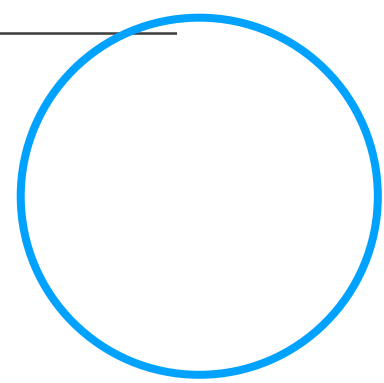


Pengaruh • Prepemrosesan dan Argumen Baris Perintah

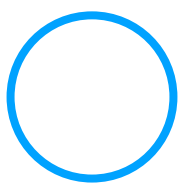
TIM DOSEN





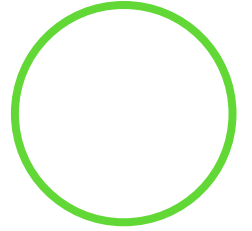
OUTLINE



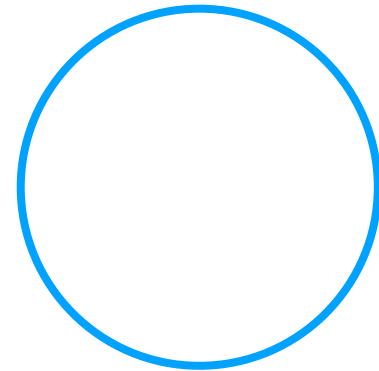
- a. Pengenalan Pengarah Praprosessor
 - b. Penggantian Macro
 - c. Penyertaan File Judul
 - d. Kompilasi Bersyarat
 - e. Penggunaan *#line*
 - f. Penggunaan *#pragma*
 - g. Penggunaan *#error*
 - h. Nama Makro Bawaan
- 



Pengenalan Pengarah Preprocessor




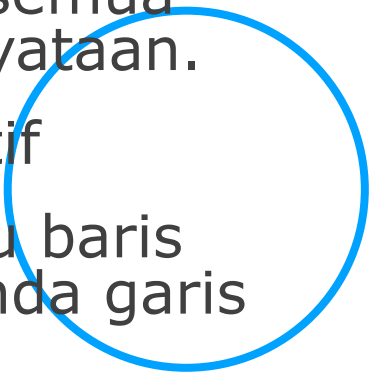

- **Preprocessor** : suatu program yang berperan penting dalam kontrol kompilasi program.
- **Preprocessor** , membaca perintah **pre-processor directive**.
- Pengarah preprosesor dipakai untuk membaca file yang berisi deklarasi fungsi dan definisi konstanta.
- Didahului oleh tanda hash (#). preprocessor dijalankan sebelum kompilasi kode yang sebenarnya dimulai, maka preprocessor mencerna semua petunjuk-petunjuk ini sebelum kode yang dihasilkan oleh pernyataan.
- macam pre-processor :
 - `#include`
 - `#define`
 - `#If --- #endif`
 - `#If --- #else --- #endif`
 - `#elif`
 - `#undef`
 - `#ifdef ... #ifndef, dll.`





PENGENALAN PENGARAH PRAPROCESSOR



- **Preprocessor** : suatu program yang berperan penting dalam kontrol kompilasi program.
 - **Preprocessor** , membaca perintah **pre-processor directive**.
 - Pengarah preprosesor dipakai untuk membaca file yang berisi deklarasi fungsi dan definisi konstanta.
 - Didahului oleh tanda hash (#). preprocessor dijalankan sebelum kompilasi kode yang sebenarnya dimulai, maka preprocessor mencerna semua petunjuk-petunjuk ini sebelum kode yang dihasilkan oleh pernyataan.
 - Tidak ada titik koma (;) pada akhir sebuah preprocessor direktif
 - Preprocessor direktif dapat diperpanjang melalui lebih dari satu baris dengan didahului karakter newline pada akhir baris dengan tanda garis miring terbalik (\)
- 
- 
- 

PENGGANTIAN MACRO●

- Preprocessor macro menggunakan kata kunci `#define`
- Preprocessor direktif ini menggantikan setiap terjadinya pengenalan di seluruh kode oleh pengganti. Penggantian ini dapat menjadi ungkapan, pernyataan, sebuah blok atau **alias**.
- Penggunaan `#define` sebagai definisi konstan dan juga bias digunakan untuk menformulasikan nama fungsi
- Sintak :

`#define` **identifier** pengganti

- Contoh : `#define` **TABLE_SIZE** 100 `#define` **KURANG(x,y)** (x-y)




`int table1[TABLE_SIZE];`

Setelah preprocessor telah menggantikan `TABLE_SIZE`, kode menjadi MENJADI : `int table1[100];`



PENYERTAAN FILE JUDUL



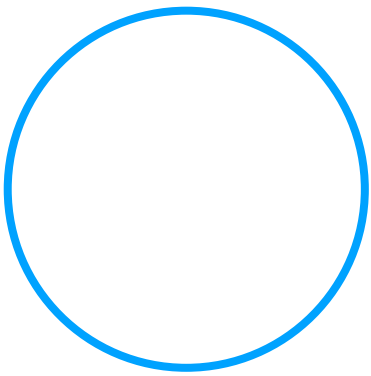


- sebuah preprocessor directive `# include` itu menggantikan dengan seluruh isi file yang ditentukan.
 - Ada dua cara untuk menentukan file yang akan disertakan:
 1. `#include "file"`
 2. `#include <file>`
 - Perbedaan antara kedua ekspresi tersebut adalah tempat (direktori) dimana compiler akan mencari file.
- 
- 
- 



PENYERTAAN FILE JUDUL – #INCLUDE “FILE”



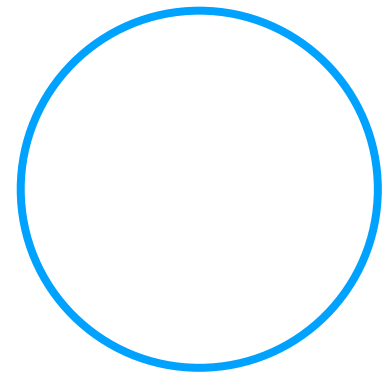



- Penyertaan file yang menggunakan tanda kutip ganda(“”).
 - file dicari dalam direktori yang sama yang memuat file yang berisi petunjuk.
 - kompiler pencarian file berada dalam direktori default tempat file konfigurasi dibuat.
 - File header lebih spesifik dan dapat dibuat sendiri.
 - Contoh : #include “operasi.h”
- 
- 
- 



PENYERTAAN FILE JUDUL – #INCLUDE <FILE>



- Penyertaan file yang menggunakan tanda <>.
 - File dicari secara langsung di mana kompilator dikonfigurasi untuk mencari file header standar.
 - Contoh : #include <iostream>
- 
- 
- 
- 

KOMPILASI BERSYARAT

- Penyertaan berkondisi seperti pernyataan IF,
- Hal tersebut dapat digunakan untuk menentukan kondisi yang ada dan berisi beberapa fungsi dalam file tersebut.
- Hal ini dapat dilakukan dengan perintah `#if`, `#elif`, `#else`, `#ifdef` atau `#ifndef` dan ditutup dengan `#endif`.
- Instruksi-instruksi processor tersebut digunakan pada beberapa porsi variasi compile program secara selektif.
- Logika yang umum adalah bahwa jika suatu ekspresi setelah `#if`, `#ifdef` atau `#ifndef` adalah benar, maka kode yang berada diantara salah satu dari mereka dan `#endif`, akan dieksekusi tapi jika yang terjadi sebaliknya, maka akan terlewati.
- `#endif` digunakan untuk menandai akhir suatu blok.
- Instruksi `#else` dapat digunakan dengan salah satu dari mereka dengan cara yang sama, yaitu dengan "else" pada perintah C/C++.



KOMPILASI BERSYARAT

➤ Bentuk :
//if elif dan else
#if kondisi
//aksi
#elif kondisi
//aksi
#else
//aksi
#endif

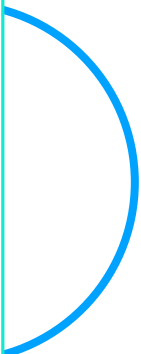

Contoh :

```
#if BIL<0 //if
#define BILANGAN Negatif
#elif BIL>0 //else if
#define BILANGAN Positif
#else //else
#define BILANGAN Nol
#endif
```

```
#include <iostream>
using namespace std;
```

```
int main(){
    //pemanggilan macro BILANGAN
    cout<<"BILANGAN "<<(BILANGAN)<<endl;

    return 0;
}
```



PENGUNAAN *#LINE*

- Penyusunan program dan beberapa error yang terjadi selama proses kompilasi, compiler menunjukkan pesan kesalahan dengan referensi ke nama file di mana kesalahan terjadi dan nomor baris, sehingga lebih mudah untuk menemukan kode yang menghasilkan kesalahan.
- *#line* direktif memungkinkan untuk mengontrol kedua hal itu, nomor baris dalam file kode serta nama file yang diinginkan muncul bila terjadi kesalahan.

➤ Sintak:

#line number "filename" *#nomor baris* "filename"

- Nomor mana adalah nomor baris baru yang akan ditetapkan ke baris kode berikutnya.
- "filename" adalah parameter opsional yang memungkinkan untuk mendefinisikan kembali nama file yang akan ditampilkan.

➤ Contoh:


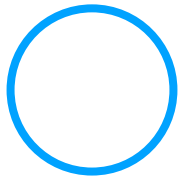
➤ *#line* 50 "assigning variable" *#line* 50 "menugaskan variabel"

➤ `int a?; int a?;`

- Kode ini akan menghasilkan kesalahan yang akan ditampilkan sebagai error in file "menetapkan variabel", line 50.



PENGUNAAN *#PRAGMA*

- Direktif ini digunakan untuk menentukan pilihan beragam compiler.
 - Pilihan ini adalah spesifik untuk platform dan compiler yang digunakan.
 - Jika compiler tidak mendukung argumen khusus untuk *#Pragma*, itu diabaikan – tidak ada kesalahan yang dihasilkan.
- 
- 




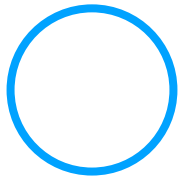
PENGUNAAN *#ERROR*

- Direktif ini Aborts proses kompilasi ketika ditemukan, menghasilkan sebuah kompilasi kesalahan yang dapat ditentukan sebagai parameter:

```
#ifndef __cplusplus
```

```
    #error A C++ compiler is required!
```


```
#endif
```

- Contoh tersebut Aborts proses kompilasi jika nama makro `__cplusplus` tidak didefinisikan (nama makro ini didefinisikan secara default di semua C ++ compiler).
- 
- 



NAMA MAKRO BAWAAN

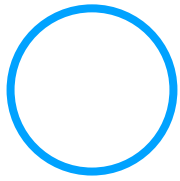
- Beberapa makro telah disediakan oleh C/C++,
- makro tersebut diawali dengan 2 tanda garis bawah (__)
- Contoh : __TIME__, __DATE__ , __FILE__ , dll



```
#include <iostream>
using namespace std;

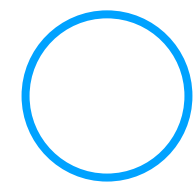
int main(){
    cout<<"File : "<<__FILE__<<endl;
    cout<<"Tanggal : "<<__DATE__<<endl;

    return 0;
}
```





QUESTIONS





TUGAS LMS

