

MODUL - WEB PROGRAMMING 1

Oleh :
Priyanto Tamami, S.Kom.

Daftar Isi

1	Dasar-Dasar Pemrograman PHP	1
1.1	Tujuan	1
1.2	Pengantar	1
1.3	Praktek	2
1.3.1	Instalasi XAMPP	2
1.3.2	Instalasi Git dan Akun Github	9
1.3.3	Instalasi Visual Studio Code	24
1.3.4	Registrasi 000webhost.com	31
1.3.5	Aplikasi Selamat Datang	34
1.3.6	Unggah ke Github	40
1.4	Kesimpulan	42
1.5	Tugas	42
2	Operator, Variabel, dan Tipe Data	43
2.1	Tujuan	43
2.2	Pengantar	43
2.3	Praktek	43
2.3.1	Variabel	43
2.3.2	Tipe Data	48
2.3.3	Operator	50
2.4	Kesimpulan	61
2.5	Tugas	61
3	Array	63
3.1	Tujuan	63
3.2	Pengantar	63
3.3	Praktek	63
3.4	Kesimpulan	65
3.5	Tugas	65
4	Struktur Kontrol	67
4.1	Tujuan	67
4.2	Pengantar	67
4.3	Praktek	67
4.3.1	Seleksi / Percabangan if	67
4.3.2	Seleksi / Percabangan switch	70

4.3.3	Perulangan <code>while</code>	71
4.3.4	Perulangan <code>do-while</code>	72
4.3.5	Perulangan <code>for</code>	73
4.3.6	Perulangan <code>foreach</code>	74
4.3.7	Kontrol <code>break</code>	76
4.3.8	Kontrol <code>continue</code>	82
4.3.9	Kontrol <code>include</code> dan <code>require</code>	86
4.3.10	Kontrol <code>goto</code>	89
4.4	Kesimpulan	90
4.5	Tugas	90

Bab 1

Dasar-Dasar Pemrograman PHP

1.1 Tujuan

Pada Bab ini diharapkan mahasiswa mengenal bentuk *syntax* dasar dari bahasa pemrograman PHP dan lingkungan kerja yang mendukungnya.

1.2 Pengantar

PHP adalah bahasa pemrograman yang dijalankan di sisi *server* dalam bentuk *scripting*, artinya bahasa pemrograman ini tidak perlu di *compile* terlebih dahulu untuk dapat dijalankan, kita cukup menyiapkan *interpreter*-nya saja.

PHP biasanya digunakan untuk membangun sebuah aplikasi Web yang dinamis, dimana halaman dapat melakukan respon terhadap *request* yang dilakukan oleh pengguna.

PHP pun telah digunakan secara luas dan menjadi alternatif gratis dibandingkan menggunakan bahasa sejenis seperti ASP milik Microsoft.

Untuk memulai melakukan praktik bahasa pemrograman menggunakan PHP, maka kita perlu mempersiapkan perangkat pendukung. Cara yang mungkin paling mudah adalah kita menggunakan aplikasi paket yang di dalamnya sudah terdapat *web server* yang mendukung PHP serta basis data yang akan digunakan, aplikasi yang mungkin dapat kita gunakan adalah :

1. LAMP, yang sebetulnya adalah singkatan dari Linux, Apache, MySQL, dan PHP. Tentunya aplikasi ini ditujukan untuk sistem operasi Linux, yang menggunakan Apache sebagai *web server* yang tentunya *plugin* untuk mendukung bahasa PHP sudah ada di dalamnya, dan MySQL sebagai basis datanya.
2. WAMP, adalah singkatan dari Windows, Apache, MySQL, dan PHP. Mirip seperti LAMP, hanya ini ditujukan bagi sistem operasi Windows.
3. MAMP, adalah singkatan dari Mac, Apache, MySQL, dan PHP. Untuk aplikasi ini dikhawasukan bagi sistem operasi Mac.

4. XAMPP, yang ini mendukung ketiga sistem operasi di atas dengan kelebihan mampu untuk mengolah bahasa pemrograman Perl.

Maka pilihan untuk praktik Web Programming 1 kita akan menggunakan XAMPP agar adaptasi antar sistem operasi lebih mudah. XAMPP dapat diunduh pada alamat <https://www.apachefriends.org>.

Aplikasi pendukung lain untuk melakukan kegiatan praktikum kita adalah sebagai berikut :

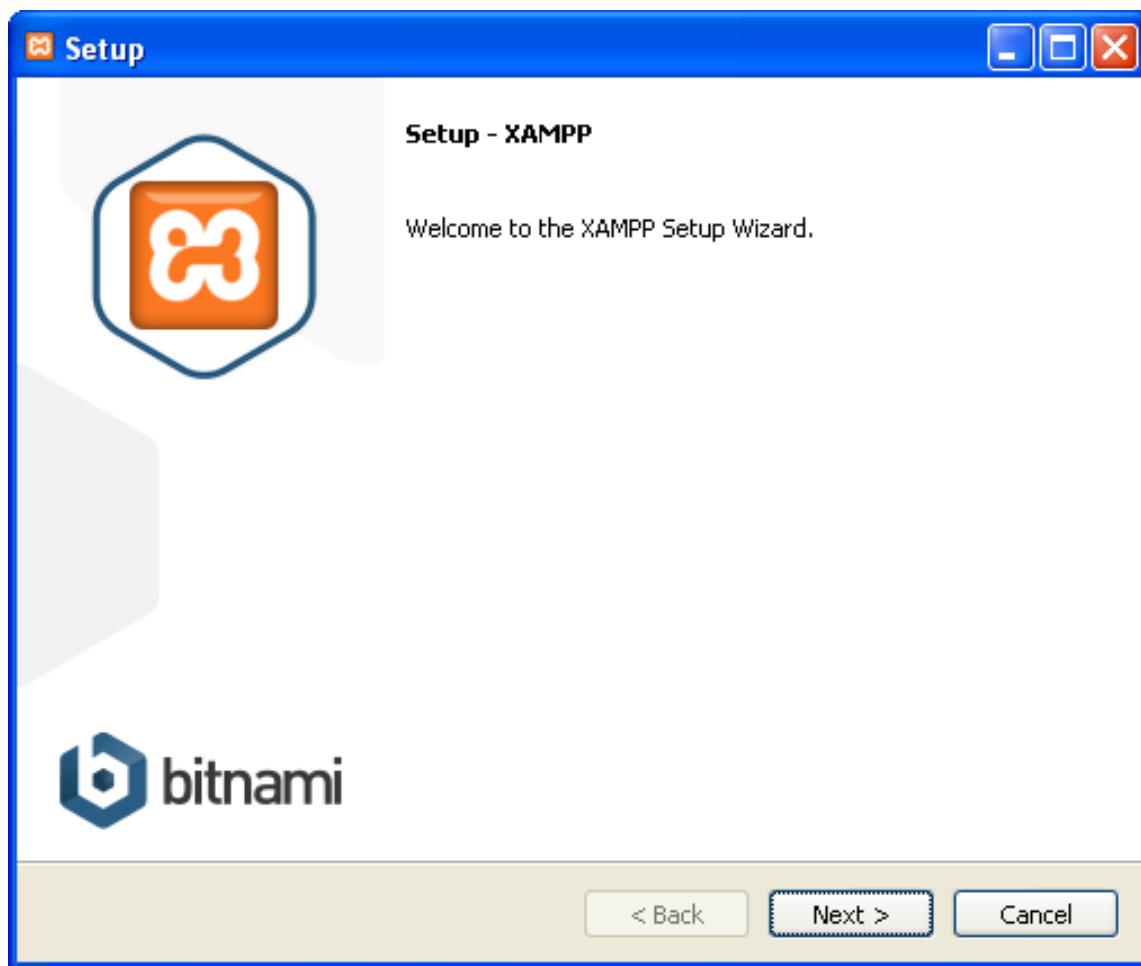
1. Git. Aplikasi ini digunakan untuk melakukan *versioning* sehingga kita lebih mudah dalam melakukan kontrol perubahan yang terjadi pada kode program yang kita bangun. Server yang kita gunakan untuk menyimpan repositori hasil *versioning* kita ada di alamat <https://github.com>. Github ini gratis. Untuk aplikasi Git dapat kita unduh di alamat <https://git-scm.com/>
2. Visual Studio Code. Aplikasi ini adalah *editor* yang akan digunakan dalam kegiatan praktikum pada mata kuliah Web Programming 1. Aplikasi ini gratis dan dapat diunduh pada alamat <https://code.visualstudio.com/> dengan dukungan instalasi untuk 3 (tiga) sistem operasi yang banyak digunakan, yaitu Linux, Windows, dan MacOS.
3. www.000webhost.com. Ini adalah layanan *hosting* gratis yang mampu menjalankan *script* PHP dengan fasilitas sistem basis data MySQL. Yang akan kita gunakan sebagai tempat aplikasi yang telah kita bangun sampai dengan akhir tatap muka mata kuliah ini.

1.3 Praktek

1.3.1 Instalasi XAMPP

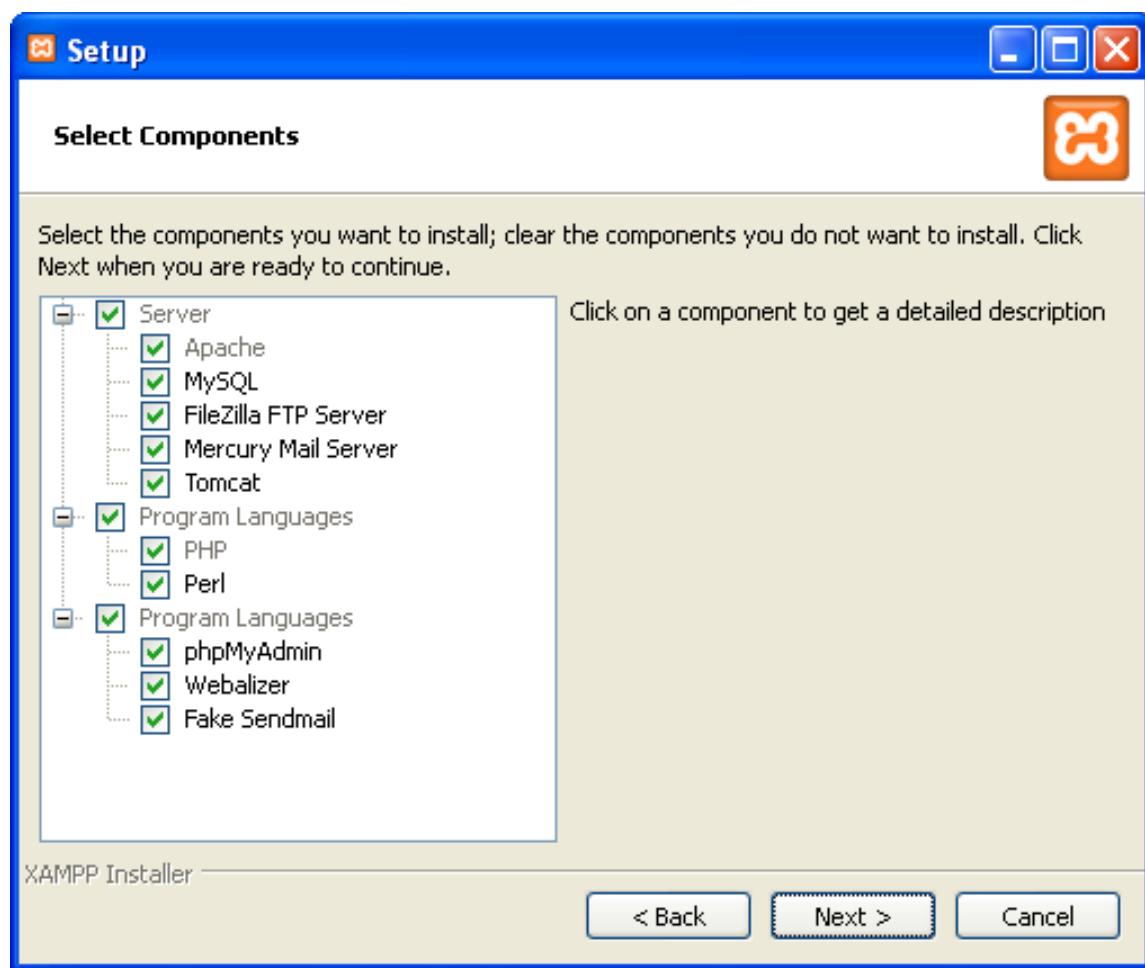
Langkah-langkah untuk instalasi XAMPP adalah sebagai berikut :

1. Tentunya mengunduh *installer* aplikasi ini dari alamat <https://www.apachefriends.org>, pilihlah yang model *installer* agar lebih mudah proses instalasinya.
2. Awal proses instalasi, kita akan bertemu dengan tampilan seperti pada gambar 1.1 berikut ini :



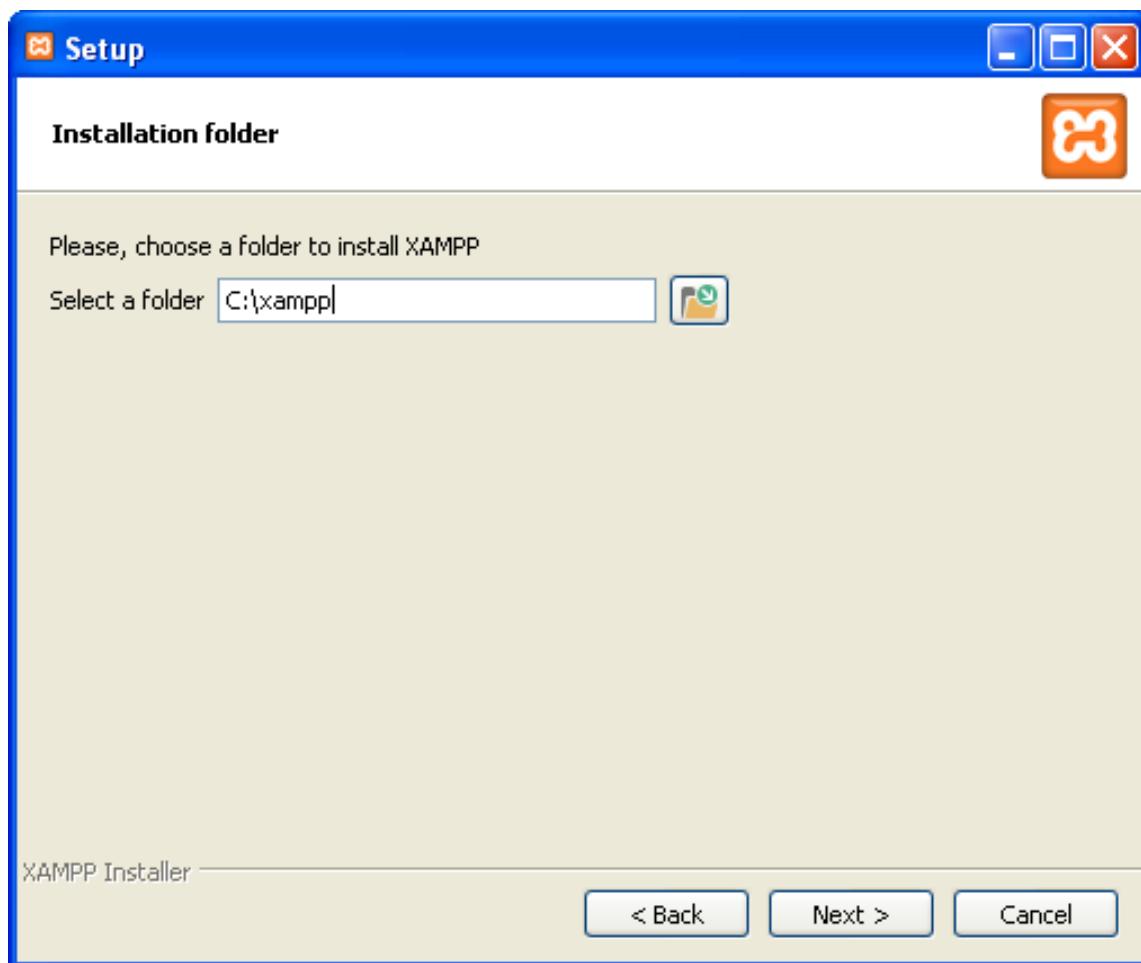
Gambar 1.1: Awal Instalasi XAMPP

3. Setelah itu kita diminta untuk memilih komponen yang akan ikut diinstall / dipasangkan seperti pada gambar 1.2 berikut :



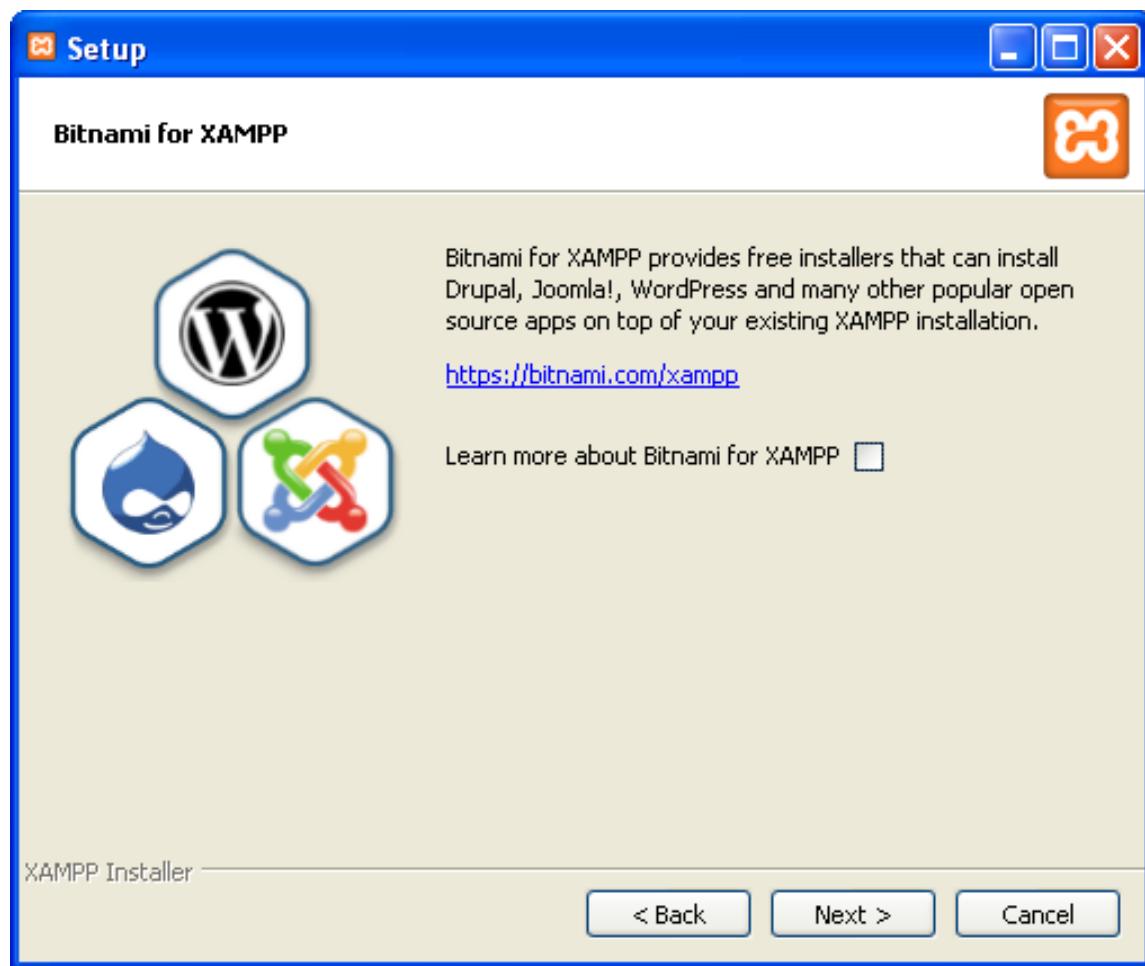
Gambar 1.2: Pemilihan Komponen

4. Selanjutnya adalah penetapan *folder* instalasi, akan ditaruh dimana hasil instalasi dari XAMPP ini. Tampilannya seperti pada gambar 1.26 berikut ini :



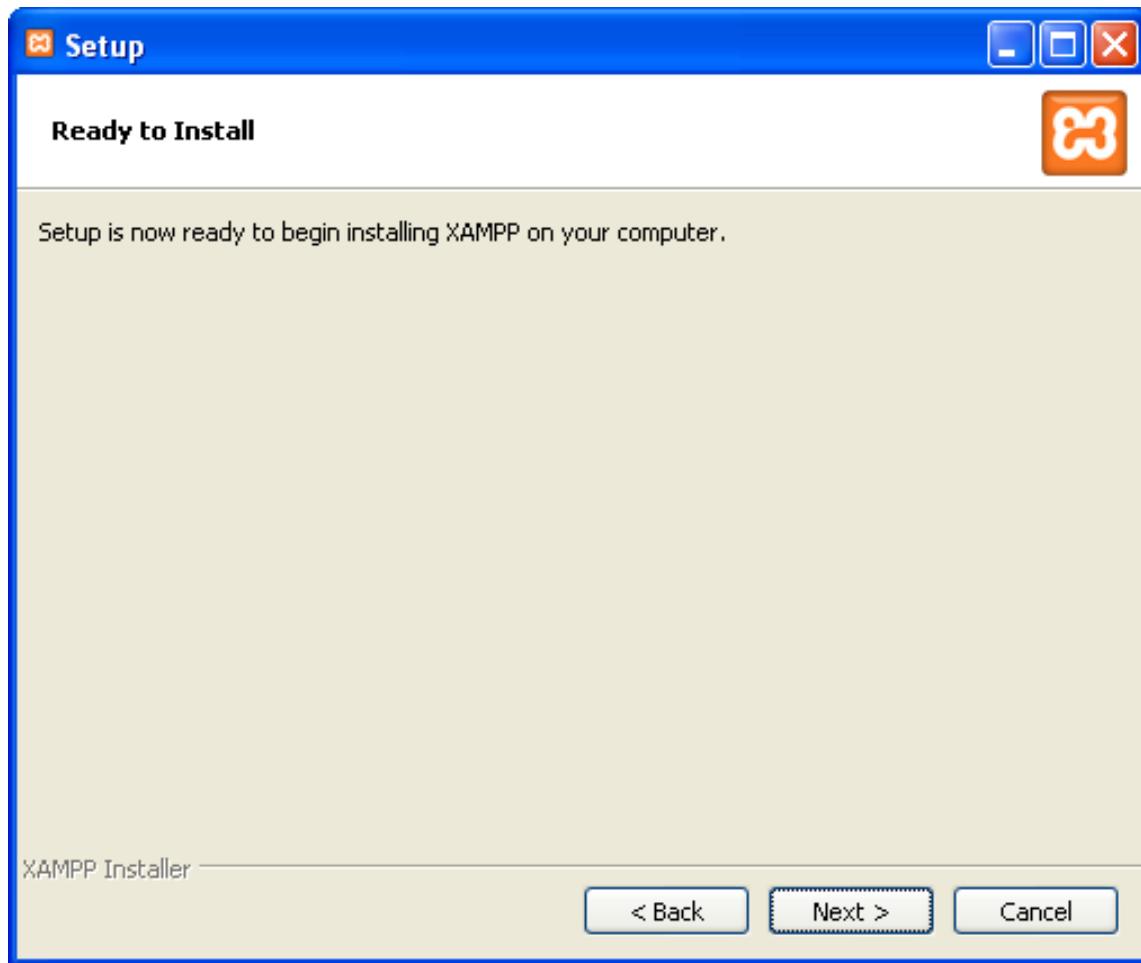
Gambar 1.3: Pemilihan *Folder* Instalasi

5. Langkah berikutnya adalah informasi mengenai Bitnami, Bitnami ini adalah aplikasi gratis yang mampu melakukan *instalasi* aplikasi lain semacam Drupal atau Joomla di atas XAMPP, tampilannya seperti pada gambar 1.4 berikut :



Gambar 1.4: Informasi Bitnami

6. Setelah menekan tombol **Next**, maka akan muncul informasi bahwa aplikasi siap diinstall seperti pada gambar 1.5 berikut :



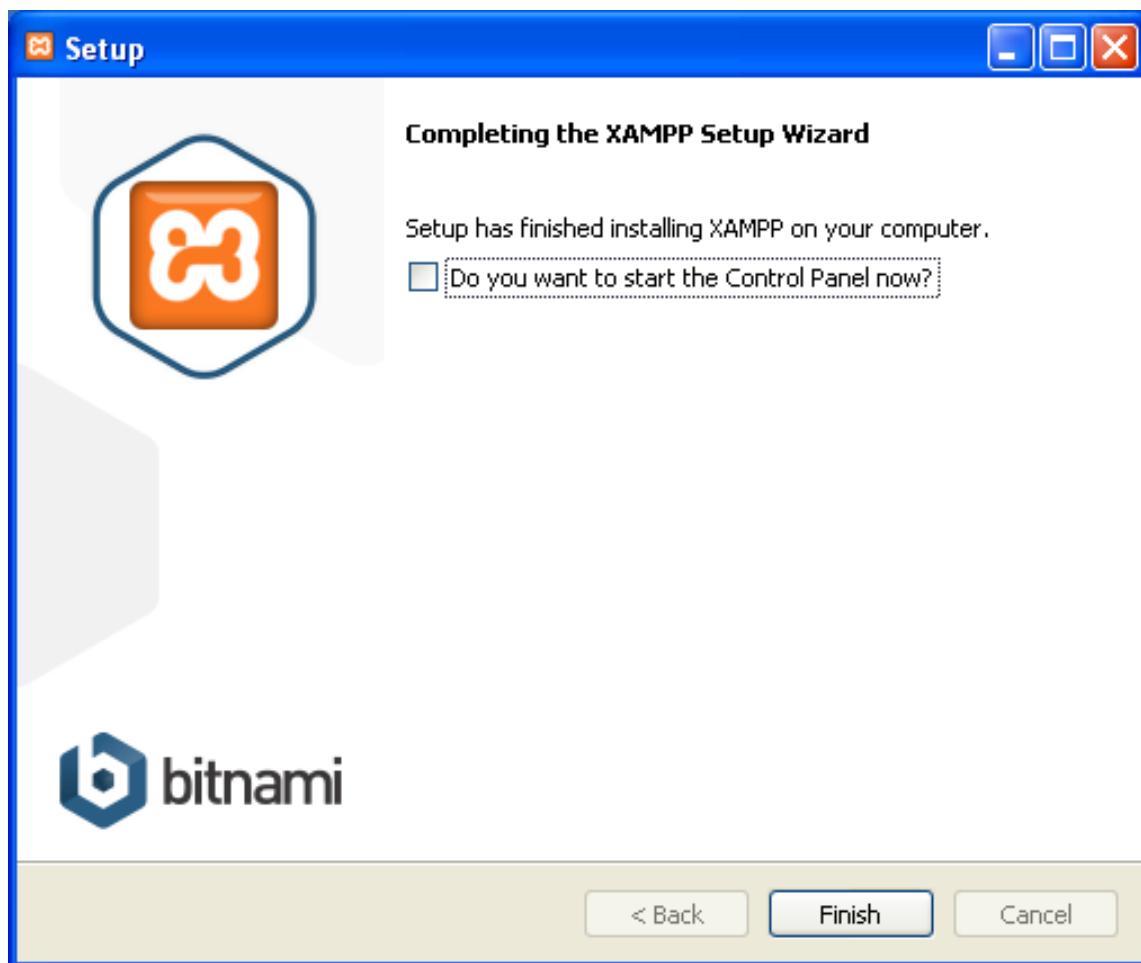
Gambar 1.5: Instalasi Sudah Siap

7. Setelah itu proses instalasi akan berjalan seperti pada gambar 1.6 berikut :



Gambar 1.6: Instalasi berjalan

8. Setelah proses instalasi selesai, maka akan muncul informasi seperti pada gambar 1.7 berikut :



Gambar 1.7: Instalasi Selesai

Sampai sini, proses instalasi XAMPP telah berhasil dan siap untuk digunakan.

1.3.2 Instalasi Git dan Akun Github

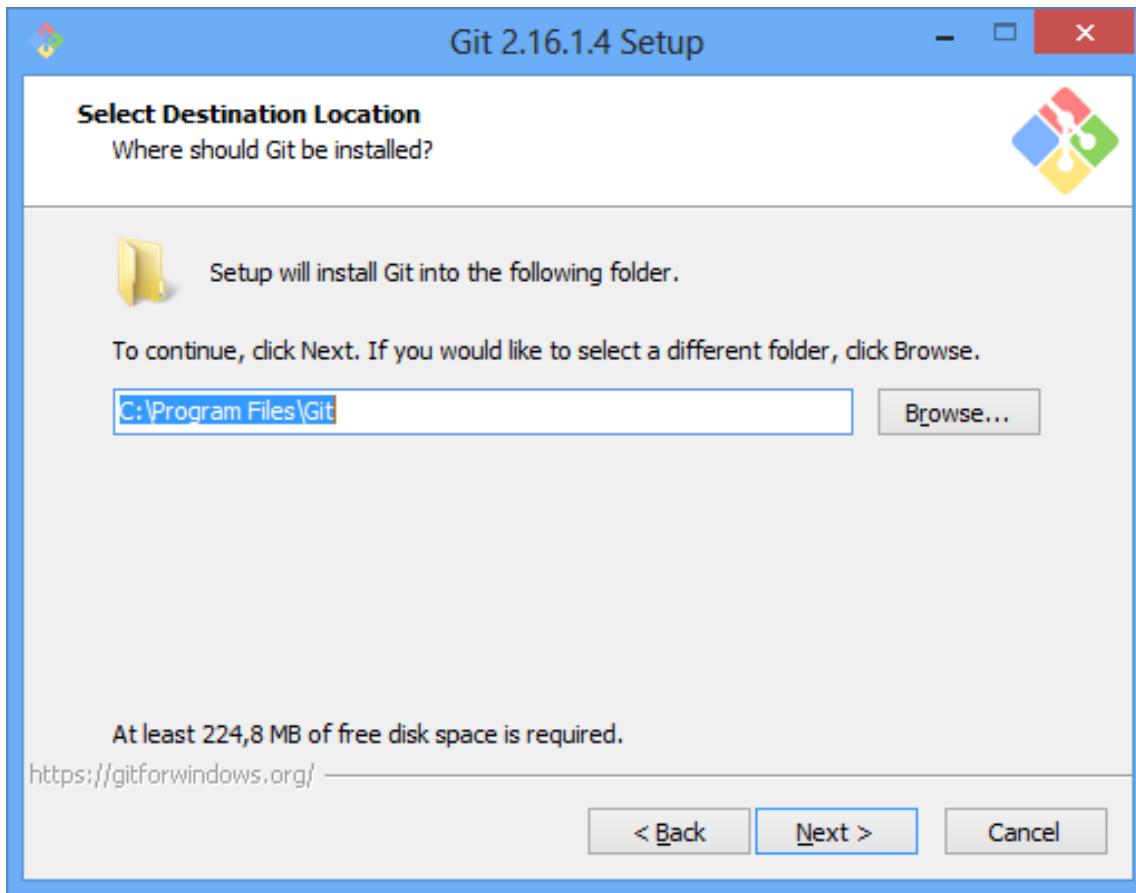
Proses instalasi dari Git adalah sebagai berikut :

1. Awalnya akan disajikan informasi mengenai lisensi seperti pada gambar 1.8 berikut :



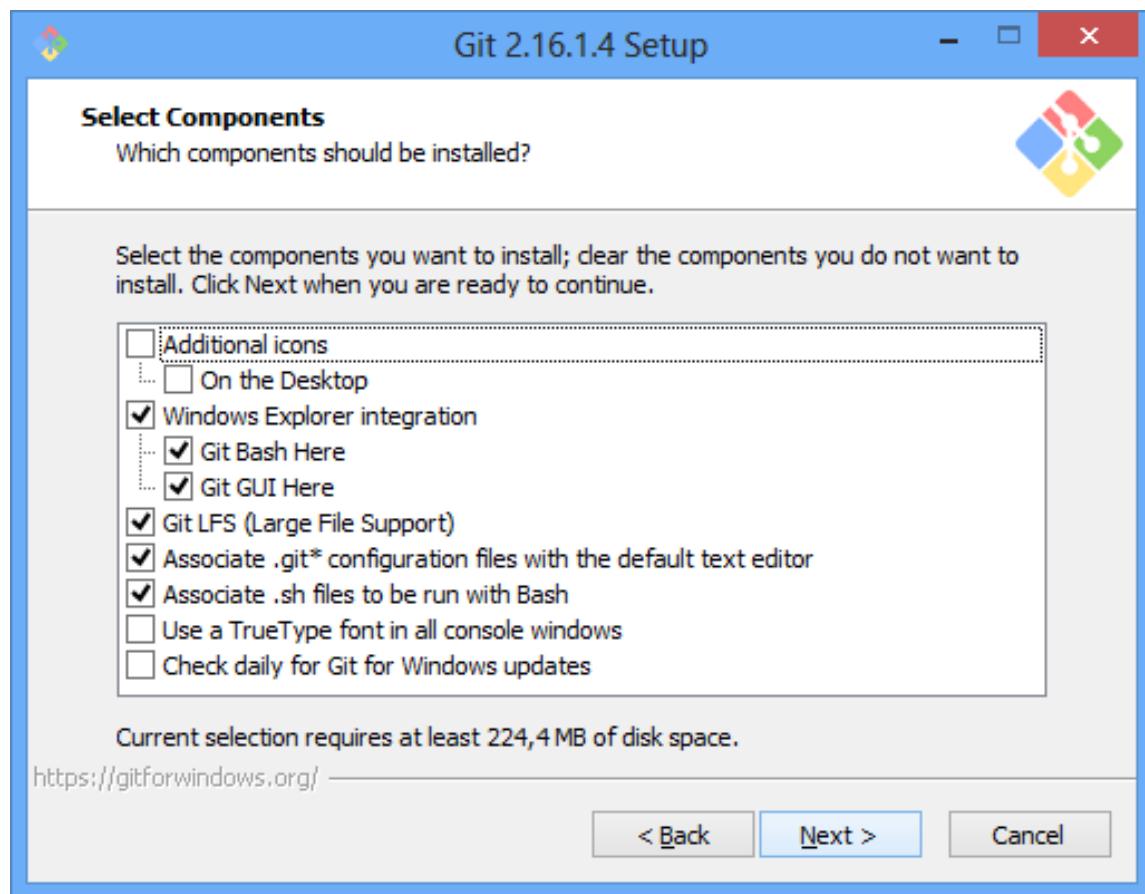
Gambar 1.8: Informasi Lisensi

2. Berikutnya akan ditampilkan lokasi dimana aplikasi Git akan diinstall, tampilannya adalah seperti pada gambar 1.9 berikut :



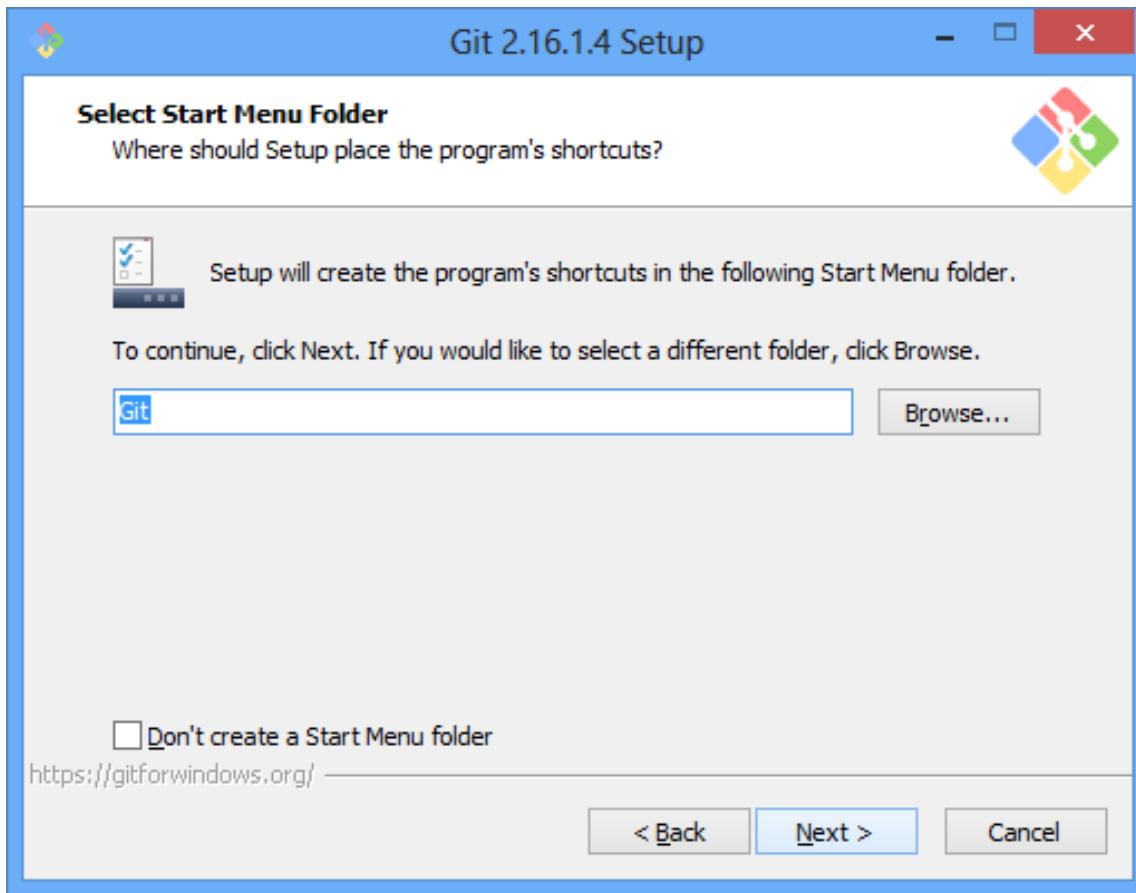
Gambar 1.9: Lokasi Tempat Instalasi Git

3. Berikutnya adalah pemilihan komponen yang akan diinstall di komputer, tampilannya adalah seperti pada gambar 1.10 berikut :



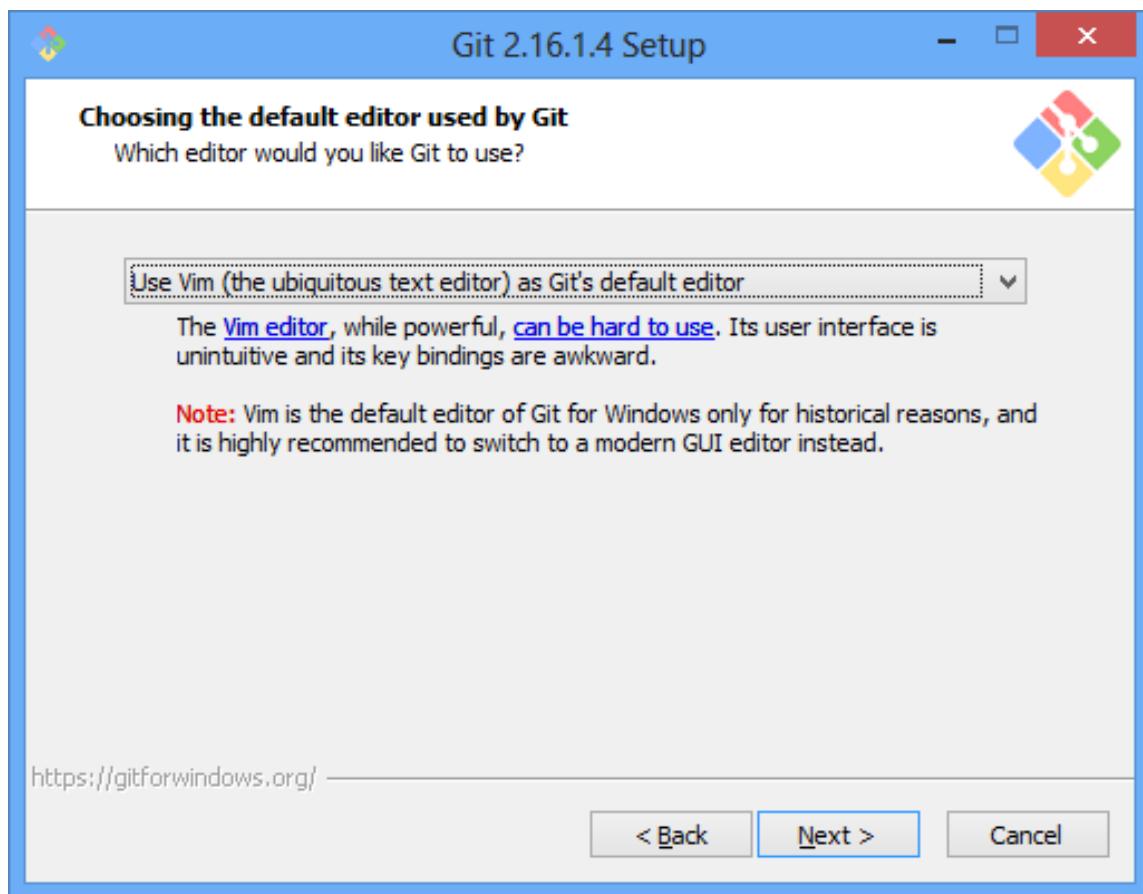
Gambar 1.10: Pemilihan Komponen

4. Selanjutnya adalah lokasi pada **start menu** dimana Git akan ditempatkan, tampilannya adalah seperti pada gambar 1.11 berikut :



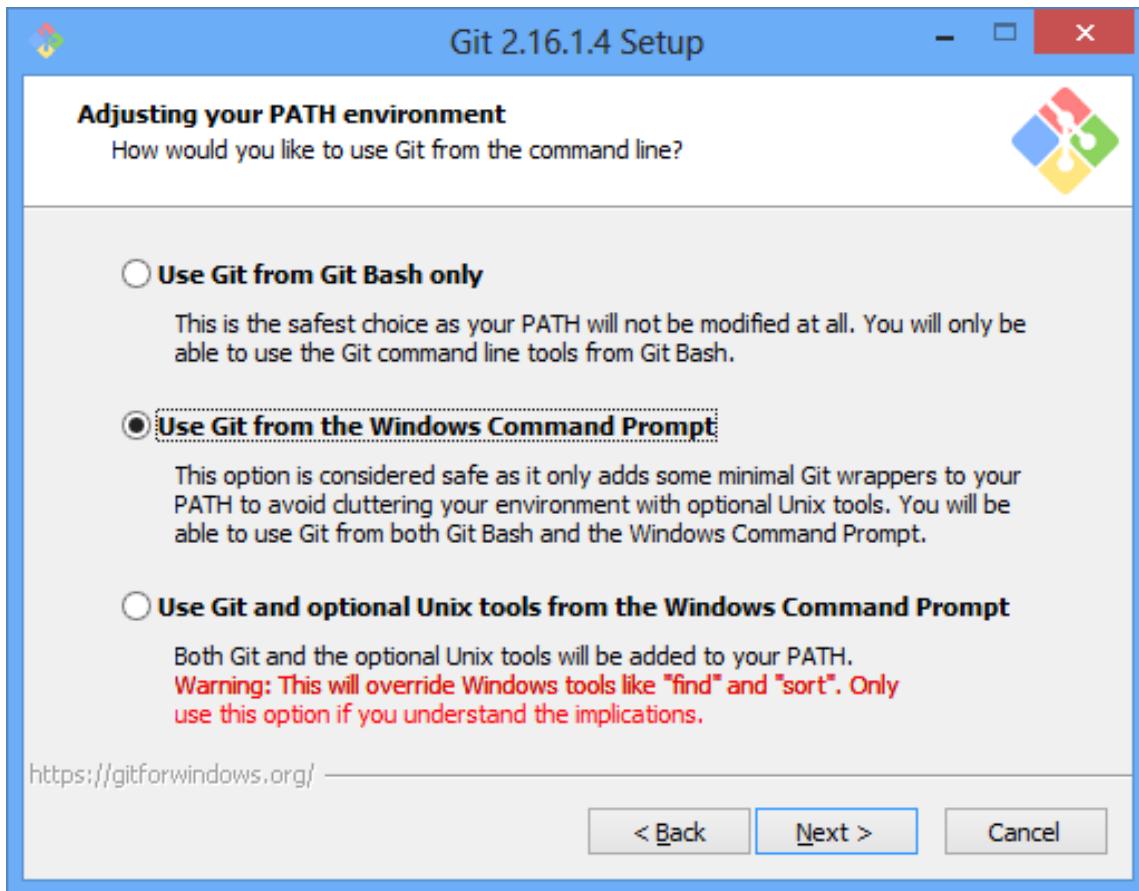
Gambar 1.11: Lokasi di Start Menu

5. Setelah itu kita diminta untuk memilih *editor* yang digunakan oleh Git seperti pada gambar 1.12 berikut :



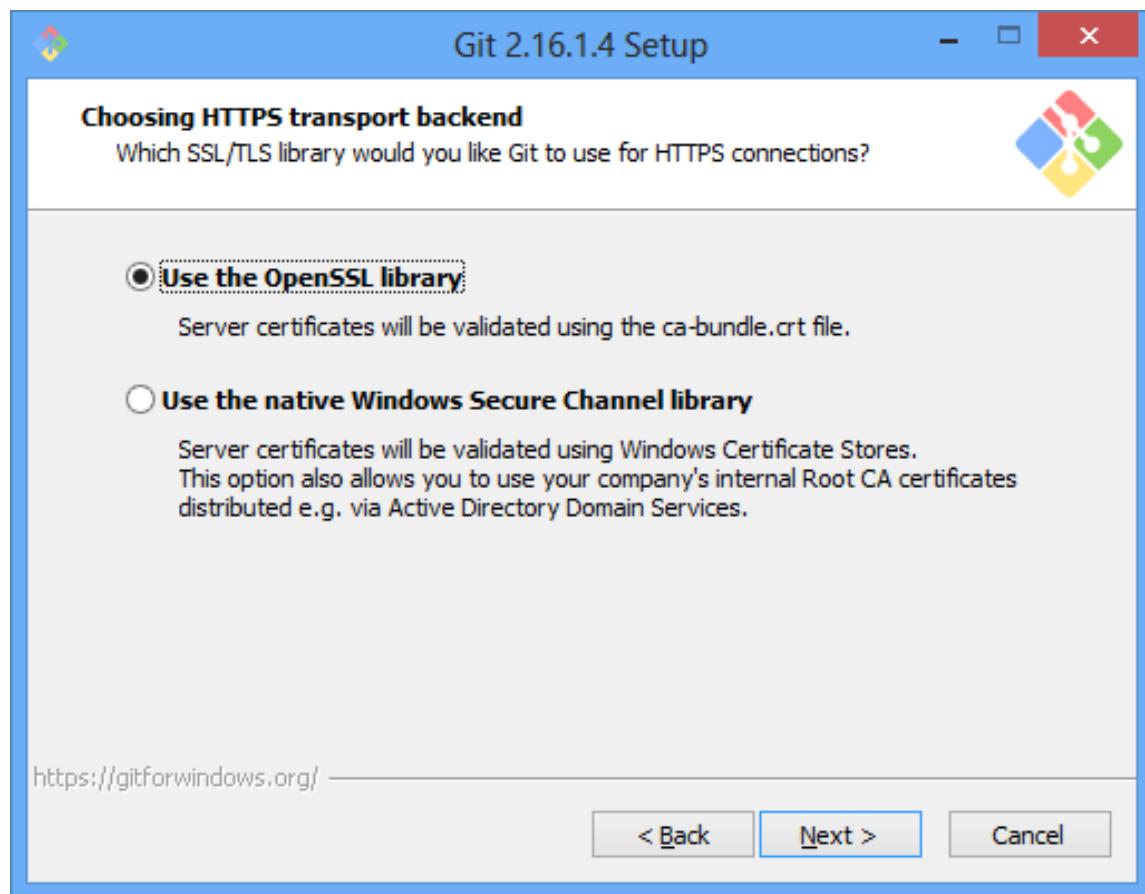
Gambar 1.12: Pemilihan *Editor*

6. Langkah berikutnya adalah mengatur *environment variable* PATH untuk penggunaan Git melalui *console* atau *command prompt*, tampilannya adalah seperti pada gambar 1.13 berikut :



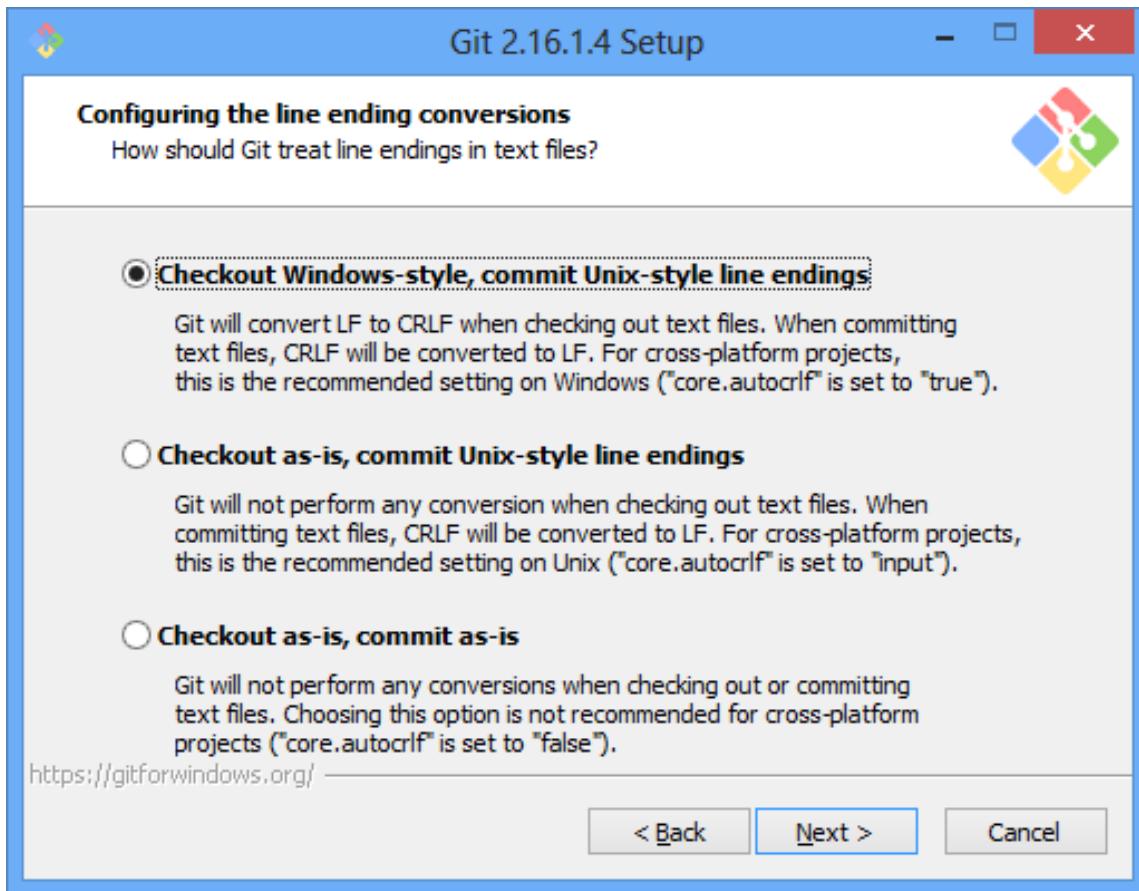
Gambar 1.13: Pengaturan PATH

7. Selanjutnya adalah memilih pustaka SSL yang akan digunakan untuk koneksi HTTPS, tampilannya adalah seperti pada gambar 1.14 berikut :



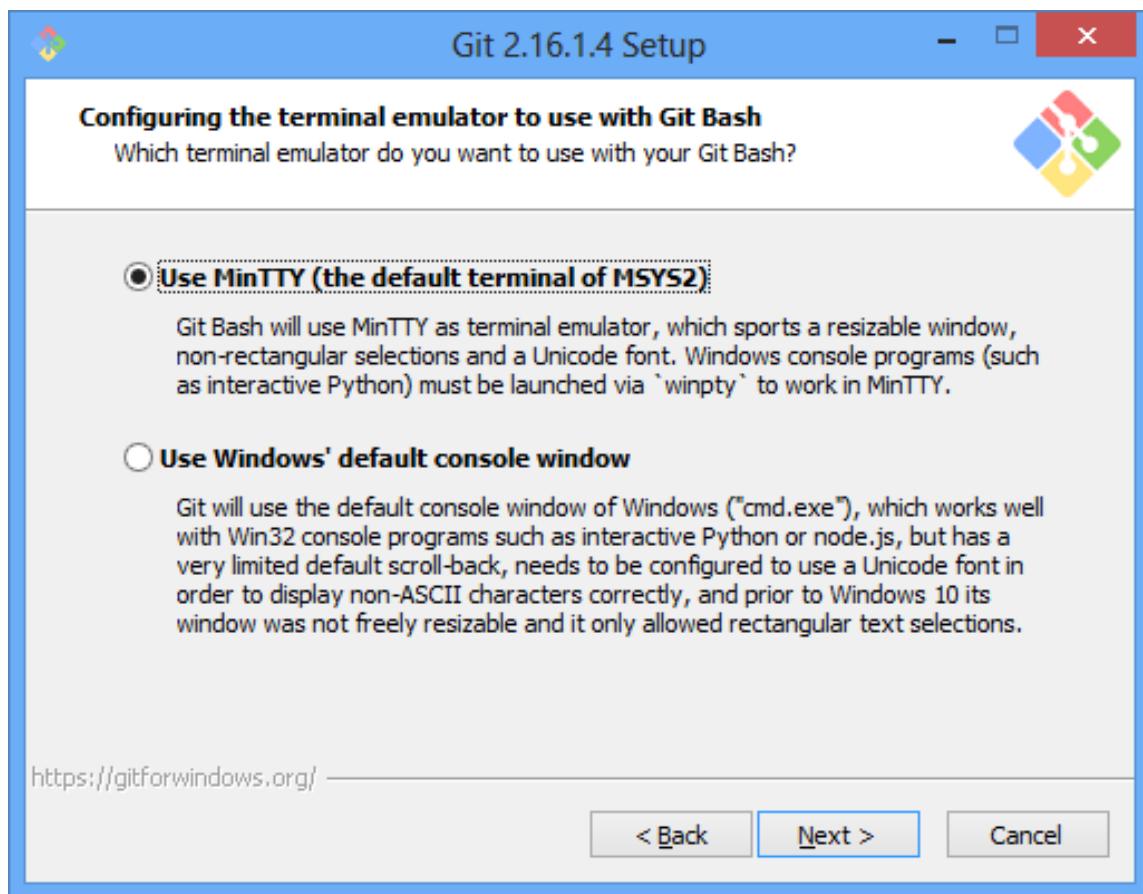
Gambar 1.14: Pemilihan Pustaka SSL

8. Berikutnya adalah memilih bagaimana cara Git memperlakukan *line endings*, tampilannya adalah seperti pada gambar 1.15 berikut :



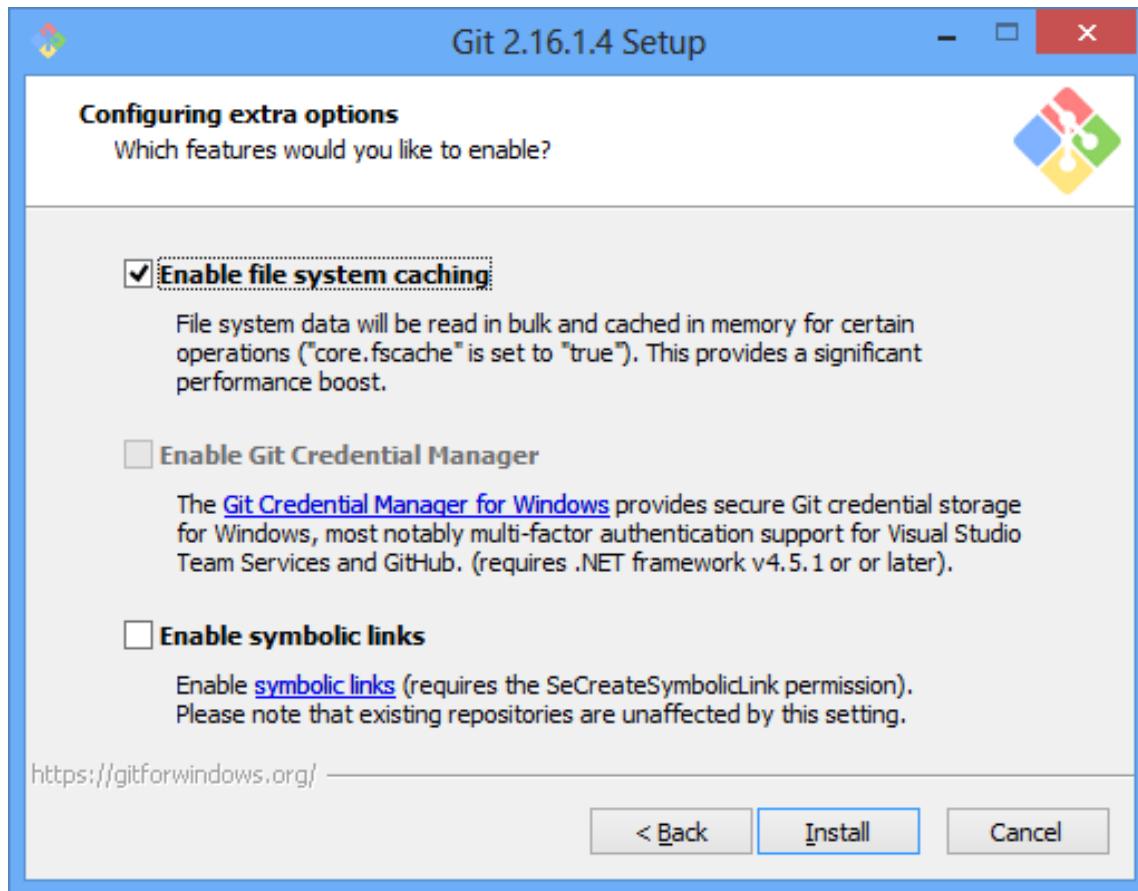
Gambar 1.15: Pemilihan Konversi *Line Ending*

9. Selanjutnya adalah pemilihan emulator terminal yang akan digunakan oleh Git, tampilannya seperti pada gambar 1.16 berikut :



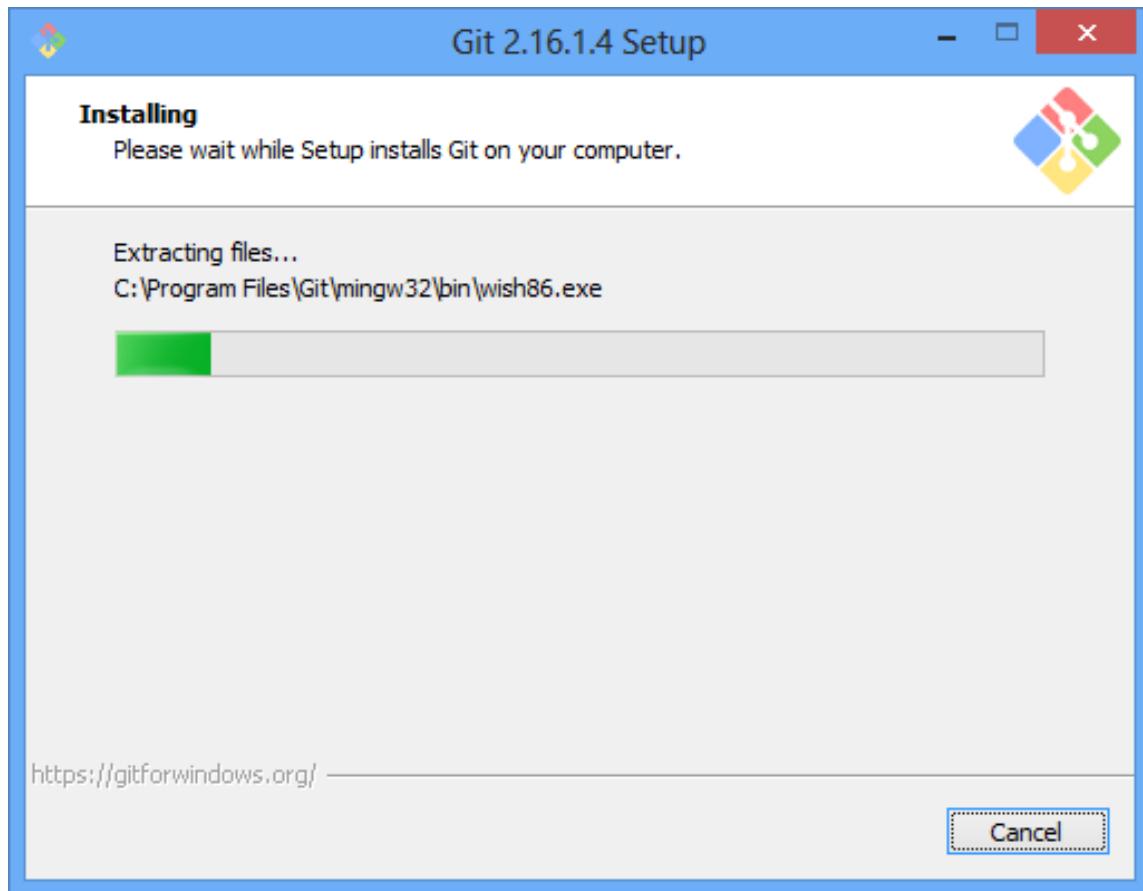
Gambar 1.16: Pemilihan Emulator Terminal

10. Langkah berikutnya adalah memilih fasilitas yang akan diaktifkan, tampilannya akan terlihat seperti pada gambar 1.17 berikut :



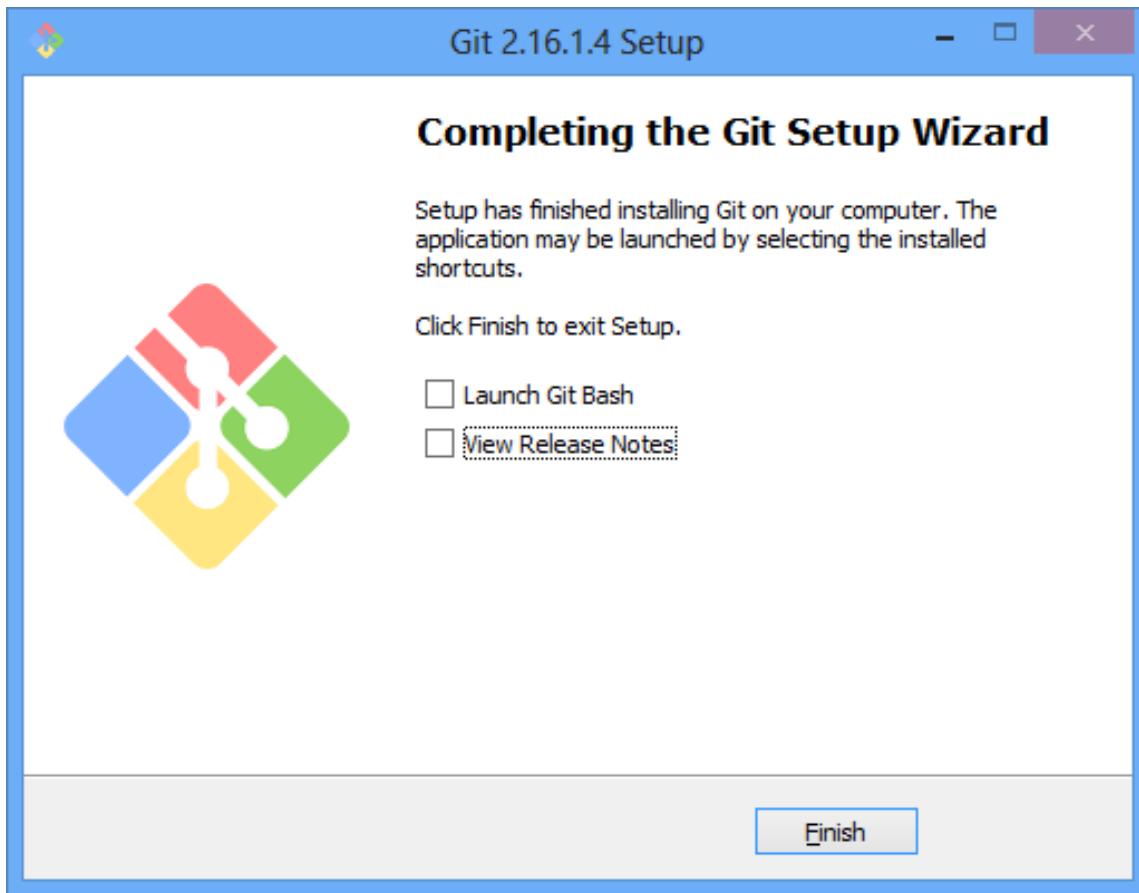
Gambar 1.17: Pemilihan Fasilitas Tambahan

11. Setelah menekan tombol **Install**, maka proses instalasi akan berjalan seperti pada gambar 1.18 berikut :



Gambar 1.18: Proses Instalasi

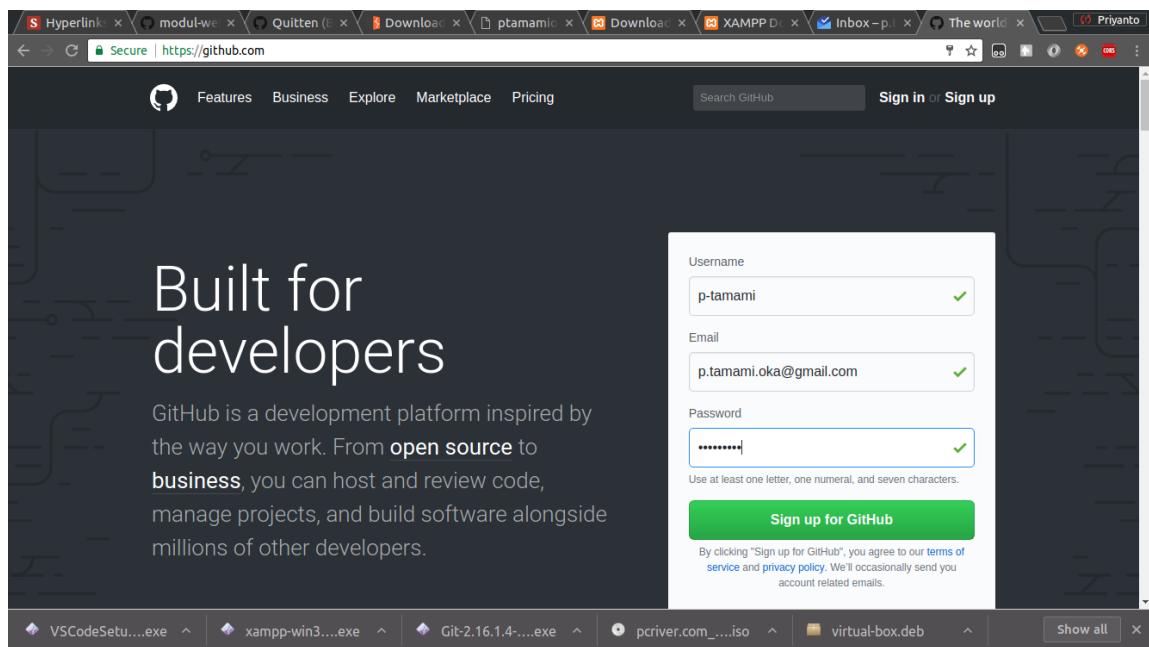
12. Setelah proses instalasi selesai, maka akan muncul jendela seperti pada gambar 1.19 berikut :



Gambar 1.19: Instalasi Selesai

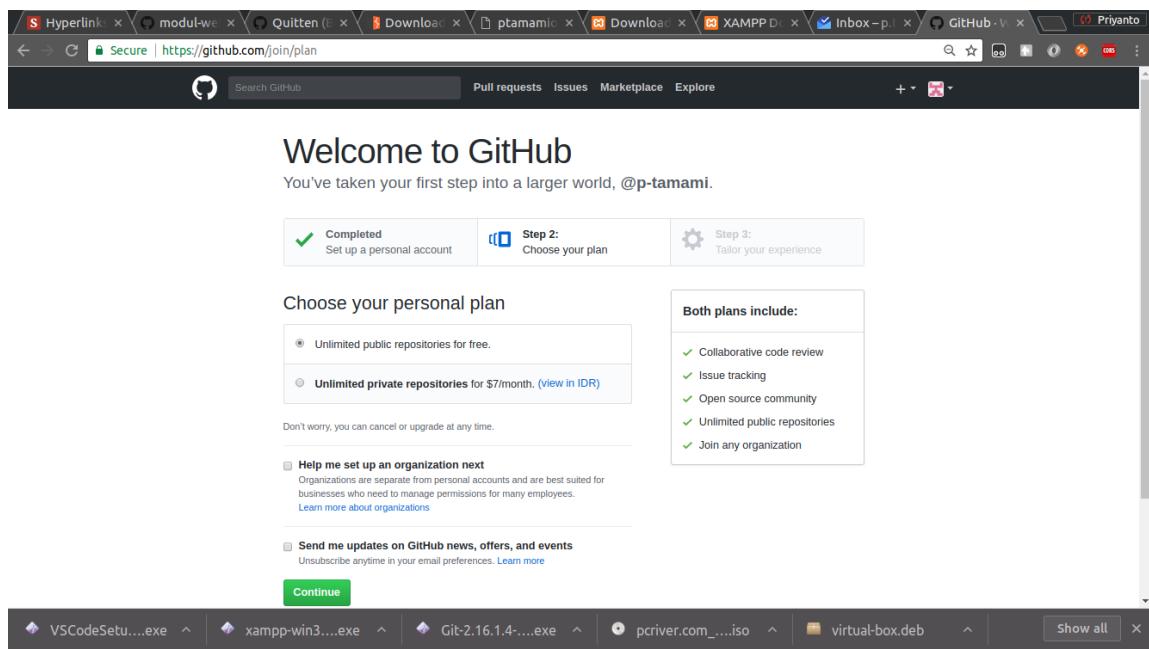
Berikutnya kita akan bahas bagaimana caranya membuat akun di Github agar kita dapat menyimpan kode yang telah kita bangun di repositori Github. Langkah-langkahnya adalah sebagai berikut :

1. Bukalah *browser* dan isikan alamatnya dengan **github.com**, sehingga nanti akan muncul tampilan seperti pada gambar 1.20 berikut :



Gambar 1.20: Halaman Depan Github

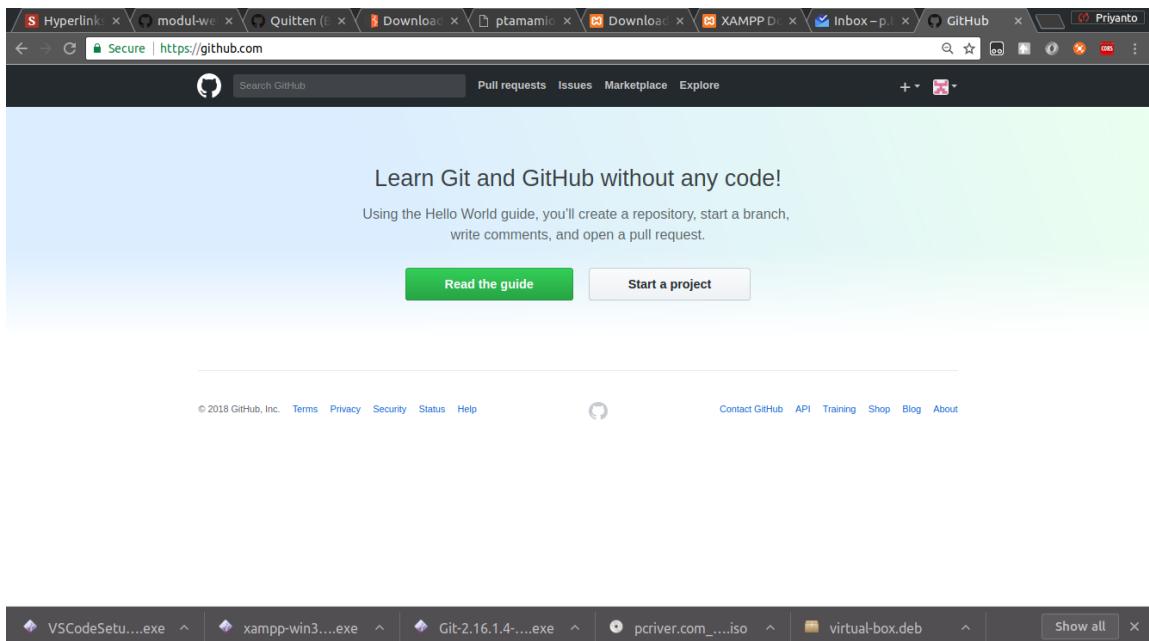
- Setelah mengisikan *username*, alamat *email*, dan *password*, kemudian menekan tombol **Sign up for Github**, maka akan tampilan halaman berikutnya seperti pada gambar 1.21 berikut :



Gambar 1.21: Halaman Selamat Datang Github

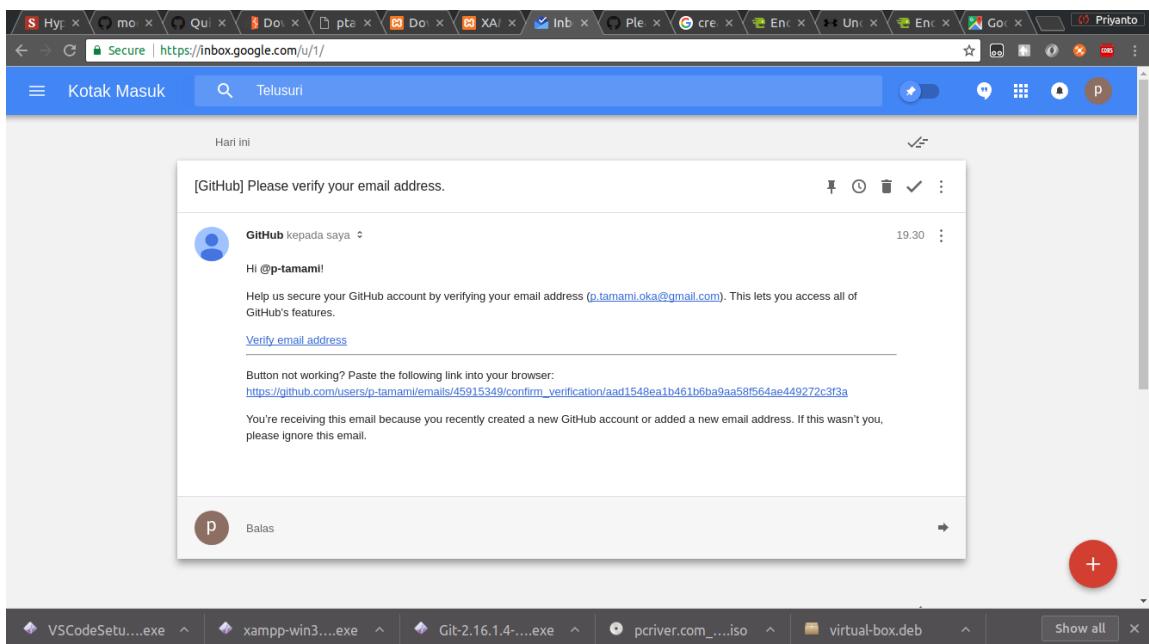
- Karena kita menggunakan versi gratis, jadi lanjutkan saja dengan menekan tombol **Continue** sehingga muncul tampilan berikutnya seperti pada gambar

1.22 :



Gambar 1.22: Halaman Pembuka Github

- Selanjutnya, untuk membuat repositori, kita diwajibkan melakukan verifikasi alamat *email*, bukalah *email* yang kita gunakan untuk mendaftar Github, lalu klik *link* yang bertuliskan *Verify email address* seperti pada gambar 1.23 berikut :

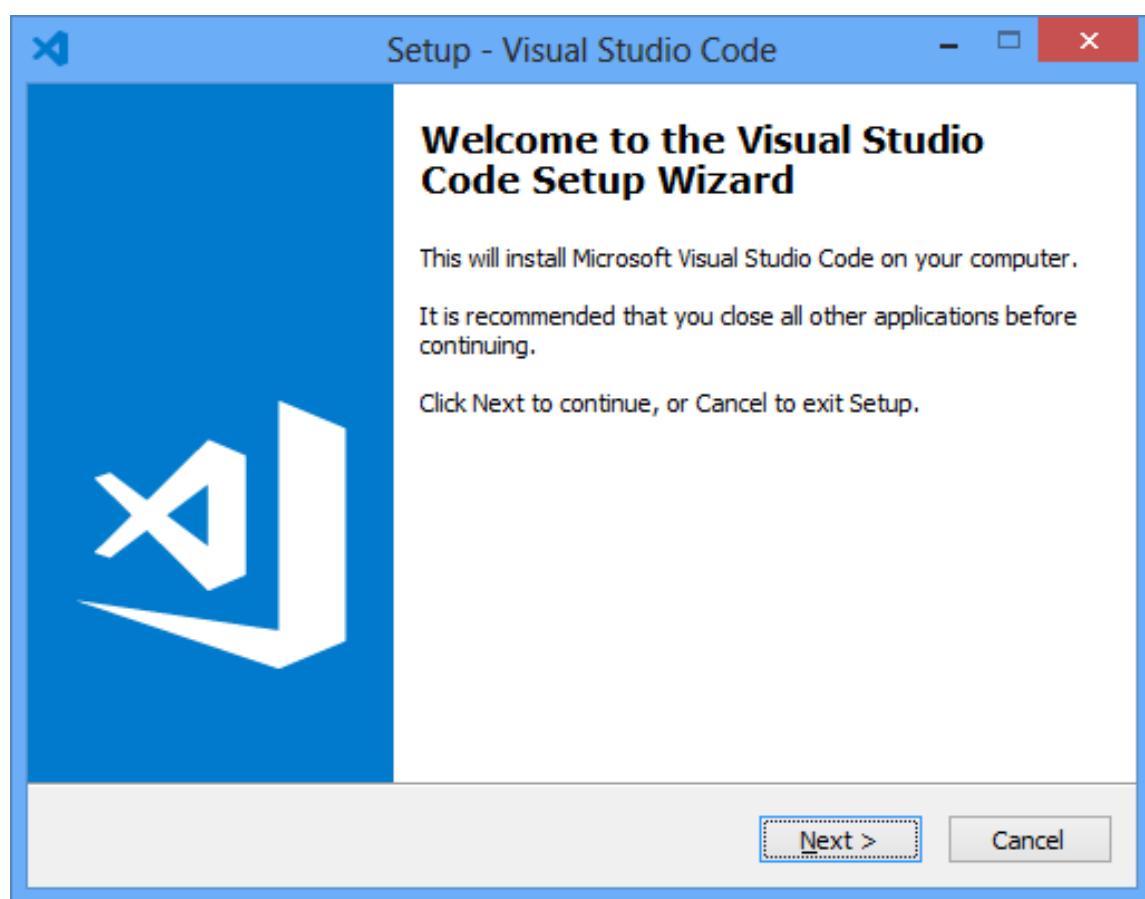
Gambar 1.23: Verifikasi Alamat *Email*

Sampai langkah ini Github sudah dapat kita gunakan untuk menyimpan kode program yang kita bangun.

1.3.3 Instalasi Visual Studio Code

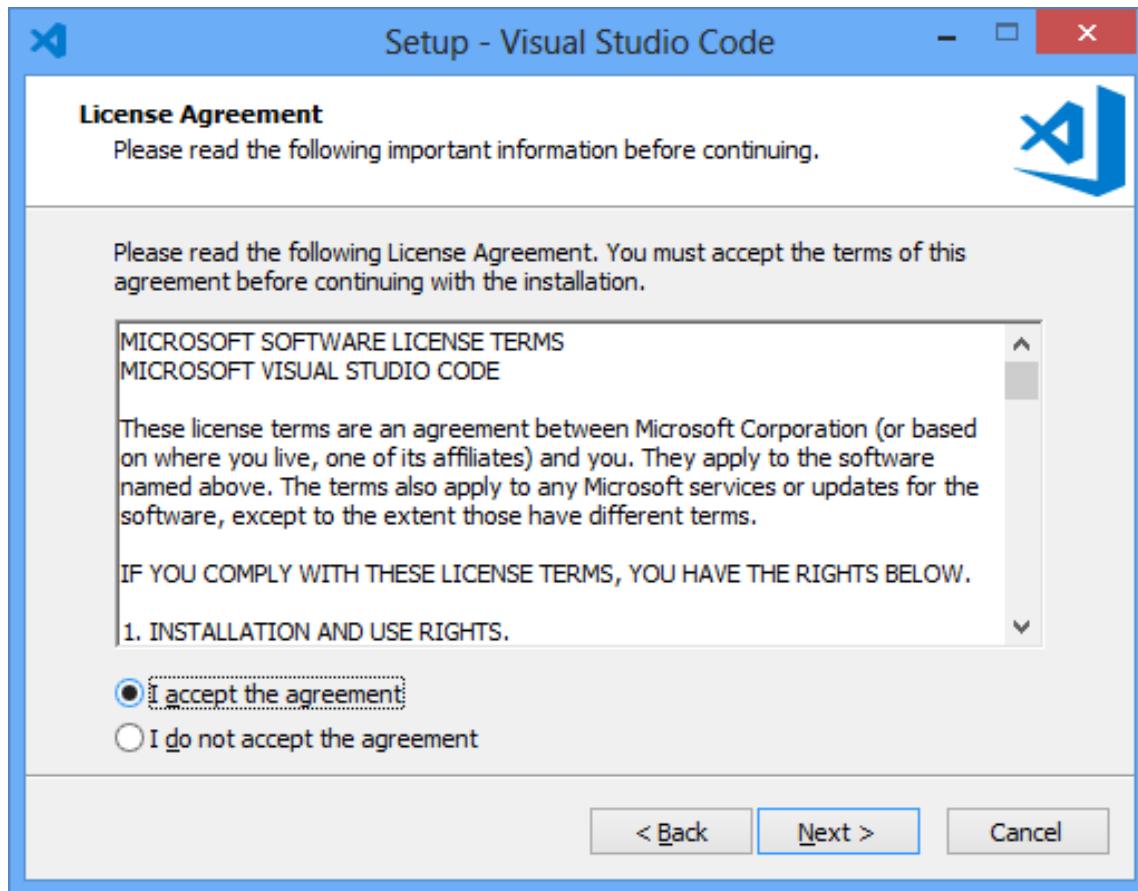
Proses instalasi Visual Studio Code adalah sebagai berikut :

1. Pertama kita akan disajikan jendela selamat datang seperti biasa, ditunjukkan oleh gambar 1.24 berikut :



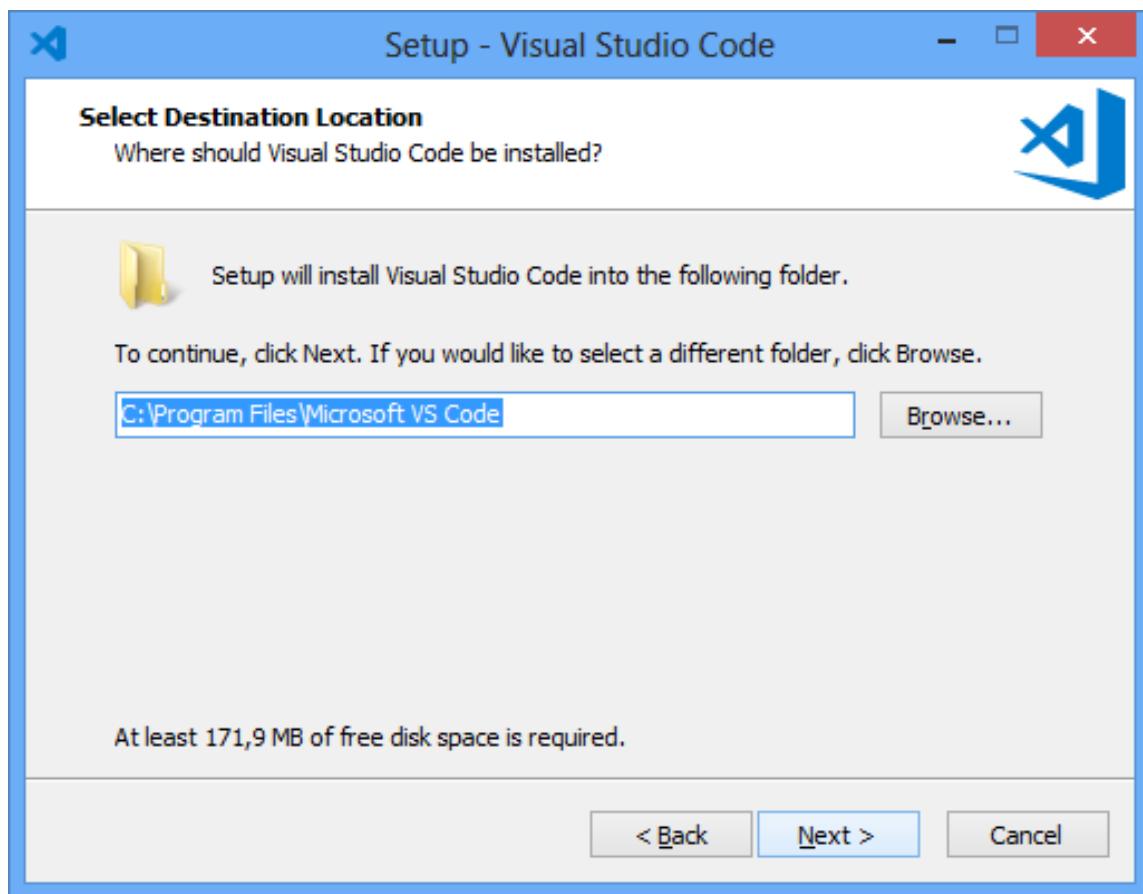
Gambar 1.24: Halaman Selamat Datang dari Visual Studio Code

2. Jendela berikutnya akan menampilkan informasi mengenai lisensi yang digunakan, tampilannya akan terlihat seperti gambar 1.25 berikut :



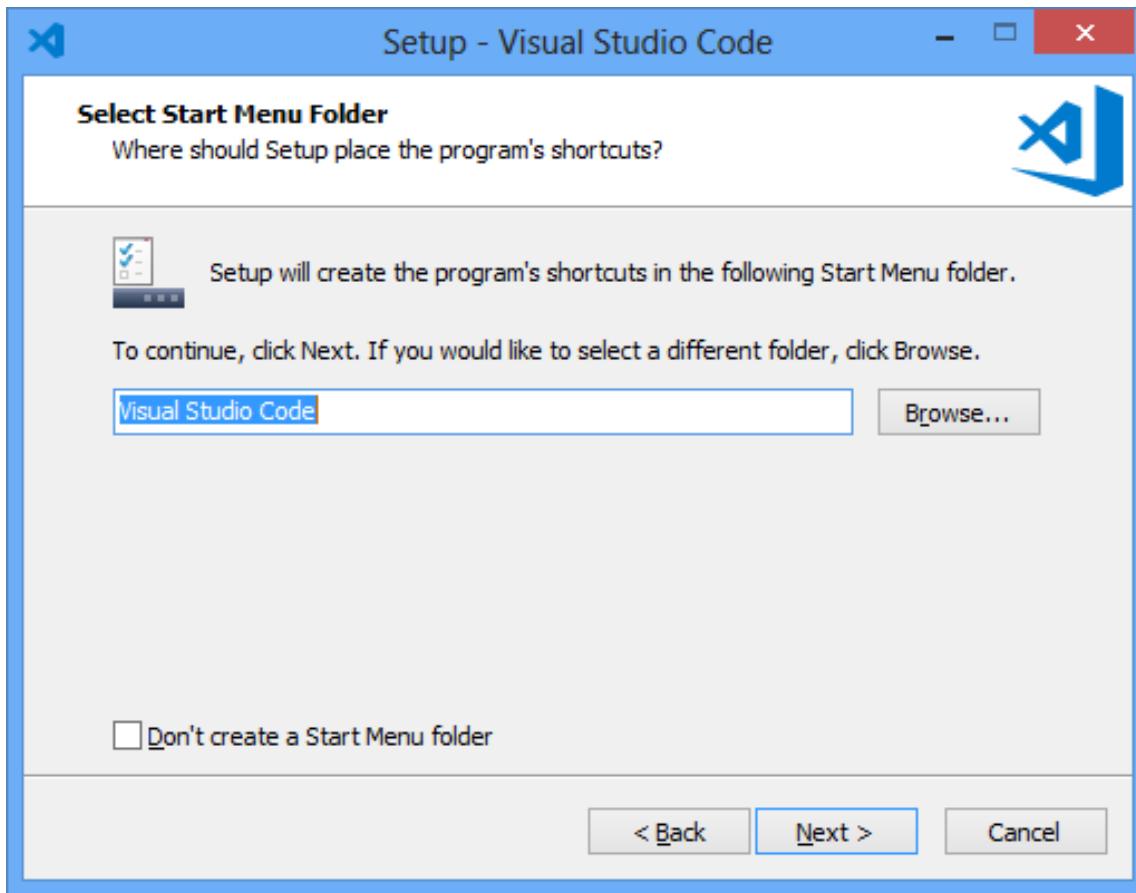
Gambar 1.25: Informasi Licensi

3. Selanjutnya adalah menentukan lokasi direktori dimana *Visual Studio Code* akan ditempatkan, tampilannya adalah seperti pada gambar 1.26 berikut ini :



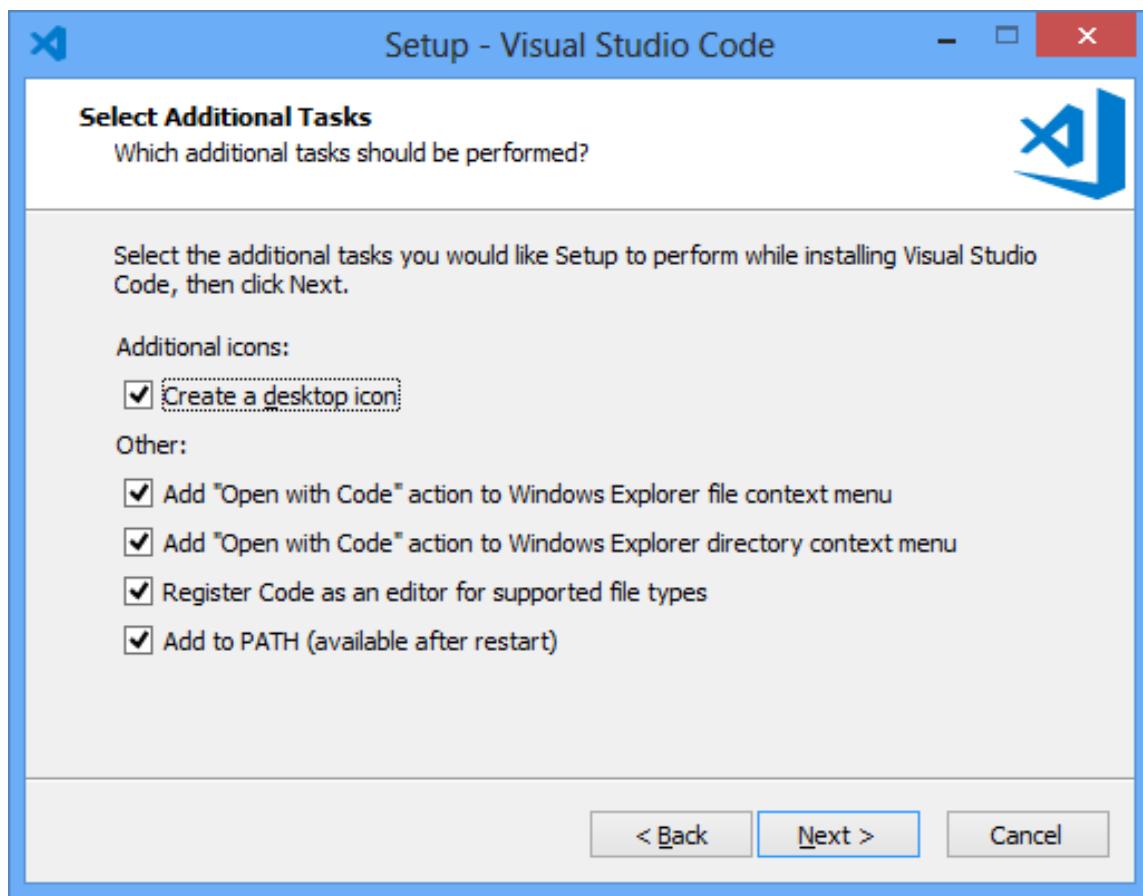
Gambar 1.26: Lokasi Direktori Instalasi

4. Berikutnya adalah menentukan lokasi direktori pada *start menu*, tampilannya akan terlihat seperti pada gambar 1.27 berikut :



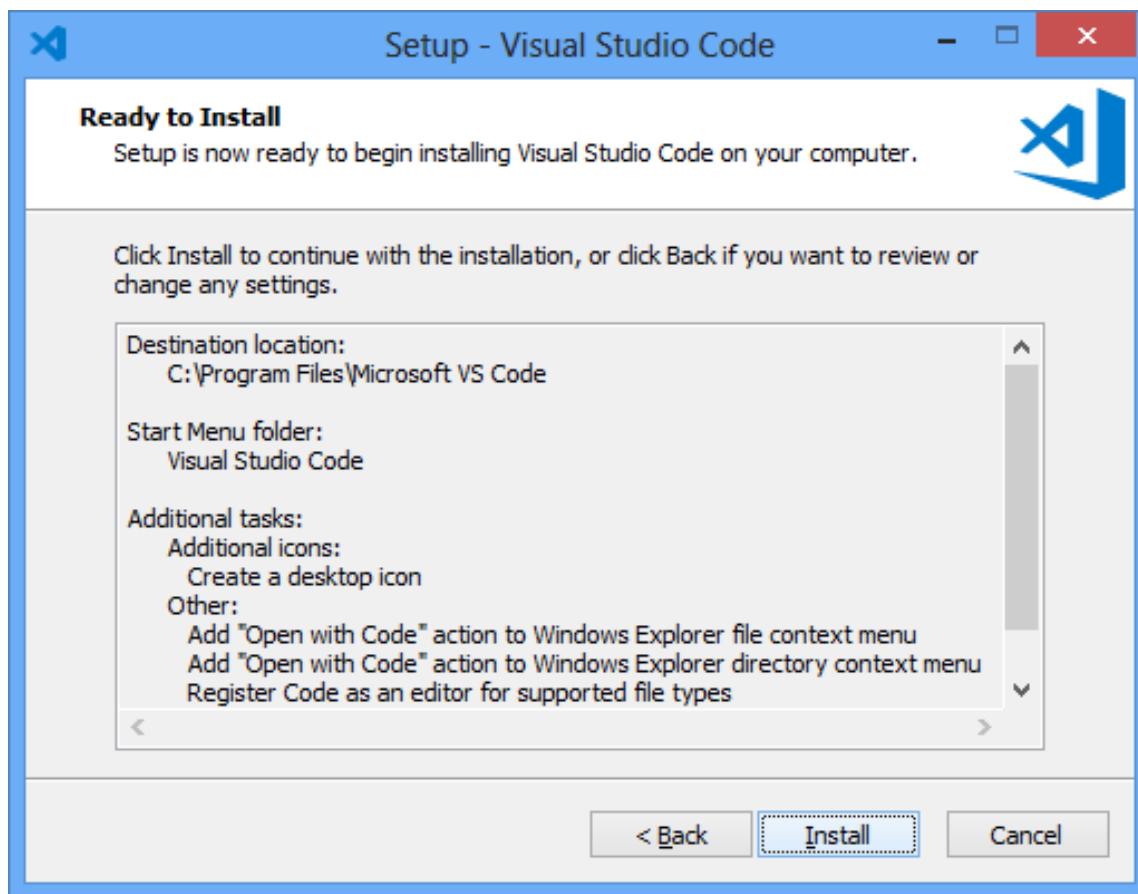
Gambar 1.27: Lokasi Pada *Start Menu*

5. Jendela berikutnya menampilkan pilihan agar pada saat instalasi dilakukan beberapa konfigurasi, tampilannya akan terlihat seperti pada gambar 1.28 berikut :



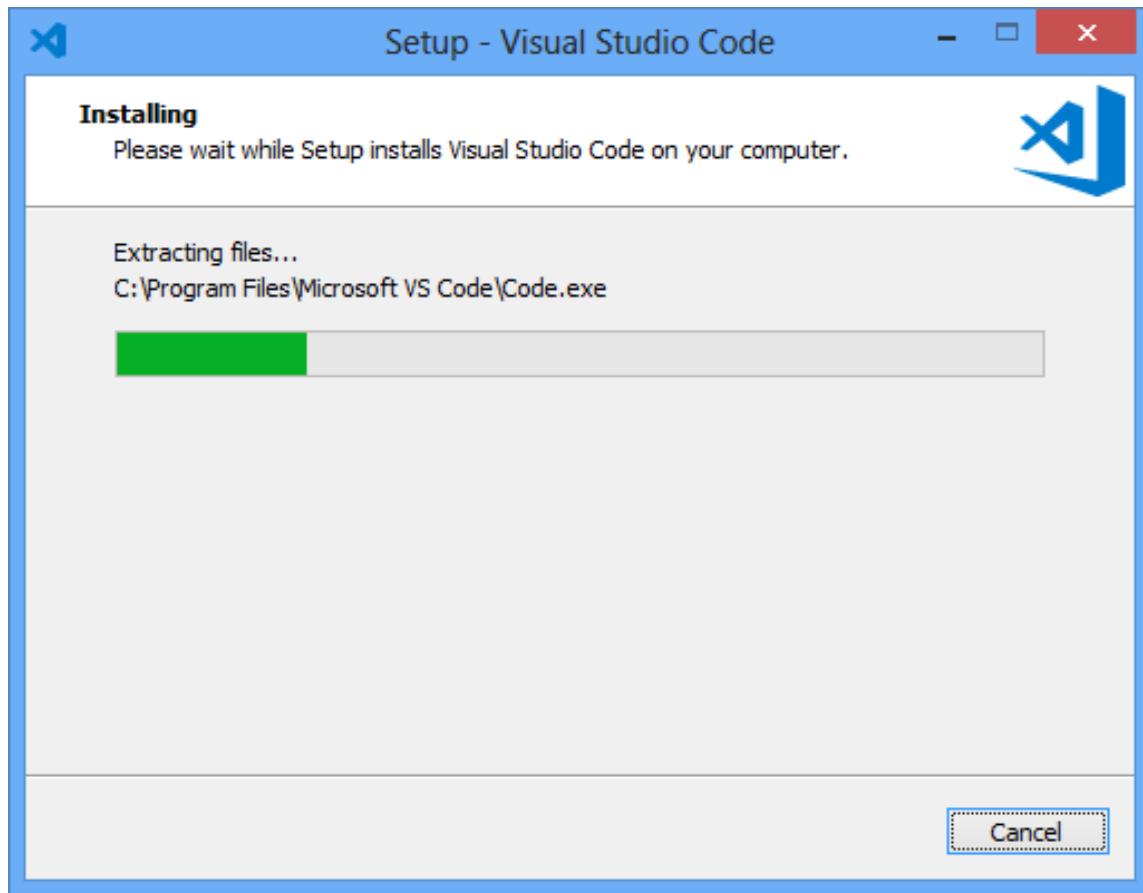
Gambar 1.28: Konfigurasi Tambahan

6. Langkah berikutnya, jendela instalasi akan menampilkan ringkasan dan proses instalasi siap dijalankan, tampilannya akan terlihat seperti gambar 1.29 berikut :



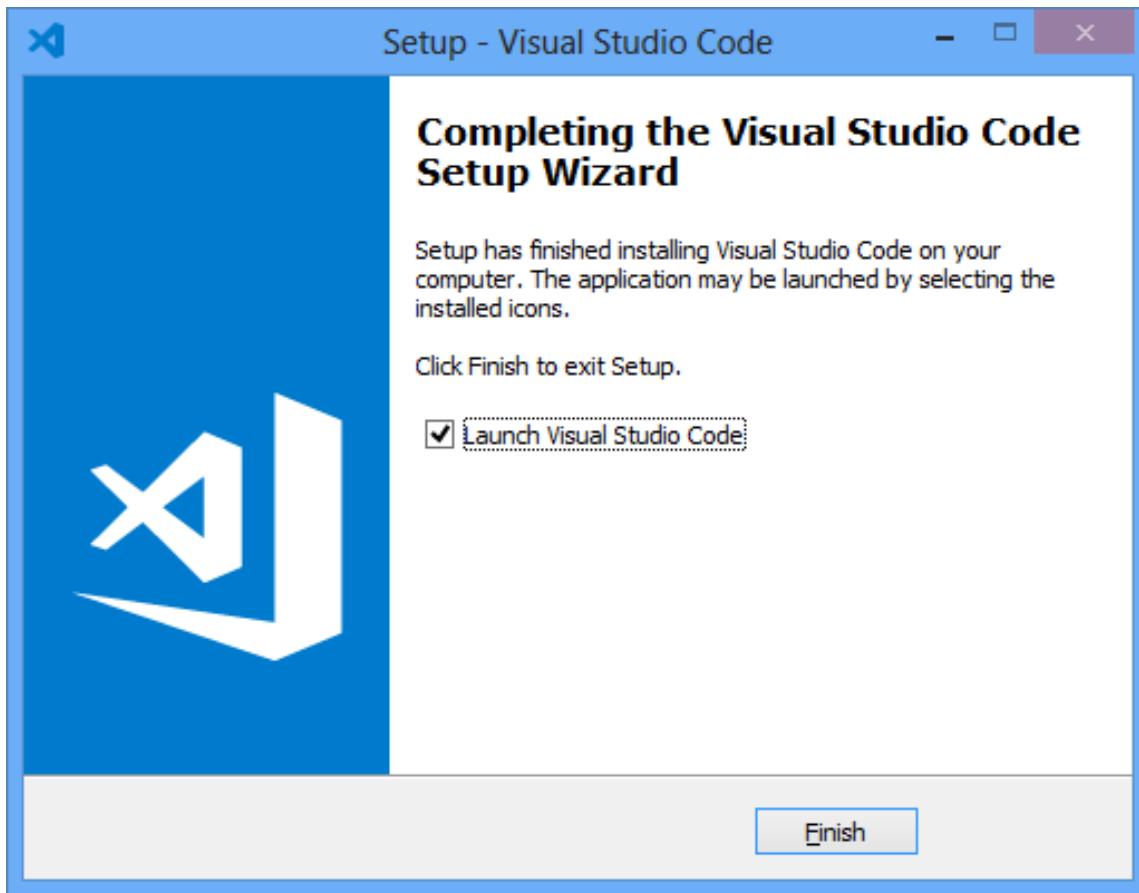
Gambar 1.29: Proses Instalasi Telah Siap

7. Jendela berikutnya akan menampilkan status proses instalasi, tampilannya akan terlihat seperti pada gambar 1.30 berikut :



Gambar 1.30: Proses Instalasi Berjalan

8. Terakhir adalah informasi bahwa proses instalasi telah selesai dan berhasil, tampilannya akan terlihat seperti pada gambar 1.31 berikut :

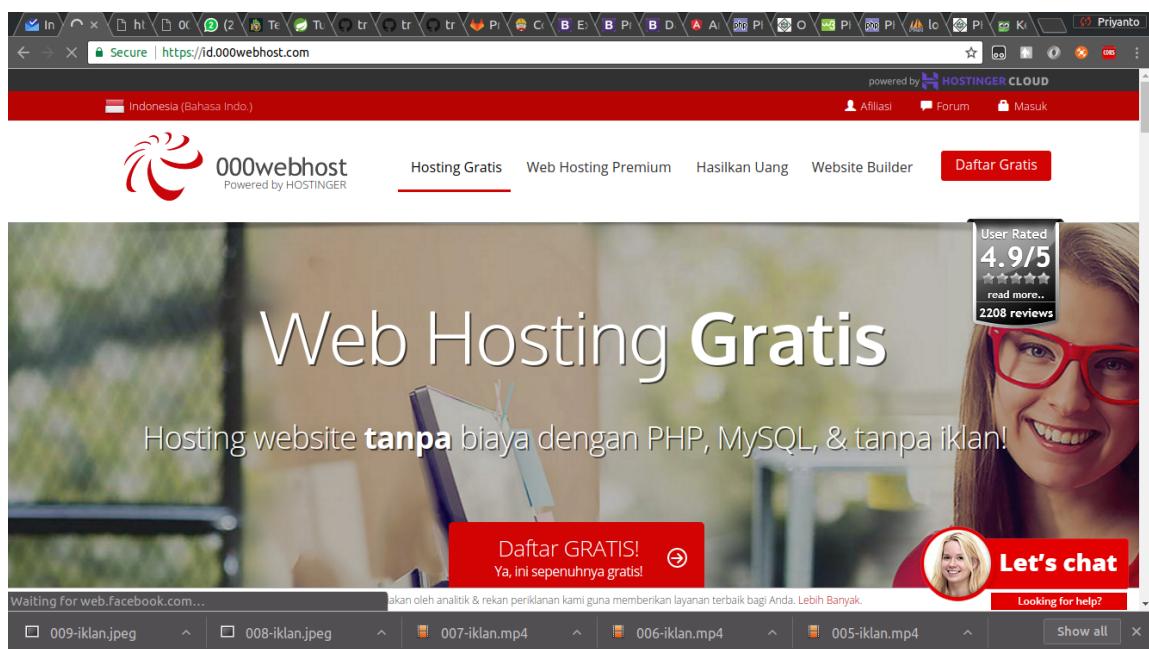


Gambar 1.31: Proses Instalasi Selesai

1.3.4 Registrasi 000webhost.com

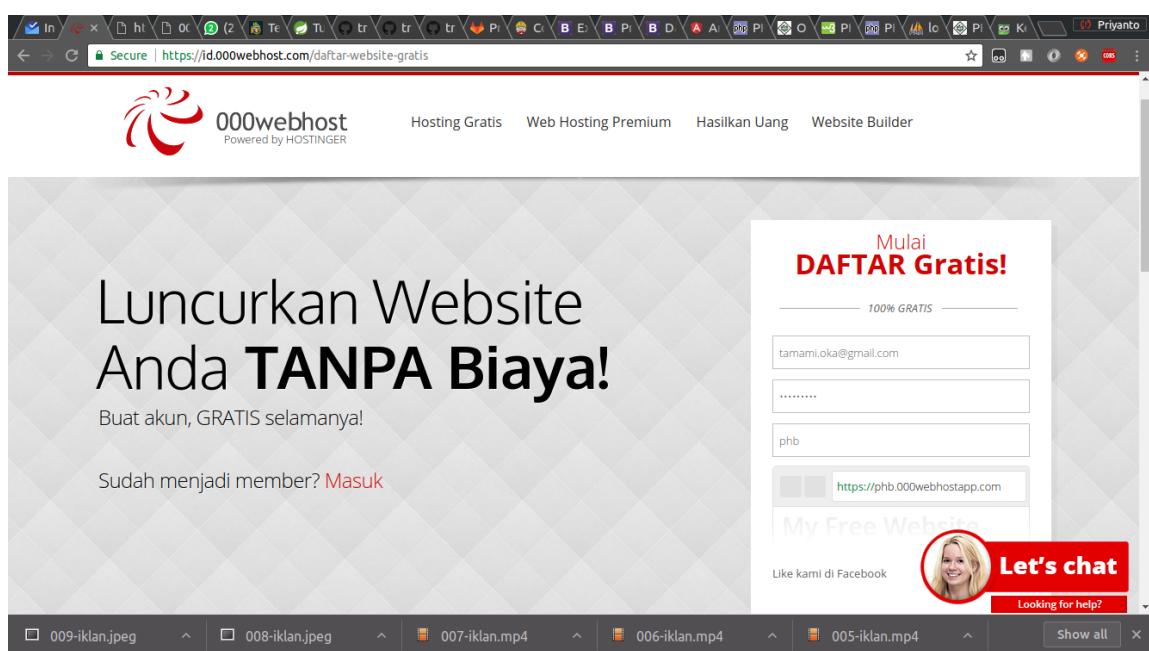
Registrasi pada website www.000webhost.com cukup mudah, langkahnya adalah sebagai berikut :

1. Mengunjungi website www.000webhost.com, apabila ingin pilihan bahasa Indonesia, kita dapat mengubahnya di bagian kiri atas, atau mengunjungi alamat id.000webhost.com, tampilan awal dari website ini seperti terlihat pada gambar 1.32 berikut :



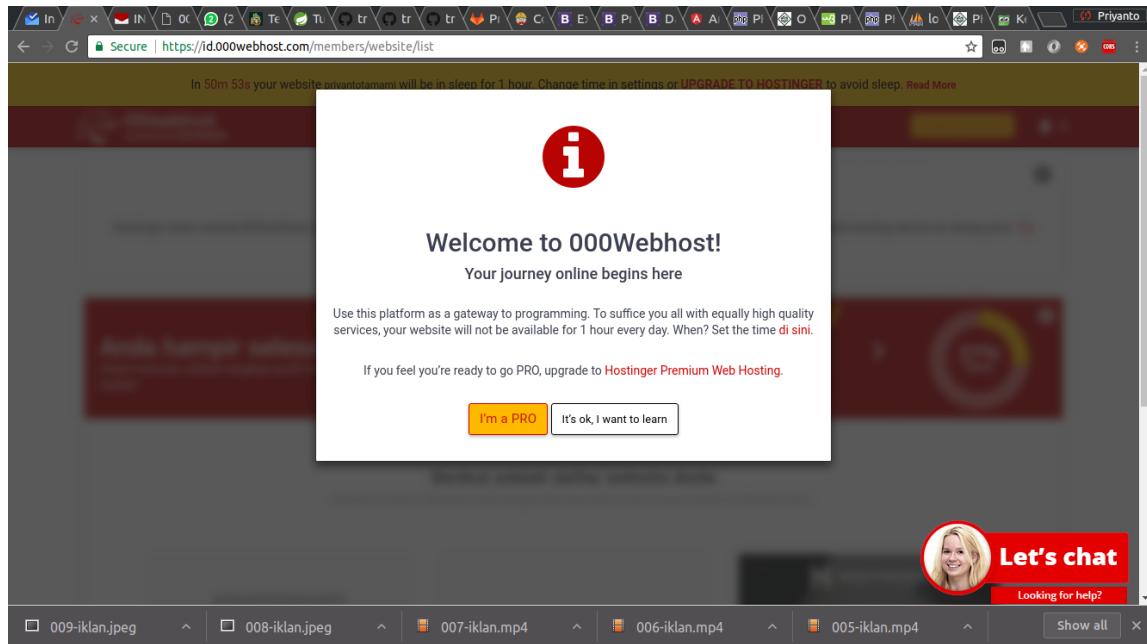
Gambar 1.32: Tampilan Awal Website id.000webhost.com

- Setelah menekan tombol "Daftar Gratis", maka kemudian akan disajikan halaman seperti pada gambar 1.33, kita perlu mengisikan alamat email, *password* untuk masuk ke halaman manajemen *website* yang kita bangun, serta alamat dari *website* yang kita inginkan. Karena sifatnya gratis, *url* yang disediakan pun mengikuti aturan dari penyedia *hosting*.



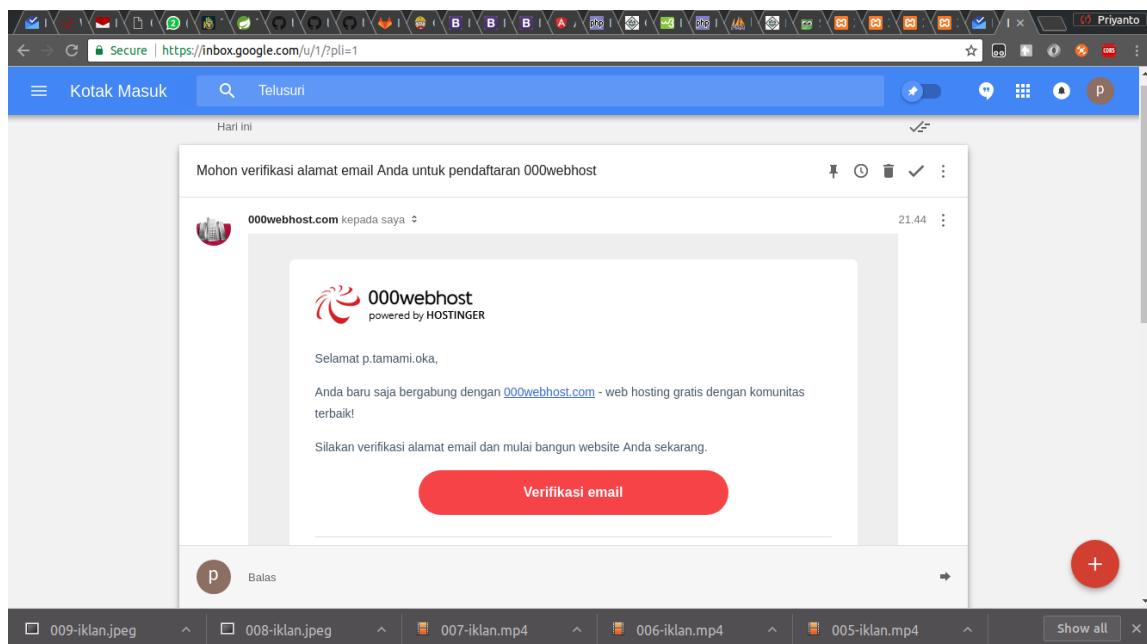
Gambar 1.33: Halaman Registrasi

3. Setelah menekan tombol pendaftaran, maka akan ditampilkan halaman selamat datang seperti pada gambar 1.34.



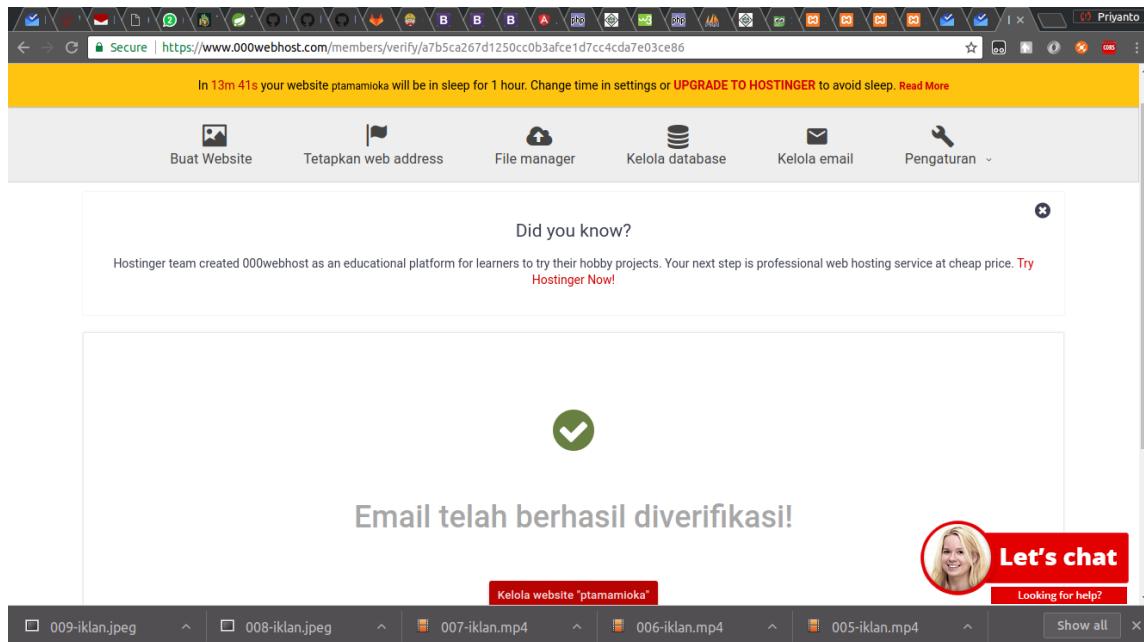
Gambar 1.34: Halaman Selamat Datang

4. Langkah berikutnya adalah melakukan verifikasi surel (*e-mail*) seperti pada gambar 1.35.



Gambar 1.35: Verifikasi Surel

5. Apabila verifikasi surel berhasil, maka akan tampil halaman seperti pada gambar 1.36.



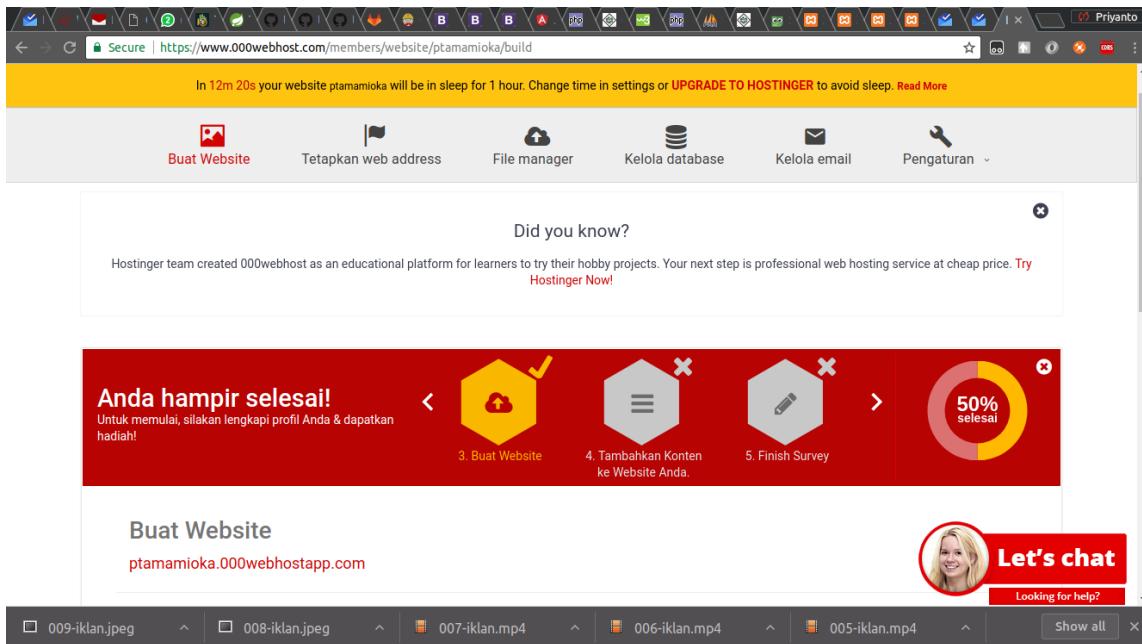
Gambar 1.36: Hasil verifikasi surel

Sampai sini, tahapan pendaftaran anggota telah berhasil kita lakukan.

1.3.5 Aplikasi Selamat Datang

Kita akan mencoba melakukan *publish* terhadap sebuah *file html* untuk membuktikan bahwa halaman *website* yang kita unggah ke id.000webhost.com dapat berhasil. Berikut adalah langkah-langkahnya :

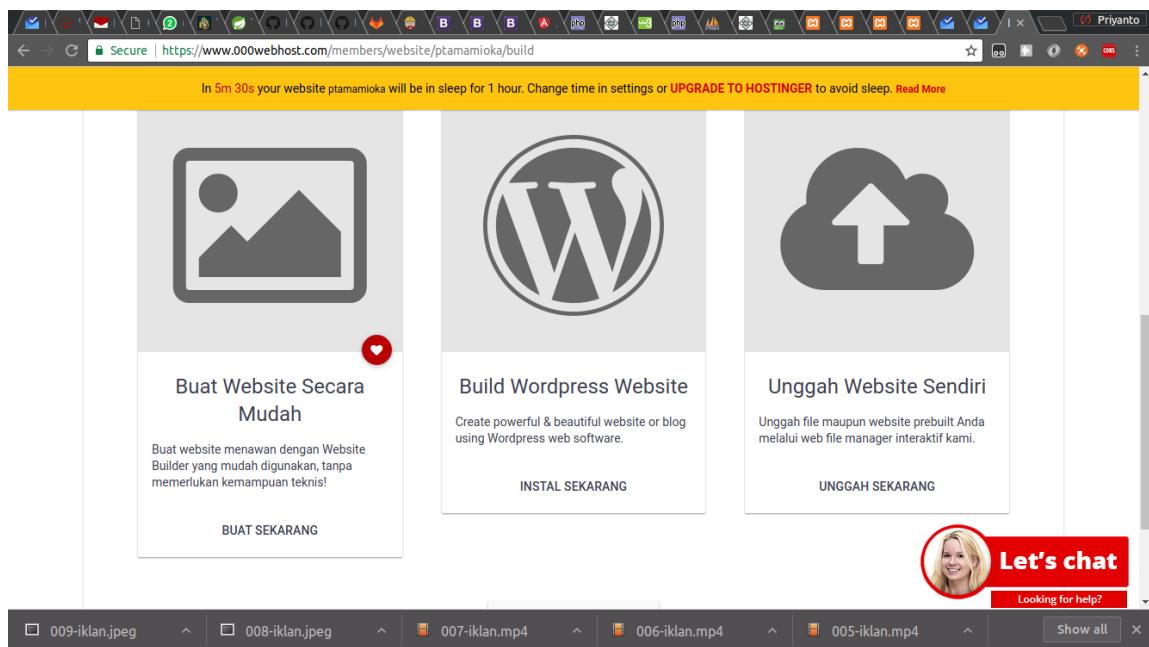
1. Langkah pertama adalah membuat sebuah halaman *website* dengan menekan tombol kiri atas sehingga tampil halaman seperti pada gambar 1.37.

Gambar 1.37: Membuat *website*

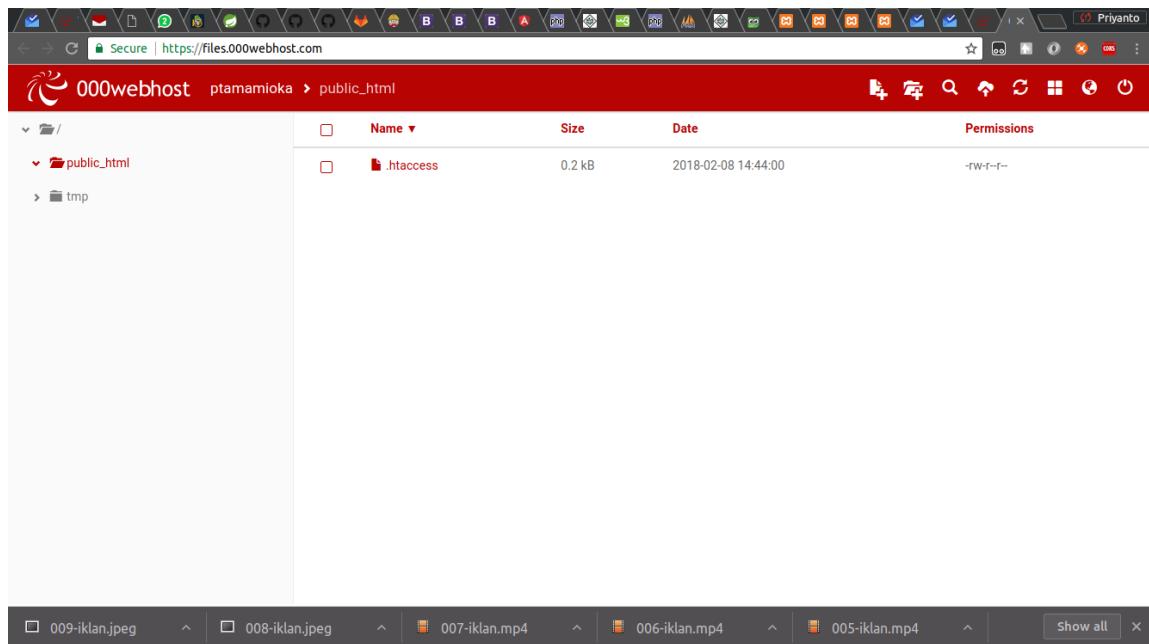
2. Selanjutnya kita membuat *file* dengan nama `index.html` terlebih dahulu, pembuatan *file* ini dapat kita lakukan dengan notepad, vim, atau Visual Studio Code yang telah kita *install*. Isi dari *file* ini adalah sebagai berikut :

`1 Hai , selamat datang .`

3. Kemudian kita *scroll* ke bawah halaman `id.000webhost.com`, maka akan muncul tampilan seperti pada gambar 1.38. Karena kita telah membuat sebuah *file* `index.html`, maka kita memilih "Unggah Website Sendiri".

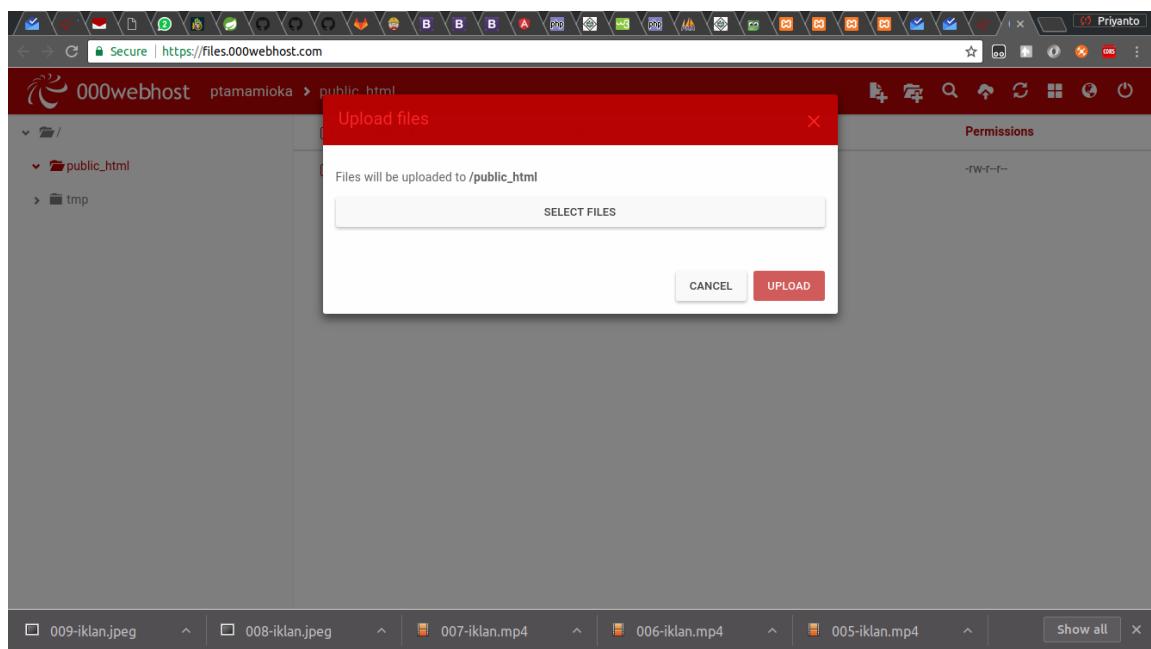
Gambar 1.38: Pilihan cara membuat *website*

4. Selanjutnya kita akan ditunjukkan halaman *file manager* dimana nantinya di dalam *folder public_html* ini *project* kita ditempatkan. Tampilannya seperti pada gambar 1.39.

Gambar 1.39: Halaman *File Manager*

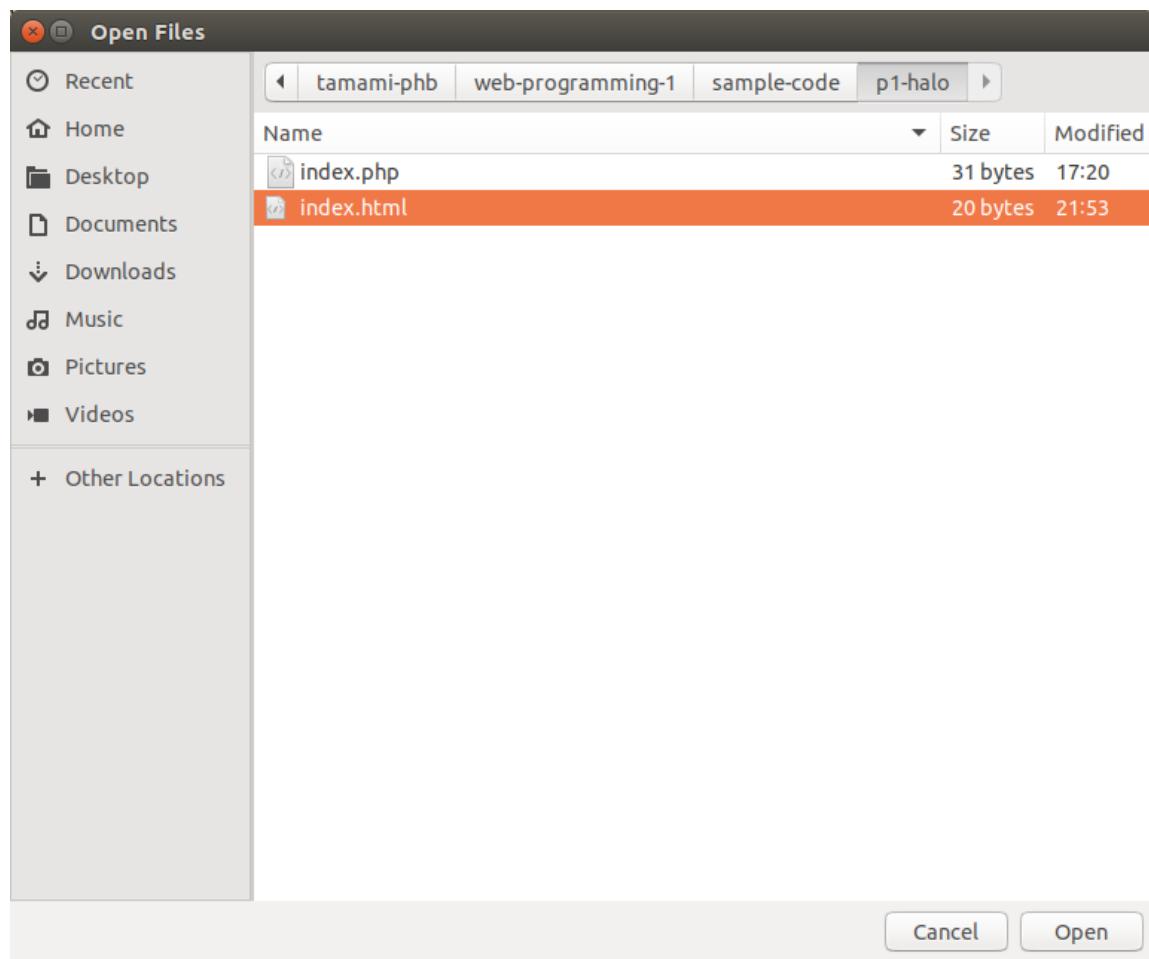
5. Pilihlah ikon dengan gambar awan dan tanda panah atas pada bagian atas kanan jendela *file manager* untuk mengunggah *file index.html* yang telah

kita buat sebelumnya. Nantinya akan muncul tampilan seperti pada gambar 1.40.



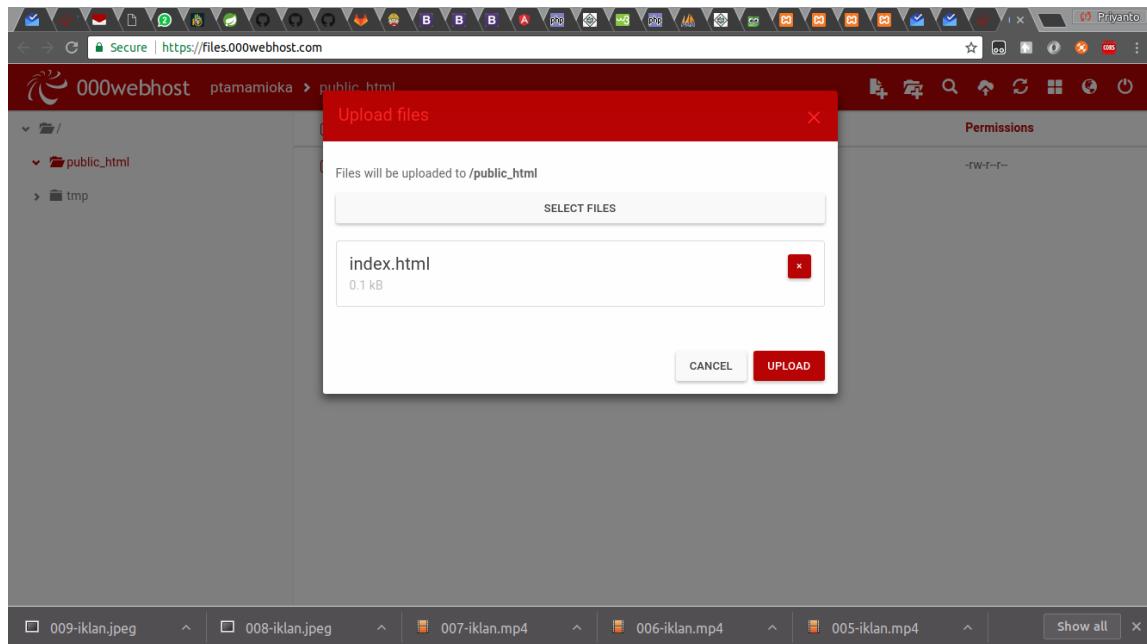
Gambar 1.40: Jendela Unggah *File*

6. Tekanlah tombol **SELECT FILES** yang berada di tengah sehingga muncul jendela pemilihan *file* seperti pada gambar 1.41, lalu memilih *file* dengan nama **index.html** yang akan kita unggah ke *server*.

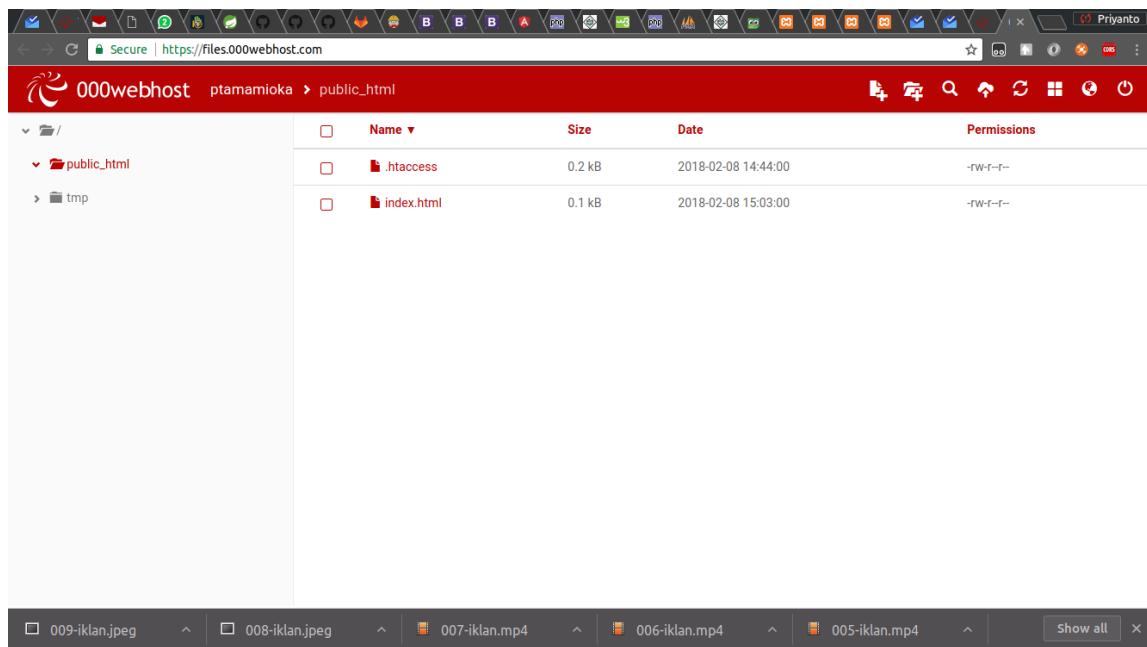


Gambar 1.41: Jendela Pemilihan *File*

7. Setelah *file* dipilih, maka jendela *Upload Files* akan menampilkan *file* yang terpilih untuk selanjutnya siap diunggah seperti pada gambar 1.42.

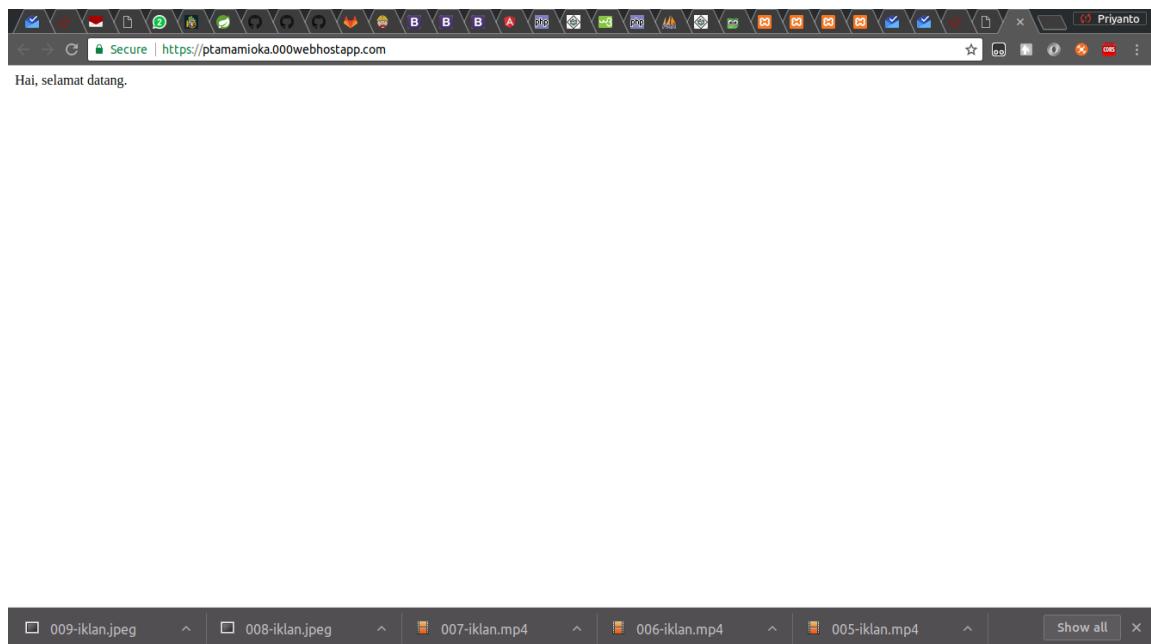
Gambar 1.42: Jendela *File Terpilih*

8. Setelah selesai terunggah, maka akan tampil *file* yang telah diunggah seperti pada gambar 1.43.

Gambar 1.43: Jendela *File Manager* Setelah File Terunggah

9. Lakukan akses ke halaman yang telah disediakan oleh www.000webhost.com yang hasilnya seperti terlihat pada gambar 1.44. Karena ini layanan gratis, jadi jeda antara waktu *file* telah terunggah dengan hasil *website* memakan

waktu yang bervariasi. Apabila protokol `https` tidak berhasil menampilkan halaman yang telah kita buat, cobalah menggunakan protokol `http`.

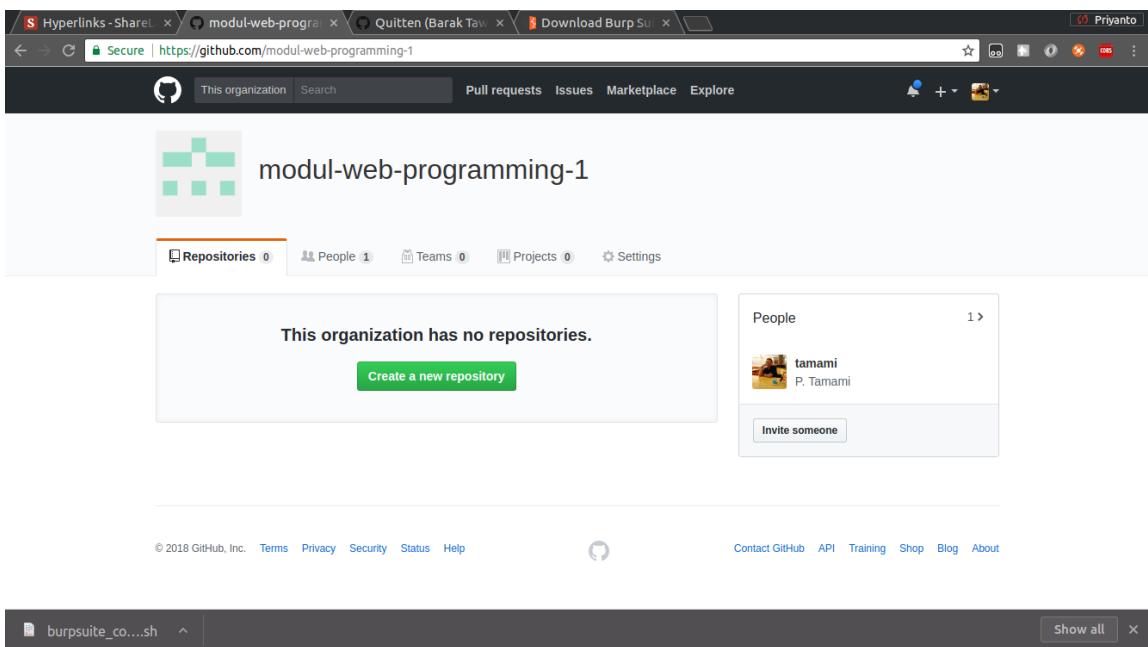


Gambar 1.44: Hasil *Website*

1.3.6 Unggah ke Github

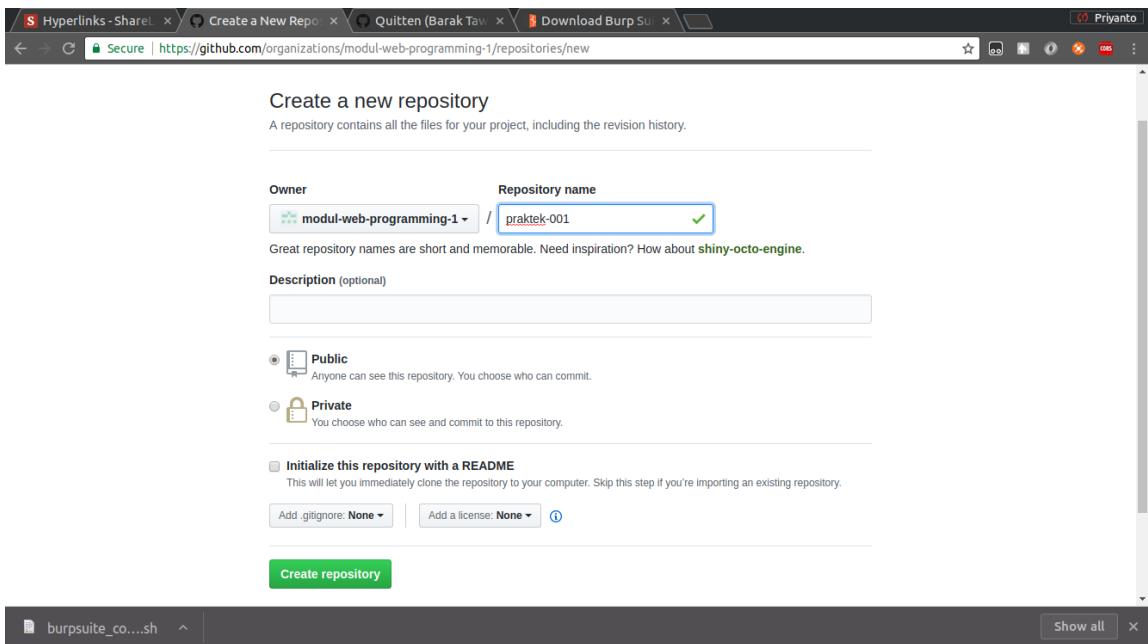
Setelah kode awal berhasil kita unggah ke `id.000webhost.com`, kita akan unggah pula kode yang kita buat ke `Github` untuk keperluan *versioning* kode. Langkahnya cukup mudah, yaitu :

1. Membuat *repository* di Github dengan menekan tombol hijau bertuliskan "Create a new repository" seperti pada gambar 1.45.



Gambar 1.45: Halaman Awal Github

2. Kemudian mengisikan nama repositorinya pada kolom yang tersedia, lalu tekan tombol **Enter** atau klik tombol ”Create Repository” seperti pada gambar 1.46 yang nantinya.

Gambar 1.46: Membuat *Repository*

3. Membuka terminal atau command prompt atau console untuk melakukan *init git* dengan lokasi direktoriya adalah tempat file *index.html* berada dengan

kode berikut :

```
1 > git init
```

4. Melakukan *staging file* dengan perintah berikut :

```
1 > git add .
```

5. Melakukan *commit* terhadap *staging file* dengan perintah berikut :

```
1 > git commit -m "init commit"
```

Opsi perintah `-m "init commit"` sebetulnya wajib, jadi setiap kita melakukan *commit*, kita diminta untuk memberikan keterangan / komentar di tiap *commit* untuk memudahkan kita mencari tahap perubahan tertentu.

6. Mendaftarkan alamat penyedia layanan repositori `git` dengan perintah berikut :

```
1 > git remote add github https://github.com/modul-web-programming  
-1/praktek-001.git
```

`github` di atas adalah nama alias, jadi boleh diganti apapun, sedangkan `url` setelahnya adalah alamat yang diberikan oleh Github setelah kita membuat sebuah *repository*.

7. Melakukan unggah kode ke Github dengan perintah berikut :

```
1 > git push -u github master
```

`github` adalah alias yang sebelumnya kita buat, sedangkan `master` adalah *branch* utama dari `git`.

1.4 Kesimpulan

Bahwa membangun sebuah aplikasi *web* menggunakan PHP yang bersifat dinamis diperlukan beberapa perangkat / aplikasi pendukung.

Dengan fasilitas gratis pun kita masih dapat melakukan *publish project* aplikasi web yang dibangun dengan PHP kita ke internet sehingga dapat diakses oleh semua orang.

1.5 Tugas

Membuat halaman selamat datang dalam bentuk HTML dan di-*publish* ke internet.

Bab 2

Operator, Variabel, dan Tipe Data

2.1 Tujuan

Mahasiswa mampu mengimplementasikan alur sederhana untuk menggunakan operator, variabel, dan tipe data pada PHP.

2.2 Pengantar

Variabel sebetulnya bisa diibaratkan sebagai wadah yang akan diisikan oleh data. Sedangkan *tipe data* bisa dianalogikan seperti jenis data yang berada pada suatu variabel. *Operator* sendiri adalah perintah untuk melakukan operasi tertentu pada satu atau lebih variabel / data.

Untuk variabel sebenarnya akan memiliki *scope* seperti berikut :

- **local**, variabel akan bersifat lokal, tidak dapat digunakan di luar deklarasinya.
- **global**, variabel akan bersifat global, dan dapat digunakan di luar deklarasinya.
- **static**, variabel jenis ini tidak akan dihapus dari memori ketika sudah dieksekusi.

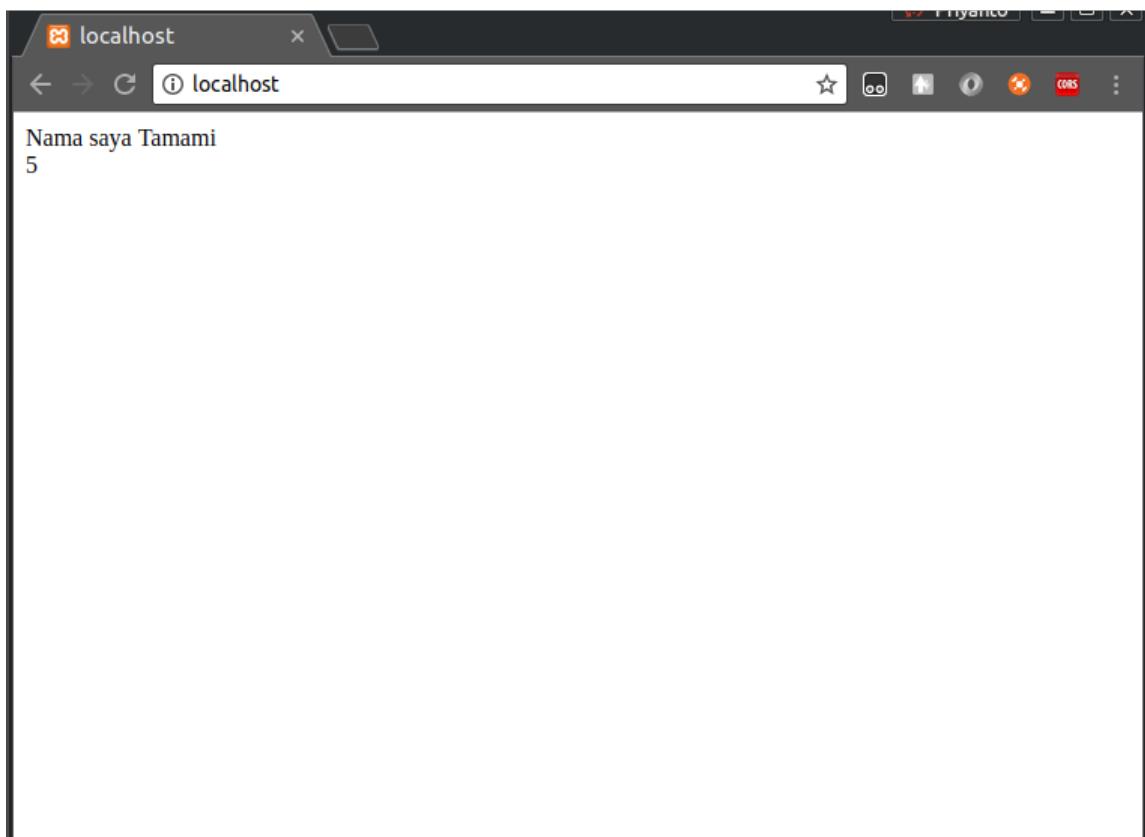
2.3 Praktek

2.3.1 Variabel

Untuk menggunakan variabel, kita cukup memberikan tanda dollar (\$) di awal nama variabel. Contoh kodennya adalah sebagai berikut :

```
1 <?php
2 $teks = "Nama saya Tamami";
3 $angka = 5;
4
5 echo $teks;
6 echo "<br>";
7 echo $angka;
8 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.1 berikut :



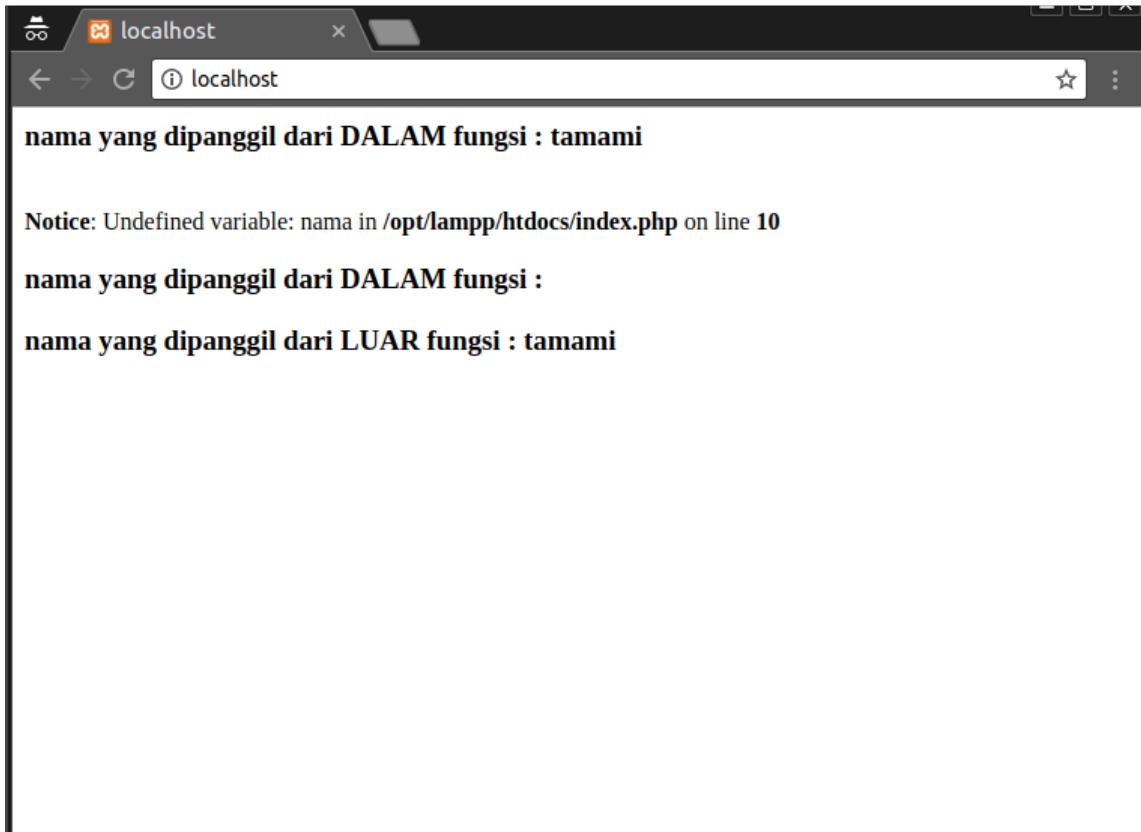
Gambar 2.1: Hasil Keluaran Variabel

Variabel *Global*

Contoh kode untuk menjelaskan bagaimana sifat variabel *global* adalah sebagai berikut :

```
1 <?php
2 $nama = "tamami";
3
4 function cetakNama1() {
5     global $nama;
6     echo "<h3>nama yang dipanggil dari DALAM fungsi : $nama</h3>";
7 }
8
9 function cetakNama2() {
10    echo "<h3>nama yang dipanggil dari DALAM fungsi : $nama</h3>";
11 }
12
13 cetakNama1();
14 cetakNama2();
15
16 echo "<h3>nama yang dipanggil dari LUAR fungsi : $nama</h3>";
17 ?>
```

Hasil dari kode di atas adalah seperti tampilan pada gambar ?? berikut :



Gambar 2.2: Hasil Keluaran Variabel Global

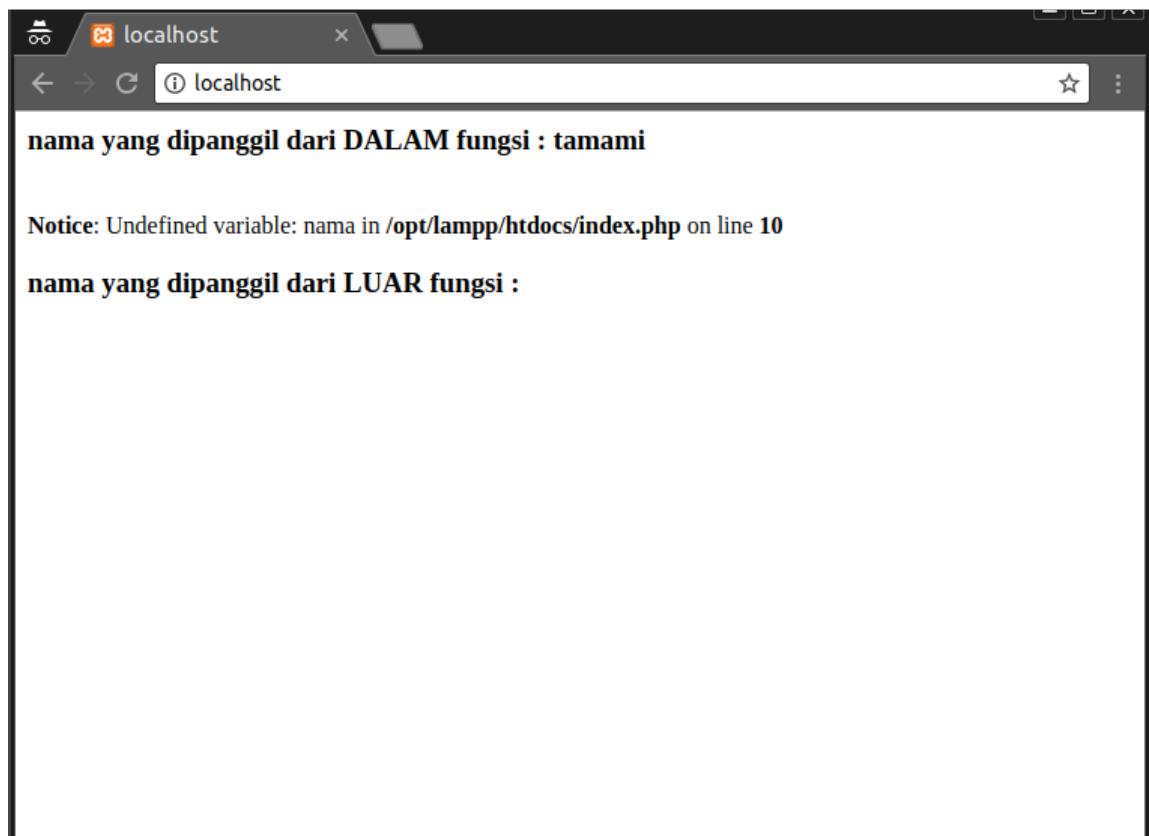
Artinya, variabel *global* dapat digunakan di dalam fungsi maupun di luar fungsi, apabila penggunaannya di dalam sebuah fungsi, maka perlu dideklarasikan terlebih dahulu di dalam fungsi tersebut dengan kata kunci **global**.

Variabel *Local*

Untuk penggunaan variabel *local*, contoh kodennya adalah sebagai berikut :

```
1 <?php
2
3 function cetakNama() {
4     $nama = "tamami";
5     echo "<h3>nama yang dipanggil dari DALAM fungsi : $nama</h3>";
6 }
7
8 cetakNama();
9
10 echo "<h3>nama yang dipanggil dari LUAR fungsi : $nama</h3>";
11 ?>
```

Hasil keluaran dari kode program tersebut adalah seperti pada gambar 2.3 berikut :



Gambar 2.3: Hasil Keluaran Variabel *Local*

Artinya, variabel `$nama` yang dideklarasikan di dalam sebuah fungsi hanya dapat digunakan di dalam fungsi yang bersangkutan.

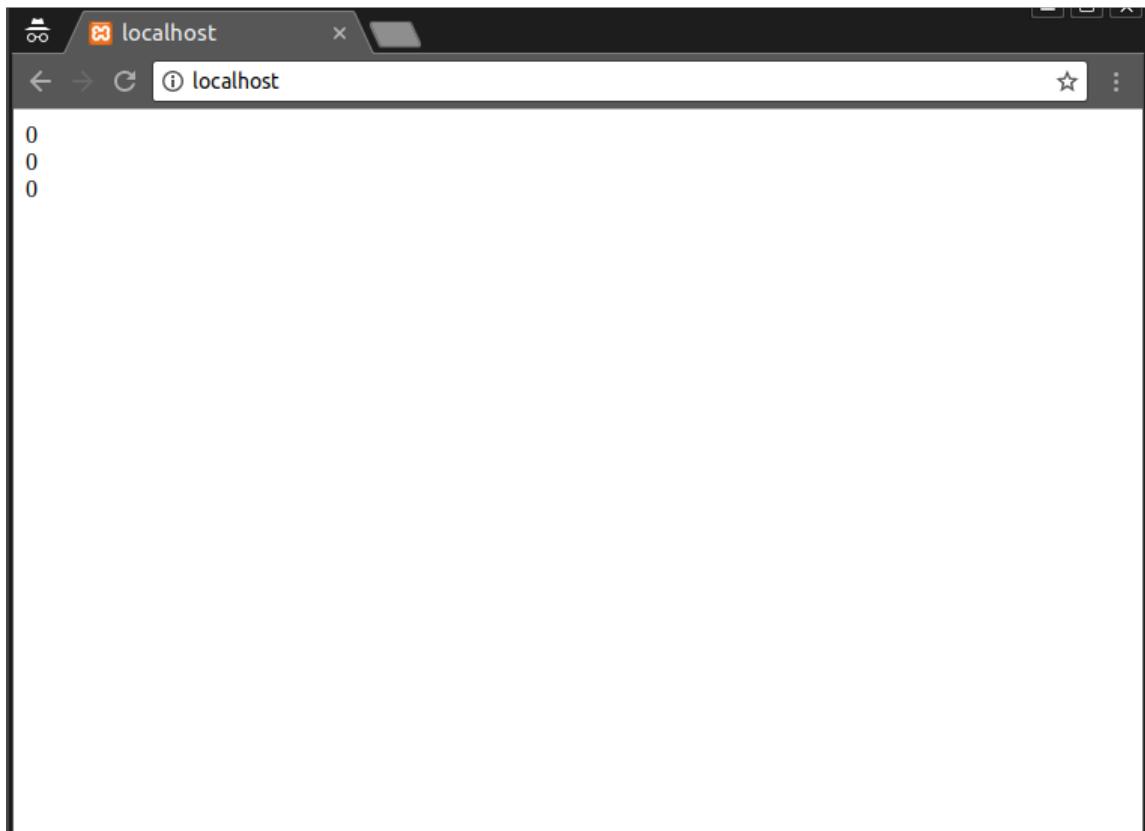
Variabel *Static*

Karena sifat dari variabel di bahasa pemrograman PHP ini akan dihapus dari memori ketika telah dieksekusi, maka bila kita ingin menggunakan variabel ini lebih lama, maka kita harus menjadikannya `static`, berikut adalah contoh kode dimana jenis variabel tidak `static` :

```

1 <?php
2 function hitung() {
3     $nilai = 0;
4     echo $nilai;
5     $nilai += 5;
6 }
7
8 hitung();
9 echo "<br>";
10 hitung();
11 echo "<br>";
12 hitung();
13 ?>
```

Hasil dari kode di atas adalah seperti gambar 2.4 berikut :



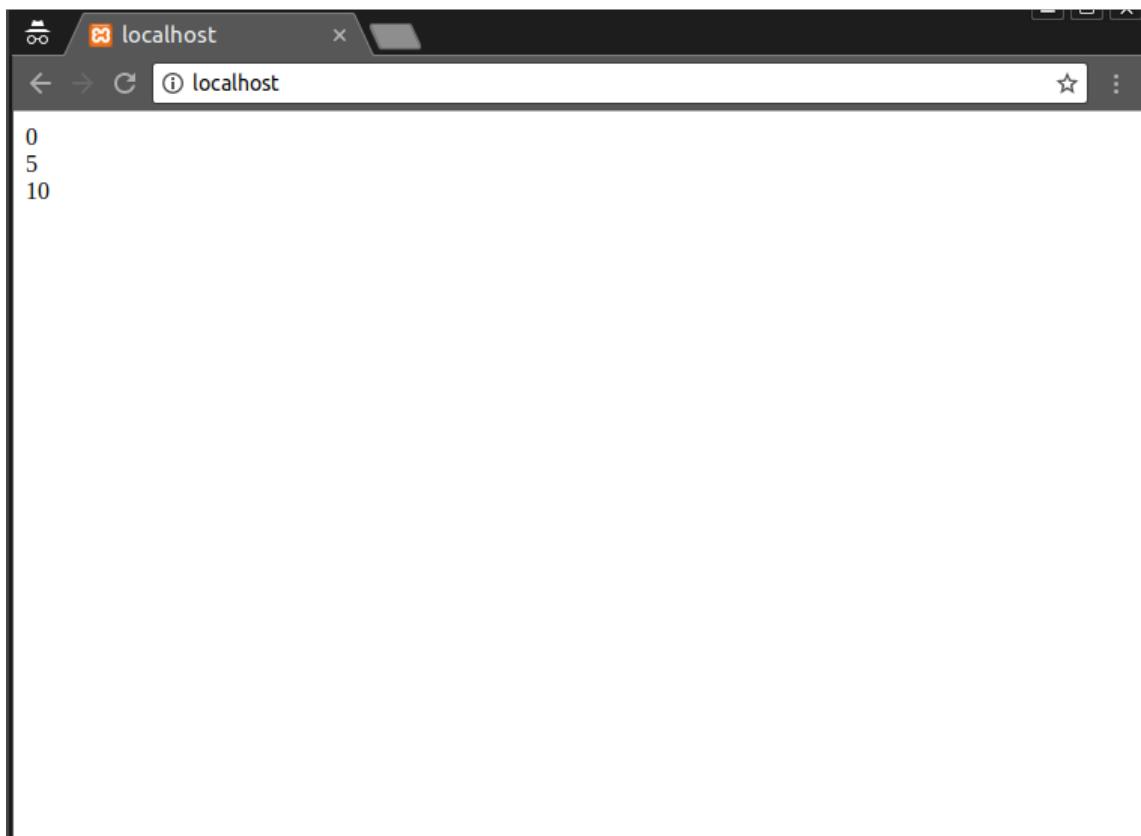
Gambar 2.4: Hasil Keluaran Variabel Tanpa static

Tampak bahwa variabel \$nilai akan selalu bernilai 0 (nol) karena setiap kali eksekusi \$nilai += 5 maka begitu keluar dari fungsi hitung(), isi variabel akan kembali menjadi 0 (nol).

Apabila kita jadikan variabel \$nilai, maka kodennya menjadi seperti ini :

```
1 <?php
2 function hitung() {
3     static $nilai = 0;
4     echo $nilai;
5     $nilai += 5;
6 }
7
8 hitung();
9 echo "<br>";
10 hitung();
11 echo "<br>";
12 hitung();
13 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 2.5 berikut :



Gambar 2.5: Hasil Keluaran Variabel Dengan `static`

Hasil dari keluaran pada gambar 2.5 tampak bahwa variabel `$nilai` akan melakukan eksekusi dan menyimpan hasil tanpa menghapusnya dari memori.

2.3.2 Tipe Data

Variabel pada dasarnya menyimpan sebuah data dengan tipe data tertentu, masing-masing tipe data dapat melakukan hal yang berbeda-beda. Tipe data yang didukung oleh PHP ini adalah sebagai berikut :

- String, tipe data ini untuk menyimpan teks
- Integer, tipe data ini untuk menyimpan angka bulat, tanpa desimal
- Float / Double, tipe data ini untuk menyimpan angka yang memiliki bilangan desimal
- Boolean, tipe data ini hanya menampung nilai `true` atau `false`
- Array, tipe data ini akan menampung baris data baik dengan tipe data yang sama, maupun tipe data yang berbeda.
- Object, tipe data ini akan menampung objek yang kita buat sendiri

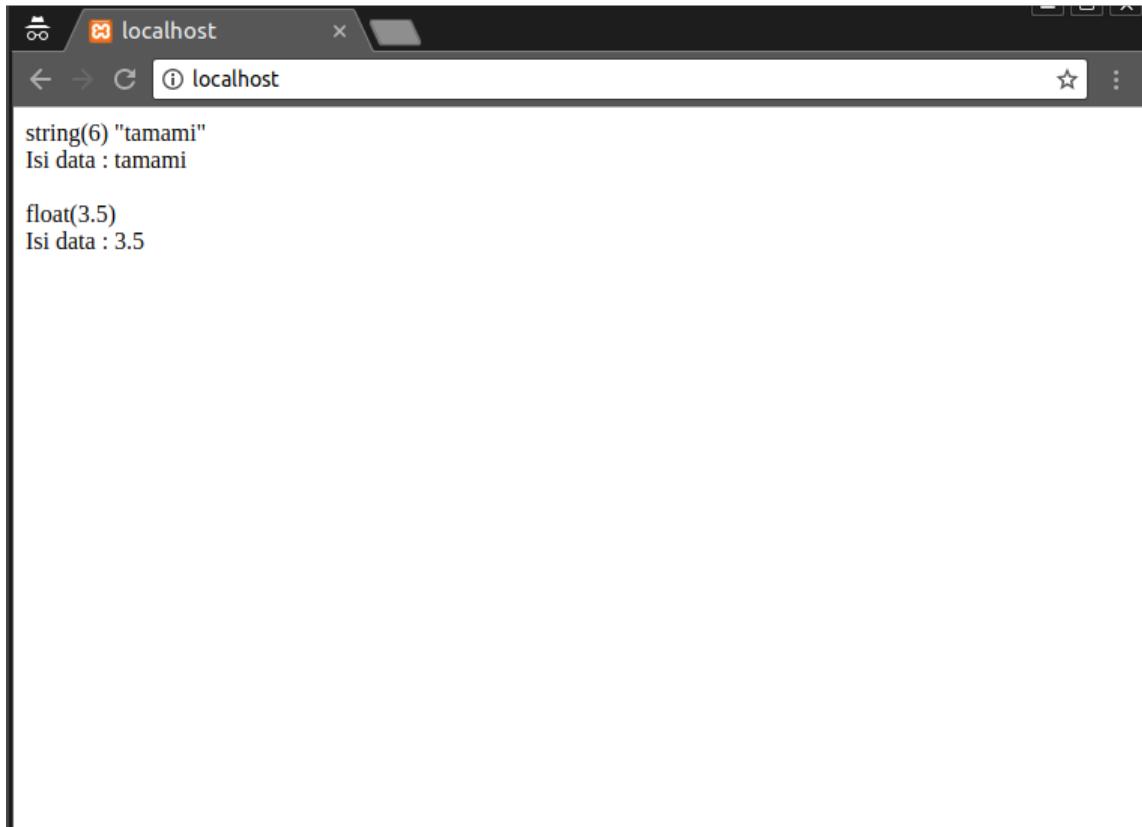
- NULL, tipe data ini artinya tidak memiliki data sama sekali.

Tipe data di PHP tidak dideklarasikan secara langsung, melainkan pada saat pengisian data ke sebuah variabel, pada saat itu juga tipe data akan ditentukan terhadap variabel tersebut.

Perhatikan kode berikut ini :

```
1 <?php
2 $data = "tamami";
3 var_dump($data);
4 echo "<br>";
5 echo "Isi data : $data";
6 echo "<br>";
7 echo "<br>";
8 $data = 3.5;
9 var_dump($data);
10 echo "<br>";
11 echo "Isi data : $data";
12 ?>
```

Kode tersebut akan menghasilkan tampilan seperti pada gambar 2.6 berikut :



Gambar 2.6: Melihat Tipe Data

Terlihat pada saat kita mengisikan variabel \$data dengan teks, maka tipe data

langsung terisi menjadi **string**, kemudian saat kita mengganti isinya menjadi 3.5 maka tipe data ikut berubah menjadi **float**.

2.3.3 Operator

Operator pada PHP akan dibagi menjadi beberapa bagian seperti berikut :

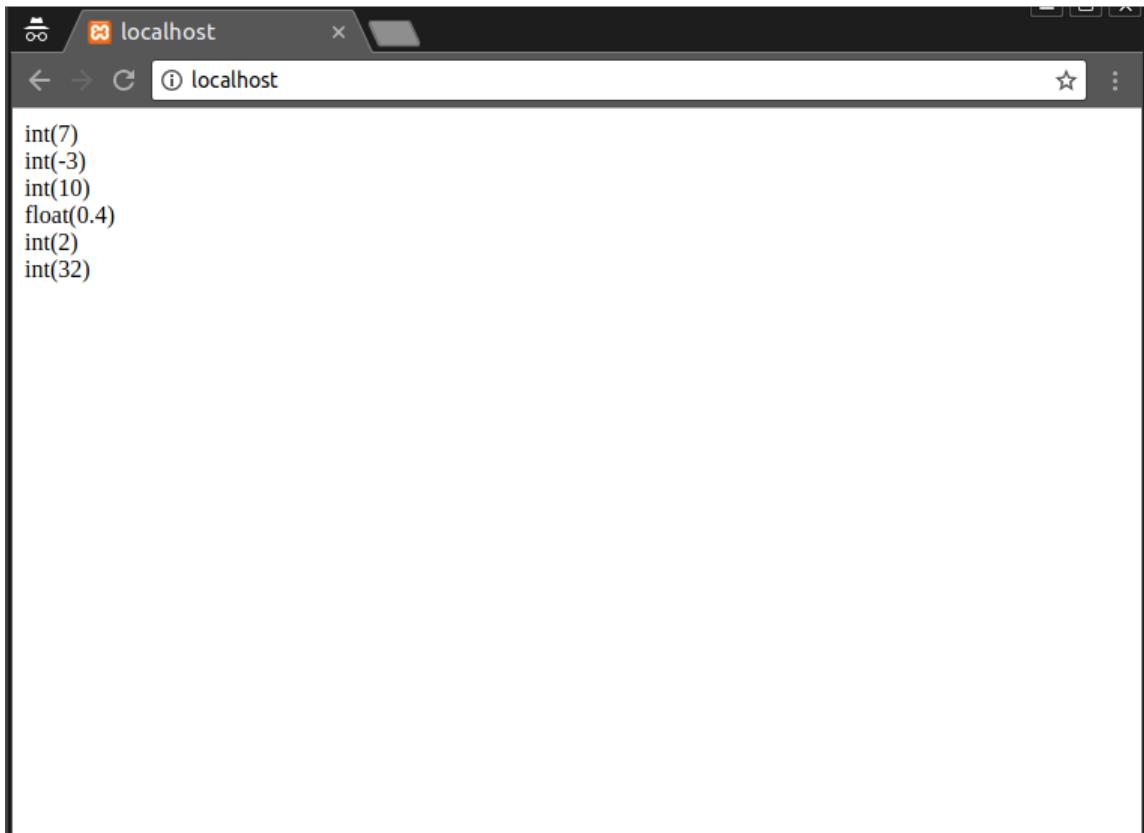
Arithmetric Operators

Operator ini berlaku untuk melakukan operasi aritmatika dasar di PHP, contoh kodennya adalah sebagai berikut :

```

1 <?php
2 $x = 2;
3 $y = 5;
4
5 $result = $x + $y; // operasi penjumlahan
6 var_dump($result);
7 echo "<br>";
8
9 $result = $x - $y; // operasi pengurangan
10 var_dump($result);
11 echo "<br>";
12
13 $result = $x * $y; // operasi perkalian
14 var_dump($result);
15 echo "<br>";
16
17 $result = $x / $y; // operasi pembagian
18 var_dump($result);
19 echo "<br>";
20
21 $result = $x % $y; // operasi sisa bagi
22 var_dump($result);
23 echo "<br>";
24
25 $result = $x ** $y; // operasi pangkat
26 var_dump($result);
27 echo "<br>";
28 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 2.7 berikut :



Gambar 2.7: Hasil Keluaran Operasi Aritmatika

Assignment Operators

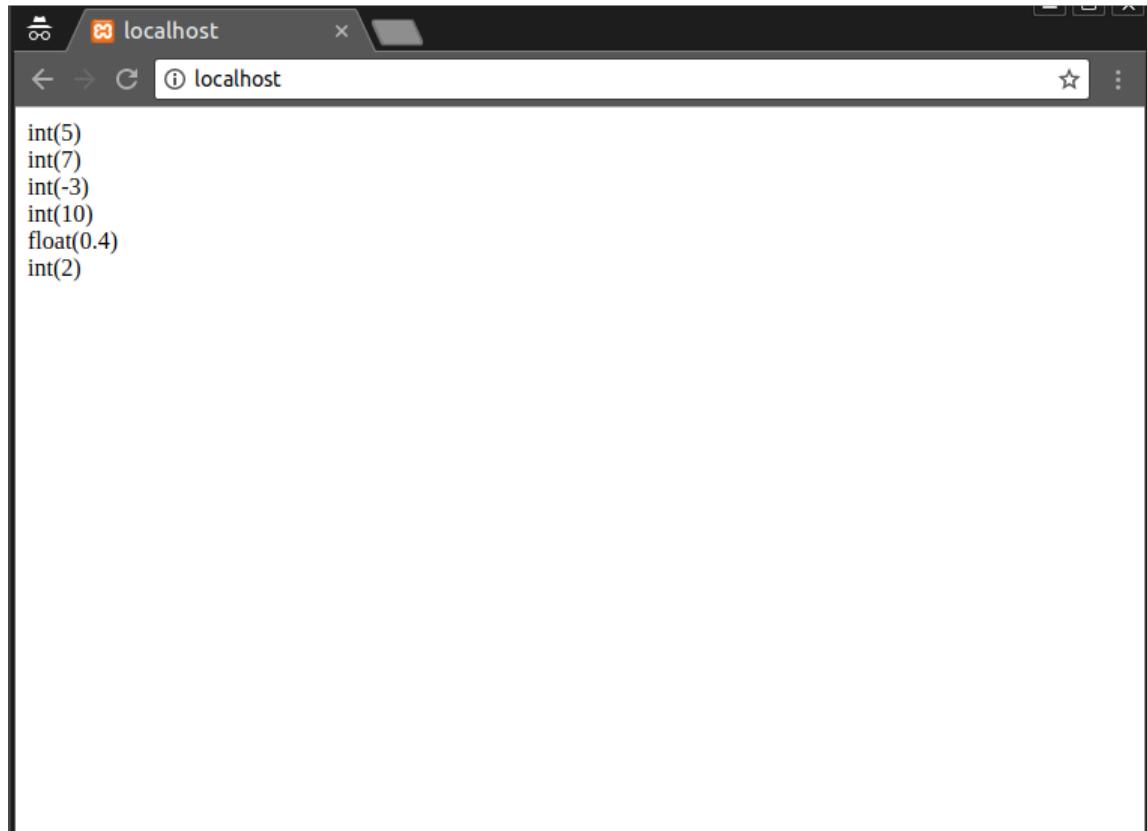
Untuk operator pengisian ditandai dengan tanda = (sama dengan). Nantinya variabel yang ada di sebelah kiri tanda = (sama dengan) akan terisi dengan ekspresi yang ada di sebelah kanan tanda = (sama dengan). Contoh kodennya adalah sebagai berikut :

```
1 <?php
2 $x = 2;
3 $y = 5;
4
5 $x = $y;           // operasi pengisian , dimana variabel x diisikan nilai
6      dari variabel y
7 var_dump($x);
8 echo "<br>";
9 $x = 2;
10
11 $x += $y;          // operasi penambahan , sama seperti x = x + y;
12 var_dump($x);
13 echo "<br>";
14 $x = 2;
15
16 $x -= $y;          // operasi pengurangan , sama seperti x = x - y;
17 var_dump($x);
18 echo "<br>";
```

```

18 $x = 2;
19
20 $x *= $y;           // operasi perkalian , sama seperti x = x * y;
21 var_dump($x);
22 echo "<br>";
23 $x = 2;
24
25 $x /= $y;           // operasi pembagian , sama seperti x = x / y;
26 var_dump($x);
27 echo "<br>";
28 $x = 2;
29
30 $x %= $y;           // operasi sisa bagi , sama seperti x = x % y;
31 var_dump($x);
32 echo "<br>";
33 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.8 berikut :



Gambar 2.8: Hasil Keluaran Dari Operator Penugasan

Comparison Operators

Operator ini digunakan untuk membandingkan dua data baik angka maupun teks, berikut adalah operator yang dapat digunakan disertai dengan contohnya :

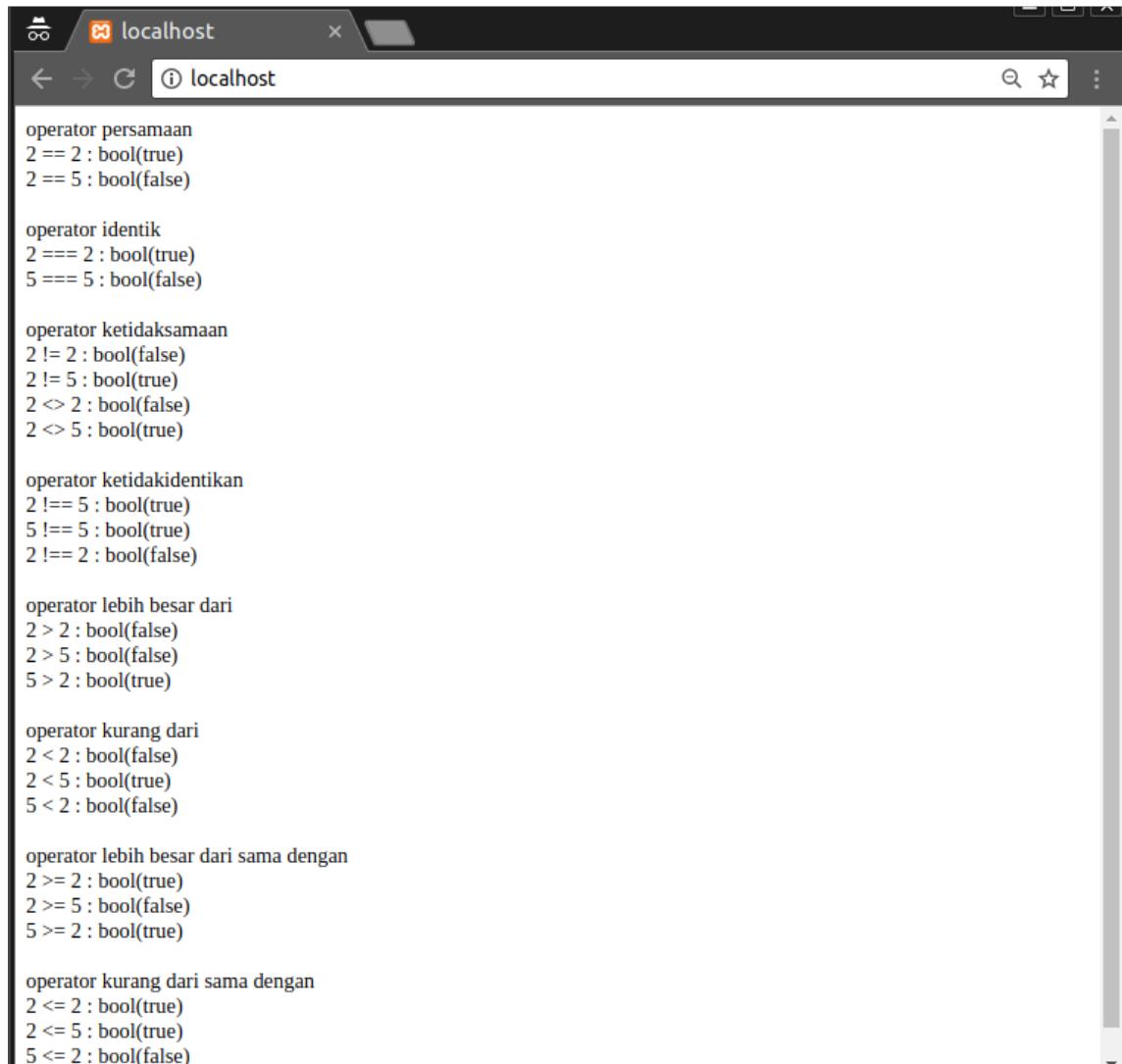
```
1 <?php
2 $w = 2;
3 $x = 2;
4 $y = 5;
5 $z = "5";
6
7 echo "operator persamaan<br>";
8 echo "$w == $x : ";
9 var_dump($w == $x);
10 echo "<br>";
11 echo "$w == $y : ";
12 var_dump($w == $y);
13 echo "<br><br>";
14
15 echo "operator identik<br>";
16 echo "$w === $x : ";
17 var_dump($w === $x);
18 echo "<br>";
19 echo "$y === $z : ";
20 var_dump($y === $z);
21 echo "<br><br>";
22
23 echo "operator ketidaksamaan<br>";
24 echo "$w != $x : ";
25 var_dump($w != $x);
26 echo "<br>";
27 echo "$w != $y : ";
28 var_dump($w != $y);
29 echo "<br>";
30 echo "$w <> $x : ";
31 var_dump($w <> $x);
32 echo "<br>";
33 echo "$w <> $y : ";
34 var_dump($w <> $y);
35 echo "<br><br>";
36
37 echo "operator ketidakidentikan<br>";
38 echo "$w !== $y : ";
39 var_dump($w !== $y);
40 echo "<br>";
41 echo "$y !== $z : ";
42 var_dump($y !== $z);
43 echo "<br>";
44 echo "$w !== $x : ";
45 var_dump($w !== $x);
46 echo "<br><br>";
47
48 echo "operator lebih besar dari<br>";
49 echo "$w > $x : ";
50 var_dump($w > $x);
51 echo "<br>";
52 echo "$w > $y : ";
53 var_dump($w > $y);
54 echo "<br>";
```

```

55 echo "$y > $w : ";
56 var_dump($y > $w);
57 echo "<br><br>";
58
59 echo "operator kurang dari<br>";
60 echo "$w < $x : ";
61 var_dump($w < $x);
62 echo "<br>";
63 echo "$w < $y : ";
64 var_dump($w < $y);
65 echo "<br>";
66 echo "$y < $w : ";
67 var_dump($y < $w);
68 echo "<br><br>";
69
70 echo "operator lebih besar dari sama dengan<br>";
71 echo "$w >= $x : ";
72 var_dump($w >= $x);
73 echo "<br>";
74 echo "$w >= $y : ";
75 var_dump($w >= $y);
76 echo "<br>";
77 echo "$y >= $w : ";
78 var_dump($y >= $w);
79 echo "<br><br>";
80
81 echo "operator kurang dari sama dengan<br>";
82 echo "$w <= $x : ";
83 var_dump($w <= $x);
84 echo "<br>";
85 echo "$w <= $y : ";
86 var_dump($w <= $y);
87 echo "<br>";
88 echo "$y <= $w : ";
89 var_dump($y <= $w);
90 echo "<br><br>";
91 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.9 berikut ini :



The screenshot shows a browser window with the address bar set to 'localhost'. The page content displays the results of a PHP script that tests various comparison operators with values 2 and 5. The output is organized into sections:

- operator persamaan**
2 == 2 : bool(true)
2 == 5 : bool(false)
- operator identik**
2 === 2 : bool(true)
5 === 5 : bool(false)
- operator ketidaksamaan**
2 != 2 : bool(false)
2 != 5 : bool(true)
2 <> 2 : bool(false)
2 <> 5 : bool(true)
- operator ketidakidentikan**
2 !== 5 : bool(true)
5 !== 5 : bool(true)
2 !== 2 : bool(false)
- operator lebih besar dari**
2 > 2 : bool(false)
2 > 5 : bool(false)
5 > 2 : bool(true)
- operator kurang dari**
2 < 2 : bool(false)
2 < 5 : bool(true)
5 < 2 : bool(false)
- operator lebih besar dari sama dengan**
2 >= 2 : bool(true)
2 >= 5 : bool(false)
5 >= 2 : bool(true)
- operator kurang dari sama dengan**
2 <= 2 : bool(true)
2 <= 5 : bool(true)
5 <= 2 : bool(false)

Gambar 2.9: Hasil Kode Operator Perbandingan

Terlihat bahwa operator ini akan membandingkan dua nilai yang ada, yang perlu diperhatikan adalah bahwa operator persamaan dengan operator identik adalah bahwa operator identik akan memeriksa persamaannya sampai ke jenis tipe data. Apabila tipe datanya berbeda, maka akan dianggap tidak identik.

Sedangkan untuk operator ketidakidentikan, maka operator akan memeriksa, apabila datanya memang berbeda, maka akan memberikan nilai `true`, kemudian apabila datanya sama, namun tipe datanya berbeda, maka akan memberikan nilai `true` pula.

Increment / Decrement Operators

Operator ini digunakan untuk menambahkan nilai atau mengurangkan nilai 1 ke dalam variabel. Berikut contoh kodenya :

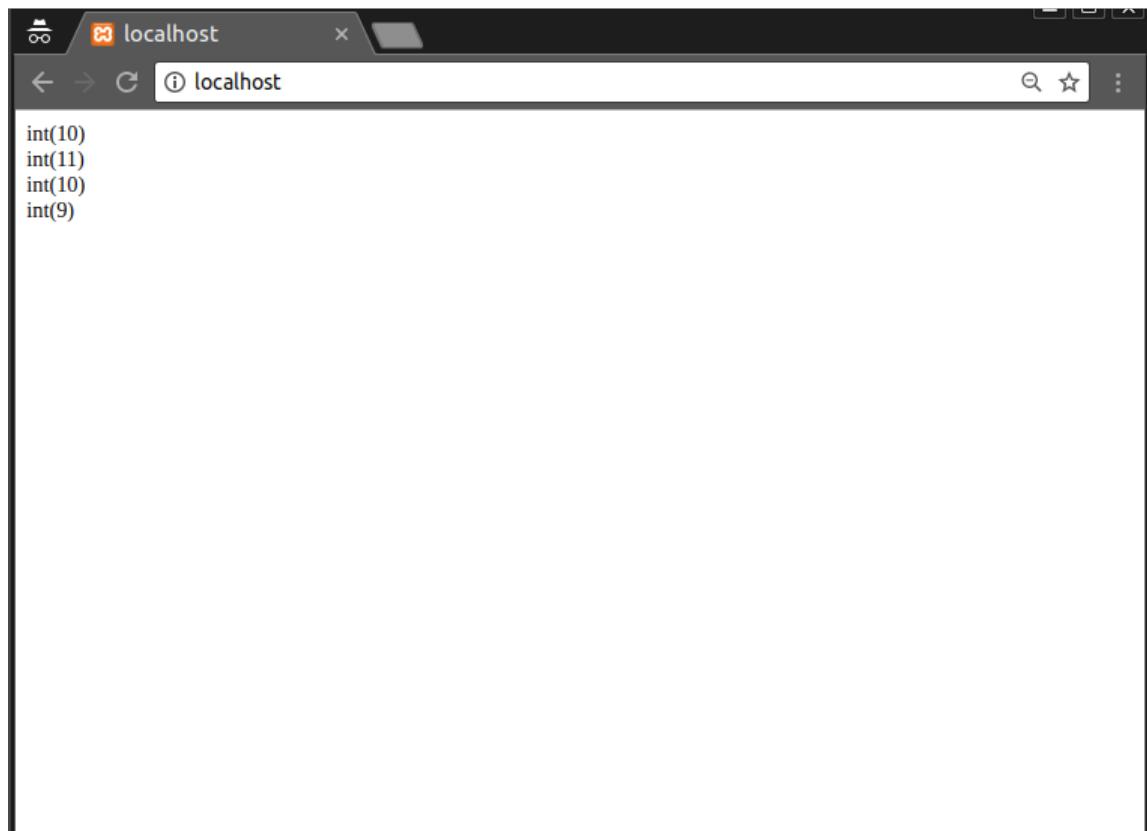
```

1 <?php
2 $nilai = 10;

```

```
3 var_dump($nilai++);  
4 echo "<br>";  
5  
6 $nilai = 10;  
7 var_dump(++$nilai);  
8 echo "<br>";  
9  
10 $nilai = 10;  
11 var_dump($nilai--);  
12 echo "<br>";  
13  
14 $nilai = 10;  
15 var_dump(--$nilai);  
16 echo "<br>";  
17 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.10 berikut :



Gambar 2.10: Hasil Operator Penambahan dan Pengurangan

Terlihat bahwa pada saat operator di tambahkan di belakang variabel, maka proses penambahan atau pengurangan dilakukan setelah ekspresi, sedangkan saat operator di tambahkan di depan variabel, maka proses penambahan atau pengurangan dilakukan sebelum ekspresi dieksekusi.

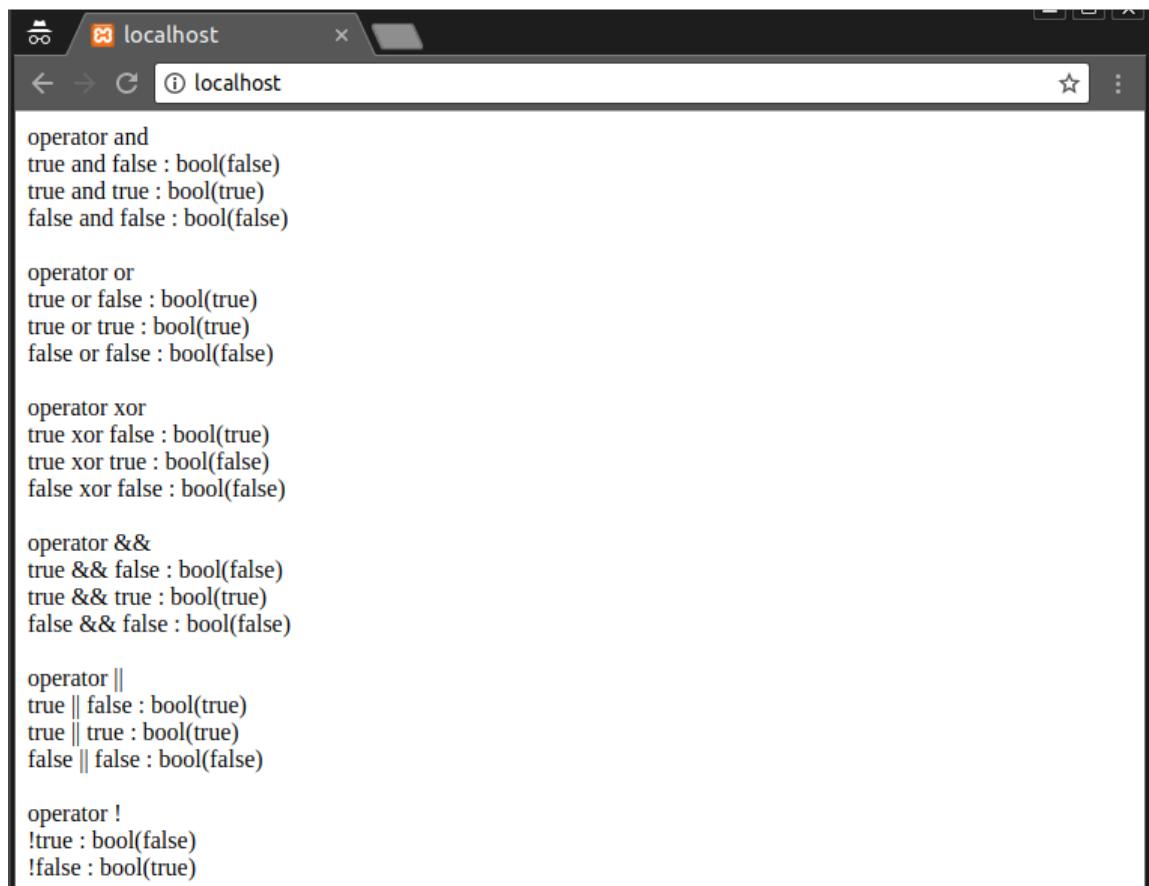
Logical Operators

Operator logika digunakan untuk melakukan pemeriksaan benar dan salah dari dua buah data. Contoh kode untuk penggunaan operator logika ini adalah sebagai berikut :

```

1 <?php
2 $a = true;
3 $b = false;
4 $c = true;
5 $d = false;
6
7 echo "operator and<br>";
8 echo "true and false : "; var_dump($a and $b); echo "<br>";
9 echo "true and true : "; var_dump($a and $c); echo "<br>";
10 echo "false and false : "; var_dump($b and $d); echo "<br><br>";
11
12 echo "operator or<br>";
13 echo "true or false : "; var_dump($a or $b); echo "<br>";
14 echo "true or true : "; var_dump($a or $c); echo "<br>";
15 echo "false or false : "; var_dump($b or $d); echo "<br><br>";
16
17 echo "operator xor<br>";
18 echo "true xor false : "; var_dump($a xor $b); echo "<br>";
19 echo "true xor true : "; var_dump($a xor $c); echo "<br>";
20 echo "false xor false : "; var_dump($b xor $d); echo "<br><br>";
21
22 echo "operator &&<br>";
23 echo "true && false : "; var_dump($a && $b); echo "<br>";
24 echo "true && true : "; var_dump($a && $c); echo "<br>";
25 echo "false && false : "; var_dump($b && $d); echo "<br><br>";
26
27 echo "operator ||<br>";
28 echo "true || false : "; var_dump($a || $b); echo "<br>";
29 echo "true || true : "; var_dump($a || $c); echo "<br>";
30 echo "false || false : "; var_dump($b || $d); echo "<br><br>";
31
32 echo "operator !<br>";
33 echo "!true : "; var_dump(!$a); echo "<br>";
34 echo "!false : "; var_dump(!$b); echo "<br>";
35 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.11 berikut :



The screenshot shows a browser window titled 'localhost' with the URL 'localhost'. The page content displays the results of a PHP script. It lists the behavior of various logical operators:

- operator and**
true and false : bool(false)
true and true : bool(true)
false and false : bool(false)
- operator or**
true or false : bool(true)
true or true : bool(true)
false or false : bool(false)
- operator xor**
true xor false : bool(true)
true xor true : bool(false)
false xor false : bool(false)
- operator &&**
true && false : bool(false)
true && true : bool(true)
false && false : bool(false)
- operator ||**
true || false : bool(true)
true || true : bool(true)
false || false : bool(false)
- operator !**
!true : bool(false)
!false : bool(true)

Gambar 2.11: Hasil Keluaran Dari Operator Logika

Terlihat bahwa operator **and** dan **&&** akan menghasilkan nilai yang sama, sedangkan operator **or** dan **||** akan menghasilkan nilai yang sama pula.

Untuk operator **xor**, maka apabila ada salah satu data yang bernilai **true** maka akan mengembalikan nilai **true**, namun apabila nilai kedua data adalah **false** atau **true**, maka hasilnya akan bernilai **false**.

Operator **!** (tanda seru) akan menegaskan hasil datanya.

String Operators

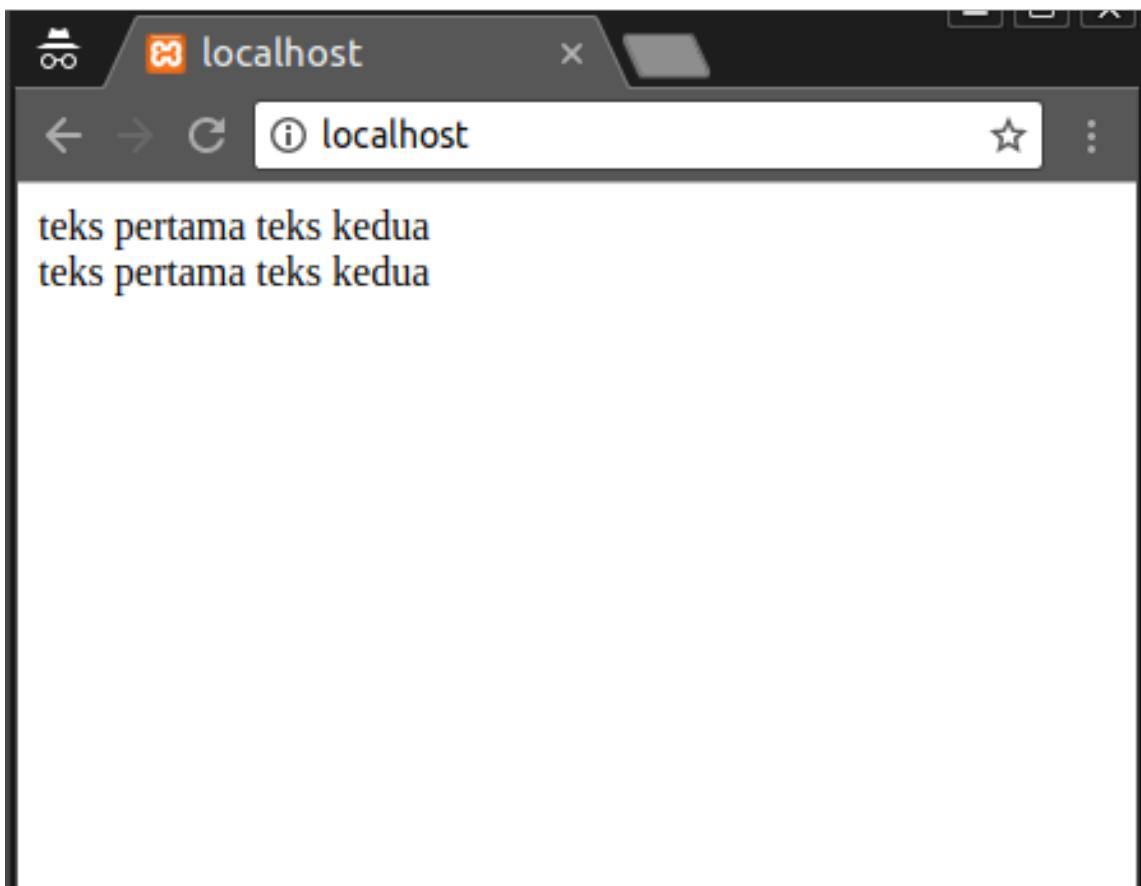
Operator teks sederhana yang disediakan oleh PHP adalah operator untuk menggabungkan 2 (dua) teks menjadi 1 rangkaian. Contoh kodennya adalah sebagai berikut :

```

1 <?php
2 $teks1 = "teks pertama ";
3 $teks2 = "teks kedua";
4
5 echo $teks1.$teks2; echo "<br>";
6 $teks1 .= $teks2;
7 echo $teks1;
8 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.12 berikut :



Gambar 2.12: Hasil Kode Penggabungan Teks

Operatornya hanya sebuah . (titik) untuk menggabungkan 2 (dua) teks menjadi satu, apabila hasilnya akan dimasukkan ke variabel pertama, maka dapat menggunakan operator .= (titik sama dengan).

Array Operators

Beberapa operator yang dapat digunakan untuk *array* seperti terlihat pada contoh kode berikut :

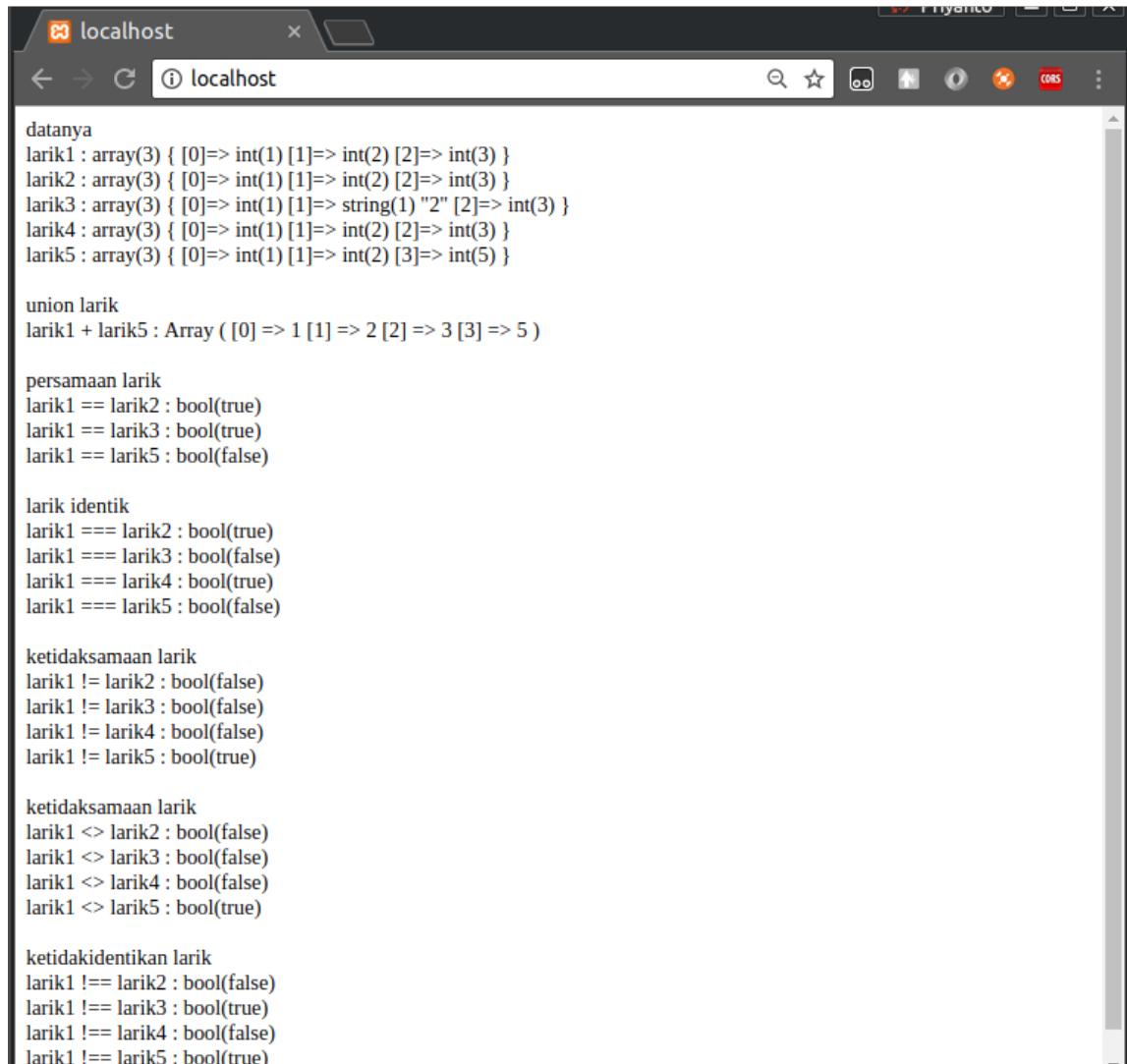
```
1 <?php
2 $larik1 = array(1, 2, 3);
3 $larik2 = array(1, 2, 3);
4 $larik3 = array(1, "2", 3);
5 $larik4 = array("0" => 1, 2, 3);
6 $larik5 = array(0 => 1, 1 => 2, 3 => 5);
7
8 echo "datanya<br>";
9 echo "larik1 : "; var_dump($larik1); echo "<br>";
```

```

10 echo "larik2 : "; var_dump($larik2); echo "<br>" ;
11 echo "larik3 : "; var_dump($larik3); echo "<br>" ;
12 echo "larik4 : "; var_dump($larik4); echo "<br>" ;
13 echo "larik5 : "; var_dump($larik5); echo "<br><br>" ;
14
15 echo "union larik<br>" ;
16 echo "larik1 + larik5 : "; print_r($larik1 + $larik5); echo "<br><br>" ;
17
18 echo "persamaan larik<br>" ;
19 echo "larik1 == larik2 : "; var_dump($larik1 == $larik2); echo "<br>" ;
20 echo "larik1 == larik3 : "; var_dump($larik1 == $larik3); echo "<br>" ;
21 echo "larik1 == larik5 : "; var_dump($larik1 == $larik5); echo "<br><br>" ;
22
23 echo "larik identik<br>" ;
24 echo "larik1 == larik2 : "; var_dump($larik1 == $larik2); echo "<br>" ;
25 echo "larik1 == larik3 : "; var_dump($larik1 == $larik3); echo "<br>" ;
26 echo "larik1 == larik4 : "; var_dump($larik1 == $larik4); echo "<br>" ;
27 echo "larik1 == larik5 : "; var_dump($larik1 == $larik5); echo "<br><br>" ;
28
29 echo "ketidak samaan larik<br>" ;
30 echo "larik1 != larik2 : "; var_dump($larik1 != $larik2); echo "<br>" ;
31 echo "larik1 != larik3 : "; var_dump($larik1 != $larik3); echo "<br>" ;
32 echo "larik1 != larik4 : "; var_dump($larik1 != $larik4); echo "<br>" ;
33 echo "larik1 != larik5 : "; var_dump($larik1 != $larik5); echo "<br><br>" ;
34
35 echo "ketidak samaan larik<br>" ;
36 echo "larik1 <>> larik2 : "; var_dump($larik1 <>> $larik2); echo "<br>" ;
37 echo "larik1 <>> larik3 : "; var_dump($larik1 <>> $larik3); echo "<br>" ;
38 echo "larik1 <>> larik4 : "; var_dump($larik1 <>> $larik4); echo "<br>" ;
39 echo "larik1 <>> larik5 : "; var_dump($larik1 <>> $larik5); echo "<br><br>" ;
40
41 echo "ketidakidentikan larik<br>" ;
42 echo "larik1 !== larik2 : "; var_dump($larik1 !== $larik2); echo "<br>" ;
43 echo "larik1 !== larik3 : "; var_dump($larik1 !== $larik3); echo "<br>" ;
44 echo "larik1 !== larik4 : "; var_dump($larik1 !== $larik4); echo "<br>" ;
45 echo "larik1 !== larik5 : "; var_dump($larik1 !== $larik5); echo "<br><br>" ;
46 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.13 berikut :



The screenshot shows a browser window with the URL `localhost`. The page content displays the result of an array operation. It includes several sections of code and their results:

- datanya**
larik1 : array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
larik2 : array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
larik3 : array(3) { [0]=> int(1) [1]=> string(1) "2" [2]=> int(3) }
larik4 : array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
larik5 : array(3) { [0]=> int(1) [1]=> int(2) [3]=> int(5) }
- union larik**
larik1 + larik5 : Array ([0] => 1 [1] => 2 [2] => 3 [3] => 5)
- persamaan larik**
larik1 == larik2 : bool(true)
larik1 == larik3 : bool(true)
larik1 == larik5 : bool(false)
- larik identik**
larik1 === larik2 : bool(true)
larik1 === larik3 : bool(false)
larik1 === larik4 : bool(true)
larik1 === larik5 : bool(false)
- ketidaksamaan larik**
larik1 != larik2 : bool(false)
larik1 != larik3 : bool(false)
larik1 != larik4 : bool(false)
larik1 != larik5 : bool(true)
- ketidakidentikan larik**
larik1 !== larik2 : bool(false)
larik1 !== larik3 : bool(true)
larik1 !== larik4 : bool(false)
larik1 !== larik5 : bool(true)

Gambar 2.13: Hasil Keluaran Operator *Array* / Larik

2.4 Kesimpulan

Bahwa variabel dapat dibentuk untuk menyimpan sebuah data dengan tipe data tertentu, yang kemudian datanya dapat diolah menggunakan operator sederhana untuk menghasilkan data baru.

2.5 Tugas

Cobalah untuk membuat beberapa variabel dengan data sebagai berikut :

Nama Variabel	Keterangan
nama	diisi nama pribadi
jurusan	diisi jurusan program studi
usia	diisi usia sekarang

Kemudian cetak hasilnya ke *browser* dengan menggunakan echo.

Bab 3

Array

3.1 Tujuan

Mahasiswa mampu mengimplementasikan penggunaan *array* / larik pada PHP untuk alur program sederhana.

3.2 Pengantar

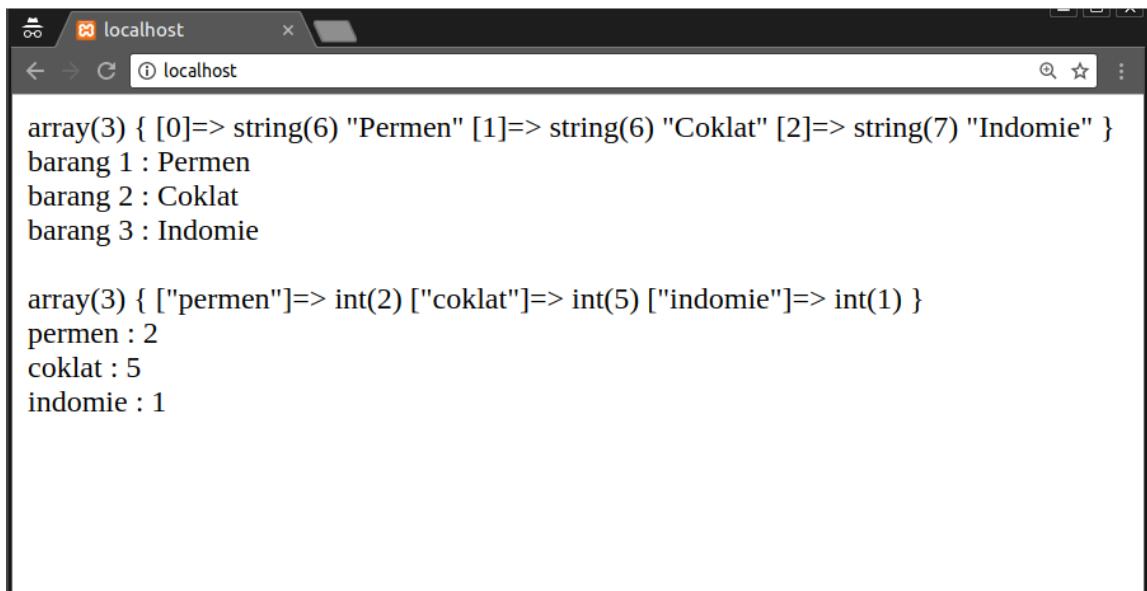
Array atau larik adalah tempat untuk menampung sekumpulan data, baik yang setipe, ataupun yang berbeda tipe. Struktur *array* / larik sendiri sebetulnya tiap elemen data terdiri dari *key* dan *value*.

3.3 Praktek

Cara membuat *array* / larik secara sederhana adalah sebagai berikut :

```
1 <?php
2
3 $daftarBarang = array("Permen", "Coklat", "Indomie");
4 $stok = array("permen" => 2, "coklat" => 5, "indomie" => 1);
5
6 var_dump($daftarBarang); echo "<br>";
7 echo "barang 1 : $daftarBarang[0]<br>";
8 echo "barang 2 : $daftarBarang[1]<br>";
9 echo "barang 3 : $daftarBarang[2]<br><br>";
10
11 var_dump($stok); echo "<br>";
12 echo "permen : $stok[permen]<br>";
13 echo "coklat : $stok[coklat]<br>";
14 echo "indomie : $stok[indomie]<br>";
15
16 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 3.1 berikut ini :

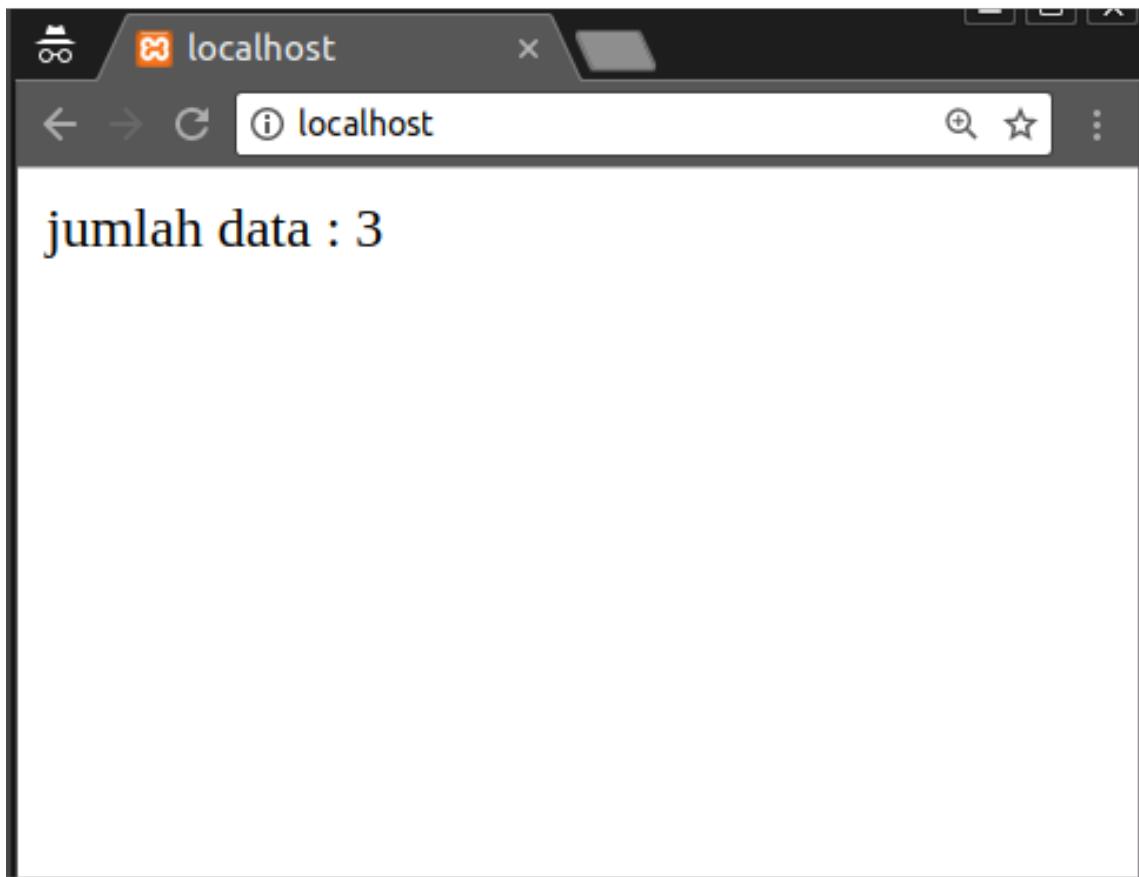


Gambar 3.1: Hasil Keluaran *Array / Larik*

Untuk mengetahui jumlah data yang ada pada *array* kita dapat menggunakan perintah berikut ini :

```
1 <?php
2 $warna = array("merah" , "kuning" , "hijau");
3
4 echo "jumlah data : "; echo count($warna);
5 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 3.2 berikut :



Gambar 3.2: Hasil Keluaran Melihat Jumlah Data Pada *Array*

3.4 Kesimpulan

Apabila ingin menggunakan variabel yang mampu menyimpan beberapa data sekali-gus, maka penggunaan *array* sudah tepat. Beberapa operasi untuk *array* ini pun sudah disediakan oleh PHP.

3.5 Tugas

Cobalah membuat himpunan data dengan *array* yang berisi pasangan data *nim* dan *nama* dengan jumlah data minimal 5 (lima) data.

Bab 4

Struktur Kontrol

4.1 Tujuan

Mahasiswa mampu memahami dan menggunakan kontrol pada PHP.

4.2 Pengantar

PHP dibangun dengan menyediakan beberapa fasilitas seperti seleksi, perulangan, fungsi dan sebagainya. Pada bab ini kita akan bahas bagaimana struktur seleksi dan perulangan di PHP.

4.3 Praktek

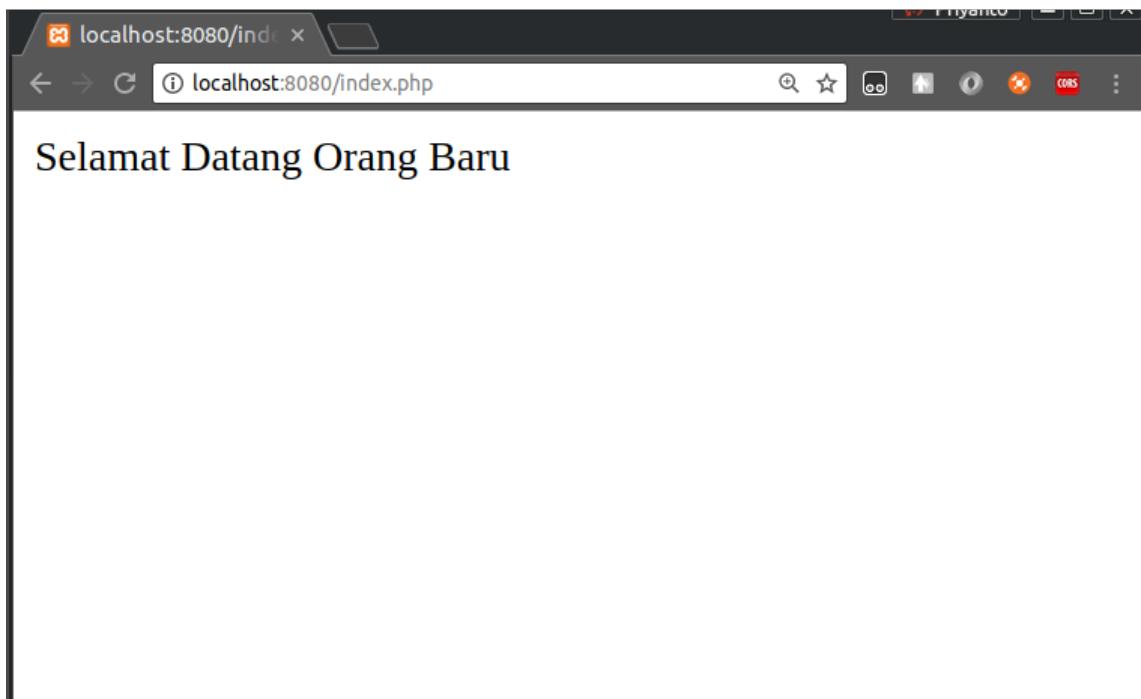
4.3.1 Seleksi / Percabangan if

Seleksi atau percabangan *if* digunakan untuk mengontrol alur berdasarkan logika *true* atau *false*.

Perhatikan kode berikut ini :

```
1 <?php  
2 $orangBaru = true;  
3  
4 if ($orangBaru) {  
5     echo "Selamat Datang Orang Baru";  
6 }  
7 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.1 berikut :



Gambar 4.1: Hasil Keluaran *if*

Dari hasil keluaran tersebut bahwa apabila nilai yang ada di antara tanda kurung (...) bernilai **true**, maka perintah yang berada di antara tanda kurung kurawal (...) akan dijalankan. Namun bila nilai yang berada di dalam tanda kurung (...) bernilai **false**, maka tidak akan menampilkan apapun di *browser*.

Bentuk lain dari *if* ini adalah *if...else*, contoh kodennya sebagai berikut :

```

1 <?php
2 $a = 5;
3 $b = 7;
4
5 if($a > $b) {
6     echo "$a lebih kecil dari $b";
7 } else {
8     echo "$a lebih besar dari $b";
9 }
10 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar ?? berikut ini :

```

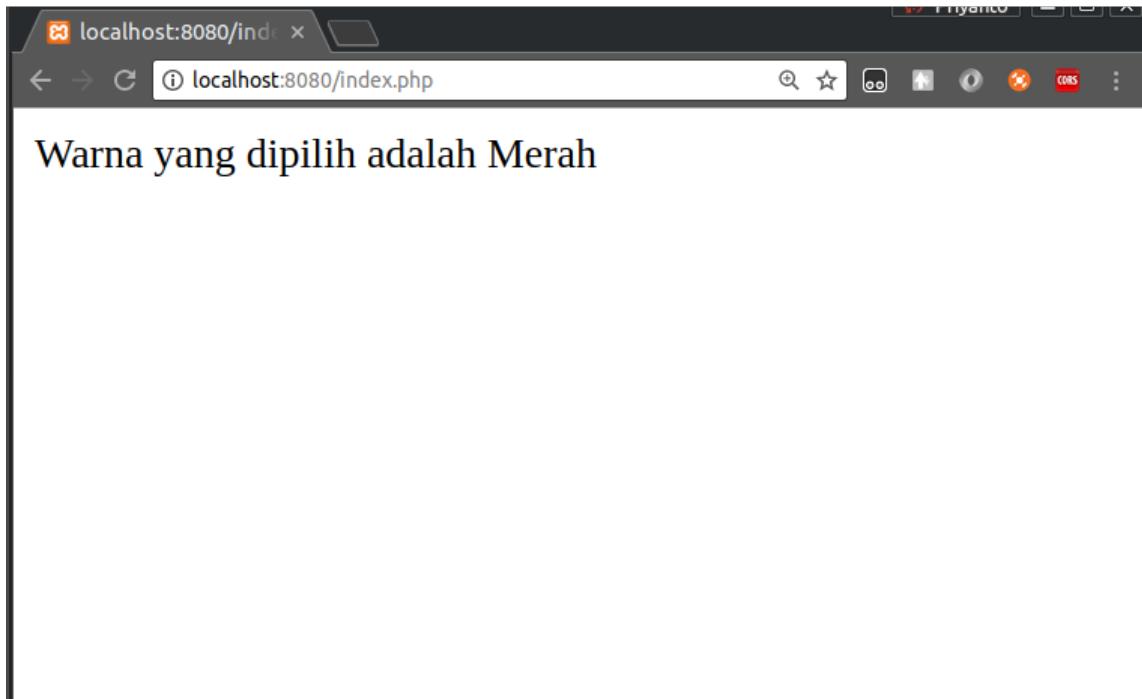
1 \centering
2 \includegraphics [width=1\textwidth]{04-03-01-002-hasil2}
3 \caption{Hasil Keluaran Contoh Kode Menggunakan \texttt{\{else\}}}
4 \label{fig:04-03-01-002}
```

Terlihat bahwa karena pernyataan yang ada di dalam tanda kurung *if(...)* menghasilkan nilai **false**, maka yang ditampilkan adalah perintah pada blok *else*.

Apabila kita ingin menyeleksi lebih dari dua pilihan **true** dan **false** saja, kita bisa menggunakan seleksi **elseif**, berikut contoh kodennya :

```
1 <?php
2 $a = "merah";
3
4 if ($a == "biru") {
5     echo "Warna yang dipilih adalah Biru";
6 } elseif($a == "merah") {
7     echo "Warna yang dipilih adalah Merah";
8 } else {
9     echo "Tidak memilih warna";
10}
11?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.2 berikut ini :



Gambar 4.2: Hasil Keluaran Perintah elseif

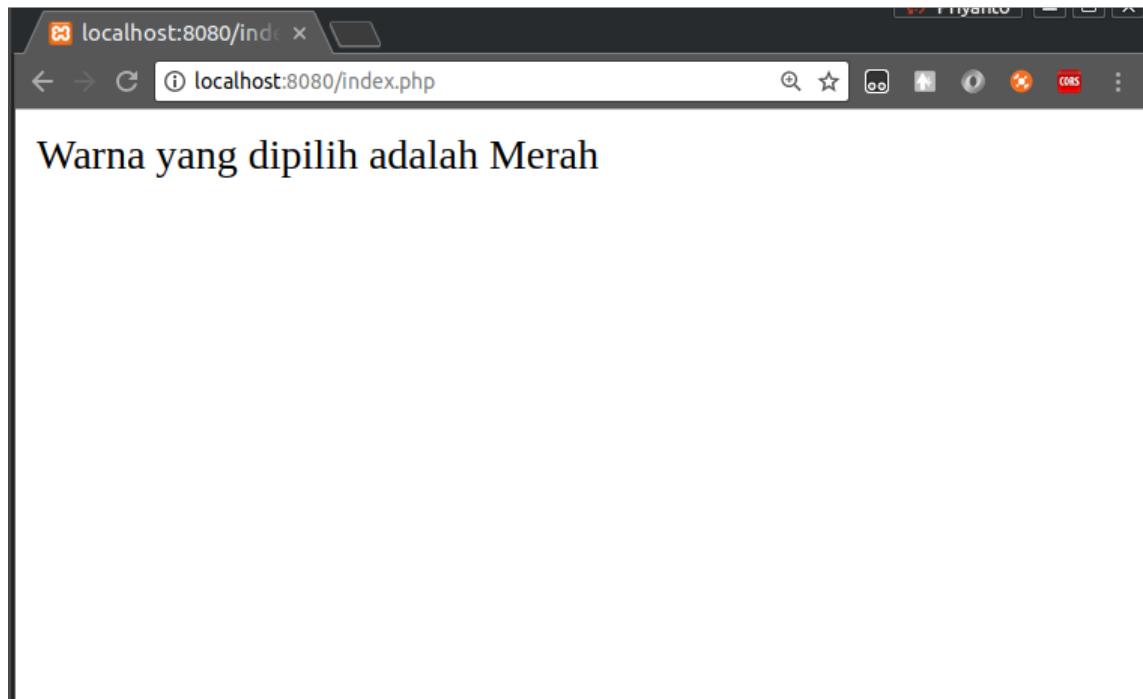
Yang dijalankan adalah perintah yang berada dalam blok `elseif` karena memenuhi kriteria bahwa nilai dari variabel `$a` adalah `merah`, apabila nilai dari variabel tersebut adalah `biru` tentu yang akan dijalankan adalah ekspresi dalam blok `if`, namun bila isi dari variabel `$a` bukan `merah` atau `biru` maka yang dijalankan adalah ekspresi dalam blok `else`.

Kode yang lain untuk melakukan seleksi dalam 1 (satu) baris adalah sebagai berikut :

```
1 <?php
2 $a = "merah";
3
4 $result = ($a == "merah") ? "Warna yang dipilih adalah Merah" : "Warna
5     yang dipilih adalah Merah";
6 echo $result;
```

```
6 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.3 berikut ini :



Gambar 4.3: Hasil Keluaran Ekspresi if Dalam Bentuk Lain

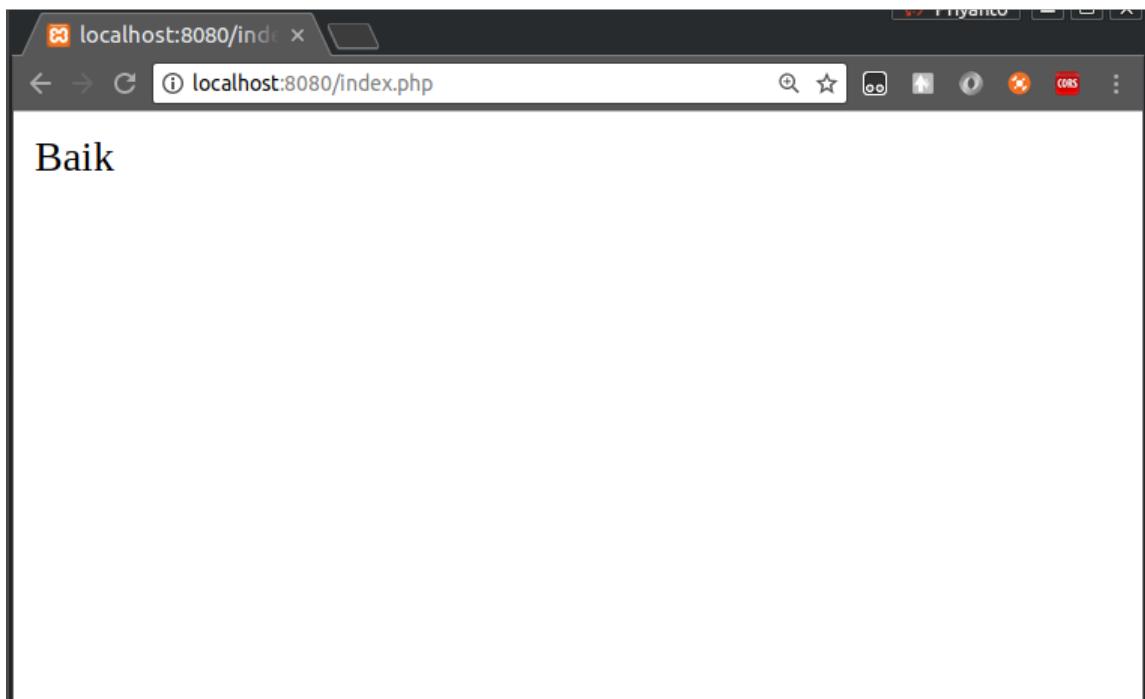
4.3.2 Seleksi / Percabangan switch

Perintah **switch** tujuan mirip dengan **if**, digunakan untuk melakukan seleksi atau percabangan alur program berdasarkan kondisi **true** atau **false**. Contoh sederhana dari penggunaan **switch** ini adalah sebagai berikut :

```
1 <?php
2 $nilai = 7;
3
4 switch($nilai) {
5     case 0:
6     case 1:
7     case 2:
8         echo "Sangat Kurang";
9         break;
10    case 3:
11    case 4:
12        echo "Kurang";
13        break;
14    case 5:
15    case 6:
16        echo "Cukup";
17        break;
18    case 7:
```

```
19 case 8:  
20     echo "Baik";  
21     break;  
22 case 9:  
23 case 10:  
24     echo "Sangat Baik";  
25     break;  
26 default :  
27     echo "Tanpa Nilai";  
28 }  
29 ?>
```

Hasil keluaran dari kode di atas adalah seperti gambar 4.4 berikut ini :



Gambar 4.4: Hasil Keluaran Ekspresi **switch**

Terlihat bahwa pada saat variabel \$nilai berisi angka 7, seleksi **switch** akan melakukan pemeriksaan nilai variabel ini, nilainya akan dibandingkan dengan tiap **case**, apabila cocok, maka blok **case** tersebut yang akan dijalankan.

Karena **case 7** kosong dan tidak ada perintah **break**, maka **case** berikutnya akan dijalankan, yaitu **case 8**.

Apabila nilai dari variabel \$nilai tidak ada nilai yang cocok pada ekspresi **case**, maka pernyataan pada blok **default** yang akan dikerjakan.

4.3.3 Perulangan while

Perintah **while** digunakan untuk melakukan perulangan selama nilai yang ada pada parameternya bernilai **true**. Struktur dasarnya seperti ini :

```

1 while( ekspresi) {
2     ...
3 }
```

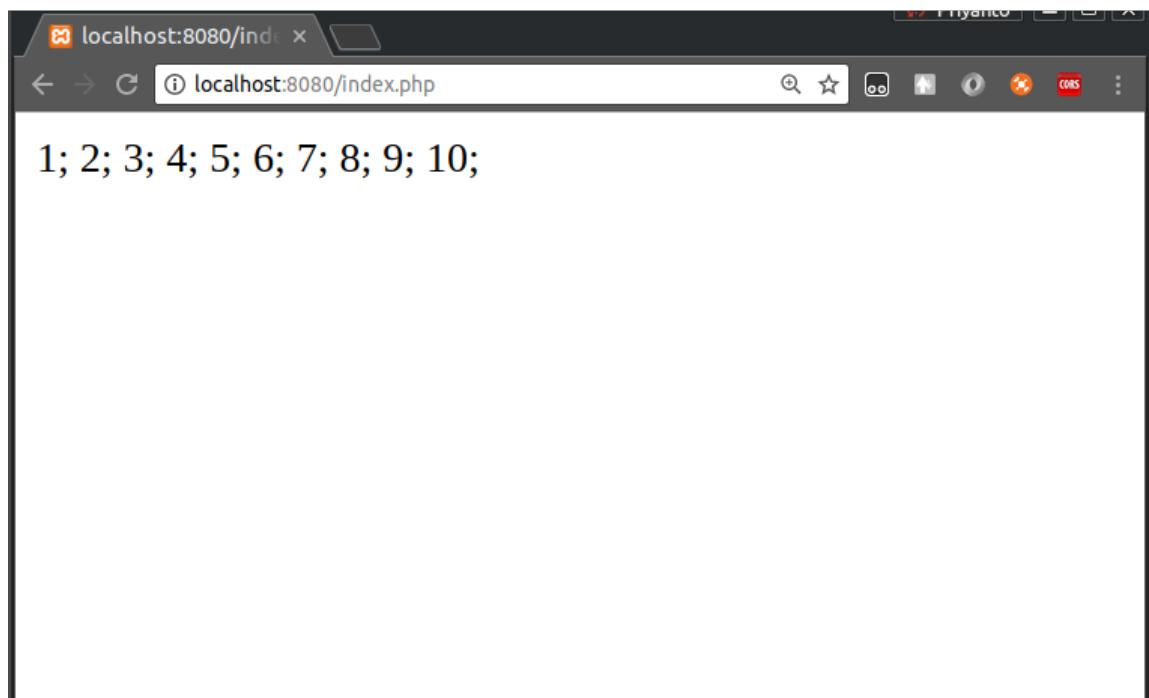
Nilai pada bagian `ekspresi` harus bernilai `true` agar perulangan dapat dijalankan, apabila dalam prosesnya nilai pada `ekspresi` bernilai `false` maka perulangan akan berhenti.

Contoh kode programnya adalah seperti berikut :

```

1 <?php
2 $nilai = 1;
3
4 while( $nilai <= 10) {
5     echo $nilai++ . " ";
6 }
7 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 4.5 berikut ini :



Gambar 4.5: Hasil Keluaran Contoh Kode `while`

4.3.4 Perulangan `do-while`

Perulangan dengan menggunakan `do..while` mirip seperti struktur `while`, hanya saja pemeriksaan `ekspresi` akan dilakukan setelah proses iterasi pertama berjalan, struktur kode secara sederhana dapat dituliskan seperti berikut :

```

1 do {
2 ...
3 } while( ekspresi);
```

Dimana nilai dari ekspresi agar iterasi berlanjut ke putaran ke-2 dan seterusnya harus bernilai `true`. Contoh implementasi kodennya adalah seperti berikut :

```

1 <?php
2 $nilai = 1;
3
4 do {
5     echo "nilai urut ke—" . $nilai++ . "<br>";
6 } while ($nilai <= 10)
7 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.6 berikut :



Gambar 4.6: Hasil Keluaran Kode `do..while`

4.3.5 Perulangan `for`

Penggunaan perintah `for` sama seperti `while` atau `do..while`, hanya saja strukturnya sedikit berbeda, mari kita lihat struktur dasar dari perintah `for` berikut :

```

1 for (eksp1; eksp2; eksp3) {
2 ...
3 }
```

Bagian `eksp1` akan dieksekusi di awal sebelum iterasi di mulai.

Begian `eksp2` adalah tempat pemeriksaan kondisi di awal tiap iterasi, pada iterasi pertama, tepat setelah `eksp1` dikerjakan. Bila `eksp2` bernilai `true` maka blok iterasi di bawahnya akan dikerjakan, namun apabila bernilai `false` maka blok iterasi di bawahnya tidak akan dijalankan.

Bagian `eksp3` akan dikerjakan di akhir tiap siklus iterasi.

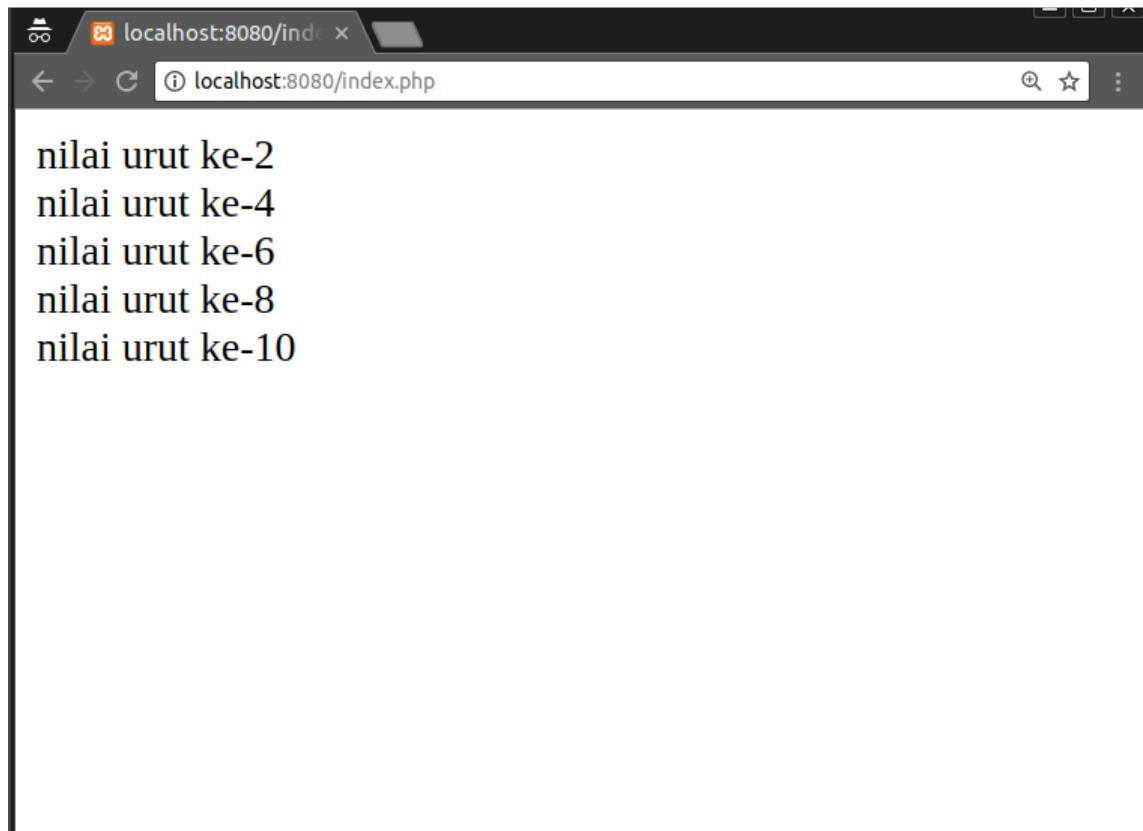
Contoh implementasi untuk perintah **for** ini adalah seperti berikut :

```

1 <?php
2 for ($nilai=1; $nilai <=10; $nilai++) {
3     if ($nilai % 2 == 0) {
4         echo " nilai urut ke-" . $nilai . "<br>" ;
5     }
6 }
7 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.7 berikut :



Gambar 4.7: Hasil Keluaran Kode **for**

4.3.6 Perulangan **foreach**

Perintah **foreach** digunakan untuk melakukan iterasi pada sebuah *array* / larik. Struktur perintahnya secara sederhana dapat dituliskan sebagai berikut :

```

1 foreach (array as $nilai) {
2 ...
3 }

```

atau seperti ini :

```

1 foreach (array as $kunci => $nilai) {
2 ...
3 }

```

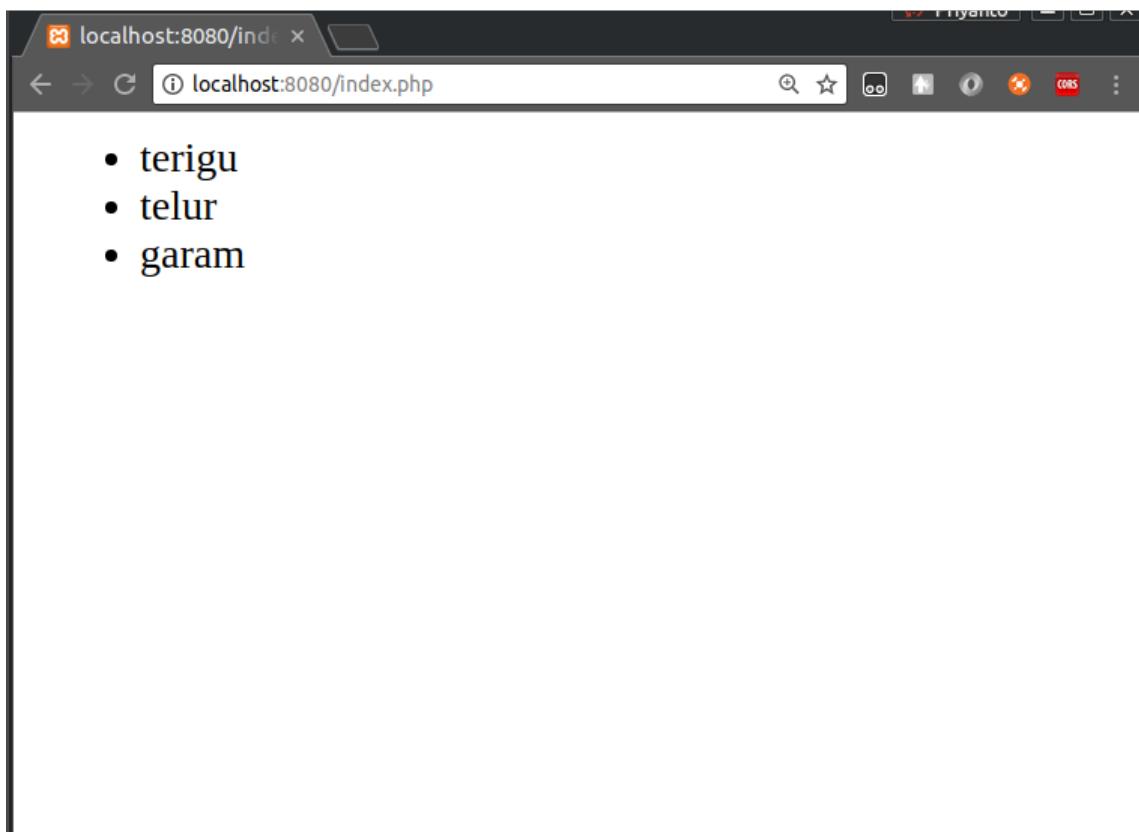
Nantinya tiap nilai yang ada pada *array* / larik akan dimasukkan kedalam variabel *\$nilai*, yang kemudian variabel ini dapat digunakan di dalam blok proses di bawahnya.

Struktur yang kedua sama saja, hanya saja isi dari *key* dari *array* akan dimasukkan ke dalam variabel *\$kunci*.

Mari kita coba implementasikan struktur perintah di atas ke dalam kode program seperti berikut :

```
1 <ul>
2 <?php
3 $data = array("terigu", "telur", "garam");
4
5 foreach($data as $nilai) {
6     echo "<li>$nilai</li>";
7 }
8 ?>
9 </ul>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.8 berikut :



Gambar 4.8: Hasil Keluaran `foreach` Tanpa *Key*

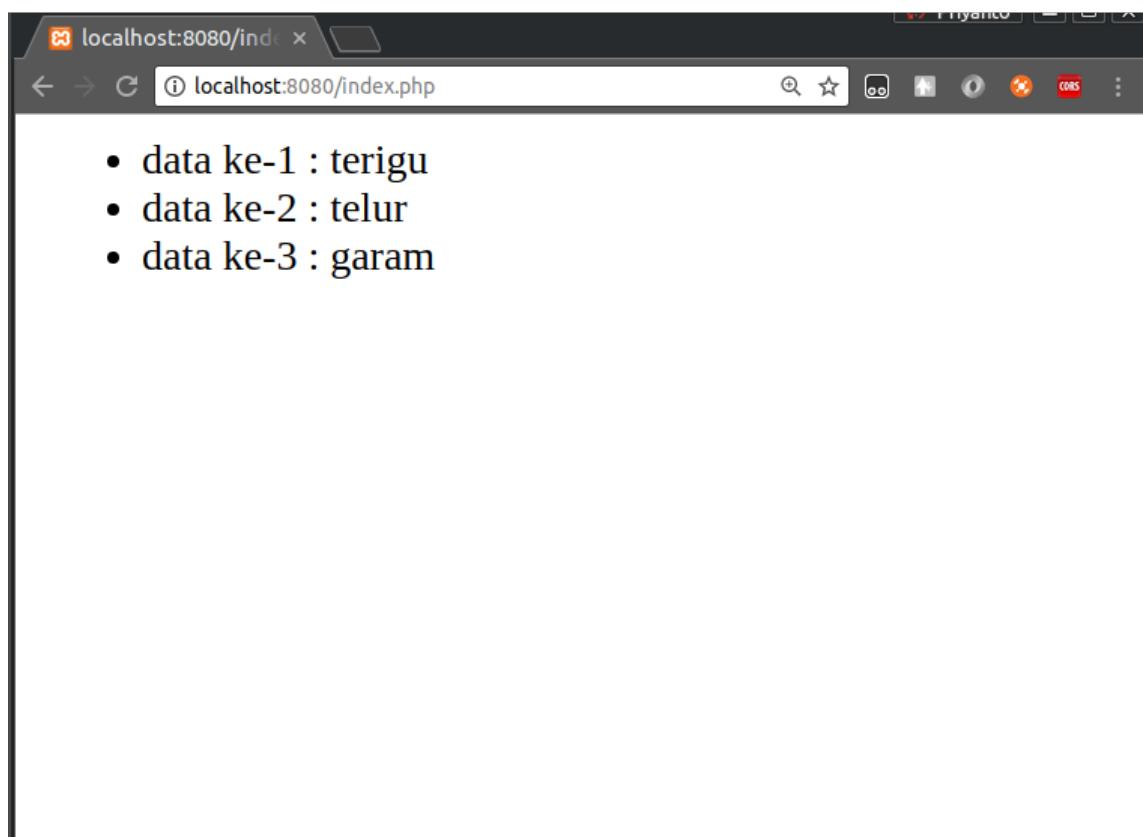
Apabila menggunakan *key* dari *array* maka kodennya dapat dibuat seperti berikut ini :

```

1 <ul>
2 <?php
3 $data = array("terigu", "telur", "garam");
4
5 foreach($data as $kunci => $nilai) {
6     echo "<li>data ke-". (++$kunci). " : $nilai</li>";
7 }
8 ?>
9 </ul>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.9 berikut :



Gambar 4.9: Hasil Keluaran `foreach` Dengan *Key*

4.3.7 Kontrol break

Perintah `break` digunakan untuk menghentikan eksekusi dari kontrol `switch`, `while`, `do-while`, `for`, dan `foreach`.

Perintah `break` ini dapat diberikan parameter berupa bilangan integer yang menandakan berapa tingkat kontrol yang akan dilakukan penghentian eksekusi.

Kita lihat terlebih dahulu bagaimana perintah `break` dapat menghentikan proses `switch` pada beberapa kode berikut :

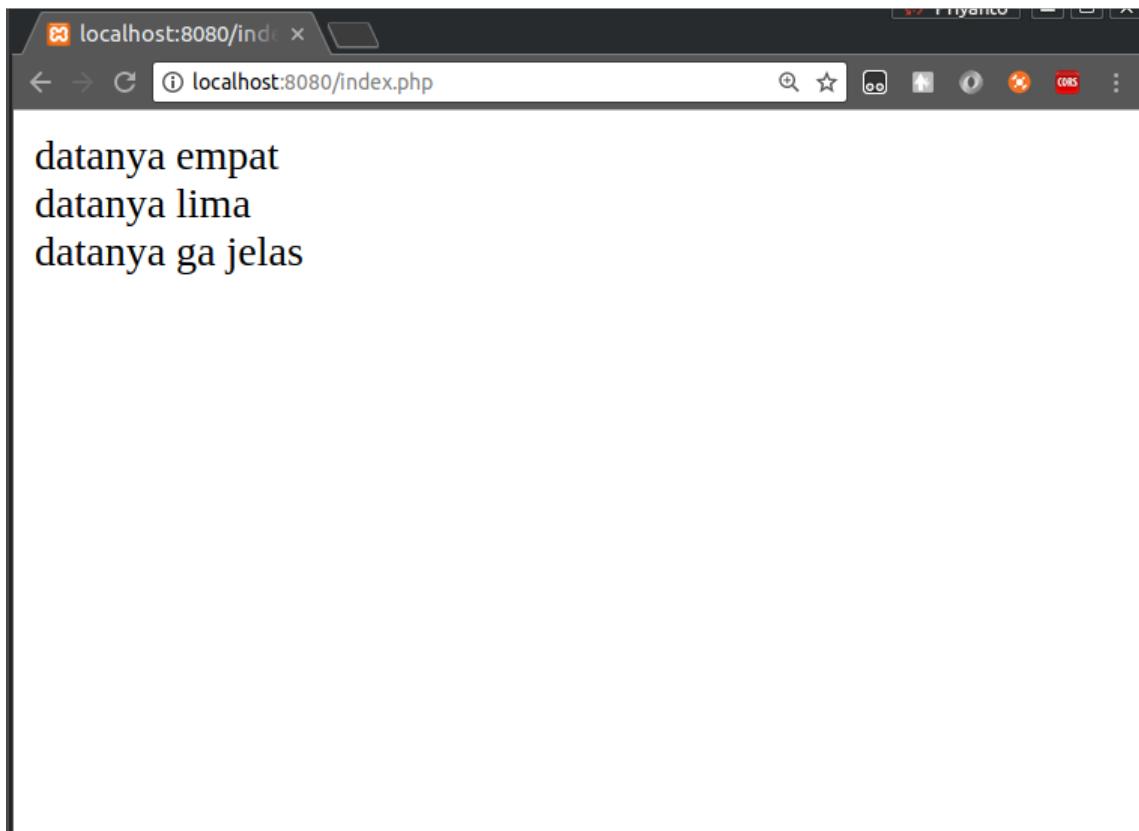
```

1 <?php
2 $data = 4;

```

```
3 switch($data) {  
4     case 4:  
5         echo "datanya empat<br>";  
6     case 5:  
7         echo "datanya lima<br>";  
8     default:  
9         echo "datanya ga jelas<br>";  
10    }  
11?  
12?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.10 berikut :



Gambar 4.10: Hasil Keluaran **switch** tanpa **break**

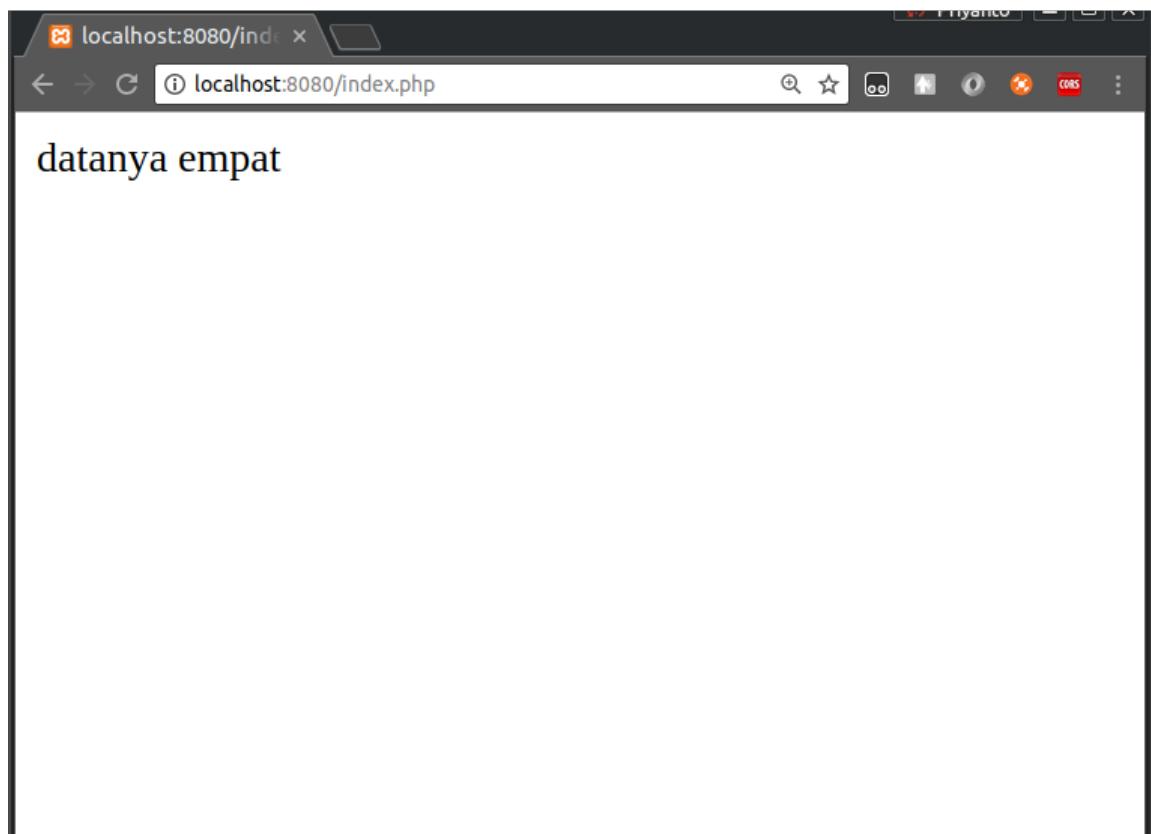
Kodenya akan terlihat lebih tepat apabila diberikan perintah **break** seperti berikut ini :

```
1 <?php  
2 $data = 4;  
3  
4 switch($data) {  
5     case 4:  
6         echo "datanya empat<br>";  
7         break;  
8     case 5:
```

```

9      echo "datanya lima<br>" ;
10     break;
11 default:
12     echo "datanya ga jelas<br>" ;
13     break;
14 }
15 ?>
```

Hasil keluaran kode di atas adalah seperti pada gambar 4.11 berikut ini :



Gambar 4.11: Hasil Keluaran **switch** dengan **break**

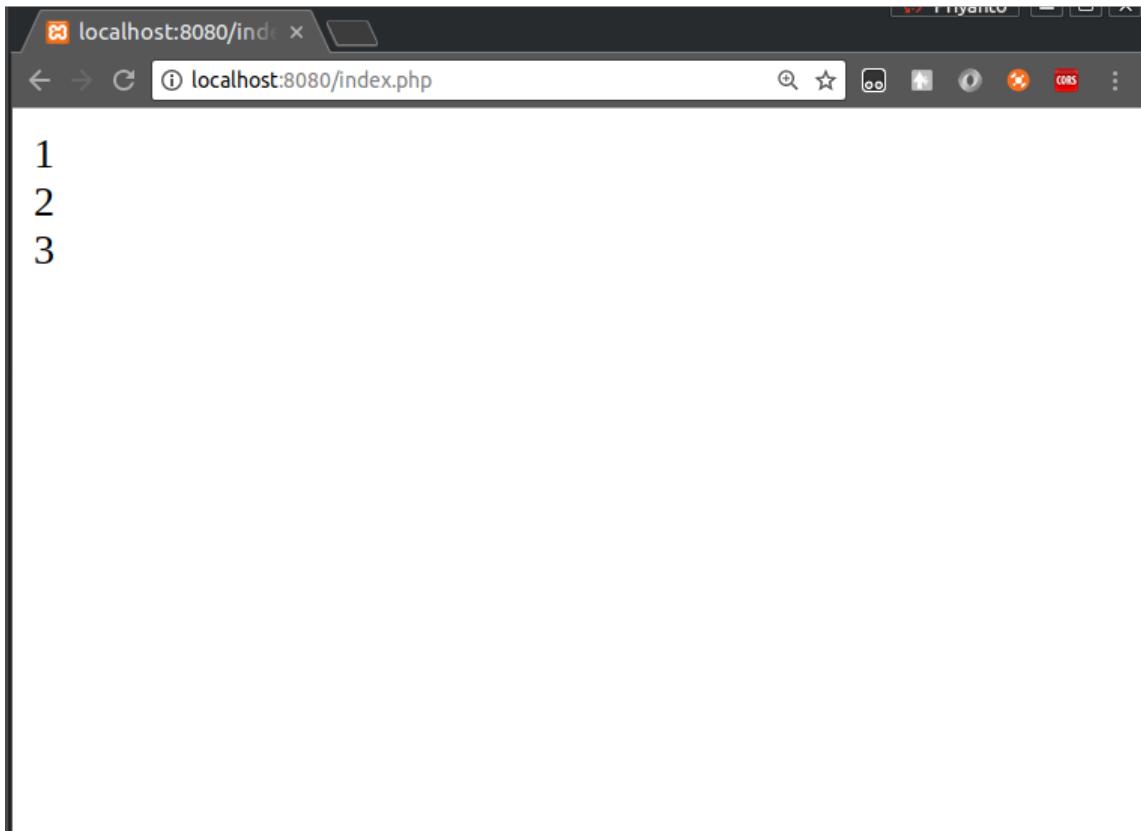
Ketika menggunakan **break** maka hanya **case** yang terseleksi saja yang akan dijalankan, sedangkan **case** berikutnya tidak akan dijalankan.

Kontrol **break** yang digunakan pada struktur **while** akan menghentikan proses iterasi yang sedang berjalan, contoh penggunaan kodenya adalah sebagai berikut :

```

1 <?php
2 $data = 4;
3 $counter = 1;
4
5 while($counter < 10) {
6     echo $counter++."<br>";
7     if($data == $counter) break;
8 }
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.12 berikut :



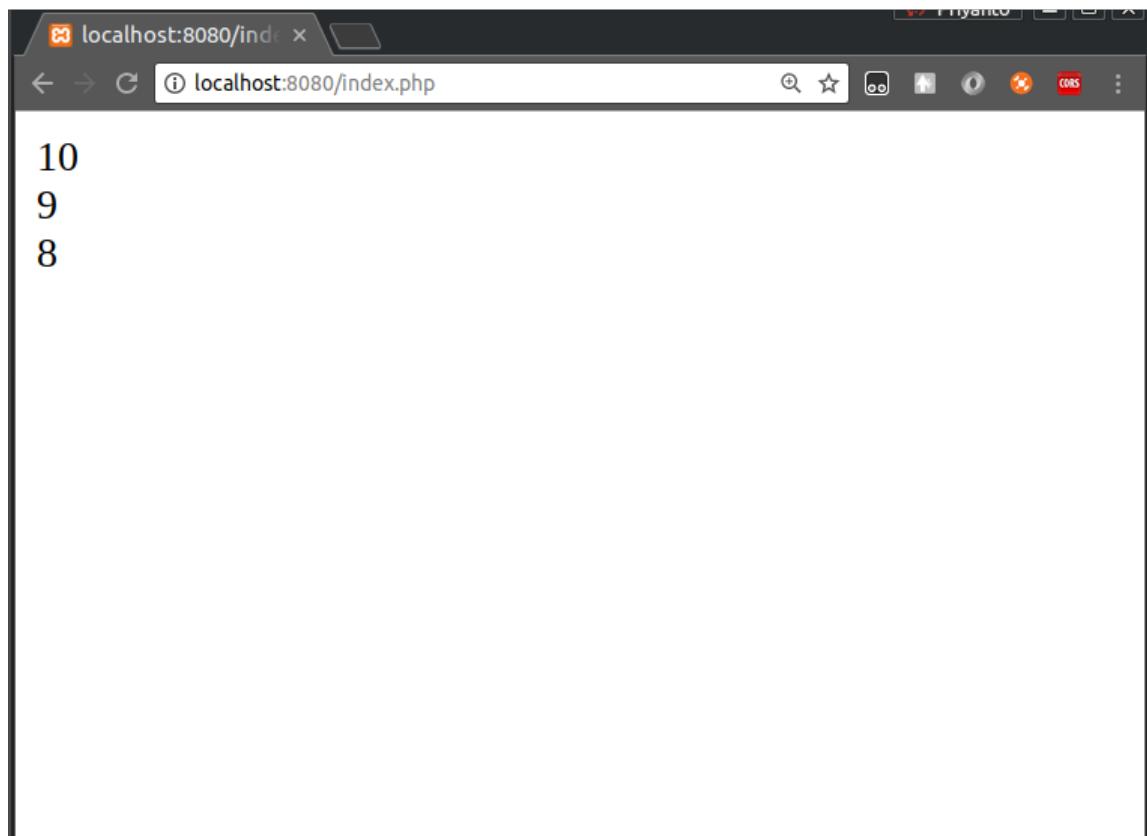
Gambar 4.12: Hasil Keluaran Penggunaan `break` pada `while`

Pada hasil keluaran, angka yang muncul hanya sampai dengan 3 (tiga), karena begitu nilai variabel `$counter` sama dengan `$data` (yang bernilai 4 (empat) maka iterasi `while` dipaksa untuk menghentikan proses iterasinya dengan `break`.

Proses penghentian iterasi pada `do-while` pun akan sama, mari kita perhatikan contoh kode program berikut :

```
1 <?php
2 $data = 7;
3 $counter = 10;
4
5 do {
6     echo $counter--."<br>";
7     if ($data == $counter) break;
8 } while($counter > 0);
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.13 berikut :

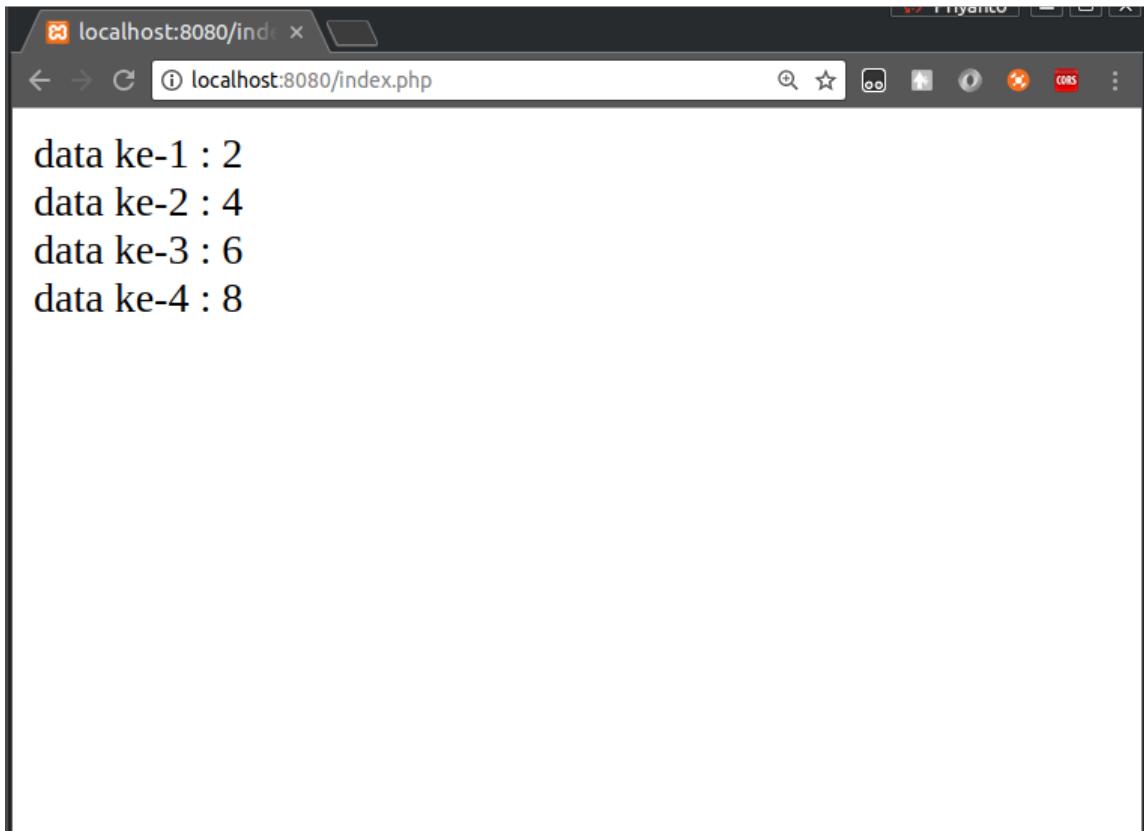


Gambar 4.13: Hasil Keluaran Kode `do-while` Dengan `break`

Untuk penggunaan perintah `break` pada iterasi `for` sama saja, mari kita perhatikan contoh kode program berikut :

```
1 <?php
2 for($counter=1; $counter < 100; $counter++) {
3     echo "data ke-". $counter . " : " . ($counter * 2) . "<br>";
4     if($counter == 4) break;
5 }
6 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.14 berikut :



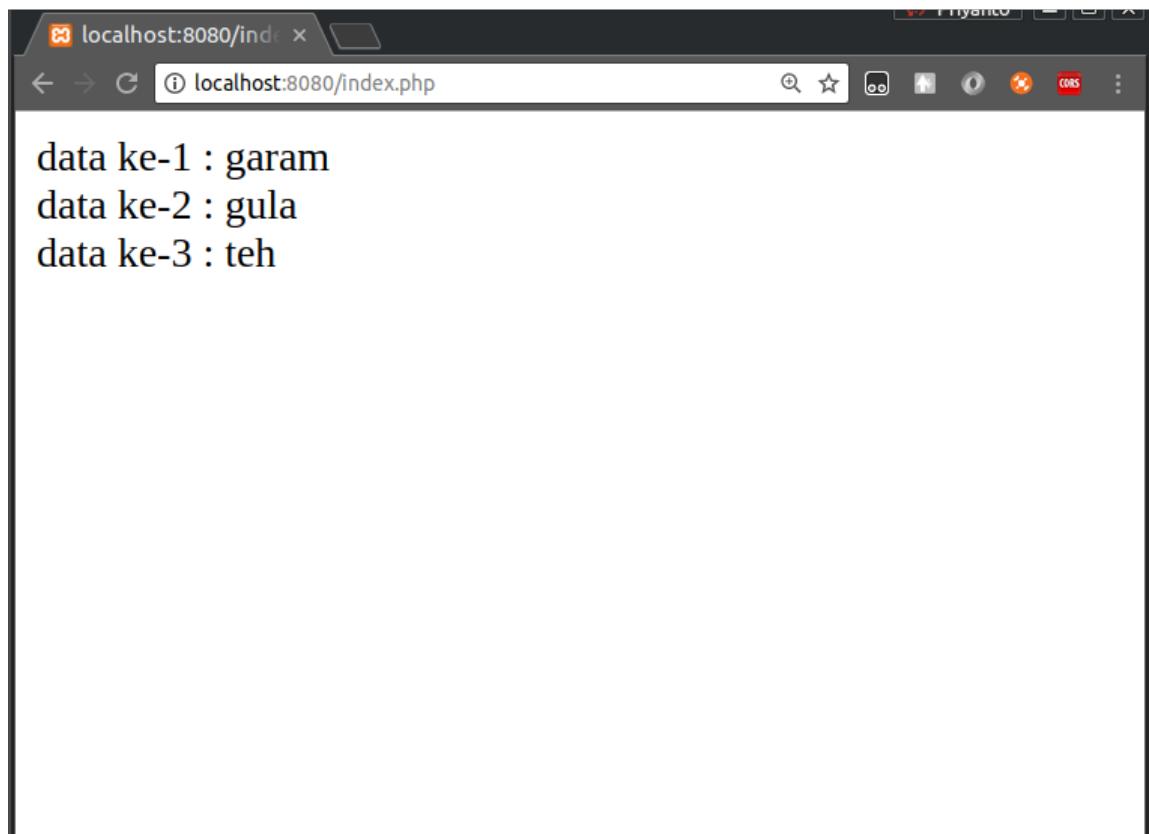
Gambar 4.14: Hasil Keluaran Penggunaan **break** pada **for**

Terlihat bahwa iterasi yang dilakukan hanya sampai iterasi ke-4, karena begitu nilai pada variabel **\$counter** bernilai 4 (empat) maka proses iterasi dihentikan oleh perintah **break**.

Untuk contoh kode **break** yang menghentikan struktur iterasi **foreach** dapat kita lihat pada baris kode berikut :

```
1 <?php
2 $data = array("garam" , "gula" , "teh" , "kopi");
3 $counter = 1;
4
5 foreach($data as $nilai) {
6     echo "data ke-". $counter++ . " : " . $nilai . "<br>";
7     if( $nilai == "teh" ) break;
8 }
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.15 berikut :



Gambar 4.15: Hasil Keluaran Penggunaan **break** Pada **foreach**

Pada *array* yang ada di kode, ada 4 (empat) komponen disana, tetapi yang dicetak hanya 3, karena pada iterasi yang mengeluarkan data **teh**, iterasi dihentikan dengan **break**.

4.3.8 Kontrol continue

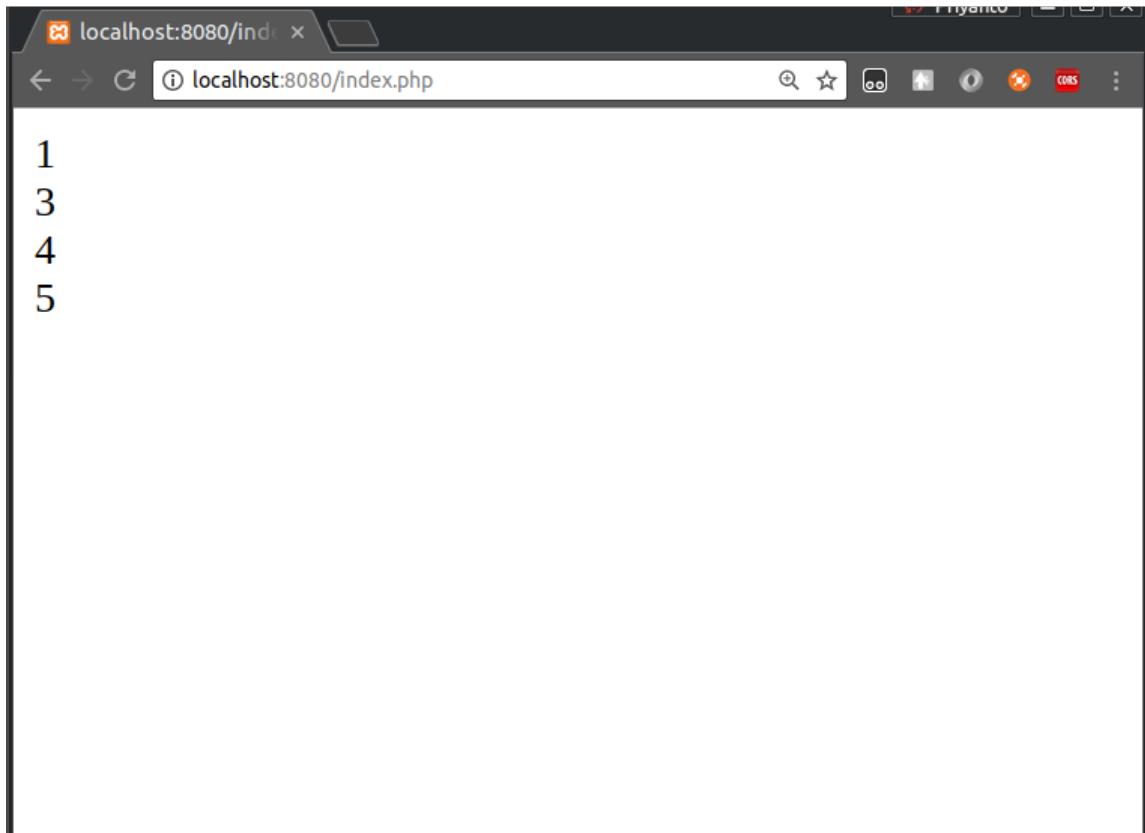
Penggunaan **continue** mirip dengan **break**, hanya saja tujuannya berbeda, pada penggunaan kontrol **continue** proses iterasi akan dilewatkan sekali ke proses iterasi berikutnya.

Mari kita perhatikan kode berikut ini :

```

1 <?php
2 for($counter=1; $counter < 6; $counter++) {
3     if($counter == 2) continue;
4     echo $counter."<br>";
5 }
6 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.16 berikut :



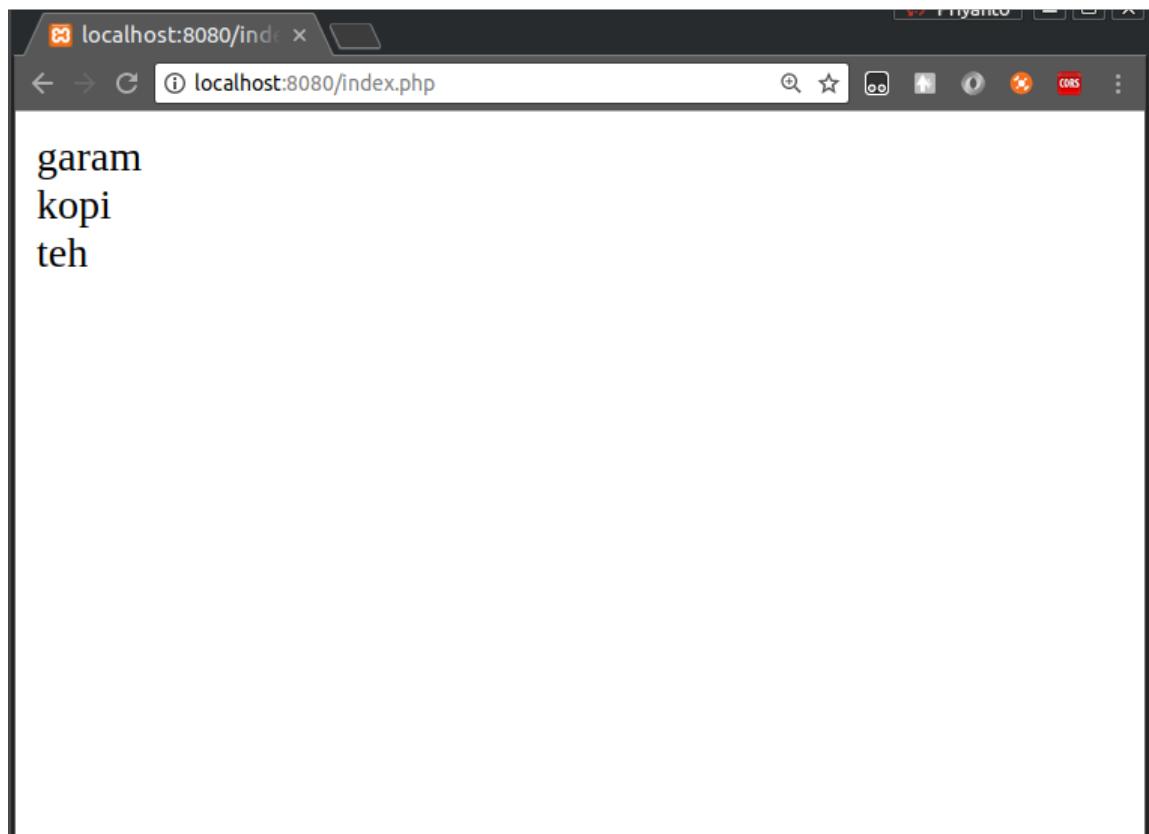
Gambar 4.16: Hasil Keluaran Penggunaan `continue` Pada `for`

Dari hasil keluaran di atas, angka 2 (dua) tidak dicetak ke *browser* karena pada saat nilai dari variabel `$counter` bernilai 2 (dua), ada perintah `continue` yang melewatkannya proses iterasi yang sedang berjalan ke proses iterasi berikutnya.

Untuk penggunaan pada `foreach`, mari kita lihat contoh kode berikut :

```
1 <?php
2 $data = array("garam" , "gula" , "kopi" , "teh");
3
4 foreach($data as $nilai) {
5     if($nilai == "gula") continue;
6     echo $nilai."<br>";
7 }
8 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 4.17 berikut :



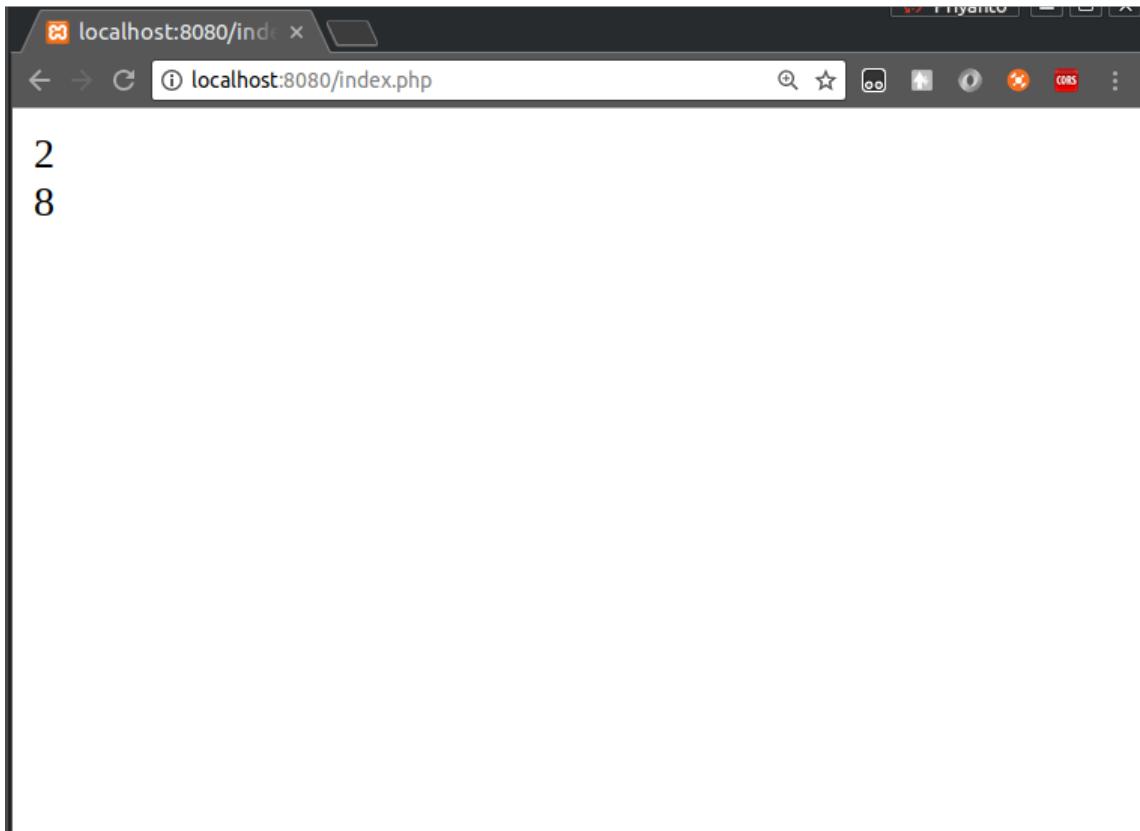
Gambar 4.17: Hasil Keluaran Penggunaan `continue` Pada `foreach`

Pada hasil keluaran tampak bahwa `gula` tidak ikut ditampilkan di *browser* karena dalam iterasinya, begitu ditemukan nilai berupa `gula` maka proses akan dilewati ke iterasi berikutnya.

Contoh penggunaan `continue` pada blok iterasi `while` adalah seperti berikut ini :

```
1 <?php
2 $counter = 1;
3 while(true) {
4     $counter *= 2;
5     if($counter == 4) continue;
6     if($counter > 15) break;
7     echo $counter."<br>";
8 }
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.18 berikut :



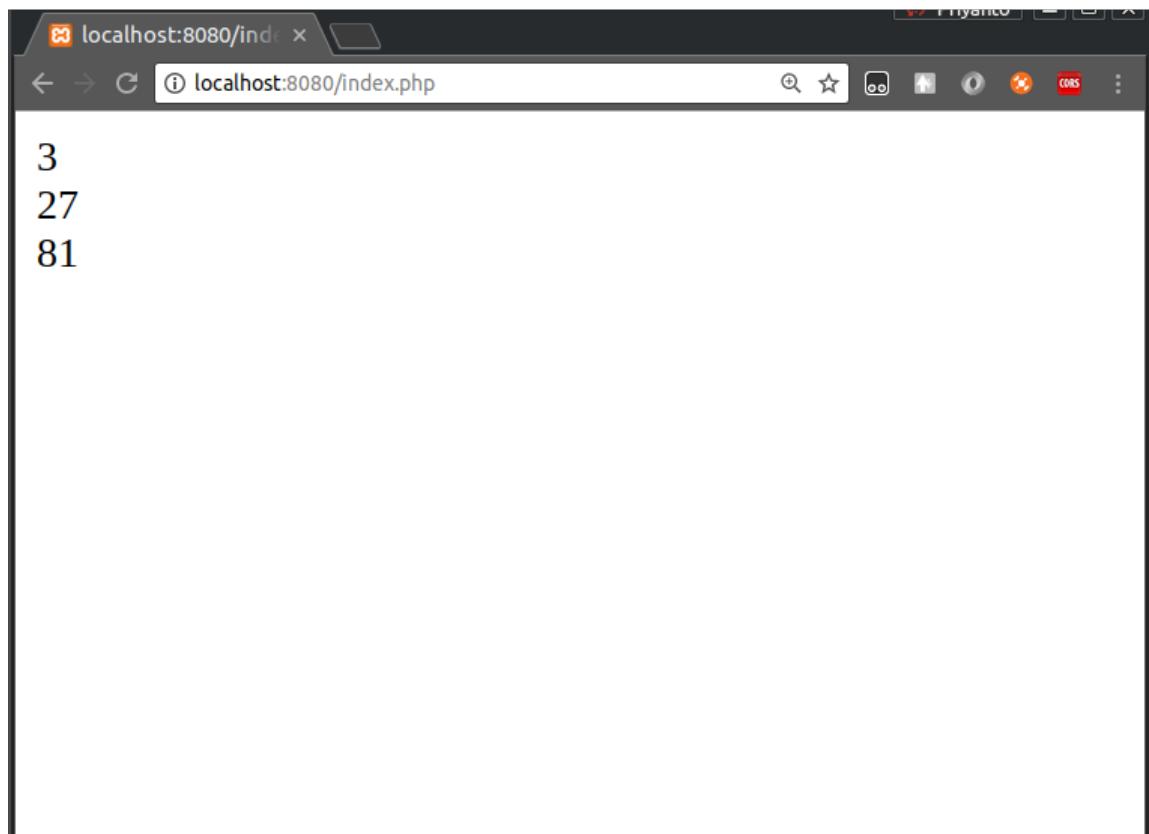
Gambar 4.18: Hasil Keluaran Penggunaan `continue` Pada `while`

Dari deretan angka di atas, angka 4 (empat) akan dilewatkan karena begitu nilai dari variabel `$counter` bernilai 4 (empat) diberikan perintah `continue` untuk melewatkannya proses dibawahnya sehingga masuk ke proses iterasi berikutnya.

Contoh penggunaan `continue` pada blok iterasi `do-while` adalah sebagai berikut :

```
1 <?php
2 $counter = 1;
3 $ok = true;
4
5 do {
6     $counter *= 3;
7     if ($counter == 9) continue;
8     $ok = ($counter > 30) ? false : true;
9     echo $counter."<br>";
10 } while($ok);
11 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.19 berikut :



Gambar 4.19: Hasil Keluaran Penggunaan `continue` Pada `do-while`

Hasil keluaran di atas akan melewatkkan proses pencetakan angka 9 (sembilan) dan dilanjutkan ke iterasi berikutnya.

4.3.9 Kontrol include dan require

Perintah `include` digunakan untuk memasukan dan mengevaluasi *file* yang ada pada parameter `include` yang isinya dapat digunakan pada *file* saat ini.

Apabila *file* yang disertakan pada parameter tidak ditemukan, maka perintah ini akan menghasilkan *warning*. Mari kita perhatikan kode berikut :

```

1 <?php
2 echo "Hai, ". $nama . "<br><br>";
3
4 include "data.php";
5 echo $nama . " baru keluar";
6 ?>

```

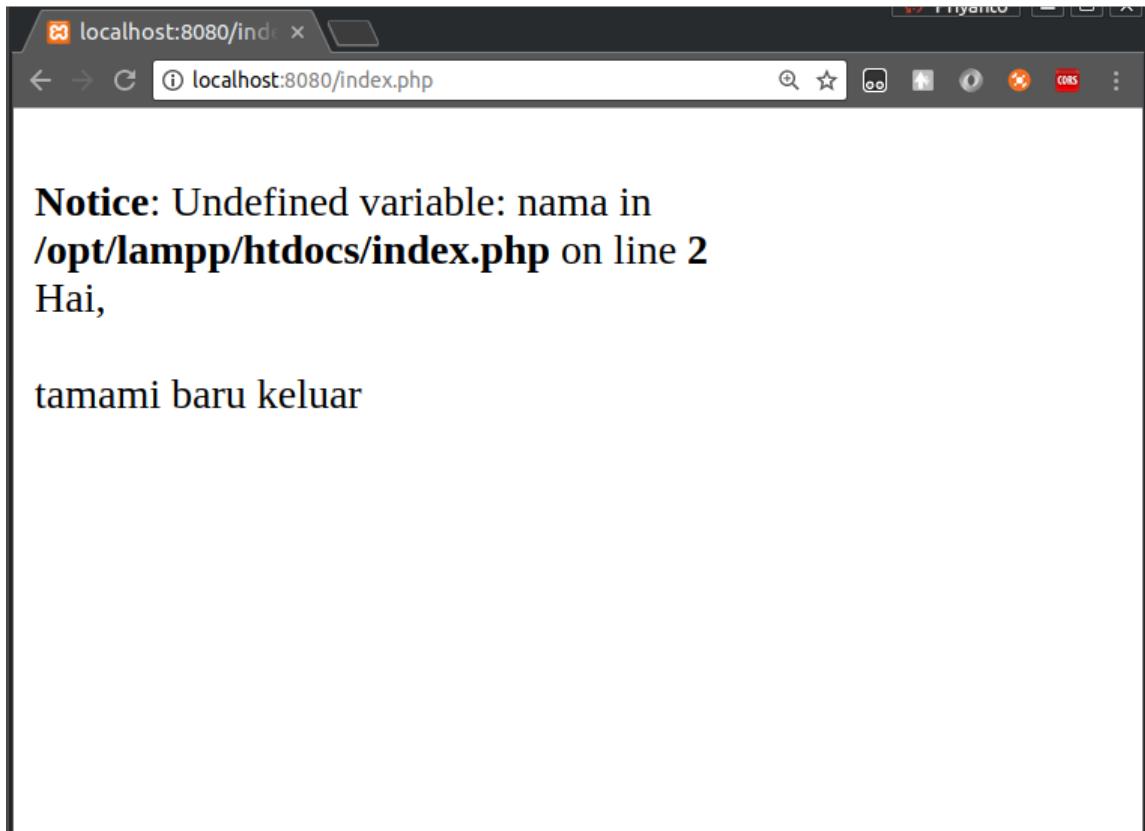
Isi dari kode di atas kita simpan dalam *file* dengan nama `index.php`, selanjutnya kita buat *file* baru dengan nama `data.php` yang disimpan dalam satu *folder* yang sama dengan `index.php`, isi kode dari *file* `data.php` adalah sebagai berikut :

```

1 <?php
2     $nama = "tamami";
3 ?>

```

Hasil keluaran saat kita melakukan akses ke `localhost/index.php` adalah seperti pada gambar 4.20 berikut :



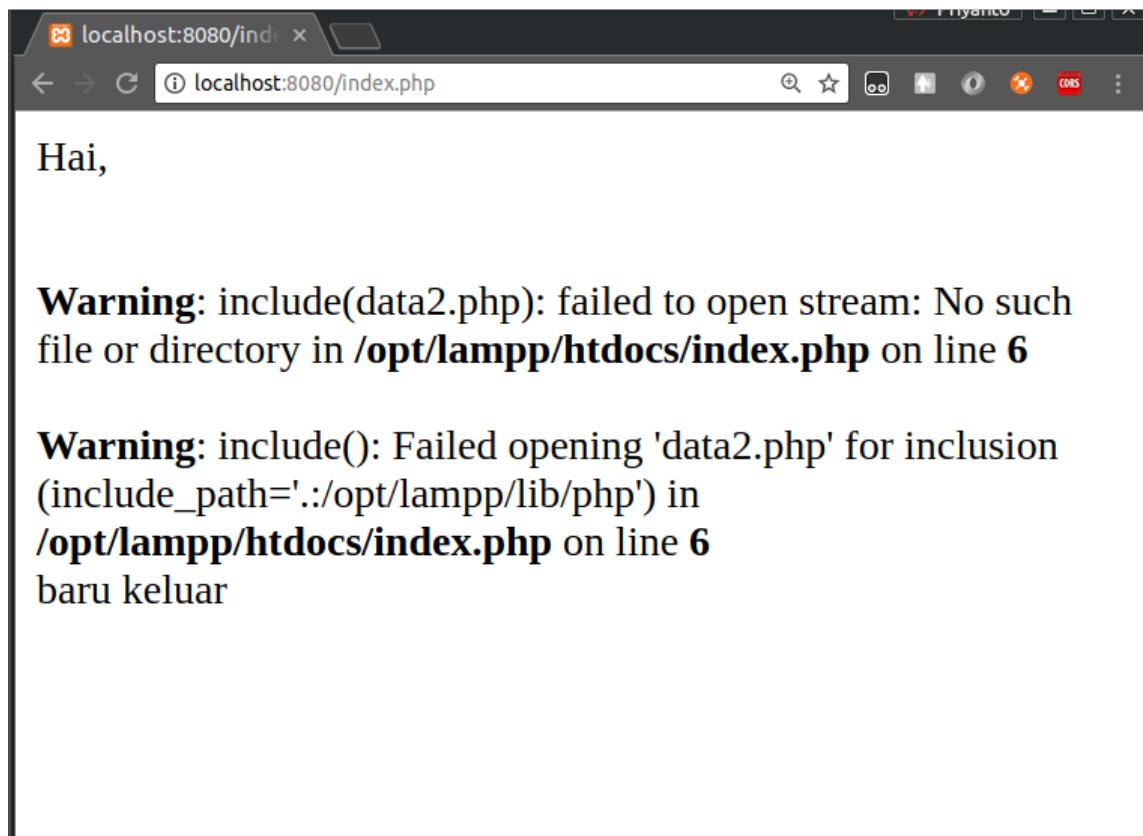
Gambar 4.20: Hasil Keluaran Contoh Penggunaan `include`

Pada gambar 4.20, pemanggilan variabel `$nama` pada baris ke-2 mengalami kegalan, karena perintah `include` baru dijalankan pada baris berikutnya, dan pada baris ke-5 variabel `$nama` berhasil dipanggil.

Sekarang coba kita ubah kode pada file `index.php` menjadi seperti berikut ini :

```
1 <?php
2 $nama = "";
3
4 echo "Hai, " . $nama . "<br><br>";
5
6 include "data2.php";
7 echo $nama . " baru keluar";
8 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.21 berikut :



Gambar 4.21: Muncul *Warning* Saat *File* Tidak Ditemukan

Pada gambar 4.21 akan keluar *warning* bahwa *file* data2.php tidak ditemukan, tetapi proses tetap dijalankan.

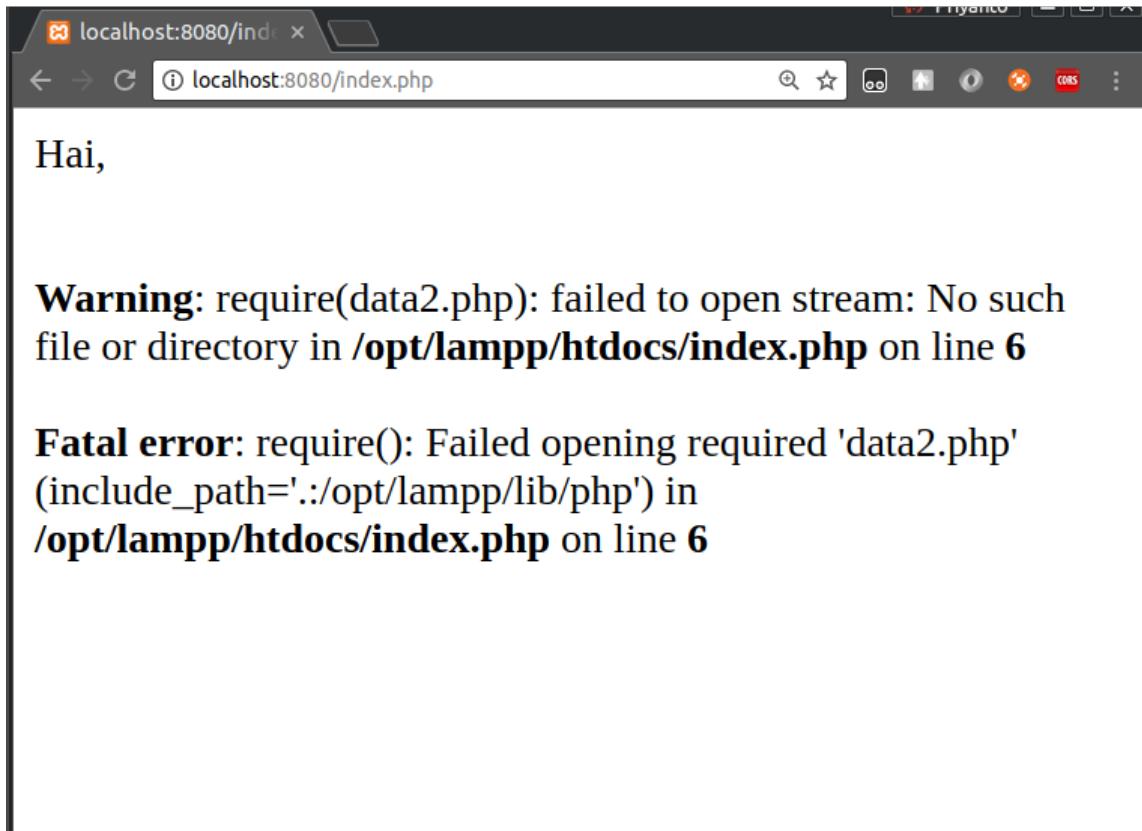
Disinilah letak bedanya antara penggunaan **include** dengan **require**, fungsi keduanya sama-sama akan memasukkan data dari *file* lain, namun bedanya, saat **require** tidak menemukan *file* yang dicari, maka perintah **require** akan mengembalikan pesan **error** dan menghentikan prosesnya. Perhatikan kode dari *file* index.php berikut :

```

1 <?php
2 $nama = " ";
3
4 echo "Hai , " . $nama . "<br><br>" ;
5
6 require "data2 . php";
7 echo $nama . " baru keluar" ;
8 ?>

```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 4.22 berikut :



Gambar 4.22: Hasil Keluaran Kesalahan Saat Menggunakan `require`

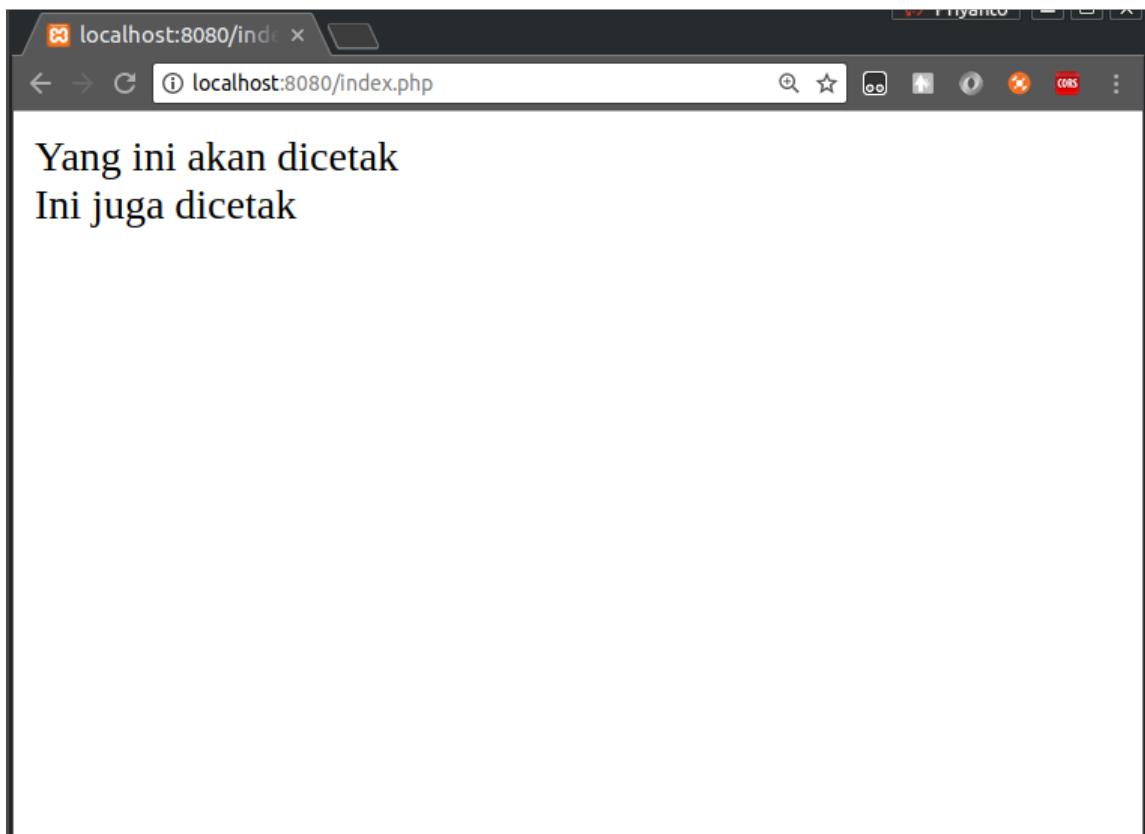
Perhatikan bahwa di akhir baris pada *browser* tidak mengeluarkan teks baru **keluar** seperti pada kode sebelumnya, ini karena perintah `require` tidak menemukan *file data2.php* dan langsung menghentikan proses dengan mengeluarkan pesan **fatal error**.

4.3.10 Kontrol goto

Perintah `goto` digunakan untuk melompat ke nama label tertentu. Mari kita lihat contoh kodennya berikut :

```
1 <?php
2 goto cetakDonk;
3 echo "Teks ini ga akan dicetak<br>";
4 echo "teks ini pun akan terlewat<br>";
5
6 cetakDonk:
7 echo "Yang ini akan dicetak<br>";
8 echo "Ini juga dicetak";
9 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.23 berikut :



Gambar 4.23: Hasil Keluaran Menggunakan Perintah `goto`

Perhatikan bahwa perintah pada baris ke-3 dan ke-4 tidak pernah dikerjakan karena kodenya akan dilewatkan ke baris 6 dan selanjutnya.

4.4 Kesimpulan

Dengan menggunakan beberapa perintah kontrol, kita dapat mengatur alur kode program apakah akan berjalan bercabang, berjalan berulang dengan iterasi, atau akan melakukan lompatan kode ke label tertentu.

4.5 Tugas

Langkah dari tugas kali ini adalah sebagai berikut :

1. Buatlah sebuah variabel `$username` lalu isikan namanya bebas.
2. Buatlah sebuah variabel `$daftar` yang berisi daftar 5 (lima) *username* dalam *array*.

3. lakukan iterasi terhadap variabel `$daftar` kemudian cocokan isinya satu per satu dengan variabel `$username`, bila cocok tampilkan pesan `Selamat Datang`, bila tidak ada yang cocok maka tampilkan pesan `Anda belum terdaftar`.

