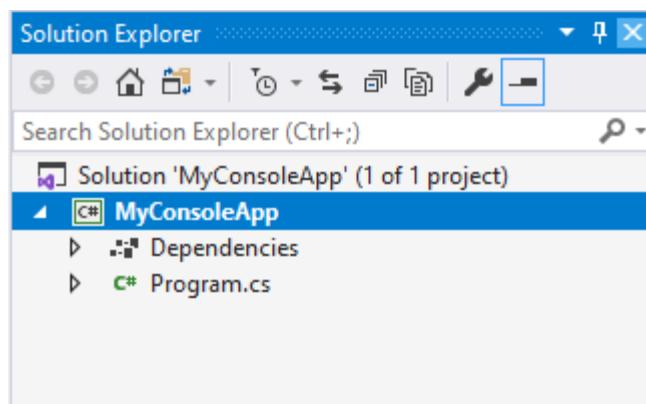


Gambar 1.4 Konfigurasi project aplikasi Console

Di dalam sebuah *solution* dapat berisi lebih dari sebuah *project*. Bila di dalam sebuah *solution* hanya terdiri dari sebuah *project*, maka nama *solution* akan sama dengan nama *project* di dalamnya. Umumnya, lebih dari sebuah *project* dibutuhkan di dalam sebuah *solution* untuk mengatasi permasalahan yang diatasi dengan aplikasi *Client Server* yang membutuhkan lebih dari sebuah aplikasi untuk menjalankan fungsinya.

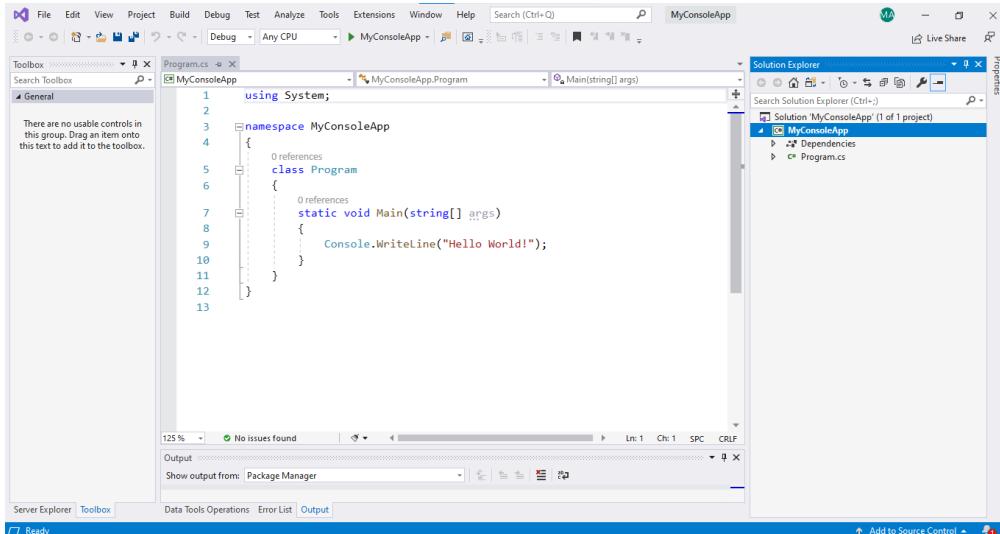
Bila sudah selesai melakukan konfigurasi *project*, tekan tombol ‘Create’ untuk membuat *project baru* aplikasi *Console* dengan C#.

*Visual Studio* secara otomatis menyiapkan sumber daya utama yang dibutuhkan dalam mengembangkan aplikasi *Console*. Di dalam *Visual Studio* akan terbentuk struktur *solution* yang di dalamnya terdapat *project* yang berisi file ‘Program.cs’ sebagai file kode program utama dalam mengembangkan aplikasi *Console*. Struktur *solution* ini dapat dilihat pada *Solution Explorer pane*.



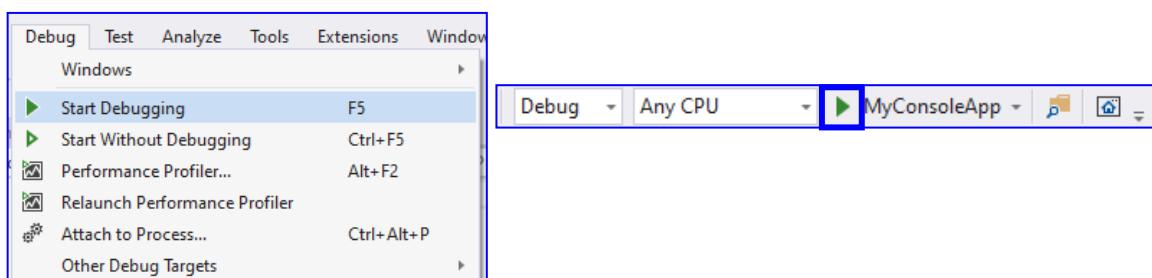
Gambar 1.5 Struktur solution pada project aplikasi Console

File ‘Program.cs’ berisi kode program yang ditulis menggunakan Bahasa Pemrograman C#. Secara default, *Visual Studio* akan menyediakan kode program untuk menampilkan teks ‘Hello World!’ yang ditempatkan di dalam *method* ‘Main()’. *Class* pada C# berada di dalam sebuah *namespace* yang secara default memiliki nama sama dengan *project*.



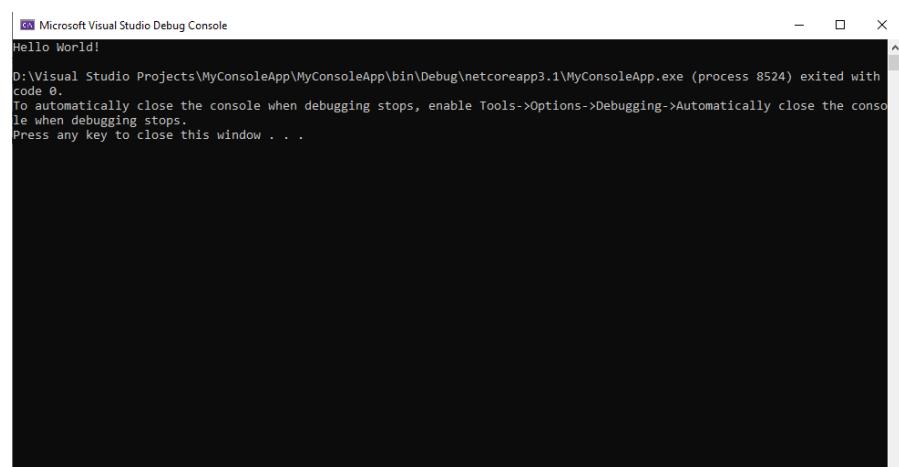
Gambar 1.6 Tampilan awal Visual Studio setelah project pertama kali dibuat

Menjalankan kode program dengan *Visual Studio*, otomatis melakukan proses *Debugging*. Untuk memulainya pilih menu *Debug* -> *Start Debugging*, atau tekan tombol dengan icon bergambar segitiga hijau yang menghadap ke kiri, atau tekan tombol 'F5' pada *Keyboard*.



Gambar 1.7 Akses fitur Debugging Visual Studio

Proses *Debugging* akan melakukan proses *Build* yang akan menghasilkan *executable file* dengan format '*.exe*'. *File* ini yang akan dijalankan pada proses *Debugging*.

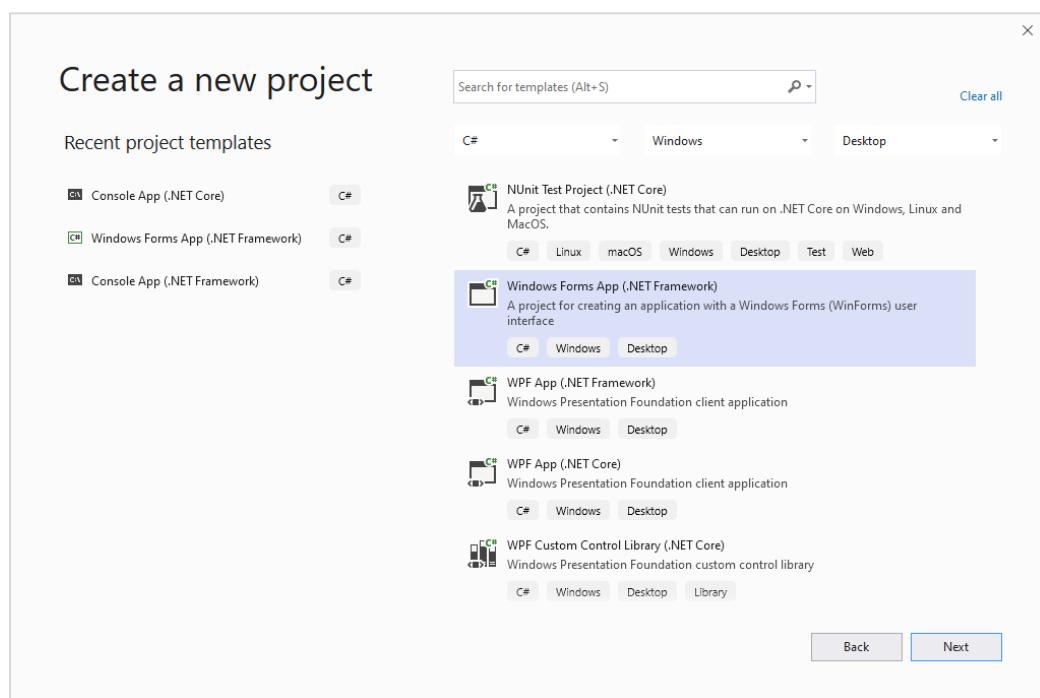


Gambar 8 Hasil eksekusi executable file aplikasi Console

Aplikasi *Console* biasanya digunakan untuk menjalankan serangkaian perintah kode program yang dieksekusi 1 kali, atau aplikasi yang menjalankan proses *service* secara *background*. *Command Prompt* hanya digunakan untuk menampilkan pesan proses apa saja yang telah dilakukan oleh aplikasi *Console* tersebut.

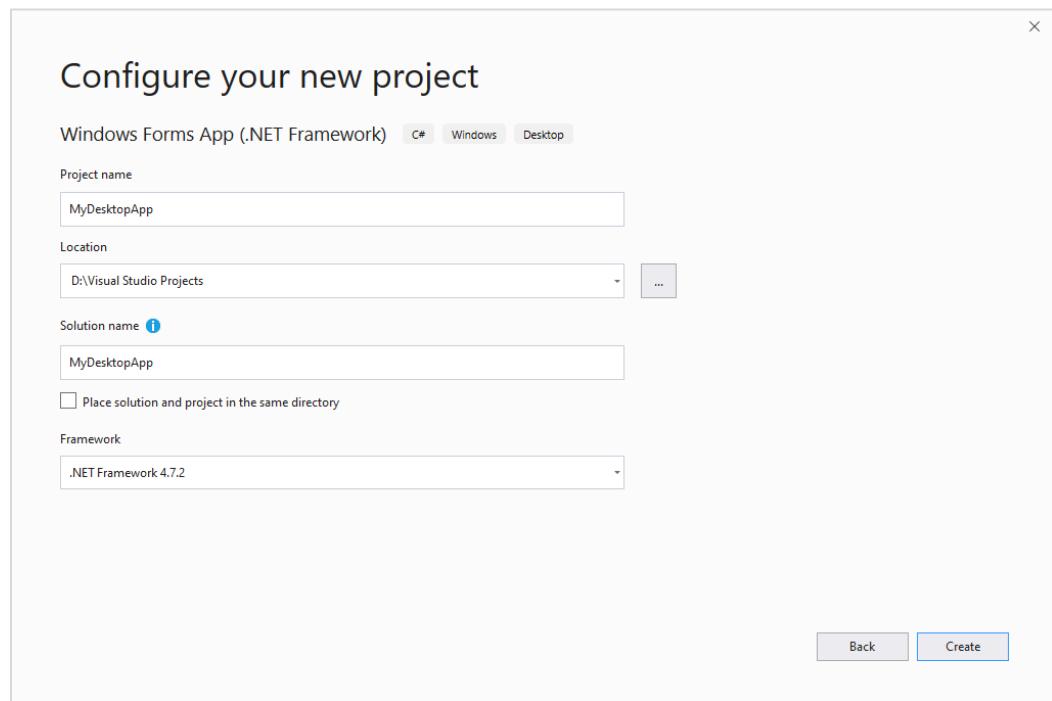
Selain aplikasi *Console*, *Visual Studio* juga dapat membuat dan menjalankan aplikasi *Windows Form*. Berikut adalah langkah-langkah untuk membuat project baru aplikasi *Windows Form* dengan Bahasa Pemrograman C# pada *Visual Studio*.

- 1) Jalankan *Visual Studio*, pilih ‘Create a new project’.
- 2) Pilih jenis aplikasi yang akan dikembangkan yaitu ‘Windows Form’. Pastikan urutan pilihan untuk tiga *dropdown* adalah ‘C#’, ‘Windows’, dan ‘Desktop’. Kombinasi pilihan ini digunakan untuk membuat *project* aplikasi *Windows Form* yang berjalan di sistem operasi *Windows* yang dikembangkan dengan Bahasa Pemrograman C#. Selanjutnya tekan tombol ‘Next’.



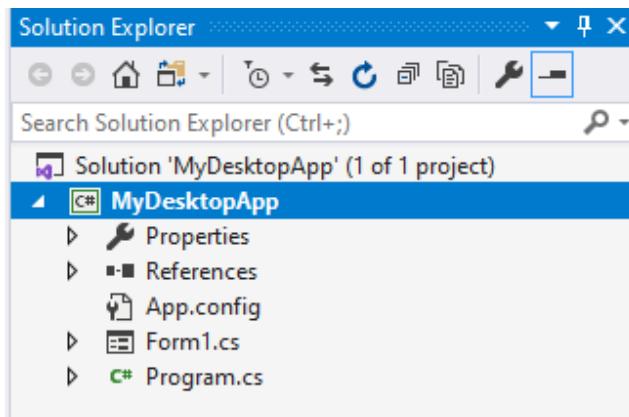
Gambar 1.9 Memilih tipe project aplikasi Windows Form

- 3) Lakukan konfigurasi *project*, lalu tekan ‘Create’. Anda dapat memilih versi *.NET Framework* yang digunakan.



Gambar 1.10 Konfigurasi project aplikasi Windows Form

Sama halnya ketika membuat *project* aplikasi *Console*, *Visual Studio* akan menyiapkan secara otomatis struktur *solution* yang berisi sumber daya *project* aplikasi *Windwos Form*.



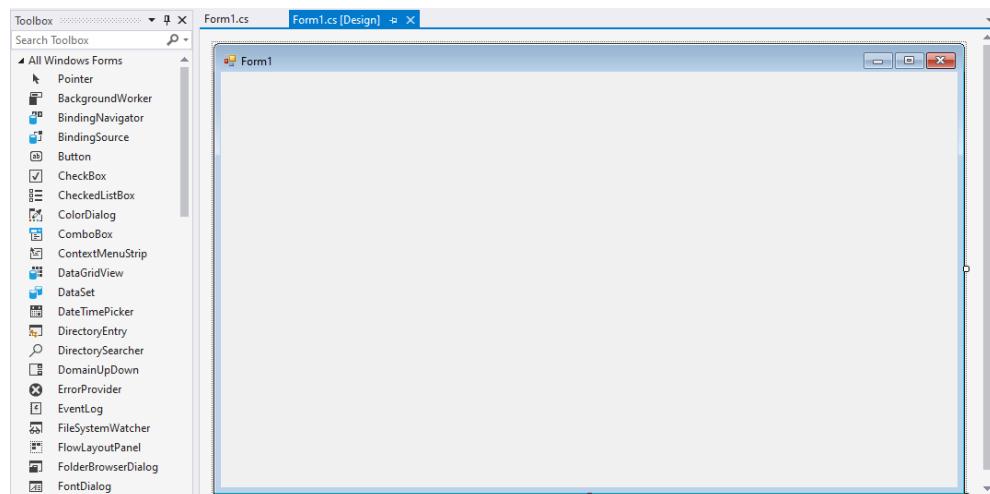
Gambar 1.11 Struktur solution pada project aplikasi Windows Form

Selain *file* utama 'Program.cs', di dalam struktur *solution* aplikasi *Windows Form* terdapat *file* *Windows Form* yang memiliki format *file* '.cs' dengan *icon* bergambar *Window*. *File* *Windows Form* ini terdiri dari mode *Code* untuk menulis kode program *C#* dan mode *Designer* untuk mendesain *User Interface Window* yang dikembangkan.

```
Form1.cs  Form1.cs [Design]
MyDesktopApp  MyDesktopApp.Form1  Form1()
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace MyDesktopApp
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19     }
20 }
```

Gambar 1.12 Mode Code file Windows Form

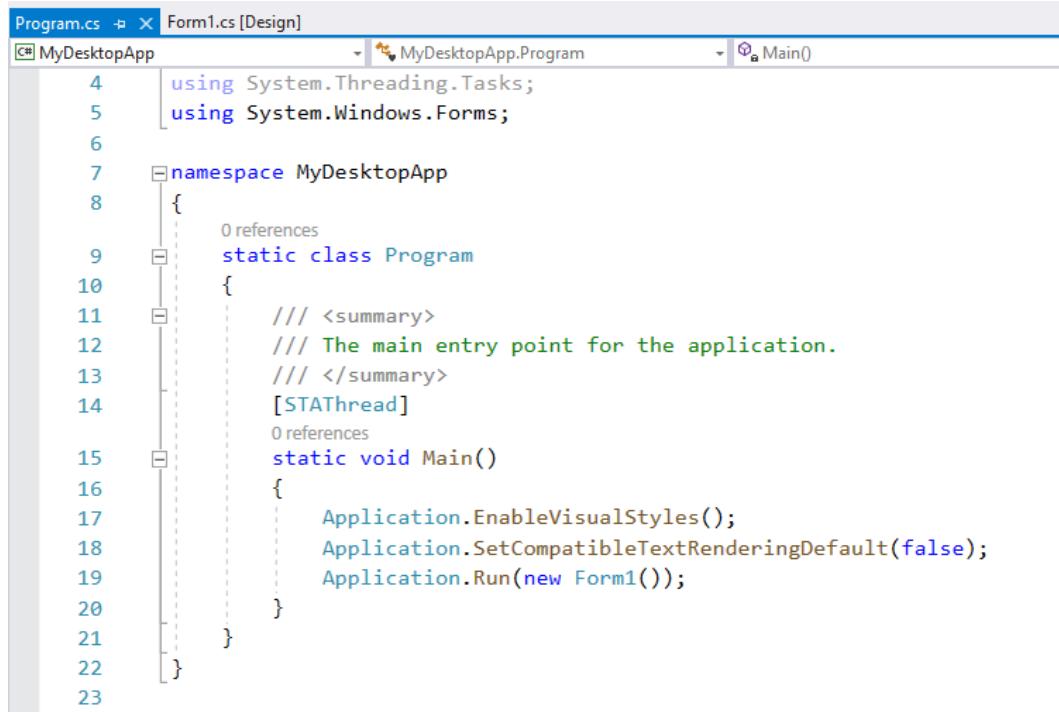
*File Windows Form* berisi sebuah *class* yang mengimplementasikan *class* ‘Form’ dari *namespace* ‘System.Windows.Form’.



Gambar 1.13 Mode Designer file Windows Form

Bila menggunakan mode *Desigener*, pada layar *Visual Studio* akan tampil *Toolbox pane* yang dapat digunakan untuk meletakkan objek *User Interface* pada *Windows Form*.

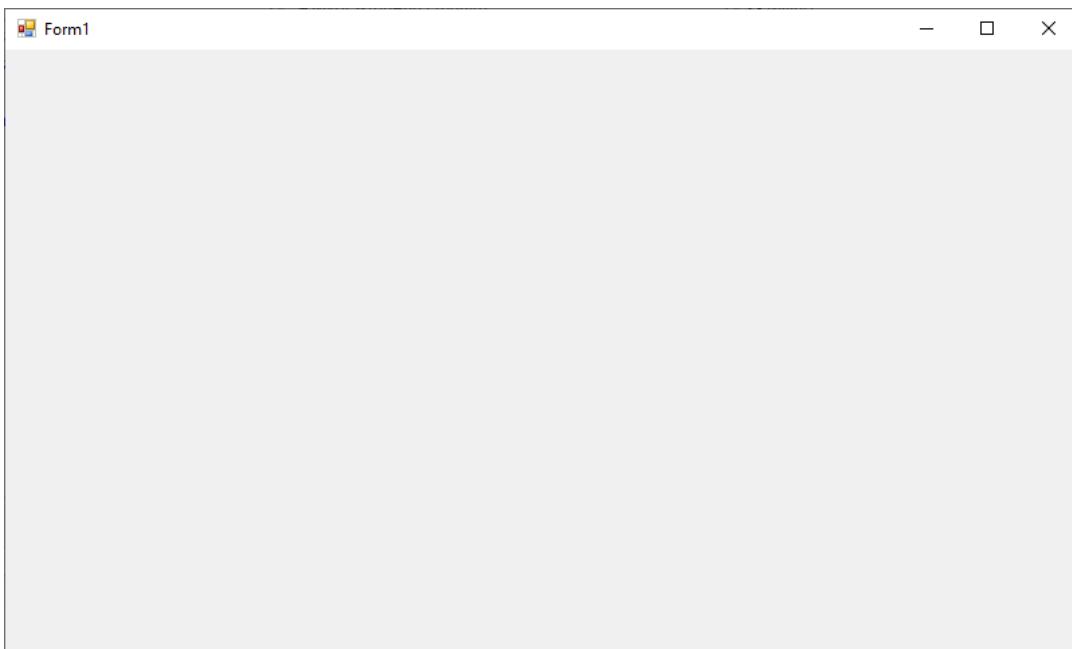
*File* utama ‘Program.cs’ pada *project* aplikasi *Windows Form* berisi konfigurasi *file Windows Form* pertama yang akan dipanggil bila aplikasi dijalankan.



```
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6
7  namespace MyDesktopApp
8  {
9      static class Program
10     {
11         /// <summary>
12         /// The main entry point for the application.
13         /// </summary>
14         [STAThread]
15         static void Main()
16         {
17             Application.EnableVisualStyles();
18             Application.SetCompatibleTextRenderingDefault(false);
19             Application.Run(new Form1());
20         }
21     }
22 }
23
```

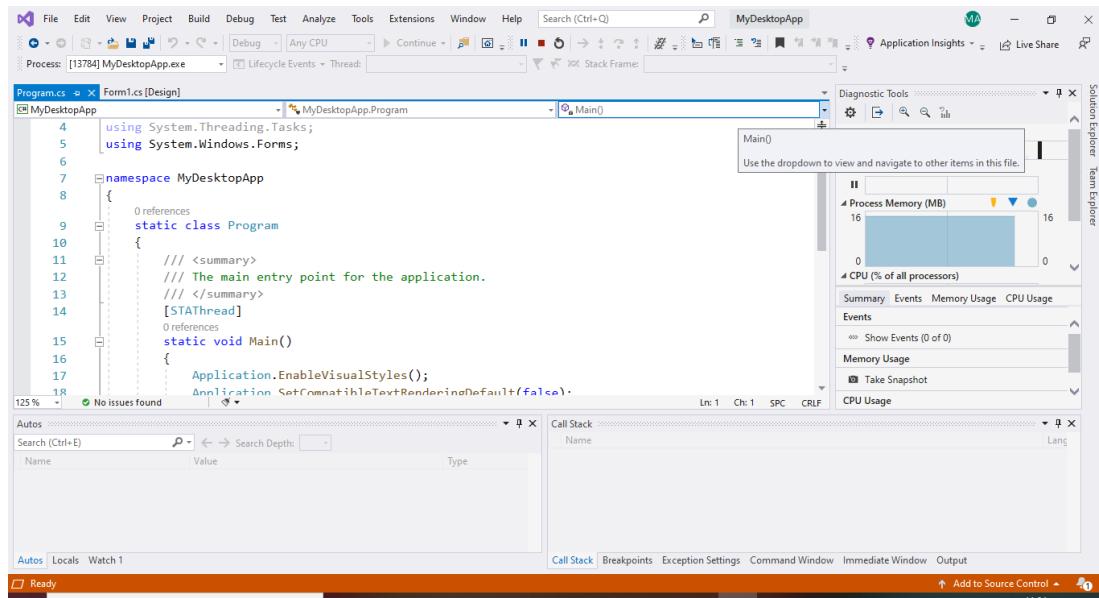
Gambar 1.14 Isi Program.cs project aplikasi Windows Form

Sama halnya dengan aplikasi *Console*, proses *Debugging* pada project aplikasi *Windows Form* akan menghasilkan *executable file* yang otomatis dijalankan pada saat *Debugging*.



Gambar 1.15 Hasil eksekusi executable file aplikasi Windows Form

Pada proses *Debugging*, dapat memantau pergerakan kode program pada layar *Visual Studio*.



Gambar 1.16 Memantau pergerakan kode program

Pesan *Runtime Error* akan ditampilkan pada layar pemantauan kode program tersebut. Pesan *Runtime Error* umumnya berisi jenis kesalahan yang terjadi, konteks kesalahan yang terjadi, baris kode program mana yang mengakibatkan keslahana tersebut terjadi.

## 2. C# sebagai Bahasa Scripting Game Engine Unity

Bahasa Pemrograman C# juga digunakan sebagai bahasa *scripting* pada *game engine Unity*. Perintah *scripting* diterapkan terhadap aksi tingkah laku objek *game*, atau tingkah laku interaksi antar objek *game* pada *Unity*. Dengan *Unity*, dapat mengembangkan aplikasi *multimedia* berupa aplikasi *game*.

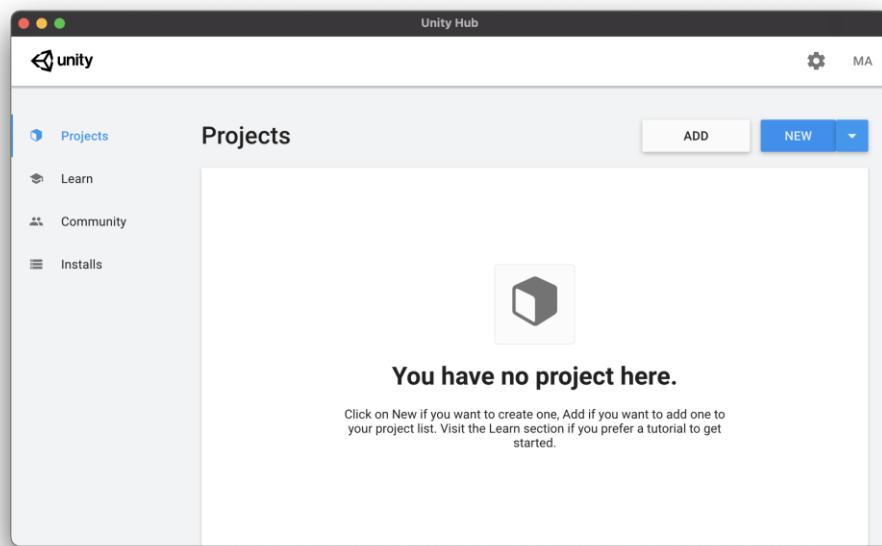
Pada *game engine Unity*, penulisan kode program C# berbeda dengan menulis kode program untuk membuat aplikasi menggunakan *Visual Studio*. Karena sebagian besar kode program untuk menjalankan game sudah disiapkan secara *out of the box* oleh *Unity*. Peran pemrogram adalah menambahkan perintah *scripting* untuk suatu objek *game* pada *Unity*.

Perintah *scripting* yang dituliskan pada *game engine Unity* dijalankan di dalam sebuah *loop* besar yang menerapkan perintahnya setiap *frame* ketika game dijalankan. Sehingga, semakin tinggi *frame rate* yang digunakan, akan lebih sering memanggil perintah *scripting*, yang menghasilkan pengendalian pergerakan objek *game* lebih *responsive*.

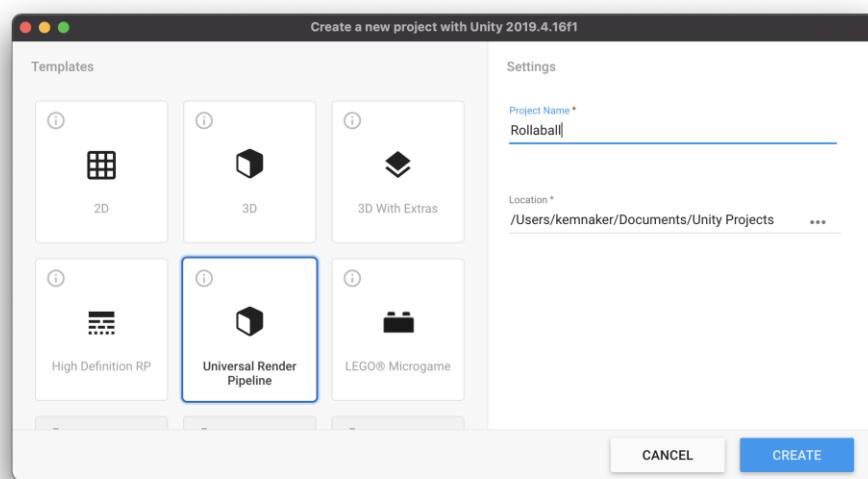
Berikut adalah langkah-langkah untuk menambahkan *scripting* C#

menggunakan *game engine Unity*. (pada modul ini menggunakan *Unity* versi 2019.4)

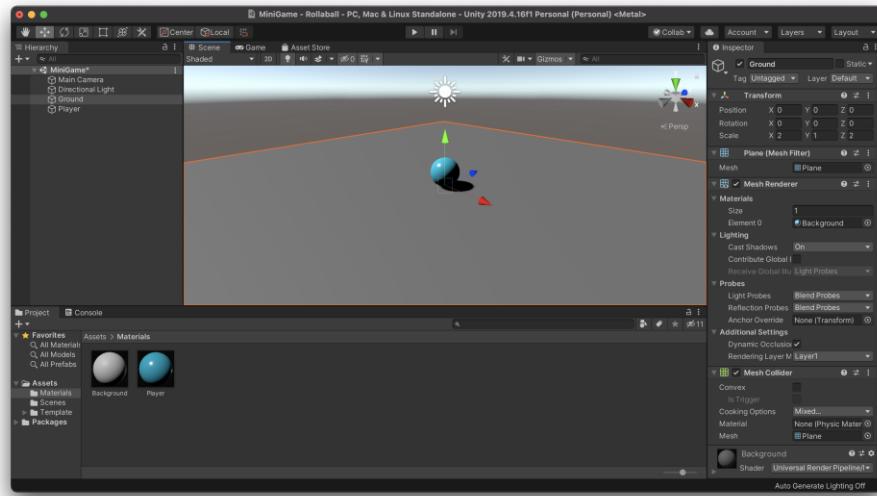
- 1) Siapkan *project game Unity*, tambahkan *object Plane* dan *object Player* pada *project* tersebut. Untuk menyiapkan *project game*, *object Plane*, dan *object Player* dapat mengikuti bagian pertama *tutorial* dari *Unity* berikut <https://learn.unity.com/project/roll-a-ball>.



Gambar 2.1 Membuat project baru dengan game engine Unity

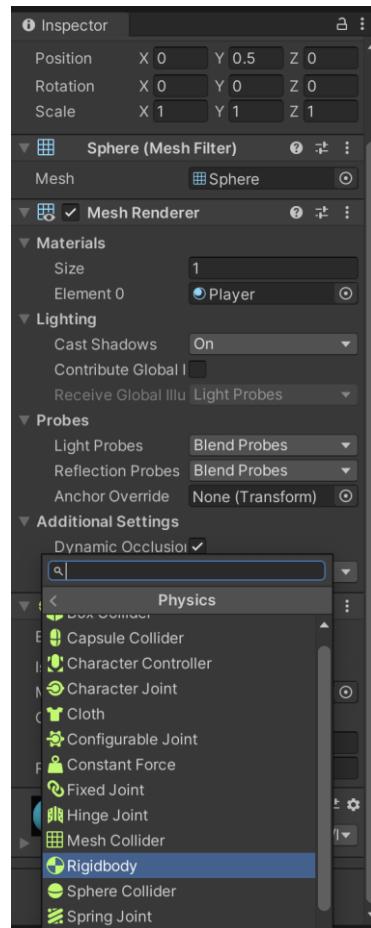


Gambar 2.2 Konfigurasi project Unity



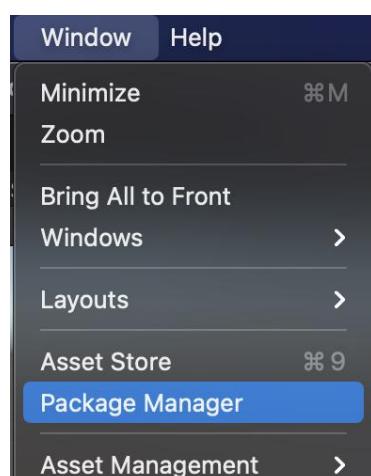
Gambar 2.3 Menyiapkan object Plane dan object Player

- 2) Tambahkan komponen ‘RigidBody’ pada *object Player* sehingga dapat memberikan *physics behavior* pada *object Player* untuk dapat menggerakkan *object* secara interaktif. Pergerakan *GameObject* pada Unity dilakukan dengan memberikan gaya berdasarkan rumus fisika terhadap *GameObject* tersebut. Bila *GameObject* berupa *Sphere* (bola) maka akan melakukan pergerakan menggelinding saat bergerak.  
Pilih *object Player* lalu tekan tombol ‘Add Component’ pada *Inspector Pane*, dan pilih Physics → RigidBody.



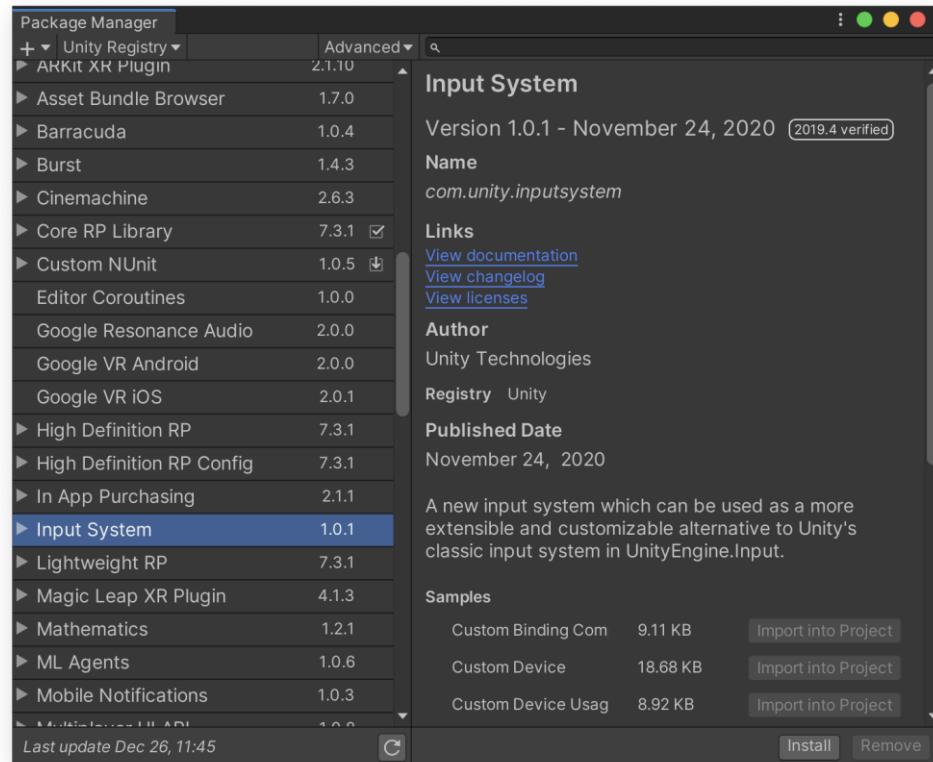
Gambar 2.4 Menambahkan komponen Rigidbody pada object Player

- 3) Tambahkan *Input System Package* supaya dapat menambahkan pengendalian objek *game* oleh pengguna melalui perangkat input seperti *keyboard*, *mouse*, dan *controller pad*. Pilih menu Window → Package Manager.



Gambar 2.5 Menu Window Package Manager

Cari *Input System*, lalu tekan tombol ‘Install’.



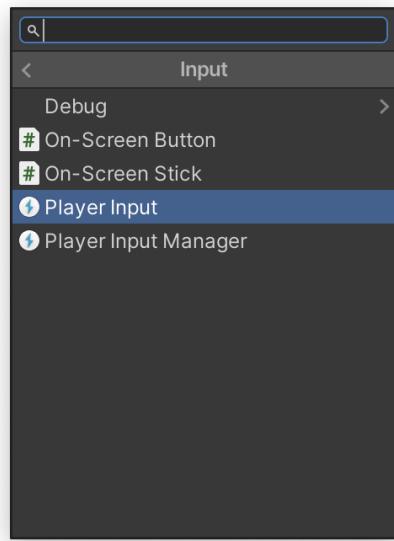
Gambar 2.6 Meng-install Input System pada Package Manager

Tekan tombol ‘Yes’ untuk menerapkan instalasi *package*, lalu tekan tombol ‘Save’ bila tedpat pertanyaan apakah ingin menyimpan progres *project* atau tidak.



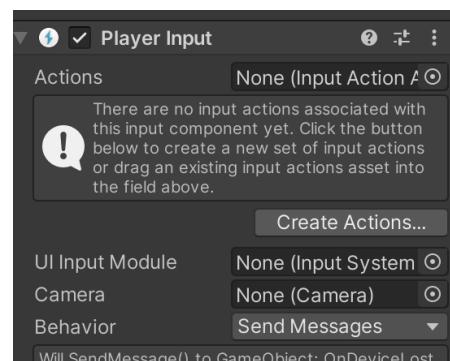
Gambar 2.7 Notifikasi instalasi package

- 4) Tambahkan komponen *Player Input* pada *object Player* untuk menerapkan *Input System*. Pilih *object Player* lalu tekan tombol ‘Add Component’ pada *Inspector Pane*, dan pilih *Input → Player Input*.

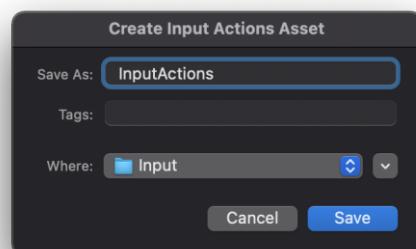


Gambar 2.8 Menambahkan komponen Player Input

Tambahkan *object Action* pada *Player Input* agar *object Player* dapat diberi *action* dengan *keyboard*. Tekan tombol ‘Create Actions...’ pada komponen Player input, buat folder ‘Input’ dan beri nama object *Action*.

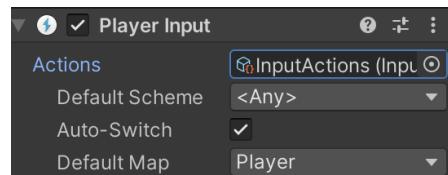


Gambar 2.9 Komponen Player Input



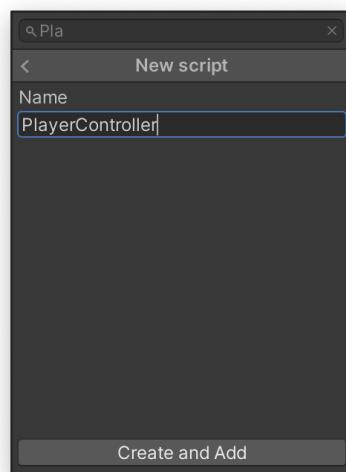
Gambar 2.10 Membuat folder Input dan memberi nama object Action

Tambahkan *object Action* ke dalam *property Action* komponen *Player Input*.



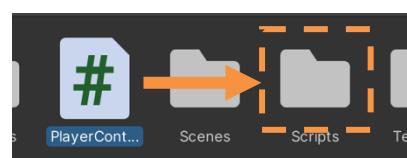
Gambar 2.11 Menambahkan object Action pada komponen Player Input

- 5) Tambahkan komponen *Script* baru pada *object Player* melalui tombol ‘Add Component’ pilih ‘New script’. Beri nama, lalu tekan tombol ‘Create and Add’.



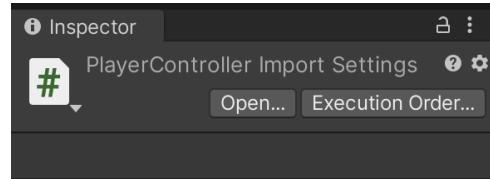
Gambar 2.12 Membuat Script baru dan menambahkannya pada object Player

Buat *folder* ‘Scripts’ pada *folder* ‘Asset’s untuk menyimpan sumber daya kode program C#. Pindahkan *file Script* yang dibuat sebelumnya ke dalam *folder* tersebut.



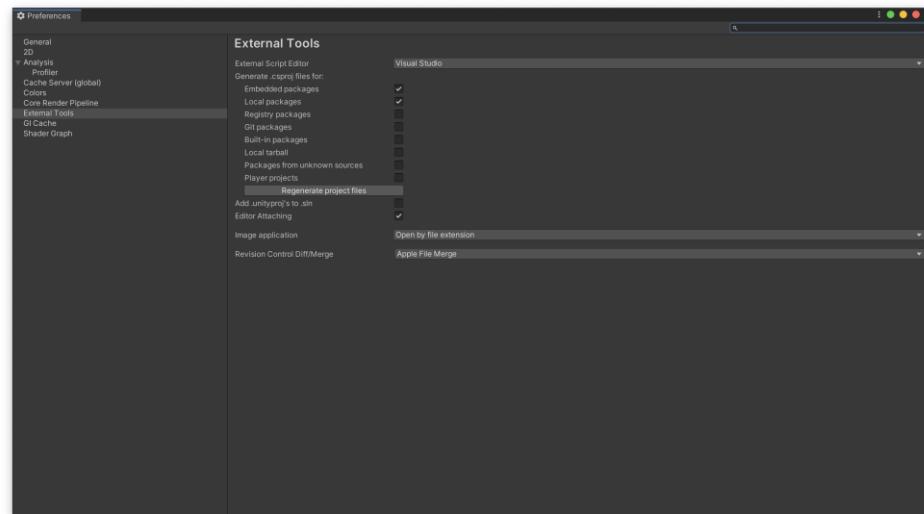
Gambar 2.13 Memindahkan file Script ke dalam folder Scripts

- 6) Pilih *file Script* yang dibuat pada *folder* ‘Scripts’. Tekan tombol ‘Open’ pada *Inspector Pane* untuk membuka *file Script* dengan *Text Editor* atau *IDE*.



Gambar 2.14 Tombol open file Script

Bila ingin membuka *file Script* dengan *Visual Studio*, pastikan *Visual Studio* digunakan sebagai *External Script Editor*.



Gambar 2.15 Konfigurasi Visual Studio sebagai External Script Editor

7) Tulis kode program sebagai berikut.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.InputSystem;
5
6  public class PlayerController : MonoBehaviour
7  {
8      public float speed = 0;
9
10     private Rigidbody rb;
11
12     private float movementX;
13     private float movementY;
14
15     void Start()
16     {
17         rb = GetComponent<Rigidbody>();
18     }
19
20     private void OnMove(InputValue movementValue)
21     {
22         Vector2 movementVector = movementValue.Get<Vector2>();
23
24         movementX = movementVector.x;
25         movementY = movementVector.y;
26     }
27
28     private void FixedUpdate()
29     {
30         Vector3 movement = new Vector3(movementX, 0.0f, movementY);
31
32         rb.AddForce(movement * speed);
33     }
34 }
35 }
```

Gambar 2.16 Kode program menggerakkan objek Player secara horizontal

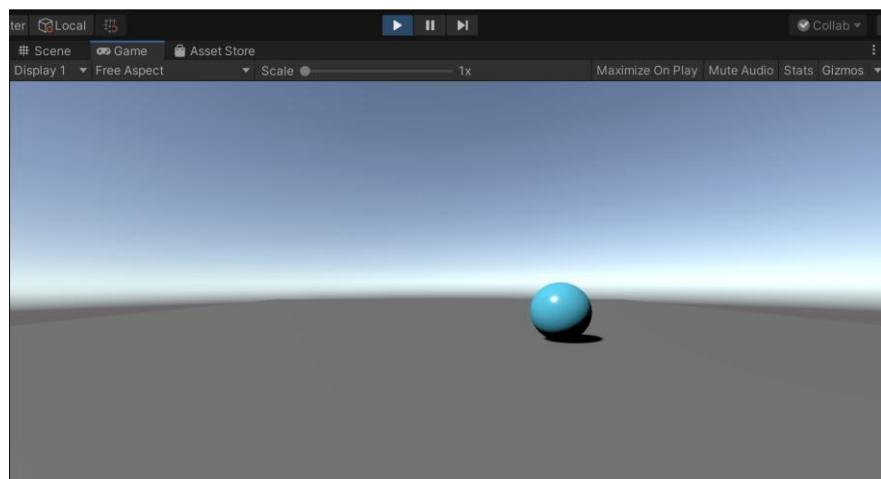
Kode program di atas memberikan perintah terhadap *object Player* untuk bergerak secara horizontal berdasarkan input arah panah *keyboard*.

Beri nilai ‘10’ pada *variable Speed* komponen *Script object Player*.



Gambar 2.17 Memberi nilai variable Speed

- 8) Tekan tombol ‘Play’ pada *Unity* untuk menjalankan game, dan tekan tombol arah panah pada *keyboard* untuk menggerakkan *object Player*.



Gambar 2.18 Hasil menjalankan game Unity

### 3. Kotlin sebagai Bahasa Pemrograman Aplikasi Mobile Android

Bahasa Pemrograman *Kotlin* pertama kali dikembangkan oleh *Jetbrains*. Mulai dibuka sebagai *open source* dengan lisensi *Apache 2* pada Februari 2012. Bahasa Pemrograman *Kotlin* dikembangkan berdasarkan Bahasa Pemrograman *Java* yang merupakan Bahasa Pemrograman *Cross-platform*.

Karena Bahasa Pemrograman *Kotlin* dikembangkan berdasarkan Bahasa Pemrograman *Java*, Bahasa Pemrograman *Kotlin* juga merupakan Bahasa Pemrograman *Cross-platform* yang dapat digunakan untuk mengembangkan aplikasi *desktop*, *web*, maupun *mobile*. Perkembangan belakangan ini, Bahasa Pemrograman *Kotlin* juga dapat digunakan untuk mengembangkan aplikasi *mobile* berbasis *IOS*. Meskipun demikian, Bahasa Pemrograman *Kotlin* sangat cocok untuk digunakan

dalam mengembangkan aplikasi *mobile* berbasis *Android*. Dengan membawa semua keunggulan bahasa pemrograman modern ke dalam *platform Android* tanpa hambatan.

Berikut adalah keunggulan Bahasa Pemrograman *Kotlin*.

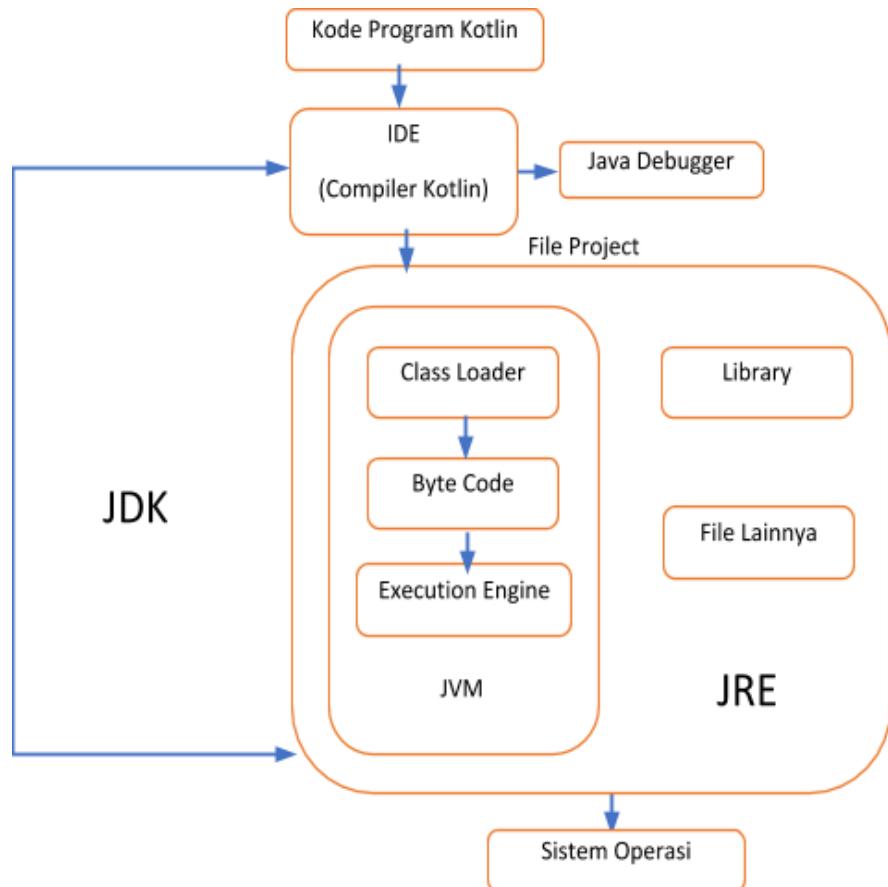
- 1) Performa : pada struktur *bytecode* yang sama, Bahasa Pemrograman *Kotlin* berjalan lebih cepat dibandingkan dengan Bahasa Pemrograman *Java*.
- 2) Mudah untuk dipelajari : Bahasa Pemrograman *Kotlin* sebagian besar memiliki kesamaan dengan Bahasa Pemrograman *Java*. Dimana perbedaannya hanya terdapat pada konsep dasar yang dapat dipelajari dengan mudah.
- 3) Terintegrasi dengan *Android Studio* : Tools *Kotlin* sudah didukung sepenuhnya oleh *Android Studio* dan kompatibel dengan sistem *Android*. Hanya membutuhkan sedikit konfigurasi untuk memulainya. *Android Studio* mempunyai fitur untuk mengkonversi kode program dalam Bahasa Pemrograman *Java* menjadi Bahasa Pemrograman *Kotlin*.
- 4) Interoperabilitas : Yaitu kemampuannya untuk dapat berinteraksi dengan bahasa pemrograman lain terutama Bahasa Pemrograman *Java*. Aplikasi dengan Bahasa Pemrograman *Kotlin* dapat menggunakan *library Android* yang tersedia. Bahkan dapat membuat *project* dengan menggunakan Bahasa Pemrograman *Java* dan *Kotlin* secara bersama-sama.
- 5) Ekspresif : Salah satu keunggulan utama Bahasa Pemrograman *Kotlin*. Dengan keunggulan ini Anda dapat melakukan banyak hal hanya dengan menulis sedikit kode program dibandingkan dengan Bahasa Pemrograman *Java*.

Aplikasi dengan Bahasa Pemrograman *Kotlin* membutuhkan perangkat lunak bernama *IDE* (*Integrated Development Environment*) untuk membuat dan membaca kode programnya. Perangkat lunak tersebut di antaranya adalah *Eclipse*, *IntelliJ IDEA* dan *Android Studio*.

*File* kode program *Kotlin* yang ditulis menggunakan *IDE* berjalan di *JRE* (*Java Runtime Environment*) sama halnya dengan *file* kode program *Java*. Untuk itu dibutuhkan *JDK* (*Java Development Kit*) untuk mengeksekusi aplikasi *Kotlin*.

*JDK* berisi komponen yang dibutuhkan untuk menjalankan *Kotlin* dan

mengirimkan hasilnya ke sistem operasi.



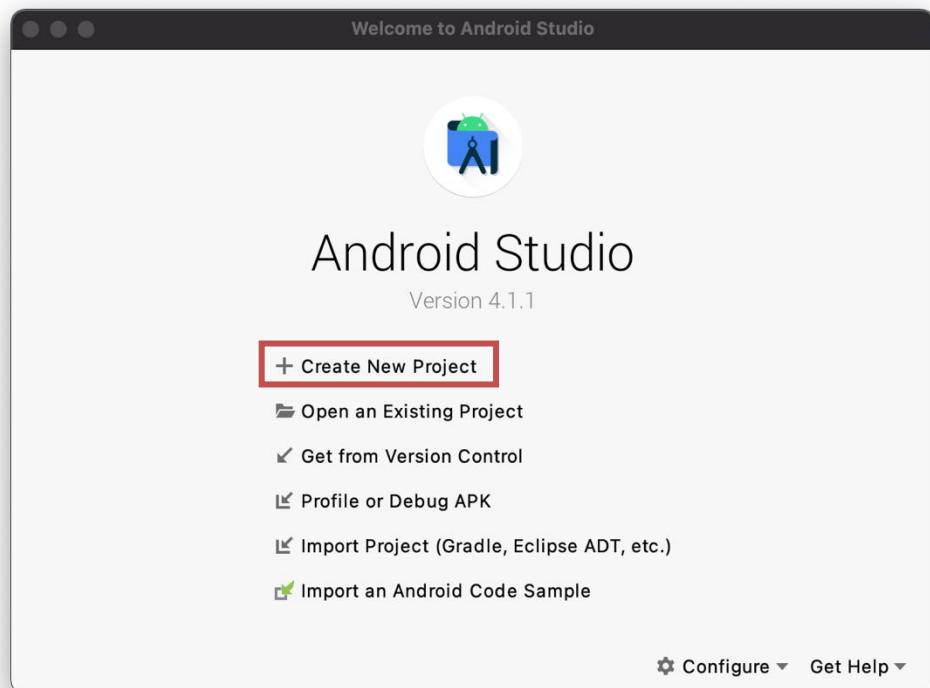
Gambar 3.1 Ilustrasi eksekusi apliksi Bahasa Pemrograman Kotlin

Sama halnya dengan kode program *Java*, kode program yang ditulis menggunakan *Kotlin* dapat dijalankan melalui *Command Line* dengan bantuan *file build.gradle*. Selain itu, kode program *Kotlin* juga dapat dijalankan menggunakan *IDE* yang dapat menjalankan kode program *Java* seperti *Eclipse* atau *IntelliJ IDEA*.

Namun, pada modul ini akan memfokuskan pada salah satu keunggulan Bahasa Pemrograman *Kotlin*, yaitu sangat cocok untuk mengembangkan aplikasi *mobile* berbasis *Android*. Untuk mengembangkan dan menjalankan aplikasi *mobile* berbasis *Android* dengan Bahasa Pemrograman *Kotlin*, dapat menggunakan *IDE Android Studio* yang dikeluarkan dan dikembangkan oleh tim *developer Google*.

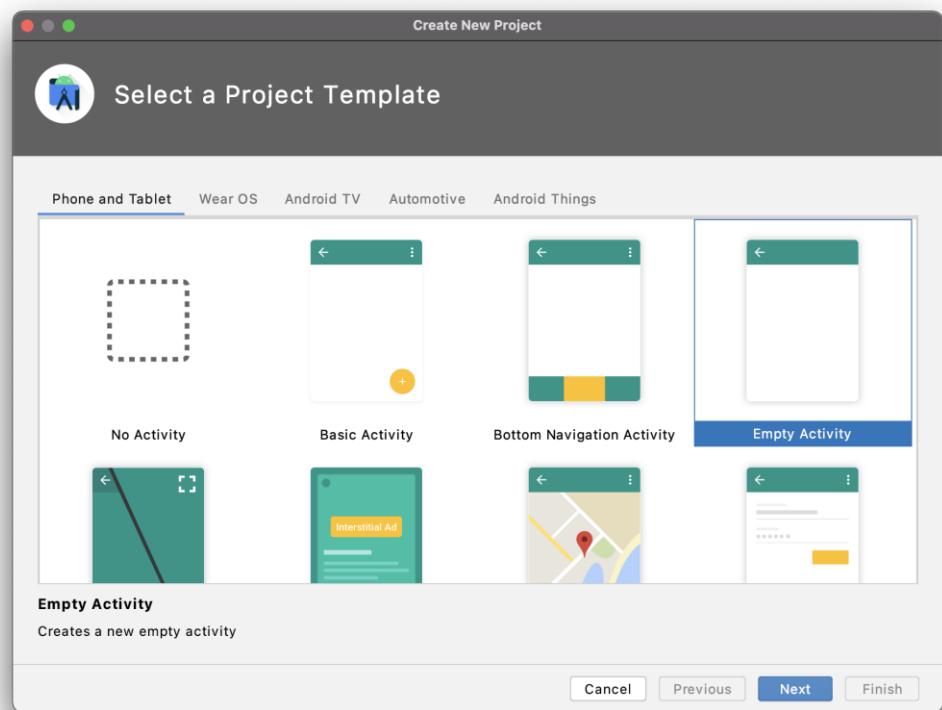
Berikut adalah langkah-langkah membuat project baru aplikasi *mobile* berbasis *Android* dengan *Kotlin* menggunakan perangkat lunak *Android Studio*. (pada modul ini menggunakan *Android Studio* versi 4.1.1)

- 1) Jalankan *Android Studio*, maka muncul *window* seperti berikut. Pilih ‘Create New Project’ untuk membuat *project* baru.



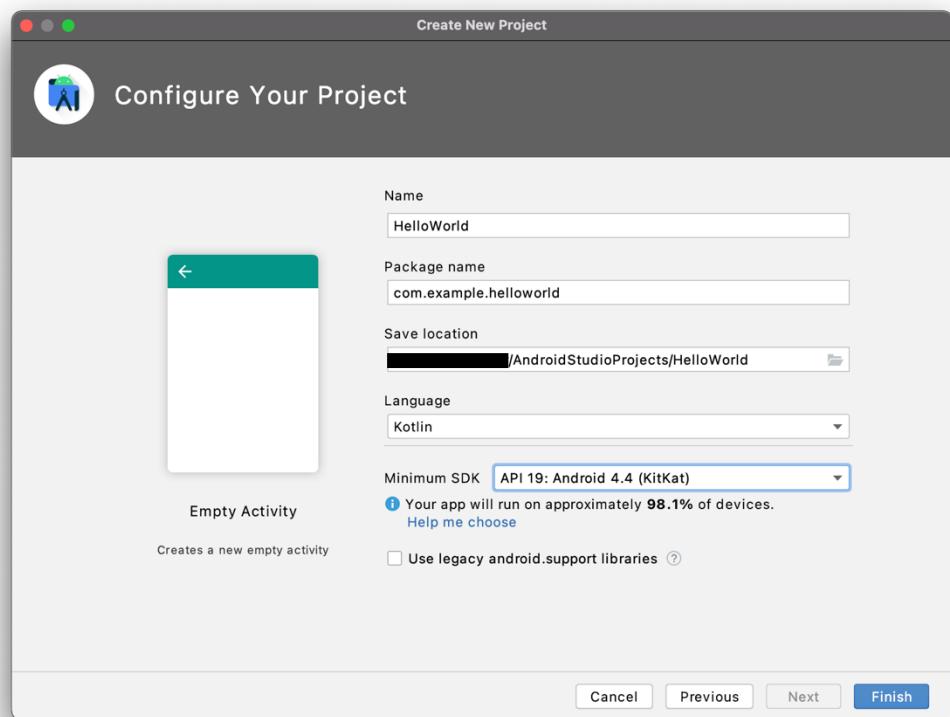
Gambar 3.2 Window awal Android Studio

- 2) Pilih *template project* yang akan dikembangkan. Anda dapat memilih *template project* untuk *smartphone/tablet*, *wear OS* seperti *smartwatch*, *Android TV*, perangkat lunak berbasis *Android* pada Otomotif, dan perangkat lunak *Internet of Things* berbasis *Android*. Pilih *template smartphone/tablet* dengan ‘Empty Activity’ untuk memulai *project smartphone/tablet* dengan tampilan awal kosong.



Gambar 3.3 Memilih template project

- 3) Lakukan konfigurasi *project*. Konfigurasi yang dapat dilakukan di antaranya:
  - a) Mengubah nama *project*
  - b) Mengubah nama *package*
  - c) Mengubah lokasi penyimpanan *project*
  - d) Mengubah bahasa pemrograman yang digunakan
  - e) Mengubah versi minimal *SDK* yang digunakan



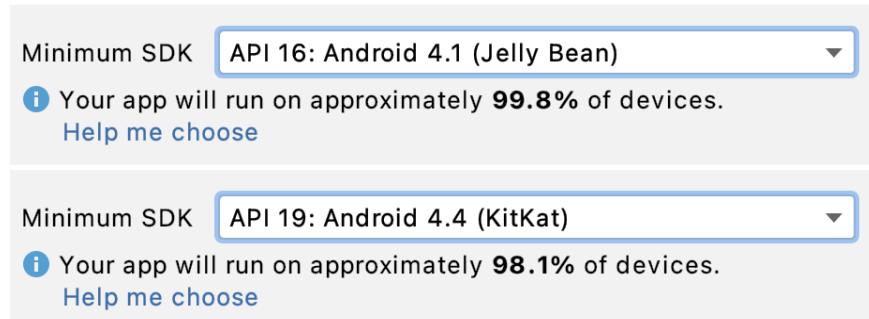
Gambar 3.4 Konfigurasi project

Nama *project* yang ditulis pada konfigurasi ini akan menjadi nama aplikasi *mobile* Anda. Sedangkan nama *package* adalah gabungan dari nama *project* dan nama domain perusahaan Anda. Bila Anda belum berencana untuk mempublikasikan aplikasi mobile yang dikembangkan, Anda dapat menggunakan nama domain perusahaan *default* yang disediakan *Android Studio* yaitu *example.com*. Nama *package* ini akan menjadi nama *folder virtual* untuk semua sumber daya kode program yang akan Anda kembangkan di dalam *project* tersebut.

Secara *default*, *Android Studio* akan membuat *folder* bernama 'AndroidStudioProjects' di dalam *folder User* yang menjalankan *Android Studio* sebagai destinasi penyimpanan *project*. Anda dapat mengubah destinasi penyimpanan *project* dengan menulis langsung alamat *folder* yang diinginkan atau dengan menekan tombol dengan *icon folder* untuk memilih *folder* destinasi penyimpanan *project* yang diinginkan.

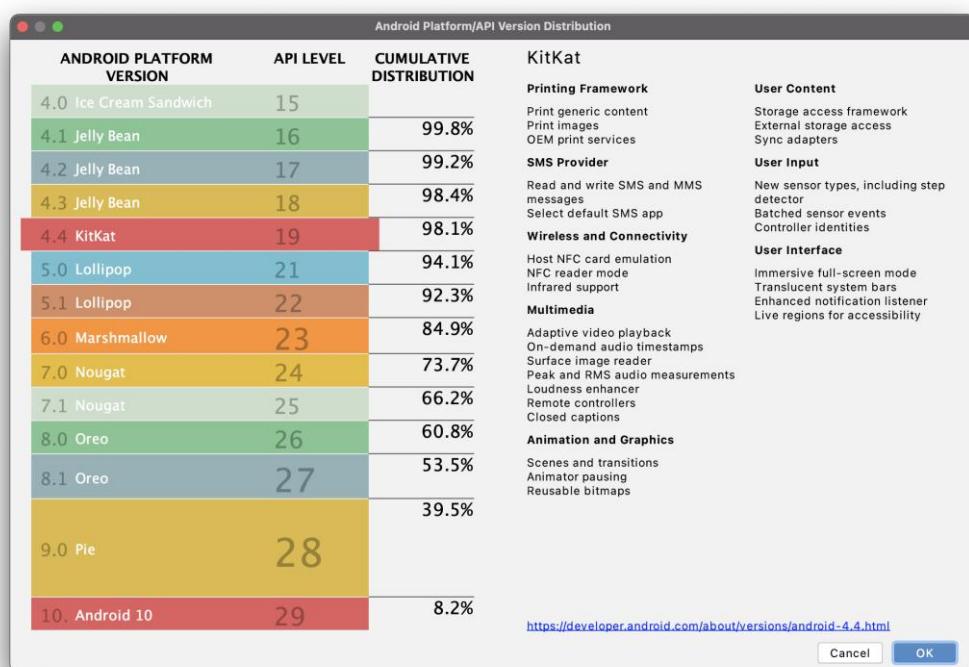
*Android Studio* menyediakan 2 opsi bahasa pemrograman untuk membuat aplikasi mobile yaitu Bahasa Pemrograman *Java* dan Bahasa Pemrograman *Kotlin*.

Pemilihan versi minimal *SDK* yang digunakan akan mempengaruhi pemilihan *library* dan beberapa *syntax* kode program yang akan ditulis. Semakin rendah versi minimal *SDK* akan semakin banyak konfigurasi yang harus diperhatikan selama pengembangan aplikasi. Namun semakin tinggi versi minimal *SDK* akan mempengaruhi kompatibilitas perangkat *Android* yang dapat menjalankan aplikasi Anda sehingga semakin sedikit pengguna yang akan menggunakan aplikasi Anda.



Gambar 3.5 Perbedaan perkiraan pengguna untuk versi minimal *SDK* yang berbeda

Anda dapat memilih versi minimal *SDK* berdasarkan jumlah pengguna dari setiap versi *SDK* tersebut dengan menekan link 'Help me choose' pada *window* yang sama. Sehingga memunculkan grafik jumlah pengguna berdasarkan versi *SDK*.



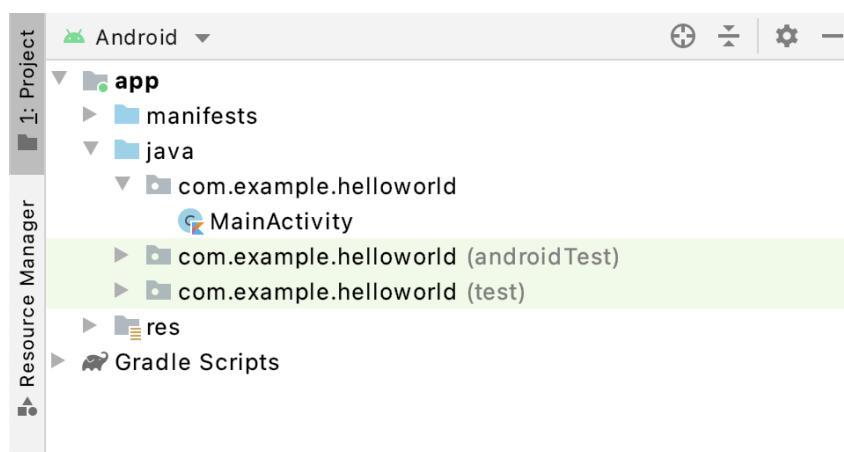
Gambar 3.6 Grafik jumlah pengguna berdasarkan versi *SDK*

Bila diperhatikan, perbedaan persentase distribusi pengguna sangat sedikit di bawah 1% pada versi *API level* 19 ke *API level* 18, maupun ke *API level* di bawahnya. Sehingga untuk saat ini pada saat modul ini dibuat, lebih baik memilih *API level* 19 sebagai versi minimum *SDK* yang digunakan.

Bila Anda memilih versi minimal *SDK* di bawah *API level* 29, akan muncul opsi untuk menggunakan *library android.support* sebagai *library* untuk *API* yang didapatkan di luar *framework API* yang disediakan secara *default* oleh *Android Studio* untuk digunakan pada *API level* lebih rendah. Namun, *library android.support* ini sudah usang dan sudah digantikan oleh *library* yang mencakup *library* sebelumnya yaitu *library AndroidX* sehingga Anda tidak perlu menceklis opsi *library android.support* tersebut.

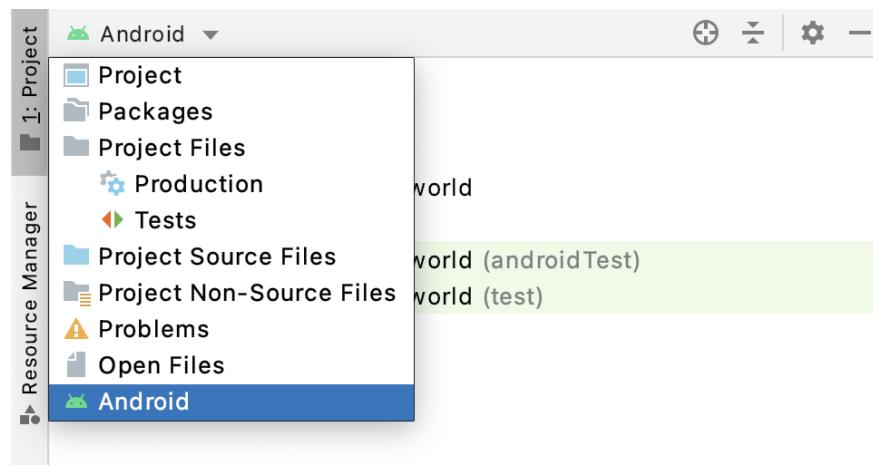
Bila sudah selesai melakukan konfigurasi *project*, tekan tombol *finish* untuk menyelesaikan konfigurasi *project* tersebut. Dalam membuat project baru ini, perangkat komputer Anda harus terhubung dengan internet. Karena *Android Studio* akan otomatis mencari dan meng-install sumber daya yang dibutuhkan dalam membuat project baru melalui internet. Proses ini akan memakan waktu yang cukup lama. Karena tidak hanya men-*download* sumber daya, *Android Studio* juga melakukan instalasi terhadap sumber daya yang didapatkan untuk menyiapkan *project* siap digunakan untuk pengembangan.

Setelah selesai men-*download* dan meng-*install* sumber daya yang diperlukan, *Android Studio* akan melakukan indexing untuk menyiapkan struktur sumber daya *project* yang akan ditampilkan di dalam *Project Pane* *Android Studio*.



Gambar 3.7 Struktur sumber daya project Android Studio dalam hierarki Android

Secara *default*, *Android Studio* akan menampilkan sumber daya *project* dalam bentuk hierarki *Android* yang terdiri dari 2 struktur *folder* utama yaitu *folder* ‘app’ dan *folder* ‘Gradle Scripts’. Anda dapat mengubah tampilan hierarki ini ke dalam bentuk hierarki *Project* dengan memilih opsi *drop-down* pada *Project Pane*.



Gambar 3.8 Opsi hierarki struktur Project

Umumnya, pengembangan aplikasi *mobile* dengan *Android Studio* menggunakan hierarki *Android* dalam mengelola sumber daya yang digunakan. Hierarki *Project* biasanya digunakan untuk mengelola sumber daya *project* bila inisiasi *indexing* awal *project* gagal dilakukan. Secara *default*, *Android Studio* akan membuat *file* kode program berbasis *Kotlin* ‘*MainActivity.kt*’ dan *file* *layout* berbasis *XML* ‘*activity\_main.xml*’.

A screenshot of the Android Studio code editor. The tabs at the top show 'activity\_main.xml' and 'MainActivity.kt'. The code editor displays the following Kotlin code for 'MainActivity.kt':

```
1 package com.example.helloworld
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }
```

The code is color-coded, with 'package', 'import', 'class', 'override', 'super', and 'setContentView' keywords in blue, and variable names like 'MainActivity', 'AppCompatActivity', ' savedInstanceState', 'super', and 'activity\_main' in purple.

Gambar 3.9 MainActivity.kt

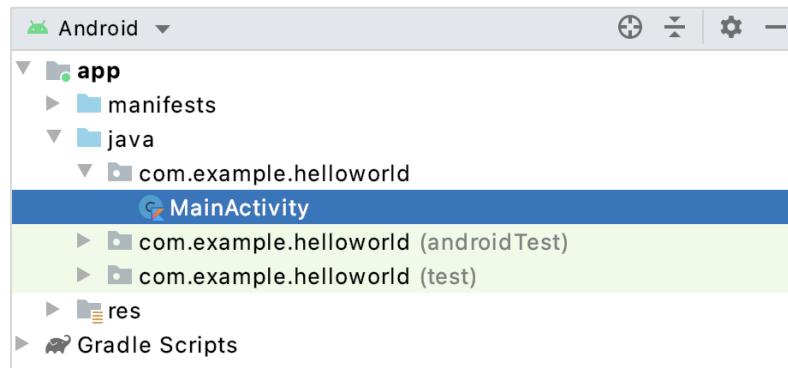
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

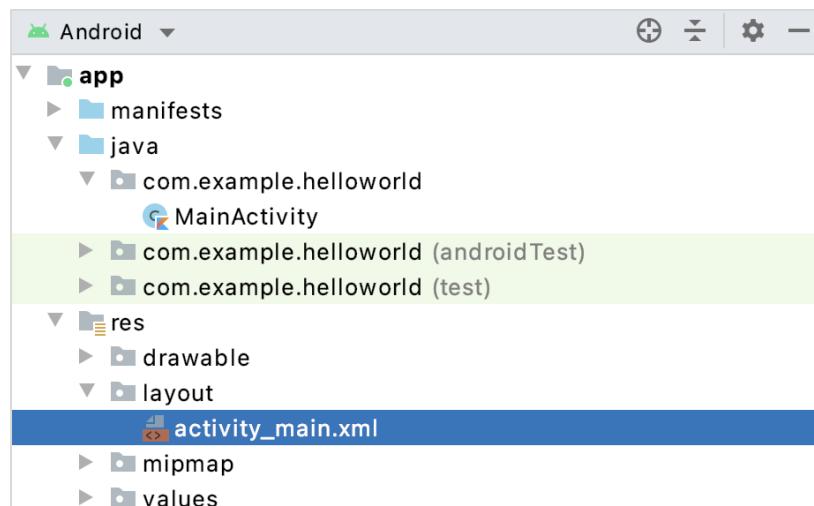
Gambar 3.10 activity\_main.xml

Pada hierarki *Android*, file ‘*MainActivity.kt*’ terletak di dalam *folder* nama *package* yang ditentukan pada saat melakukan konfigurasi *project* yang dilakukan sebelumnya pada *folder* ‘*app/java*’.



Gambar 3.11 Letak file *MainActivity.kt* pada hierarki *Android*

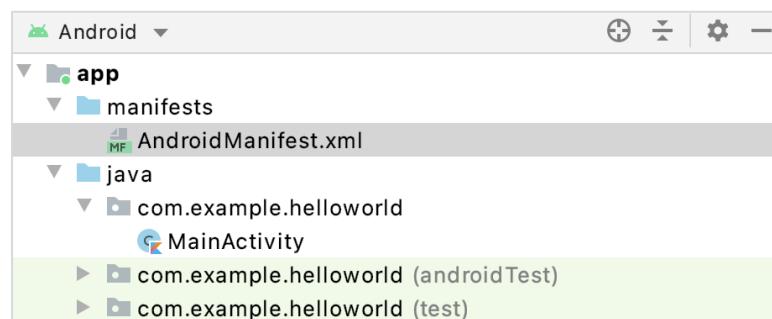
Sedangkan, pada hierarki *Android*, file ‘*activity\_main.xml*’ terletak di dalam *folder layout* pada *folder* ‘*app/res*’.



Gambar 3.12 Letak file *activity\_main.xml* pada hierarki *Android*

*File ‘MainActivity.kt’ merupakan file utama dalam menulis kode program menggunakan Bahasa Pemrograman Kotlin. Sedangkan file ‘activity\_main.xml’ merupakan salah satu file sumber daya yang digunakan pada file ‘MainActivity.kt’ untuk mendefinisikan layout User Interface aplikasi mobile yang dikembangkan.*

Selain kedua file tersebut, terdapat file ‘AndroidManifest.xml’ yang terdapat di dalam folder ‘app/manifest’. File ini berisi informasi penting dari aplikasi mobile yang dikembangkan seperti, sumber daya icon yang digunakan, nama aplikasi, permission yang dibutuhkan (untuk mengakses fitur-fitur perangkat mobile), serta Activity yang digunakan oleh aplikasi mobile tersebut.



Gambar 3.13 Letak file AndroidManifest.xml pda hierarki Android

Setiap Activity yang digunakan oleh aplikasi harus didefinisikan di dalam file ini. Secara default, ‘MainActivity’ ditunjuk sebagai Activity pertama yang akan dipanggil bila aplikasi tersebut dijalankan.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld">

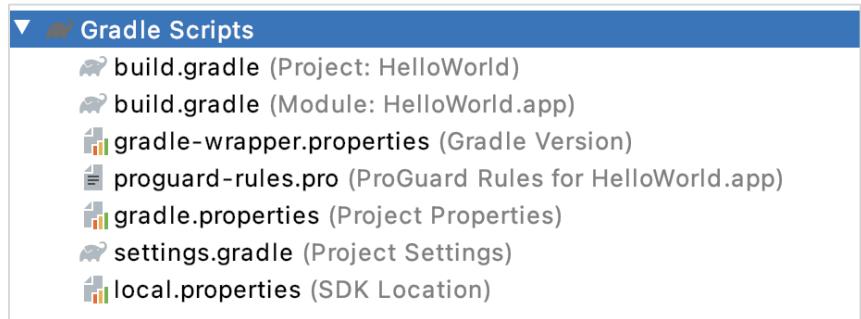
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HelloWorld">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Gambar 3.14 AndroidManifest.xml

Android Studio menyediakan file Gradle yang memungkinkan proses Build aplikasi, yaitu proses compiling kode program untuk menghasilkan

*file APK (Android Package Kit)*, dilakukan secara otomatis. Di dalam *file Gradle* terdapat pendefinisian struktur *project* aplikasi, konfigurasi aplikasi, dan *dependency* sumber daya eksternal yang dibutuhkan dalam melakukan *Build* aplikasi. Umumnya di dalam sebuah *project* *Android Studio* terdapat 2 *file Gradle*, yaitu ‘*build.gradle*’ *level project* dan ‘*build.gradle*’ *level module*. Secara *default* ‘*build.gradle*’ *level module* bernama ‘*app*’. Pada hierarki *Android*, *file Gradle* tersebut terdapat di dalam *folder* ‘*Gradle Scripts*’.



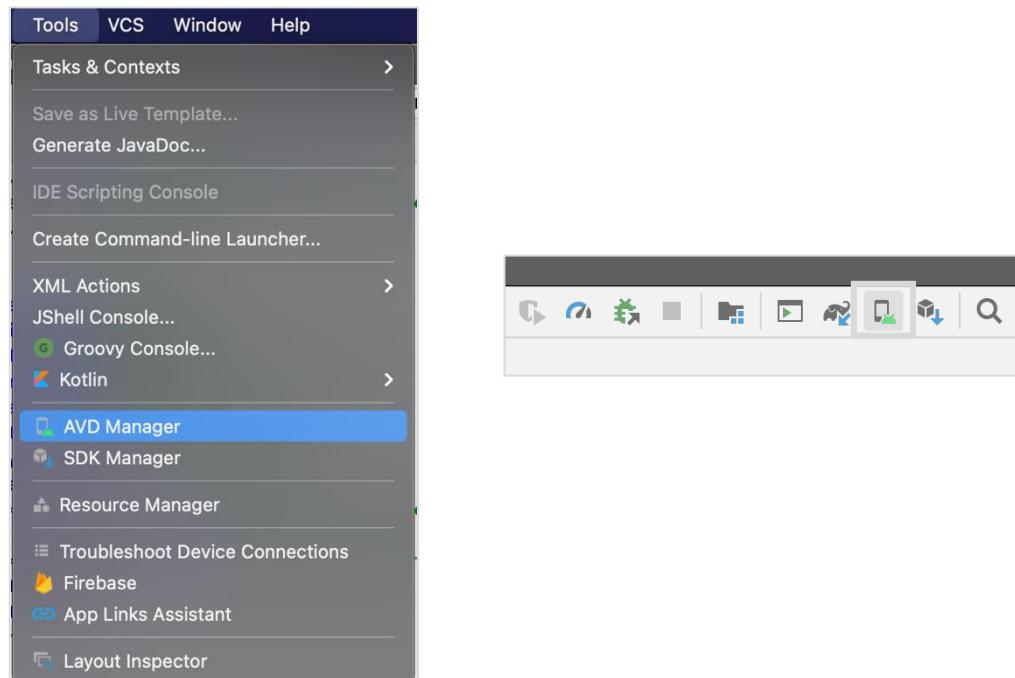
Gambar 3.15 Letak file Gradle pada hierarki Android

*File ‘build.gradle’ level module* merupakan *file Gradle* yang sering dimodifikasi. Karena di dalam *file* ini terdapat konfigurasi *dependency* sumber daya eksternal yang dibutuhkan aplikasi.

Proses *Running* pada *Android Studio* diawali dengan proses *Build* berdasarkan *file Gradle*. Terdapat 2 destinasi *Running* pada *Android Studio*, yaitu *Running* pada perangkat *Emulator* dan *Running* pada perangkat fisik.

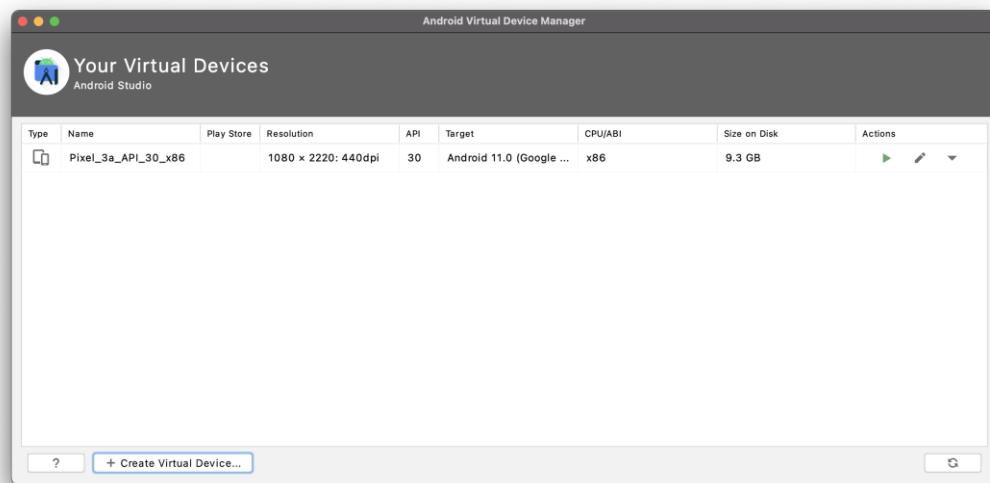
Untuk melakukan *Running* pada perangkat *Emulator*, Anda harus menyiapkan perangkat *Emulator* melalui fitur *AVD (Android Virtual Device) Manager* pada *Android Studio*. Berikut adalah langkah-langkah untuk membuat perangkat *Emulator* baru dengan *AVD Manager*.

- 1) Fitur *AVD Manager* dapat Anda akses melalui menu *Tools* → *AVD Manager* atau melalui tombol icon bergambar perangkat mobile dengan logo *Android*.



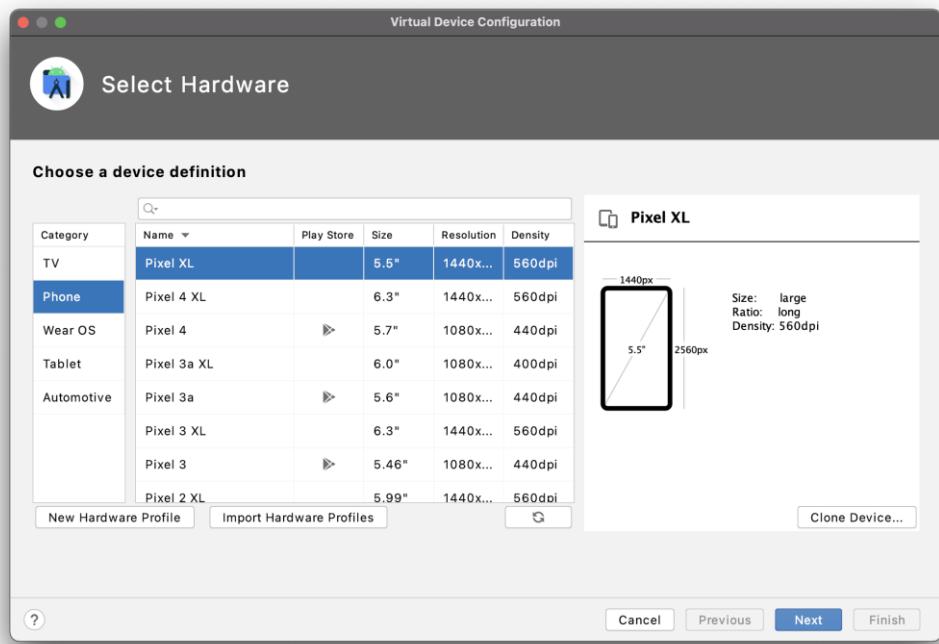
Gambar 3.16 Akses fitur AVD Manager

- 2) Untuk menambah perangkat *Emulator* baru, klik tombol ‘Create Virtual Device’ di bagian kiri bawah *window AVD Manager*.



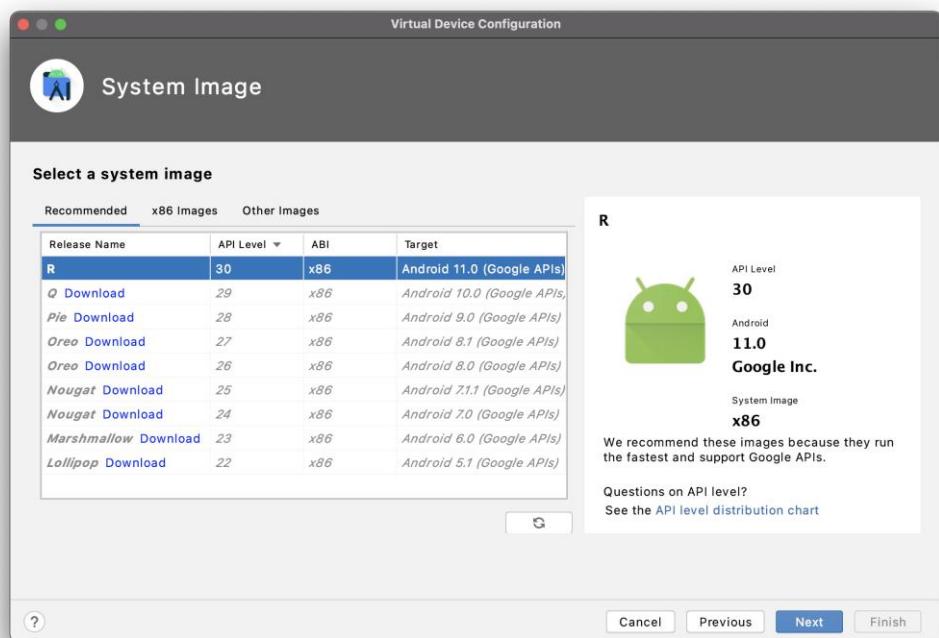
Gambar 3.17 Window AVD Manager

- 3) Pilih kategori perangkat, dan tipe perangkat yang ingin dibuat lalu tekan tombol ‘Next’.



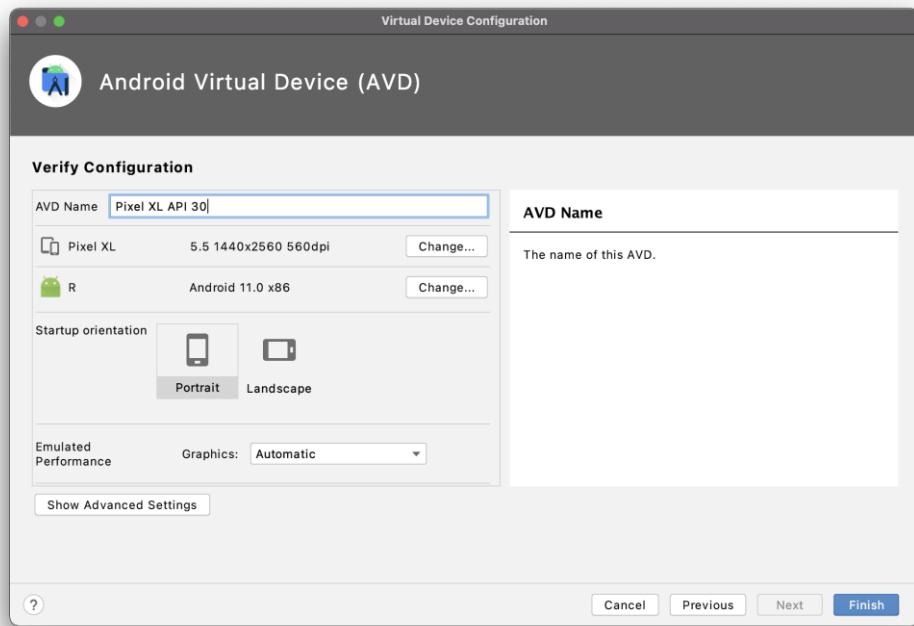
Gambar 3.18 Memilih kategori dan tipe perangkat

- 4) Pilih versi API level/OS perangkat *Emulator* lalu tekan tombol ‘Next’.



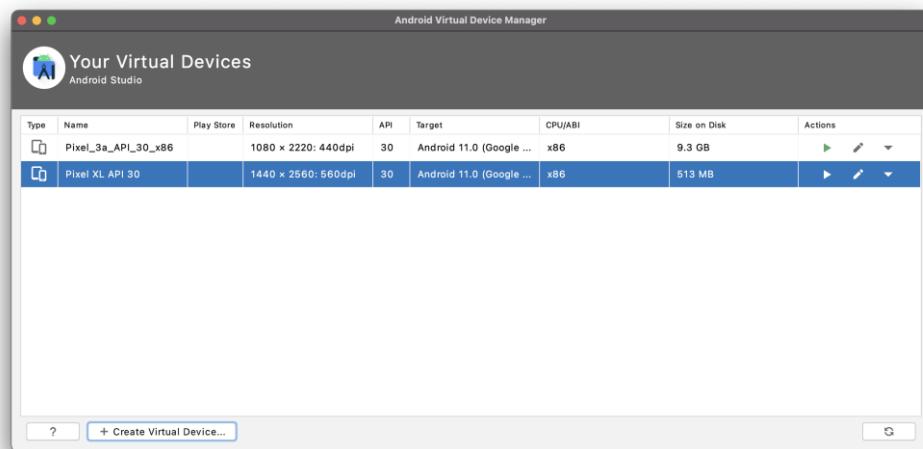
Gambar 3.19 Memilih API level

- 5) Beri nama perangkat *Emulator* dan periksa konfigurasi perangkat *Emulator* tersebut. Lalu tekan finish untuk meng-*install* perangkat *Emulator* sesuai dengan konfigurasi yang telah ditentukan.



Gambar 3.20 Konfigurasi perangkat Emulator

- 6) Bila berhasil, perangkat *Emulator* baru akan muncul di dalam *list window AVD Manager*.



Gambar 3.21 Hasil instalasi perangkat Emulator

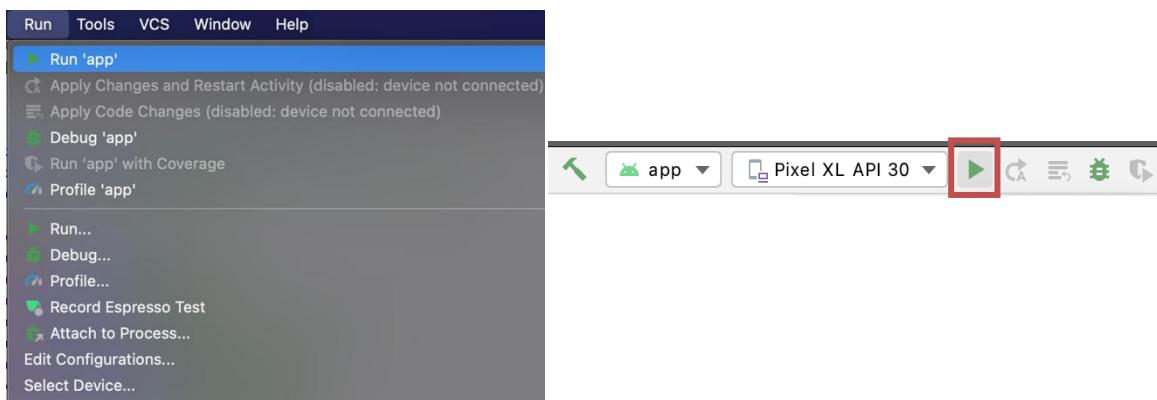
Bila sudah berhasil meng-install perangkat *Emulator*, Anda dapat memilihnya melalui *toolbar AVD* di bagian atas *Android Studio*.



Gambar 3.22 Toolbar AVD

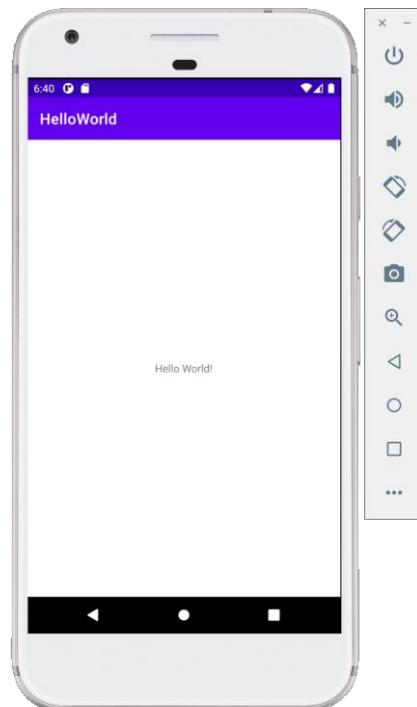
Untuk memulai proses *Running*, pilih *menu Run → Run 'app'* atau tekan

tombol dengan *icon* bergambar segitiga hijau yang menghadap ke kiri.



Gambar 3.23 Akses fitur Running Android Studio

Ketika memulai proses *Running*, perangkat *Emulator* akan dijalankan secara otomatis, dan proses *Build* akan dijalankan. Bila sudah selesai melakukan *Build*, *APK* yang terbentuk akan ter-*install* secara otomatis pada perangkat *Emulator* dan selanjutnya menjalankan aplikasi yang ter-*install* pada perangkat tersebut.



Gambar 3.24 Hasil proses Running pada perangkat Emulator

Bila Anda ingin melakukan *Running* pada perangkat fisik *Android*, Anda harus menyiapkan perangkat *mobile* berupa *smartphone* atau *tablet* berbasis *Android* dan kabel *USB* yang kompatibel dengan perangkat tersebut.

Sebelumnya, Anda harus melakukan konfigurasi terhadap perangkat fisik Anda untuk menyalakan fitur *USB Debugging* pada *Developer Tools*.

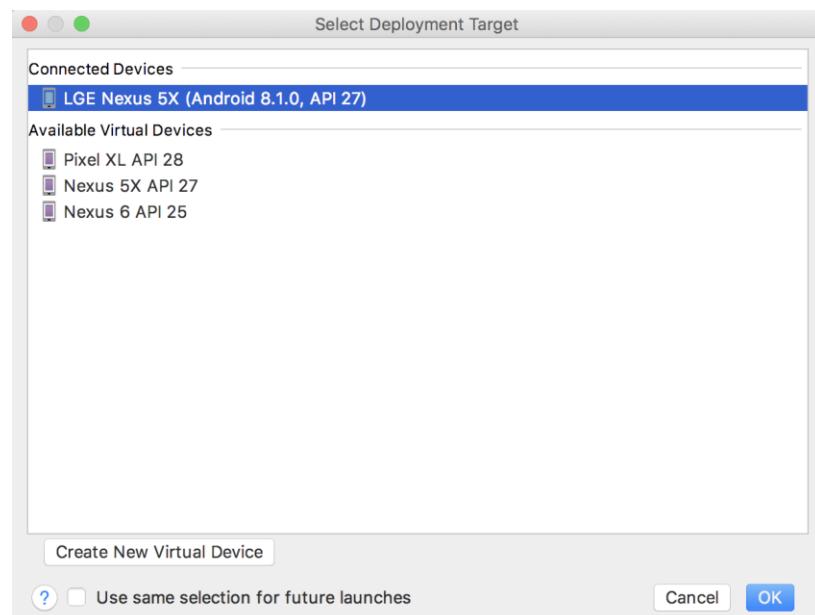
Fitur ini tidak aktif secara *default* pada perangkat mobile yang dijual secara komersil. Aktifasi fitur ini sepenuhnya adalah tanggung jawab Anda bila terjadi kerusakan karena mengaktifkan fitur tersebut.

Setiap perangkat memiliki langkah aktifasi fitur *USB Debugging* yang berbeda. Pastikan fitur ini sudah aktif, hubungkan perangkat *mobile* Anda dengan perangkat pengembangan *PC* Anda. Maka akan muncul notifikasi pada perangkat *mobile* untuk meminta permission terkait *USB Debugging*.



Gambar 3.25 Notifikasi terkait USB Debugging

Menjalankan proses *Running* dengan perangkat fisik menggunakan menu yang sama dengan proses *Running* dengan perangkat *Emulator*. Pastikan memilih perangkat yang sesuai pada saat menjalankan proses *Running*.



Gambar 3.26 Memilih perangkat yang sesuai untuk Running

#### 4. PHP sebagai Bahasa Scripting Server-Side Aplikasi Web

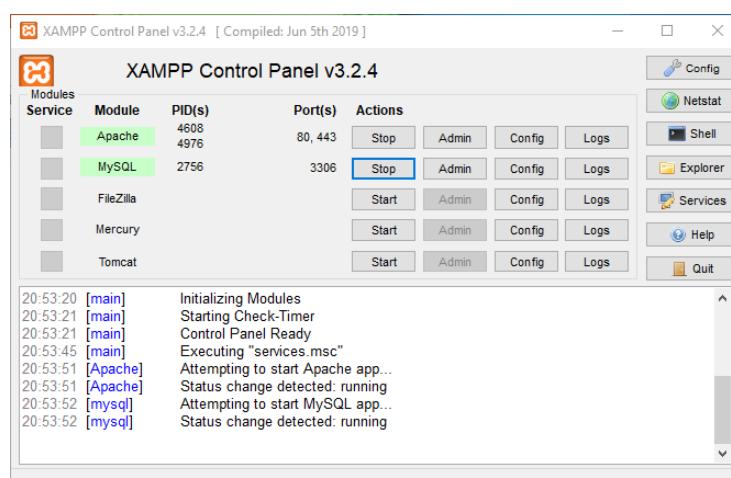
*PHP (Hypertext Preprocessor)* adalah Bahasa *Scripting* yang digunakan untuk mengembangkan *server-side* aplikasi berbasis *web*. Merupakan Bahasa *Scripting* yang dapat ditanamkan pada *HTML* untuk

menghasilkan aplikasi *web* dinamis yang membutuhkan pemrosesan data dari *web server*. Umumnya melakukan pemrosesan data yang terdapat pada *database*. Seluruh perintah *scripting* dengan *PHP* dijalankan di dalam *web server*.

Untuk menjalankan kode program *PHP*, suatu perangkat komputer harus memiliki *interpreter PHP*. *Zend Engine* digunakan sebagai *interpreter PHP* yang standar. *Interpreter* ini dapat digunakan di hampir semua *web server* yang berjalan pada masing-masing sistem operasi. *Web server* tersebut di antaranya *Apache*, dan *Nginx*. Bila menjalankan kode program *PHP* dengan *web server*, hasil *output* kode program akan dikirimkan ke aplikasi pengguna dalam bentuk *HTTP Response*.

Pada umumnya, kode program *PHP* dijalankan melalui *web server*. Anda dapat menggunakan *bundle web server* dan *database* yang populer seperti *XAMPP* untuk pengembangan aplikasi *web*. *XAMPP* sendiri merupakan singkatan, yaitu A untuk *Apache* sebagai *web server*, M untuk *MySQL/MariaDB* sebagai *database*, P yang pertama untuk *PHP* dan P yang kedua untuk *Pearl* sebagai bahasa pemrograman yang digunakan, serta X berarti *Cross Platform* yang artinya dapat dijalankan pada sistem operasi yang berbeda seperti *Windows*, *MacOS*, maupun *Linux*.

Untuk menjalankan kode program *PHP* menggunakan *XAMPP*, *web server Apache* harus diaktifkan terlebih dahulu. Bila kode program *PHP* yang dibuat mengandung operasi data pada *database*, *database MySQL* juga harus diaktifkan. *Apache* dan *MySQL* dapat diaktifkan melalui *XAMPP Control Panel*. Tekan tombol ‘Start’ pada *module Apache* dan *MySQL* untuk mengaktifkannya.

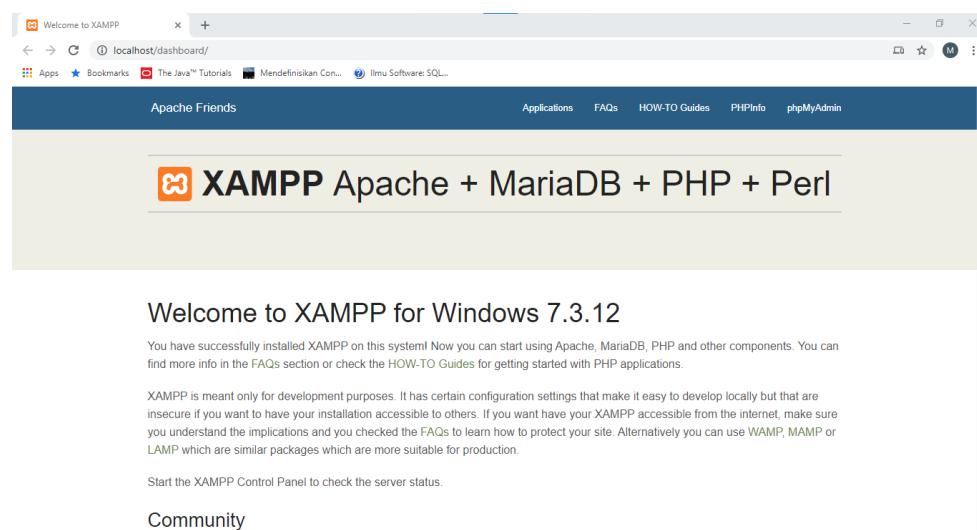


Gambar 4.1 Mengaktifkan Apache dan MySQL dengan XAMPP Control

## Panel

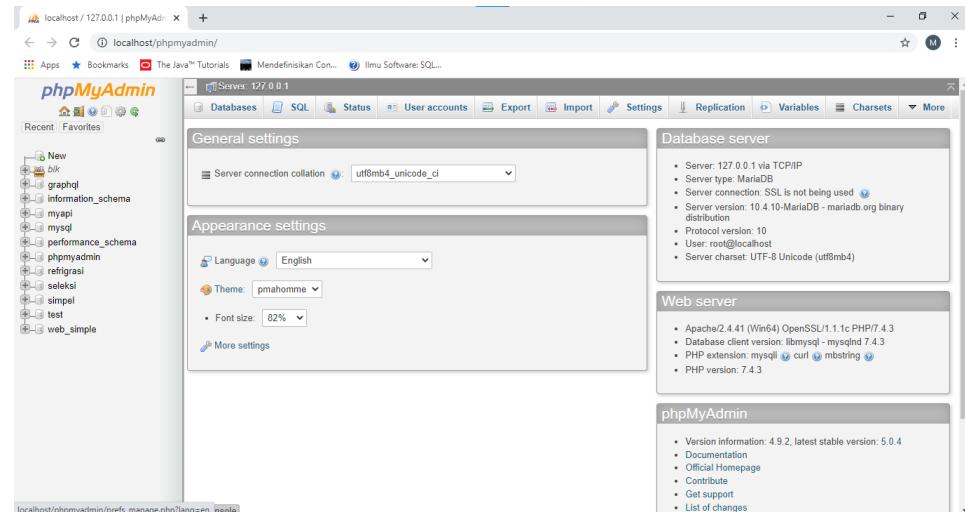
Secara *default*, *Apache* akan dijalankan pada *port* 80 sebagai *port* utama dan *port* 443 sebagai *port* *SSL*, sedangkan *MySQL* akan dijalankan pada *port* 3306. Pastikan semua *port* tersebut tidak digunakan oleh perangkat lunak lainnya pada perangkat komputer Anda.

Untuk mengecek apakah web server *Apache* berjalan dengan baik, tekan tombol ‘Admin’ pada *module Apache*. Bila berjalan dengan baik, halaman *dashboard Apache* akan terbuka pada *browser default* perangkat komputer Anda dengan alamat *url* ‘<http://localhost/dashboard>’.



Gambar 4.2 Tampilan halaman dashboard Apache

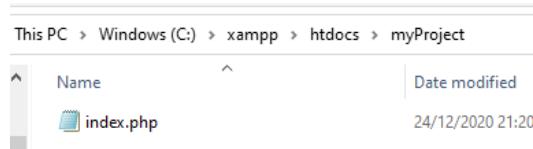
Melalui halaman *dashboard* ini, Anda dapat mengakses *PHPInfo* yang berisi informasi terkait konfigurasi *PHP* yang digunakan, juga dapat mengakses aplikasi *phpMyAdmin* yaitu aplikasi berbasis web untuk melakukan pengelolaan *database MySQL/MariaDB*. Aplikasi *phpMyAdmin* juga dapat diakses melalui tombol ‘Admin’ pada *module MySQL*.



Gambar 4.3 Tampilan aplikasi phpMyAdmin

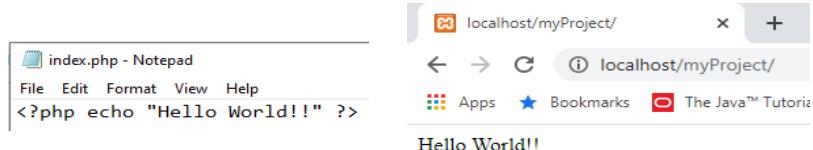
Aplikasi yang dijalankan melalui *web server* dapat diakses menggunakan *web browser*, yaitu *tools* yang dapat mengakses *web server* untuk menampilkan halaman *web* dengan cara mengirimkan *HTTP Request* dan menerima *HTTP Response*. Terdapat berbagai macam *web browser* di antaranya, *Google Chrome*, *Microsoft Edge*, *Mozilla Firefox*, *Safari*, *Opera*, dan masih banyak lagi. Di antara semua browser tersebut, *Google Chrome* memiliki pengguna terbanyak, diikuti oleh *Microsoft Edge* dan *Mozilla Firefox*.

Bila menggunakan *XAMPP*, aplikasi yang dikembangkan diakses melalui alamat ‘`http://localhost`’ yang dituliskan pada *toolbar* alamat *URL web browser*. Alamat ini terhubung dengan *folder* ‘`xampp/htdocs`’ sehingga semua *project* aplikasi yang dikembangkan dengan *PHP* harus disimpan di dalam *folder* ini bila menggunakan *XAMPP*. Simpanlah semua *file* dengan format ‘`.php`’ yang Anda tulis dengan *Text Editor* seperti *Notepad*, *Notepad++*, *Sublime*, *Atom*, *Visual Studio Code*, dll ke dalam *folder* ‘`xampp/htdocs`’.



Gambar 4.4 Contoh menyimpan project myProject di dalam folder xampp/htdocs

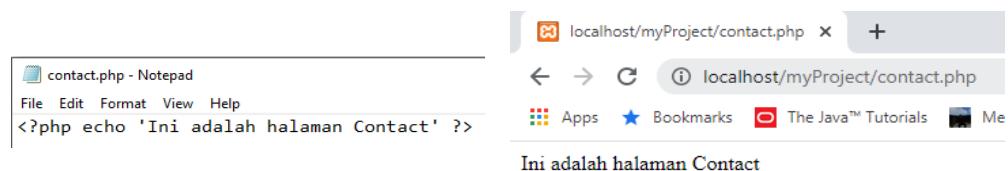
Untuk mengakses aplikasi project pada contoh gambar di atas, dilakukan dengan memasukkan ‘`http://localhost/myProject`’ pada *toolbar* alamat *URL web browser*.



Gambar 4.5 Menjalankan project PHP pada browser

Semua *browser* secara *default* akan mencari *file* bernama ‘index’ di dalam *folder* alamat URL yang diakses. Baik *file* ‘index’ yang memiliki format ‘.html’ maupun *file* ‘index’ yang memiliki format ‘.php’, sehingga untuk mengaksesnya tidak perlu mencantumkan nama *file* ‘index’ untuk diakses. Bila kode program mengandung *syntax PHP*, format *file* haruslah ‘.php’.

Untuk mengakses *file* dengan format ‘.php’ selain *file* ‘index.php’, di dalam *toolbar* alamat *URL browser* harus dicantumkan nama lengkap *file* yang ingin dijalankan.

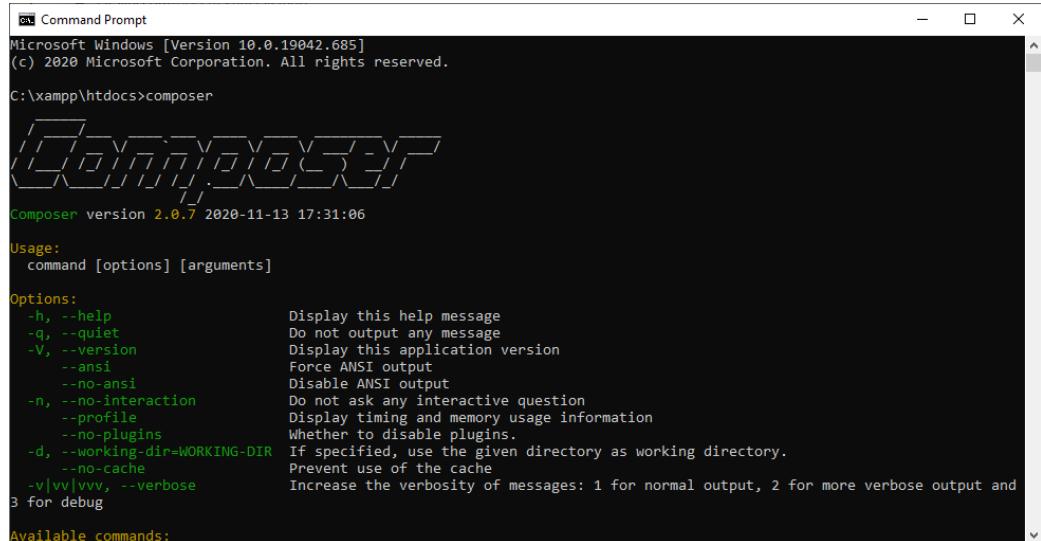


Gambar 4.6 Menjalankan file contact.php pada browser

## 5. PHP sebagai Bahasa Pemrograman dalam Framework Server-Side Aplikasi Web

Selain melalui *Web Server*, kode program *PHP* dapat juga dijalankan melalui perintah *CLI* dengan *syntax* ‘.php’. *Syntax* ini digunakan bila mengembangkan *server-side* aplikasi *web* menggunakan *Framework PHP*, seperti *Laravel*.

Untuk membuat *project Laravel*, harus meng-install *Composer*, yaitu *Dependency Manager* untuk memperoleh *library PHP* melalui *internet*.



```
Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs>composer

Composer version 2.0.7 2020-11-13 17:31:06

Usage:
  command [options] [arguments]

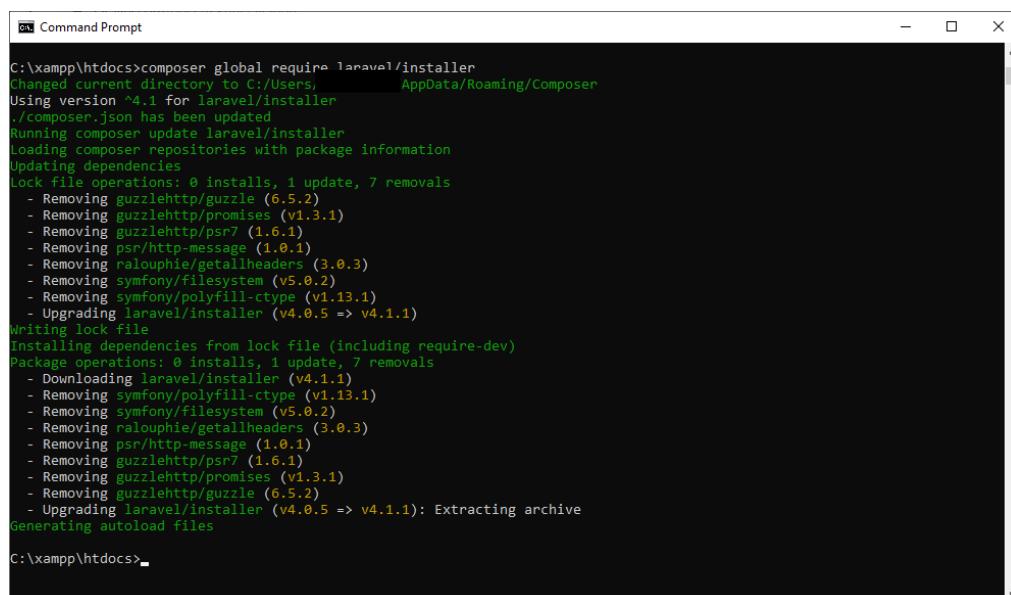
Options:
  -h, --help           Display this help message
  -q, --quiet          Do not output any message
  -V, --version         Display this application version
  --ansi               Force ANSI output
  --no-ansi             Disable ANSI output
  -n, --no-interaction Do not ask any interactive question
  --profile             Display timing and memory usage information
  --no-plugins          Whether to disable plugins.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache            Prevent use of the cache
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
  3 for debug

Available commands:
```

Gambar 5.1 Tampilan aplikasi Composer dijalankan pada Command Prompt

Berikut adalah langkah-langkah untuk membuat project baru *server-side* aplikasi *web* dengan *Framework Laravel*.

- 1) Pastikan *Dependency Manager Composer* ter-install pada perangkat pengembangan Anda. Jalankan *Command Prompt/Terminal*, tulis syntax ‘composer global require laravel/installer’ untuk mendownload *installer Laravel* sebagai *dependency global*. Pastikan pula koneksi internet tersedia pada perangkat pengembangan Anda sebelum menjalankan syntax tersebut.



```
C:\xampp\htdocs>composer global require laravel/installer
Changed current directory to C:/Users/AppData/Roaming/Composer
Using version 4.1 for laravel/installer
./composer.json has been updated
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 7 removals
- Removing guzzlehttp/guzzle (6.5.2)
- Removing guzzlehttp/promises (v1.3.1)
- Removing guzzlehttp/psr7 (1.6.1)
- Removing psr/http-message (1.0.1)
- Removing ralouphie/getallheaders (3.0.3)
- Removing symfony/filesystem (v5.0.2)
- Removing symfony/polyfill ctype (v1.13.1)
- Upgrading laravel/installer (v4.0.5 => v4.1.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 7 removals
- Downloading laravel/installer (v4.1.1)
- Removing symfony/polyfill ctype (v1.13.1)
- Removing symfony/filesystem (v5.0.2)
- Removing ralouphie/getallheaders (3.0.3)
- Removing psr/http-message (1.0.1)
- Removing guzzlehttp/psr7 (1.6.1)
- Removing guzzlehttp/promises (v1.3.1)
- Removing guzzlehttp/guzzle (6.5.2)
- Upgrading laravel/installer (v4.0.5 => v4.1.1): Extracting archive
Generating autoload files

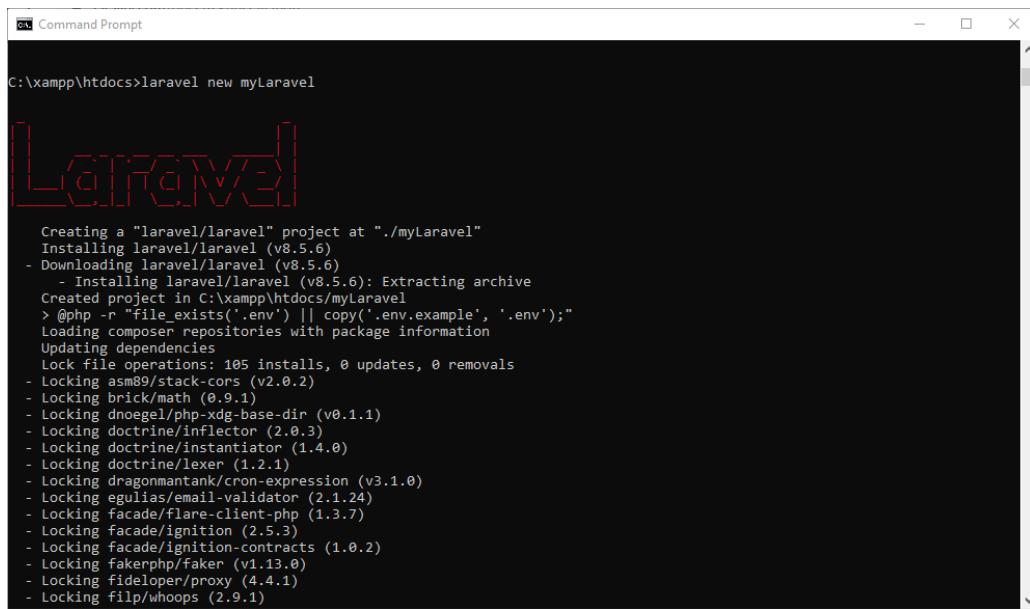
C:\xampp\htdocs>
```

Gambar 5.2 Eksekusi syntax laravel/installer pada Command Prompt

*Installer Laravel* adalah *dependency* yang digunakan untuk memperoleh *template* struktur sumber daya *Framework Laravel* versi

terbaru (pada saat modul ini dibuat versi terbaru adalah Laravel 8).

- 2) Jalankan syntax ‘laravel {namaProject}’ di dalam folder ‘xampp/htdocs’. {namaProject} diisi dengan nama *project* yang ingin dibuat.



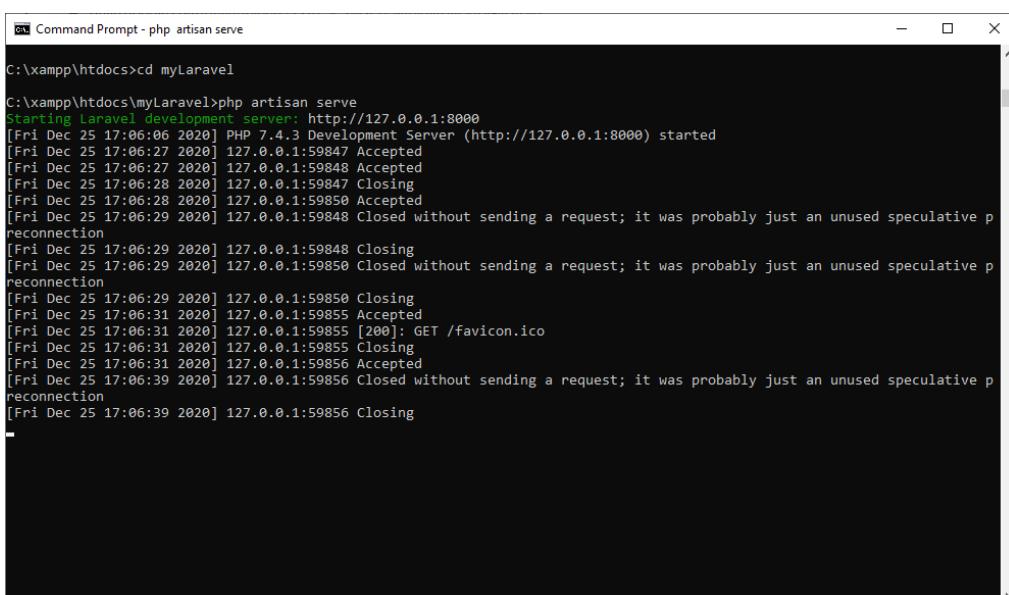
```
C:\xamp\htdocs>laravel new myLaravel

[laravel] Creating a "laravel/laravel" project at "./myLaravel"
[laravel] Installing laravel/laravel (v8.5.6)
[laravel] - Downloading laravel/laravel (v8.5.6)
[laravel]   - Installing laravel/laravel (v8.5.6): Extracting archive
[laravel] Created project in C:\xamp\htdocs\myLaravel
[laravel] > @php -r "file_exists('.env') || copy('.env.example', '.env');"
[laravel] Loading composer repositories with package information
[laravel] Updating dependencies
[laravel] Lock file operations: 105 installs, 0 updates, 0 removals
[laravel] - Locking asm89/stack-cors (v2.0.2)
[laravel] - Locking brick/math (v0.9.1)
[laravel] - Locking dnogeel/php-xdg-base-dir (v0.1.1)
[laravel] - Locking doctrine/inflector (2.0.3)
[laravel] - Locking doctrine/instantiator (1.4.0)
[laravel] - Locking doctrine/lexer (1.2.1)
[laravel] - Locking dragonmantank/cron-expression (v3.1.0)
[laravel] - Locking egilas/email-validator (2.1.24)
[laravel] - Locking facade/flare-client-php (1.3.7)
[laravel] - Locking facade/ignition (2.5.3)
[laravel] - Locking facade/ignition-contracts (1.0.2)
[laravel] - Locking fakerphp/faker (v1.13.0)
[laravel] - Locking fideloper/proxy (4.4.1)
[laravel] - Locking filp/whoops (2.9.1)
```

Gambar 5.3 Eksekusi perintah installer Laravel

Hasil eksekusi pada contoh di atas akan menghasilkan sebuah *folder* bernama ‘myLaravel’ yang berisi semua sumber daya *Framework Laravel* yang terbentuk di dalam *folder* ‘xampp/htdocs’.

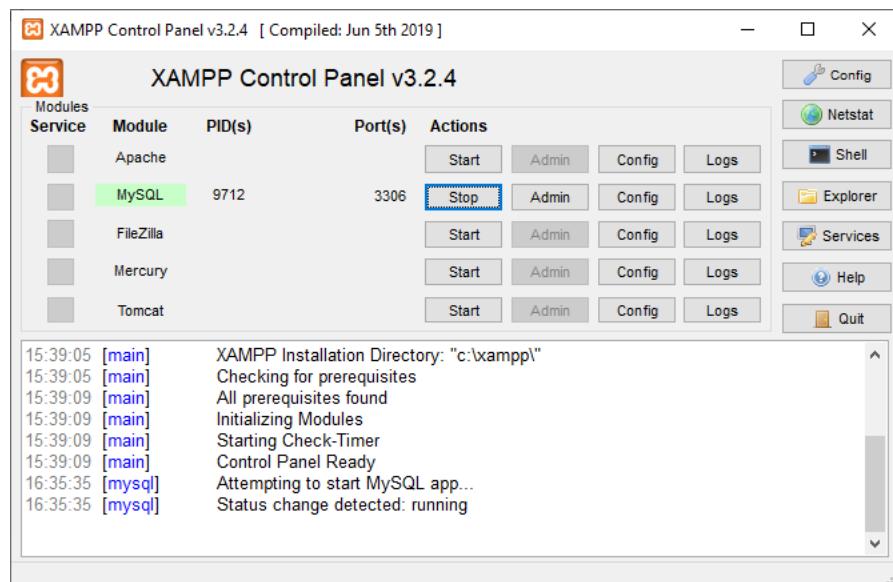
Untuk menjalankan project PHP yang dibuat dengan *Framework Laravel*, jalankan syntax ‘php artisan serve’ pada *folder project*. Perpindahan *folder* pada Command Prompt dapat dilakukan dengan cara menulis syntax ‘cd {alamat tujuan}’.



```
C:\xamp\htdocs>cd myLaravel
C:\xamp\htdocs\myLaravel>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Fri Dec 25 17:06:06 2020] PHP 7.4.3 Development Server (http://127.0.0.1:8000) started
[Fri Dec 25 17:06:27 2020] 127.0.0.1:59847 Accepted
[Fri Dec 25 17:06:27 2020] 127.0.0.1:59848 Accepted
[Fri Dec 25 17:06:28 2020] 127.0.0.1:59847 Closing
[Fri Dec 25 17:06:28 2020] 127.0.0.1:59850 Accepted
[Fri Dec 25 17:06:29 2020] 127.0.0.1:59848 Closed without sending a request; it was probably just an unused speculative p reconnection
[Fri Dec 25 17:06:29 2020] 127.0.0.1:59848 Closing
[Fri Dec 25 17:06:29 2020] 127.0.0.1:59850 Closed without sending a request; it was probably just an unused speculative p reconnection
[Fri Dec 25 17:06:29 2020] 127.0.0.1:59850 Closing
[Fri Dec 25 17:06:31 2020] 127.0.0.1:59855 Accepted
[Fri Dec 25 17:06:31 2020] 127.0.0.1:59855 [200]: GET /favicon.ico
[Fri Dec 25 17:06:31 2020] 127.0.0.1:59855 Closing
[Fri Dec 25 17:06:31 2020] 127.0.0.1:59856 Accepted
[Fri Dec 25 17:06:39 2020] 127.0.0.1:59856 Closed without sending a request; it was probably just an unused speculative p reconnection
[Fri Dec 25 17:06:39 2020] 127.0.0.1:59856 Closing
```

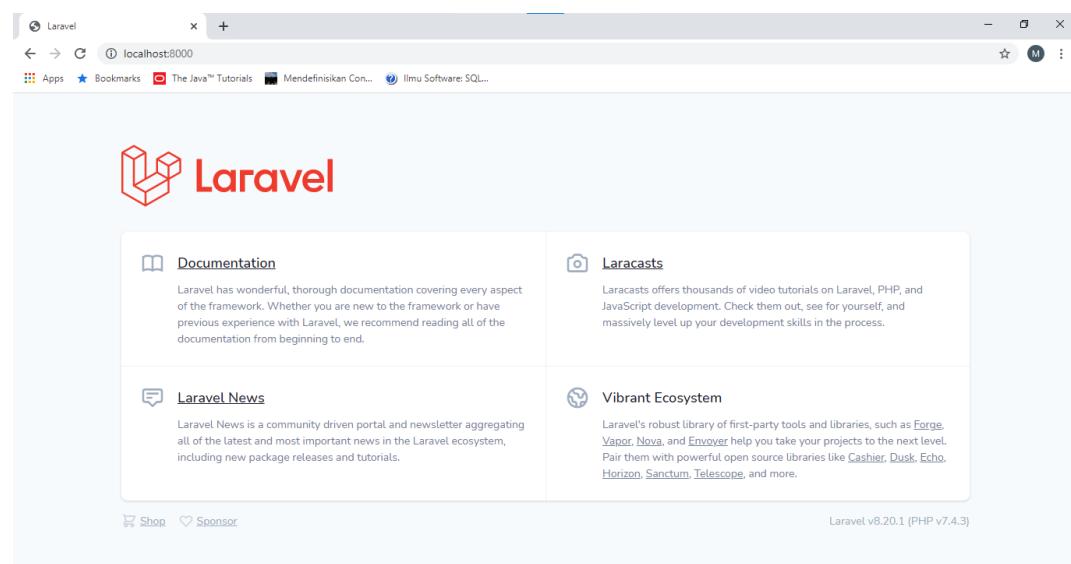
Gambar 5.4 Eksekusi syntax php artisan serve

*Syntax ‘php artisan serve’ akan menjalankan kode program PHP yang mengaktifkan Web Server default Framework Laravel. Sehingga tidak perlu lagi mengaktifkan Web Server Apache melalui XAMPP Control Panel. Jika sudah berhasil menjalankan syntax ‘php artisan serve’, fitur yang perlu diaktifkan melalui XAMPP Control Panel hanya MySQL saja.*



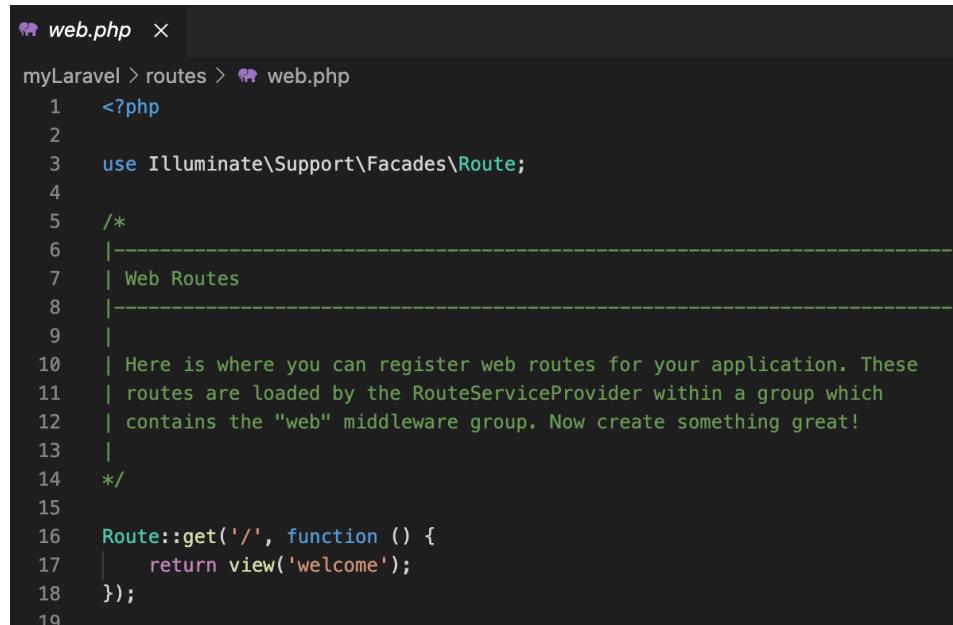
Gambar 5.5 Mengaktifkan MySQL

Secara default, Web Server Laravel dijalankan pada ‘<http://localhost:8000>’. Masukkan alamat tersebut di dalam toolbar alamat URL Web Browser untuk menjalankan aplikasi web yang dikembangkan dengan Framework Laravel.



Gambar 5.6 Hasil eksekusi aplikasi web dengan Framework Laravel 8

Aplikasi *web* yang dikembangkan dengan *Laravel* dijalankan berdasarkan pendefinisian *route* pada file ‘routes/web.php’. Pendefinisian *route* ini menghubungkan *HTTP Request* berupa alamat *URL* yang diminta ke *web server* dengan *HTTP Response* yang dihasilkan oleh *web server*.



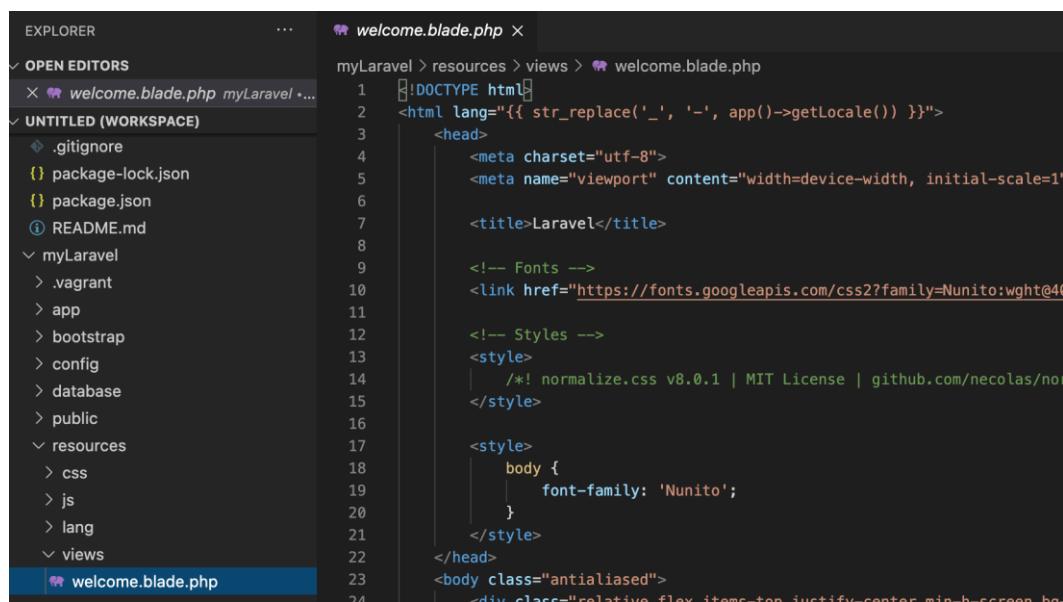
```

routes/web.php ×
myLaravel > routes > routes/web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6  |-----
7  | Web Routes
8  |-----
9  |
10 | Here is where you can register web routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | contains the "web" middleware group. Now create something great!
13 |
14 */
15
16 Route::get('/', function () {
17     return view('welcome');
18 });
19

```

Gambar 5.7 Contoh file routes/web.php

Pada contoh di atas, *HTTP Request* pada alamat *URL* ‘/’ dihubungkan dengan *HTTP Response* berupa *file View* bernama ‘welcome’. *File View* ini terletak pada *folder* ‘resources/views’. Format *file View default* pada *Laravel* adalah ‘blade.php’.



```

EXPLORER ...
OPEN EDITORS
  × welcome.blade.php myLaravel ...
UNTITLED (WORKSPACE)
  .gitignore
  package-lock.json
  package.json
  README.md
  myLaravel
    .vagrant
    app
    bootstrap
    config
    database
    public
  resources
    css
    js
    lang
    views
      welcome.blade.php
welcome.blade.php ×
myLaravel > resources > views > welcome.blade.php
1  <!DOCTYPE html>
2  <html lang="{{ str_replace('_', '-', app()->getLocale()) }}>
3      <head>
4          <meta charset="utf-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1">
6
7          <title>Laravel</title>
8
9          <!-- Fonts -->
10         <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@400,600,700&display=swap" rel="stylesheet">
11
12         <!-- Styles -->
13         <style>
14             /* normalize.css v8.0.1 | MIT License | github.com/necolas/normalize.css */
15         </style>
16
17         <style>
18             body {
19                 font-family: 'Nunito';
20             }
21         </style>
22     </head>
23     <body class="antialiased">
24         <div class="relative flex items-top justify-center min-h-screen bg-gray-100 dark:bg-gray-900 sm:items-center sm:pt-0" style="background-color: #f1f3f4;">

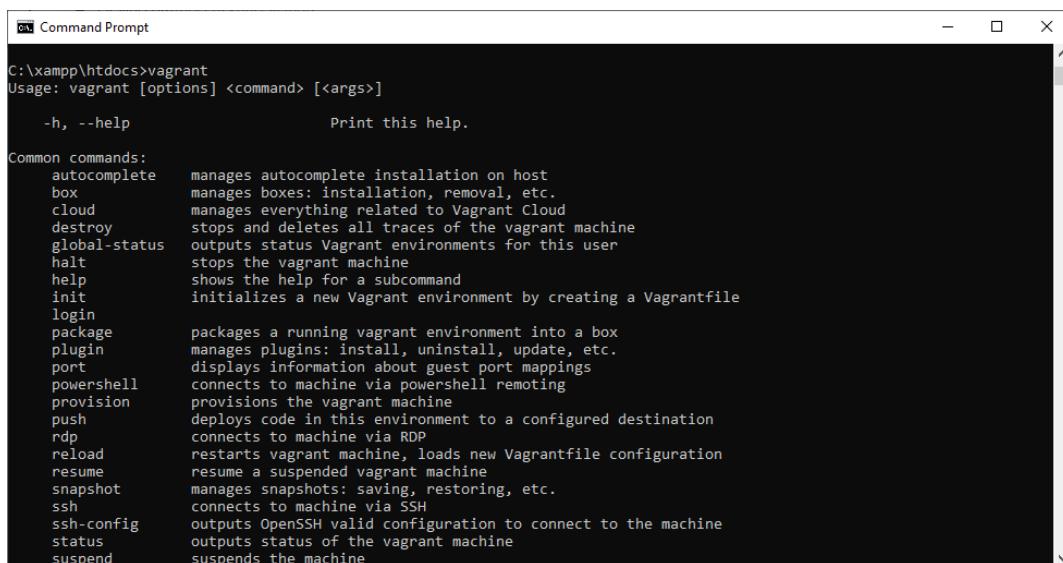
```

Gambar 5.8 Contoh file View welcome.blade.php

Selain melalui *syntax* ‘`php artisan serve`’, *project Laravel* juga dapat

disimpan dan dijalankan di dalam *Virtual Machine/Emulator*, seperti *VirtualBox*. Penggunaan *Virtual Machine* berguna bila kita tidak ingin meng-*install* lingkungan pengembangan *server-side*, seperti *database MySQL*, dan *web server Apache* pada perangkat komputer, melainkan di-*install* di dalam *Virtual Machine*. Sehingga fungsionalitas perangkat komputer dipisahkan antara lingkungan pengembangan aplikasi *web*, dengan lingkungan pekerjaan biasa.

Untuk men-*setting* lingkungan pengembangan aplikasi *web* pada *Virtual Machine* dengan mudah, dapat dilakukan dengan *Laravel Homestead*. Untuk meng-*install* *Laravel Homestead*, perangkat Anda harus ter-*install* *VirtualBox* dan *Vagrant*.



```
C:\xampp\htdocs>vagrant
Usage: vagrant [options] <command> [<args>]

-h, --help                  Print this help.

Common commands:
autocomplete    manages autocomplete installation on host
box             manages boxes: installation, removal, etc.
cloud            manages everything related to Vagrant Cloud
destroy          stops and deletes all traces of the vagrant machine
global-status   outputs status Vagrant environments for this user
halt             stops the vagrant machine
help              shows the help for a subcommand
init              initializes a new Vagrant environment by creating a Vagrantfile
login             connects to a vagrant machine via SSH
package           packages a running vagrant environment into a box
plugin            manages plugins: install, uninstall, update, etc.
port              displays information about guest port mappings
powershell        connects to machine via powershell remoting
provision         provisions the vagrant machine
push              deploys code in this environment to a configured destination
rdp               connects to machine via RDP
reload            restarts vagrant machine, loads new Vagrantfile configuration
resume            resume a suspended vagrant machine
snapshot          manages snapshots: saving, restoring, etc.
ssh               connects to machine via SSH
ssh-config        outputs OpenSSH valid configuration to connect to the machine
status            outputs status of the vagrant machine
suspend           suspends the machine
```

Gambar 5.9 Perintah Vagrant pada Command Prompt

Berikut adalah langkah-langkah untuk memasang *Laravel Homestead* pada sebuah *project Laravel*.

- 1) Pastikan *VirtualBox* dan *Vagrant* sudah ter-*install*. Jalankan *syntax* ‘composer require laravel/homestead --dev’ pada *Command Prompt/Terminal* di *folder project Laravel* yang ingin dipasang *Laravel Homestead*.

```
myLaravel % composer require laravel/homestead --dev
Using version ^11.2 for laravel/homestead
./composer.json has been updated
Running composer update laravel/homestead
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
  - Locking laravel/homestead (v11.2.4)
  - Locking symfony/yaml (v5.2.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
  - Downloading symfony/yaml (v5.2.1)
  - Installing symfony/yaml (v5.2.1): Extracting archive
  - Installing laravel/homestead (v11.2.4): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/sail
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
```

Gambar 5.10 Eksekusi syntax composer require laravel/homestead

Syntax tersebut akan meng-install *Laravel Homestead* sebagai *dependency* khusus pada tahap pengembangan/*development*. Sehingga pada tahap peluncuran/*production* tidak akan ter-install pada *project* tersebut.

- 2) Jalankan *syntax* untuk membuat *folder* ‘vendor/bin/homestead’ pada *project*. Bila perangkat yang Anda gunakan adalah perangkat *Windows*, jalankan *syntax* ‘vendor\\bin\\homestead make’. Dan bila perangkat yang Anda gunakan adalah perangkat *MacOS/Linux*, jalankan *syntax* ‘php vendor/bin/homestead/ make’.

```
myLaravel % php vendor/bin/homestead make
Homestead Installed!
```

Gambar 5.11 Eksekusi syntax php vendor/bin/homestaaed make

Perintah ini akan menghasilkan *file* ‘Homestead.yml’ di dalam *folder project* yang berisi konfigurasi lingkungan pengembangan aplikasi *web* pada *project* tersebut.

```
! Homestead.yaml ×  
myLaravel > ! Homestead.yaml  
1   ip: 192.168.10.10  
2   memory: 2048  
3   cpus: 2  
4   provider: virtualbox  
5   authorize: ~/.ssh/id_rsa.pub  
6   keys:  
7   | - ~/.ssh/id_rsa  
8   folders:  
9   | -  
10  |   map: '/Users/[REDACTED]'  
11  |   to: /home/vagrant/code  
12  sites:  
13  | -  
14  |   map: homestead.test  
15  |   to: /home/vagrant/code/public  
16  databases:  
17  | - homestead  
18  features:  
19  | -  
20  |   mariadb: false  
21  | -  
22  |   ohmyzsh: false  
23  | -  
24  |   webdriver: false  
25  name: mylaravel  
26  hostname: mylaravel  
27
```

Gambar 5.12 File Homestead.yml

Secara *default*, *Web Server Homestead* terkonfigurasi pada alamat IP ‘192.168.10.10’. Alamat IP ini terhubung dengan nama *domain* ‘http://homestead.test’.

- 3) Tambahkan alamat IP dan nama *domain* pada konfigurasi ‘Homestead.yml’ ke dalam file ‘hosts’ perangkat komputer Anda. Bila perangkat Anda adalah perangkat *Windows*, tambahkan alamat IP dan nama *domain* pada file yang berlokasi pada ‘C:\Windows\System32\drivers\etc\hosts’. Dan bila perangkat Anda adalah perangkat *MacOS/Linux*, tambahkan alamat IP dan nama *domain* pada file yang berlokasi pada ‘/etc/hosts’.

```

myLaravel — nano - sudo — 80x24
GNU nano 2.0.6          File: /etc/hosts          Modified

## 
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1      localhost
255.255.255.255 broadcasthost
::1            localhost

192.168.10.10  homestead.test

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No
^C Cancel

```

Gambar 5.13 Menambahkan alamat IP dan domain Homestead

- 4) Jalankan syntax ‘vagrant up’ pada project untuk memasang *project* ke dalam *Virtual Machine* (sebelum menjalankan ‘vagrant up’, pastikan file ‘id\_rsa.pub’ tersedia di dalam folder ‘.ssh’ sebagai *file private key* untuk mengakses *SSH Vagrant*). Bila terjadi *error* terkait *private key*, buatlah *private key SSH* menggunakan *PuTTYgen* atau mengikuti *tutorial* berikut (perangkat *Windows*) <http://backendtime.com/setup-laravel-homestead-windows/#ssh-key> atau (perangkat *MacOS/Linux*) step 5 pada <https://medium.com/fabcoding/set-up-laravel-homestead-on-mac-step-by-step-271a3489ff1>.

```

myLaravel % vagrant up
Bringing machine 'mylaravel' up with 'virtualbox' provider...
==> mylaravel: Importing base box 'laravel/homestead'...
==> mylaravel: Matching MAC address for NAT networking...
==> mylaravel: Checking if box 'laravel/homestead' version '9.6.1' is up to date
...
==> mylaravel: A newer version of the box 'laravel/homestead' for provider 'virtualbox' is
==> mylaravel: available! You currently have version '9.6.1'. The latest is vers
ion
==> mylaravel: '9.7.2'. Run `vagrant box update` to update.
==> mylaravel: Setting the name of the VM: mylaravel
==> mylaravel: Clearing any previously set network interfaces...
==> mylaravel: Preparing network interfaces based on configuration...
mylaravel: Adapter 1: nat
mylaravel: Adapter 2: hostonly
==> mylaravel: Forwarding ports...
mylaravel: 80 (guest) => 8000 (host) (adapter 1)
mylaravel: 443 (guest) => 44300 (host) (adapter 1)
mylaravel: 3306 (guest) => 33060 (host) (adapter 1)
mylaravel: 4040 (guest) => 4040 (host) (adapter 1)
mylaravel: 5432 (guest) => 54320 (host) (adapter 1)
mylaravel: 8025 (guest) => 8025 (host) (adapter 1)
mylaravel: 9600 (guest) => 9600 (host) (adapter 1)

```

Gambar 5.14 Menjalankan vagrant up

Status *project* yang aktif pada *Vagrant* dapat dilihat melalui *syntax* ‘vagrant global-status’.

```
laptop:~ myLaravel % vagrant global-status
      id     name      provider      state      directory
-----
[...]
a5e1405  mylaravel    virtualbox    running   /Users/kemnaker/Documents/Laravel Projects/myLaravel

The above shows information about all known Vagrant environments on this machine. This data is cached and may not be completely up-to-date (use "vagrant global-status --prune" to prune invalid entries). To interact with any of the machines, you can go to that directory and run Vagrant, or you can use the ID directly with Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
```

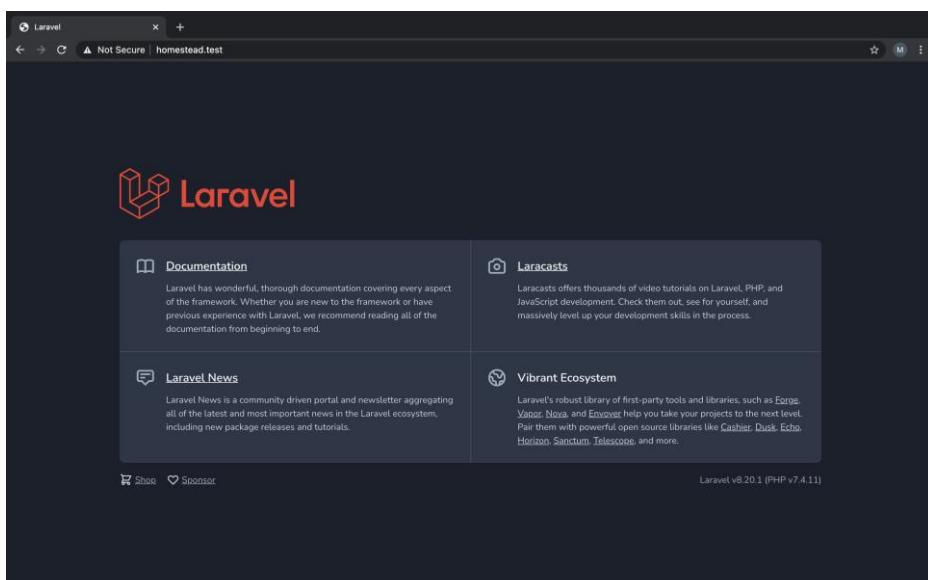
Gambar 5.15 Eksekusi vagrant global-status

Untuk mencopot *project* dari *Vagrant*, jalankan *syntax* ‘vagrant dsestroy {id project Vagrant}’.

```
myLaravel % vagrant destroy a5e1405
mylaravel: Are you sure you want to destroy the 'mylaravel' VM? [y/N] y
==> mylaravel: Forcing shutdown of VM...
==> mylaravel: Destroying VM and associated drives...
```

Gambar 5.16 Mencopot project dari Vagrant

Bila sudah terpasang pada *Vagrant*, *project* dengan *Laravel Homestead* dapat dijalankan melalui nama *domain* sesuai konfigurasi pada file ‘Homestead.yml’.



Gambar 5.17 Hasil eksekusi project Laravel dengan Homestead

Secara *default*, *Laravel Homestead* menggunakan *Nginx* sebagai *Web Server*, dan *MySQL* sebagai *Database*. Dengan menggunakan *Laravel Homestead*, perangkat komputer Anda tidak perlu di-*install XAMPP*, *LAMP*, *WAMP*, *MAMP*, atau di-*install Web Server* dan *Database* tersendiri lagi. Semua kebutuhan lingkungan pengembangan aplikasi *web* sudah disediakan oleh *Laravel Homestead*.

## **6. JavaScript sebagai Bahasa Scripting Client-Side Aplikasi Web**

*JavaScript* adalah Bahasa *Scripting* yang digunakan untuk mengembangkan *client-side* aplikasi berbasis *web*. Bersama dengan *HTML* dan *CSS*, oleh *W3C (World Wide Web Consortium)* *JavaScript* ditetapkan sebagai bahasa standar yang digunakan untuk membangun halaman *website*. Ditanamkan pada *HTML* untuk menghasilkan aplikasi *web* dinamis yang tidak memerlukan pemrosesan data dari *web server*. Dengan kata lain, perintah *scripting* dengan *JavaScript*, berbeda dengan *PHP*, dijalankan pada *Web Browser* pengguna. Umumnya perintah *scripting* dengan *JavaScript* melakukan manipulasi terhadap komponen *HTML*, serta mengirimkan *HTTP Request* dan menerima *HTTP Response* melalui *AJAX* ke *web server*.

Kode program *JavaScript* dijalankan dengan *interpreter JavaScript* yang disebut *JavaScript Engine*. *Interpreter* ini sudah ter-*install* secara *default* pada *Web Browser* seperti *Google Chrome*, *Microsoft Edge*, *Mozilla Firefox*, *Safari*, *Opera*, dll. Sehingga untuk menjalankan kode program *JavaScript* dilakukan dengan mengakses *file* yang ingin dijalankan menggunakan *Web Browser*. Pada *Web Browser*, *JavaScript Engine* ini dijalankan bersamaan dengan *Rendering Engine* untuk menjalankan perintah *HTML*.

*JavaScript Engine* ini mulanya hanyalah sebuah *interpreter* yang menjalankan kode *scripting*, namun belakangan ini berkembang menjadi *JIT (Just-In-Time) Compiler* yaitu *compiler* yang melakukan proses *compiling* pada saat sebuah perintah *script* dilalui *interpreter*.

Berbeda dengan *PHP*, menjalankan *JavaScript* tidak membutuhkan *Web Server*, karena perintah *scripting* *JavaScript* dijalankan langsung oleh *Web Browser* pengguna.

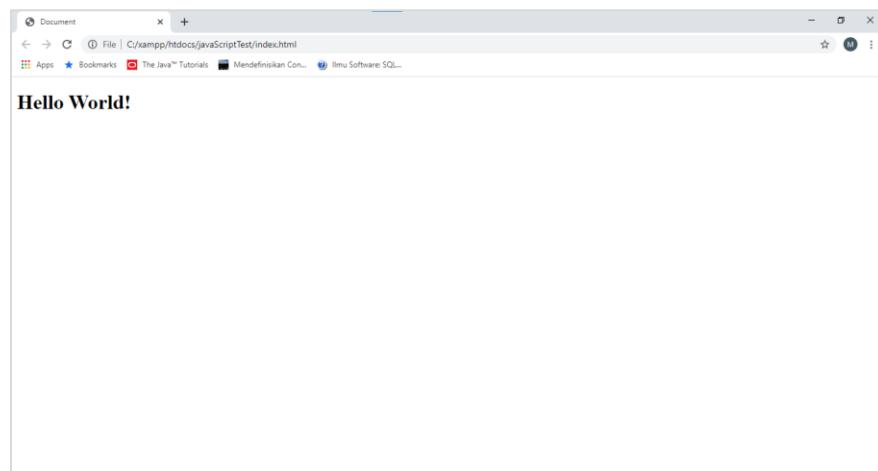
Karena *JavaScript* merupakan bahasa *scripting* untuk *HTML*, perintah *scripting* *JavaScript* harus ditanamkan ke dalam *file* dengan *format* ‘.html’ di dalam *tag <script></script>* pada *HTML*.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1 id="demo"></h1>
    <script>
        document.getElementById("demo").innerHTML = "Hello World!";
    </script>
</body>
</html>
```

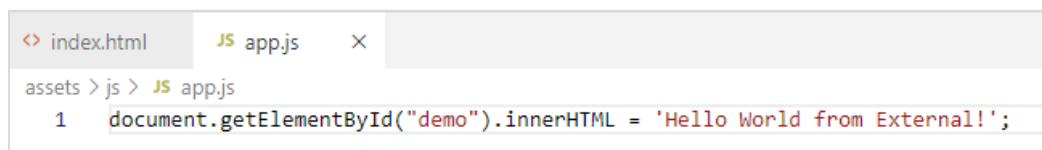
Gambar 6.1 Menanamkan scripting JavaScript di dalam file HTML

*File HTML* yang ditanamkan *scripting JavaScript* dapat langsung dijalankan dengan *Web Browser* dengan menulis alamat fisik *file* tersebut di dalam *toolbar alamat URL Web Browser*.



Gambar 6.2 Menjalankan file HTML yang tertanam scripting JavaScript

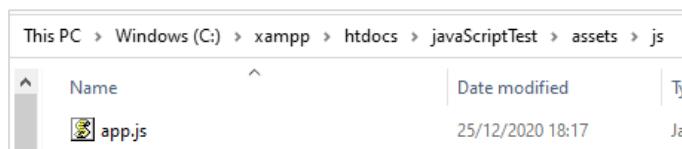
Perintah *scripting JavaScript* juga dapat ditulis terpisah dengan *file HTML* yang ditanamkannya. Umumnya digunakan untuk menulis perintah *scripting JavaScript* yang digunakan oleh lebih dari sebuah *file HTML*. Perintah *scripting JavaScript* ditulis di dalam *file* dengan *format '.js'*.



```
assets > js > JS app.js
1  document.getElementById("demo").innerHTML = 'Hello World from External!';
```

Gambar 6.3 Contoh file scripting JavaScript yang terpisah dengan file HTML

*File scripting JavaScript ini umumnya disimpan di dalam folder ‘assets/js’ pada suatu project.*

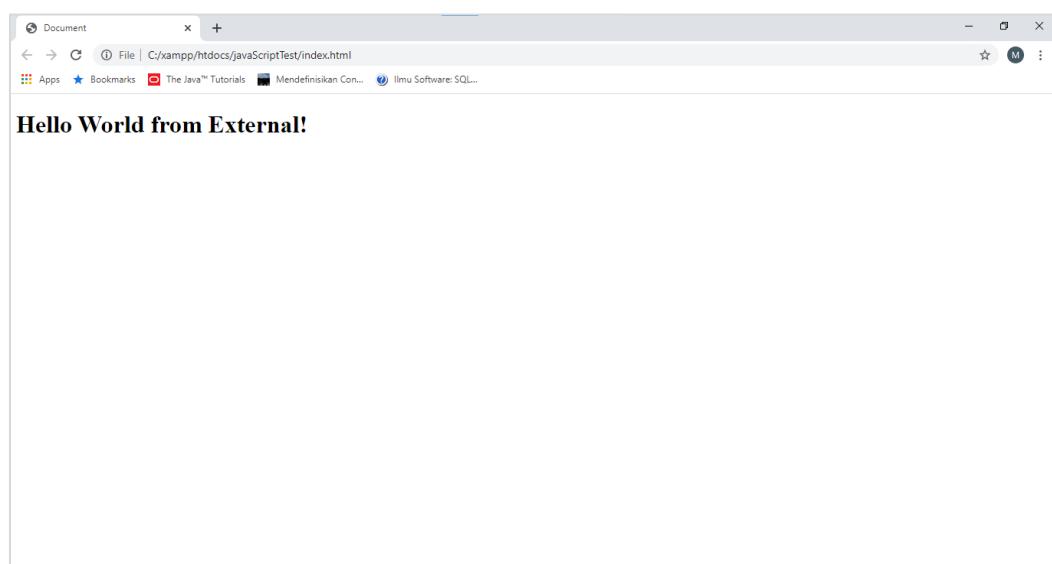


Gambar 6.4 Contoh penempatan file JavaScript pada suatu project

Untuk menghubungkannya dengan dengan *file HTML*, menggunakan *tag <script></script>* dengan atribut ‘src’ yang memiliki *value* alamat fisik *file scripting JavaScript* tersebut. *Tag <script></script>* ditempatkan di dalam elemen *<body>* *HTML*.

```
<body>
  <h1 id="demo"></h1>
  <script src="assets/js/app.js"></script>
</body>
```

Gambar 6.5 Contoh penempatan elemen script di dalam elemen body



Gambar 6.6 Contoh hasil eksekusi dari file JavaScript

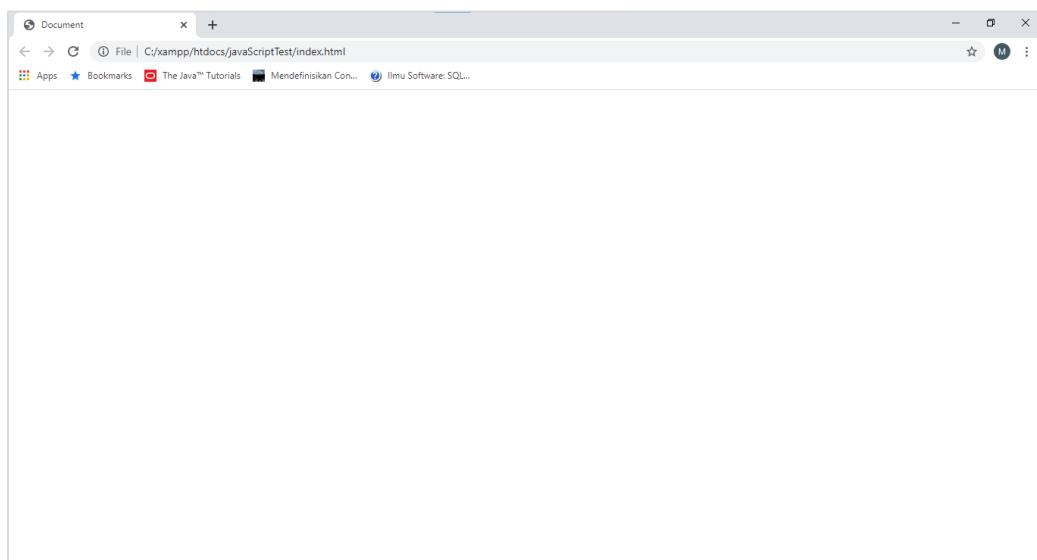
*Tag <script></script>* juga dapat ditempatkan di dalam elemen *<head>* *HTML*.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="assets/js/app.js"></script>
</head>
```

Gambar 6.7 Contoh penempatan elemen script di dalam elemen head

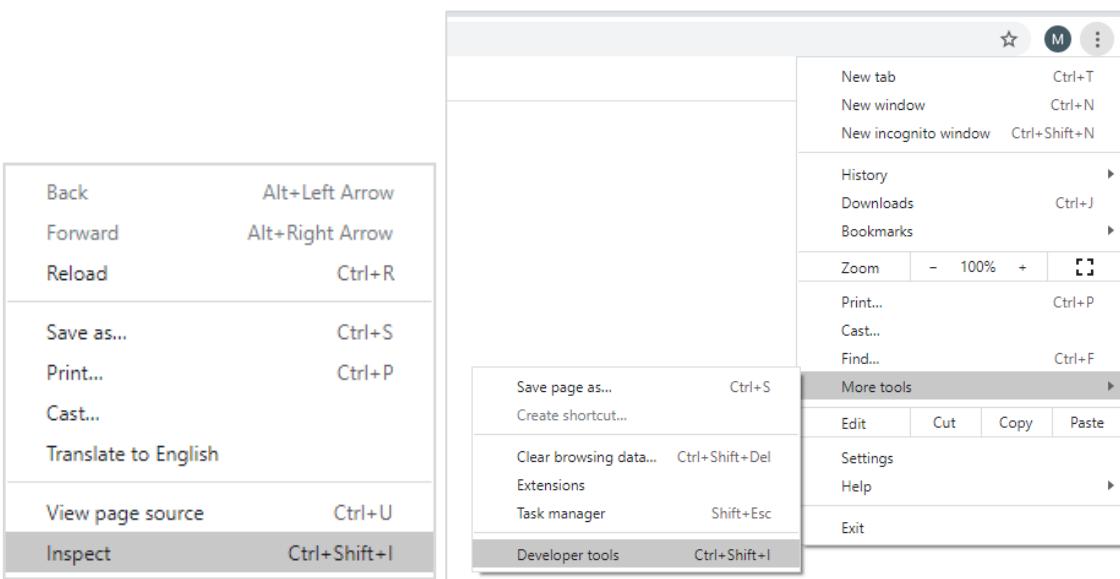
Perlu diperhatikan, ketika menempatkan *tag <script></script>*. Karena

*JavaScript* merupakan bahasa *scripting*, kode program dengan *JavaScript* akan dibaca satu per satu dari baris pertama di atas sampai baris terakhir di bawah. Sehingga beberapa perintah *scripting* harus ditempatkan setelah elemen *HTML* yang ingin dimanipulasinya. Pada contoh penempatan tag `<script></script>` di dalam elemen `<head>` *HTML* di atas, hasil eksekusi tidak akan memunculkan apapun di dalam layar *Web Browser* karena elemen yang ingin dimanipulasi terdapat di bawah tag `<script></script>`.



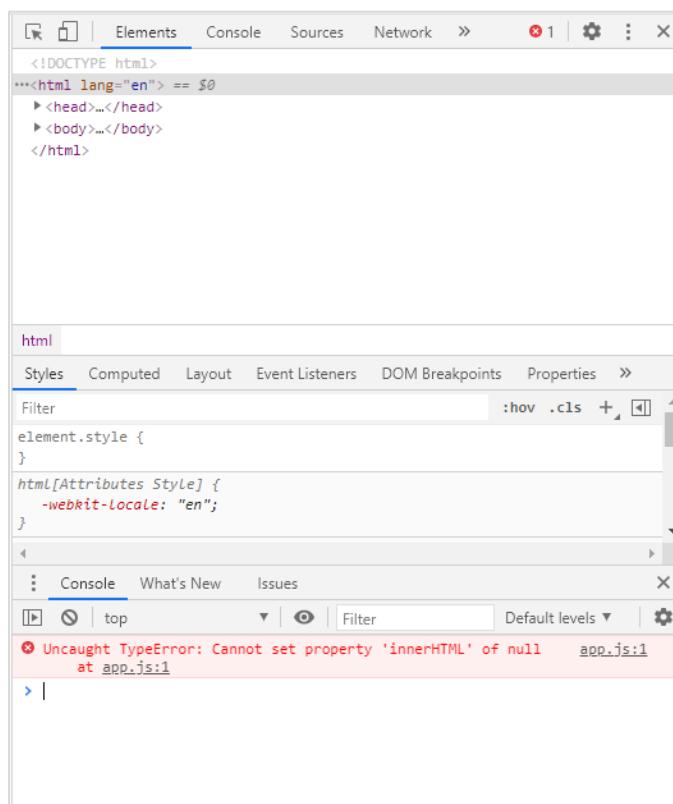
Gambar 6.8 Contoh hasil eksekusi bila elemen HTML yang ingin dimanipulasi terdapat di bawah tag script

Selain sebagai *tools* untuk menjalankan kode program *JavaScript*, *Web Browser* juga dapat digunakan sebagai *tools Debugging* kode program *JavaScript*. Fitur Debugging ini dapat diakses melalui fitur Developer Tools. Setiap *Web Browser* memiliki langkah yang berbeda untuk menjalankan fitur ini. Bila menggunakan Google Chrome, dapat dijalankan dengan klik kanan layar browser → Inspect, atau menekan tombol Menu → More Tools → Developer tools.



Gambar 6.9 Menjalankan fitur Developer Tools menggunakan Google Chrome

Melalui fitur *Developer Tools*, dapat melihat dan memodifikasi kode *HTML*, *CSS* dan juga melihat pesan *error* (pada bagian console) sebagai hasil *Debugging* dari *scripting JavaScript*.



Gambar 6.10 Contoh tampilan fitur Developer Tools Google Chrome

Perintah *scripting JavaScript* juga dapat ditanamkan ke dalam *file* dengan *format* '.php' yang memiliki *syntax* kode *HTML*. Namun, bila ditanamkan pada *file* dengan *format* '.php', untuk menjalankannya

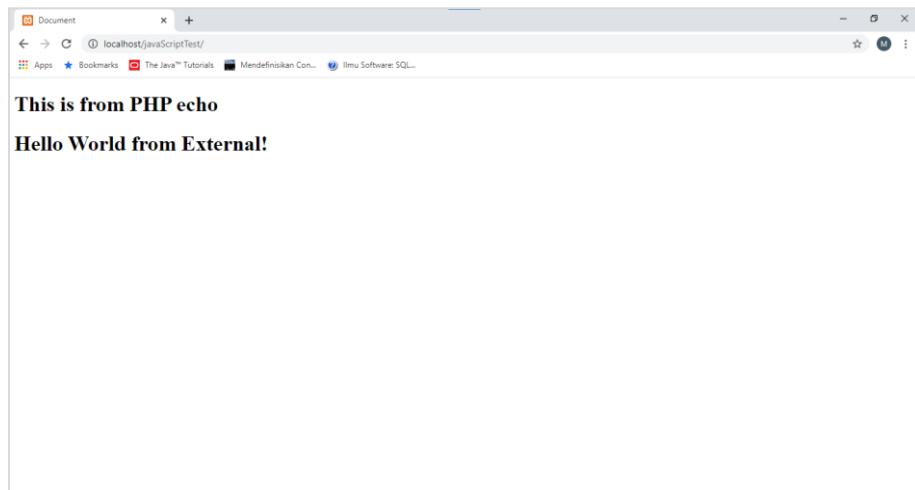
membutuhkan *Web Server* karena merupakan *file* kode sumber untuk Bahasa Pemrograman *PHP*.



```
index.php <pre><?php echo '<h1>This is from PHP echo</h1>' ?>
<h1 id="demo"></h1>
<script src="assets/js/app.js"></script>
<h1>Hello World from External!</h1></pre>
```

Gambar 6.11 Contoh menanamkan scripting JavaScript ke dalam file PHP

Bila menggunakan *XAMPP* sebagai *bundle Web Server* dan *Database*, *file PHP* ini harus ditempatkan di dalam *folder* ‘xampp/htdocs’, dan diakses melalui alamat ‘<http://localhost>’.



Gambar 6.12 Contoh hasil eksekusi file PHP yang ditanamkan scripting JavaScript

Pada contoh di atas menghasilkan 2 elemen `<h1>` HTML yang diproses oleh 2 aplikasi yang berbeda. Elemen `<h1>` yang pertama dengan tulisan ‘This is from PHP echo’ berasal dari perintah ‘echo’ Bahasa Pemrograman *PHP* yang diproses oleh *Web Server Apache*. Sedangkan elemen `<h1>` yang kedua dengan tulisan ‘Hello World from External’ berasal dari perintah ‘innerHTML’ *scripting JavaScript* yang diproses oleh *Web Browser Google Chrome*.

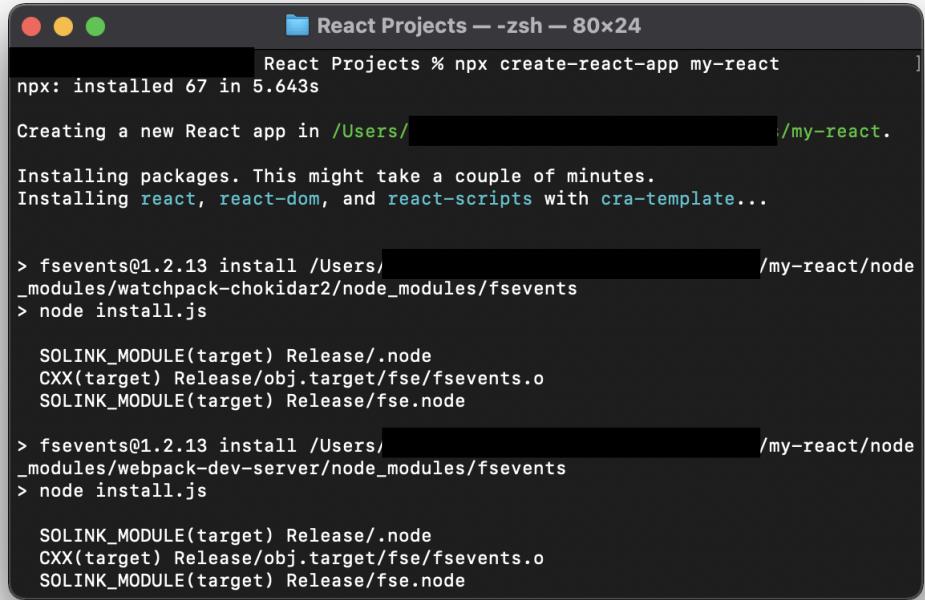
## **7. JavaScript sebagai Bahasa Pemrograman dalam Framework Client-Side Aplikasi Web**

Beberapa tahun ini, pengguna *JavaScript* meningkat pesat. Karena fungsinya tidak terbatas pada *scripting HTML*, tapi juga berkembang menjadi bahasa utama dalam sebuah *Framework* untuk *Client-Side*. Contohnya adalah *React.js*. *React.js* adalah *Framework JavaScript* yang dikembangkan oleh *Facebook* dan komunitas *open-source*. Tidak hanya sebagai *Framework Client-Side* aplikasi *web*, *React.js* juga dapat dijalankan untuk *Server-Side* menggunakan *Node.js*, bahkan dapat dijalankan sebagai aplikasi *mobile* pada *Android* dan *iOS* dengan *React Native*.

Keunggulan dari *React.js* adalah kemampuannya dalam mendeklarasikan komponen halaman *web* yang interaktif sehingga dapat menghasilkan *Progressive Web Application (PWA)*. Selain itu, dengan *React.js* juga dapat membuat *Single Page Application (SPA)* yaitu aplikasi yang hanya membutuhkan satu halaman sehingga memberikan *User Experience* yang tinggi pada pengguna.

Untuk dapat memulai membuat *project React.js*, perangkat pengembangan harus ter-*install Node.js* dan *npm*. *Node.js* adalah salah satu *JavaScript Engine* yang memungkinkan suatu perangkat untuk menjalankan *JavaScript* pada lingkungan *server-side*. Sedangkan *npm (Node Package Manager)* adalah *Dependency Manager* untuk Bahasa Pemrograman *JavaScript*.

Untuk membuat *project React.js*, jalankan *syntax* ‘*npx create-react-app {nama-project}*’ pada *Command Prompt/Terminal* di *folder* yang ingin ditambahkan *project React.js* baru. Pastikan koneksi internet tersedia pada perangkat pengembangan Anda sebelum menjalankan *syntax* tersebut.



```
React Projects — zsh — 80x24
React Projects % npx create-react-app my-react
npx: installed 67 in 5.643s
Creating a new React app in /Users/[REDACTED]/my-react.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

> fsevents@1.2.13 install /Users/_modules/watchpack-chokidar2/node_modules/fsevents
> node install.js

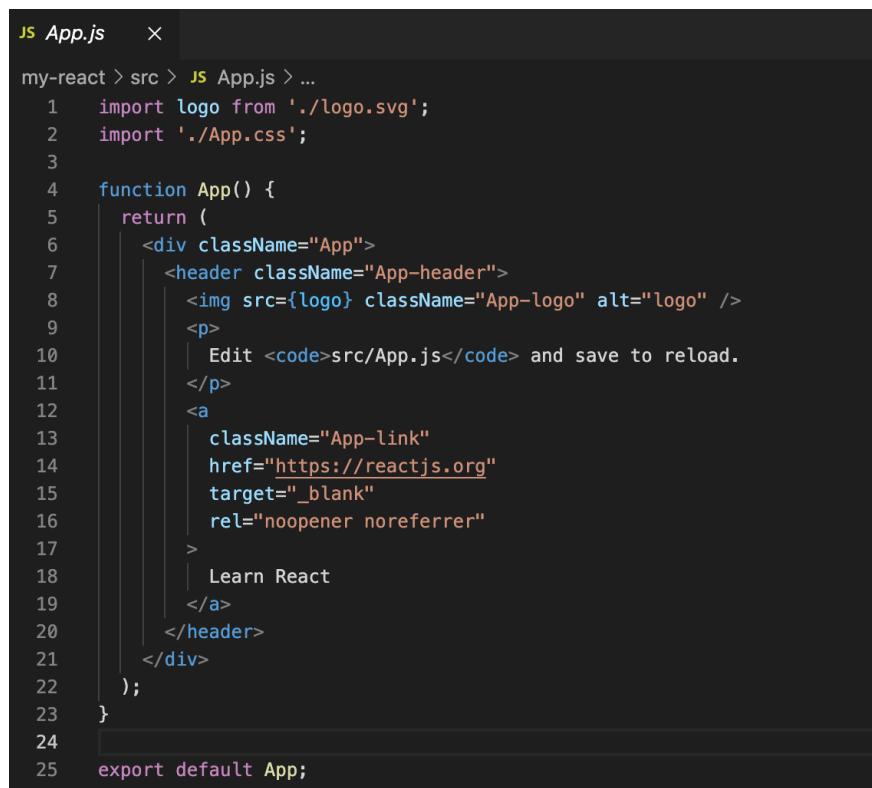
  SOLINK_MODULE(target) Release/.node
  CXX(target) Release/obj.target/fse/fsevents.o
  SOLINK_MODULE(target) Release/fse.node

> fsevents@1.2.13 install /Users/_modules/webpack-dev-server/node_modules/fsevents
> node install.js

  SOLINK_MODULE(target) Release/.node
  CXX(target) Release/obj.target/fse/fsevents.o
  SOLINK_MODULE(target) Release/fse.node
```

Gambar 7.1 Eksekusi syntax `npx create-react-app`

Syntax tersebut akan menghasilkan *folder* sesuai dengan nama *project* yang berisi semua sumber daya project *React.js*. Secara *default project* *React.js* akan berisi *file JavaScript* dengan nama ‘App.js’ sebagai *component* utama aplikasi *React.js* dan *file* ‘index.js’ sebagai *file* utama yang memanggil *component* ‘App’ di dalam *folder* ‘src’.



```
JS App.js  ×

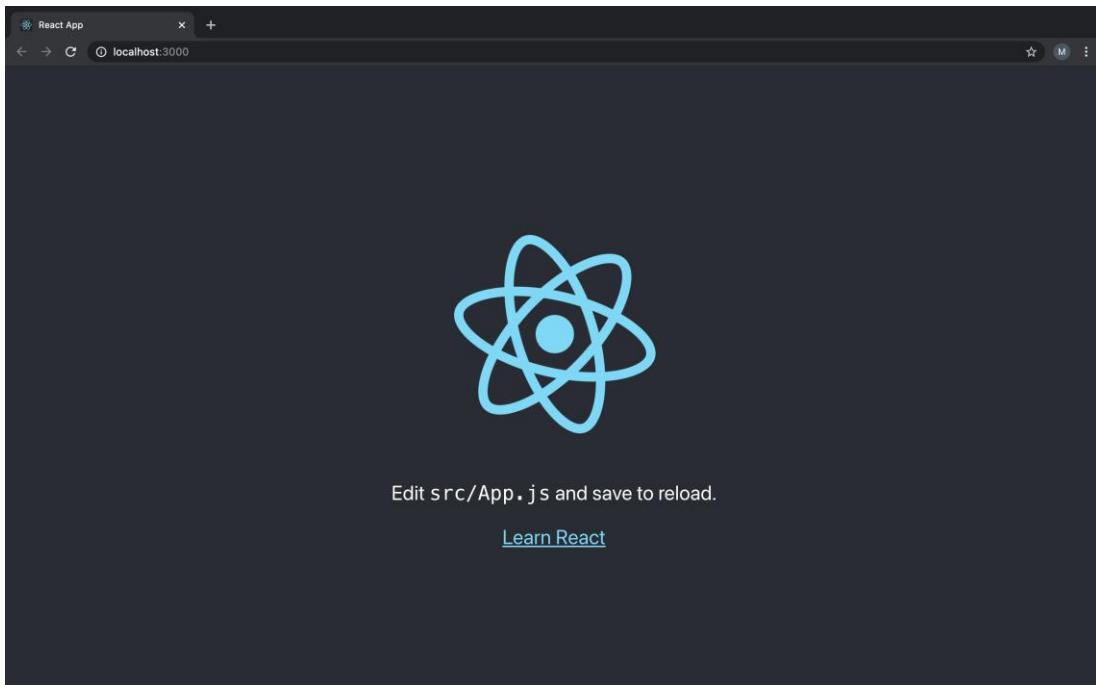
my-react > src > JS App.js > ...
1  import logo from './logo.svg';
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>
10            | Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13            | className="App-link"
14            | href="https://reactjs.org"
15            | target="_blank"
16            | rel="noopener noreferrer"
17          >
18            |   Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24
25  export default App;
```

Gambar 7.2 Tampilan awal file App.js

```
JS index.js  ×  
my-react > src > JS index.js  
1 import React from 'react';  
2 import ReactDOM from 'react-dom';  
3 import './index.css';  
4 import App from './App';  
5 import reportWebVitals from './reportWebVitals';  
6  
7 ReactDOM.render(  
8   <React.StrictMode>  
9     <App />  
10    </React.StrictMode>,  
11    document.getElementById('root')  
12  );  
13  
14 // If you want to start measuring performance in your app, pass a function  
15 // to log results (for example: reportWebVitals(console.log))  
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
17 reportWebVitals();  
18
```

Gambar 7.3 Tampilan awal file index.js

Untuk menjalankan aplikasi *React.js*, jalankan syntax ‘npm start’ pada *Command Prompt/Terminal* di *folder project React.js*. Secara *default* aplikasi *React.js* akan dijalankan pada alamat ‘http://localhost:3000’.

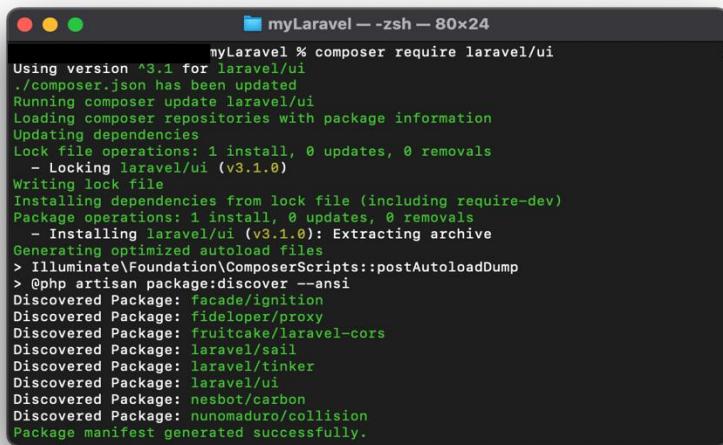


Gambar 7.4 Hasil eksekusi npm start pada project React.js

Selain sebagai aplikasi yang terpisah, *React.js* juga dapat ditanamkan pada *project* aplikasi web yang sudah ada. Contohnya, *React.js* dapat ditanamkan pada *project Laravel* sebagai *scaffolding* untuk *Client-Side*. *Scaffolding Client-Side Laravel* dengan *React.js* dapat dengan mudah diterapkan menggunakan *library* ‘laravel/ui’. Berikut adalah langkah-langkah untuk menanamkan *React.js* pada *project Laravel* sebagai

*scaffolding Client-Side.*

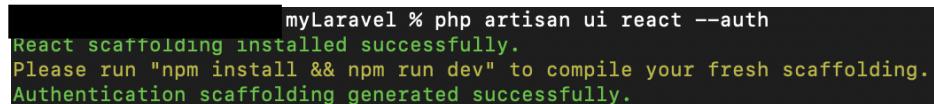
- 1) Tambahkan *library* ‘laravel/ui’ pada *project Laravel* menggunakan *Composer*. Jalankan syntax ‘composer require laravel/ui’ pada *folder project Laravel*.



```
myLaravel % composer require laravel/ui
Using version ^3.1 for laravel/ui
./composer.json has been updated
Running composer update laravel/ui
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/ui (v3.1.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Installing laravel/ui (v3.1.0): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/sail
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
```

Gambar 7.5 Eksekusi syntax composer require laravel/ui

- 2) Jalankan syntax ‘php artisan ui react --auth’ untuk menanamkan *React.js* serta membuat halaman untuk fitur otentikasi pengguna.



```
myLaravel % php artisan ui react --auth
React scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.
```

Gambar 7.6 Eksekusi syntax php artisan ui react –auth

*Syntax ini akan menghasilkan file ‘app.js’ dan folder ‘components’ di dalam folder ‘resources/js’, serta menambahkan file View untuk fitur otentikasi pada folder ‘resources/views’.*

- 3) Jalankan syntax ‘npm install && npm run dev’ pada *folder project Laravel* untuk melakukan *scaffolding* *React.js*. *Syntax ini akan menghasilkan file ‘app.js’ pada folder ‘public/js’ dan file ‘app.css’ pada folder ‘public/css’.*

Dengan melakukan *scaffolding*, *component* yang didefinisikan dengan *React.js* dapat digunakan pada *file View project Laravel*.

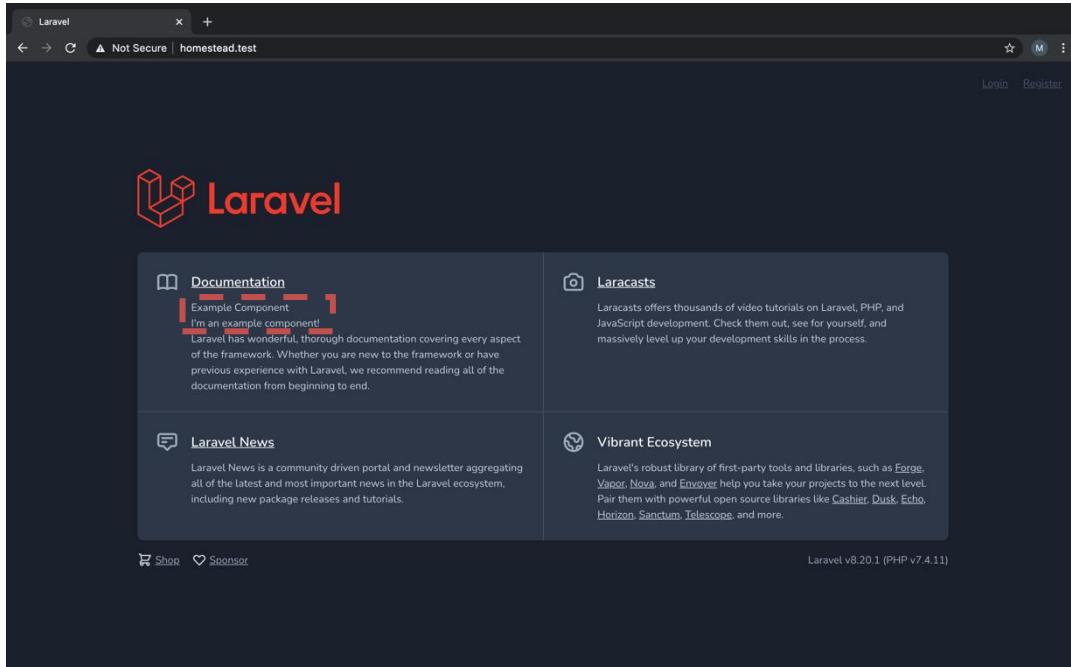
```
myLaravel > resources > js > components > JS Example.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3
4 function Example() {
5     return (
6         <div className="container">
7             <div className="row justify-content-center">
8                 <div className="col-md-8">
9                     <div className="card">
10                         <div className="card-header">Example Component</div>
11                         <div className="card-body">I'm an example component!</div>
12                     </div>
13                 </div>
14             </div>
15         </div>
16     );
17 }
18
19
20 export default Example;
21
22 if (document.getElementById('example')) {
23     ReactDOM.render(<Example />, document.getElementById('example'));
24 }
```

Gambar 7.7 Contoh component yang didefinisikan dengan React.js

```
myLaravel > resources > views > 🐻 welcome.blade.php
30
31
32         @if (Route::has('register'))
33             <a href="{{ route('register') }}" class="ml-4 text-gray-600 dark:text-gray-400" data-bbox="380 108 977 125" data-label="Text">
34                 Register
35             @endif
36         @endauth
37     </div>
38     @endif
39     <div class="max-w-6xl mx-auto sm:px-6 lg:px-8">
40         <div class="flex justify-center pt-8 sm:justify-start sm:pt-0" data-bbox="380 380 977 400" data-label="Text">
41             <div class="flex flex-col items-center" data-bbox="380 400 977 500" data-label="Text">
42                 <img alt="Laravel logo" data-bbox="380 400 977 500" data-label="Image"/>
43             </div>
44         </div>
45     </div>
46
47     <div class="mt-8 bg-white dark:bg-gray-800 overflow-hidden shadow-lg rounded-lg" data-bbox="380 600 977 700" data-label="Text">
48         <div class="grid grid-cols-1 md:grid-cols-2" data-bbox="380 650 977 700" data-label="Text">
49             <div class="p-6" data-bbox="380 650 600 700" data-label="Text">
50                 <div class="flex items-center" data-bbox="380 680 600 750" data-label="Text">
51                     <img alt="User icon" data-bbox="380 680 420 750" data-label="Image"/>
52                     <div class="ml-4 text-lg leading-7 font-semibold" data-bbox="420 680 600 750" data-label="Text">
53                         {{ $user->name }}
54                     </div>
55                 </div>
56             <div class="ml-12" data-bbox="600 650 977 700" data-label="Text">
57                 <div class="mt-2 text-gray-600 dark:text-gray-400" data-bbox="600 700 977 750" data-label="Text">
58                     <div id="example"></div>
59                 </div>
60             </div>
61         </div>
62     </div>
63 
```

Gambar 7.8 Contoh penggunaan component example pada file View welcome.blade.php

Untuk menjalankan *project Laravel* yang ditanamkan *React.js*, dapat dilakukan dengan syntax ‘`php artisan serve`’ maupun menjalankannya melalui *Virtual Machine Vagrant* yang diintegrasikan dengan *Laravel Homestead*.



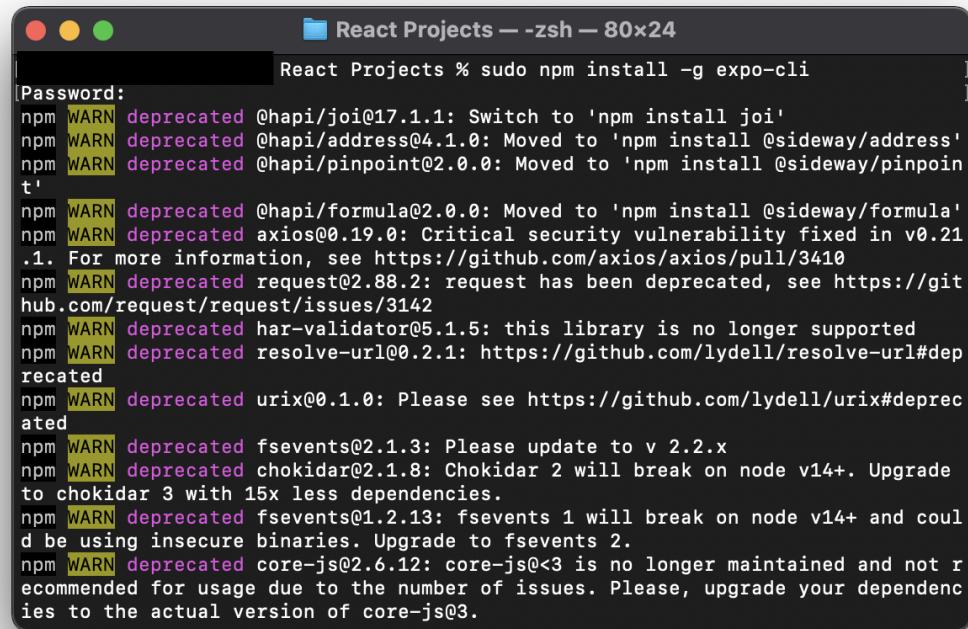
Gambar 7.9 Contoh hasil menjalankan aplikasi Laravel yang ditanamkan React.js

## 8. JavaScript sebagai Bahasa Pemrograman dalam Framework Aplikasi Mobile

Penggunaan *JavaScript* untuk mengembangkan aplikasi *mobile* juga dapat dilakukan menggunakan *Framework React Native*. Kelebihan dari menggunakan React Native adalah menghasilkan aplikasi *mobile hybrid*, yaitu aplikasi *mobile* yang dapat dijalankan pada *platform Android* maupun *iOS*.

Terdapat 2 cara untuk memulai project *React Native*. Yaitu menggunakan *Expo CLI* dan *React Native CLI*. Pada modul ini akan diperkenalkan memulai *project React Native* dengan *Expo CLI*. Untuk penjelasan mengenai memulai *project React Native* dengan *React Native CLI* dapat Anda baca pada dokumentasi *official React Native* berikut <https://reactnative.dev/docs/environment-setup>. Pastikan *platform perangkat pengembangan* dan target perangkat *mobile* yang ingin dikembangkan ketika menggunakan *React Native CLI* untuk memulai *project React Native*.

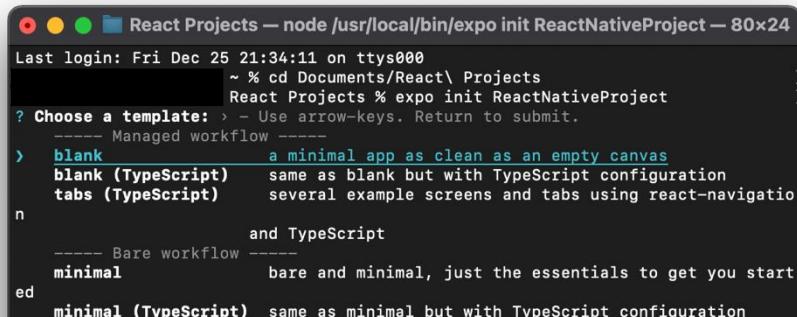
Untuk dapat menggunakan *Expo CLI*, pastikan *Node.js* dan *npm* sudah ter-*install*. Pastikan pula koneksi internet tersedia pada perangkat pengembangan Anda. Jalankan *syntax* ‘*npm install -g expo-cli*’ untuk meng-*install* *Expo CLI*.



```
React Projects — zsh — 80x24
Password: 
npm WARN deprecated @hapi/joi@17.1.1: Switch to 'npm install joi'
npm WARN deprecated @hapi/address@4.1.0: Moved to 'npm install @sideway/address'
npm WARN deprecated @hapi/pinpoint@2.0.0: Moved to 'npm install @sideway/pinpoint'
npm WARN deprecated @hapi/formula@2.0.0: Moved to 'npm install @sideway/formula'
npm WARN deprecated axios@0.19.0: Critical security vulnerability fixed in v0.21
.1. For more information, see https://github.com/axios/axios/pull/3410
npm WARN deprecated request@2.88.2: request has been deprecated, see https://git
hub.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#dep
recated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprec
ated
npm WARN deprecated fsevents@2.1.3: Please update to v 2.2.x
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade
to chokidar 3 with 15x less dependencies.
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and cou
ld be using insecure binaries. Upgrade to fsevents 2.
npm WARN deprecated core-js@2.6.12: core-js@<3 is no longer maintained and not r
ecommended for usage due to the number of issues. Please, upgrade your dependenc
ies to the actual version of core-js@3.
```

Gambar 8.1 Eksekusi npm install -g expo-cli

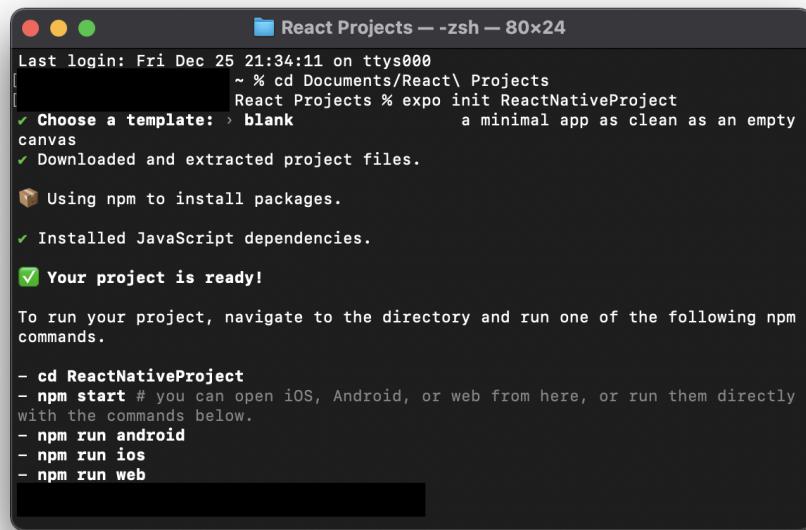
Jalankan *syntax* ‘expo init {NamaProject}’ untuk memulai *project* baru *React Native* dengan *Expo CLI*.



```
React Projects — node /usr/local/bin/expo init ReactNativeProject — 80x24
Last login: Fri Dec 25 21:34:11 on ttys000
~ % cd Documents/React\ Projects
React Projects % expo init ReactNativeProject
? Choose a template: > - Use arrow-keys. Return to submit.
---- Managed workflow ----
> blank a minimal app as clean as an empty canvas
blank (TypeScript) same as blank but with TypeScript configuration
tabs (TypeScript) several example screens and tabs using react-navigatio
n
and TypeScript
---- Bare workflow ----
minimal bare and minimal, just the essentials to get you start
ed
minimal (TypeScript) same as minimal but with TypeScript configuration
```

Gambar 8.2 Eksekusi expo init

Pilih *template project*.



```
Last login: Fri Dec 25 21:34:11 on ttys000
[ ~ % cd Documents/React\ Projects
[ React Projects % expo init ReactNativeProject
[ ✓ Choose a template: > blank           a minimal app as clean as an empty
  canvas
✓ Downloaded and extracted project files.

📦 Using npm to install packages.

✓ Installed JavaScript dependencies.

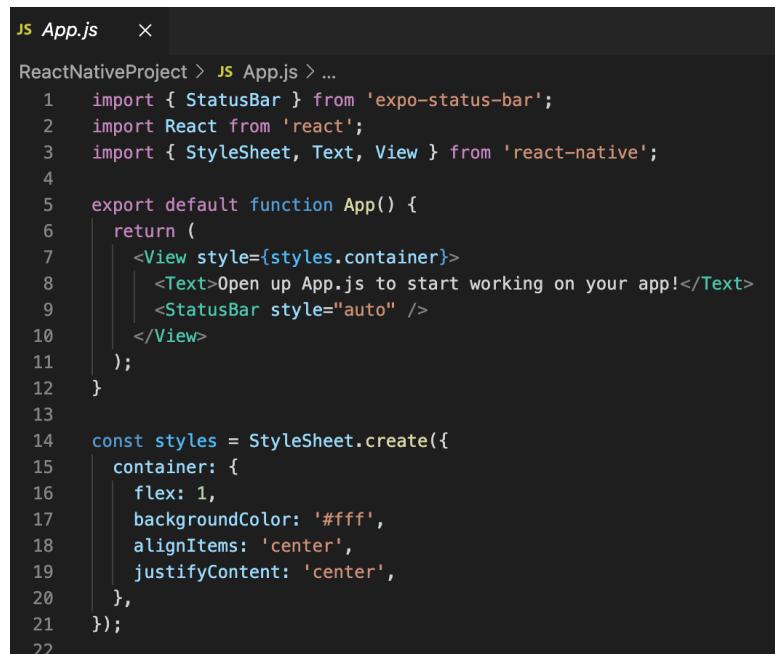
✓ Your project is ready!

To run your project, navigate to the directory and run one of the following npm
commands.

- cd ReactNativeProject
- npm start # you can open iOS, Android, or web from here, or run them directly
with the commands below.
- npm run android
- npm run ios
- npm run web
```

Gambar 8.3 Eksekusi expo init setelah memilih template project

Syntax tersebut menghasilkan *folder* dengan nama *project* yang berisi sumber daya *project React Native*. File utama *project React Native* bernama ‘App.js’.

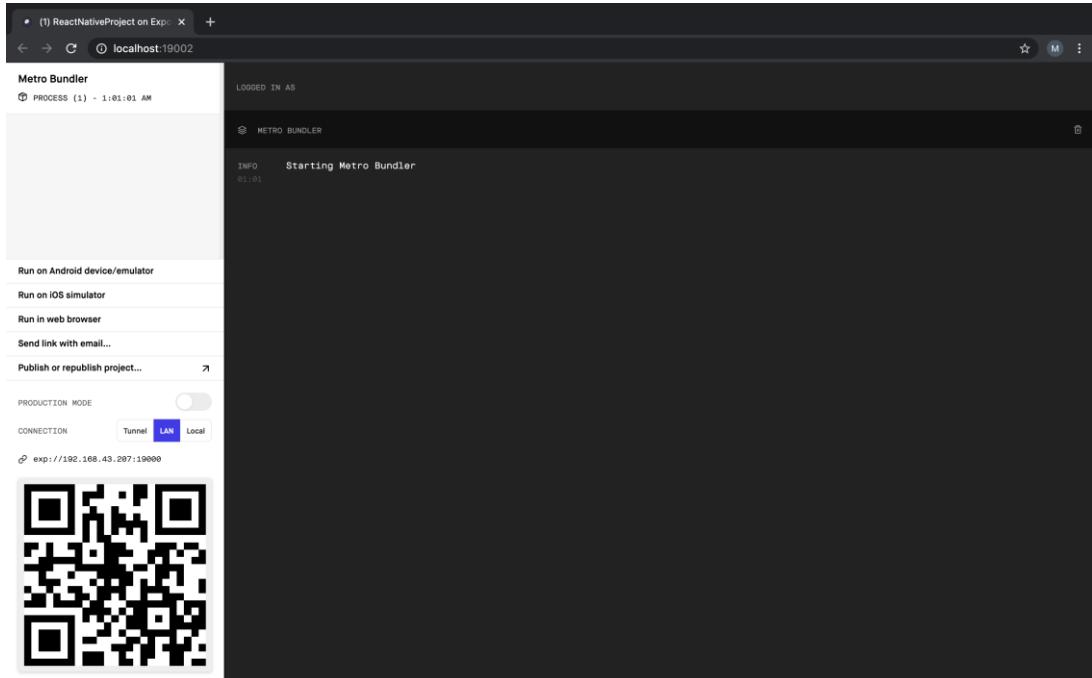


```
JS App.js  ×

ReactNativeProject > JS App.js > ...
1 import { StatusBar } from 'expo-status-bar';
2 import React from 'react';
3 import { StyleSheet, Text, View } from 'react-native';
4
5 export default function App() {
6   return (
7     <View style={styles.container}>
8       <Text>Open up App.js to start working on your app!</Text>
9       <StatusBar style="auto" />
10    </View>
11  );
12}
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });
22
```

Gambar 8.4 Tampilan awal file App.js pada project React Native

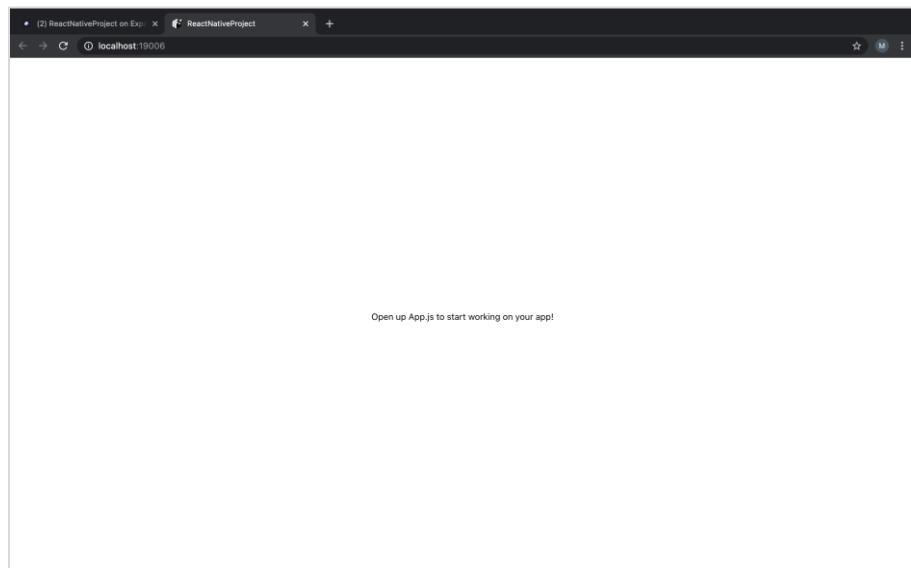
Untuk menjalankan aplikasi *mobile* yang dibuat menggunakan *React Native*, jalankan syntax ‘npm start’ pada *folder project React Native*, sehingga memunculkan aplikasi *Expo* seperti berikut pada *Web Browser*.



Gambar 8.5 Aplikasi Expo pada Web Browser

Melalui aplikasi *Expo* ini, dapat menjalankan aplikasi *mobile* pada *Web Browser*, pada perangkat fisik *mobile* baik *Android* maupun *iOS*, dan *simulator* perangkat *mobile* seperti *iOS Simulator* serta *Android Virtual Device*.

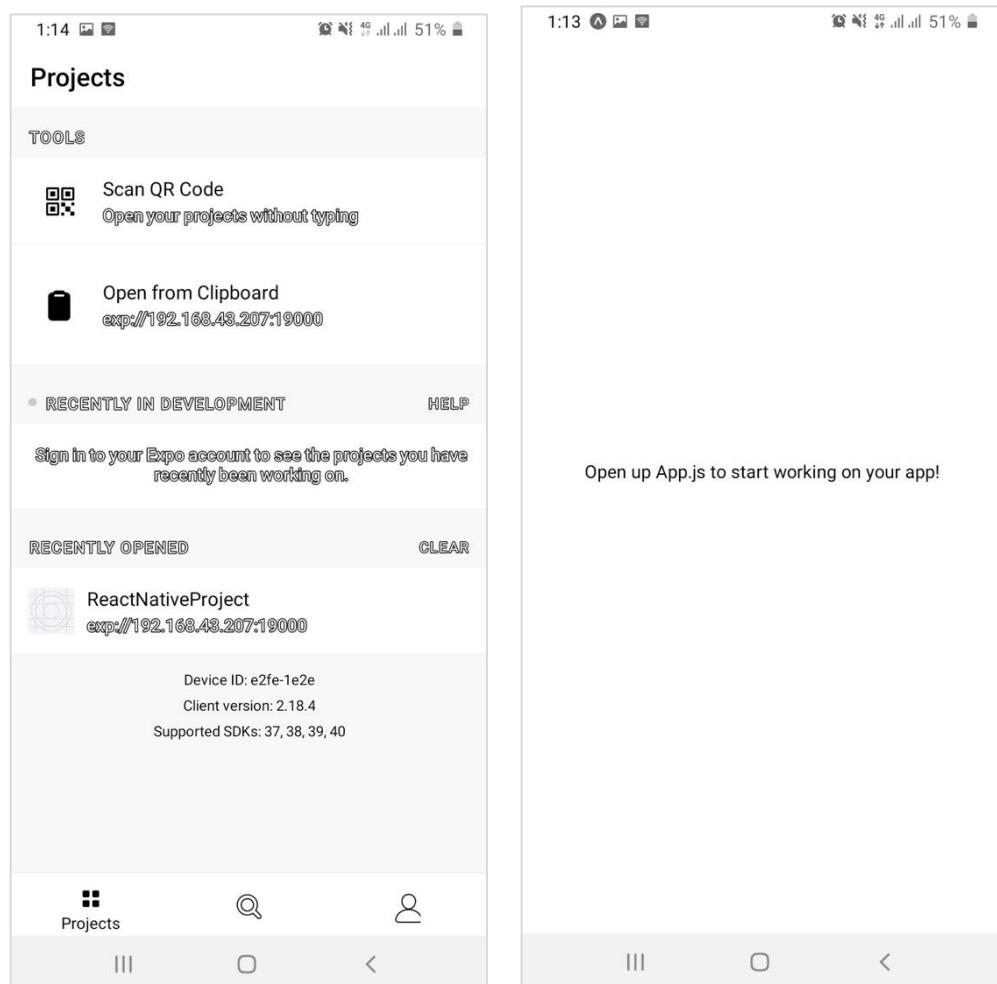
Tekan tombol ‘Run in web browser’ pada aplikasi *Expo* untuk menjalankan aplikasi *mobile React Native* pada *Web Browser*.



Gambar 8.6 Menjalankan aplikasi mobile React Native pada Web Browser

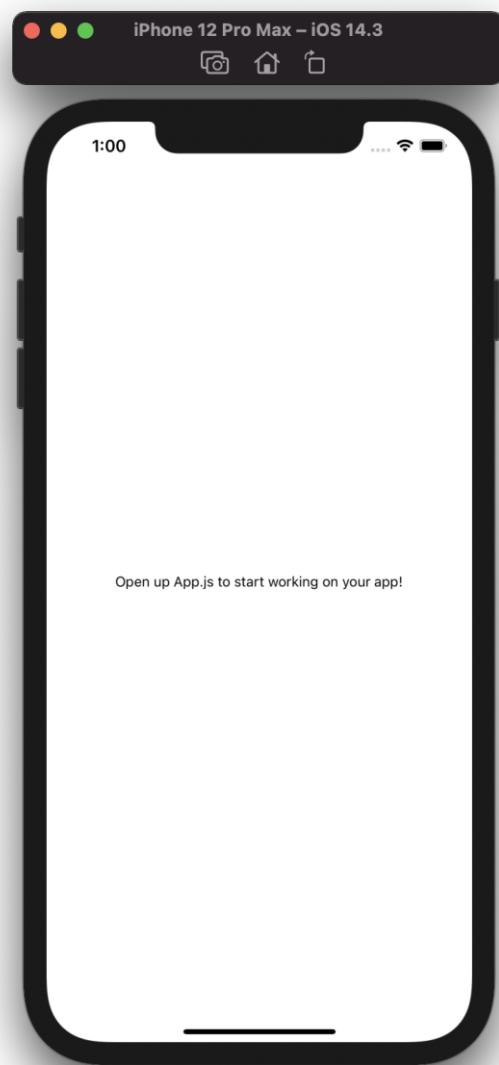
Untuk menjalankannya pada perangkat fisik *mobile*, gunakan *QR Code* yang tertera pada aplikasi *Expo*. Bila menggunakan perangkat *Android*,

*install* terlebih dahulu aplikasi *Expo* dari *Expo Project* melalui *Google Play*, lalu scan *QR Code* pada layar. Bila menggunakan perangkat *iOS* dapat langsung *scan QR Code* dengan *QR scanner* pada aplikasi *Camera iOS*. Pastikan perangkat fisik *mobile* berada pada jaringan nirkabel yang sama dengan perangkat pengembangan.



Gambar 8.7 Hasil menjalankan aplikasi mobile React Native dengan perangkat fisik Android

Bila ingin menjalankan aplikasi *mobile React Native* dengan *emulator* perangkat *mobile*, ikuti dokumentasi *official Expo* berikut (*iOS Simulator*) <https://docs.expo.io/workflow/ios-simulator/> (*Android Virtual Device*) <https://docs.expo.io/workflow/android-studio-emulator/>.



Gambar 8.8 Hasil menjalankan aplikasi mobile React Native dengan iOS Simulator



## **BUKU INFORMASI**

# **MELAKUKAN PENGATURAN SOFTWARE TOOLS PEMROGRAMAN J.620100.012.01**



KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

## DAFTAR ISI

DAFTAR ISI -----	2
BAB I PENDAHULUAN -----	4
A. Tujuan Umum -----	4
B. Tujuan Khusus -----	4
BAB II MELAKUKAN KONFIGURASI TOOLS UNTUK PEMROGRAMAN -----	5
A. Pengetahuan yang diperlukan dalam Melakukan Konfigurasi Tools Untuk Pemrograman -----	5
1. Cara menentukan target hasil dari konfigurasi -----	5
2. Cara mengkonfigurasikan tools pemrograman agar dapat digunakan sebagaimana mestinya -----	13
B. Keterampilan yang Diperlukan dalam Melakukan Konfigurasi Tools Untuk Pemrograman -----	18
C. Sikap Kerja dalam yang Diperlukan dalam Melakukan Konfigurasi Tools Untuk Pemrograman -----	18
BAB III MENGGUNAKAN TOOLS SESUAI KEBUTUHAN PEMBUATAN PROGRAM -----	19
A. Pengetahuan yang Diperlukan dalam Menggunakan Tools Sesuai Kebutuhan Pembuatan Program -----	19
1. Cara mengidentifikasi fitur-fitur dasar yang dibutuhkan untuk mendukung pembuatan program -----	19
2. Cara menguasai fitur-fitur dasar tools untuk pembuatan program dikuasai-----	21
B. Keterampilan yang Diperlukan dalam Menggunakan Tools Sesuai Kebutuhan Pembuatan Program-----	26
C. Sikap Kerja yang Diperlukan dalam Menggunakan Tools Sesuai Kebutuhan Pembuatan Program -----	26
DAFTAR PUSTAKA -----	38
A. Dasar Perundang-undangan -----	39
B. Buku Referensi -----	39
C. Majalah atau Buletin-----	40

Modul Pelatihan Berbasis Kompetensi Melakukan Pengaturan <i>Software Tools</i> Pemrograman	Kode Modul J.620100.012.01
D. Referensi Lainnya ----- 40	
<b>DAFTAR PERALATAN/MESIN DAN BAHAN ----- 40</b>	
A. Daftar Peralatan/Mesin----- 41	
B. Daftar Bahan----- 41	
<b>DAFTAR PENYUSUN..... 41</b>	

## BAB I

### PENDAHULUAN

#### A. Tujuan Umum

Setelah mempelajari modul ini peserta latih diharapkan mampu melakukan pengaturan software *tools* pemrograman dengan benar.

#### B. Tujuan Khusus

Adapun tujuan mempelajari unit kompetensi melalui buku informasi melakukan pengaturan software *tools* pemrograman ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut: (*diambil dari elemen dan KUK*)

1. Melakukan konfigurasi tools untuk pemrograman, termasuk menentukan target hasil dari konfigurasi, dan memastikan tools pemrograman setelah dikonfigurasikan, tetap bisa digunakan sebagaimana mestinya.
2. Menggunakan tools sesuai kebutuhan pembuatan program termasuk mengidentifikasi fitur-fitur dasar yang dibutuhkan untuk mendukung pembuatan program, dan menguasai fitur-fitur dasar tools untuk pembuatan program.

## **BAB II**

### **MELAKUKAN KONFIGURASI TOOLS UNTUK PEMROGRAMAN**

#### **A. Pengetahuan yang diperlukan dalam Melakukan konfigurasi tools untuk pemrograman**

##### **1. Cara Mengetahui menentukan target hasil dari konfigurasi**

Alat (Tools) pemrograman atau alat pengembangan perangkat lunak adalah program komputer yang digunakan pengembang perangkat lunak untuk membuat, men-debug, memelihara, atau mendukung program dan aplikasi lain. Istilah ini biasanya mengacu pada program yang relatif sederhana, yang dapat dikombinasikan bersama untuk menyelesaikan tugas, sebanyak mungkin menggunakan beberapa alat untuk memperbaiki objek fisik. Alat yang paling dasar adalah editor kode sumber dan kompilator atau interpreter, yang digunakan di mana-mana dan terus menerus. Tool lain digunakan tergantung pada bahasa, metodologi pengembangan, dan individu, dan sering digunakan untuk tugas diskrit, seperti debugger atau profiler. Tools dapat berupa program diskrit, dieksekusi secara terpisah - sering dari baris perintah - atau mungkin bagian dari satu program besar, yang disebut lingkungan pengembangan terintegrasi (IDE).

##### **a. Pengertian Konfigurasi Perangkat Lunak**

Konfigurasi perangkat lunak adalah proses pemasangan, pelacakan juga pengendali perubahan dalam perangkat lunak, dimana konfigurasi ini merupakan bagian dari bidang lintas disiplin yang lebih besar dari manajemen konfiguras

Dalam proses konfigurasi, administrator dapat memilih konfigurasi dalam bentuk format standar (*template*) atau memilih konfigurasi sesuai keinginan (*customize*).

1. Dalam komputer dan jaringan komputer, konfigurasi sering mengacu pada hardware spesifik dan rincian perangkat lunak dalam hal perangkat yang terpasang, kapasitas atau kemampuan, pada sistem yang dibuat.
2. Dalam jaringan, konfigurasi sering berarti topologi jaringan.
3. Dalam menginstal hardware dan software, konfigurasi kadang-

kadang proses metodis mendefinisikan pilihan yang disediakan.

File konfigurasi merupakan sebutan lain untuk file pengaturan. Ekstensi file konfigurasi yang populer adalah .VMX, .OPS, .INS dan .CSF.

### b. Ekstensi File Konfigurasi terpopuler

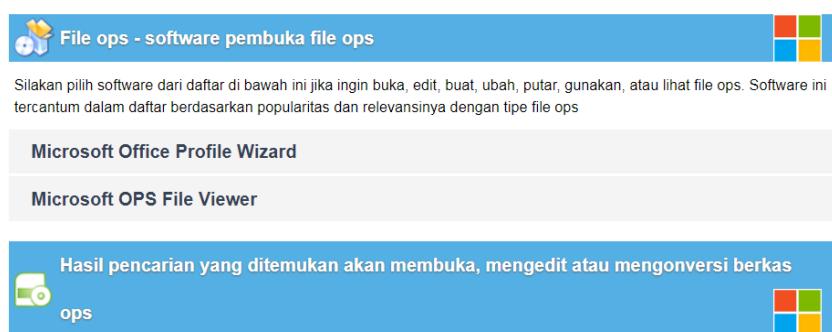
Ada beberapa ekstensi file terpopuler dan sering digunakan. Ekstensi tersebut adalah:

#### 1) .VMX

.VMX berada dibawah kategori Berkas Pelaporan Pembuatan Model Data ERP. Ini paling banyak diasosiasikan dengan visual Infor ERP, dibuat oleh Infor, sebuah penyedia perangkat lunak solusi juga pengiriman dengan performa paling tinggi yang memenuhi kebutuhan model-model dalam industri pabrik seperti dibuat atas permintaan, pembuatan rancangan atas pesanan, dibuat untuk persediaan, dikonfigurasi atas pesanan, dirakit atas pesanan.

#### 2) .OPS

.ops adalah jenis pengaturan berkas yang dibuat oleh Profile Wizard dari Microsoft Office, program yang digunakan untuk mengkonfigurasi informasi pengguna untuk program Microsoft Office. Pengguna dapat melihat berkas OPS dengan menggunakan program penampil berkas OPS yang didasarkan pada Kotak Sumber daya Office.



Gambar 1. File pembuka .ops

#### 3) .INS

Mereka mengandung pengaturan untuk koneksi internet lewat panggilan telepon atau dial up dari Windows dan pengaturan pita lebar. Koneksi

dan pengaturan masuk juga disimpan dalam berkas ini.

Berkas ini biasanya dirujuk oleh IE untuk membentuk LAN

#### **4) .CSF**

Memuat kebijakan dan definisi warna. Membuat perubahan dengan Adobe Photoshop. Berbagi berkas tidak dapat dilakukan dalam Photoshop. Keistimewaan seperti penyelarasan warna, warna pracetak berbeda untuk wilayah-wilayah.

Mewujudkan gambar konsistensi dan warna yang bagus.

## **2. Cara mengkonfigurasikan *tools* pemrograman agar dapat digunakan sebagaimana mestinya**

### a. Pastikan tools pemrograman sesuai kebutuhan

Tools pemrograman seperti Integrated Development Environment (IDE), Environment development, testing tools, dan project management telah terpasang.

### b. Pastikan tools pemrograman dapat digunakan

Tools pemrograman yang akan dijalankan dapat running di komputer dan tidak terjadi kesalahan (error) atau pun tidak terbaca.

## **B. Keterampilan yang diperlukan dalam Melakukan Konfigurasi Tools Untuk Pemrograman**

1. Menentukan target hasil konfigurasi
2. Mengkonfigurasikan *tools* pemrograman agar dapat digunakan sebagaimana mestinya

## **C. Sikap Kerja yang diperlukan dalam Melakukan Konfigurasi Tools Untuk Pemrograman**

1. Harus teliti dalam menentukan target hasil konfigurasi
2. Harus cekatan dalam mengkonfigurasikan *tools* pemrograman agar dapat digunakan sebagaimana mestinya dalam mengkonfigurasi *software tools*

## **BAB III**

### **MENGGUNAKAN TOOLS SESUAI KEBUTUHAN PEMBUATAN PROGRAM**

#### **A. Pengetahuan yang diperlukan dalam Menggunakan Tools Sesuai Kebutuhan Pembuatan Program**

##### **1. Cara mengidentifikasi fitur-fitur dasar yang dibutuhkan untuk mendukung pembuatan program**

Identifikasi fitur-fitur dasar yang dibutuhkan dalam pembuatan program adalah:

- Memastikan fitur IDE telah terpasang

Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak. Dengan adanya aplikasi ini, pemrogram akan lebih mudah menulis barisan kode pemrograman.

- Memastikan aplikasi testing tools telah terpasang

Aplikasi ini diperlukan untuk mengetes hasil dari barisan kode program yang telah dituliskan.

- Memastikan aplikasi project management telah terpasang

- Memastikan coding standar telah terpasang

- Memastikan aplikasi debugger dan logger telah terpasang

- Memastikan aplikasi deployment tools

##### **2. Cara menguasai fitur-fitur dasar tools untuk pembuatan program**

- Memahami penggunaan dari IDE

- Memahami penggunaan dari aplikasi testing tools

- Memahami aplikasi project management

- Memahami penggunaan coding standar, dan melakukan coding

- Memahami penggunaan aplikasi debugger dan logger

- Memahami penggunaan aplikasi deployment tools

#### **B. Keterampilan yang diperlukan dalam Menggunakan Tools Sesuai Kebutuhan Pembuatan Program**

- Mengidentifikasi fitur-fitur dasar yang dibutuhkan untuk mendukung pembuatan program

- Menguasai Fitur-fitur dasar tools untuk pembuatan program

### **C. Sikap Kerja yang diperlukan dalam Menggunakan Tools Sesuai Kebutuhan Pembuatan Program**

1. Harus cermat mengidentifikasi fitur-fitur dasar yang dibutuhkan untuk mendukung pembuatan program
2. Harus cermat menguasai Fitur-fitur dasar tools untuk pembuatan program dalam menggunakan tools pemrograman



**BUKU INFORMASI**

**MENYUSUN FUNGSI, FILE ATAU  
SUMBER DAYA PEMROGRAMAN YANG LAIN  
DALAM ORGANISASI YANG RAPI**

**J.620100.015.01**



KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

## DAFTAR ISI

DAFTAR ISI -----	2
BAB I PENDAHULUAN -----	4
A. Tujuan Umum -----	4
B. Tujuan Khusus -----	4
BAB II MENGELOLA SUMBER DAYA PEMROGRAMAN SESUAI KARAKTER -----	5
A. Pengetahuan yang Diperlukan dalam Mengelola Sumber Daya Pemrograman Sesuai Karakter -----	5
1. Cara Membuat Nama <i>File</i> , Fungsi, Variabel, Konstanta, dan Sumber Daya Pemrograman Lain Sesuai Konteks-----	5
2. Cara Melengkapi dengan Penulisan Komentar di Awal Setiap Fungsi/ Prosedur/ Program Mengenai Deskripsi Fungsi/ Prosedur/ Program Tersebut; <i>Initial State</i> dan <i>Final State</i> ; <i>Author</i> (Pembuat); Versi dan/atau Tanggal -----	8
3. Cara Melengkapi Badan <i>Source Code</i> dengan Komentar/ Keterangan yang Cukup, yang Memberikan Penjelasan atas Baris-baris Instruksi -----	12
B. Keterampilan yang Diperlukan dalam Mengelola Sumber Daya Pemrograman Sesuai Karakter -----	14
C. Sikap Kerja dalam Mengelola Sumber Daya Pemrograman Sesuai Karakter -----	15
BAB III MENGORGANISASIKAN SUMBER DAYA PEMROGRAMAN SESUAI KONTEKS	16
A. Pengetahuan yang Diperlukan dalam Mengorganisasikan Sumber Daya Pemrograman Sesuai Konteks -----	16
1. Cara Penyusunan <i>Folder</i> dan Sub-sub <i>Folder</i> Sesuai Konteks dan Isinya -----	16
2. Cara Membuat <i>File</i> “readme” yang Mengandung Penjelasan Mengenai Struktur/ Hirarki <i>Folder</i> Serta Penjelasan Mengenai Sumber Daya Pemrograman -----	18

Modul Pelatihan Berbasis Kompetensi Bidang <i>Software Development</i> Subbidang Pemrograman	Kode Modul J.620100.015.01
B. Keterampilan yang Diperlukan dalam Mengorganisasikan Sumber Daya Pemrograman Sesuai Konteks -----	20
C. Sikap Kerja yang Diperlukan dalam Mengorganisasikan Sumber Daya Pemrograman Sesuai Konteks -----	21
DAFTAR PUSTAKA -----	22
A. Dasar Perundang-undangan -----	22
B. Buku Referensi -----	22
C. Referensi Lainnya -----	22
DAFTAR PERALATAN/MESIN DAN BAHAN -----	24
A. Daftar Peralatan/Mesin-----	24
B. Daftar Bahan-----	24
DAFTAR PENYUSUN -----	25
Judul Modul: Menyusun Fungsi, File atau Sumber Daya Pemrograman yang Lain dalam Organisasi yang Rapi Buku Informasi	Halaman: 3 dari 25
Versi: 2018	

## **BAB I**

### **PENDAHULUAN**

#### **A. Tujuan Umum**

Setelah mempelajari modul ini peserta latih diharapkan mampu menyusun fungsi, *file* atau sumber daya pemrograman yang lain dalam organisasi yang rapi.

#### **B. Tujuan Khusus**

Adapun tujuan mempelajari unit kompetensi melalui buku informasi Menyusun Fungsi, *File* atau Sumber Daya Pemrograman yang Lain Dalam Organisasi yang Rapi ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut:

1. Mengelola sumber daya pemrograman sesuai karakter yang meliputi kegiatan membuat nama *file*, fungsi, variabel, konstanta, dan sumber daya pemrograman lain sesuai konteks, melengkapi dengan penulisan komentar di awal setiap fungsi/prosedur/program mengenai deskripsi fungsi/ prosedur/program tersebut; *initial state* dan *final state*; *author* (pembuat); versi dan/atau tanggal, serta melengkapi badan *source code* dengan komentar/keterangan yang cukup, yang memberikan penjelasan atas baris-baris instruksi;
2. Mengorganisasikan sumber daya pemrograman sesuai konteks yang meliputi kegiatan menyusun *folder* dan sub-sub *folder* sesuai konteks dan isinya dan membuat *file* “*readme*” yang mengandung penjelasan mengenai struktur/ hirarki *folder* serta penjelasan mengenai sumber daya pemrograman;

## **BAB II**

### **MENGELOLA SUMBER DAYA PEMROGRAMAN SESUAI KARAKTER**

#### **A. Pengetahuan yang Diperlukan dalam Mengelola Sumber Daya Pemrograman Sesuai Karakter**

1. Cara Membuat Nama *File*, Fungsi, Variabel, Konstanta, dan Sumber Daya Pemrograman Lain Sesuai Konteks

Dokumentasi yang baik sangat diperlukan terutama dalam perbaikan *bug* sistem dan pengembangan sistem, karena perbaikan *bug* dan pengembangan sistem kemungkinan dilakukan oleh orang yang berbeda dari pembuat sistem di awal. Sehingga akan sangat efisien dan membantu bila dokumentasi program dibuat dengan baik.

Dokumentasi yang dimaksud pada modul ini mencakup penulisan nama *file*, fungsi, variabel, konstanta, dan sumber daya pemrograman lainnya, komentar atau keterangan yang cukup pada setiap fungsi/ prosedur/ program, *initial* dan *final state*, termasuk penulisan nama pembuat dan versi (atau tanggal pembuatan).

Abelson dan Sussman dalam buku *Structure and Interpretation of Computer Programs* menulis "*Programs should be written for people to read, and only incidentally for machines to execute*", yang bila diterjemahkan secara sederhana berbunyi "program harus ditulis untuk dibaca manusia/ orang dan secara kebetulan (sekaligus) untuk dieksekusi oleh mesin".

##### **1.1 Cara Membuat Nama *File***

Dokumentasi berkaitan sangat erat dengan aturan penamaan yang memungkinkan pemberian nama dengan logis dan konsisten *file* secara elektronik (atau secara fisik). Hal ini memastikan file dapat ditemukan lokasi penyimpanannya yang aman serta dapat dipanggil secepat mungkin (timely

fashion). Idealnya waktu yang paling tepat untuk memikirkan bagaimana *file* dinamakan dan bagaimana struktur penyimpanannya adalah di awal pembuatan proyek.

Penamaan *file* secara konsisten, logis, dan di lokasi yang sesuai akan membedakan *file-file* yang mirip dalam sekejap. Dan dengan demikian akan memudahkan penyimpanan dan pemanggilan kembali *file*. Dengan konsistensi dan pengaplikasian standar logis kita akan diuntungkan dari segi penyimpanan yang aman dan kemampuan untuk mengetahui lokasi dan mengaksesnya.

Penamaan *file* dapat terdiri dari nama *file* itu sendiri dan ditambahkan dengan identifikasi nama atau jabatan pembuat, tanggal pembuatan, dan versi.

Berikut aturan penamaan secara umum yang disarankan diterapkan dalam membuat nama *file*:

- Hindari penggunaan spasi

Pada *browser*, *web server* dan bahasa pemrograman tidak menangani spasi secara konsisten, beberapa sistem akan memperlakukan spasi sebagai dua nama *file*, beberapa *server* akan mengganti spasi pada nama *file* Anda dengan "%20". Sehingga sangat disarankan untuk tidak menggunakan spasi untuk penamaan file. Sebagai pengganti spasi, Anda dapat menggunakan tanda hubung (-) atau *underscore*.

- Gunakan huruf kecil

Penamaan *file* disarankan untuk menggunakan huruf kecil dikarenakan pada beberapa kasus terutama *web server* bersifat *case sensitive* (huruf kapital/besar dan huruf kecil dianggap berbeda).

- Bila menggunakan angka dalam penamaan disarankan untuk menggunakan dua digit angka atau lebih sesuai kebutuhan.

- Bila menggunakan tanggal, penulisan dimulai dari tahun, bulan kemudian tanggal.
- Bila mencantumkan nama pembuat, untuk mempersingkat dapat menggunakan inisial nama.
- Hindari penggunaan kata-kata umum seperti “draf” atau “program” sebagai awalan nama.
- Urutkan elemen penamaan dengan urutan yang paling pantas untuk memudahkan pencarian.
- Hindari penggunaan karakter-karakter khusus dalam penamaan *file*.
- Nama *file* disarankan agar tidak terlalu panjang, untuk menghindari masalah pada saat *back-up*, *restore*, atau penduplikasian (*copy*).
- Penamaan idelanya akan membuat kumpulan file dapat dengan mudah dicari dan ditampilkan secara kronologi (*chronologically*).

## 1.2 Cara Membuat Nama Fungsi, Variable, Konstanta, dan Sumber Daya Pemrograman Lainnya

Sama halnya dengan penulisan nama *file*, penulisan nama fungsi, variabel, konstanta, dan sumber daya pemrograman lainnya juga menggunakan prinsip dasar yang hampir sama seperti pembahasan di atas.

Pemilihan nama yang baik untuk bagian dari program (dalam bahasan ini berupa fungsi atau variabel atau konstanta dan sumber daya pemrograman lainnya) adalah hal yang sangat penting.

Sesuai dengan pembahasan kita sebelumnya bahwa menulis sebuah program (*coding*) harus dapat dibaca oleh manusia. Penamaan fungsi, variabel, konstanta, dan sumber daya pemrograman lainnya adalah cara yang sangat tepat untuk memberi tahu pembaca apa kegunaannya. Dan penamaan tersebut dapat mempengaruhi kualitas dari *coding*. Walaupun pada akhirnya akan terdapat dokumentasi program, namun tetap penamaan

pada fungsi, variable, konstanta, dan sumber daya pemrogram lainnya adalah cara yang sangat efektif untuk menyampaikan informasi tentang *code*, salah satunya karena nama yang sangat baik secara instan (lebih cepat) memberitahukan apa yang terjadi pada *code* dibandingkan harus melihat/ merujuk pada dokumentasi program dan menelusuri *code* tersebut.

Berikut panduan umum penamaan yang baik:

- Hindari penggunaan karakter atau kata yang ilegal yang memang dikhkususkan oleh *compiler*
- Selektif dalam penggunaan singkatan untuk penamaan, gunakan hanya singkatan umum yang Anda yakin semua orang mengetahui arti dari singkatan tersebut, bila tidak sebaiknya jangan gunakan singkatan, walaupun terlihat panjang dan tidak efektif namun dibandingkan dengan harus mempelajari/ menelusuri *code* dikarenakan pembaca tidak mengerti arti singkatan tersebut
- Pilihlah nama yang konsisten dengan tingkat abstraksi dari fungsi, variable, konstanta, dan sumber daya pemrogram lainnya
- Berilah nama dengan mempertimbangkan kegunaan dari fungsi, variable, konstanta, dan sumber daya pemrogram lainnya
- Hindari penggunaan kata-kata yang memiliki arti luas

2. Cara Melengkapi dengan Penulisan Komentar di Awal Setiap Fungsi/ Prosedur/ Program Mengenai Deskripsi Fungsi/ Prosedur/ Program Tersebut; *Initial State* dan *Final State*; *Author* (Pembuat); Versi dan/atau Tanggal

Pemberian komentar adalah salah satu tambahan *tool* (alat) yang dapat dipilih oleh seorang pemrogram untuk digunakan atau tidak (bila penamaan dilakukan dengan benar maka penulisan komentar bisa tidak perlu dilakukan lagi) untuk menjelaskan kepada pembaca (manusia) tentang fungsi/ prosedur/ program.

Komentar adalah bagian dari *source code*, walaupun komentar secara teknikal tidak penting untuk mesin dan tidak dieksekusi, namun karena berguna untuk manusia (pembaca), maka harus diperlakukan sebagai bagian dari *source code* tersebut, dimana berarti bahwa bila ada perubahan pada *source code* maka komentarnya juga harus disesuaikan, dan bila *code* tersebut dihilangkan maka komentarnya juga harus dihapus.

Berikut beberapa situasi dimana penggunaan komentar lebih efektif dibandingkan dengan *tools* lainnya:

- Pembaca memerlukan konteks, yang tidak dapat diekspresikan dengan *tools* lain
- Algoritma dan struktur data yang rumit/ kompleks
- *Tools* lain tidak dapat diterapkan, baik karena keterbatasan *size* atau waktu, dalam kasus seperti ini komentar bersifat sementara (tidak permanen)

Dalam fungsi/ prosedur/ program terdapat tiga jenis komentar, yaitu:

- a. Komentar/ deskripsi singkat (*brief description*) adalah komentar pendek yang tidak lebih dari satu baris atau biasa disebut komentar *inline*. Pada bahasa pemrograman C/C++ misalnya, deskripsi singkat diawali dengan dua garis miring atau *slash* (//), contoh:

```
//isi deskripsi singkat
```

- b. Komentar/ deskripsi detail (*detail description*) adalah komentar yang lebih panjang dan lebih detail (lebih dari satu baris). Pada bahasa pemrograman C/C++ misalnya, deskripsi detail diawali dengan garis miring dan tanda bintang /\*) dan ditutup dengan tanda bintang dan garis miring (\*), contoh:

```
/* isi deskripsi detail  
yang panjangnya lebih dari 1 baris */
```

- c. *In-body description/ comment* yang berisi rangkaian semua blok komentar yang ditemukan di dalam badan fungsi/ prosedur/ program, *in-body*

*description* bias juga berperan sebagai detail deskripsi atau dapat menjelaskan kumpulan detail implementasi

Setiap bahasa pemrograman memiliki syntax sendiri untuk menandakan komentar, seperti pada pada html, pembuatan komentar dilakukan dengan cara diawali dengan tanda <!-- dan ditutup dengan tanda -->

Contoh:

```
<!-- contoh komentar pada html -->
```

Pada bahasa Phyton komentar diawali dan ditutup menggunakan tiga tanda kutip (""""")

```
 """ contoh komentar pada bahasa  
 pemrograman Phyton """
```

Berikut beberapa resiko penggunaan komentar:

- Komentar perlu juga perlu mendapat perhatian (*need maintenance*), ketika *code* ada perubahan Anda juga harus mengubah komentar
- Tidak mudah untuk membuat kometar yang baik, dimana komentar yang baik adalah yang presisi dan relevan, sehingga belum tentu pemrogram dapat menjelaskan apa yang diprogram dengan jelas dan benar
- Komentar tidak diperiksa oleh *compiler* sehingga tidak ada jaminan tidak ada kesalahan dalam komentar terutama bila *code* sudah mengalami modifikasi
- Kesalahan penulisan komentar (*typography*) dapat mengacaukan *code*

Secara umum isi dari komentar disarankan sebagai berikut:

1. Pemberian komentar dilakukan untuk memperjelas kegunaannya serta sebagai catatan tambahan dan umumnya dapat berisi deskripsi tentang fungsi/ prosedur/ program, kondisi sebelum eksekusi program (*initial state*) dan kondisi setelah eksekusi (*final state*), nama atau inisial pembuat, versi dan/atau tanggal pembuatan

2. Informasi yang sudah tertulis dalam *file* dokumentasi lainnya sebaiknya tidak lagi ditulis dalam komentar karena hal tersebut menjadi duplikasi, disarankan isi komentar adalah lebih mengklarifikasi tentang fungsi/prosedur/ program terkait dibandingkan sebagai dokumentasi program
3. Gunakan kata-kata yang spesifik dan tidak menimbulkan salah paham karena dapat diasumsikan menjadi beberapa arti
4. Komentar bukanlah sebuah karya seni sehingga hindari penggunaan tanda atau kata-kata yang tidak perlu dan terlalu panjang

```
/*
(c) .-. (c)      (c) .-. (c)
/ .-. \      / .-. \      / .-. \      / .-. \      / .-. \      / .-. \
\ ( Y ) /_ \ ( Y ) /_
(_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_)
|| M || || O || || N || || R || || E || || Y ||
(_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_) (_.-/-'-'(.-_
`-`-`-`-`-`-`-
```

\*/

Gambar 2.1 – Contoh pembuatan komentar yang tidak perlu

5. Hindari penggunaan kata atau karakter yang sudah dikhkusukan oleh *complier*, seperti pada bahasa pemrograman C/C++ berikut beberapa contoh kata atau karakter yang sudah dikhkusukan oleh *complier* dan bila kata-kata atau karakter tersebut ingin digunakan dalam komentar maka ada penambahan karakter yang harus ditulis di awal kata, misalnya pada bahasa pemrograman C/C++ adalah tanda garis miring terbalik atau *backslash* (\):

- \struct
- \union
- \enum
- \fn
- \var
- \def
- \typedef

- \file
  - \namespace
  - \package
  - \interface
  - \a
6. Seperti telah dibahas sebelumnya bahwa komentar adalah bagian dari *code* maka bila dilakukan modifikasi pada *code* maka komentarnya juga harus diperbarui
7. Tidak semua baris *code* perlu diberi komentar
3. Cara Melengkapi Badan *Source Code* dengan Komentar/ Keterangan yang Cukup, yang Memberikan Penjelasan atas Baris-baris Instruksi  
Penulisan komentar selain dapat dilakukan di awal fungsi/ prosedur/ program, dapat juga dilakukan pada baris-baris instruksi (*code*), namun seperti telah dibahas sebelumnya bahwa tidak perlu setiap baris *code* diberi komentar, berikut contoh ilustrasi penggunaan komentar yang tidak perlu.

```
i = i + 1;      //Tambah 1 ke i
```

Gambar 2.2 – Contoh pembuatan komentar yang tidak perlu

```
/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X                               X
X           Tambah 1 ke i           X
X                               X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX/
```

Gambar 2.3 – Contoh lain pembuatan komentar yang tidak perlu

Saran penulisan komentar dan resiko penulisan komentar pada baris-baris *code* pada prinsipnya sama dengan penulisan pada awal setiap fungsi/ prosedur/ program, perbedaannya hanya terletak pada pada baris-baris code komentar

umunya lebih singkat dan menjelaskan hanya kegunaannya saja dan itu hanya dilakukan bila dirasa perlu saja sebagai informasi tambahan dan tidak berisi detail pembuat, versi dan lain sebagainya yang biasa tertulis pada komentar di awal fungsi/ prosedur/ program .

```
#include<iostream.h>
#include<conio.h>

1 { /* Contoh Komentar dalam Bahasa C/C++
     @prosedur untuk menampilkan kalimat
     @initial state: layar kosong
     @final state: layar menampilkan tulisan "Contoh Komentar dalam C/C++"
     @versi 1.0
     @rini <rini@contoh.com>
 */

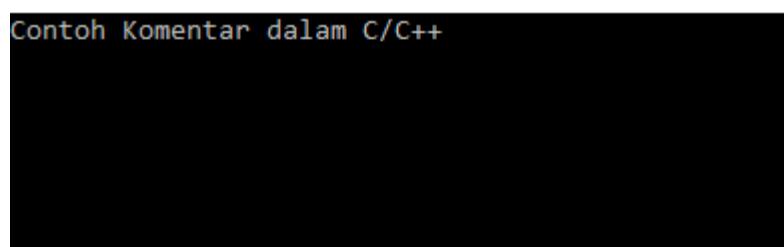
void display(){

2 → //Kalimat di bawah ini akan ditampilkan
    cout<<"Contoh Komentar dalam C/C++"<<endl;

3 {
    /*Tulisan di bawah ini tidak akan ditampilkan
     karena termasuk di dalam blok komentar
    cout<<"TIDAK TAMPIL"<<endl;
}

4 void main() {
    display();    //menjalankan prosedur display()
    getch();      /*perintah agar program berakhir setelah pengguna menekan
                   key apapun*/
}
```

Gambar 2.4 – Contoh penggunaan komentar



Gambar 2.5 –Output program dari gambar 2.4

Dari contoh program pada Gambar 2.4 berikut penjelasan tiap bagianya yang diberi angka dengan warna biru:

1. Pada bagian nomor 1 adalah contoh penggunaan komentar di awal setiap fungsi/ prosedur/ program mengenai deskripsi fungsi/ prosedur/ program tersebut; *initial state* dan *final state*; *author* (pembuat); versi dan/atau tanggal yang telah kita bahasa sebelumnya pada poin (subbab) sebelumnya.
2. Bagian nomor 2 adalah contoh penggunaan komentar pada badan fungsi yang menginformasikan kegunaan dari baris kode tersebut, sekaligus contoh penggunaan komentar *inline*.
3. Bagian nomor 3 adalah contoh penggunaan *detail description* dan penggunaan komentar untuk memberi tahu *compiler* untuk tidak mengeksekusi kode yang ada di dalamnya.
4. Bagian nomor 4 adalah contoh penggunaan penggunaan komentar pada badan fungsi yang menginformasikan kegunaan dari baris kode tersebut, sekaligus contoh penggunaan komentar *inline* (seperti pada bagian nomor 2) dan *detail description*.

Dari contoh tersebut dapat dilihat dengan mudah oleh bahkan orang awam sekalipun tentang kegunaan prosedur hingga baris kode.

Selain hal-hal yang telah dibahas di atas, untuk mempermudah perbaikan *bug* dan pengembangan program, disarankan untuk memperhatikan *indentation* dalam penulisan baris-baris program, sehingga akan memudahkan pembaca untuk memahami struktur program.

## **B. Keterampilan yang Diperlukan dalam Mengelola Sumber Daya Pemrograman Sesuai Karakter**

1. Membuat nama *file*, fungsi, variabel, konstanta, dan sumber daya pemrograman lain sesuai konteks.
2. Melengkapi dengan penulisan komentar di awal setiap fungsi/prosedur/program mengenai deskripsi fungsi/ prosedur/program tersebut; *initial state* dan *final state*; *author* (pembuat); versi dan/atau tanggal.

3. Melengkapi badan *source code* dengan komentar/keterangan yang cukup, yang memberikan penjelasan atas baris-baris instruksi

### C. Sikap kerja

Harus bersikap secara:

1. Cekatan, cermat, disiplin, dan bertanggung jawab dalam membuat nama *file*, fungsi, variabel, konstanta, dan sumber daya pemrograman lain sesuai konteks.
2. Cekatan, cermat, disiplin, dan bertanggung jawab dalam melengkapi dengan penulisan komentar di awal setiap fungsi/prosedur/program mengenai deskripsi fungsi/ prosedur/program tersebut; *initial state* dan *final state*; *author* (pembuat); versi dan/atau tanggal.
3. Cekatan, cermat, disiplin, dan bertanggung jawab dalam melengkapi badan *source code* dengan komentar/ keterangan yang cukup, yang memberikan penjelasan atas baris-baris instruksi.

## **BAB III**

### **MENGORGANISASIKAN SUMBER DAYA PEMROGRAMAN**

### **SESUAI KONTEKS**

#### **A. Pengetahuan yang Diperlukan dalam Mengorganisasikan Sumber Daya Pemrograman Sesuai Konteks**

##### 1. Cara Penyusunan *Folder* dan Sub-sub *Folder* Sesuai Konteks dan Isinya

Seperti telah disinggung sebelumnya bahwa dokumentasi yang baik dalam sebuah program adalah sangat membantu ketika perbaikan *bug* dan pengembangan program. Dimulai dari cara penamaan *file*, fungsi, variabel, konstanta, dan sumber daya pemrograman lain sesuai konteks, pemberian komentar di awal setiap fungsi/ prosedur/ program dan di badan *source code* hingga penulisan dengan *indentation* yang benar. Pada bab ini akan dibahas cara menyusun folder dan sub-sub folder sesuai konteks dan isinya.

Sebuah sistem terdiri dari banyak *file*, yang dapat berupa konten teks, kode program, konten media, dan lain-lain. Ketika Anda sedang membuat sebuah program, Anda perlu merangkai *file-file* ini menjadi sebuah struktur yang sesuai pada komputer Anda, memastikan *file-file* tersebut dapat saling terhubung antara satu dengan yang lain, dan memastikan bahwa semua konten sudah benar sebelum akhirnya Anda mengunggah semua *file* ke sebuah *server* (untuk kasus pembuatan situs).

Program yang Anda buat dapat terdiri dari banyak *file*. Ketika Anda sedang mengerjakan sebuah program secara lokal pada komputer pribadi Anda, Anda harus menyimpan semua *file* yang terkait pada satu *folder* yang sesuai dengan struktur program. *Folder* ini dapat Anda simpan dimanapun sesuka Anda, tetapi Anda harus meletakkannya di suatu tempat yang mudah untuk ditemukan, seperti pada Desktop, pada *folder* Home, atau pada *folder* dasar di *hard drive* Anda.

Namun setelahnya, bila program dibuat lebih dari satu orang, maka akan diperlukan integrasi *file* dan kode-kode program. Untuk itu perlu dipikirkan di awal pembuatan perihal *folder* hingga sub-sub *folder* untuk mengorganisasikan sumber daya pemrograman terkait, seperti gambar, kode program, dan lain sebagainya.

Selain penulisan nama, pengorganisasian *file* juga sangat penting untuk memudahkan pemahaman, pencarian, dan pengorganisasian yang efisien. Berikut beberapa alasan pentingnya pengorganisasian *file*:

- Mempercepat kerja. Setiap Anda menghabiskan waktu lebih dari 30 detik untuk mencari file atau data, Anda secara tidak langsung mengganggu alur kerja Anda dan sangat mudah untuk Anda terkena stress.
- Anda akan lebih memegang kendali atas apa yang Anda lakukan. Menempatkan sesuatu, bahkan hal kecil sekalipun dengan teratur dapat menimbulkan perasaan yang baik sehingga akan menularkan ke pekerjaan Anda.
- Anda akan dapat merasakan manfaat dari pelajaran sebuah desain, dimana sebuah desain adalah tentang menempatkan sesuatu pada lokasi yang tepat dan dengan urutan yang benar.

Berikutnya, mari kita lihat contoh struktur yang disarankan untuk pembuatan sebuah situs. Hal yang paling sering Anda temukan pada proyek situs apapun adalah sebuah file indeks HTML dan *folder* untuk menyimpan gambar, *file* CSS, dan *file* skrip. Berikut folder utama yang umumnya diperlukan:

1. *index.html*

*File* ini secara umum berisi halaman beranda, yaitu gambar dan teks yang orang dapat lihat saat pertama kali membuka website Anda.

2. *Folder images*

*Folder* ini akan berisi semua gambar yang akan digunakan pada situs Anda.

3. *Folder styles*

*Folder* ini berisi kode CSS yang digunakan untuk konten Anda, termasuk pengaturan teks dan warna latar belakang).

#### 4. *Folder scripts*

*Folder* ini berisi semua kode JavaScript yang digunakan untuk menambah fungsionalitas interaktif pada situs Anda.

### 2. Cara Membuat *File* “readme” yang Mengandung Penjelasan Mengenai Struktur/Hirarki *Folder* Serta Penjelasan Mengenai Sumber Daya Pemrograman

*File* “readme” sangatlah krusial namun merupakan cara paling awal (*basic*) dalam mendokumentasikan program (proyek) Anda, dimana setiap proyek harus memiliki *File* “readme”.

*File* “readme” adalah *file* teks yang memperkenalkan dan menjelaskan program. *File* “readme” berisi informasi yang umumnya dibutuhkan untuk memahami tentang program tersebut.

*File* “readme” dibuat agar pengguna dapat mengetahui dengan cepat mengenai cara instalasi program atau bagaimana bekolaborasi dengan Anda.

Semua pemrogram yang terlibat dalam pembuatan program yang dimaksud, harus berkontribusi dalam pembuatan *file* “readme”.

*File* “readme” dibuat sebelum Anda menunjukan program atau proyek Anda kepada orang lain atau sebelum Anda mempublikasikannya.

*File* “readme” umumnya disimpan pada direktori utama program (proyek) Anda karena di direktori inilah pengguna (*user*) akan memulai. Aplikasi Versioning Control System (VCS) atau *code hosting service* seperti GitHub, Bitbucket, GitLab juga akan mencari *file* “readme” Anda dan menampilkannya bersama dengan daftar *file* dan direktori proyek Anda.

Berikut beberapa saran isi dari *file* “readme” yang baik, namun perlu diingat bahwa setiap proyek berbeda, berikut hanya garis besar secara umum:

1. *File* “readme” dapat sangat panjang dan sangat detail, terlalu panjang lebih baik dibanding telalu pendek, disarankan dibandingkan menghilangkan informasi yang mungkin dibutuhkan, pertimbangkan untuk menggunakan bentuk lain sebagai dokumentasi proyek Anda.
2. Nama dan deskripsi sedapat mungkin dibuat agar mewakili proyek Anda pada tingkatan yang paling tinggi.
3. *File* “readme” adalah media yang baik untuk menuliskan kelebihan dari program Anda dan juga di *file* ini *user* dapat mengetahui dengan spesifik tentang apa yang dapat dilakukan oleh program Anda atau fitur-fiturnya (*features*).
4. Tergantung dari program apa yang Anda buat, bila memungkinkan di *file* “readme” dapat juga ditampilkan *screenshot* atau video (dapat berupa GIFs).
5. Bila proyek Anda hanya dapat dijalankan di konteks tertentu, perihal persyaratan ini (*requirements*) ini harus dituliskan atau disampaikan agar menghindari *user* untuk mencoba menggunakan program Anda namun akhirnya gagal karena ketidak sesuaian persyaratan yang dibutuhkan, misalnya Operating System (OS) tidak sesuai.
6. Dsarankan untuk menuliskan panduan untuk proses instalasi program terutama bila ada pengaturan khusus, untuk menghindari kebingungan dan *user* dapat menggunakan program Anda secepatnya.
7. Gunakan contoh bila perlu untuk membantu untuk mendemonstrasikan kegunaan program Anda, bila terlalu panjang dapat diberikan berupa *link*.
8. Tuliskan bagaimana *user* dapat memperoleh bantuan tentang program yang Anda buat. Dapat berupa kombinasi dari *issue tracker*, *chat room*, alamat *e-mail*, dan lain sebagainya.
9. Bila program Anda bersifat *open source*, ada kemungkinan *user* ingin melakukan perubahan pada program Anda, akan sangat membantu bila ada informasi dimana mereka harus mulai. Mungkin ada script yang harus mereka jalankan atau variabel lingkungan yg harus diatur. Tuliskan dengan

jelas langkah-langkah ini, informasi ini juga mungkin akan berguna untuk Anda sendiri di kemudian hari.

10. Bila Anda berniat untuk membuat versi berikutnya, ide bagus untuk menulisakannya di *file* "readme".
11. Bila Anda terbuka terhadap kontribusi dari luar, informasikan apa persyaratan agar kontribusinya dapat diterima.
12. Tunjukan apresiasi Anda terhadap semua yang telah berkontribusi dalam proyek Anda dengan menuliskannya.
13. Untuk proyek *open source*, informasikan cara untuk lisensinya.
14. Bila Anda kehabisan tenaga atau waktu untuk proyek Anda, tambahkan catatan di atas *file* "readme" yang menginformasikan status proyek Anda dan mengatakan bahwa pengembangan (*development*) akan berjalan lambat atau dihentikan seluruhnya. Hal ini dapat menjadi info bagi pembaca untuk mengambil alih proyek Anda atau secara sukarela bergabung sebagai pengurus (*maintainer*) atau pemilik (*owner*) agar proyek Anda dapat tetap berjalan. Anda juga dapat secara eksplisit meminta bantuan *maintener*.

*File* "readme" dapat ditulis di aplikasi yang mendukung pembuatan *file* format teks (*text file format*) atau *text editor* manapun. Selain itu ada juga aplikasi yang menyediakan template *file* "readme".

## **B. Keterampilan yang Diperlukan dalam Mengorganisasikan Sumber Daya Pemrograman Sesuai Konteks**

1. Menyusun *folder* dan sub-sub *folder* sesuai konteks dan isinya.
2. Membuat *file* "readme" yang mengandung penjelasan mengenai struktur/ hirarki *folder* serta penjelasan mengenai sumber daya pemrograman.

### C. Sikap kerja

Harus bersikap secara:

1. Cekatan, cermat, disiplin, dan bertanggung jawab dalam menyusun *folder* dan sub-sub *folder* sesuai konteks dan isinya.
2. Cekatan, cermat, disiplin, dan bertanggung jawab dalam membuat *file* "readme" yang mengandung penjelasan mengenai struktur/ hirarki *folder* serta penjelasan mengenai sumber daya pemrograman.



## BUKU INFORMASI

# MENULIS KODE DENGAN PRINSIP SESUAI GUIDELINES DAN BEST PRACTICES **J.620100.016.01**



KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

Modul Pelatihan Berbasis Kompetensi Bidang Software Development	Kode Modul J.620100.016.01
<b>DAFTAR ISI</b>	
<b>DAFTAR ISI -----</b>	<b>2</b>
<b>BAB I PENDAHULUAN -----</b>	<b>4</b>
A. Tujuan Umum -----	4
B. Tujuan Khusus -----	4
<b>BAB II MENERAPKAN CODING GUIDELINES dan BEST PRACTICE dalam PENULISAN PROGRAM (KODE SUMBER) -----</b>	<b>5</b>
A. Menerapkan Coding-Guidelines dan Best Practices dalam Penulisan Program (Kode Sumber) -----	5
1. Pengertian Guidelines dan Best Practice -----	5
2. Lingkup panduan/ best practice dalam pembuatan kode program	5
B. Keterampilan yang Diperlukan dalam Menerapkan Coding-Guidelines dan Best Practices dalam Penulisan Program (Kode Sumber)-----	11
Sikap Kerja dalam yang Diperlukan dalam Menerapkan Coding-Guidelines dan Best Practices dalam Penulisan Program-----	11
<b>BAB III MENGGUNAKAN UKURAN PERFORMANSI dalam MENULISKAN KODE SUMBER -----</b>	<b>12</b>
A. Pengetahuan yang Diperlukan dalam Menggunakan ukuran performansi dalam menuliskan kode sumber -----	12
1. Cara mengidentifikasi resources komputasi yang diperlukan -----	12
2. Cara memperkirakan kebutuhan resource oleh program	17
B. Keterampilan yang Diperlukan dalam Menggunakan ukuran performansi dalam menuliskan kode sumber -----	18
C. Sikap Kerja yang Diperlukan dalam Menggunakan ukuran performansi dalam menuliskan kode sumber-----	19
<b>DAFTAR PUSTAKA -----</b>	<b>20</b>
A. Dasar Perundang-undangan -----	20
B. Buku Referensi -----	20
C. Majalah atau Buletin-----	20
D. Referensi Lainnya -----	20
<b>DAFTAR PERALATAN/MESIN DAN BAHAN -----</b>	<b>21</b>

Modul Pelatihan Berbasis Kompetensi Bidang Software Development	Kode Modul J.620100.016.01
A. Daftar Peralatan/Mesin-----	21
B. Daftar Bahan-----	21
DAFTAR PENYUSUN.....	21
Judul Modul Menulis kode dengan prinsip sesuai guidelines dan best practice Versi: 2018	Halaman: 3 dari 21

## BAB I

### PENDAHULUAN

#### A. Tujuan Umum

Setelah mempelajari modul ini peserta latih diharapkan mampu menulis kode dengan prinsip sesuai *guidelines* dan *best practice* dengan benar.

#### B. Tujuan Khusus

Adapun tujuan mempelajari unit kompetensi melalui buku informasi menulis kode dengan prinsip sesuai *guidelines* dan *best practice* ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut: (*diambil dari elemen dan KUK*)

1. Menerapkan coding- guidelines dan best practices dalam penulisan program (kode sumber)
2. Menggunakan ukuran performansi dalam menuliskan kode sumber

## **BAB II**

### **MENERAPKAN CODING- GUIDELINES DAN BEST PRACTICES DALAM PENULISAN PROGRAM (KODE SUMBER)**

#### **A. Pengetahuan yang diperlukan dalam menerapkan coding- guidelines dan best practices dalam penulisan program (kode sumber)**

##### **1. Guidelines dan Best Practice**

Guidelines adalah kumpulan aturan yang dijadikan acuan oleh suatu kelompok atau perusahaan.

Best Practice adalah kumpulan aturan informal yang diadopsi sekelompok pengguna karena dianggap praktek terbaik yang dikenal selama ini.

Dalam konteks pemrograman guidelines dan best practice perlu menjadi acuan dalam membuat kode program. Diharapkan dengan mengadopsi kedua pendekatan tersebut kode yang dihasilkan lebih dapat di-maintain, baik oleh pemrogram tersebut di masa depan maupun oleh orang lain. Kemudahan untuk me-maintain kode (maintainability) merupakan salah satu kualitas yang diharapkan dari kode program yang dihasilkan.

Guidelines dan best practice akan berkembang sesuai dengan perkembangan teknologi dan juga perkembangan Bahasa dan paradigm pemrograman, karenanya dalam dokumen ini akan dibahas hal-hal tentang hal tersebut, dan jika perlu beberapa konvensi terkait Bahasa pemrograman tertentu juga ditampilkan. Detil konvensi bagi Bahasa tertentu dapat dilihat pada dokumen terkait dengan Bahasa pemrograman tersebut dan tidak dibahas secara detil disini.

##### **2. Lingkup panduan/ best practice dalam pembuatan kode program**

###### **1) Layout**

Cara menuliskan kode mempengaruhi kemudahan kode untuk dimengerti baik oleh penulis kode tersebut (terutama setelah kode tersebut ditulis pada waktu yang telah lalu (lama)). Meski pada mayoritas bahasa pemrograman tidak ada keharusan (aturan) dalam penulisan kode, sebagai perkecualian adalah Phyton) namun pada prakteknya sebaiknya guideline/best practice

penulisan kode diperhatikan.

## A. Komentar

Komentar berguna untuk menjelaskan kode. Terdapat dua jenis komentar, *block comment* dan *line comment* (disebut juga sebagai *inline comment*). Block comment adalah komentar yang lebih dari satu baris (biasanya diberi tanda awal dan akhir), sementara Line comment adalah komentar satu baris (sering juga di bagian kanan suatu instruksi).

```
/*
Ini isi komentar blok .
Variasi 1

*/
*****\n*
* Ini isi komentar blok . Variasi2 *
*                                         *\n*****
*/
```

Block comment adalah komentar yang lebih dari satu baris (biasanya diberi tanda awal dan akhir), sementara Line comment adalah komentar satu baris (sering juga di bagian kanan suatu instruksi).

Kegunaan komentar:

a. Sebagai pseucode (algoritma) pada tahap awal pembuatan kode

```
/* loop backwards through all elements returned by the server  
(they should be processed chronologically)*/  
for (i = (numElementsReturned - 1); i >= 0; i--){  
    /* process each element's data */  
    updatePattern(i, returnedElements[i]);  
}
```

b. Sebagai deskripsi untuk menunjukkan tujuan dari pembuat kode tersebut.

```
list = [f(b), f(b), f(c), f(d), f(a), ...];
// Need a stable sort. Besides, the performance really does not matter.
insertion sort(list);
```

- c. Untuk menggabungkan informasi tambahan seperti logo, diagram, flowchart dan lainnya

d. Meta data Komentar blok biasa diberikan di awal modul/ subroutine/

class yang menyatakan secara singkat deskripsi kode tsb termasuk informasi parameter masukan/keluaran, library yang dipakai, dll.

e. Debugging. Komentar untuk membantu dalam proses debugging dengan menjadikan sebagian kode sebagai komentar agar tidak dieksekusi. Sebagai contoh:

```
if (opt.equals ("e"))
    opt_enabled = true;

/*
if (opt.equals ("d"))
    opt_debug = true;
*/

if (opt.equals ("v"))
    opt_verbose = true;

list = [f (b), f (b), f (c), f (d), f (a), ...];
// Need a stable sort. Besides, the performance really does no
insertion_sort (list);
```

## B. Indentasi

Sebagai contoh perhatikan dua potongan kode di bawah ini.

```
public class Main {
    public int number;
    public void assignIfPositive(int a) {
        if (a > 0) {
            System.out.println("a is positive");
            number = a;
        }
    }
}
```

```
public class Main {
    public int number;
    public void assignIfPositive(int a) {
        if (a > 0) {
            System.out.println("a is positive");
            number = a;
        }
    }
}
```

Bagi mesin, kedua kode tersebut tidak berbeda namun bagi manusia kode yang di kanan lebih baik karena lebih mudah dimengerti. Pada kode tersebut,

indentasi dipergunakan untuk mengelompokkan perintah dalam satu blok untuk mempermudah struktur program. Program dalam hal ini dibuat sebagai alat komunikasi antar manusia. Dengan indentation misalnya selain (lebih) mudah dimengerti juga lebih memudahkan mencari bug/kesalahan.

Terdapat beberapa metoda indentasi sebagai berikut.

**a. Penempatan kurung**

Kurung dapat ditempatkan terkait dengan kelompok pernyataan setelah perintah iterasi. Aturan penempatan tergantung dari style yang dipergunakan. Beberapa di antaranya adalah sebagai berikut.

```
while (x == y) {  
    something();  
    somethingelse();  
}
```

```
while (x == y)  
{    something();  
    somethingelse();  
}
```

```
while (x == y)  
{  
    something();  
    somethingelse();  
}
```

```
while (x == y)  
{    something();  
    somethingelse(); }
```

```
while (x == y)  
{  
    something ();  
    somethingelse ();  
}
```

```
while (x == y) {  
    something();  
    somethingelse();  
}
```

```
while (x == y)  
{  
    something();  
    somethingelse();  
}
```

```
while (x == y)  
{ something();  
    somethingelse(); }
```

**b. Indentasi Vertikal**

Selain indentasi biasa, indentasi vertical dipergunakan untuk memudahkan terbacanya bug karena kesalahan pengetikan. Perhatikan contoh di bawah ini.

```
const columns = [
    { name: 'id'      , index: 'id'           , width: 55 },
    { name: 'invdate', index: 'invdate'       , width: 90 },
    { name: 'name'    , index: 'name asc, invdate', width: 100 },
    { name: 'amount'  , index: 'amount'        , width: 80 , align: "right" },
    { name: 'tax'     , index: 'tax'           , width: 80 , align: "right" },
    { name: 'total'   , index: 'total'         , width: 80 , align: "right" },
    { name: 'note'    , index: 'note'          , width: 150, sortable: false }
];
```

**2) Penamaan**

Berbagai elemen program harus diberi nama seperti variable, konstanta, subroutine, modul, kelas, dll. Penamaan harus mengikuti aturan yang konsisten untuk memudahkan mengerti alur logika program tersebut. Nama yang dipilih harus menggambarkan apa yang dimaksud dan bukan mengenai bagaimana. Idealnya, program harus bisa dibaca seperti suatu karangan atau prosa.

Beberapa aturan penamaan adalah sebagai berikut.

- Pergunakan nama yang jelas, tidak mempunyai berbagai arti.

Misalkan :

```
Analyse(...) // subroutine or function or method
nnsmcomp1 // variable
```

Nama yang mempunyai banyak arti tidak menjelaskan bahkan membingungkan. Nama variable satu huruf (i, atau j) hanya dipergunakan pada iterasi saja.

- Pergunakan kata kerja untuk memberi nama routine/prosedur/ function atau method yang melakukan suatu operasi/proses terhadap objek tertentu. Nama dibentuk dengan menggabungkan beberapa kata dengan huruf pertama huruf lalu diteruskan dengan kata-kata berawal

huruf besar atau kata yang diawali “\_” (garis bawah). Contoh sbb.

```
calculateKineticEnergy (...)  
calculate_kinetic_energy (...)
```

- c. Pada Bahasa berorientasi objek, jangan melakukan pengulangan kata nama kelas sebagai bagian dari nama method/ function.

```
class solver {  
    // DON'T USE  
    int solverGridSize;  
    // USE INSTEAD  
    int gridSize;  
    ...
```

- d. Variabel yang bernilai Boolean (T/F) ataupun function diberi awalan “is” untuk menjelaskan nilai benar/salahnya.

```
if ( dataIsLoaded ) {                                // boolean variable  
    ...  
  
while ( ! simulation.isFinished() ) { // member function returning a bool  
    ...
```

- e. Jangan menggunakan term seperti “Flag”, Penanda selain untuk variable bernilai benar/salah.

```
integrationFlag          // expect a boolean type  
integrationMethodType   // expect a status value
```

- f. Variabel global atau local lebih baik diberi awalan g\_ atau l\_. Konstanta (literal) diberi nama dengan huruf besar semua dengan garis bawah sbg kata sambung.

```
# define M_PI      3.14159265358979323846 /* pi */  
# define M_SQRT2   1.41421356237309504880 /* sqrt(2) *
```

### 3) Variabel

#### a. Deklarasi

Variabel sebaiknya di deklarasikan meskipun beberapa Bahasa memperbolehkan penggunaan variable yang belum dideklarasikan sebelumnya. Selain itu nilai awal langsung diberikan ketika deklarasi. Setiap variable dideklarasikan dalam satu baris agar lebih jelas. Dalam Bahasa Java:

int scoreBob, scoreJohn = 10;  
hanya memberikan nilai 10 kepada scoreJohn sementara scoreBob tidak mendapat nilai awal. Sebaiknya sbb.

int scoreBob = 10;

int scoreJohn = 10;

b. *Scoping*

Minimalkan scope dari variable. Variabel hanya dapat dipergunakan pada bagian program tertentu.

4) Struktur *kendali*

Kondisional

Iterasi

5) *Struktur Modul*

Parameter

Keterkaitan

6) *Kelas dan Objek*

Arti

Polymorph

Pemanfaatan *berlebih*

**B. Keterampilan yang diperlukan dalam Menulis Kode dengan Prinsip sesuai Guidelines dan Best Practices**

1. Mengidentifikasi guidelines dan best practice yang dilakukan
2. Mendeteksi ketidaksesuaian kode dengan acuan
3. Mengubah kode agar sesuai dengan acuan tersebut

**C. Sikap Kerja yang diperlukan dalam Menulis Kode dengan Prinsip sesuai Guidelines dan Best Practices**

1. Harus cermat dalam mengidentifikasi konvensi yang perlu diterapkan sesuai dengan paradigma pemrograman yang dipergunakan
2. Harus cermat dalam mengubah kode agar sesuai dengan konvensi
3. Harus konsisten dalam menggunakan konvensi

## **BAB III**

### **MENGGUNAKAN UKURAN PERFORMANSI DALAM MENULISKAN KODE SUMBER**

#### **A. Pengetahuan yang diperlukan dalam Menggunakan Ukuran Performansi dalam Menuliskan Kode Sumber**

##### **1. Cara mengidentifikasi resources komputasi yang diperlukan**

Setiap program atau kode ketika dijalankan akan menggunakan sumber daya komputasi, seperti memori (space) dan waktu eksekusi. Kode yang baik selain bekerja sesuai dengan spesifikasinya juga harus efisien dalam menggunakan sumber daya komputasi tersebut.

Resources meliputi penggunaan memori dan lama eksekusi. Efisiensi dalam kode sumber terkait dengan efisiensi langkah proses (kecepatan) dan efisiensi penggunaan memori.

Setiap kode memiliki kompleksitas yang menggambarkan bagaimana resources yang diperlukan. Biasanya sumber daya dituliskan dalam suatu notasi, semacam big O(f), f adalah fungsi yang menyatakan hubungan antara sumber daya yang dipakai dengan jumlah masukan. Tabel di bawah menggambarkan hubungan tersebut.

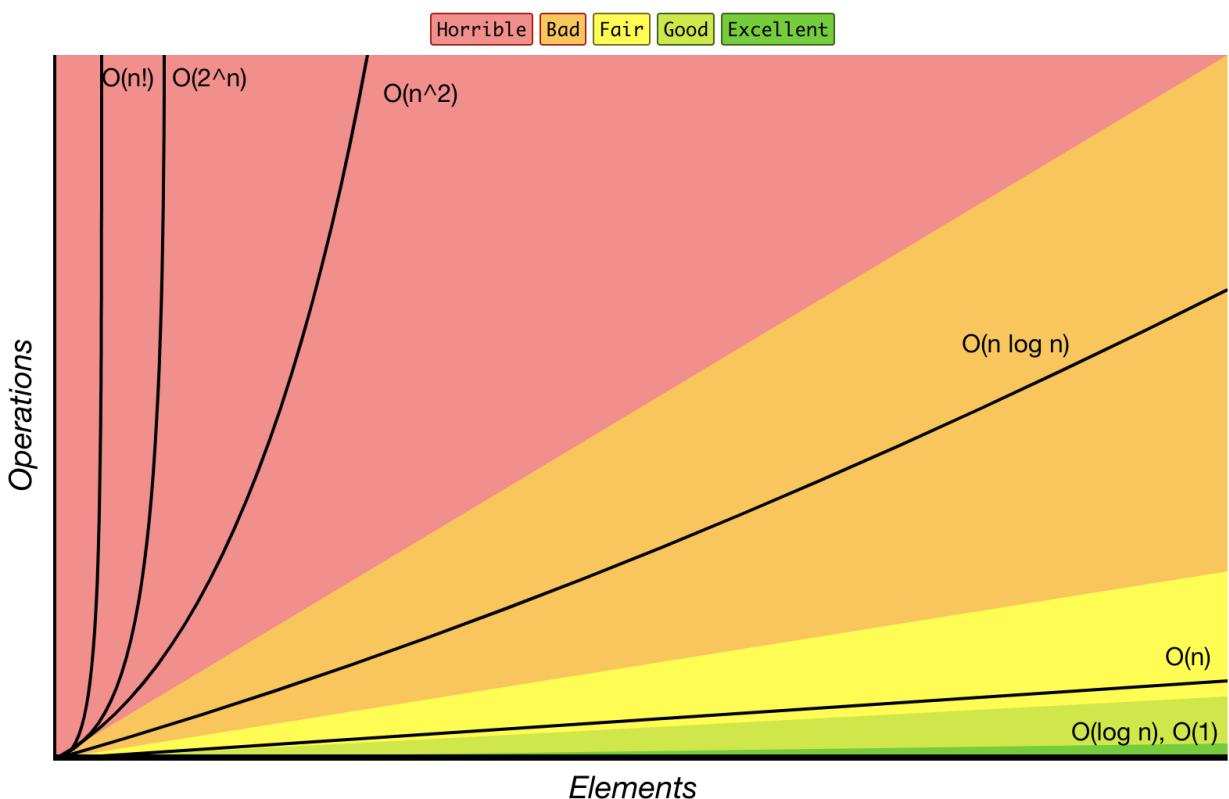
Laju Pertumbuhan	Nama	Arti
1	Konstan/Tetap	Resource yang diperlukan tidak terkait dengan jumlah data yang diproses
Log(n)	logarithmik	Resource berkembang dalam skala logarithmik terhadap jumlah data
n	linear	Resource berkembang linier sesuai jumlah data
n log(n)	linearithmik	Resource berkembang linier sesuai jumlah data
n^2	Kuadratik	Resource berkembang linier sesuai jumlah data
n^3	Kubik	Resource berkembang n^3 terhadap

		penambahan jumlah data
$2^n$	Eksponensial	Resource berkembang eksponensial dengan jumlah data

Untuk sumber daya waktu eksekusi maka notasi di atas dapat diberikan contoh seperti di bawah ini.

Laju Pertumbuhan	Nama	Contoh Kode	Penjelasan
$O(1)$	Konstan/Tetap	$a = b + 1;$	Satu perintah dilakukan sekali
$O(\log(n))$	logarithmik	<pre>while(n&gt;1){     n=n/2; }</pre>	Pencarian biner, lama waktu berbanding logaritmik dengan jumlah data ( $n$ )
$O(n)$	linear	<pre>for(c=0; c&lt;n; c++){     a+=1; }</pre>	Iterasi, lama waktu sesuai dengan jumlah data
$O(n \log(n))$	linearithmik	Quicksort, Mergesort	Algoritma/program pengurutan yang efisien
$O(n^2)$	Kuadratik	<pre>for(c=0; c&lt;n; c++){     for(i=0; i&lt;n; i++){         a+=1;     } }</pre>	Iterasi ganda (iterasi di dalam iterasi)
$O(n^3)$	Kubik	<pre>for(c=0; c&lt;n; c++){     for(i=0; i&lt;n; i++){         for(x=0; x&lt;n; x++){             a+=1;         }     } }</pre>	Iterasi tripel (iterasi di dalam iterasi yang berada di dalam iterasi lain).

		}	
O( $2^n$ )	Eksponensial	Mencoba memecahkan password dengan mencobakan semua kombinasi yang mungkin	Pencarian lengkap (Exhaustive search)



Kompleksitas kode menggambarkan hubungan antara waktu eksekusi (jumlah operasi) dengan jumlah data (Elements). Ada kode yang baik (excellent) artinya meski data bertambah besar maka waktu eksekusinya tidak terlalu berubah, sementara ada algoritma yang buruk (horrible) karena waktu eksekusi akan meningkat secara drastis ketika data bertambah. Setiap pemrogram harus mengetahui kompleksitas kode untuk tahu kelebihan/kekurangan kodenya terkait dengan waktu eksekusi.

Penjelasan dari beberapa fungsi kompleksitas untuk waktu eksekusi adalah sebagai berikut.

- O(1) Konstan/ Tetap

Algoritma/ kode dengan kompleksitas konstan akan membutuhkan waktu eksekusi yang sama tidak tergantung pada ukuran dari masukan. Contoh, adalah waktu untuk mengakses satu nilai dari array berindeks. Contoh lain adalah proses pop() dan push() terhadap array.

```
var arr = [ 1,2,3,4,5];
arr[2]; // => 3
```

- O(n) Linear

Algoritma/kode memiliki kompleksitas waktu linier jika waktu untuk eksekusinya bertambah lama sesuai dengan perubahan jumlah masukan. Hal ini seperti waktu yang diperlukan untuk membaca buku yang tergantung dari jumlah halaman buku tersebut. Semakin tebal buku (jumlah halaman bertambah) waktu yang diperlukan untuk membaca juga bertambah.

Contoh pada proses pencarian linier (linear search):

```
//Jika iterasi dipergunakan untuk mencetak isi array
for (var i = 0; i < array.length; i++) {
    console.log(array[i]);
}
var arr1 = [orange, apple, banana, lemon]; //=> 4 langkahs
var arr2 = [apple, htc,samsung, sony, motorola]; //=> 5 langkah
```

- O(log n) - Logarithmic

Algoritma/kode dengan kompleksitas logaritma jika waktu untuk eksekusinya selaras dengan logaritma dari jumlah masukan n.

Contoh 1;

```
for (var i = 1; i < n; i = i * 2)
    console.log(i);
}
```

Contoh 2;

```
for (i = n; i >= 1; i = i/2)
    console.log(i);
}
```

- O(n^2) - Quadratic

Algoritma/kode dengan kompleksitas kuadratik jika waktu untuk eksekusinya selaras dengan kuadrat dari jumlah masukan n. Kode yang berisi iterasi bersarang (iterasi dalam iterasi) terhadap jumlah data akan memberikan kompleksitas kuadratik. Kode pengurutan seperti bubble sort, selection sort, dan insertion sort mempunyai kompleksitas kuadratik.

Contoh:

```
for(var i = 0; i < length; i++) { //has O(n) time complexity
    for(var j = 0; j < length; j++) { //has O(n^2) time complexity
        // More loops?
    }
}
```

- $O(n^3)$  - Kubik

Algoritma/kode dengan kompleksitas kubik jika waktu untuk eksekusinya selaras dengan pangkat tiga dari jumlah masukan n. Kode yang berisi iterasi bersarang dua kali (iterasi dalam iterasi yang berada dalam iterasi) terhadap jumlah data akan memberikan kompleksitas kubik.

Contoh:

```
for(var i = 0; i < length; i++) { //has O(n) time complexity
    for(var j = 0; j < length; j++) { //has O(n^2) time complexity
        for(var k = 0; k < length; k++) { //has O(n^3) time complexity
        }
    }
}
```

Fungsi kompleksitas juga bisa diinterpretasikan untuk memperkirakan space (memori) yang dibutuhkan. Dalam konteks memori yang dipergunakan maka interpretasi dari beberapa fungsi tersebut adalah sbb.

- $O(1)$  Konstan/ Tetap

Memori yang dipergunakan tidak tergantung dari jumlah data yang diolah, meskipun data yang diolah lebih banyak namun memori tidak berubah. Salah satu contoh adalah jika membaca data nilai siswa dari file untuk dicari rata-rata nilai. Jika data tersebut tidak disimpan namun langsung dijumlahkan maka memori tetap.

- $O(n)$  Linear

Jumlah space tergantung jumlah data. Sebagai contoh adalah proses membaca data nilai dari file dan menyimpan nilai masing-masing siswa dalam array (dinamis) maka space yang dibutuhkan tergantung jumlah data.

## 2. Cara memperkirakan kebutuhan resource oleh program

Dalam memperkirakan sumber yang akan dipergunakan suatu kode program, tabel di bawah bisa dipergunakan.

- Kompleksitas dari operasi struktur data

**Common Data Structure Operations**

Data Structure	Time Complexity								Space Complexity	
	Average				Worst					
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion		
Array	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	
Stack	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	
Queue	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	
Singly-Linked List	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	
Doubly-Linked List	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	
Skip List	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n \log(n))$	
Hash Table	N/A	$O(1)$	$O(1)$	$O(1)$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$	
Binary Search Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	
Cartesian Tree	N/A	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$	
B-Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	
Red-Black Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	
Splay Tree	N/A	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	N/A	$O(\log(n))$	$O(\log(n))$	$O(n)$	
AVL Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	
KD Tree	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	

Tabel tersebut menggambarkan kompleksitas proses tertentu dengan menggunakan data yang ditulis dalam struktur tertentu. Kompleksitas dinyatakan untuk keadaan terbaik dan terburuk. Meskipun kompleksitas kode berada pada  $O(f)$  tertentu namun dalam banyak hal kompleksitas bisa lebih baik daripada itu, Sebagai contoh baris pertama menyatakan bahwa untuk data yang dinyatakan dalam bentuk array maka waktu untuk:

- mengakses satu elemen array tersebut (Access) adalah  $O(1)$  artinya tidak tergantung pada ukuran array,
- proses mencari elemen tertentu (Search) adalah  $O(n)$  artinya waktunya tergantung pada ukuran array, semakin besar array semakin besar waktu yang dibutuhkan untuk mencari elemen tersebut,
- proses pemasukan data baru (insertion) juga  $O(n)$ ,

- proses penghapusan satu elemen data (delete) untuk array adalah  $O(n)$ , dst.

b. Kompleksitas dari proses pengurutan array

## Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	
Quicksort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Heapsort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Tree Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
Shell Sort	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
Cubesort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Untuk kode yang akan dikembangkan bisa dilihat proses dan struktur data mana yang dipergunakan atau algoritma pengurutan mana yang dipergunakan. Dengan melihat pada proses utama pada kode yang (akan) dibuat dapat diperkirakan serta jumlah data yang akan diproses akan dapat disimpulkan resource yang diperlukan.

## B. Keterampilan yang diperlukan dalam Menggunakan Ukuran Performansi dalam Menuliskan Kode Sumber

1. Mengidentifikasi resource komputasi yang diperlukan
2. Memperkirakan kebutuhan resource oleh program

**C. Sikap Kerja yang diperlukan dalam Menggunakan Ukuran Performansi dalam Menuliskan Kode Sumber**

4. Harus cermat dalam memilih metoda identifikasi
5. Harus teliti dalam melakukan perhitungan
6. Harus taat azas dalam melakukan perkiraan kebutuhan resource



**BUKU INFORMASI**

**MENGIMPLEMENTASIKAN PEMROGRAMAN**

**TERSTRUKTUR**

**J.620100.017.02**

KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

## DAFTAR ISI

<b>DAFTAR ISI -----</b>	<b>2</b>
<b>BAB I PENDAHULUAN -----</b>	<b>5</b>
A. Tujuan Umum -----	5
B. Tujuan Khusus -----	5
<b>BAB II MENGGUNAKAN TIPE DATA DAN <i>CONTROL PROGRAM</i> -----</b>	<b>6</b>
A. Pengetahuan yang Diperlukan dalam Menggunakan Tipe Data dan <i>Control Program</i> -----	6
1. Standar tipe data sesuai-----	6
2. Standar syntax program -----	7
3. Standar struktur kontrol program -----	7
B. Keterampilan yang Diperlukan dalam Menggunakan Tipe Data dan <i>Control Program</i> -----	8
C. Sikap Kerja dalam Menggunakan Tipe Data dan <i>Control Program</i> -----	8
<b>BAB III MEMBUAT PROGRAM SEDERHANA -----</b>	<b>9</b>
A. Pengetahuan yang Diperlukan dalam Membuat Program Sederhana ---	9
1. Standar masukan / keluaran-----	9
2. Program baca tulis untuk memasukan data dari keyboard dan menampilkan ke layar monitor -----	12
3. Struktur kontrol percabangan dan pengulangan dalam membuat program -----	14
B. Keterampilan yang Diperlukan dalam Membuat Program Sederhana ---	17
C. Sikap Kerja yang Diperlukan dalam Membuat Program Sederhana -----	17
<b>BAB IV MEMBUAT PROGRAM MENGGUNAKAN PROSEDUR DAN FUNGSI -----</b>	<b>18</b>
A. Pengetahuan yang Diperlukan dalam Membuat Program Menggunakan Prosedur dan Fungsi -----	18
1. Pembuatan program dengan menggunakan prosedur beserta aturan penulisannya -----	18
2. Pembuatan program dengan menggunakan fungsi beserta aturan penulisannya -----	19

Modul Pelatihan Berbasis Kompetensi Bidang <i>Software Development</i> Subbidang Pemrograman	Kode Modul J.620100.017.02
3. Pembuatan program dengan menggunakan prosedur dan fungsi secara bersamaan ----- 20 4. Pembuatan keterangan untuk setiap prosedur dan fungsi ----- 22 B. Keterampilan yang Diperlukan dalam Membuat Program Menggunakan Prosedur dan Fungsi ----- 22 C. Sikap Kerja yang Diperlukan dalam Membuat Program Menggunakan Prosedur dan Fungsi ----- 22 <b>BAB V MEMBUAT PROGRAM MENGGUNAKAN <i>ARRAY</i> ----- 24</b> A. Pengetahuan yang Diperlukan dalam Membuat Program Menggunakan <i>Array</i> ----- 24 1. Dimensi <i>array</i> ----- 24 2. Tipe data <i>array</i> ----- 28 3. Panjang <i>array</i> ----- 28 4. Pengurutan <i>array</i> ----- 28 B. Keterampilan yang Diperlukan dalam Membuat Program Menggunakan <i>Array</i> ----- 30 C. Sikap Kerja yang Diperlukan dalam Membuat Program Menggunakan <i>Array</i> ----- 30 <b>BAB VI MEMBUAT PROGRAM UNTUK AKSES FILE ----- 31</b> A. Pengetahuan yang Diperlukan dalam Membuat Program untuk akses file ----- 31 1. Program untuk menulis data dalam media penyimpanan ----- 31 2. Program untuk membaca data dari media penyimpanan ----- 32 B. Keterampilan yang Diperlukan dalam Membuat Program untuk akses file ----- 33 C. Sikap Kerja yang Diperlukan dalam Membuat Program untuk akses file 33 <b>BAB VI MENGKOMPILASI PROGRAM ----- 34</b> A. Pengetahuan yang Diperlukan dalam Mengkompilasi Program ----- 34 1. Pengkoreksian kesalahan program ----- 34 2. Pembebasan program dari kesalahan syntax ----- 35 B. Keterampilan yang Diperlukan dalam Mengkompilasi Program ----- 36 C. Sikap Kerja yang Diperlukan dalam Mengkompilasi Program ----- 36	

Modul Pelatihan Berbasis Kompetensi Bidang <i>Software Development</i> Subbidang Pemrograman	Kode Modul J.620100.017.02
<hr/>	
DAFTAR PUSTAKA -----	37
A. Dasar Perundang-undangan -----	37
B. Buku Referensi -----	37
C. Majalah atau Buletin-----	37
D. Referensi Lainnya -----	37
DAFTAR PERALATAN/MESIN DAN BAHAN -----	38
A. Daftar Peralatan/Mesin-----	38
B. Daftar Bahan-----	38
DAFTAR PENYUSUN -----	39
<hr/>	

## BAB I

### PENDAHULUAN

#### A. Tujuan Umum

Setelah mempelajari modul ini peserta latih diharapkan mampu mengimplementasikan pemrograman terstruktur.

#### B. Tujuan Khusus

Adapun tujuan mempelajari unit kompetensi melalui buku informasi Mengimplementasikan Pemrograman Terstruktur ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut:

1. Dapat menggunakan tipe data dan *control program* yang meliputi kegiatan menentukan tipe data sesuai standar, menggunakan syntax program yang dikuasai sesuai standar dan menggunakan struktur kontrol program yang dikuasai sesuai standar.
2. Dapat membuat program sederhana yang meliputi kegiatan membuat program baca tulis untuk memasukkan data dari keyboard dan menampilkan ke layar monitor termasuk variasinya sesuai standar masukan/keluaran dan menggunakan struktur kontrol percabangan dan pengulangan dalam membuat program.
3. Dapat membuat program menggunakan prosedur dan fungsi yang meliputi kegiatan membuat program dengan menggunakan prosedur sesuai aturan penulisan program, membuat program dengan menggunakan fungsi sesuai aturan penulisan program, membuat program dengan menggunakan prosedur dan fungsi secara bersamaan sesuai aturan penulisan program dan membuat keterangan untuk setiap prosedur dan fungsi.
4. Dapat membuat program menggunakan *array* yang meliputi kegiatan menentukan dimensi *array*, menentukan tipe data *array*, menentukan panjang *array* dan menentukan pengurutan *array*.
5. Dapat membuat program untuk akses file yang meliputi kegiatan membuat program untuk menulis data dalam media penyimpanan dan membuat program untuk membaca data dari media penyimpanan.
6. Dapat mengkompilasi program yang meliputi kegiatan mengoreksi kesalahan program dan membebaskan program dari kesalahan syntax.

## **BAB II**

### **MENGGUNAKAN TIPE DATA DAN CONTROL PROGRAM**

#### **A. Pengetahuan yang Diperlukan dalam Menggunakan Tipe Data dan Control Program**

##### 1. Standar Tipe Data Sesuai

Variable dalam bahasa pemrograman adalah tempat untuk menampung data yang akan digunakan dalam proses pemrograman. Jenis data yang akan ditampung bisa berbeda-beda sehingga tipe data variable harus disesuaikan dengan data yang akan ditampung. Berikut ini adalah 7 tipe data dasar dalam bahasa pemrograman C++ :

Gambar 1

Tipe Data	Ukuran Memori	Jangkauan Nilai	Jumlah Digit
Char	1 Byte	-128 s.d 127	
Int	2 Byte	-2,147,435,648 s.d 2,147,435,647	
Short	2 Byte	-32768 s.d 32767	
Long	4 Byte	-2,147,435,648 s.d 2,147,435,647	
Float	4 Byte	$3.4 \times 10^{-38}$ s.d $3.4 \times 10^{+38}$	5 – 7
Double	8 Byte	$1.7 \times 10^{-308}$ s.d $1.7 \times 10^{+308}$	15 – 16
Long Double	10 Byte	$3.4 \times 10^{-4932}$ s.d $1.1 \times 10^{+4932}$	19

Selain tipe data dasar, terdapat juga tipe data tambahan yang dimiliki C++ dengan pemberian kata Unsigned di depan nama tipe data dan digunakan hanya untuk data positif saja.

Gambar 2

Tipe Data	Jumlah Memori	Jangkauan Nilai
Unsigned Integer	2 Byte	0 – 65535
Unsigned Character	1 Byte	0 – 255
Unsigned Long Integer	4 Byte	0 – 4,294,967,295

## 2. Standar *Syntax* Program

Struktur program c++ terdiri dari sejumlah blok fungsi, dan setiap blok fungsi tersebut memiliki satu atau beberapa pernyataan yang melaksanakan tugas tertentu. Setiap pernyataan yang ditulis harus diakhiri dengan tanda titik koma (;) untuk menandakan akhir dari sebuah pernyataan.

```
#include <iostream.h>
main()
{
    statement;
    .....
}
```

## 3. Standar Struktur Kontrol Program

Terdapat 3 struktur kontrol program di dalam bahasa pemrograman c++, antara lain:

- *Sequential structure* / urut, merupakan bentuk dasar dari urutan program yang di eksekusi dari atas ke bawah sesuai urutan penulisan kode program.
- *Selection* / *desicion structure* (pemilihan), merupakan pernyataan percabangan yang digunakan untuk memecahkan persoalan dalam pengambilan keputusan di antara sekian pernyataan yang ada. Struktur pemilihan dalam c++ terdiri dari *if*, *if-else*, dan *switch-case*.

```
// Struktur kontrol if                                // Struktur kontrol switch-case
if(kondisi)                                         switch( expression )
{
    statement;                                     {
}
                                            case kondisi1:
                                            statement1;
                                            break;
// Struktur kontrol if-else                         case kondisi2:
if(kondisi)                                         statement2;
{
    statement1;                                     break;
} else
{
    statement2;                                     default:
}
                                            statement3;
                                            break;
```

- *Repetition* / *looping structure* (perulangan), merupakan proses perulangan yang digunakan untuk mengulangi pernyataan yang telah dibuat. Bentuk struktur perulangan di dalam c++ antara lain, *for*, *while*, dan *do-while*.

```
// Struktur kontrol perulangan for
for (variabel = nilai_awal ; kondisi_stop ; variabel++)
{
    statement;
}

// Struktur kontrol perulangan while
while (kondisi)
{
    statement;
}

// Struktur kontrol perulangan do-while
do
{
    statement;
}
while (kondisi);
```

## **B. Keterampilan yang Diperlukan dalam Menggunakan Tipe Data dan Control Program**

1. Menentukan tipe data sesuai standar.
2. Menggunakan syntax program yang dikuasai sesuai standar.
3. Menggunakan struktur kontrol program yang dikuasai sesuai standar.

## **C. Sikap kerja**

Harus bersikap secara:

1. Dapat menerapkan sikap kerjasama dengan sesama peserta dalam menentukan tipe data sesuai standar.
2. Peserta dapat berperan aktif dan komunikatif dalam menentukan *syntax* program yang dikuasai sesuai standar.
3. Analitik dalam menggunakan struktur kontrol program yang dikuasai sesuai standar.

## BAB III

### MEMBUAT PROGRAM SEDERHANA

#### A. Pengetahuan yang Diperlukan dalam Membuat Program Sederhana

1. Standar masukan / keluaran.

Perintah standar *input* / masukan dalam c++ antara lain :

- *Scanf()*, digunakan untuk memasukan berbagai jenis data. Untuk data tipe *string* yang mengandung spasi atau *tab* akan dianggap sebagai data terpisah.

```

/*
Format Penulisan : scanf("penentu_format", &nama_variabel);

Keterangan :
Simbol & merupakan pointer yang digunakan untuk menunjuk ke alamat
variabel memori yang dituju
*/

scanf("%d", &a);
scanf("%d", &a);
```

Berikut ini adalah penentu format untuk masing-masing tipe data :

No	Tipe Data	Penentu Format untuk printf()
1	Integer	%d
2	Floating Point	
	✓ Bentuk Desimal	%e atau %f
	✓ Bentuk Berpangkat	%e atau %f
3	Double Precision	%lf
4	Character	%c
5	String	%s
6	Unsigned Integer	%u
7	Long Integer	%ld
8	Long unsigned integer	%lu
9	Unsigned Hexadecimal Integer	%x
10	Unsigned Octal Integer	%o

- *Gets()*, digunakan untuk memasukan data *string*, perbedaanya dengan *scanf()* adalah dapat menerima *string* yang mengandung spasi atau *tab* dan masing-masing dianggap sebagai satu kesatuan data.

```

// Format Penulisan : gets(nama-variable-array);
gets(nama);
```

- *Cin()*, merupakan sebuah objek didalam c++ yang digunakan untuk memasukan suatu data. Untuk menggunakan fungsi *cin()* diperlukan file header *iostream.h*.

```
// Format Penulisan : cin>>nama_variabel;  
cin>>a;
```

- *Getch()* / *get character* and echo, dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan tidak akan ditampilkan di layar. *File header* yang harus disertakan adalah *conio.h*.

```
// Format Penulisan : nama_variabel = getch();  
a = getch();
```

- *Getche()*, dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan ditampilkan di layar. *File header* yang harus disertakan adalah *conio.h*.

```
// Format Penulisan : nama_variabel = getche();  
a = getche();
```

Selain itu fungsi getch() dan getche() dapat digunakan untuk menahan agar tidak langsung balik kembali ke dalam *listing* program dan hasil dari program yang di eksekusi dapat dilihat. Karena fungsi getch() merupakan fungsi masukan, maka sebelum program keluar harus dimasukan sebuah karakter terlebih dahulu.

Perintah standar *output* / keluaran dalam c++ antara lain :

- *Printf()*, merupakan fungsi keluaran yang paling umum digunakan untuk menampilkan informasi ke layar.

```
// Format Penulisan : printf("string-kontrol", argumen-1, argumen-2, ...);  
  
int a = 1;  
char b = 'A'  
printf("%c Merupakan Abjad ke - %d", b, a);  
  
// Hasil = A Merupakan Abjad ke - 1
```

Keterangan :

*String* kontrol dapat berupa keterangan yang akan ditampilkan pada layar berserta penentu formatnya. Penentu format dipakai untuk memberi tahu kompiler mengenai jenis data yang dipakai dan akan ditampilkan.

Untuk argumen dapat berupa variabel, konstanta dan ungkapan.

Gambar 3 Penentu Format *printf()*

No	Tipe Data	Penentu Format untuk printf()
1	Integer	%d
2	Floating Point	
	✓ Bentuk Desimal	%e atau %f
	✓ Bentuk Berpangkat	%e atau %f
3	Double Precision	%lf
4	Character	%c
5	String	%s
6	Unsigned Integer	%u
7	Long Integer	%ld
8	Long unsigned integer	%lu
9	Unsigned Hexadecimal Integer	%x
10	Unsigned Octal Integer	%o

- *Puts()*, berasal dari kata PUT STRING fungsi penggunaannya sama dengan *printf()* yaitu digunakan untuk mencetak *string* kelayar. Perbedaannya adalah pada *printf()* harus menentukan tipe data untuk *string* dan untuk mencetak pindah baris memerlukan notasi \n, sedangkan pada *puts()* tidak perlu penentu tipe data *string* karena fungsi ini khusus untuk tipe data *string* dan untuk mencetak pindah baris tidak perlu notasi \n karena sudah diberikan secara otomatis.

```
// Format Penulisan : puts("String");

char a[10] = "Programmer";
puts("Profesi saya adalah ");
puts(a);

/* Hasil :
   Profesi saya adalah
   Programmer
*/
```

- *Putchar()*, digunakan untuk menampilkan sebuah karakter ke layar dengan penampilan karakter tidak diakhiri dengan pindah baris.

```
// Format Penulisan : putchar('char');
putchar('A');
putchar('B');
putchar('C');

// Hasil = ABC
```

- *Cout()*, merupakan sebuah objek di dalam c++ yang digunakan untuk menampilkan suatu data ke layar. Untuk menggunakan fungsi ini diperlukan file header *iostream.h*.

```
// Format Penulisan : cout<<"String data";  
  
cout<<"Saya adalah seorang Programmer";  
  
// Hasil = Saya adalah seorang Programmer
```

2. Program baca tulis untuk memasukkan data dari *keyboard* dan menampilkan ke layar monitor termasuk variasinya sesuai standar masukan/keluaran telah dibuat.

Dalam pembuatan program baca tulis dalam C++ yang perlu di persiapkan adalah variabel yang akan digunakan, jenis data yang akan dimasukan, proses yang diperlukan untuk menangani masukan tersebut, serta hasil keluaran yang akan ditampilkan. Program baca tulis yang akan dibuat menggunakan variasi fungsi standar masukan dan keluaran.

Berikut ini ada contoh program sederhana untuk memasukan data mahasiswa berupa nama, jurusan, nilai tugas, UTS, dan UAS, serta akan dicari nilai rata-rata dari nilai tersebut. Setelah itu hasil akan ditampilkan kelayar beserta nilai rata-rata yang sudah dihitung.

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

void main()
{
    // Deklarasi variabel yang dibutuhkan
    int tugas, uts, uas, rata_rata;
    char nama [50], jurusan[20];

    // Memasukan data dari keyboard
    printf("Nama = "); gets(nama);
    printf("Jurusan = "); scanf("%s",jurusan);
    cout<<"Nilai Tugas = "; cin>>tugas;
    cout<<"Nilai UTS = "; cin>>uts;
    cout<<"Nilai UAS = "; scanf("%d", &uas);

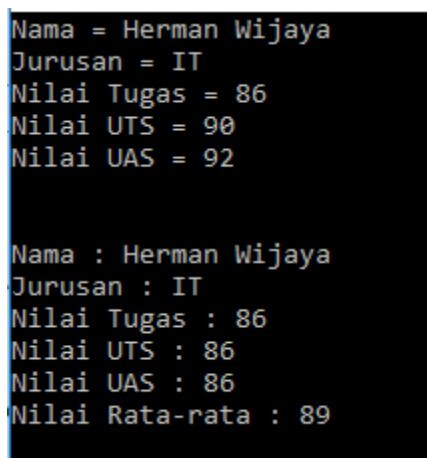
    // Proses perhitungan nilai rata-rata
    rata_rata = (tugas + uts + uas) / 3;

    printf("\n\n"); // mencetak pindah baris

    // Menampilkan hasil ke layar output
    printf("Nama : %s", nama);
    puts(""); // fungsi put() yang digunakan untuk mencetak pindah baris
    printf("Jurusan : %s", jurusan); puts("");
    cout<<"Nilai Tugas : "<<tugas; puts("");
    cout<<"Nilai UTS : "<<tugas; puts("");
    cout<<"Nilai UAS : "<<tugas; puts("");
    cout<<"Nilai Rata-rata : "<<rata_rata; puts("");

    getch(); // untuk menahan layar output / bisa juga menggunakan getche()
}
```

*Script* di atas merupakan contoh variasi penggunaan fungsi masukan dan keluaran yang dijelaskan sebelumnya. Berikut contoh *output script* di atas:



```
Nama = Herman Wijaya
Jurusan = IT
Nilai Tugas = 86
Nilai UTS = 90
Nilai UAS = 92

Nama : Herman Wijaya
Jurusan : IT
Nilai Tugas : 86
Nilai UTS : 86
Nilai UAS : 86
Nilai Rata-rata : 89
```

Dalam implementasinya, pemilihan fungsi masukan dan keluaran yang akan digunakan tidak harus bervariasi seperti contoh *script* di atas. Maksudnya, untuk menampilkan keluaran anda dapat memilih salah satu fungsi saja (*printf*, *puts* atau *cout*), begitu juga untuk fungsi masukan (*scanf*, *cin* atau *gets*). Penggunaan *syntax*

fungsi masukan atau keluaran disesuaikan dengan kebiasaan *programmer* masing – masing.

### 3. Struktur kontrol percabangan dan pengulangan dalam membuat program.

Fungsi struktur kontrol percabangan adalah untuk memecahkan masalah pemrograman yang memerlukan proses pengambilan keputusan dari beberapa kondisi yang ada.

- *If* dan *if-else*

Digunakan untuk menyatakan kondisi sederhana yang hanya memerlukan satu kondisi.

```
int a=10, b=11, c=2, d=2;
char grade;

// if standar
if(a < b){
    cout<<"Nilai A < Nilai B";
}

cout<<endl; // digunakan untuk pindah baris

// if-else
if(a != b){
    cout<<"Nilai A tidak sama Nilai B";
}
else{
    cout<<"Nilai A sama Nilai B";
}
```

- *If-else* majemuk dan *if*-bersarang

Penggunaan if-else majemuk bertujuan untuk menjalankan proses pemilihan kondisi yang lebih dari 1, sedangkan if-bersarang digunakan untuk kondisi if di dalam if lain.

```
// if-else Majemuk
if(c > d){
    cout<<"Nilai C lebih besar dari Nilai D";
}
else if(d > c){
    cout<<"Nilai D lebih besar dari Nilai C";
}
else{
    cout<<"Nilai C sama dengan Nilai D";
}

cout<<endl;

// if-bersarang
if(b > a){
    if(b > c){
        if(b > d){
            cout<<"Nilai B lebih besar dari Nilai A, C, dan D";
        }
        else{
            cout<<"Nilai B tidak lebih besar dari Nilai D";
        }
    }
    else{
        cout<<"Nilai B tidak lebih besar dari Nilai C";
    }
}
else{
    cout<<"Nilai B tidak lebih besar dari Nilai A";
}
```

- *Switch-case*

Struktur kontrol ini dirancang khusus untuk menangani pengambilan keputusan yang melibatkan banyak alternatif.

```
// switch-case
grade = 'B';
switch(grade)
{
    case 'A' :
        cout<<"Grade A";
        break;
    case 'B' :
        cout<<"Grade B";
        break;
    case 'C' :
        cout<<"Grade C";
        break;
    default :
        cout<<"No Grade";
        break;
}
```

Berikut ini ada hasil output dari script diatas :

```
Nilai A < Nilai B
Nilai A tidak sama Nilai B
Nilai C sama dengan Nilai D
Nilai B lebih besar dari Nilai A, C, dan D
Grade B
```

Sedangkan fungsi struktur kontrol pengulangan digunakan untuk menjalankan sebuah statement secara berulang sesuai dengan kondisi tertentu.

Berikut ini adalah contoh penggunaan *script* perulangan *for*, *for*-bersarang, *while*, dan *do-while* beserta hasil outputnya.

```
int i, j;
char grade;

cout<<"for"<<endl;
for(i = 1; i <= 10; i++)
{
    cout<<i<<" ";
}

cout<<"\n\nfor bersarang\n";
for(i = 1; i <= 10; i++)
{
    for(j = 1; j <= i; j++)
    {
        cout<<j<<" ";
    }
    cout<<endl;
}

cout<<"\n\nwhile\n";
i=1;
while(i<=10)
{
    cout<<i<<" ";
    i++;
}

i=1;
cout<<"\n\ndo-while\n";
do
{
    cout<<i<<" ";
    i++;
}
while(i<10);

for
1 2 3 4 5 6 7 8 9 10

for bersarang
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10

while
1 2 3 4 5 6 7 8 9 10

do-while
1 2 3 4 5 6 7 8 9
```

## **B. Keterampilan yang Diperlukan dalam Membuat Program Sederhana**

1. Membuat program baca tulis untuk memasukan data dari *keyboard* dan menampilkan ke layar monitor termasuk variasinya sesuai standar masukan / keluaran.
2. Menggunakan struktur kontrol percabangan dan pengulangan dalam membuat program

## **C. Sikap kerja**

1. Dapat bekerjasama dan komunikatif dengan sesama peserta dalam membuat program baca tulis untuk memasukan data dari keyboard dan menampilkan kelayar monitor termasuk variasinya sesuai standar masukan / keluaran.
2. Bersikap analitik dalam menggunakan struktur kontrol percabangan dan pengulangan dalam membuat program.

## **BAB IV**

### **MEMBUAT PROGRAM MENGGUNAKAN PROSEDUR DAN FUNGSI**

#### **A. Pengetahuan yang Diperlukan dalam Membuat Program Menggunakan Prosedur Dan Fungsi**

1. Pembuatan program dengan menggunakan prosedur beserta aturan penulisannya.

Prosedur merupakan sebuah *block* pernyataan yang dibungkus menjadi satu kesatuan dan bertugas untuk menjalankan proses tertentu. *Block* prosedur terpisah dari program utama dan hanya akan menjalankan proses yang di definisikan didalam *block* nya tanpa mengembalikan nilai apapun. Tujuan penggunaan prosedur adalah untuk mengurangi pengulangan dalam penulisan *syntax* program yang sama serta membuat *syntax* program lebih terstruktur dan mudah dipahami.

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

// Deklarasi Prosedur
void prosedur_tanpa_parameter()
{
    cout<<"Prosedur ini hanya akan menampilkan data";
}

void prosedur_dengan_parameter(int a, int b)
{
    int c = a + b;
    cout<<"Prosedur ini akan menampilkan Nilai "<<c;
}

// Program utama
void main()
{
    // Pemanggilan Prosedur tanpa parameter
    prosedur_tanpa_parameter();

    cout<<"\n\n";

    // Pemanggilan Prosedur dengan parameter
    prosedur_dengan_parameter(10, 6);
    getch();
}
```

Script diatas adalah contoh penggunaan prosedur didalam c++ dan akan menghasilkan *output* seperti berikut :

```
Prosedur ini hanya akan menampilkan data
Prosedur ini akan menampilkan Nilai 16
```

Dalam pembuatan prosedur terdapat beberapa aturan penulisan diantaranya :

- Pembuatan prosedur diawali dengan *syntax void* (tidak ada nilai balik) sebelum penulisan nama prosedur.
  - Nama prosedur boleh ditulis dengan bebas tanpa menggunakan spasi dan nama-nama fungsi yang sudah terdapat didalam bahasa pemrograman c++.
  - Penggunaan parameter prosedur diletakan di antara tanda kurung yang terletak di belakang nama prosedur dan penggunaan parameter bersifat *optional*.
  - Pernyataan atau perintah prosedur diletakan diantara tanda kurung kurawal.
2. Pembuatan program dengan menggunakan fungsi beserta aturan penulisannya.
- Pengertian fungsi sama dengan prosedur hanya saja fungsi memiliki nilai yang akan dikembalikan dan harus medefinisikan tipe dari nilai yang akan dikembalikan.
- Aturan penulisan dari fungsi :
- Penulisan nama fungsi diawali dengan tipe dari data yang menjadi nilai balik fungsi.
  - Nama fungsi ditulis tanpa spasi dan tidak boleh menggunakan fungsi yang sudah terdapat didalam bahasa pemrograman c++.
  - Parameter fungsi di letakan didalam tanda kurung setelah nama fungsi.
  - Pernyataan atau perintah fungsi diletakan didalam *block* fungsi atau diantara tanda kurung kurawal.
  - Nilai balik fungsi (*Return*) ditulis setelah perintah didalam badan *block* fungsi.

Berikut ini adalah contoh *script* penggunaan fungsi beserta hasilnya :

```
#include <constrea.h>

// Deklarasi Fungsi
float luas_lingkaran()
{
    int r=7;
    int luas = 3.14 * r * r;

    return luas; // Nilai balik
}

int luas_persegi(int sisi)
{
    int luas = sisi * sisi;

    return luas; // Nilai balik
}

// Program utama
void main()
{
    float lingkaran;
    int persegi;

    // Pemanggilan Fungsi tanpa parameter
    lingkaran = luas_lingkaran();

    cout<<"Luas Lingkaran = "<<lingkaran;

    cout<<"\n\n";

    // Pemanggilan Fungsi dengan parameter
    persegi = luas_persegi(5);

    cout<<"Luas Persegi = "<<persegi;

    getch();
}
```

```
Luas Lingkaran = 153
Luas Persegi = 25
```

3. Pembuatan program dengan menggunakan prosedur dan fungsi secara bersamaan.

Penggunaan prosedur dan fungsi sama-sama bertujuan untuk mencegah penulisan perintah / proses yang sama secara berulang-ulang sehingga struktur program menjadi lebih singkat dan mudah dipahami. Masing-masing dari prosedur dan fungsi dapat digunakan untuk mendefinisikan suatu tugas tertentu dan dapat menghasilkan nilai yang berbeda-beda sesuai dengan nilai yang di masukan kedalam parameter fungsi.

```
// Fungsi hitung luas lingkaran
float luas_lingkaran()
{
    int r=7;
    int luas = 3.14 * r * r;

    return luas; // Nilai balik
}

// Fungsi hitung luas persegi
int luas_persegi(int sisi)
{
    int luas = sisi * sisi;

    return luas; // Nilai balik
}

// Prosedur untuk menampilkan hasil dari perhitungan fungsi
void print_luas(float luas_lingkaran, int luas_persegi)
{
    cout<<"Luas lingkaran = "<<luas_lingkaran<<endl;
    cout<<"Luas Persegi = "<<luas_persegi;
}

// Program utama
void main()
{
    float lingkaran;
    int persegi;

    // Pemanggilan Fungsi
    lingkaran = luas_lingkaran();
    persegi = luas_persegi(5);

    // Pemanggilan Prosedur
    print_luas(lingkaran, persegi);
    getch();
}
```

Pada contoh *script* diatas terdapat fungsi untuk menghitung luas lingkaran, fungsi untuk menghitung luas persegi dan prosedur untuk menampilkan hasil perhitungan dari kedua fungsi tersebut. Pada program utama nilai balik dari fungsi akan dijadikan sebagai parameter pada saat pemanggilan prosedur dan perintah di dalam prosedur akan menampilkan hasil luas lingkaran dan persegi.

```
Luas lingkaran = 153
Luas Persegi = 25
```

Penggunaan prosedur dan fungsi secara bersamaan dapat dilakukan sesuai dengan kebutuhan dan di kombinasikan dengan fungsi input serta fungsi-fungsi lain yang terdapat di dalam C++.

4. Pembuatan keterangan untuk prosedur dan fungsi.

Pembuatan keterangan/*comment* dalam sebuah blok *script* bertujuan untuk menjelaskan kegunaan dari setiap blok *script* yang dibuat. Hal ini sangat membantu programmer untuk memahami alur logika jalannya aplikasi, sehingga proses apa saja yang berjalan pada aplikasi dapat dipahami dengan lebih mudah oleh programmer yang membuat aplikasi ataupun programmer lain yang akan melanjutkan pengembangan aplikasi.

Tidak ada aturan khusus dalam pembuatan keterangan/*comment* dalam sebuah blok *script*, yang penting keterangan/*comment* yang dibuat dapat dimengerti dengan jelas dalam mempresentasikan kegunaan atau alur jalannya blok *script* terkait. Perhatikan contoh pembuatan keterangan di bawah ini.

```
// Fungsi hitung luas persegi
int luas_persegi(int sisi)
{
    int luas = sisi * sisi;

    return luas; // Nilai balik
}

// Prosedur untuk menampilkan hasil dari perhitungan fungsi
void print_luas(float luas_lingkaran, int luas_persegi)
{
    cout<<"Luas lingkaran = "<<luas_lingkaran<<endl;
    cout<<"Luas Persegi = "<<luas_persegi;
}
```

Dengan adanya keterangan pada baris kode seperti contoh di atas, setiap blok *script* yang dibuat dapat lebih mudah dipahami kegunaannya.

**B. Keterampilan yang Diperlukan dalam Membuat Program Menggunakan Prosedur Dan Fungsi**

1. Membuat program dengan menggunakan prosedur sesuai aturan penulisan program.
2. Membuat program dengan menggunakan fungsi sesuai aturan penulisan program.
3. Membuat program dengan menggunakan prosedur dan fungsi secara bersamaan sesuai aturan penulisan program.
4. Membuat keterangan untuk setiap prosedur dan fungsi

**C. Sikap Kerja**

1. Dapat bekerjasama antar sesama peserta dalam pembuatan program dengan menggunakan prosedur dan fungsi.

2. Peserta dapat komunikatif dan analitik dalam pembuatan keterangan untuk setiap prosedur dan fungsi.

## BAB V

### MEMBUAT PROGRAM MENGGUNAKAN *ARRAY*

#### A. Pengetahuan yang Diperlukan dalam Membuat Program Menggunakan Array

##### 1. Dimensi *Array*

*Array* atau larik adalah tipe data terstruktur yang terdiri dari sejumlah komponen yang mempunyai tipe data yang sama. *Array* mempunyai jumlah komponen yang tetap dan indeks pada *array* berfungsi untuk membedakan variabel yang satu dengan yang lainnya.

Dimensi *array* menunjukkan jumlah indeks yang dimiliki oleh suatu *array*, berikut adalah *jenis array* berdasarkan dimensinya :

- *Array* berdimensi satu

*Array* jenis ini merupakan *array* yang dibentuk menjadi satu baris data dengan kata lain hanya memiliki satu indeks.

Berikut adalah contoh *script* penggunaan *array* dimensi satu beserta *output* yang dihasilkan.

```
/*
    Bentuk umum pendeklarasian array dimensi satu :
    tipe_data nama_array[jumlah_maksimum_elemen_array];
*/
int nilai[5];
int i;

for(i = 1; i <= 5; i++)
{
    cout<<"Masukan nilai ke- "<<i<<" = ";
    cin>>nilai[i]; // Nilai ditulis kedalam masing-masing index
}

cout<<endl;

for(i = 1; i <= 5; i++)
{
    cout<<"Nilai ke- "<<i<<" = ";
    cout<<nilai[i]<<endl; // Nilai dibaca dari masing-masing index
}
```

```
Masukan nilai ke- 1 = 98
Masukan nilai ke- 2 = 100
Masukan nilai ke- 3 = 76
Masukan nilai ke- 4 = 88
Masukan nilai ke- 5 = 90

Nilai ke- 1 = 98
Nilai ke- 2 = 100
Nilai ke- 3 = 76
Nilai ke- 4 = 88
Nilai ke- 5 = 90
```

- *Array* berdimensi dua

*Array* dimensi dua tersusun dalam bentuk baris dan kolom, dimana indeks pertama menunjukkan baris dan indeks kedua menunjukkan kolom. *Array* jenis ini bisa digunakan untuk pendataan penjualan, pendataan nilai dan sebagainya.

Berikut adalah contoh *script* penggunaannya beserta *output* yang dihasilkan.

```
/*
    Bentuk umum pendeklarasian array dimensi dua :
    tipe_data nama_array[index-1][index-2];
 */

int nilai[3][3];
int i, j;

for(i = 1; i <= 3; i++) // Perulangan sesuai jumlah index-1
{
    cout<<"Nilai Semester "<<i<<endl;
    for(j = 1; j <= 3; j++) // Perulangan sesuai jumlah index-2
    {
        cout<<"Masukan Nilai ke - "<<j<<" = ";
        cin>>nilai[i][j]; // Nilai ditulis kedalam masing-masing index
    }
    cout<<endl;
}

for(i = 1; i <= 3; i++) // Perulangan sesuai jumlah index-1
{
    cout<<"Nilai Semester "<<i<<endl;
    for(j = 1; j <= 3; j++) // Perulangan sesuai jumlah index-2
    {
        cout<<"Nilai ke "<<j<<" : ";
        cout<<nilai[i][j]<<endl; // Nilai dibaca dari masing-masing index
    }
    cout<<endl;
}

getch();
```

```
Nilai Semester 1
Masukan Nilai ke - 1 = 100
Masukan Nilai ke - 2 = 78
Masukan Nilai ke - 3 = 96

Nilai Semester 2
Masukan Nilai ke - 1 = 98
Masukan Nilai ke - 2 = 97
Masukan Nilai ke - 3 = 80

Nilai Semester 3
Masukan Nilai ke - 1 = 76
Masukan Nilai ke - 2 = 75
Masukan Nilai ke - 3 = 90

Nilai Semester 1
Nilai ke 1 : 100
Nilai ke 2 : 78
Nilai ke 3 : 96

Nilai Semester 2
Nilai ke 1 : 98
Nilai ke 2 : 97
Nilai ke 3 : 4264708

Nilai Semester 3
Nilai ke 1 : 1
Nilai ke 2 : 4264708
Nilai ke 3 : 90
```

- *Array berdimensi tiga*

*Array berdimensi tiga* tersusun dalam bentuk baris, kolom dan isi dari baris dimana indeks pertama menunjukkan jumlah baris, indeks kedua menunjukkan jumlah kolom dan indeks ketiga menunjukkan jumlah isi dari baris.

Berikut ini adalah contoh *script* penggunaan *array* berdimensi tiga beserta *output* yang dihasilkan.

```

/*
Bentuk umum pendeklarasian array dimensi tiga :
tipe_data nama_array[index-1][index-2][index-3];
*/
int nilai[2][3][2];
int i, j, k;
for(i = 0; i < 2; i++) // Perulangan sesuai jumlah index-1
{
    cout<<"Mahasiswa ke "<<(i+1)<<endl;
    for(j = 0; j < 3; j++) // Perulangan sesuai jumlah index-2
    {
        cout<<"Nilai Semester "<<(j+1)<<endl;
        for(k = 0; k < 2; k++) // Perulangan sesuai jumlah index-3
        {
            cout<<"Masukan Nilai ke - "<<(k+1)<<" = ";
            cin>>nilai[i][j][k]; // Nilai ditulis kedalam masing-masing index
        }
        cout<<endl;
    }
    cout<<endl;
}

for(i = 0; i < 2; i++) // Perulangan sesuai jumlah index-1
{
    cout<<"Nilai Mahasiswa ke "<<(i+1)<<endl;
    for(j = 0; j < 3; j++) // Perulangan sesuai jumlah index-2
    {
        cout<<"Nilai Semester "<<(j+1)<<endl;
        for(k = 0; k < 2; k++) // Perulangan sesuai jumlah index-3
        {
            cout<<"Nilai ke "<<(k+1)<<" : ";
            cout<<nilai[i][j][k]<<endl; // Nilai dibaca dari masing-masing index
        }
        cout<<endl;
    }
    cout<<endl;
}
getch();

```

```

Nilai Mahasiswa ke 1
Nilai Semester 1
Nilai ke 1 : 90
Nilai ke 2 : 87

Nilai Semester 2
Nilai ke 1 : 95
Nilai ke 2 : 100

Nilai Semester 3
Nilai ke 1 : 78
Nilai ke 2 : 77

Nilai Mahasiswa ke 2
Nilai Semester 1
Nilai ke 1 : 90
Nilai ke 2 : 96

Nilai Semester 2
Nilai ke 1 : 90
Nilai ke 2 : 90

Nilai Semester 3
Nilai ke 1 : 80
Nilai ke 2 : 91

```

## 2. Tipe data *Array*

Tipe data yang dapat digunakan pada array ada tipe data primitive atau tipe data dasar. Berikut adalah contoh penggunaannya :

```
char nama[4][20] = {"Andi", "Aldo", "Adi", "Ade"};
char grade[4] = {'A','B','C','D'};
int nilai[4] = {80, 78, 56, 45};

int i, j, k;

for(i = 0; i < 4; i++)
{
    cout<<nama[i]<<" mendapatkan nilai "<<nilai[i]<<" dan grade "<<grade[i]<<endl;
}
getch();
```

Andi mendapatkan nilai 80 dan grade A  
Aldo mendapatkan nilai 78 dan grade B  
Adi mendapatkan nilai 56 dan grade C  
Ade mendapatkan nilai 45 dan grade D

## 3. Panjang *Array*

Panjang dari *array* ditentukan dengan memasukan ukuran maksimum pada tanda kurung siku setelah nama variabel *array*. Contoh jika ukuran *array* adalah 5 maka maksimum jumlah data yang dapat dimasukan adalah 5 data dan untuk mengakses data dari *array* tersebut dapat menggunakan nilai *index array* yang dimulai dari 0.

## 4. Pengurutan *Array*

Dalam penggunaan *array* dengan tipe data int, data yang terdapat di dalam *array* tersebut bisa dilakukan proses pengurutan secara ASC maupun DESC. Proses pengurutan *array* dilakukan dengan bantuan struktur kontrol perulangan besar dan variabel penampung. Dari perulangan yang pertama akan didapatkan *index-1* dan perulangan kedua sebagai *index-2*. Untuk pengurutan secara ASC proses perbandingan dilakukan untuk mendapatkan nilai yang terbesar antara *index-1* dan *index-2*. Nilai terbesar akan di masukan ke dalam variabel penampung, *index-1* diisi dengan nilai terkecil, dan *index-2* diisi dengan nilai yang di tampung sebelumnya. Proses berlanjut sampai perulangan selesai dan akan didapatkan hasil *array* dengan urutan data dari yang terkecil sampai terbesar. Pengurutan secara DESC akan melakukan proses yang sama, hanya saja perbandingan dilakukan untuk mendapatkan nilai terbesar.

```

// Prosedur ini berfungsi untuk melakukan proses pengurutan
// Parameter yang diterima adalah data_array dan jenis pengurutan
void arr_sort(int arr[10], char type[4])
{
    int i, j, tmp;
    for(i=0; i<9; i++) // Perulangan dari index 0
    {
        for(j=i+1; j<10; j++) // Perulangan dari index 1
        {
            if(strcmp(type, "DESC") == 0) // untuk membandingkan string
            {
                if(arr[i] < arr[j]) // Pengurutan DESC
                {
                    tmp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = tmp;
                }
            }
            else if(strcmp(type, "ASC") == 0) // untuk membandingkan string
            {
                if(arr[i] > arr[j]) // Pengurutan ASC
                {
                    tmp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = tmp;
                }
            }
        }
    }
}

void main ()
{
    int nilai[10] = {76, 90, 85, 76, 91, 95, 83, 88, 72, 65};
    int i;

    cout<<"Nilai sebelum diurutkan : "<<endl;
    for(i=0; i<10; i++)
    {
        cout<<nilai[i]<<" ";
    }

    arr_sort(nilai, "ASC"); // Memanggil prosedur untuk pengurutan secara ASC

    cout<<endl<<endl<<"Pengurutan secara ASC : "<<endl;
    for(i=0; i<10; i++)
    {
        cout<<nilai[i]<<" ";
    }

    arr_sort(nilai, "DESC"); // Memanggil prosedur untuk pengurutan secara DESC

    cout<<endl<<endl<<"Pengurutan secara DESC : "<<endl;

    for(i=0; i<10; i++)
    {
        cout<<nilai[i]<<" ";
    }
    getch();
}

```

Pada contoh *script* diatas proses pengurutan di lakukan di dalam prosedur yang menerima parameter masukan data *array* dan jenis pengurutan dalam bentuk *string* dan berikut adalah hasil outputnya :

```
Nilai sebelum diurutkan :  
76 90 85 76 91 95 83 88 72 65  
  
Pengurutan secara ASC :  
65 72 76 76 83 85 88 90 91 95  
  
Pengurutan secara DESC :  
95 91 90 88 85 83 76 76 72 65
```

## **B. Keterampilan yang Diperlukan dalam Membuat Program Menggunakan Array**

1. Menentukan dimensi *array*.
2. Menentukan tipe data *array*.
3. Menentukan panjang *array*.
4. Melakukan pengurutan *array*.

## **C. Sikap Kerja**

1. Harus bersikap analitik dalam menentukan dimensi *array*, tipe data *array* dan panjang *array*.
2. Bekerjasama dan komunikatif dalam melakukan proses pengurutan *array*.

## BAB VI

### MEMBUAT PROGRAM UNTUK AKSES *FILE*

#### A. Pengetahuan yang Diperlukan dalam Membuat Program Untuk Akses File

1. Program untuk menulis data dalam media penyimpanan

Untuk menulis data ke dalam media penyimpanan atau suatu file, *header* yang perlu ditambahkan adalah *fstream.h*. Setelah menambahkan header tersebut kita bisa menggunakan perintah untuk membuka / membuat *file* dan menuliskan data kedalam file tersebut.

```
char nama_file[50];
char isi_file[100];

cout<<"Masukan Nama File = "; gets(nama_file);
cout<<endl<<"Menulis kedalam File "<<nama_file<<endl;
cout<<"-----"<<endl;
cout<<"Masukan isi file = "; gets(isi_file);

/*
    class ofstream digunakan untuk menuliskan karakter kedalam file
    deklarasi : ofstream nama_objek;
    buka file : nama_objek.open("nama_file");
    ios::app : digunakan untuk menambahkan data pada file dan jika
               file yang dimaksud tidak ada maka akan dibuat secara otomatis

    proses deklarasi object dan buka file dapat disederhanakan menjadi :
    ofstream nama_objek("nama_file");
*/

// penggunaan ios::app adalah optional
ofstream tulis_file(nama_file, ios::app);

tulis_file<<isi_file; // proses menulis ke dalam file

tulis_file.close(); // menutup file

cout<<"-----"<<endl;
cout<<"File "<<nama_file<<" telah di tulis"<<endl;

getch();

Masukan Nama File = Akses File.txt
Menulis kedalam File Akses File.txt
-----
Masukan isi file = Saat ini saya sedang belajar cara menulis ke dalam file.
-----
File Akses File.txt telah di tulis
```

Pada *script* diatas penggunaan perintah *ios::app* bertujuan agar jika *file* yang akan dibuat sudah ada maka data yang dimasukan akan ditambahkan kedalam *file* tersebut.

2. Program untuk membaca data dari media penyimpanan

Setelah membuat *file* dan menuliskan data kedalamnya, maka proses selanjutnya adalah membaca isi *file* tersebut. Isi *file* yang dibaca dapat ditampilkan ke layar *output* dengan ditampung kedalam variabel terlebih dahulu. Berikut ini adalah contoh *script* untuk membaca *file* yang telah dibuat sebelumnya dan hasil data yang dibaca ditampilkan ke layar *output*.

```
char nama_file[50];
char tampung[100];

cout<<"Masukan Nama File = "; gets(nama_file);
cout<<endl<<"Membaca dari File "<<nama_file<<endl;
cout<<"-----"<<endl;

// class ifstream digunakan untuk membaca data dari file
ifstream baca_file(nama_file);

// fungsi eof() untuk mendeteksi akhir dari file
while(!baca_file.eof())
{
    // fungsi getline() digunakan untuk membaca data hingga baru baru
    baca_file.getline(tampung, 100);
    cout<<tampung<<endl; // menampilkan data ke layar
}

baca_file.close(); // menutup file

cout<<"-----"<<endl;
cout<<"File "<<nama_file<<" selesai di baca"<<endl;

getch();
```

```
Masukan Nama File = Akses File.txt

Membaca dari File Akses File.txt
-----
Saat ini saya sedang belajar cara menulis ke dalam file.
-----
File Akses File.txt selesai di baca
```

Dapat dilihat pada *script* diatas proses membaca baca dimulai dengan membuka *file* terlebih dahulu dan terdapat kondisi perulangan selama data yang ada di dalam *file* belum mencapai akhir maka data akan di tumpung ke variable dan ditampilkan ke layar. Setelah kondisi perulangan berakhir / mencapai akhir dari *file* maka file akan ditutup. Hasil dari *file* yang ditampilkan kelayar dapat dilihat pada *output* diatas.

**B. Keterampilan yang Diperlukan dalam Membuat Program Untuk Akses *File***

1. Membuat program untuk menulis data dalam media penyimpanan.
2. Membuat program untuk membaca data dari media penyimpanan.

**C. Sikap Kerja**

1. Dapat bekerjasama dan komunikatif dalam membuat program untuk menulis data dalam media penyimpanan.
2. Bersikap analitik dalam membuat program untuk membaca data dari media penyimpanan.

## BAB VII

### MENGKOMPILASI PROGRAM

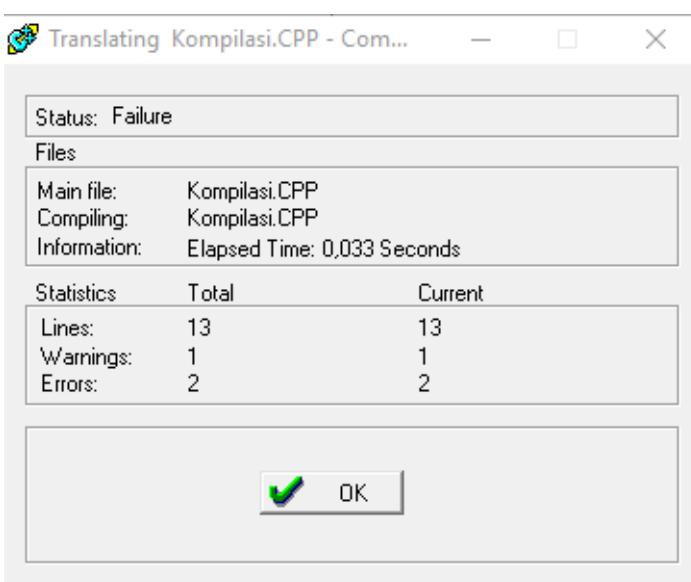
#### A. Pengetahuan yang Diperlukan dalam Mengkompilasi Program

##### 1. Pengkoreksian kesalahan program

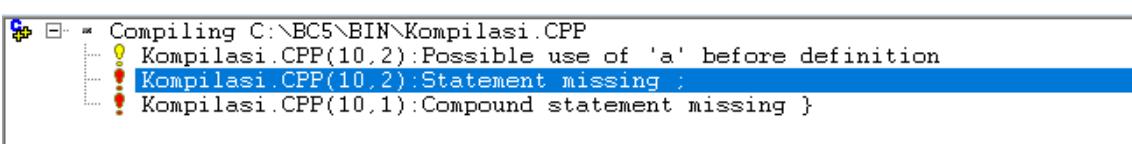
Proses kompilasi merupakan suatu proses menerjemahkan program dari bahasa manusia ke dalam bahasa yang dimengerti oleh komputer, yaitu bahasa mesin. Tombol untuk proses kompilasi dapat ditemukan di *menubar* atas pada IDE Borland C++ seperti yang ditunjukkan pada gambar dibawah.



Hasil kompilasi akan menampilkan kotak dialog *Compile*, dan dari dialog tersebut kita bisa tahu apabila terdapat kesalahan pada *syntax*.



Seperti terlihat pada *dialog* diatas, hasil kompilasi menunjukkan dari 13 baris *syntax* program terdapat 1 *warnings* dan 2 *errors*. Untuk melihat lebih *detail errors* dan *warning* yang dihasilkan dapat dilihat pada *dialog message*.



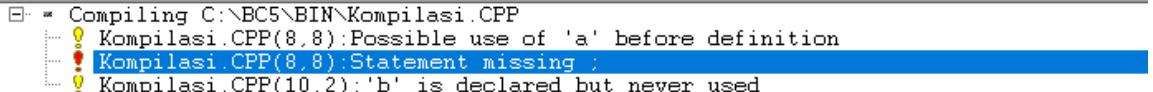
Dari *dialog message* kita bisa mengetahui keterangan mengenai *errors* dan *warnings* apa yang terjadi pada hasil kompilasi beserta posisi *line* tepatnya *errors* dan *warnings* tersebut terjadi. Untuk lebih mudah mengetahui letak *syntax* yang

*error* hanya perlu *double klik* pada salah satu *message* tersebut maka posisi *cursor* akan langsung menuju pada bagian yang terdapat *errors* atau *warning*.

## 2. Pembebasan program dari kesalahan *syntax*

Setelah mengetahui cara mengkompilasi dan mengoreksi kesalahan pada *syntax*, proses selanjutnya adalah membebaskan program yang sudah dibuat dari kesalahan – kesalahan *syntax*. Dari *message* yang ditampilkan kita bisa melihat hasil dari kompilasi program dan jika terdapat *error message* maka bagian *syntax* yang *error* perlu direvisi.

Pada contoh gambar dibawah ini adalah kesalahan yang paling umum terjadi, yaitu tidak ada tanda *semicolon* (;) pada akhir baris *syntax* program. Kesalahan ini akan mengakibatkan *syntax* program yang ada dibawahnya menjadi tidak terbaca pada saat proses kompilasi. Itulah sebabnya apabila ada *error* seperti ini maka *line* berikutnya yang akan menjadi fokus apabila kita melakukan *double klik* pada *message* tersebut.



```
Compiling C:\BC5\BIN\Kompilasi.CPP
Kompilasi.CPP(8,8):Possible use of 'a' before definition
Kompilasi.CPP(8,8):Statement missing ;
Kompilasi.CPP(10,2):'b' is declared but never used

void main()
{
    int a, b;

    cout<<"Nilai a = "<<a
    cout<<"Nilai b = "<<b;

}
```

Selain kesalahan tersebut banyak kesalahan – kesalahan umum yang biasa terjadi seperti *Undefined symbol* yang terjadi karena penggunaan variabel tanpa deklarasi sebelumnya, atau *call to undefined function* karena menggunakan *function* tanpa *include header* yang di perlukan.

*Warning message* tidak akan mengganggu jalannya program, namun *output* yang dihasilkan mungkin tidak sesuai dengan yang diharapkan. Seperti pada gambar diatas *warning "possible se of 'a' before definition"* dikarenakan variabel yang digunakan belum di definisikan atau diberi nilai awal, sehingga nilai *random* yang akan diberikan pada variabel tersebut yang mungkin akan mengakibatkan hasil *output* yang salah.

Dalam pembebasan program dari kesalahan *syntax* yang diperlukan adalah ketelitian dan kejelian *programmer* untuk menerjemahkan maksud dari *error message* yang dihasilkan proses kompilasi, serta tindakan yang tepat untuk menangani kesalahan tersebut.

### **B. Keterampilan yang Diperlukan dalam Mengkompilasi Program**

1. Mengoreksi kesalahan program.
2. Membebaskan program dari kesalahan syntax.

### **C. Sikap Kerja**

1. Analitik dalam mengoreksi kesalahan program.
2. Dapat bekerjasama dan komunikasi dalam membebaskan program dari kesalahan *syntax*.



**BUKU INFORMASI**

**MENGGUNAKAN *LIBRARY* ATAU  
KOMPONEN *PRE-EXISTING***

**J.620100.019.002**

KEMENTERIAN KETENAGAKERJAAN R.I.  
**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS**  
**DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**  
Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

## DAFTAR ISI

DAFTAR ISI -----	2
BAB I PENDAHULUAN -----	4
A. Tujuan Umum -----	4
B. Tujuan Khusus -----	4
BAB II MELAKUKAN PEMILIHAN UNIT-UNIT <i>REUSE</i> YANG POTENSIAL -----	5
A. Pengetahuan yang Diperlukan dalam Melakukan Pemilihan Unit-unit <i>Reuse</i> yang Potensial -----	5
1. Cara Mengidentifikasi <i>Class</i> Unit-unit <i>Reuse</i> yang Sesuai -----	6
2. Cara Menghitung Keuntungan Efisiensi dari Pemanfaatan Komponen <i>Reuse</i> -----	6
3. Cara Memastikan Lisensi, Hak Cipta dan Hak Paten Tidak Dilanggar dalam Pemanfaatan Komponen <i>Reuse</i> Tersebut -----	7
B. Keterampilan yang Diperlukan dalam Melakukan Pemilihan Unit-unit <i>Reuse</i> yang Potensial -----	9
C. Sikap Kerja dalam Melakukan Pemilihan Unit-unit <i>Reuse</i> yang Potensial -----	9
BAB III MELAKUKAN INTEGRASI <i>LIBRARY</i> ATAU KOMPONEN <i>PRE-EXISTING</i> DENGAN <i>SOURCE CODE</i> YANG ADA-----	10
A. Pengetahuan yang Diperlukan dalam Melakukan Integrasi <i>Library</i> atau Komponen <i>Pre-existing</i> dengan <i>Source Code</i> yang Ada -----	10
1. Cara Mengidentifikasi Ketergantungan Antar Unit -----	10
2. Cara Menghindari Penggunaan Komponen yang Sudah <i>Obsolete</i> --	11
3. Cara Menerapkan Program yang Dihubungkan dengan <i>Library</i> -----	11
B. Keterampilan yang Diperlukan dalam Melakukan Integrasi <i>Library</i> atau Komponen <i>Pre-existing</i> dengan <i>Source Code</i> yang Ada -----	13
C. Sikap Kerja yang Diperlukan dalam Melakukan Integrasi <i>Library</i> atau Komponen <i>Pre-existing</i> dengan <i>Source Code</i> yang Ada -----	13
BAB IV MELAKUKAN PEMBAHARUAN <i>LIBRARY</i> ATAU KOMPONEN <i>PREEEXISTING</i> YANG DIGUNAKAN -----	14

Modul Pelatihan Berbasis Kompetensi Bidang <i>Software Development</i> Subbidang Pemrograman	Kode Modul J.620100.019.002
A. Pengetahuan yang Diperlukan dalam Melakukan Pembaharuan <i>Library</i> atau Komponen <i>Preexisting</i> yang Digunakan ----- 14	
1. Cara Mengidentifikasi Cara-cara Pembaharuan <i>Library</i> atau Komponen <i>Pre-existing</i> ----- 14	
2. Cara Memberlakukan Pembaharuan <i>Library</i> atau Komponen <i>Preexisting</i> dengan Berhasil ----- 21	
B. Keterampilan yang Diperlukan dalam Melakukan Pembaharuan <i>Library</i> atau Komponen <i>Preexisting</i> yang Digunakan----- 21	
C. Sikap Kerja yang Diperlukan dalam Melakukan Pembaharuan <i>Library</i> atau Komponen <i>Preexisting</i> yang Digunakan----- 21	
DAFTAR PUSTAKA ----- 22	
A. Dasar Perundang-undangan ----- 22	
B. Buku Referensi ----- 22	
C. Referensi Lainnya ----- 22	
DAFTAR PERALATAN/MESIN DAN BAHAN ----- 23	
A. Daftar Peralatan/Mesin----- 23	
B. Daftar Bahan----- 23	
DAFTAR PENYUSUN ----- 24	

## **BAB I**

### **PENDAHULUAN**

#### **A. Tujuan Umum**

Setelah mempelajari modul ini peserta latih diharapkan mampu Menggunakan *Library* atau Komponen *Pre-Existing*.

#### **B. Tujuan Khusus**

Adapun tujuan mempelajari unit kompetensi melalui buku informasi Menggunakan *Library* atau Komponen *Pre-Existing* ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut:

1. Melakukan pemilihan unit-unit *reuse* yang potensial yang meliputi kegiatan mengidentifikasi *class* unit-unit *reuse* yang sesuai, menghitung keuntungan efisiensi dari pemanfaatan komponen *reuse*, serta memastikan lisensi, hak cipta dan hak paten tidak dilanggar dalam pemanfaatan komponen *reuse* tersebut;
2. Melakukan integrasi *library* atau komponen *pre-existing* dengan *source code* yang ada yang meliputi kegiatan mengidentifikasi ketergantungan antar unit, menghindari penggunaan komponen yang sudah *obsolete*, serta menerapkan program yang dihubungkan dengan *library*;
3. Melakukan pembaharuan *library* atau komponen *preexisting* yang digunakan yang meliputi kegiatan mengidentifikasi cara-cara pembaharuan *library* atau komponen *pre-existing* dan memberlakukan pembaharuan *library* atau komponen *preexisting* dengan berhasil.

## BAB II

### MELAKUKAN PEMILIHAN UNIT-UNIT *REUSE* YANG POTENSIAL

#### A. Pengetahuan yang Diperlukan dalam Melakukan Pemilihan Unit-unit *Reuse* yang Potensial

##### 1. Cara Mengidentifikasi Class Unit-unit *Reuse* yang Sesuai

Dalam pemrograman, *reusable code* adalah penggunaan kembali kode/*script* serupa dari berbagai fungsi. Bukan berarti menyalin dan menuliskan kode yang sama dari 1 blok *script* ke blok lainnya, melainkan metode penggunaan *script* yang sama, tanpa harus menulis ulang *script* tersebut pada tiap blok *script*. Terdapat banyak cara untuk menerapkan konsep *reusable* pada suatu aplikasi. Cara ini mengikuti ketentuan umum dalam pemrograman, "Setiap bagian dari pengetahuan harus memiliki representasi tunggal dan tidak ambigu". Konsep ini akan membantu *developer* untuk menyusun struktur dari aplikasi agar tidak berantakan ketika melakukan *debugging* aplikasi.

Misalkan dalam suatu aplikasi terdapat 3 *form*, dan setiap *form* terdapat *input* yang mengharuskan *user* untuk memasukan karakter angka (contoh: telepon, umur, nilai, dll). Berikut ini adalah *script* untuk validasi karakter angka dalam PHP:

```
26 $number = "0899337766asd";
27 if (is_numeric($number)) {//validasi karakter angka
28     echo "karakter berupa angka";
29 } else {//jika ditemukan karakter selain angka
30     echo "terdapat karakter bukan angka";
31 }
```

Tanpa konsep *reusable* maka *script* pada gambar di atas akan diketik sebanyak 3 kali pada masing-masing *form*.

Dengan konsep *reusable* maka *script* dibuat dalam bentuk fungsi, dan dipanggil sebanyak 3 kali pada setiap form yang membutuhkan validasi. Berikut contoh *script* dalam PHP pada gambar berikut :

```
25 function validasi_angka($number){  
26     if (is_numeric($number)) {//validasi karakter angka  
27         return "karakter berupa angka";  
28     } else {//jika ditemukan karakter selain angka  
29         return "terdapat karakter bukan angka";  
30     }  
31 }  
32  
33 //memanggil fungsi |  
34 validasi_angka("0899337766asd");  
35 //output "terdapat karakter bukan angka"
```

Dengan begitu pada setiap *form* yang membutuhkan validasi angka, fungsi kita hanya perlu mengetikkan “validasi\_angka(\$number)” pada ketiga *form*.

## 2. Cara Menghitung Keuntungan Efisiensi dari Pemanfaatan Komponen *Reuse*

Manfaat dengan penggunaan kembali suatu *script code/reuseable code* sangat jelas dapat terlihat. Untuk setiap aplikasi yang kita rancang, kita dapat menggunakannya kembali. Hal ini akan sangat menghemat waktu, meningkatkan fitur dalam program kita, dan umumnya membuat pemrograman yang lebih hemat biaya. Seiring pertumbuhan, kumpulan *reuseable code* atau disebut juga *library* pastinya akan semakin dikembangkan menjadi lebih mudah, dan mudah untuk meletakkan kerangka kerja untuk aplikasi yang lebih canggih, tanpa memakan waktu berlebih. Sebagian besar produk pihak ketiga menggunakan filosofi ini. Berikut ini adalah keuntungan yang didapatkan dengan menggunakan konsep *reuseable code*:

- Karena *script* dan fungsi yang ada dapat digunakan kembali maka waktu yang dibutuhkan untuk mengembangkan aplikasi baru menggunakan metode *reuse* akan lebih cepat.
- Dengan menggunakan metode *reuseable/penggunaan kembali*, biaya aplikasi akan lebih murah. Karena kita juga secara otomatis *programmer* dapat mengerjakan hal lainnya (menulis *script code* lainnya).
- Tidak perlu menulis *script* code yang sudah ada berulang-ulang.
- Penggunaan kembali juga mengurangi waktu pengembangan proyek.

- Memungkinkan fungsionalitas silang karyawan dari satu proyek ke proyek lain.
- Dengan konsep penggunaan kembali juga akan memungkinkan standarisasi dari *script code* perangkat lunak yang dikembangkan (jadi dalam 1 fungsi dengan kegunaan yang sama, menggunakan standar *code* yang sama).

### 3. Cara Memastikan Lisensi, Hak Cipta dan Hak Paten Tidak Dilanggar dalam Pemanfaatan Komponen *Reuse* Tersebut

Banyak programmer yang memanfaatkan *library* pihak ketiga untuk mengembangkan suatu aplikasi, dan *library* yang digunakan paling banyak yang bersifat *open source*. Selain karena gratis, dengan menggunakan *library* maka akan memangkas biaya serta waktu dalam pengembangan aplikasi. Akan tetapi penyalahgunaan *library open source* tanpa mengetahuinya aturan jelas mengenai hak cipta atau lisensi dari *library* tersebut bisa menghambat pengembangan aplikasi. Berikut ini adalah beberapa aturan agar kita sebagai pengembang aplikasi tidak melanggar lisensi, hak paten ataupun hak cipta dari *library open source* yang digunakan:

- **Jangan gunakan *library open source* yang tidak memiliki persyaratan lisensi.** Beberapa *library open source* di internet tidak menampilkan pemberitahuan lisensi, akan tetapi itu tidak berarti bahwa *library* dapat digunakan secara bebas. Orang yang mempostingnya mungkin tidak mematuhi persyaratan lisensi atau pembuat *library* mungkin belum menerapkan lisensi *open source* pada *script* yang telah dibuat. Kita lebih baik menghindari menggunakan *library* tanpa lisensi, atau tanyakan langsung pada pengembang *library* untuk menerapkan/memberitahukan lisensi *library* yang dia buat.
- **Jangan melanggar lisensi *open source*.** Penggunaan perangkat lunak *open source* mungkin sulit untuk dilacak oleh pemilik perangkat lunak, tetapi itu tidak berarti penggunaan dan ketidakpatuhan tidak diperhatikan. Melanggar lisensi *open source* dapat menyebabkan aplikasi yang kita

kembangkan menjadi bermasalah dalam mata hukum dan memalukan publik, dan bahkan dapat membahayakan investasi atau akuisisi. Hal ini juga dapat menyebabkan calon pelanggan menolak untuk membeli produk yang kita kembangkan karena takut akan tanggung jawab hilir. Pengembang telah melakukan upaya besar untuk membuat perangkat lunak mereka *open source* termasuk biaya lisensi sebelumnya. Penyalahgunaan perangkat lunak tidak adil bagi para pengembang tersebut dan merusak inovasi yang mereka harapkan dapat difasilitasi.

- **Tentukan apa yang aplikasi kita butuhkan.** Suatu saat nanti kita harus menyediakan sebuah list yang berisi daftar library yang kita gunakan. Calon investor dan pengakuisisi akan meminta daftar itu, dan mempertahankan daftar terbaru akan menghemat waktu dan usaha kita ketika permintaan itu datang. Sebagian besar unduhan perangkat lunak *open source* menyertakan file "license.txt" atau "copying.txt". Simpan salinan lisensi itu dan catat perangkat lunak apa saja yang dicakupnya.
- **Memahami lisensi permisif dan lisensi copyleft.** Lisensi *open source* terbagi menjadi dua jenis: permisif (BSD, MIT, dan Apache) dan lisensi copyleft (GPL, LGPL, Eclipse Public Licence, Mozilla Public Licence, dan Common Development and Distribution License). Sebagian besar perusahaan dan pengguna layanan mereka tidak memiliki masalah hukum mengenai penggunaan perangkat lunak di bawah lisensi permisif. Namun demikian, mematuhi lisensi *copyleft* harus lebih hati-hati, dan mungkin tidak konsisten dengan rencana tertentu untuk menjaga perangkat lunak tetap dapat digunakan.
- **Memenuhi persyaratan yang lisensi.** Baik permisif atau *copyleft*, semua lisensi *open source* memiliki persyaratan/*terms*. Berarti kita harus menyertakan salinan lisensi yang berlaku saat mendistribusikan perangkat lunak *open source*. Biasanya tidak cukup hanya dengan menyertakan tautan atau bentuk lisensi yang singkat. Penting untuk mengembangkan strategi pengiriman pemberitahuan lisensi yang sesuai dengan sebagian besar lisensi *open source* tanpa membingungkan.

- **Memahami lisensi *open source* mana yang berfungsi dengan perangkat lunak terdistribusi.** Kebanyakan lisensi *open source* (selain Affero GPL) tidak memiliki ketentuan untuk perangkat lunak *as-a-service* (SaaS). Untuk elemen terdistribusi dari SaaS dan sistem cloud (seperti JavaScript) atau perangkat lunak terdistribusi (termasuk aplikasi *mobile* dan pengujian beta), Anda dapat menggunakan perangkat lunak di bawah lisensi permisif, tetapi kita harus sangat berhati-hati sebelum menggunakan perangkat lunak di bawah lisensi *copyleft*. Gunakan perangkat lunak GPL hanya jika dijalankan 100% dalam prosesnya sendiri tanpa kode tertaut. Gunakan perangkat lunak LGPL hanya sebagai pustaka yang terhubung secara dinamis. Dan gunakan perangkat lunak *copyleft* lainnya hanya jika kita belum memodifikasi API. Distribusi sesuai dengan aturan pasar aplikasi *mobile* mungkin tidak kompatibel dengan kepatuhan dengan lisensi *copyleft* tertentu (seperti GPL atau LGPL).

## B. Keterampilan yang Diperlukan dalam Melakukan Pemilihan Unit-unit *Reuse* yang Potensial

1. Mengidentifikasi *class* unit-unit *reuse* yang sesuai
2. Menghitung keuntungan efisiensi dari pemanfaatan komponen *reuse*
3. Memastikan lisensi, hak cipta dan hak paten tidak dilanggar dalam pemanfaatan komponen *reuse* tersebut

## C. Sikap kerja

Harus bersikap secara:

1. Analitis dan teliti dalam mengidentifikasi *class* unit-unit *reuse* yang sesuai
2. Analitis dan teliti dalam menghitung keuntungan efisiensi dari pemanfaatan komponen *reuse*
3. Analitis dan teliti dalam memastikan lisensi, hak cipta dan hak paten tidak dilanggar dalam pemanfaatan komponen *reuse* tersebut

## **BAB III**

### **MELAKUKAN INTEGRASI *LIBRARY* ATAU KOMPONEN *PRE-EXISTING* DENGAN *SOURCE CODE* YANG ADA**

#### **A. Pengetahuan yang Diperlukan dalam Melakukan Integrasi *Library* atau Komponen *Pre-existing* dengan *Source Code* yang Ada**

##### **1. Cara Mengidentifikasi Ketergantungan Antar Unit**

Pilihan perangkat lunak dependen yang buruk dapat menimbulkan konsekuensi yang mahal. Masalah kompatibilitas dan kesalahan *runtime* yang menjadikan perangkat lunak yang dikembangkan menjadi tidak berguna. Untuk perangkat lunak yang bergantung pada sejumlah paket, menyelesaikan masalah ini bisa sangat memakan waktu dan membuat stres, terutama jika penyebaran dibiarkan hingga menit terakhir batas waktu proyek.

Masalah dapat terjadi jika perangkat lunak yang dibangun bergantung pada versi khusus perangkat lunak dependen, dan perangkat lunak dependen tidak tersedia pada aplikasi kita. Jika perangkat lunak yang kita kembangkan *in-house* diambil dan digunakan oleh pengguna, pasti akan digunakan pada pilihan yang lebih luas dari sistem operasi dan lingkungan daripada yang kita gunakan untuk menguji perangkat lunak kita.

Merupakan hal yang baik untuk mengembangkan *code* secara terjaga dengan menanamkan pemikiran bahwa sebuah *software* dibuat akan selalu berubah sesuai dengan keputusan implementasi, sehingga *source code* yang dibuat akan lebih global/mudah diubah sesuai dengan pengembangan sistem kedepannya. Berikut ini adalah beberapa hal yang perlu diingat dalam mengidentifikasi ketergantungan antara unit-unit pada sebuah *library* atau *software*:

- Hindari penggunaan *source code* / *library* yang sudah *deprecated/obsolete* (usang).
- Cobalah untuk menjaga pengembangan dan dependensi lingkungan penggunaan yang dimaksudkan sedekat mungkin.
- Hindari ketergantungan versi perangkat lunak tertentu.

- Siapkan infrastruktur pengujian yang memungkinkan pengujian di bawah berbagai lingkungan yang mewakili pengguna target Anda.
- Uji perangkat lunak Anda pada *platform* target secara teratur jika memungkinkan. Jika perangkat lunak ini akan digunakan pada infrastruktur, lihat apakah Anda dapat memperoleh akses untuk melakukan pengujian.

## 2. Cara Menghindari Penggunaan Komponen yang Sudah *Obsolete*

Jika Anda aktif dalam mengembangkan perangkat lunak, maka Anda mungkin pernah mendengar istilah "*deprecated*" ketika membaca dokumentasi dari perangkat lunak yang ingin digunakan. Ketika berbicara mengenai fungsi atau method pada suatu *library*, fitur perangkat lunak, atau praktik perangkat lunak tertentu, berarti *library* tersebut memiliki versi terbaru atau dengan kata lain, versi *library* yang digunakan sudah "usang" / ada alternatif yang lebih baik dari *library* yang ingin digunakan. Biasanya dari dokumentasi *library* yang ingin digunakan, pengembang *library* pasti akan mencantumkan *pemberitahuan* mengenai versi terbaru dari *library*. Sebelum anda menggunakan suatu *library*, pastikan *library* yang anda gunakan adalah yang terbaru, cari dokumentasi dari *library* sebelum menggunakannya.

## 3. Cara Menerapkan Program yang Dihubungkan dengan *Library*

Untuk kedepannya, Penulis akan memberikan contoh dalam menerapkan penggunaan *library* dengan menggunakan bahasa pemrograman PHP.

Untuk menggunakan *source code* eksternal / *library* pada PHP, kita dapat menggunakan perintah "include". Perhatikan instruksi berikut ini:

- Buatlah sebuah file "function.php" pada *localhost* anda, seperti berikut ini:

*Script pada file function.php*

```
<?php  
  
function test(){  
    echo "hello world";  
}
```

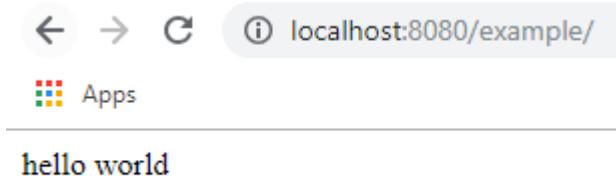
- Buat 1 lagi file "index.php", lalu ketikan *script* berikut ini.

*Script pada file index.php*

```
<?php  
  
include "function.php";  
  
test();
```

- Buat 1 lagi file "index.php", lalu ketikan *script* berikut ini.
- Sekarang jalankan script index.php yang dibuat pada browser.

Hasil dari *script* yang dibuat



*File "function.php"* diasumsikan sebagai sebuah *library* yang kita *download* dari situs *open source*, yang menampung sebuah fungsi untuk menampilkan tulisan "hello world". *File index.php* yang kita buat adalah program yang kita sedang kembangkan, dan membutuhkan *library* "function.php" untuk menampilkan tulisan "hello world". Untuk itu lah diketikan perintah "include", untuk mendeklarasikan penggunaan komponen yang terdapat pada "function.php", agar bisa digunakan pada "index.php". Sehingga dengan memanggil fungsi "test()" pada *file* index.php, maka *browser* menampilkan tulisan "hello world".

Untuk mengetahui, perintah-perintah apa yang dapat digunakan dari sebuah *library* (seperti pada contoh sebelumnya, dari mana kita mengetahui bahwa fungsi “`test()`” berguna untuk menampilkan tulisan “hello world”? ), biasanya pengembang *library* akan memberikan dokumentasi mengenai fitur-fitur apa saja yang disediakan dari *library* yang dibuatnya, dan juga cara penginstalasiannya.

**B. Keterampilan yang Diperlukan dalam Melakukan Integrasi *Library* atau Komponen *Pre-existing* dengan *Source Code* yang Ada**

1. Mengidentifikasi ketergantungan antar unit
2. Menghindari penggunaan komponen yang sudah *obsolete*
3. Menerapkan program yang dihubungkan dengan *library*

**C. Sikap kerja**

Harus bersikap secara:

1. Analitis dan teliti dalam mengidentifikasi ketergantungan antar unit
2. Analitis dan teliti dalam menghindari penggunaan komponen yang sudah *obsolete*
3. Analitis dan teliti dalam menerapkan program yang dihubungkan dengan *library*

## **BAB IV**

### **MELAKUKAN PEMBAHARUAN *LIBRARY* ATAU KOMPONEN *PREEXISTING* YANG DIGUNAKAN**

#### A. Pengetahuan yang Diperlukan dalam Melakukan Pembaharuan *Library* atau Komponen *Preexisting* yang Digunakan

#### 1. Cara Mengidentifikasi Cara-cara Pembaharuan *Library* atau Komponen *Pre-existing*

Penggunaan *library* dalam PHP dipermudah dengan adanya *composer*. *Composer* adalah sebuah manajer dependensi yang dibuat khusus untuk bahasa pemrograman PHP. *Composer* dapat memecahkan masalah berikut ini:

- Resolusi dependensi untuk paket PHP.
  - Solusi pemecahan untuk paket PHP.
  - Menjaga agar semua paket atau komponen dapat diperbarui.

## 1.1. Instalasi *composer*

Penulis akan menjelaskan cara instalasi *composer* pada sistem operasi *windows*. Sebelumnya, pastikan laptop/PC yang digunakan sudah terinstal aplikasi *web-server*, penulis menggunakan XAMPP.

- *Download composer* dari situs resminya, <https://getcomposer.org/>. Pilih jenis file "Composer-setup.exe".

Klik link yang diwarnai merah

[Home](#) | [Getting Started](#) | [Download](#) | [Documentation](#) | [Browse Packages](#)

[Download Composer](#) Latest: v1.8.0

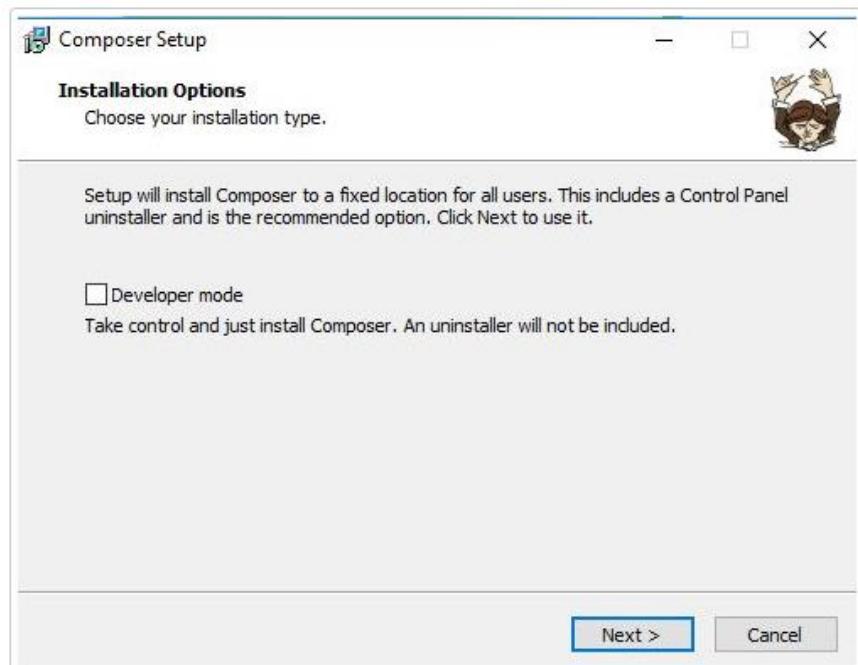
## Windows Installer

The installer will download composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

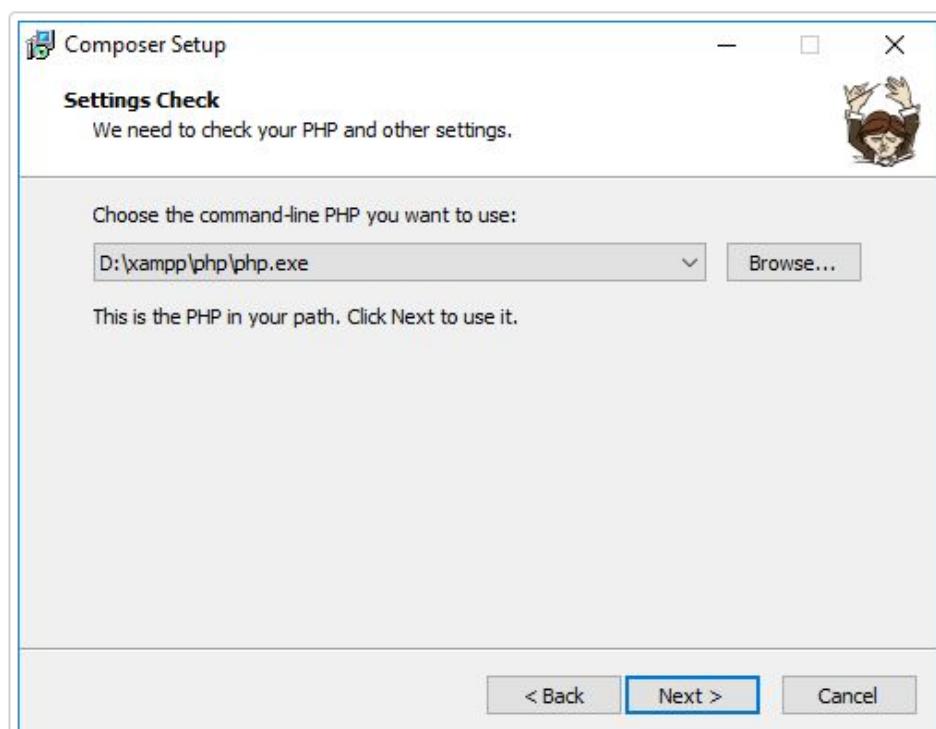
Download and run [Composer-Setup.exe](#) - it will install the latest composer version whenever it is executed.

## Command-line installation

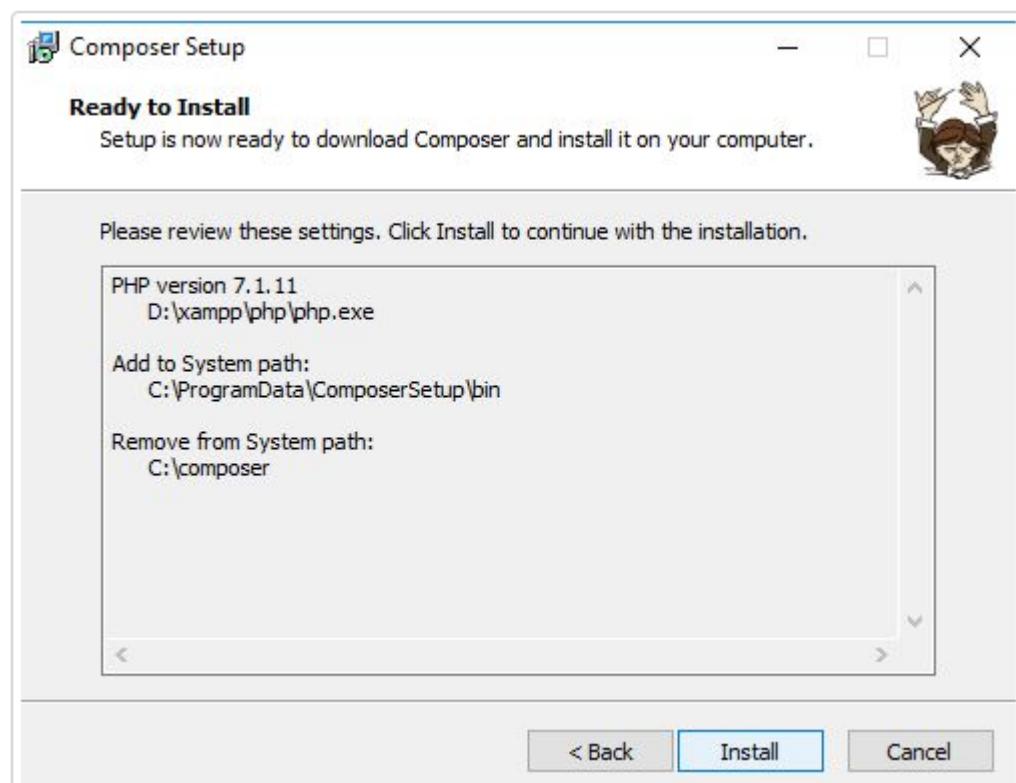
- Klik dua kali pada file "Composer-setup.exe" yang sudah di-download. Nanti akan muncul dialog box seperti gambar berikut.



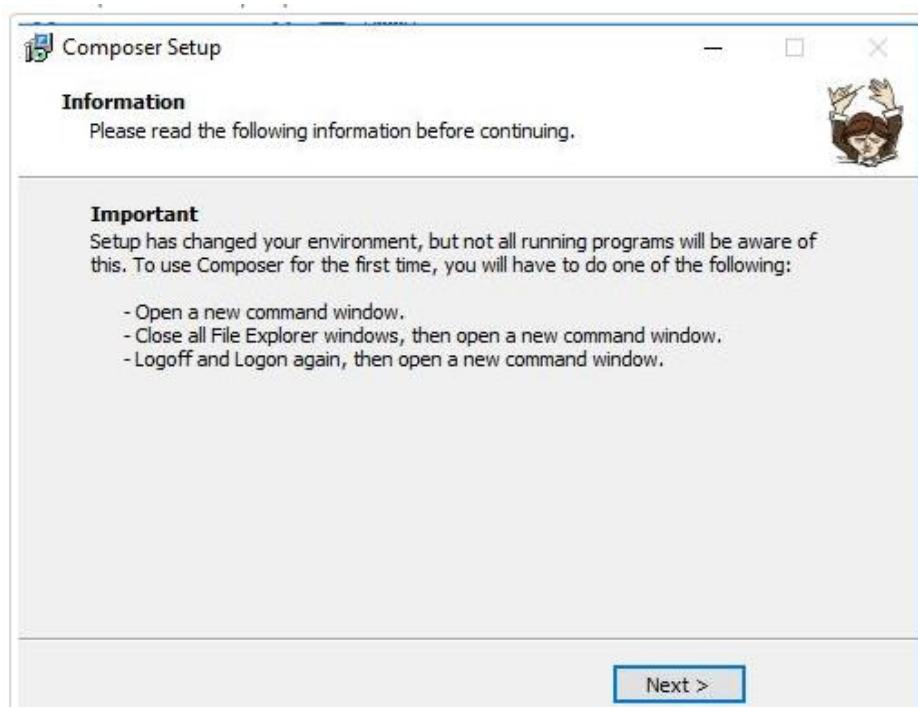
- Klik *Next* untuk melakukan instalasi.
- *Composer* akan meminta lokasi *file PHP* berada. Biasanya ada dalam folder "C:\XAMPP\php\php.exe" atau pada *path* lain, tergantung dimana anda melakukan instalasi XAMPP. Klik *browse* untuk mencari *file PHP* yang sudah diinstal saat menginstall XAMPP.



- Lalu klik *next* kembali. Jika muncul jendela *proxy*, biarkan kosong saja.



- Klik *next* lagi. Sehingga muncul jendela "Composer Ready to install".
- Tunggu beberapa saat sampai proses installasi selesai.



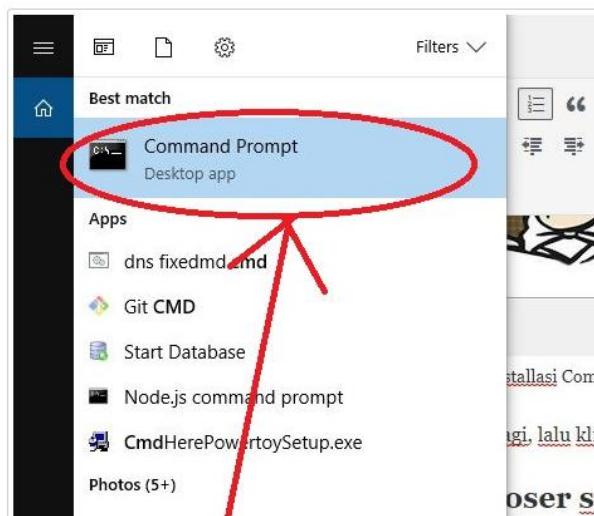
### Instalasi selesai



- Kemudian klik *next* lagi, lalu klik *finish*. Anda telah berhasil menginstal *composer*.

Untuk mengetes apakah composer berhasil diinstall lakukan langkah berikut:

- Pastikan XAMPP sudah diaktifkan (Apache *service*) dengan mengklik tombol *start*.
- Buka *start up* program *windows*, ketik CMD.



- Klik program "*Command Prompt*".

- Lalu ketik *composer*.

Ketik "composer" pada *command prompt*

```
C:\ Command Prompt  
Microsoft Windows [Version 10.0.17134.472]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\User>composer
```

- Maka akan tampil seperti gambar di bawah ini jika instalasi *composer* sebelumnya sudah berhasil.

```
Command Prompt
Microsoft Windows [Version 10.0.17134.472]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>composer

Composer version 1.7.3 2018-11-01 10:05:06

Usage:
  command [options] [arguments]

Options:
  -h, --help          Display this help message
  -q, --quiet         Do not output any message
  -V, --version       Display this application version
  --ansi             Force ANSI output
  --no-ansi          Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
  --profile          Display timing and memory usage information
  --no-plugins        Whether to disable plugins.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  -v|vv|vvv|vvvv --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
  3 for debug
```

## 1.2. Instalasi *library* PHP menggunakan *composer*

Masuk ke situs [packagist.org](https://packagist.org), *repository* khusus yang menyediakan *library* untuk *composer*. Katakanlah kita ingin meng-*install* facebook PHP SDK, kita tinggal melakukan pencarian dengan kata kunci "facebook PHP", dan masuk ke hasil pencarian yang diinginkan.

Cari package "facebook php"

Packagist The PHP Package Repository

Browse Submit Create account Sign in

facebook php

facebook/graph-sdk  
Facebook SDK for PHP

facebook/webdriver  
A PHP client for Selenium WebDriver

facebook/php-sdk-v4  
Facebook SDK for PHP

hybridauth/hybridauth

Contoh libary yang akan di-download

PHP ↓ 6 261 429  
★ 2 438

PHP ↓ 14 697 530  
★ 2 925

PHP ↓ 6 526 785  
★ 2 424

PHP ↓ 1 192 251

Active filters

type: library Clear all

Package type

framework 1  
laravel-package 1  
library 206  
package 1  
phpbb-extension 1  
project 5  
silverstripe-module 1  
symfony-bundle 3  
yii-extension 2

Tags

Setelah Anda masuk ke halaman *library* yang diinginkan, *scroll* ke bagian bawah situs, dan anda akan menemukan penjelasan dari *library* tersebut. Mulai dari cara instalasi *library*, cara penggunaan, prasyarat dalam menggunakan *library*, dll.

### Penjelasan penggunaan dari *library*

The screenshot shows the Packagist website with the URL [https://packagist.org/packages/facebook/graph-sdk](#). The page title is "Facebook PHP SDK". The top navigation bar includes links for "Browse", "Submit", "Create account", and "Sign in". Below the title, there's a "README" section. It features a "build passing" badge, a "Scrutinizer unknown" badge, and a "Latest Stable 5.7.0" badge. A note states: "This repository contains the open source PHP SDK that allows you to access the Facebook Platform from your PHP app." Under the "Installation" section, it says: "The Facebook PHP SDK can be installed with Composer. Run this command:" followed by a code block: "composer require facebook/graph-sdk". A note below it cautions: "Please be aware, that there are issues when using the Facebook SDK together with Guzzle 6.x. php-graph-sdk v5.x only works with Guzzle 5.x out of the box. However, there is a workaround to make it work with Guzzle 6.x." The "Upgrading to v5.x" section notes: "Upgrading from v4.x? Facebook PHP SDK v5.x introduced breaking changes. Please [read the upgrade guide](#) before upgrading." The "Usage" section includes a note: "Note: This version of the Facebook SDK for PHP requires PHP 5.4 or greater." and a code example: "Simple GET example of a user's profile."

Ikuti instruksi berikut ini untuk melakukan instalasi *library*:

- Buat sebuah *folder* pada "c:/XAMPP/htdocs" (atau tergantung dimana Anda melakukan instalasi XAMPP). Nama *folder* bebas, penulis memberikan contoh dengan nama *folder* "composertest".
- Lihat pada situs yang dibuka. Bagian "*Pre-requisites*" adalah persyaratan yang dibutuhkan untuk menggunakan *library* ini. Jika tidak tertera, berarti *library* ini tidak membutuhkan prasyarat tertentu.
- Setelah semua persyaratan sudah dipenuhi. kita lanjut ke bagian *installation*.
- Buka *command prompt*, dan pindahkan *path command prompt* Anda, ke *folder* pada xampp/htdocs yang sebelumnya telah dibuat.

The screenshot shows a Windows Command Prompt window. The title bar says "cmd Select C:\WINDOWS\system32\cmd.exe". The window displays the following text:  
Microsoft Windows [Version 10.0.17134.472]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\User>cd ../../xampp/htdocs/composertest  
  
C:\xampp\htdocs\composertest>

- Ketikan *command* yang tertera pada kotak marah di gambar berikut ini pada *command prompt*, lalu tekan *enter*.

### Installation

The Facebook PHP SDK can be installed with [Composer](#). Run this command:

```
composer require facebook/graph-sdk
```

- Jika nanti muncul pemeberitahuan "*No composer.json in current directory, do you want to use the one at C:\xampp\htdocs? [Y,n]?*", ketikan "n" dan tekan *enter* (pilihan "n" agar file composer.json yang di-*generate* muncul pada *folder* yang anda buat). Maka nanti akan di-*generate* sebuah *file* dengan nama "*compser.json*". *File* ini berguna untuk jika untuk mempermudah *update library* nantinya.

```
C:\WINDOWS\system32\cmd.exe - composer require facebook/graph-sdk
Microsoft Windows [Version 10.0.17134.472]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>cd ../../xampp\htdocs/composertest

C:\xampp\htdocs\composertest>composer require facebook/graph-sdk
No composer.json in current directory, do you want to use the one at C:\xampp\htdocs? [Y,n]? y
```

- Tunggu sampai proses instalasi *library* selesai.

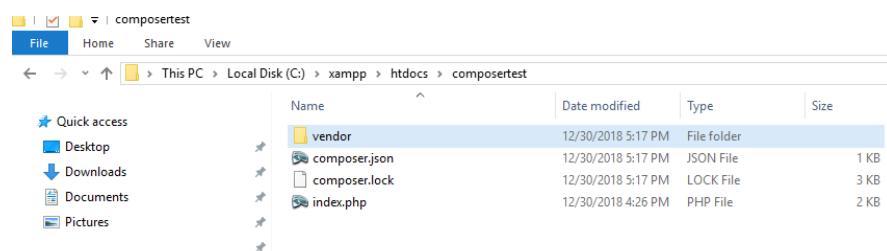
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.472]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>cd ../../xampp\htdocs/composertest

C:\xampp\htdocs\composertest>composer require facebook/graph-sdk
No composer.json in current directory, do you want to use the one at C:\xampp\htdocs? [Y,n]? n
Using version ^5.7 for facebook/graph-sdk
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Installing facebook/graph-sdk (5.7.0): Downloading (100%)
facebook/graph-sdk suggests installing paragonie/random_compat (Provides a better CSPRNG option in PHP 5)
facebook/graph-sdk suggests installing guzzlehttp/guzzle (Allows for implementation of the Guzzle HTTP client)
Writing lock file
Generating autoload files

C:\xampp\htdocs\composertest>
```

- Setelah selesai, maka pada *folder* project anda akan muncul *file* dan *folder* baru seperti gambar berikut ini.

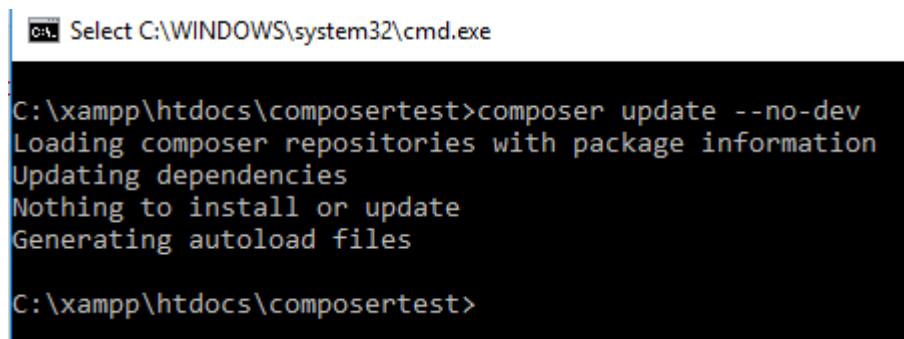


- Dengan begitu proses instalasi *library* berhasil.

## 2. Cara Memberlakukan Pembaharuan *Library* atau Komponen *Pre-existing* dengan Berhasil

Seperti yang penulis jelaskan pada poin sebelumnya, *file composer.json* berguna untuk mempermudah kita ketika melakukan *update library* ke versi yang terbaru. Untuk melakukan *update* menggunakan *composer*, ketikan perintah "composer – no-dev", sehingga semua *library* pada *project* kita akan diperbarui secara otomatis.

### Update *library* pada *project*



```
PS C:\xampp\htdocs\composertest>composer update --no-dev
Loading composer repositories with package information
Updating dependencies
Nothing to install or update
Generating autoload files

C:\xampp\htdocs\composertest>
```

Gambar di atas menampilkan informasi bahwa ketika perintah *update* dilakukan, belum ada pembaharuan dari penyedia *library*, sehingga *library* pada *project* "composertest" sudah pada versi yang paling baru.

## B. Keterampilan yang Diperlukan dalam Melakukan Pembaharuan *Library* atau Komponen *Preexisting* yang Digunakan

1. Mengidentifikasi cara-cara pembaharuan *library* atau komponen *pre-existing*
2. Memberlakukan pembaharuan *library* atau komponen *preexisting* dengan berhasil

## C. Sikap kerja

Harus bersikap secara:

1. Analitis dan teliti dalam mengidentifikasi cara-cara pembaharuan *library* atau komponen *pre-existing*
2. Analitis dan teliti dalam memberlakukan pembaharuan *library* atau komponen *preexisting* dengan berhasil



## **BUKU INFORMASI**

**MELAKUKAN *DEBUGGING*  
J.620100.025.02**

KEMENTERIAN KETENAGAKERJAAN R.I.

**DIREKTORAT JENDERAL PEMBINAAN PELATIHAN DAN PRODUKTIVITAS  
DIREKTORAT BINA STANDARDISASI KOMPETENSI DAN PELATIHAN KERJA**

Jl. Jend. Gatot Subroto Kav. 51 Lt. 6.A Jakarta Selatan  
2018

## **DAFTAR ISI**

DAFTAR ISI -----	2
BAB I PENDAHULUAN -----	4
A. Tujuan Umum -----	4
B. Tujuan Khusus -----	4
BAB II MEMPERSIAPKAN KODE PROGRAM -----	5
A. Pengetahuan yang Diperlukan dalam Mempersiapkan Kode Program --	5
1. Cara Menyiapkan Kode Program Sesuai Spesifikasi -----	5
2. Cara Menyiapkan <i>Debugging Tools</i> untuk Melihat Proses Suatu Modul -----	9
B. Keterampilan yang Diperlukan dalam Mempersiapkan Kode Program ---	10
C. Sikap Kerja dalam Mempersiapkan Kode Program -----	10
BAB III MELAKUKAN <i>DEBUGGING</i> -----	11
A. Pengetahuan yang Diperlukan dalam Melakukan <i>debugging</i> -----	11
1. Cara Melakukan Kompilasi Kode Program Sesuai Bahasa Permrograman yang Sesuai -----	11
2. Cara Menganalisa Kriteria Lulus Build -----	14
3. Cara Menganalisa Kriteria Eksekusi Aplikasi-----	14
4. Cara Menganalisa Kode Kesalahan-----	17
B. Keterampilan yang Diperlukan dalam Melakukan <i>Debugging</i> -----	20
C. Sikap Kerja yang Diperlukan dalam Melakukan <i>Debugging</i> -----	21
BAB IV MEMPERBAIKI PROGRAM-----	22
A. Pengetahuan yang Diperlukan dalam Memperbaiki Program -----	22
1. Cara merumuskan perbaikan terhadap kesalahan kompilasi maupun <i>build</i> -----	22
2. Cara Melakukan perbaikan -----	23
B. Keterampilan yang Diperlukan dalam Memperbaiki Program -----	26
C. Sikap Kerja yang Diperlukan dalam Memperbaiki Program -----	26
DAFTAR PUSTAKA -----	27
A. Dasar Perundang-undangan -----	27
B. Buku Referensi -----	27

Modul Pelatihan Berbasis Kompetensi Bidang <i>Software Development</i> Subbidang Pemrograman	Kode Modul J.620100.025.02
C. Majalah atau Buletin----- 27	
D. Referensi Lainnya ----- 27	
<b>DAFTAR PERALATAN/MESIN DAN BAHAN ----- 28</b>	
A. Daftar Peralatan/Mesin----- 28	
B. Daftar Bahan----- 28	
<b>DAFTAR PENYUSUN ----- 29</b>	

## **BAB I**

### **PENDAHULUAN**

#### **A. Tujuan Umum**

Setelah mempelajari modul ini peserta latih diharapkan mampu melakukan *debugging*.

#### **B. Tujuan Khusus**

Adapun tujuan mempelajari unit kompetensi melalui buku informasi Menyiapkan Informasi dan Laporan Pelatihan ini guna memfasilitasi peserta latih sehingga pada akhir pelatihan diharapkan memiliki kemampuan sebagai berikut:

1. Dapat mempersiapkan kode program yang meliputi menyiapkan kode program sesuai spesifikasi dan menyiapkan *debugging tools* untuk melihat proses suatu modul.
2. Dapat melakukan debugging yang meliputi mengkompilasi kode program sesuai bahasa pemrograman yang digunakan, menganalisa kriteria lulus *build*, menganalisa kriteria eksekusi aplikasi dan menganalisa kode kasalahan.
3. Dapat memperbaiki program yang meliputi merumuskan perbaikan terhadap kesalahan kompilasi maupun *build* dan melakukan perbaikan.

## **BAB II**

### **MEMPERSIAPKAN KODE PROGRAM**

#### **A. Pengetahuan yang Diperlukan dalam Mempersiapkan Kode Program**

##### 1. Cara Menyiapkan Kode Program Sesuai Spesifikasi

Bahasa Pemrograman (*programming language*) adalah sebuah instruksi standar untuk memerintah komputer agar menjalankan fungsi tertentu. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer. Bahasa ini memungkinkan seorang programmer dapat menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan/diteruskan, dan jenis langkah apa secara persis yang akan diambil dalam berbagai situasi.

Fungsi bahasa pemrograman yaitu memerintah komputer untuk mengolah data sesuai dengan alur berpikir yang kita inginkan. Keluaran dari bahasa pemrograman tersebut berupa program/aplikasi. Contohnya adalah program yang digunakan oleh kasir di mal-mal atau swalayan, penggunaan lampu lalu lintas di jalan raya, dll.

Bahasa Pemrograman yang kita kenal ada banyak sekali di belahan dunia, tentang ilmu komputer dan teknologi dewasa ini. Perkembangannya mengikuti tingginya inovasi yang dilakukan dalam dunia teknologi. Contoh bahasa pemrograman yang kita kenal antara lain adalah untuk membuat aplikasi game, antivirus, web, dan teknologi lainnya.

Bahasa pemrograman komputer yang kita kenal antara lain adalah Java, Visual Basic, C++, C, Cobol, PHP, .Net, dan ratusan bahasa lainnya. Namun tentu saja kebutuhan bahasa ini harus disesuaikan dengan fungsi dan perangkat yang menggunakananya.

Secara umum bahasa pemrograman terbagi menjadi 4 kelompok, yaitu :

- a. *Object Oriented Language* (Visual dBase, Visual FoxPro, Delphi, Visual C, Java)

- b. *High Level Language* (seperti Pascal dan Basic)
- c. *Middle Level Language* (seperti bahasa C)
- d. *Low Level Language* (seperti bahasa Assembly)

Proses pembuatan program yaitu kita menulis kode sumber pada teks *editor* misalnya notepad kemudian mengubahnya menjadi bahasa mesin yang bisa dieksekusi oleh CPU. Proses pengubahan kode sumber (*source code*) menjadi bahasa mesin (*machine language*) ini terdiri dari 3 macam yaitu kompilasi, interpretasi dan kompilasi sekaligus interpretasi.

a. Kompilasi

Dalam proses kompilasi semua kode sumber dibaca terlebih dahulu dan jika tidak ada kesalahan dalam menulis program maka akan dibentuk kode mesinnya sehingga program bisa dijalankan. Program yang melakukan tugas ini disebut *Compiler*. Program hasil kompilasi akan berbentuk *executable*. Program bisa langsung dijalankan tanpa harus memiliki *Compiler* di komputer yang menjalankan program tersebut. Bahasa yang menggunakan teknik kompilasi misalnya bahasa C, C++, Pascal, Assembly dan masih banyak lagi.

b. Interpretasi (*Interpretation*)

Bahasa yang menggunakan teknik interpretasi akan membaca kode sumber perbaris dan dieksekusi perbaris. Jika ditemukan kesalahan dalam penulisan program maka di baris kesalahan itulah program akan dihentikan. Program yang melakukan tugas ini disebut Interpreter. Pada teknik interpretasi tidak ada akan dihasilkan program *standalone*, artinya untuk menjalankan program kita harus mempunyai kode sumbernya sekaligus interpreter program tersebut. Bahasa yang menggunakan teknik interpretasi misalnya bahasa Perl, Python, Ruby dan masih banyak lagi.

c. Kompilasi Sekaligus Interpretasi

Ada juga bahasa pemrograman yang menghasilkan programnya dengan teknik kompilasi sekaligus interpretasi. Misalnya bahasa java. Dalam pembuatan program java kode sumber diubah menjadi bytecode. Meskipun tampak seperti bahasa mesin namun ini bukanlah bahasa mesin dan tidak

*executable*. Untuk menjalankan *bytecode* tersebut kita membutuhkan *Java Runtime Environment* (JRE) yang bertugas sebagai interpreter sehingga menghasilkan program dari *bytecode* tersebut.

Setiap bahasa pemrograman memiliki struktur dan *syntax* yang berbeda-beda. Meskipun secara konsep memiliki banyak kesaamaan, para programmer diharapkan memiliki pengetahuan yang mendalam untuk mengimplementasikan konsep tersebut ke dalam bahasa pemrograman yang mereka kuasai. Termasuk dalam menggunakan metode Pemrograman Berorientasi Objek, beberapa bahasa pemrograman sepenuhnya mendukung metode tersebut.

Pada bahasan ini akan menggunakan bahasa pemrograman java, maka hal yang perlu dipersiapkan adalah :

a. *Java Development Kit* (JDK)

Adalah sebuah paket pengembangan yang dibutuhkan untuk membuat dan menjalankan program-program berbasis java. JDK untuk java SE dapat didownload pada situs <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

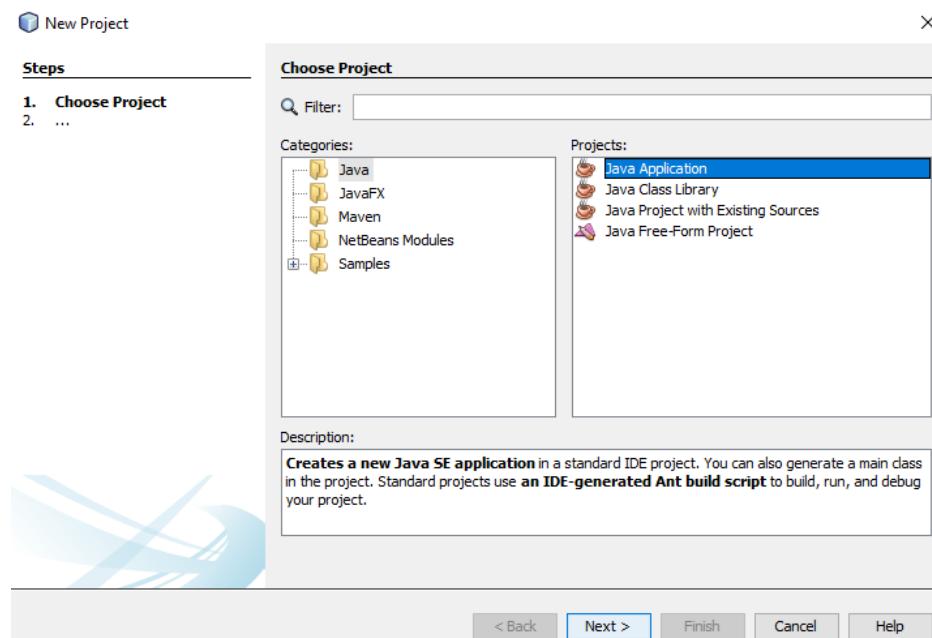
b. Netbeans IDE

NetBeans IDE adalah perangkat lunak *open source* yang di tulis dengan bahasa pemrograman Java. NetBeans IDE digunakan untuk menyesuaikan/menulis kode bahasa pemrograman khususnya Java. Tapi tidak hanya mendukung bahasa pemrograman Java, NetBeans IDE juga mendukung banyak bahasa pemrograman yaitu: HTML/JS, PHP and C/C++ dan lain – lain. Netbeans IDE didownload pada situs <https://netbeans.org/downloads/>

Setelah semua kebutuhan selesai ter-*install*, silahkan ikuti langkah-langkah berikut ini:

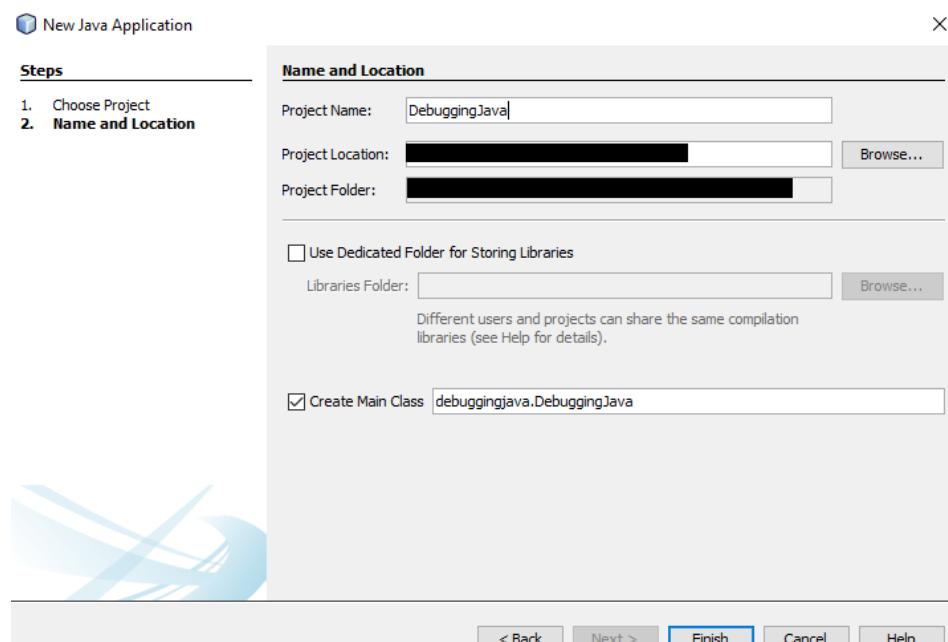
- a. Buka aplikasi Netbeans IDE, kemudian arahkan ke menu File dan pilih *New Project*. Pada jendela yang muncul, pilih Java *Application* kemudian klik *next*.

Gambar 1



- b. Isi data proyek, cukup mengganti project *name* saja (DebuggingJava) dan '*Create Main Class*' tidak perlu dicentang. Setelah itu klik *finish*.

Gambar 2



- c. Setelah proyek sudah terbuat, lalu masukin kode program seperti di bawah ini ke dalam *class* DebuggingJava.java

Gambar 3

```
public static void main(String[] args) {  
    // TODO code application logic here  
    Double lebar,luas, panjang, Luas;  
    Luas = 0.0;  
    Scanner in = new Scanner(System.in);  
  
    System.out.print("Panjang : ");  
    panjang = in.nextDouble();  
  
    System.out.print("Lebar : ");  
    lebar = in.nextDouble();  
  
    luas = panjang * lebar;  
    System.out.print("Luas :" +luas);  
}
```

## 2. Cara Menyiapkan *Debugging Tools* untuk melihat proses suatu modul

*Debugging* adalah tugas yang sangat penting dalam proses pengembangan perangkat lunak, karena program yang salah dapat memiliki konsekuensi yang signifikan bagi penggunanya. Beberapa bahasa yang lebih rentan terhadap beberapa jenis kesalahan karena mereka tidak memerlukan spesifikasi kompiler untuk melakukan pengecekan sebanyak bahasa lainnya. Penggunaan alat analisis statis dapat membantu mendeteksi beberapa kemungkinan masalah. *Debug* sering dilakukan dengan IDE seperti Visual Studio, NetBeans, dan Eclipse. *Standalone debugger* seperti gdb juga digunakan dan ini sering kurang dalam menyediakan lingkungan visual, biasanya menggunakan baris perintah.

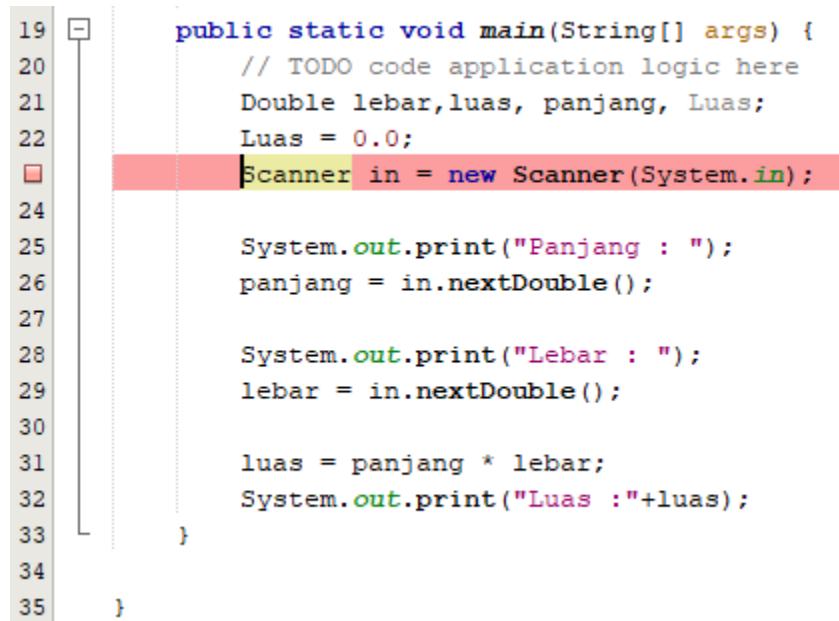
Secara umum, *debug* dapat dikatakan sebagai proses eksekusi program per baris *code*. Karena dieksekusi per baris *code* maka dapat dilihat perkembangan yang terjadi pada aplikasi setiap kali suatu baris (perintah) dijalankan. *User* dapat memperhatikan objek yang muncul, objek yang terhapus, hingga perubahan nilai dari suatu objek. Dalam bahasan ini akan digunakan fitur *debug* dari aplikasi Netbeans IDE yang sudah ter-*install* sebelumnya.

Berdasarkan pemahaman di atas, ikuti langkah berikut untuk mempersiapkan proses *debugging* pada Netbeans IDE

a. *Setting Breakpoint*

Pilih baris kode yang akan dianalisa, kemudian klik pada nomor baris tersebut, lalu baris akan berwarna merah yang menunjukkan baris *breakpoint*.

Gambar 4



A screenshot of a Java code editor showing a simple program to calculate the area of a rectangle. The code is as follows:

```
19 public static void main(String[] args) {
20     // TODO code application logic here
21     Double lebar,luas, panjang, Luas;
22     Luas = 0.0;
23     Scanner in = new Scanner(System.in);
24
25     System.out.print("Panjang : ");
26     panjang = in.nextDouble();
27
28     System.out.print("Lebar : ");
29     lebar = in.nextDouble();
30
31     luas = panjang * lebar;
32     System.out.print("Luas :" +luas);
33 }
34
35 }
```

The line `Scanner in = new Scanner(System.in);` is highlighted with a red background, indicating it is a breakpoint.

**B. Keterampilan yang Diperlukan dalam Mempersiapkan Kode Program**

1. Menyiapkan kode program sesuai spesifikasi.
2. Menyiapkan *debugging tools* untuk melihat proses suatu modul.

**C. Sikap Kerja yang Diperlukan dalam Mempersiapkan Kode Program**

Harus bersikap secara:

1. Menyiapkan kode program sesuai spesifikasi.
2. Menyiapkan *debugging tools* untuk melihat proses suatu modul.

## **BAB III**

### **MELAKUKAN *DEBUGGING***

#### **A. Pengetahuan yang Diperlukan dalam Melakukan *Debugging***

1. Cara Melakukan Kompilasi Kode Program Sesuai Bahasa Pemrograman yang Sesuai

Bahasa pemrograman Java adalah salah satu bahasa pemrograman populer yang paling banyak digunakan untuk saat ini. Selain bersifat sumber terbuka (*open source*), bahasa pemrograman Java juga tidak memerlukan lingkungan pengembangan (IDE) khusus sehingga kode program Java dapat dibuat menggunakan *teks editor* seperti Notepad, Notepad++ atau *text editor* lainnya.

Berbeda dengan bahasa pemrograman Java, bahasa pemrograman lain seperti Visual Basic, Visual Foxpro atau Borland Delphi bersifat komersial dan mempunyai lingkungan pengembangan (IDE) tersendiri. Untuk membuat program Java yang dapat dieksekusi (*executable*), Anda cukup melalui tiga langkah atau tahapan mudah yaitu:

- a. Menulis kode program Java, bisa menggunakan *editor* Notepad, Notepad++ atau yang lain.
- b. Mengkompilasi (*compile*) file kode sumber Java (Java source code) yang berekstensi .java.
- c. Menjalankan (*run*) file *bytecode* Java yang berekstensi .class.

Berikut adalah cara untuk melakukan kompilasi kode program java :

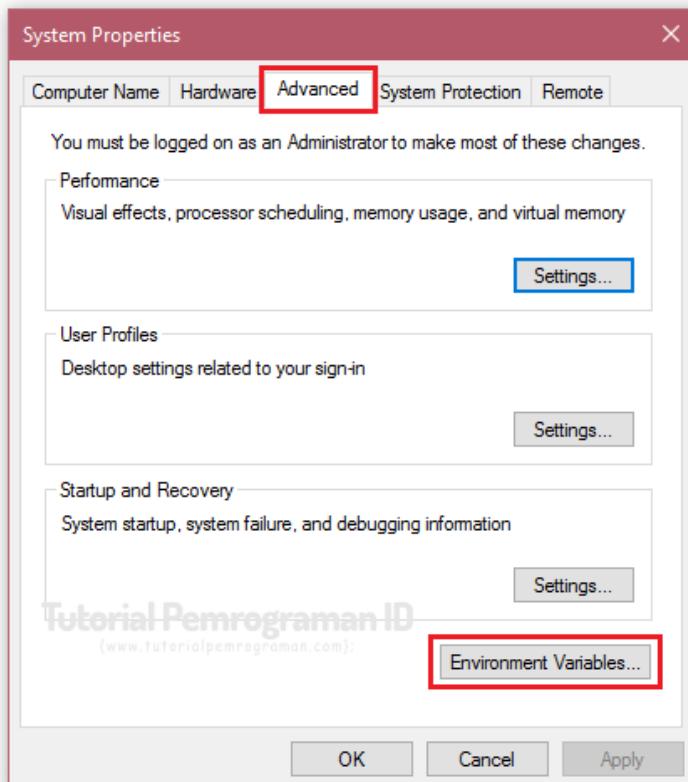
- a. *Update PATH Environment Variable*

*Install* versi JDK terbaru pada <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>. Jika sudah terinstall, buka *folder* instalasi Java dan cari *folder Bin*. *Path* dari *directory Bin* akan seperti ini 'C:\Program Files\Java\jdk1.8.0\_102\bin'.

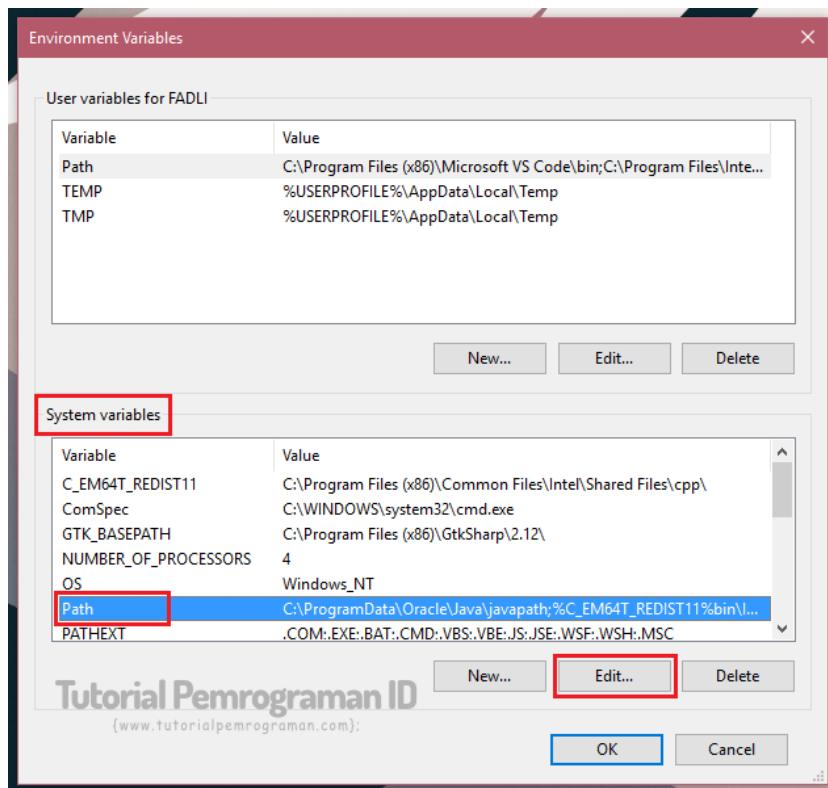
Cara *setting path variable* :

1. Buka *Advanced system settings*, pilih tab *Advanced* dan klik *Environment Variables*

Gambar 5

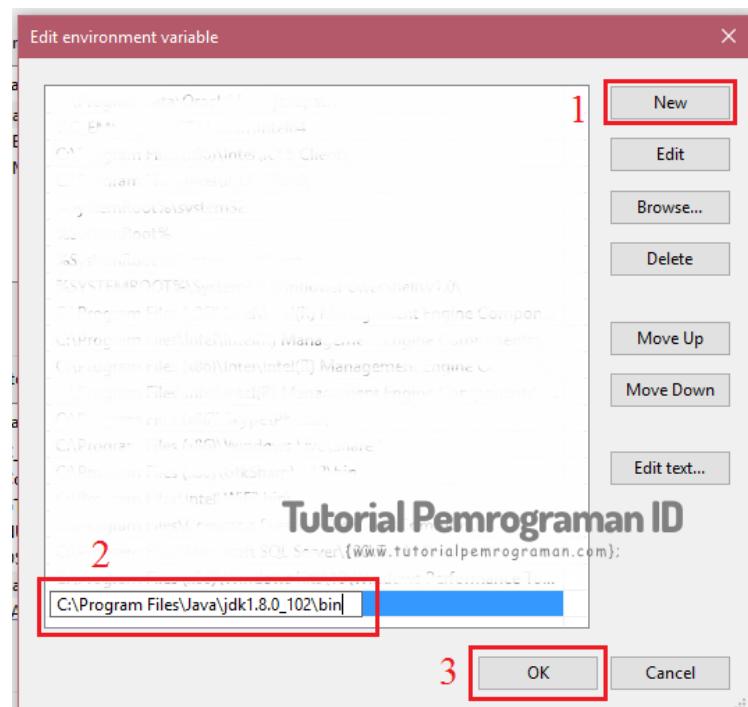


2. Pada *System Variables*, cari variable *Path*, kemudian klik *Edit*  
Gambar 6



3. Klik *New*, kemudian paste-kan alamat *folder* yang tadi telah di *copy* (C:\Program Files\Java\jdk1.8.0\_102\bin).

Gambar 7



b. Kompilasi Kode Program

Teknik kompilasi yang digunakan saat ini yaitu menggunakan *command prompt*. Jika *Path* dari *directory* Java sudah ditambahkan ke dalam *System Variables* maka untuk melakukan kompilasi kode program dapat dilakukan melalui *command prompt*, dengan cara buka *command prompt* kemudian masuk ke dalam *folder* tempat kode program java yang akan dikompilasi.

```
cd C:\namafolder
```

Lalu masukan perintah di bawah untuk mulai mengkompilasi kode program

```
C:\namafolder > javac debugging.java
```

2. Cara Menganalisa Kriteria Lulus *Build*

Pada bahasan ini, untuk melakukan *build* pada bahasa pemrograman java akan menggunakan Netbeans IDE. Berikut adalah langkah-langkah untuk melakukan build pada Netbeans IDE

- a. Buka netbeans pastikan *project* anda dapat di run dengan baik sebelum membuat jar. karena percuma jika anda membuat file .jar dengan *project* yang tidak bisa dijalankan.
- b. Pada menu bar klik *run* --> *clean and build project* atau bisa juga menggunakan *shortkey* shift + f11.
- c. Jika proses *build* berhasil dilakukan maka akan terdapat tulisan "*BUILD SUCCESSFUL* (total time: 0 seconds)".

Namun Jika proses *build* gagal maka terdapat tulisan pesan *error "BUILD FAILED (total time: 0 seconds)"*. Biasanya penyebab kegagalan *build* adalah adanya kesalahan peulisan sintaks pada baris program yang akan di *build*.

3. Cara Menganalisa Kriteria Eksekusi Aplikasi

Untuk melakukan eksekusi aplikasi, tulislah program seperti di bawah ini :

Gambar 8

```
public static void main(String[] args) {
    // TODO code application logic here
    Double alas, tinggi, Luas;
    Scanner in = new Scanner(System.in);

    System.out.print("Alas : ");
    alas = in.nextDouble();

    System.out.print("Tinggi : ");
    tinggi = in.nextDouble();

    Luas = 0.5 * alas * tinggi;
    System.out.print("Luas :" + Luas);
}
```

Jika sudah maka jalankan program perhitungan luas segtiga tersebut pada aplikasi Netbeans IDE dengan klik menu *run* -> *run project* atau bisa dengan menggunakan *shortkey* f6. Jika program berhasil dijalankan maka akan muncul seperti gambar di bawah ini :

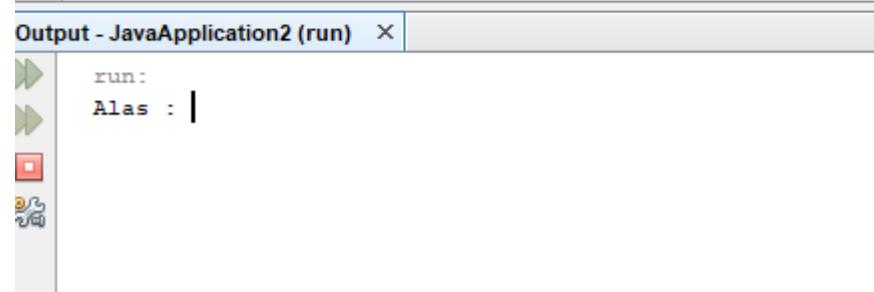
Gambar 9

```
19  public static void main(String[] args) {
20      // TODO code application logic here
21      Double alas, tinggi, Luas;
22      Scanner in = new Scanner(System.in);

23
24      System.out.print("Alas : ");
25      alas = in.nextDouble();

26
27      System.out.print("Tinggi : ");
28      tinggi = in.nextDouble();

29
30      Luas = 0.5 * alas * tinggi;
31      System.out.print("Luas :" + Luas);
32  }
33
34
35
```



Pada gambar 9, pengguna diminta untuk memasukkan nilai alas, jika sudah dimasukan nilai alas, kemudia tekan tombol *enter* untuk melanjutkan proses eksekusi program.

Gambar 10

The screenshot shows a Java code editor and an output window. The code editor displays a Java main method with variable declarations and input/output statements. The output window shows the execution of the program, prompting for 'Alas' and 'Tinggi', and then displaying the calculated area 'Luas'.

```
19 | public static void main(String[] args) {
20 |     // TODO code application logic here
21 |     Double alas, tinggi, Luas;
22 |     Scanner in = new Scanner(System.in);
23 |
24 |     System.out.print("Alas : ");
25 |     alas = in.nextDouble();
26 |
27 |     System.out.print("Tinggi : ");
28 |     tinggi = in.nextDouble();
29 |
30 |     Luas = 0.5 * alas * tinggi;
31 |     System.out.print("Luas :" + Luas);
32 |
33 |
34 }
35 }
```

Output - JavaApplication2 (run) ×

run:  
Alas : 4  
Tinggi : |

Kemudian masukan nilai tinggi pada saat eksekusi program yang sedang berjalan. Jika sudah tekan tombol *enter* pada *keyboard* untuk melanjutkan proses eksekusi dari program.

Gambar 11

The screenshot shows a Java application named "JavaApplication2". The code in the editor is:

```
19 public static void main(String[] args) {
20     // TODO code application logic here
21     Double alas, tinggi, Luas;
22     Scanner in = new Scanner(System.in);
23
24     System.out.print("Alas : ");
25     alas = in.nextDouble();
26
27     System.out.print("Tinggi : ");
28     tinggi = in.nextDouble();
29
30     Luas = 0.5 * alas * tinggi;
31     System.out.print("Luas :" + Luas);
32 }
33
34 }
```

The line `Luas = 0.5 * alas * tinggi;` is highlighted with a yellow background. The output window shows:

Output - JavaApplication2 (run) ×

```
run:
Alas : 4
Tinggi : 8
Luas :16.0BUILD SUCCESSFUL (total time: 3 minutes 46 seconds)
```

Apabila aplikasi sudah tereksekusi sepenuhnya, maka akan terdapat tulisan "*BUILD SUCCESSFUL*" beserta waktu total eksekusi aplikasi tersebut seperti pada gambar 10.

#### 4. Cara Menganalisa Kode Kesalahan

Secara garis besar *error* dalam pemrograman Java ini ada beberapa jenis, yaitu:

a. *Syntax error*

*Syntax* merupakan suatu aturan penulisan yang sudah ditetapkan pada struktur elemen - elemen dalam bahasa pemrograman. Jadi *syntax error* ini adalah kesalahan dalam *coding* karena aturan penulisan yang tidak sesuai atau kesalahan pada konstruksi kode, misalnya :

1. Salah dalam menuliskan *keyword Java*
2. Tidak menggunakan tanda kurung kurawal untuk pernyataan

3. Tidak menggunakan tanda atau karakter yang sesuai, misalnya tidak menggunakan tanda " ", untuk tipe data *String*.

*Syntax error* ini mudah ditelusuri atau ditemukan karena *compiler* akan memberi tahu kita di mana letak kesalahan dalam penulisan kode program.

Gambar 12

```
public class JavaApplication2 {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        //Seharusnya "Selamat datang di pemrograman Java"  
        System.out.println("Selamat datang di pemrograman Java");  
    }  
  
}
```

Kode di atas memiliki *error*, yaitu seharusnya teks selamat datang di pemrograman Java berada dalam tanda " ". Oleh karena itu, setelah anda melakukan *compile*, maka *compiler* akan memberi tahu *error* tersebut seperti gambar di bawah ini :

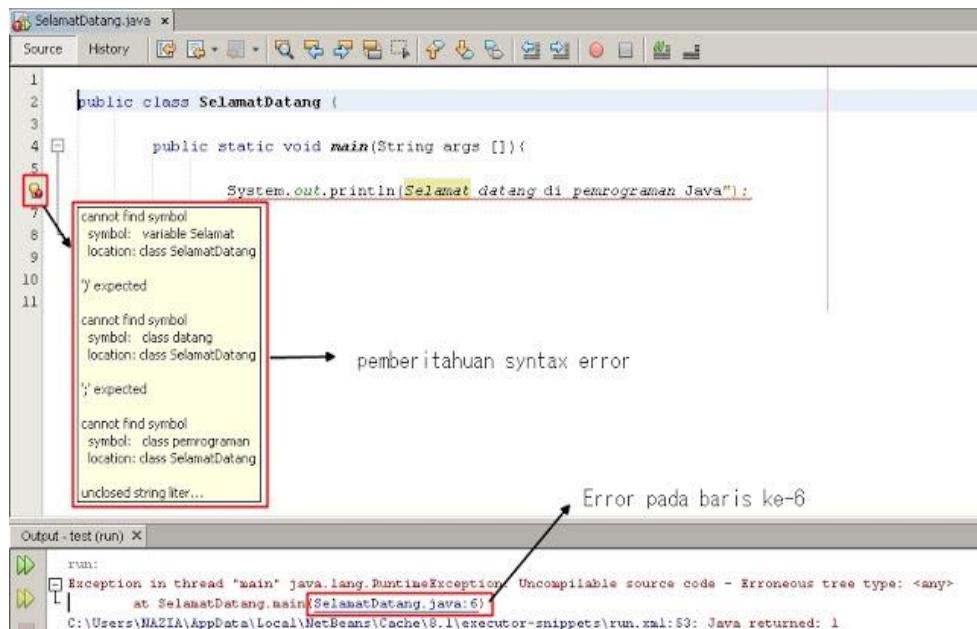
1. *Syntax error* pada *command prompt*

Gambar 13

```
C:\Users\NAZIA\Documents\Java Notepad>java SelamatDatang  
Error: Main method is not static in class SelamatDatang, please define the main  
method as:  
    public static void main(String[] args)  
  
C:\Users\NAZIA\Documents\Java Notepad>javac SelamatDatang.java  
SelamatDatang.java:5: error: ')' expected  
    System.out.println("Selamat datang di pemrograman Java");  
                                         ^  
SelamatDatang.java:5: error: ';' expected  
    System.out.println("Selamat datang di pemrograman Java");  
                                         ^  
SelamatDatang.java:5: error: unclosed string literal  
    System.out.println("Selamat datang di pemrograman Java");  
                                         ^  
SelamatDatang.java:5: error: not a statement  
    System.out.println("Selamat datang di pemrograman Java");  
                                         ^  
4 errors
```

2. *Syntax error* pada Netbeans IDE

Gambar 14



b. *Runtime error*

*Runtime error* merupakan salah satu jenis *error* yang terjadi karena program secara tiba-tiba dihentikan ketika program tersebut sedang berjalan. *Runtime error* ini dapat terjadi bila program tidak bisa menangani operasi yang tidak mungkin untuk dilakukan.

Contoh *runtime error* ini dapat terjadi karena:

1. Ketika program meminta *user* untuk menginputkan angka, namun *user* malah menginputkan huruf
2. Ketika melakukan perhitungan pembagian angka 1 dengan 0

Sebagai contoh:

Gambar 15

```
public static void main(String[] args) {
    System.out.println( 1 / 0 );
}
```

Pada program di atas akan menampilkan pesan error karena program diminta untuk membagi angka 1 dengan angka nol. Pada *command prompt* akan menampilkan pesan *error* seperti di bawah ini :

Gambar 16

```
C:\Users\NAZIA\Documents\Java Notepad>java CekError
Exception in thread "main" java.lang.ArithmaticException: / by zero
at CekError.main(CekError.java:5)
```

c. *Logic error*

*Logic error* terjadi bila program berjalan dan tidak memberikan hasil seperti yang diinginkan. Misalnya anda membuat program konversi dari nilai centimeter ke nilai milimeter. Seperti sudah diketahui bahwa :

$$1 \text{ cm} = 10 \text{ mm}$$

Contoh program sederhananya :

Gambar 16

```
public static void main(String[] args) {  
    System.out.println("1 cm itu sama dengan ");  
    //Konversi cm ke mm  
    System.out.println( 1.0 / 10.0 + "mm");  
}
```

Setelah mengeksekusi program di atas, maka akan terlihat pada *command prompt* seperti di bawah ini:

Gambar 17

C:\Users\NAZIA\Documents\Java Notepad>javac Konversi.java  
C:\Users\NAZIA\Documents\Java Notepad>java Konversi  
1 cm itu sama dengan  
0.1mm

A screenshot of a terminal window on a Windows operating system. The window has a black background and white text. It shows the command 'javac Konversi.java' being run, followed by the output of the program 'Konversi'. The output is '1 cm itu sama dengan' on the first line and '0.1mm' on the second line, both enclosed in a yellow rectangular box.

Program di atas berjalan dengan mulus ketika dieksekusi baik ketika menggunakan IDE Netbeans ataupun Eclipse, namun bila diperhatikan terlihat nilai bahwa 1 cm itu sama dengan 0.1 mm, tentu saja ini adalah hal yang salah, dan inilah yang disebut dengan *logic error*.

## B. Keterampilan yang Diperlukan dalam Melakukan *Debugging*

1. Menjelaskan cara melakukan kompilasi kode program sesuai bahasa pemrograman yang digunakan.
2. Menganalisa kriteria lulus *build*.
3. Menganalisa kriteria eksekusi aplikasi.
4. Menganalisa kode kesalahan.

### C. Sikap Kerja yang Diperlukan dalam Melakukan *Debugging*

Harus bersikap secara:

1. Cekatan dalam melakukan langkah-langkah, panduan dan pedoman.
2. Teliti ketika dalam melakukan debugging.
3. Pantang menyerah saat melakukan proses debugging.

## BAB IV

### MEMPERBAIKI PROGRAM

#### A. Pengetahuan yang Diperlukan dalam Memperbaiki Program

1. Cara merumuskan perbaikan terhadap kesalahan kompilasi maupun *build*

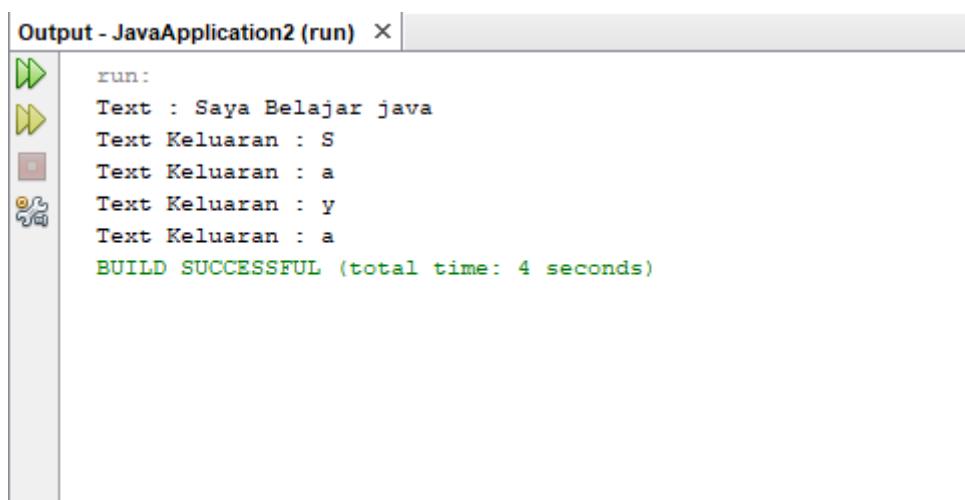
Untuk merumuskan perbaikan, buatlah sebuah program menggunakan Netbeans IDE untuk mencetak text satu per satu seperti contoh di bawah ini:

Gambar 18

```
public static void main(String[] args) {  
    // TODO code application logic here  
    Scanner in = new Scanner(System.in);  
    |  
    System.out.print("Text : ");  
    String text = in.nextLine();  
    int i;  
  
    for(i=0; i < 4; i++){  
        System.out.println("Text Keluaran : "+text.charAt(i));  
    }  
}
```

Kemudian jalankan program tersebut dengan menekan menu *Run -> Run Project*. Ketika program tersebut dijalankan tidak ada *error* yang terjadi, namun hasil yang diinginkan tidak sesuai dengan kebutuhan. Berikut adalah hasil keluaran dari eksekusi program diatas :

Gambar 19



```
Output - JavaApplication2 (run) ×  
run:  
Text : Saya Belajar java  
Text Keluaran : S  
Text Keluaran : a  
Text Keluaran : y  
Text Keluaran : a  
BUILD SUCCESSFUL (total time: 4 seconds)
```

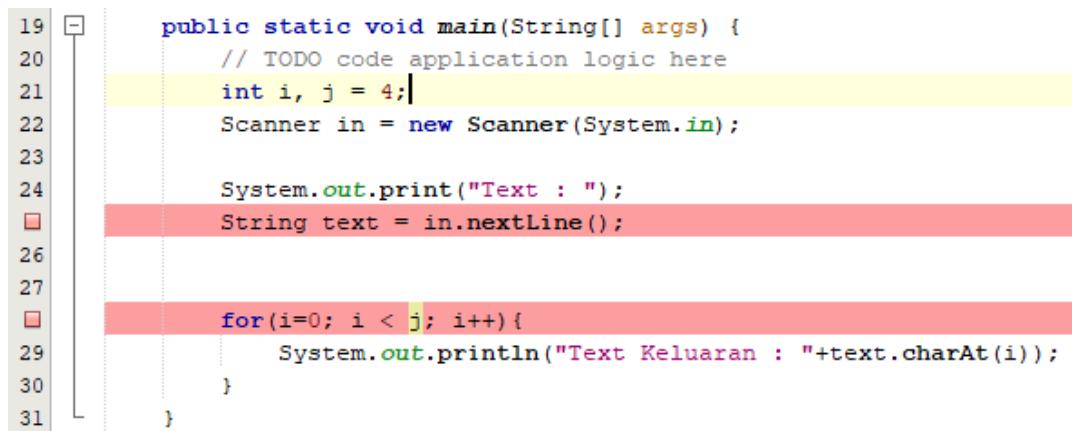
Ini merupakan salah satu jenis *logic error* karena program berjalan tanpa ada *error* pada saat kompilasi maupun *build*, namun hasil keluaran dari program yang dieksekusi tidak sesuai dengan keinginan.

## 2. Cara Melakukan perbaikan

Selanjutnya adalah memperbaiki program pada gambar 19, agar hasil keluaran dari eksekusi program tersebut sesuai dengan keinginan. Berikut adalah langkah-langkah untuk melakukan perbaikan program :

- Lakukan breakpoint seperti pada gambar di bawah ini

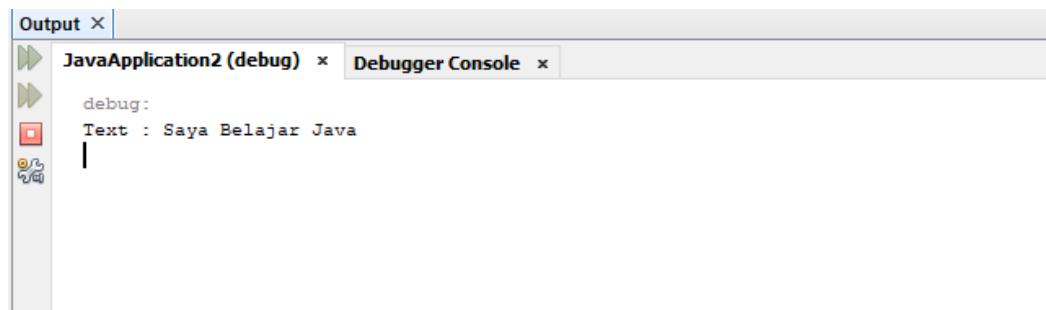
Gambar 20



```
19 public static void main(String[] args) {
20     // TODO code application logic here
21     int i, j = 4;
22     Scanner in = new Scanner(System.in);
23
24     System.out.print("Text : ");
25     String text = in.nextLine();
26
27     for(i=0; i < j; i++){
28         System.out.println("Text Keluaran : "+text.charAt(i));
29     }
30 }
```

- Lalu pilih menu *Debug -> Debug File*, maka akan muncul *console output* seperti pada gambar dibawah dan ketikan text kemudian tekan tombol *enter* pada *keyboard*

Gambar 21



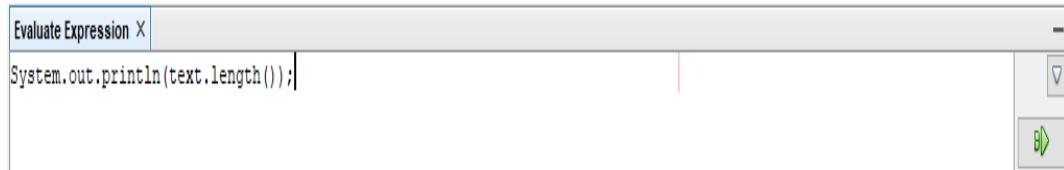
The screenshot shows the Eclipse IDE's Output view. The title bar says "Output X" and "JavaApplication2 (debug) x Debugger Console x". The content area displays the following text:  
debug:  
Text : Saya Belajar Java

- Pilih menu *Debug -> Evaluate Expression*
- Klik *step into*



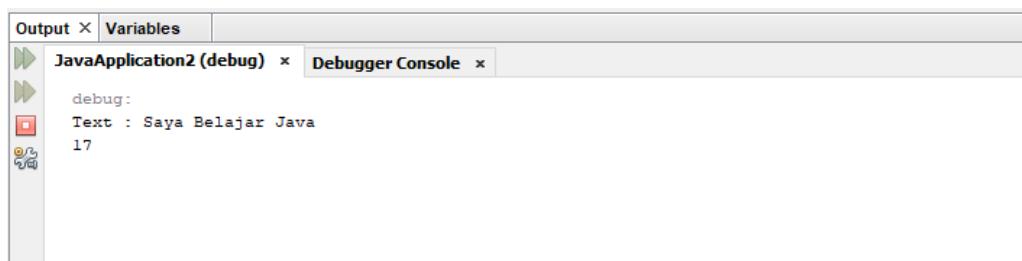
- Masukkan baris kode berikut untuk menghitung panjang *text* yang tadi sudah dimasukkan dan tekan tombol *evaluate code fragment*, seperti berikut

Gambar 21



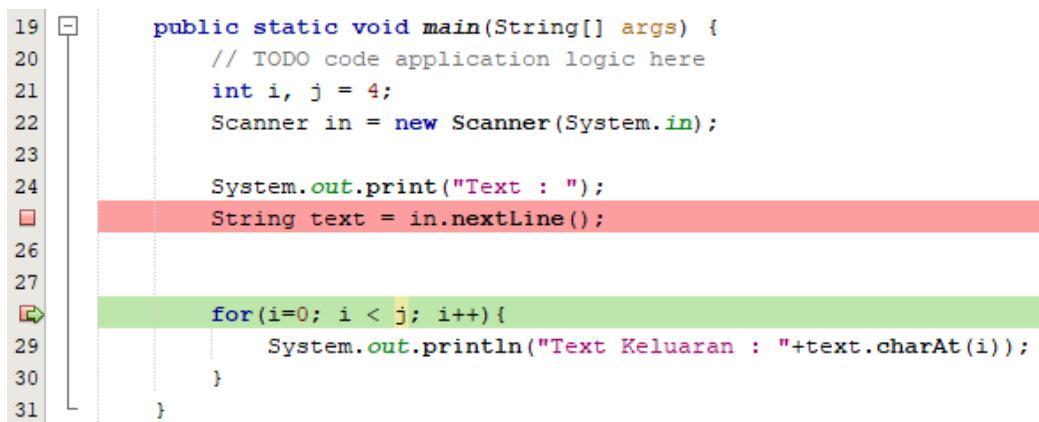
- f. Maka akan muncul nilai panjang dari *text* yang telah dimasukkan

Gambar 22



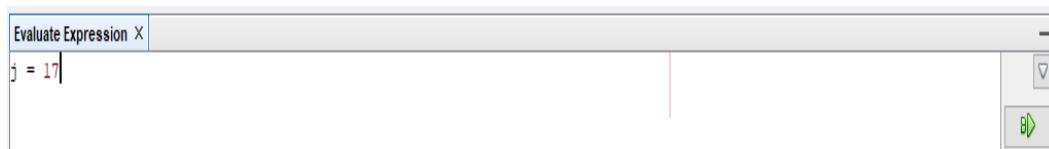
- g. Klik *step into* maka, eksekusi akan berhenti pada baris selanjutnya

Gambar 23



- h. Lalu ganti nilai *j* menjadi 17, angka 17 di dapat dari panjang *text* yang dimasukkan, kemudian tekan tombol *evaluate code fragment*

Gambar 24



- i. Kemudian pilih tombol *Continue* untuk menyelesaikan proses *debugging*

Gambar 25



Maka *console output* akan mengeluarkan hasil seperti berikut

Gambar 26

```
Output X | JavaApplication2 (debug) x Debugger Console x
debug:
Text : Saya belajar java
Text Keluaran : S
Text Keluaran : a
Text Keluaran : y
Text Keluaran : a
Text Keluaran :
Text Keluaran : b
Text Keluaran : e
Text Keluaran : l
Text Keluaran : a
Text Keluaran : j
Text Keluaran : a
Text Keluaran : r
Text Keluaran :
Text Keluaran : j
Text Keluaran : a
Text Keluaran : v
Text Keluaran : a
BUILD SUCCESSFUL (total time: 3 minutes 1 second)
```

- j. Kesalahan logika pada program di atas adalah pengulangan cetak "Text Keluaran" hanya terjadi selama 4 kali sedangkan panjang dari *Text* adalah 17. Maka dari itu *edit* program di atas menjadi seperti ini

Gambar 27

```
public static void main(String[] args) {
    // TODO code application logic here
    int i;
    Scanner in = new Scanner(System.in);

    System.out.print("Text : ");
    String text = in.nextLine();

    for(i=0; i < text.length(); i++){
        System.out.println("Text Keluaran : "+text.charAt(i));
    }
}
```

## **B. Keterampilan yang Diperlukan dalam Memperbaiki Program**

1. Merumuskan perbaikan terhadap kesalahan kompilasi maupun build.
2. Melakukan perbaikan.

## **C. Sikap kerja yang Diperlukan dalam Memperbaiki Program**

Harus bersikap secara:

1. Cekatan dalam melakukan langkah-langkah, panduan dan pedoman.
2. Teliti ketika dalam memperbaiki program.
3. Pantang menyerah saat memperbaiki program.