

MODUL PRAKTIKUM

# PEMROGRAMAN C

PENGENALAN BAHASA PEMROGRAMAN C  
TEKNIK INFORMATIKA



AH-201302-C

**PROGRAM STUDI SARJANA-1 TEKNIK INFORMATIKA**

---

**MODUL PRAKTIKUM  
BAHASA PEMROGRAMAN C**

**ARWIN HALIM, S. KOM., M.KOM.**

**STMIK MIKROSKIL**

JL. THAMRIN NO 112, 124, 140.  
MEDAN

<http://www.mikroskil.ac.id>

## Daftar Isi

Modul I Pengenalan DevC++ dan Pemrograman C.....	1
Modul II Variabel, Konstanta, dan Operator Aritmatika.....	10
Modul III Struktur Keputusan.....	17
Modul IV Struktur Perulangan .....	23
Modul V Array .....	29
Modul VI String .....	36
Modul VII Fungsi .....	42
Modul VIII Pointer.....	48
Modul IX Pointer Lanjutan .....	53
Modul X Command Line Arguments.....	58
Modul XI Struct .....	62
Modul XII File .....	66
LAMPIRAN .....	L-1

# Modul I

## Pengenalan DevC++ dan Pemrograman C

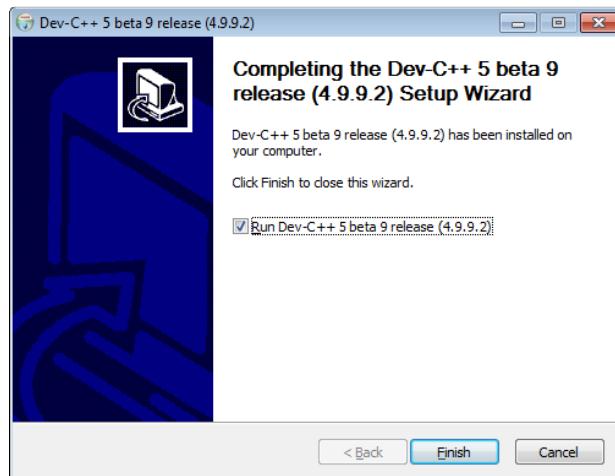
### Objektif

Memperkenalkan Lingkungan IDE DevC++ dan Pemrograman C.

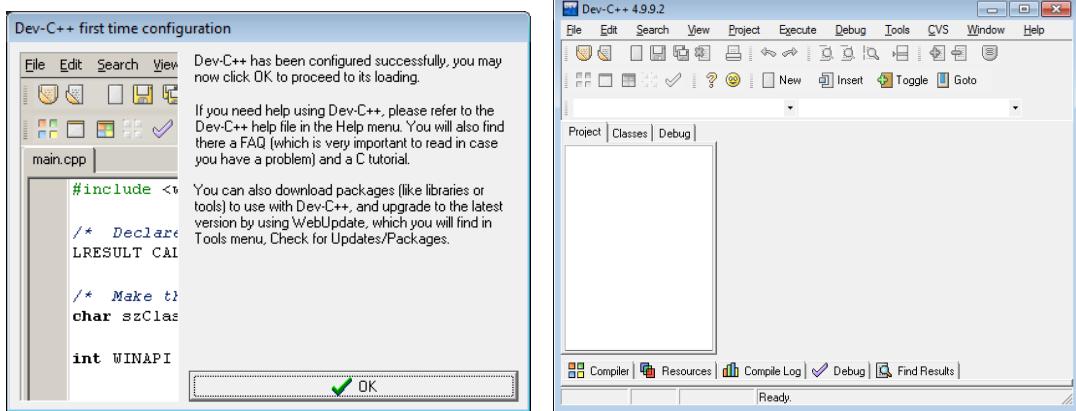
### Teori

Lingkungan Pengembangan – *Integrated Development Environment* (IDE).

DevC++ merupakan salah satu IDE yang dapat digunakan sebagai editor untuk menghasilkan dan mengeksekusi program yang ditulis dengan bahasa pemrograman C dan C++. IDE ini dapat diperoleh secara gratis pada <http://www.bloodshed.net/devcpp.html>. Pada buku panduan ini, kita akan menggunakan editor DevC++ versi 4.9.9.2. Setelah mengunduh berkas instalasi, ikuti langkah-langkah **instalasinya** sampai selesai dan muncul tampilan sebagai berikut.

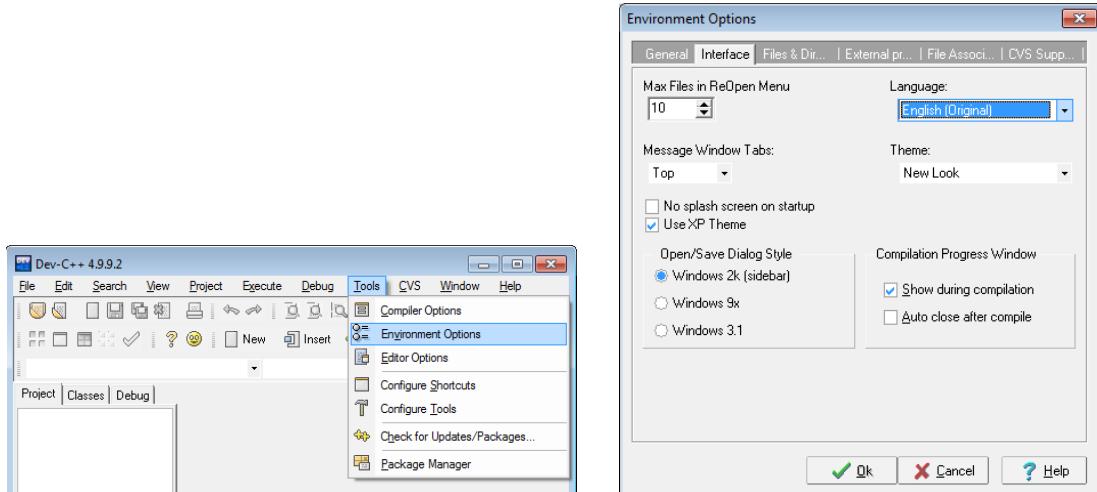


Tekan tombol '**Finish**'. Pada saat pertama kali editor digunakan, kita harus melakukan pengaturan awal untuk editor. Klik tombol '**Next**' sampai tampilan awal editor DevC++ dengan theme "New Look" muncul pada layar.

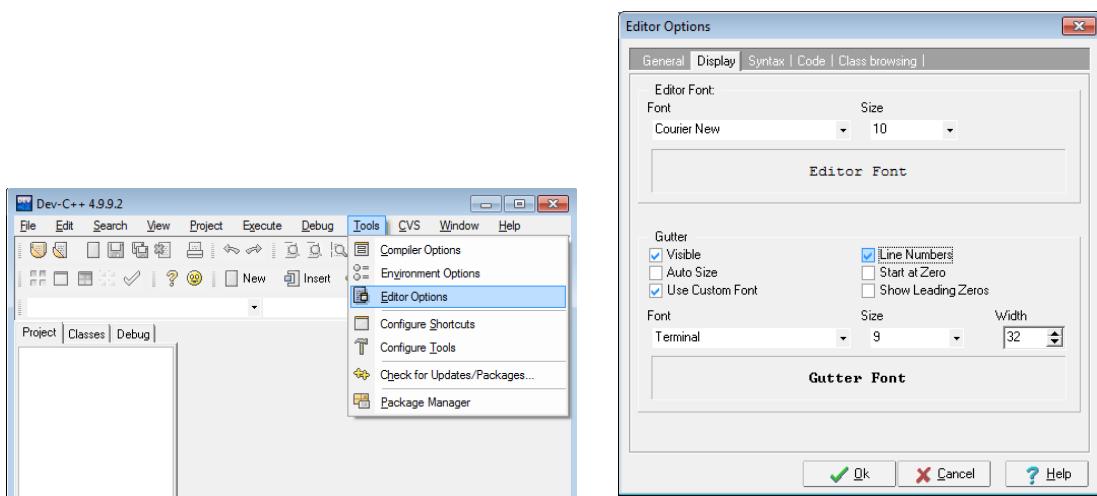


Pengaturan awal yang telah dilakukan masih dapat disesuaikan kembali pada editor DevC++. Semua pengaturan dapat dilakukan pada menu '**Tools**'. Salah satu pengaturan yang dapat dilakukan adalah pengaturan lingkungan IDE dan pengaturan Editor.

*Pengaturan Lingkungan IDE* dapat dilakukan dengan memilih menu '**Tools**' – '**Environment Options**'. Misalnya, kita akan menggunakan tema dari XP, maka klik tab '**Interface**' dan centang '**Use XP Theme**' seperti yang terlihat pada gambar sebelah kanan.



*Pengaturan Editor* juga dapat dilakukan dengan memilih menu '**Tools**' – '**Editor Options**'. Misalnya kita akan menampilkan nomor baris pada editor, maka klik tab '**Display**' dan centang '**Line Numbers**' seperti yang terlihat pada gambar sebelah kanan berikut.

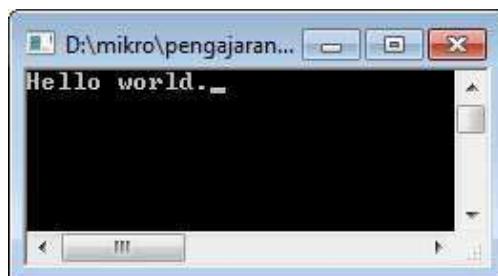


Setelah selesai melakukan pengaturan pada editor, maka kita sudah dapat menggunakan editor untuk menuliskan dan mengeksekusi kode program.

### Catatan

- ð Bahasa C adalah **bahasa native** pada UNIX
- ð Bahasa C mendukung pemrograman **modular**
- ð Bahasa C: "**Compact, fast, and powerful.**"
- ð Bahasa C bersifat **case-sensitive**.
- ð Bahasa C memiliki **struktur baris yang bebas** (free-form line structure).
- ð **Setiap akhir dari statement harus ditandai dengan tanda semicolon (;**

### Contoh 1

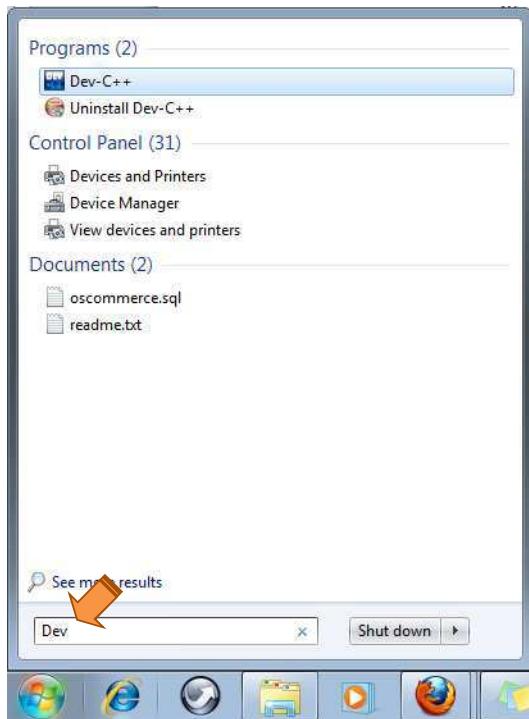


Langkah yang dilakukan untuk membuka Editor DevC++:

1. Pilih tombol '**Start**' dan pilih DevC++.



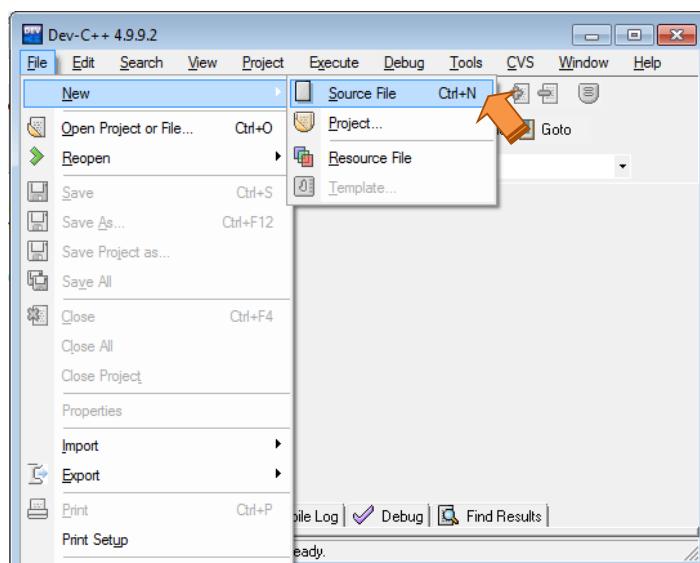
- Masukkan teks : “**Dev**” untuk mencari program pada komputer.



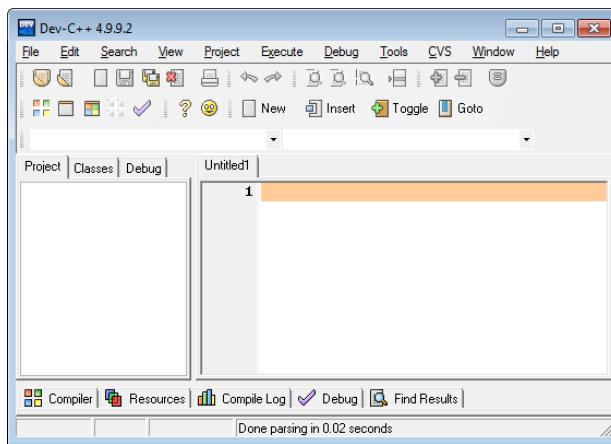
- Pilih ‘**Dev-C++**’ sehingga editor DevC++ muncul pada layar dan siap untuk digunakan.

Langkah selanjutnya adalah mengetikkan kode program pada editor dan mengeksekusinya:

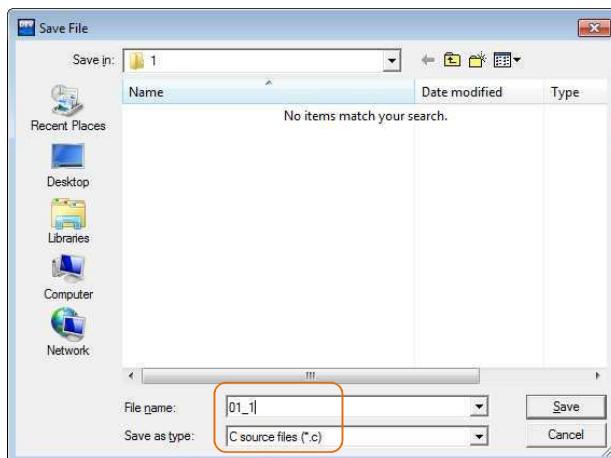
- Buatlah sebuah berkas baru dengan memilih menu ‘**File**’ – ‘**New**’ – ‘**Source File**’ atau melalui shortcut “**Ctrl + N**”.



2. Editor akan menampilkan sebuah **lembar kerja baru** yang kosong.



3. Simpan berkas terlebih dahulu dengan nama "**01\_1.c**" dengan memilih menu '**File**' - '**Save As**' atau melalui shortcut '**Ctrl + F12**'.



4. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1 // Nama File: 01_1.c
2 // Contoh Program Hello World.
```

Baris 1 & 2: tanda "://" berarti komentar pada satu baris program C.

```
3
4 #include<stdio.h>
```

Baris 4: menyisipkan header file yang berisi fungsi standar input dan output pada program. Header file "stdio.h" – standar input output.

✓ `#include<path-spec>`

```
5 int main() {
```

Baris 5: **titik awal eksekusi program** bahasa C dan menunggu nilai kembali berupa angka (int).

Tanda ‘{‘ mengawali isi dari fungsi main.

```
6     printf("Hello World!");
```

Baris 6: fungsi printf() merupakan perintah pada bahasa C untuk menampilkan variabel dan teks.

```
✓ int printf( const char* format, ... );
```

```
7     getch();
```

Baris 7: fungsi getch() merupakan perintah pada bahasa C untuk menunggu input sebuah karakter pada keyboard.

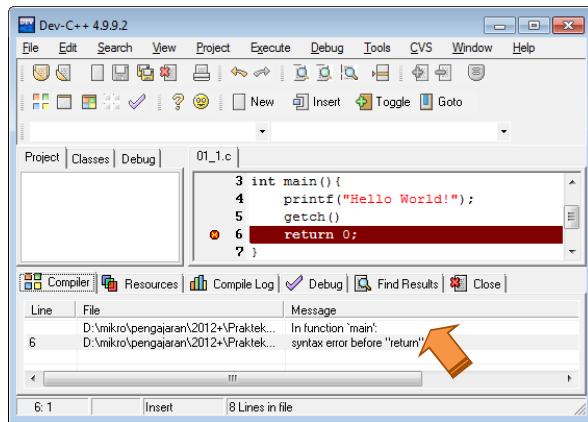
```
8     return 0;
```

Baris 8: mengembalikan angka 0 ke fungsi main pada baris 5.

```
9 }
```

Baris 9: tanda ‘}’ mengakhiri isi dari fungsi main.

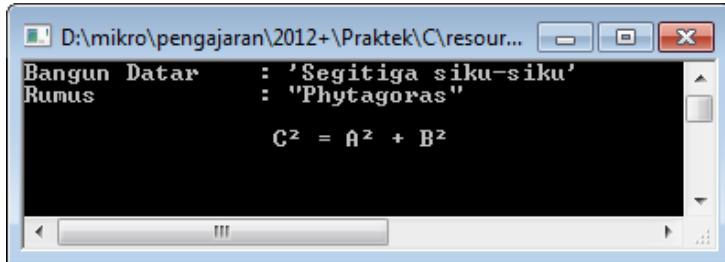
5. Setelah program selesai ditulis, kita dapat melakukan kompilasi kode program dengan memilih menu ‘Execute’ – ‘Compile’ atau shortcut “Ctrl +F9”. **Pastikan tidak ada kesalahan sintaks pada program.**
  - Jika terdapat kesalahan, maka panel hasil compiler akan muncul pada bagian bawah editor, misalnya seperti gambar berikut ini. Semua kesalahan **harus diperbaiki** terlebih dahulu sebelum dilanjutkan ke langkah berikutnya.



- Hasil eksekusi dari program yang telah ditulis dapat dilihat dengan memilih menu 'Execute' - 'Run' atau shortcut "Ctrl+F10".

⇒ **Cara cepat** untuk melakukan kompilasi dan menjalankan program pada DevC++ adalah memilih menu 'Execute' - 'Compile & Run' atau shortcut "F9".

## Contoh 2



Langkah pengerjaan kode program:

- Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '01\_2.c'.
- Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1 /* Nama File: 01_2.c
2 Contoh Program Hello World.
3 */

```

Baris 1-3: "/\* ... \*/" berarti komentar untuk beberapa baris pada program C.

```

4
5 #include<stdio.h>

```

Baris 5: menyisipkan header file "stdio.h" – standart input output.

```
6 int main() {
```

Baris 6: titik awal eksekusi program bahasa C.

```
7     printf("Bangun Datar \t: \'Segitiga siku-siku\'\n");
8     printf("Rumus \t\t: \"Phytagoras\" \n");
```

Baris 7 & 8: menampilkan teks (string) biasa dan karakter khusus.

- ∂ \t = horizontal tab
- ∂ \' = single quotation mark
- ∂ \" = double quotation mark
- ∂ \n = new line

✓ Daftar lengkap *Reserved Character* dilihat di lampiran

```
9     printf("\n\t\t C\x0fd = A\x0fd + B\x0fd \n");
```

Baris 9: menampilkan karakter ASCII pada *console*.

- ∂ ASCII code: \x0fd (HEX) = 253 (DEC) = ^

✓ Daftar Lengkap Kode ASCII dapat dilihat di lampiran.

```
10    getch();
11    return 0;
12 }
```

3. Compile dan Run kode program yang telah ditulis (**F9**).

## Latihan Modul 1

1. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '01\_a1.c'.

```
D:\mikro\pengajaran\2012+\Praktek\C\resources\1\01_a1.exe
=====
          Informasi Matakuliah
=====
Kode MataKuliah : MKB222           Jenjang Studi   : S1
Nama MataKuliah : Bahasa Pemrograman I   Program Studi   : TI
SKS MataKuliah  : 2                 Semester       : II
=====
```

2. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '01\_a2.c'.

```
D:\mikro\pengajaran\2012+\Praktek\C...
Program Hello World dengan Bahasa C
// Nama File: 01_1.c
// Contoh Program Hello World.
#include<stdio.h>

int main(){
    printf("Hello World! \n");
    getch();
    return 0;
}
```

3. Rancang program dengan tampilan berikut dan simpan dengan nama '01\_a3.c':

```
D:\mikro\pengajaran\2012+\Prakt...
Sintaks matematika:
 $\mu = \langle Ex \rangle / n$ 
 $f(x) = \frac{1}{2}(\sin^n \alpha + \cos \beta)$ 
 $\sqrt{4} = \pm 2$ 
```

4. Bentuklah inisial nama Anda (2 huruf balok besar) dengan menggunakan kode ASCII 176 (Ⓐ) dan simpan dengan nama '01\_a4.c'.

## Modul II

### Variabel, Konstanta, dan Operator Aritmatika

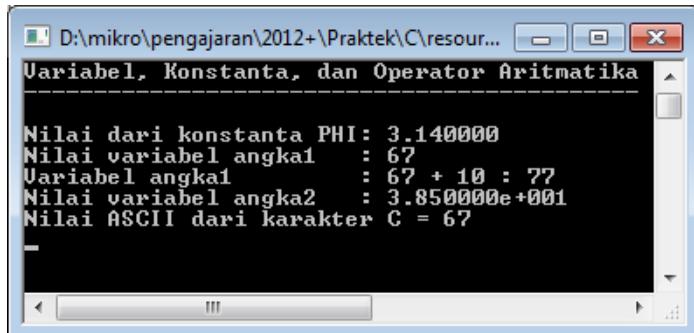
#### Objektif

Deklarasi dan inisialisasi variabel, serta memprosesnya melalui operator aritmatika.

#### Catatan

- ð Nama variabel harus dimulai dari **karakter atau garis bawah**. Contoh: angka1, \_angka, i, I, jumlah\_angka.
- ð Nama variabel **tidak boleh sama** dengan *keyword* pada bahasa C.
- ð Daftar keyword bahasa C dapat dilihat di lampiran.
- ð Nama variabel bersifat **case-sensitive**.
- ð Nama variabel sebaiknya memiliki **arti dan pendek**.

#### Contoh 1



```
Nilai dari konstanta PHI: 3.140000
Nilai variabel angka1 : 67
Variabel angka1 : 67 + 10 : 77
Nilai variabel angka2 : 3.850000e+001
Nilai ASCII dari karakter C = 67
```

Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '02\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*
2   * Nama File: 02_1.c
3   */
4
5  #include<stdio.h>
6  #define PHI 3.14
```

Baris 6: mendefinisikan sebuah konstanta bernama PHI dan berisi nilai 3.14

- ∂ Directive `#define` **tidak diakhiri** dengan tanda semicolon (`;`)
- ∂ Defenisi konstanta juga dapat dilakukan dengan menggunakan keyword `const`.

✓ `#define token value`  
 ✓ `const [type] var = value;`

```
7 int main() {
8     printf("Variabel, Konstanta, dan Operator Aritmatika\n ");
9     printf("-----\n\n");
10    printf("Nilai dari konstanta PHI: %f \n", PHI);
```

Baris 10: menampilkan konstanta yang bertipe data `float` (numerik dengan desimal) dengan format `%f`.

`printf ("Nilai dari konstanta PHI: %f \n", PHI);`

Teks tanpa format apapun, akan ditampilkan sama persis ke *console*

Nilai PHI ditampilkan dalam format floating point (%f)

✓ Daftar format tipe data bahasa C dapat dilihat pada Lampiran.

```
11 int angka1;
```

Baris 11: deklarasi variabel angka1 dengan tipe data int.

✓ Daftar tipe data bahasa C dapat dilihat pada Lampiran.

```
12 angka1 = 67;
```

Baris 12: inisialisasi variabel angka1 dengan nilai 67

- ∂ **Pastikan variabel telah dideklarasikan terlebih dahulu** sebelum diinisialisasi dan sesuai dengan tipe data nilai yang akan dimasukkan.

```
13     printf ("Nilai variabel angka1: %d \n", angka1);
```

Baris 13: mencetak variabel angka1 dengan format integer (int).

```
14
15     angka1 += 10;
```

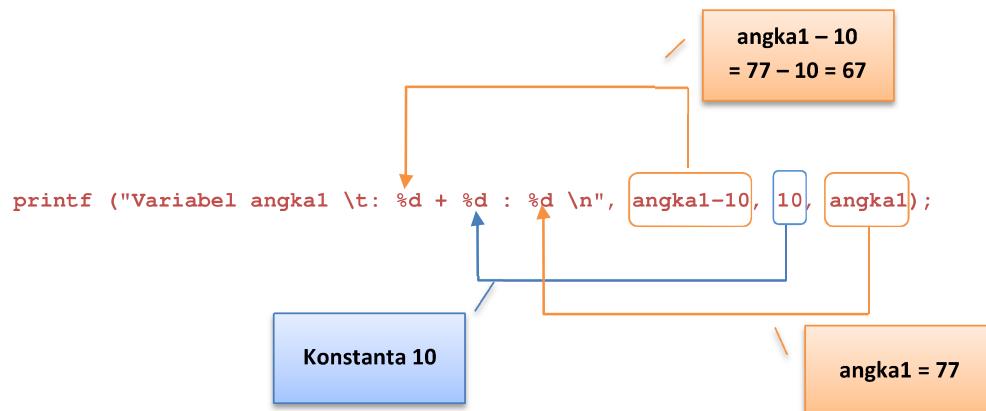
Baris 15: menambahkan variabel angka1 dengan nilai 10 dan menyimpannya kembali pada variabel angka1. Nilai variabel angka1 sekarang:  $67 + 10 = 77$ .

- ∂ “ += ” : tanda ‘+’ dan ‘=’ tidak boleh dipisah dengan spasi.

✓ Daftar Operator aritmatika dapat dilihat pada lampiran.

```
16     printf ("Variabel angka1 \t: %d + %d : %d \n", angka1-10,
10, angka1);
```

Baris 16: menampilkan nilai dari variabel (angka1 - 10), konstanta 10, dan nilai variabel angka1.



```
17     float angka2 = angka1 / 2.0;
```

Baris 15: mendeklarasikan sekaligus menginisialisasikan nilai dari variabel angka2 yang bertipe float. Nilai angka2 diperoleh dari perhitungan aritmatik variabel angka1 dibagi dengan bilangan floating 2.0.

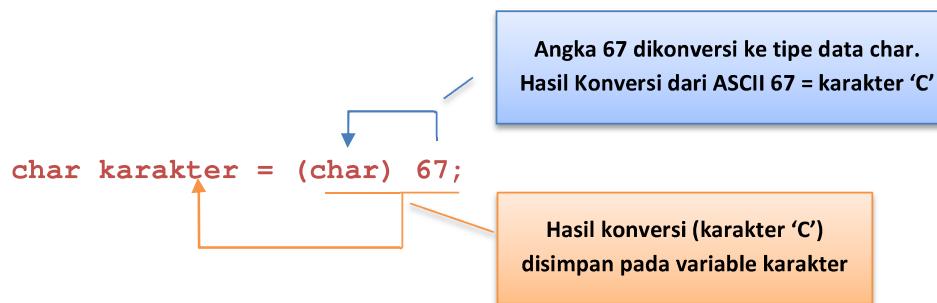
- ∂ Nilai integer dibagi nilai integer menghasilkan keluaran dengan tipe integer.
- ∂ Nilai integer dibagi nilai float menghasilkan keluaran dengan tipe float.

```
18
19     printf ("Nilai variabel angka2: %e \n", angka2);
```

Baris 19: menampilkan variabel angka2 yang bertipe data float dalam bentuk eksponensial (%e).

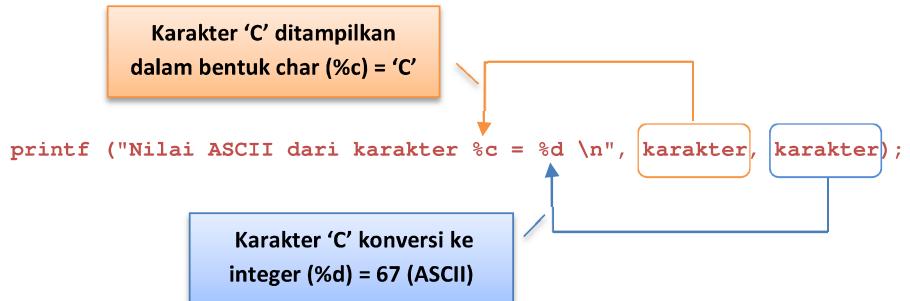
```
20
21     char karakter = (char) 67;
```

Baris 21: mendeklarasikan variabel karakter dengan tipe data char dan menginisialisasikannya dengan karakter dengan kode ASCII 67.



```
22     printf ("Nilai ASCII dari karakter %c = %d \n", karakter,
23                           karakter);
```

Baris 22: menampilkan variabel karakter dalam format yang berbeda, sehingga hasil *output console* disesuaikan dengan format *output*.

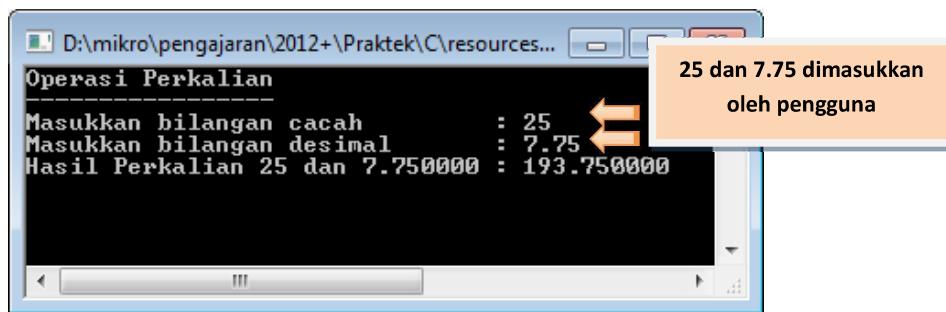


```

24     getch();
25     return 0;
26 }
```

3. Compile dan Run kode program yang telah ditulis (**F9**).

## Contoh 2



Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '02\_2.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1 /*
2  Nama File: 02_2.c
3 */
4 #include<stdio.h>
5
6 int main() {
7     int angka1;
8     double angka2, hasil;
```

Baris 8: deklarasikan variabel yang memiliki tipe data yang sama sekaligus.

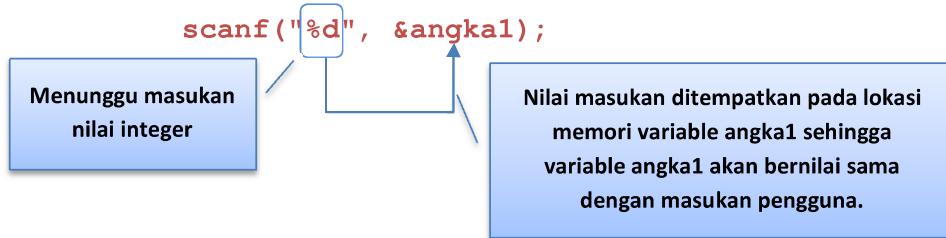
Variabel angka2 dan hasil1 akan memiliki tipe data double.

```

9     printf("Operasi Perkalian\n");
10    printf("-----\n");
11
12    printf("Masukkan bilangan cacah \t: ");
13    scanf("%d", &angka1);
```

Baris 14: fungsi `scanf()` merupakan perintah pada bahasa C untuk membaca masukan dari standard input (keyboard) dan menyimpannya pada lokasi alamat memori yang dimasukkan pada argumen.

```
✓ int scanf( const char * format, ... );
```



```
14     printf("Masukkan bilangan desimal\t: ");
15     scanf("%lf", &angka2);
```

Baris 16: menunggu masukan pengguna dari keyboard dan menerima bilangan desimal (%lf).

- θ Bilangan desimal ditandai dengan karakter **titik** (.) contohnya: 2.0, 0.898, dan 6.22345.

```
16     hasil = angka1 * angka2;
```

Baris 17: melakukan operasi aritmatika perkalian antara variabel angka1 (int) dan angka2 (double), sehingga menghasilkan keluaran bertipe double.

```
17     printf ("Hasil Perkalian %d dan %lf : %lf", angka1, angka2,
18                           hasil);
```

Baris 13: mencetak variabel angka1 (%d), angka2 (%lf), dan hasil (%lf).

```
19     getch();
20     return 0;
21 }
```

## Latihan Modul 2

1. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '02\_a1.c'.

```
Menghitung Berat Badan Ideal (BMI)
_____
BMI = berat / (tinggi)^2
Masukkan berat badan <kg>      : 62.5
Masukkan tinggi badan <m>       : 1.75
BMI = 20.408163
```

2. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '02\_a2.c'.

```
Konversi Jarak
_____
1 km = 0.621370 mil
Masukkan jarak pertama <km>      : 2
Hasil konversi                      : 1.242740 mil

1 mil = 1.609347 km
Masukkan jarak kedua <mil>        : 2
Hasil konversi                      : 3.218694 km

Total jarak <jarak pertama + jarak kedua>:
5.218694 km atau 3.242740 mil
```

3. Rancang program dengan tampilan berikut dan simpan dengan nama '03\_a3.c':

```
Perhitungan Gaji
_____
Masukkan Gaji Pokok      : Rp. 10000000
Masukkan Tunjangan        : Rp. 3000000
Masukkan Bonus            : Rp. 2000000
Penghasilan Tidak Kena Pajak : Rp. 12000000
Pajak Penghasilan: 5 %
Gaji Kotor                : Rp. 15000000
Besar Pajak                : Rp. 690000
Gaji Bersih                : Rp. 14310000

Gaji Kotor = Gaji Pokok + Tunjangan + Bonus
Besar Pajak = (Gaji Kotor - PTKP) * PPh
Gaji Bersih = Gaji Kotor - Besar Pajak
```

## Modul III

### Struktur Keputusan

#### Objektif

Membuat program mampu mengeksekusi pernyataan yang berbeda tergantung pada kondisi.

#### Catatan

- ⇒ Bahasa C memiliki tiga jenis perintah yang mendukung struktur keputusan:

- **if**

```
if(control-expression)
    program-statement;
```

- **if - else**

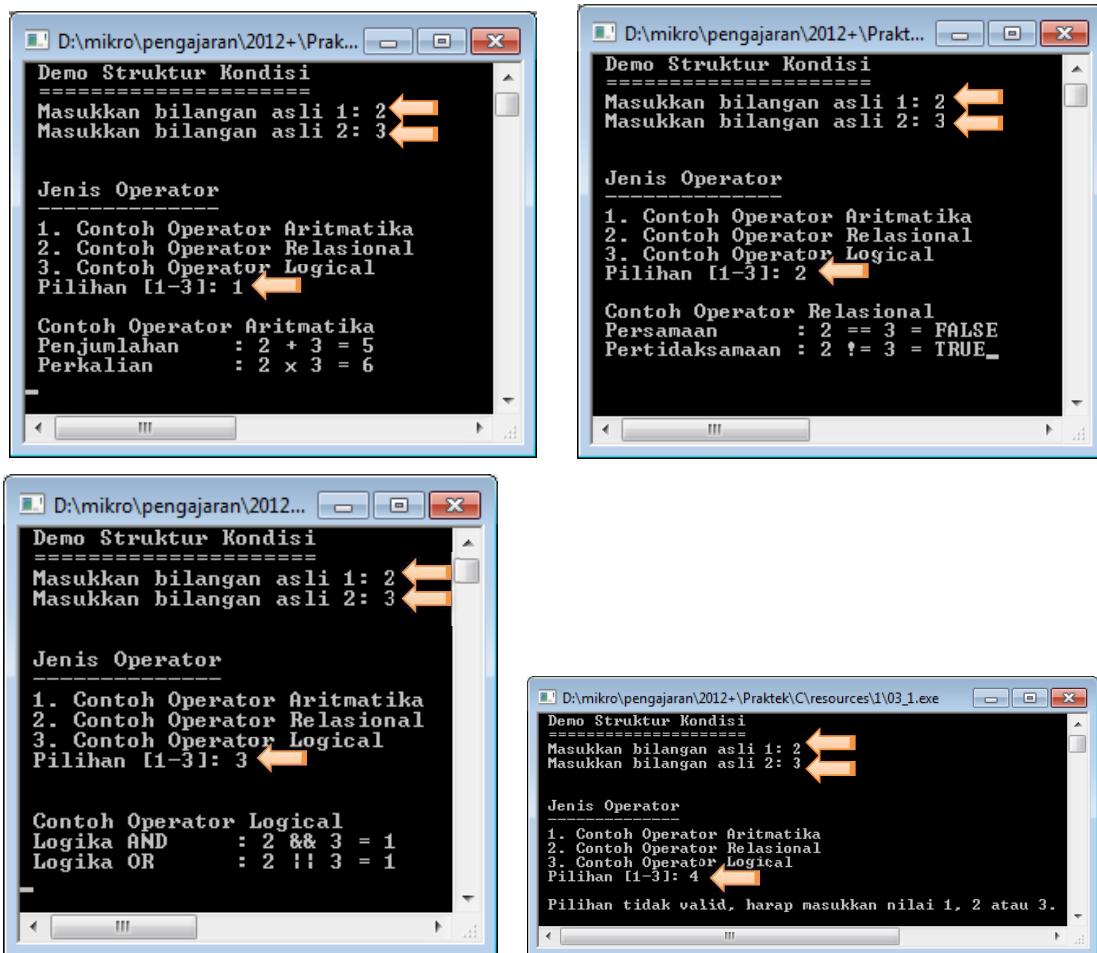
```
if(control expression)
    statement-1;
else
    statement-2;
```

- **switch**

```
switch (integer-expression)
{
    case constant1:
        statement-1;
        break;
    case constant2:
        statement-2;
        break;
    ...
    default:
        statement-1;
}
```

- ⇒ Gunakan '{' dan '}' untuk mengeksekusi lebih dari satu perintah pada satu kondisi.  
⇒ Bahasa C memungkinkan kondisi bersarang.

## Contoh



Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '03\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1  /*
2   * Nama File: 03_1.c
3  */
4 #include<stdio.h>
5
6 int main() {
7     int angka1, angka2;
```

Baris 7: deklarasikan variabel angka1 dan angka2 yang memiliki tipe data yang sama sekaligus.

```

8     printf(" Demo Struktur Kondisi\n");
9     printf(" ======\n");
10
11    printf (" Masukkan bilangan asli 1: ");
12    scanf("%d", &angka1);
13
14    printf (" Masukkan bilangan asli 2: ");
15    scanf("%d", &angka2);

```

Baris 12 & 15: perintah untuk meminta masukan dari pengguna dan disimpan pada variabel angka1 dan angka2 yang telah dideklarasikan terlebih dahulu dengan tipe data integer.

```

16    int pilihan;

```

Baris 16: deklarasi sebuah variabel pilihan yang akan menampung nilai dari menu pilihan yang dipilih pengguna.

```

17    printf("\n\n Jenis Operator\n");
18    printf(" -----\n");
19    printf(" 1. Contoh Operator Aritmatika\n");
20    printf(" 2. Contoh Operator Relasional \n");
21    printf(" 3. Contoh Operator Logical \n");
22    printf(" Pilihan [1-3]: ");
23    scanf("%d", &pilihan);

```

Baris 17-22: menampilkan semua pilihan yang disediakan oleh program.

Baris 23: menunggu masukan pilihan dari pengguna dan menyimpannya pada variabel pilihan.

```

24    if (pilihan == 1)
25    {
26        printf("\n Contoh Operator Aritmatika\n");
27        printf(" Penjumlahan \t: %d + %d = %d \n", angka1,
28               angka2, angka1 + angka2);
29        printf(" Perkalian \t: %d x %d = %d \n", angka1, angka2,
30               angka1 * angka2);
31    }

```

Baris 24: mengecek kondisi nilai dari variabel pilihan. Jika pilihan adalah satu, maka baris 26 – 29 akan dieksekusi dan langsung keluar dari blok IF-ELSE. Jika tidak, maka baris 26 – 29 tidak akan dieksekusi.

```

31     else if (pilihan == 2)
32     {
33         printf("\n Contoh Operator Relasional\n");
34         printf(" Persamaan \t: %d == %d = ", angka1, angka2);
35         if (angka1 == angka2)
36             printf("TRUE");
37         else
38             printf("FALSE");
39
40         printf("\n Pertidaksamaan\t: %d != %d = ", angka1,
41                angka2);
42         if (angka1 != angka2)
43             printf("TRUE");
44         else
45             printf("FALSE");
45 }

```

Baris 31: mengecek nilai variabel pilihan, jika pilihan bernilai dua maka baris 33 – 44 akan dieksekusi dan langsung keluar dari blok IF-ELSE. Jika tidak, maka baris 33 – 44 tidak akan dieksekusi.

```

46     else if (pilihan == 3)
47     {
48         printf("\n\n Contoh Operator Logical\n");
49         printf(" Logika AND \t: %d && %d = %d\n", angka1,
50                angka2, angka1 && angka2);
50         printf(" Logika OR \t: %d || %d = %d\n", angka1,
51                angka2, angka1 || angka2);
51     }

```

Baris 46: mengecek nilai variabel pilihan, jika pilihan bernilai tiga maka baris 48 – 50 akan dieksekusi dan langsung keluar dari blok IF-ELSE. Jika tidak, maka baris 48 – 50 tidak akan dieksekusi.

```

52     else
53         printf("\n Pilihan tidak valid, harap masukkan nilai
54             1, 2 atau 3.");
54

```

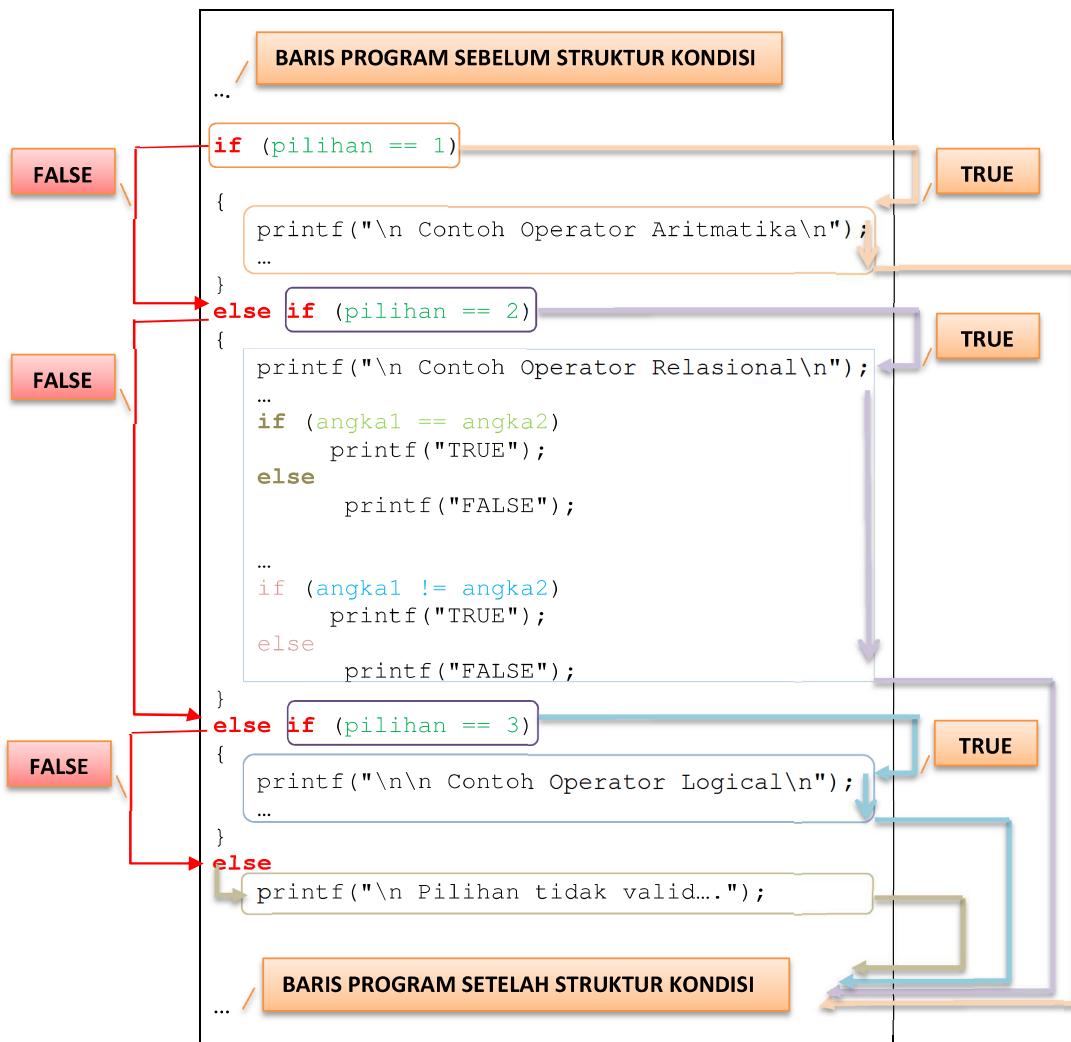
Baris 52: tidak melakukan pengecekan nilai variabel pilihan lagi. Baris 52 akan dieksekusi jika semua pilihan tidak sesuai dengan kondisi if sebelumnya, sehingga baris 53 akan dieksekusi.

```

55     getch();
56     return 0;
57 }

```

Ilustrasi dari struktur percabangan dapat dilihat pada gambar berikut.



### Latihan Modul 3

1. Perbaharui program pada '02\_a1.c' dan simpan dengan nama '03\_a1.c'.

The screenshot shows a Windows-style application window. The title bar reads "D:\mikro\pengajaran\2012+\Praktek\C\r..." and the main window title is "Menghitung Berat Badan Ideal <BMI>". Inside the window, the following text is displayed:

```
BMI = berat / (tinggi)^2
Masukkan berat badan <kg>      : 62.5
Masukkan tinggi badan <m>       : 1.75
BMI = 20.408163
```

Below this, the text "Kategori: Normal" is shown. To the right of the window is a separate table:

BMI	Kategori
< 15	Berat badan sangat kurang
15 - 18.49	Berat badan kurang (underweight)
18.5 - 23.99	Normal
24 - 29	Kelebihan berat badan (overweight)
> 29	Obesitas

2. Ubahlah struktur keputusan pada program **contoh modul III**, menjadi switch-case dan simpan dengan nama '03\_a2.c'.
3. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '03\_a3.c'.

The screenshot shows two side-by-side windows of a C program. Both windows have the title "Menampilkan Nama Bulan dalam Setahun".

The left window shows the following interaction:

```
Masukkan angka bulan [1-12]: 3
Nama Bulan: Maret
```

The right window shows the following interaction:

```
Masukkan angka bulan [1-12]: 0
Angka Bulan tidak valid.-
```

4. Buatlah program untuk mengecek tahun yang dimasukkan pengguna apakah tahun kabisat atau bukan dan simpan dengan nama '03\_a4.c'

## Modul IV

### Struktur Perulangan

#### Objektif

Membuat program mampu melakukan perulangan tergantung pada kondisi.

#### Catatan

- đ Bahasa C memiliki tiga jenis perintah yang mendukung struktur keputusan:

- **for**

```
for(<initialization>; <continuation>; <action>)  
    <statement>;
```

- **while**

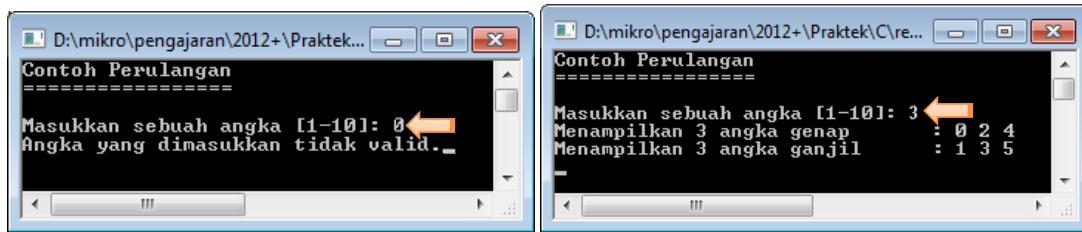
```
while(<expression>)  
    <statement>;
```

- **do - while**

```
do  
{  
    <statement>;  
} while (<expression>);
```

- đ Gunakan ‘{’ dan ‘}’ untuk mengeksekusi **lebih dari satu** perintah pada satu perulangan for dan while.
- đ Bahasa C memungkinkan perulangan **bersarang**.

## Contoh 1



Langkah penggerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '04\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1  /*
2   * Nama File: 04_1.c
3  */
4  #include<stdio.h>
5
6  int main() {
7      int i, angka;
```

Baris 7: deklarasikan variabel i dan angka yang memiliki tipe data yang sama sekaligus sebelum digunakan pada program.

```

8      printf("Contoh Perulangan\n");
9      printf("=====\\n\\n");
10     printf("Masukkan sebuah angka [1-10]: ");
11     scanf("%d", &angka);
12
```

Baris 11: memasukkan nilai untuk variabel angka yang akan digunakan dalam perulangan.

```

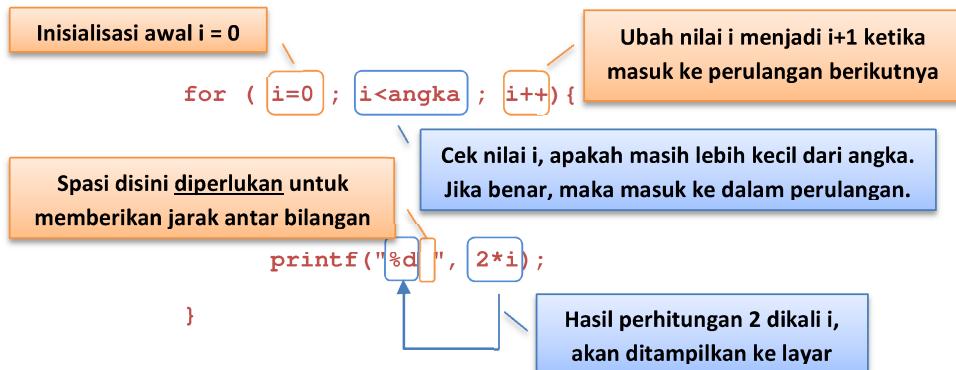
13     if(angka < 1 || angka > 10)
14         printf("Angka yang dimasukkan tidak valid.");
```

Baris 13&14: validasi masukan angka, supaya nilai angka yang diperbolehkan untuk diproses adalah 1-10.

```

15     else
16     {
17         printf("Menampilkan %d angka genap \\t: ", angka);
18         for (i=0; i<angka; i++){
19             printf("%d ", 2*i);
20         }
21         printf("\\n");
```

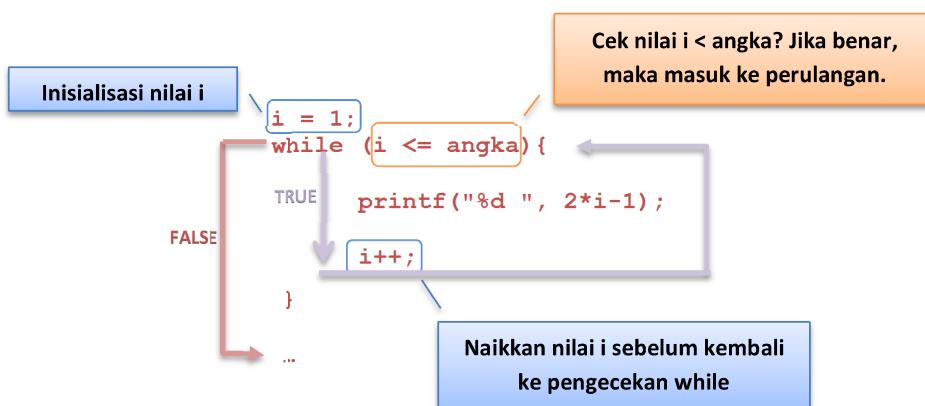
Baris 18: perulangan for menggunakan variabel i, variabel ini harus sudah dideklarasikan terlebih dahulu. Baris 18-20: menampilkan bilangan genap sebanyak nilai variabel angka.



```

22     printf("Menampilkan %d angka ganjil \t: ", angka);
23
24
25     i = 1;
26     while (i <= angka){
27         printf("%d ", 2*i-1);
28         i++;
29     }
30     printf("\n");
31
  
```

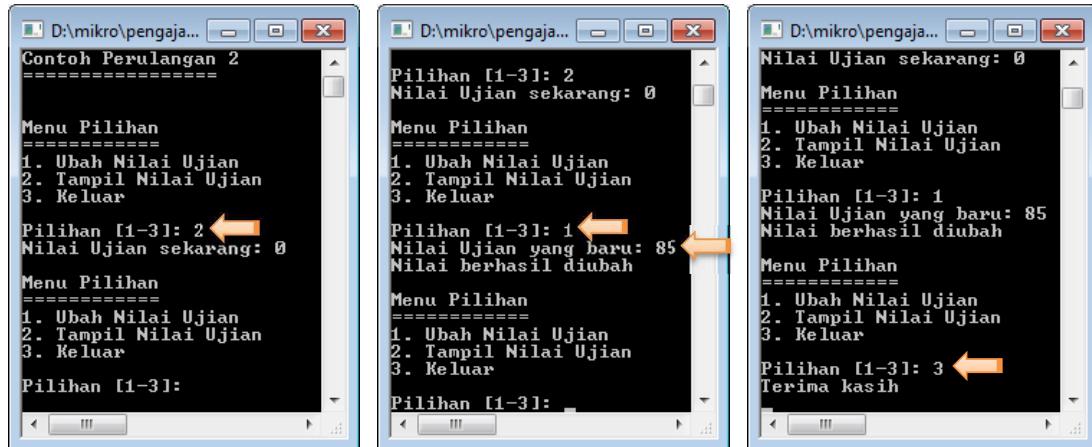
Baris 52: masih menggunakan variabel 'i' yang sama (jika kita mengubah nama variabel, misal variabel 'j', maka akan berefek pada isi perulangan pada while, kita harus mengganti seluruh nama variabel 'i' menjadi variabel 'j' pada **while**)



```

32     getch();
33     return 0;
34 }
```

## Contoh 2



Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '04\_2.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1  /*
2   Nama File: 04_2.c
3  */
4 #include<stdio.h>
5
6 int main() {
7     int pilihan, nilai=0;
8     printf("Contoh Perulangan 2\n");
9     printf("=====\\n\\n");
```

Baris 7-9: deklarasi variabel pilihan dan nilai, serta inisialisasi awal nilai dengan angka 0.

```
10    do{
```

Baris 10: memulai baris perulangan dari do-while.

```

11     printf("\nMenu Pilihan\n");
12     printf("=====\\n");
13     printf("1. Ubah Nilai Ujian\\n");
14     printf("2. Tampil Nilai Ujian\\n");
```

```

15     printf("3. Keluar\n\n");
16
17     printf("Pilihan [1-3]: ");
18     scanf("%d",&pilihan);

```

Baris 18: menunggu masukan pilihan dari pengguna.

```

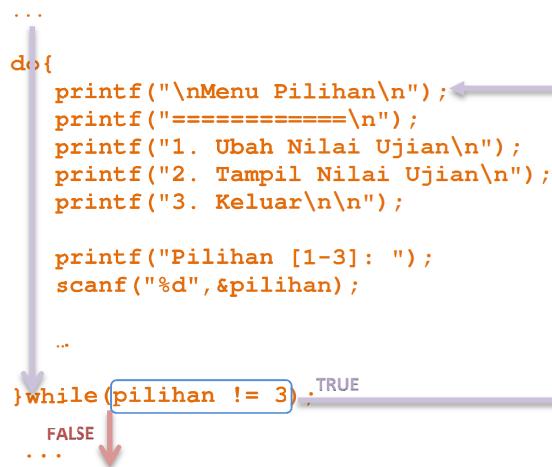
19     switch(pilihan){
20         case 1:
21             printf("Nilai Ujian yang baru: ");
22             scanf("%d",&nilai);
23             printf("Nilai berhasil diubah\n");
24             break;
25         case 2:
26             printf("Nilai Ujian sekarang: %d\n",nilai);
27             break;
28         case 3:
29             printf("Terima kasih\n");
30             break;
31     default:
32         printf("Pilihan salah\n");
33     }

```

Baris 19-33: memproses pilihan yang telah dimasukkan.

```
34 }while(pilihan != 3);
```

Baris 34: mengecek pilihan, jika bernilai TRUE, maka eksekusi akan dilakukan dari baris 10 kembali. Jika FALSE, maka perulangan tidak dieksekusi lagi.



```

35     getch();
36     return 0;
37 }

```

## Latihan Modul 4

1. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '04\_a1.c'.

```

D:\mikro\pengajaran\201...
Tampilkan deret 3, 6, 9, ...
Masukkan nilai N: 4
Deret : 3, 6, 9, 12

```

2. Tulislah program dengan perulangan while untuk menghasilkan keluaran berikut dan simpan dengan nama '04\_a3.c'.

```

D:\mikro\pengajaran\2012+\Prak...
Menampilkan huruf
N : 5
5 huruf kecil pertama: a b c d e
5 huruf besar pertama: A B C D E

```

3. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '04\_a2.c'.

```

D:\mikro\pengajaran\2012+\Prak...
Menghitung total dan rata-rata
Masukkan jumlah data: 5
Data ke 1 : 15
Data ke 2 : 16
Data ke 3 : 26
Data ke 4 : 4.5
Data ke 5 : 6.5
Total data : 68.000000
Rata-rata : 13.600000

```

4. Buatlah program untuk menampilkan pola dan simpan dengan nama '04\_a4.c'

```

D:\mikro\pengajaran\...
Menampilkan pola
Masukkan satu karakter : x
N : 5
Hasil Pola
x
x
x
x
x

```

## Modul V

### Array

#### Objektif

Menggunakan Array untuk menyimpan dan memproses data kolektif yang homogen.

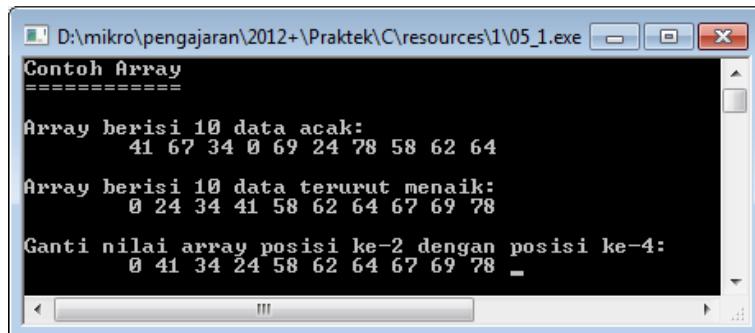
#### Catatan

- Bahasa C memiliki bentuk array:

```
tipeData namaVariabel[jumlahData];
```

- Karakter '[' dan ']' menjadi penanda pada variabel array.
- Indeks awal pada array selalu dimulai dari **nol** (0).

#### Contoh 1



Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '05\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1 /*  
2  * Nama File: 05_1.c  
3  * Array  
4 */  
5  
6 #include<stdio.h>  
7 #include<stdlib.h>  
8 #define MAX 10
```

Baris 7: menambahkan library stdlib.h supaya program dapat menggunakan fungsi random (rnd).

Baris 8: mendefinisikan konstanta MAX dengan nilai 10.

```
9  int main(){
10    int i, j, tmp;
11    int data[MAX];
```

Baris 11: mendeklarasikan variabel data bertipe array dari integer yang mampu menampung angka sebanyak MAX (10) nilai

Nilai MAX = 10  
Indeks dari 0 – 9 -> 10 data

indeks	0	1	2	3	4	6	7	8	9
nilai	?	?	?	?	?	?	?	?	?

Pada saat deklarasi nilai dari array  
sangat bervariasi dan tidak diketahui

```
12    printf("Contoh Array\n");
13    printf("=====\\n\\n");
14
15    for (i = 0; i < MAX; i++){
16        data[i] = rand() % 100;
17    }
```

Baris 15-17: Looping array data dari indeks 0 sampai indeks MAX-1

Baris 16: rand() = fungsi random pada bahasa C yang terdapat pada library stdlib.h. Nilai random tersebut akan dimodulus 100 supaya nilai pada array data indeks ke-i adalah 0 sampai 99.

```
18    printf("Array berisi %d data acak: \\n\\t", MAX);
19    for (i = 0; i < MAX; i++){
20        printf ("%d ",data[i]);
21    }
```

Baris 19-21: looping menampilkan seluruh nilai dari array data.

Baris 20: data[i] berarti mengakses nilai dari array data pada indeks ke i.

```

22     // bubble-sort
23     for(i = 0; i<MAX-1; i++) {
24         for (j=i; j<MAX; j++) {
25             if (data[i] > data[j]) {
26                 tmp = data[i];
27                 data[i] = data[j];
28                 data[j] = tmp;
29             }
30         }
31     }

```

Baris 22-31: implementasi dari algoritma sorting bubble sort.

```

32     printf("\n\nArray berisi %d data terurut menaik: \n\t",
33           MAX);
34     for (i = 0; i< MAX; i++) {
35         printf ("%d ",data[i]);
36     }
37     printf("\n\nGanti nilai array posisi ke-2 dengan posisi ke-
38           4:\n\t");
39     tmp = data[1];

```

Baris 38: menyimpan nilai dari array data ke-1 pada variabel tmp.

```

39     data[1] = data[3];

```

Baris 39: menimpa nilai dari array data ke-1 dengan nilai array data ke-3

```

40     data[3] = tmp;

```

Baris 40: menimpa nilai dari array data ke-3 dengan nilai pada variabel tmp yang sebelumnya merupakan nilai dari array data ke-1

```

41     for (i = 0; i< MAX; i++) {
42         printf ("%d ",data[i]);
43     }

```

Baris 41-42: looping menampilkan seluruh nilai dari array data.

```

44     getch();
45     return 0;
46 }

```

## Contoh 2

```
D:\mikro\pengajaran\20...
=====
Contoh Matriks Dua Dimensi
=====
Matrix:
 1      2
 3      4

Matrix2:
 5      6
 7      8

Matrix3:
 6      8
10     12
```

Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '05\_2.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1 /* 
2  Nama File: 05_2.c
3  Array 2 Dimensi
4 */
5
6 #include<stdio.h>
7 #define N 2
8
9 int main(){
10    int i,j;
11    int matrix1[N][N] = { {1,2},{3,4} };
12    int matrix2[N][N] = { {5,6},{7,8} };
```

baris 11 & 12: deklarasi dan inisialisasi array matrix1 dan matrix2

Jumlah baris dan kolom harus sesuai dengan yang dideklarasikan

Nilai yang diinisialisasi harus memiliki tipe data yang sesuai dengan deklarasi.

int matrix1[N][N] = { {1,2},{3,4} };

N = 2

Harus 2 buah nilai

```
13     int matrix3[N][N];
```

baris 13: deklarasikan array matrix3 bertipe integer dengan ukuran dimensi 2x2.

```
14     printf("Contoh Matriks Dua Dimensi\n");
15     printf("=====\\n\\n");
16
17     printf("Matrix:\\n");
18     for (i=0; i<N; i++){
19         for (j=0; j<N; j++)
20             printf("\t%d", matrix1[i][j]);
21         printf("\\n");
22     }
23     printf("\\n");
```

baris 17-23: dua buah looping untuk menampilkan isi dari array matrix1 yang berdimensi 2.

```
24     printf("Matrix2:\\n");
25     for (i=0; i<N; i++){
26         for (j=0; j<N; j++)
27             printf("\t%d", matrix2[i][j]);
28         printf("\\n");
29     }
30     printf("\\n");
31
32     for (i=0; i<N; i++)
33         for (j=0; j<N; j++)
34             matrix3[i][j] = matrix1[i][j]+matrix2[i][j];
35     printf("\\n");
```

baris 34: proses menghitung nilai dari array matrix3 dengan menambahkan nilai matrix1 dan matrix2

```
36     printf("Matrix3:\\n");
37     for (i=0; i<N; i++){
38         for (j=0; j<N; j++)
39             printf("\t%d", matrix3[i][j]);
40         printf("\\n");
41     }
```

Baris 36-41: dua buah looping untuk menampilkan isi dari array matrix3 yang berdimensi 2.

```
42     getch();
43     return 0;
44 }
```

## Latihan Modul 5

1. Tulislah program yang memanfaatkan array 1 dimensi yang berisi jumlah hari pada bulan untuk menghasilkan keluaran berikut dan simpan dengan nama '05\_a1.c'.

```
Jumlah hari pada bulan tertentu pada tahun 2013
=====
Masukkan nilai bulan [1-12]: 4
Jumlah hari pada bulan ke-4 : 30
```

2. Tulislah program dengan perulangan while untuk menghasilkan keluaran berikut dan simpan dengan nama '05\_a2.c'.

```
Masukkan nilai n = 4
Nilai ke-1 :15
Nilai ke-2 :9
Nilai ke-3 :16
Nilai ke-4 :-2
Data :
      15 9 16 -2

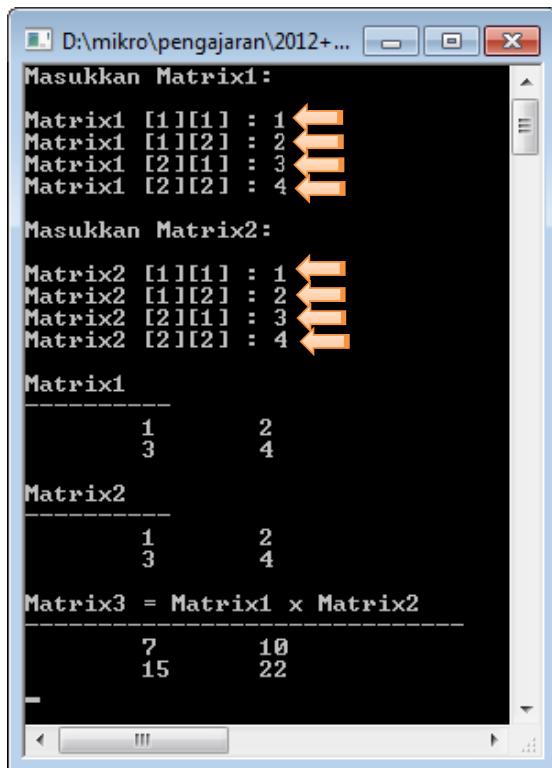
Total data      : 38
Rata-rata      : 9.500000
```

3. Tulislah program untuk menghasilkan 15 data acak yang disimpan pada array untuk menghasilkan keluaran berikut dan simpan dengan nama '05\_a3.c'

```
Cek Genap, Ganjil, dan Prima pada Array
=====
Array berisi 15 data acak:
        41 67 34 0 69 24 78 58 62 64 5 45 81 27 61

Jumlah Genap    : 7
Jumlah Ganjil   : 8
Jumlah Prima    : 4
```

4. Tulislah program untuk menginput nilai dari matrix1 dan matrix2 dengan dimensi 2x2, kemudian hitunglah matrix3 yang merupakan hasil perkalian dari matrix1 dan matrix2 seperti keluaran berikut dan simpan dengan nama '05\_a4.c'.



D:\mikro\pengajaran\2012+...

```
Masukkan Matrix1:  
Matrix1 [1][1] : 1  
Matrix1 [1][2] : 2  
Matrix1 [2][1] : 3  
Matrix1 [2][2] : 4  
  
Masukkan Matrix2:  
Matrix2 [1][1] : 1  
Matrix2 [1][2] : 2  
Matrix2 [2][1] : 3  
Matrix2 [2][2] : 4  
  
Matrix1  
-----  
      1      2  
      3      4  
  
Matrix2  
-----  
      1      2  
      3      4  
  
Matrix3 = Matrix1 x Matrix2  
-----  
      7      10  
     15      22
```

## Modul VI

### String

#### Objektif

Membuat program yang memiliki string dan memprosesnya.

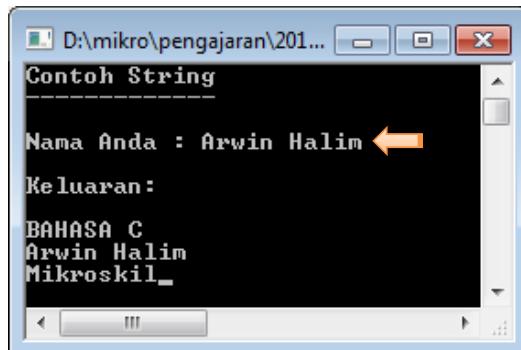
#### Catatan

- Bahasa C memiliki bentuk string:

```
char namaVariabel[jumlahData];
char* string;
```

- String pada array dari karakter harus diakhiri dengan karakter berkode ASCII 0 ('\0').
- Library pada bahasa C yang mendukung operasi string: string.h

#### Contoh 1



Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '05\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*
2   Nama File: 05_1.c
3   String
4 */
5
6 #include<stdio.h>
```

```

7  int main(){
8      int i;
9      char nama[100];

```

Baris 9: deklarasi string nama dengan panjang maksimum 100 karakter

```

10     char kuliah[100] = {'B', 'A', 'H', 'A', 'S', 'A', ' ', 'C', '\0'};

```

Baris 10: deklarasi string kuliah dan inisialisasi dengan cara memasukkan satu per satu karakter pada array sampai diakhiri dengan tanda null ('\0')

Variabel kuliah:

indeks	0	1	2	3	4	6	7	8	9	10	...	99
nilai	'B'	'A'	'H'	'A'	'S'	'A'	' '	'C'	'\0'	?	?	?

Karakter setelah null ('\0')  
tidak akan diproses pada string

Setiap indeks pada array  
hanya berisi satu karakter

Karakter null ('\0') merupakan  
penanda akhir dari string

```

11     char kampus[100] = "Mikroskil";

```

Baris 11 : deklarasi string kampus dan inisialisasi dengan menggunakan tanda petak ganda (""). Jika menggunakan tanda petik ganda, maka penanda akhir string (null) akan otomatis ditambahkan pada akhir string.

```

12     printf("Contoh String\n");
13     printf("-----\n");
14     printf("\nNama Anda : ");
15     gets(nama);

```

Baris 15: menunggu masukan string untuk variabel nama dari pengguna, menggunakan fungsi gets (library stdio.h)

```

16     puts("\nKeluaran:\n");
17     puts(kuliah);

```

Baris 17: menampilkan string dengan fungsi puts

```

18     i=0;

```

```

19     while (nama[i] != '\0'){
20         putchar(nama[i]);
21         i++;
22     }
23     putchar('\n');

```

Baris 18-23: menampilkan string dengan cara iterasi satu per satu isi dari array nama sampai ditemukan penanda akhir string (null).

```

24
25     printf("%s", kampus);

```

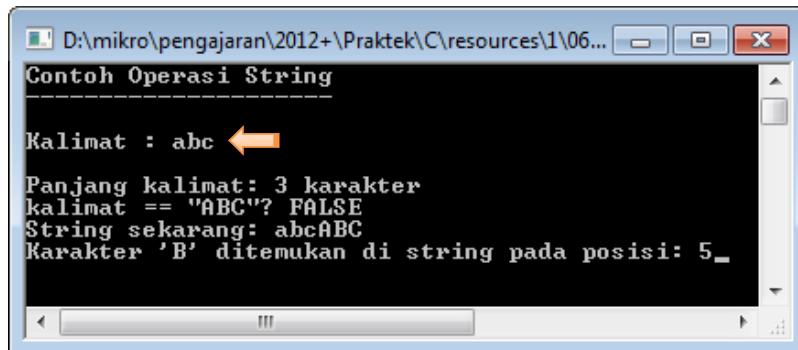
Baris 25: menampilkan string dengan menggunakan fungsi printf.

```

26     getch();
27     return 0;
28 }

```

## Contoh 2



Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '05\_2.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1  /*
2   * Nama File: 06_1.c
3   * Operasi String
4   */
5
6 #include<stdio.h>
7 #include<string.h>
8
9 int main(){
10     int i;

```

```
11     char string[100];
```

Baris 11: deklarasi variabel string dengan array karakter yang memiliki panjang maksimum 100.

```
12     char* tmp = "ABC\0";
```

Baris 12: deklarasi variabel tmp bertipe string dengan pointer karakter dan melakukan inisialisasi variabel dengan nilai string ABC.

- ð Karakter '\0' (null) menunjukkan akhir dari string.

```
13     printf("Contoh Operasi String \n");
14     printf("-----\n");
15     printf("\nKalimat : ");
16     gets(string);
```

Baris 16: menunggu masukan pengguna berupa string.

- ð Fungsi gets merupakan fungsi dari library stdio.h

```
17
18     printf("\nPanjang kalimat: %d karakter\n", strlen(string));
```

Baris 18: fungsi strlen(<var>) mengembalikan panjang karakter dari string.

- ð Fungsi strlen merupakan fungsi dari library string.h

```
19     printf("kalimat == \"ABC\"? ");
20     if (strcmp(string,tmp) == 0)
21         printf("TRUE\n");
22     else
23         printf("FALSE \n");
```

Baris 20: fungsi strcmp(<var1>, <var2>) membandingkan dua variabel string.

Fungsi ini mengembalikan nilai 0 jika <var1> dan <var2> sama.

- ð Fungsi strcmp merupakan fungsi dari library string.h

```
24
25     strcat(string,tmp);
```

Baris 25: fungsi strcat(<var1>,<var2>) menggabungkan string <var1> dan <var2> serta hasilnya disimpan pada <var1>.

- ð Fungsi strcat merupakan fungsi dari library string.h

```
26     printf("String sekarang: ");
```

```
27 puts(string);
```

Baris 27: fungsi puts digunakan untuk menampilkan variabel string.

- ③ Fungsi puts merupakan fungsi dari library stdio.h

```
28
29     printf ("Karakter 'B' ditemukan di string pada posisi: %d",
30             strchr(string, 'B')-string+1);
31 }
```

Baris 29: fungsi strchr(<var1>,<char>) mengembalikan nilai pointer untuk karakter (<char>) yang pertama kali ditemukan pada string (<var1>).

- ③ Fungsi strchr merupakan fungsi dari library string.h

```
32     getch();
33     return 0;
34 }
```

## Latihan Modul 6

1. Tulislah program string yang memanfaatkan array untuk menghasilkan keluaran berikut dan simpan dengan nama '06\_a1.c'.

```
D:\mikro\pengajaran\2012+\Pr...
Masukkan sebuah kalimat: nama
Keluaran: n_a_m_a
```

2. Tulislah program pengecekan palindrom untuk menghasilkan keluaran berikut dan simpan dengan nama '06\_a2.c'.

```
Pengecekan Huruf
=====
Masukkan sebuah kalimat: abCd EF GH
Jumlah huruf kecil : 3
Jumlah huruf besar : 5
```

3. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '06\_a3.c'

```
Pengecekan Palindrom
=====
Masukkan sebuah kalimat: ana
Palindrom: True
```

4. Tulislah program untuk menghasilkan keluaran berikut dan simpan dengan nama '06\_a4.c'

```
Operasi String
=====
Masukkan nama lengkap: hello, world
Jumlah kata : 2
Kata - 1 : hello, - 6 karakter
Kata - 2 : world - 5 karakter
```

## Modul VII

### Fungsi

#### Objektif

Membuat fungsi-fungsi pada program untuk memecah sistem yang besar.

#### Catatan

- Bahasa C memiliki bentuk definisi fungsi:

```
return-type function-name (argument declarations) {  
    program-statement;  
}
```

- Tanda ‘‘ dan ‘’ wajib ditulis walaupun fungsi hanya mengeksekusi satu baris program.

#### Contoh

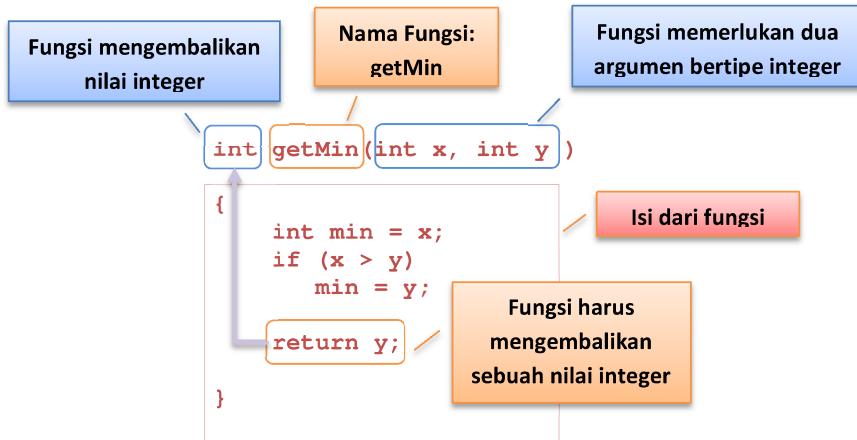


Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '07\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*  
2   Nama File: 07_1.c  
3 */  
4 #include<stdio.h>  
5 int getMin(int x, int y){  
6     int min = x;  
7     if (x > y)  
8         min = y;  
9     return min;  
10 }
```

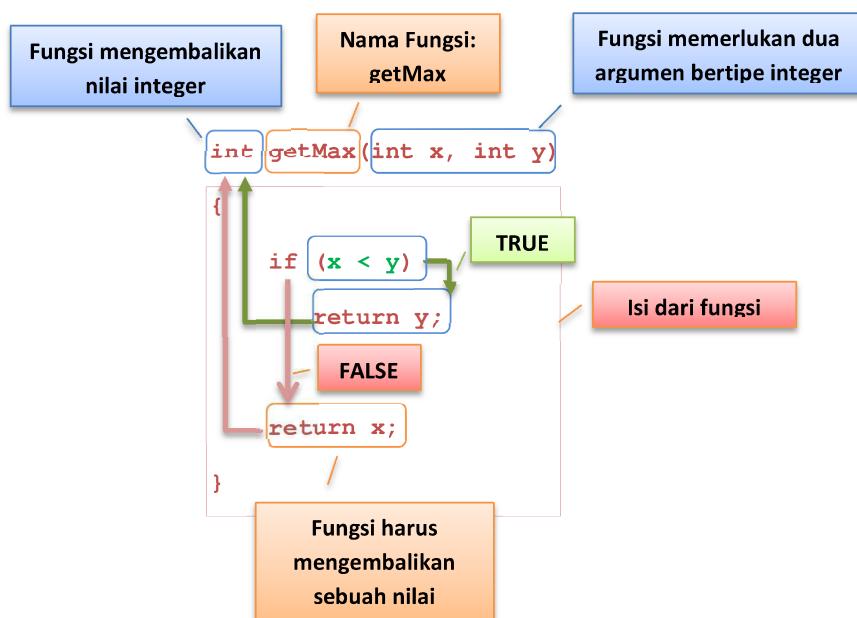
Baris 6-11: mendefenisikan sebuah fungsi getMin dengan dua argumen bertipe data integer.



```

11 int getMax(int x, int y){
12     if (x < y)
13         return y;
14     return x;
15 }
16
  
```

Baris 12-16: mendefenisikan sebuah fungsi getMax dengan dua argumen bertipe data integer.

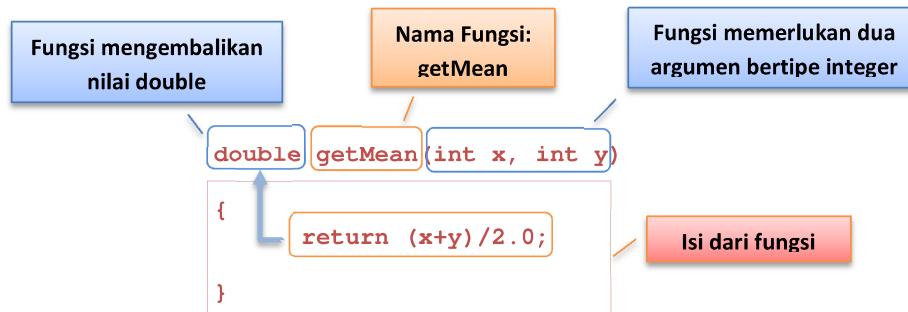


```

17 double getMean(int x, int y){
18     return (x+y)/2.0;
19 }
20

```

Baris 18-20: mendefinisikan sebuah fungsi getMean yang memerlukan dua argumen bertipe integer.

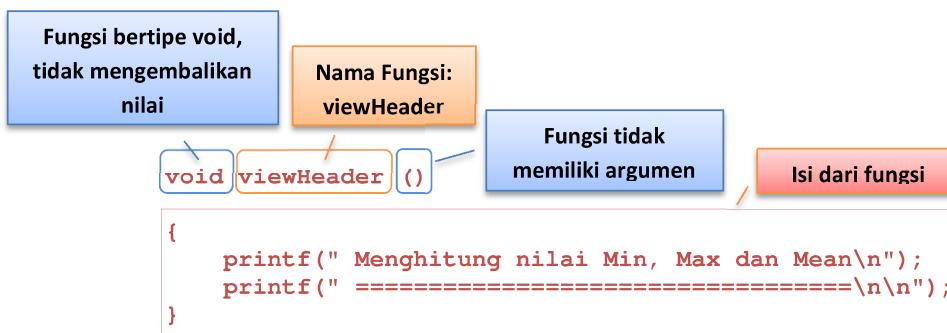


```

21 void viewHeader(){
22     printf(" Menghitung nilai Min, Max dan Mean\n");
23     printf(" ======\n\n");
24 }

```

Baris 22-25: mendefenisikan sebuah fungsi viewHeader tanpa parameter dan tidak memerlukan nilai balik.



```

25
26 int main(){
27     int x,y;
28     viewHeader();

```

Baris 29: memanggil fungsi viewHeader yang telah didefinisikan sebelumnya.

```

29     printf(" Masukkan nilai 1: ");
30     scanf("%d",&x);
31     printf(" Masukkan nilai 2: ");

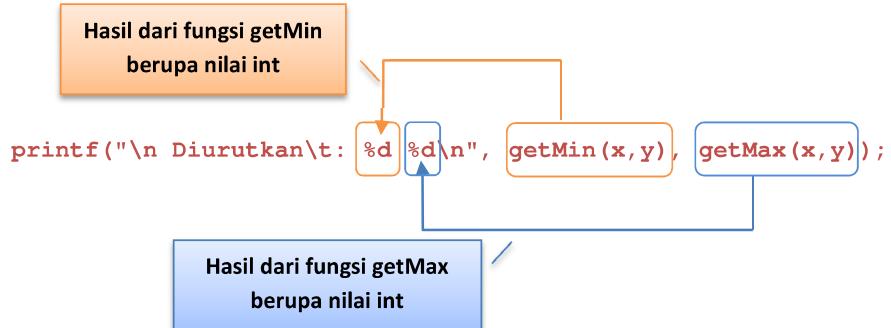
```

```
32     scanf ("%d", &y);
```

Baris 31&33: menunggu masukkan pengguna dan disimpan pada x dan y.

```
33
34     printf ("\n Diurutkan\t: %d %d\n", getMin (x,y), getMax (x,y));
```

Baris 35: memanggil fungsi getMin dan getMax yang masing-masing mengembalikan sebuah nilai integer.



```
35     printf ("\n Rata-rata\t: %lf\n", getMean (x,y) );
36 }
```

Baris 36: menampilkan hasil dari perhitungan fungsi getMean berupa double.

```
37     getch();
38     return 0;
39 }
```

## Latihan Modul 7

1. Ubahlah fungsi pada program **contoh modul VII**, supaya menerima tiga masukan:

- int getMin(int x, int y, int z)
- int getMax(int x, int y, int z)
- double getMean(int x, int y, int z)

Hasilkan keluaran seperti berikut dan simpan dengan nama '07\_a1.c'.

```
Menghitung nilai Min, Max dan Mean
=====
Masukkan nilai 1: 7
Masukkan nilai 2: 5
Masukkan nilai 3: 10
Nilai min      : 5
Nilai max      : 10
Nilai mean     : 7.333333
```

2. Buatlah sebuah fungsi fibonaci yang digunakan untuk menampilkan keluaran seperti berikut dan simpan dengan nama '07\_a2.c'

```
Deret Fibonaci dengan fungsi
=====
Masukkan panjang deret: 9
Fibonaci: 1 1 2 3 5 8 13 21 34
```

3. Buatlah sebuah fungsi isNumeric untuk mengecek argumen karakter merupakan bilangan 1 digit atau bukan "int isNumeric(char karakter)". Hasilkan keluaran seperti berikut dan simpan dengan nama '07\_a3.c'

```
Cek isNumeric
=====
Masukkan 1 karakter: a
isNumeric: FALSE
```

```
Cek isNumeric
=====
Masukkan 1 karakter: 9
isNumeric: TRUE
```

4. Buatlah sebuah fungsi isPrime untuk mengecek apakah argumen merupakan bilangan prima atau tidak “int isPrime(int x)”. Hasilkan keluaran berikut dan simpan dengan nama ‘07\_a4.c’



The image shows two side-by-side screenshots of a terminal window titled "Cek Prima". Both windows have a title bar "D:\mikro\pengajaran..." and a status bar at the bottom.

**Screenshot 1:**  
The window displays:  
Cek Prima  
=====  
Masukkan sebuah angka : 13  
isPrime: TRUE

**Screenshot 2:**  
The window displays:  
Cek Prima  
=====  
Masukkan sebuah angka : 12  
isPrime: FALSE

Both screenshots have an orange arrow pointing to the input value in each respective window.

## Modul VIII

### Pointer

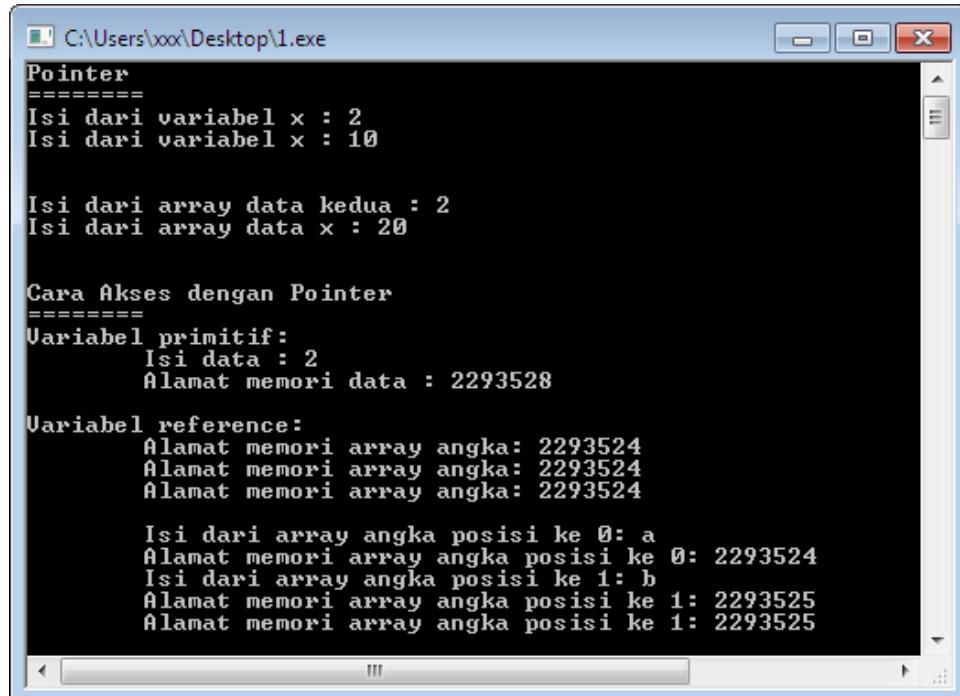
#### Objektif

Memahami cara kerja pointer dan penggunaannya dalam memanipulasi array.

#### Catatan

- Bahasa C menyediakan pointer untuk mengakses data pada alamat tertentu pada memori komputer.
- Tanda '\*' pada variabel menunjukkan penggunaan pointer.
- Tanda '&' pada variabel mengembalikan alamat memori yang ditempati variabel

#### Contoh



```
C:\Users\xxx\Desktop\1.exe
Pointer
=====
Isi dari variabel x : 2
Isi dari variabel x : 10

Isi dari array data kedua : 2
Isi dari array data x : 20

Cara Akses dengan Pointer
=====
Variabel primitif:
    Isi data : 2
    Alamat memori data : 2293528

Variabel reference:
    Alamat memori array angka: 2293524
    Alamat memori array angka: 2293524
    Alamat memori array angka: 2293524

    Isi dari array angka posisi ke 0: a
    Alamat memori array angka posisi ke 0: 2293524
    Isi dari array angka posisi ke 1: b
    Alamat memori array angka posisi ke 1: 2293525
    Alamat memori array angka posisi ke 1: 2293525
```

Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '08\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1  /*
2   * Nama File: 08_1.c
3  */
4  #include<stdio.h>
5
6  int main(){
7      int x = 2;
8      int *ptrx;
```

Baris 8: deklarasi pointer dengan nama ptrx bertipe int.

```

9      long arr[] = { 1,2,3 };
10     long *ptrarr;
```

Baris 10: deklarasi pointer dengan nama ptrx bertipe long.

```

11
12     printf("Pointer \n");
13     printf("===== \n");
14     printf("Isi dari variabel x : %d\n", x);
```

Baris 14: menampilkan nilai awal dari variabel x.

```
15     ptrx = &x;
```

Baris 15: pointer ptrx menunjuk ke alamat memori x

```
16     *ptrx = 10;
```

Baris 16: mengubah nilai dari alamat memori yang ditunjuk oleh ptrx menjadi nilai 10.

```
17     printf("Isi dari variabel x : %d\n\n", x);
```

Baris 17: menampilkan nilai dari variabel x sekarang.

```

18     printf("Isi dari array data kedua : %d\n", arr[1]);
19     ptrarr = &arr;
```

Baris 19: menampilkan nilai awal variabel array arr indeks pertama

```
20     *(ptrarr+1) = 20;
```

Baris 20: mengubah nilai array dengan menggunakan pointer

```
21     printf("Isi dari array data x : %d\n\n", arr[1]);
```

Baris 21: menampilkan nilai variabel array arr indeks pertama untuk sekarang

```
22
23     int data = 2;
24     char angka[] = {'a', 'b', 'c', 'd'};
25
26     printf("Cara Akses dengan Pointer \n");
27     printf("=====\\n");
28     printf("Variabel primitif:\\n");
29     printf("\\tIsi data : %d\\n", data);
30     printf("\\tAlamat memori data : %d\\n\\n", &data);
```

Baris 24: menampilkan alamat memroei dari data dengan menggunakan '&

```
31     printf("Variabel reference:\\n");
32     printf("\\tAlamat memori array angka: %d\\n", angka);
```

Baris 26: menampilkan alamat memori awal dari array tanpa tanda '&

```
33     printf("\\tAlamat memori array angka: %d\\n", &angka);
```

Baris 27: menampilkan alamat memori awal dari array dengan menggunakan tanda &

```
34     printf("\\tAlamat memori array angka: %d\\n\\n", &angka[0]);
```

Baris 28: menampilkan alamat memori awal dari array dengan menggunakan tanda & dan indeks array

```
35
36     printf("\\tIsi dari array angka posisi ke 0: %c\\n",
37         angka[0]);
38     printf("\\tAlamat memori array angka posisi ke 0: %d\\n",
39         &angka[0]);
40     printf("\\tIsi dari array angka posisi ke 1: %c\\n",
41         angka[1]);
42     printf("\\tAlamat memori array angka posisi ke 1: %d\\n",
43         &angka[1]);
44     printf("\\tAlamat memori array angka posisi ke 1: %d\\n",
45         &angka[0]+1);
```

Baris 36-40: menampilkan isi dan alamat dari array arr

```
42     getch();
43     return 0;
44 }
```

## Latihan Modul 8

1. Coba ketikkan dan jalankan program berikut serta simpan dengan nama 08\_a1.c,

```

1. #include<stdio.h>
2.
3. int main(){
4.     int angka[] = { 1,2,3,4 };
5.     int *ptrangka = angka;
6.
7.     int x = sizeof(angka) / sizeof(int);
8.
9.     printf("Nilai x : %d",x);
10.    getch();
11.    return 0;
12. }
```

Tambahkan beberapa data pada array angka dan lihat perubahan nilai dari x. Coba tarik kesimpulan dari percobaan tersebut.

2. Buatlah sebuah prosedur `showAllArrayPtr (int n, int data[100], int size)` yang menampilkan isi dari array data sebanyak n buah dengan menggunakan pointer. Jika nilai n melebihi jumlah data pada array maka muncul pesan kesalahan. Simpan berkas dengan nama 08\_02.c
3. Modifikasi prosedur `showAllArrayPtr` menjadi prosedur:  
`showAllArrayPtrParam(int n, int data[100], int size, int state)` dan simpan dengan nama 08\_03.c
- a. Jika nilai state = 0, maka prosedur akan menampilkan semua informasi array data yang berindeks genap
  - b. Jika tidak, maka prosedur akan menampilkan semua informasi array data yang berindeks ganjil
4. Buatlah prosedur `showArrayAddr(char data[20])` yang menampilkan alamat yang digunakan oleh array data. Simpan berkas dengan nama 08\_a4.c
5. Coba ketikkan dan jalankan program berikut serta simpan dengan nama 08\_a5.c,

```

1. #include<stdio.h>
2.
3. int main(){
4.     int angka1 = 5, angka2 = 15;
5.     int * p1, * p2;
6.     printf("Nilai Awal\n");
7.     printf("-----\n");
8.     printf("Angka 1 : %d\n", angka1);
9.     printf("Angka 2 : %d\n\n", angka2);
```

```
10.     p1 = &angka1;
11.     p2 = &angka2;
12.     *p1 = 10;
13.     *p2 = *p1;
14.     p1 = p2;
15.     *p1 = 20;
16.
17.
18.     printf("Nilai sekarang\n");
19.     printf("-----\n");
20.     printf("Angka 1 : %d\n", angka1);
21.     printf("Angka 2 : %d\n", angka2);
22.     getch();
23.     return 0;
24. }
```

Coba tarik kesimpulan dari keluaran program diatas.

6. Berdasarkan soal no 5, buatlah program untuk swap dua variabel dengan menggunakan pointer dan simpan dengan nama 08\_a6.c.

## Modul IX

### Pointer Lanjutan

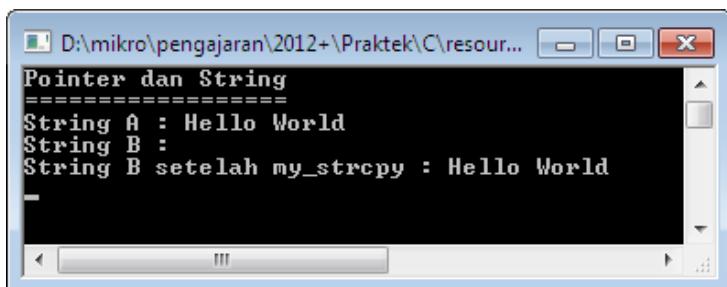
#### Objektif

Implementasi pointer untuk manipulasi string dan parameter fungsi.

#### Catatan

- ð Bentuk string pada bahasa C dapat direpresentasikan dengan pointer.
- ð Parameter fungsi pada bahasa C dapat berupa parameter byval dan byref.

#### Contoh 1



```
D:\mikro\pengajaran\2012+\Praktek\C\resour...
=====
Pointer dan String
=====
String A : Hello World
String B :
String B setelah my_strcpy : Hello World
```

Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '07\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*
2   Nama File: 09_1.c
3   */
4
5 #include<stdio.h>
6
7 void my_strcpy(char *d, char *s) {
```

Baris 7: membuat prosedur my\_strcpy yang menerima dua paramater pointer yang merepresentasikan bentuk string.

```
8     char *p = d;
```

Baris 8: membuat variabel pointer p yang merujuk bentuk string d.

```
9     while (*s != 0){
10         *p++ = *s++;
11     }
```

Baris 9-11: looping untuk membaca seluruh nilai dari bentuk string s dan memasukkannya ke memori yang ditunjuk oleh pointer p

```
12     *p = 0;
```

Baris 12: menambahkan penanda akhir dari bentuk string

```
13 }
14
15 int main(){
16     char strA[100] = "Hello World";
17     char strB[80] = "";
```

Baris 16 & 17: deklarasi dan inisialisasi bentuk string.

```
18     printf("Pointer dan String\n");
19     printf("=====*\n");
20     printf("String A : %s \n", strA);
21     printf("String B : %s \n", strB);
22     my_strcpy(strB, strA);
23     printf("String B setelah my_strcpy : %s \n", strB);
24
```

Baris 18-24: menampilkan nilai variabel sebelum dan sesudah memanggil prosedur my\_strcpy.

```
25     getch();
26     return 0;
27 }
```

## Contoh 2

```
D:\mikro\pengajaran\2012+\Prakt...
=====
Pointer dan Fungsi
=====
a : 3
b sebelum increase : 5
b setelah increase : 6
c sebelum perkalian : 0
c setelah perkalian : 18
```

Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '09\_2.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1 /*
2      Nama File: 09_2.c
3 */
4
5 #include<stdio.h>
6
7 void increase(int *a){
8     (*a)++;
9 }
```

Baris 7-9: mendefinisikan prosedur increase yang memiliki parameter pointer a dan isi dari prosedur adalah meningkatkan nilai yang ditunjuk pointer a.

```
10
11 void perkalianRekursif (int a, int b, int *c){
12     if (b!=0){
13         *c += a;
14         perkalianRekursif(a,b-1, c);
15     }
16 }
```

Baris 11-16: prosedur perkalianRekursif yang memanfaatkan parameter byref untuk menampung hasil perhitungan.

```
17
18 int main(){
19     int a = 3;
20     int b = 5;
21     int c = 0;
22     printf("Pointer dan Fungsi\n");
23     printf("=====\\n");
```

```
24     printf("a \t\t\t: %d \n\n", a);
25     printf("b sebelum increase\t: %d \n", b);
26     increase(&b);
```

Baris 26: memanggil prosedur increase yang sudah didefinisikan sebelumnya dengan sebuah parameter alamat dari variabel b.

```
27     printf("b setelah increase\t: %d \n\n", b);
28
29     printf("c sebelum perkalian\t: %d \n", c);
30     perkalianRekursif(a,b,&c);
```

Baris 30: memanggil prosedur perkalianRekursif yang menerima parameter variabel primitif dan pointer.

```
31     printf("c setelah perkalian\t: %d \n", c);
32 }
```

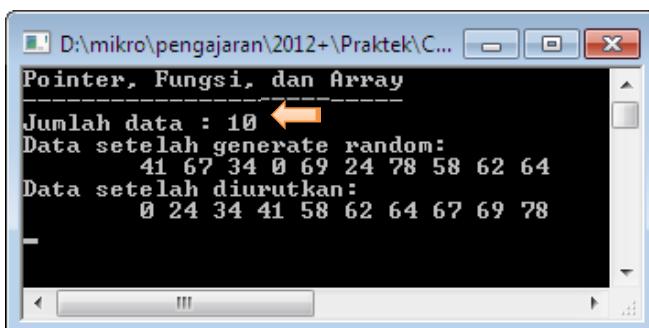
Baris 31: menampilkan hasil perkalian yang tersimpan pada variabel c.

```
33     getch();
34     return 0;
35 }
```

## Latihan Modul 9

1. Buatlah prosedur dengan menggunakan pointer (tanpa library string.h):
  - int my\_strlen(char \*s)
  - void my\_strcat(char \*hasil, char \*s1, char \*s2)
 Simpan dengan nama '09\_a1.c'.
  
2. Buatlah sebuah fungsi isNumeric(char \*s) yang digunakan untuk mengecek bentuk string merupakan bilangan atau bukan. Nb: belum mengecek nilai negatif ('-') dan positif ('+').
 Simpan dengan nama '09\_a2.c'
  
3. Buatlah sebuah prosedur fiboRekursif (int n, int \*hasil) yang digunakan untuk menghitung nilai fibonaci dan hasilnya akan disimpan pada parameter prosedur yang kedua. Simpan dengan nama '09\_03'
  
4. Buatlah sebuah prosedur :
  - generateRandomPtr (int \*array, int size) untuk memasukkan nilai array secara random antara 0-99 sebanyak size buah
  - showArrayPtr(int \*array, int size) untuk menampilkan nilai array sebanyak size buah.
  - bubbleSortPtr(int \*array, int size) untuk mengurutkan nilai array secara menaik.

Ketiga prosedur harus menggunakan pointer dan tidak menggunakan indeks dari array. Hasilkan keluaran seperti berikut dan simpan dengan nama '09\_a4'



```
Jumlah data : 10
Data setelah generate random:
 41 67 34 0 69 24 78 58 62 64
Data setelah diurutkan:
 0 24 34 41 58 62 64 67 69 78
```

## Modul X

### Command Line Arguments

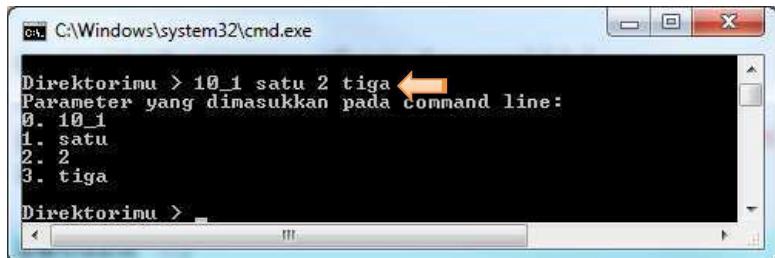
#### Objektif

Membaca dan memproses parameter yang dikirim oleh pengguna melalui command line.

#### Catatan

- Eksekusi program bahasa C dapat dilakukan melalui command line dengan menuliskan nama file eksekusinya, seperti 10\_2
- Argument dituliskan setelah nama file eksekusinya.
- Pemisah antar argument/parameter pada command line adalah tanda spasi (' ')

#### Contoh 1



Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '10\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*
2   *      Nama File: 10_1.c
3   */
4
5  #include<stdio.h>
6
7  int main(int argc, char *argv[]) {
```

Baris 7: menambahkan parameter pada fungsi main, yaitu parameter integer dan parameter pointer of array.

```

8     if (argc <=1)
9         printf("Tidak argumen yang dimasukkan pada command
line.");

```

Baris 8 & 9: pengecekan argc yang telah menyimpan jumlah dari argumen pada command line.

```

10    else {
11        int i;
12        printf("Parameter yang dimasukkan pada command line:
\n");
13        for (i=0; i<argc; i++)
14            printf("%d. %s \n", i, argv[i]);
15    }
16

```

Baris 10-15: menampilkan seluruh parameter yang dimasukkan pada command line.

```

17     getch();
18     return 0;
19 }

```

## Contoh 2

```

C:\Windows\system32\cmd.exe
Direktorimu $ 10_2 1 12 23
Hasil Penjumlahan dari 12 dan 23 = 35
Direktorimu $ 10_2 2 12 23
Hasil Pengurangan dari 12 dan 23 = -11
Direktorimu $ 10_2 3 12 23
Hasil Perkalian dari 12 dan 23 = 276
Direktorimu $ 10_2 4
10_2 [/2/3] [param1] [param2].
1: menambahkan nilai [param1] dengan [param2]
2: mengurangi nilai [param1] dengan [param2]
3: mengalikan nilai [param1] dengan [param2]
Direktorimu $

```

Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '10\_2.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```

1  /*
2   Nama File: 10_2.c
3   */
4
5 #include<stdio.h>
6 #include<stdlib.h>

```

Baris 6: menambahkan library stdlib yang mendefinisikan fungsi atoi.

```

7  int main(int argc, char *argv[]){
8      if (argc != 4){
9          printf("10_2 [1/2/3] [param1] [param2].\n");
10         printf("\t1: menambahkan nilai [param1] dengan [param2]
11             \n");
12         printf("\t2: mengurangi nilai [param1] dengan [param2]
13             \n");
13     }

```

Baris 8: mengecek jumlah perintah yang dimasukkan, jika perintah lebih sedikit dari seharusnya, maka muncul informasi penggunaan perintah

```

14     else {
15         int command = atoi(argv[1]);
16         int param1 = atoi(argv[2]);
17         int param2 = atoi(argv[3]);

```

Baris 15-17: perintah atoi melakukan konversi nilai dari bentuk string ke bentuk integer.

```

18         switch (command) {
19             case 1 :
20                 printf("Hasil Penjumlahan dari %d dan %d
21 = %d\n", param1, param2, param1+param2);
21                 break;
22             case 2 :
23                 printf("Hasil Pengurangan dari %d dan %d
23 = %d\n", param1, param2, param1-param2);
24                 break;
25             case 3 :
26                 printf("Hasil Perkalian dari %d dan %d =
26 %d\n", param1, param2, param1*param2);
27                 break;
28             default :
29                 printf("Perintah tidak dikenal\n");
30         }
31     }

```

Baris 18-27: menyeleksi parameter yang dimasukkan dan melakukan operasi sesuai dengan perintah.

```

32     getch();
33     return 0;
34 }

```

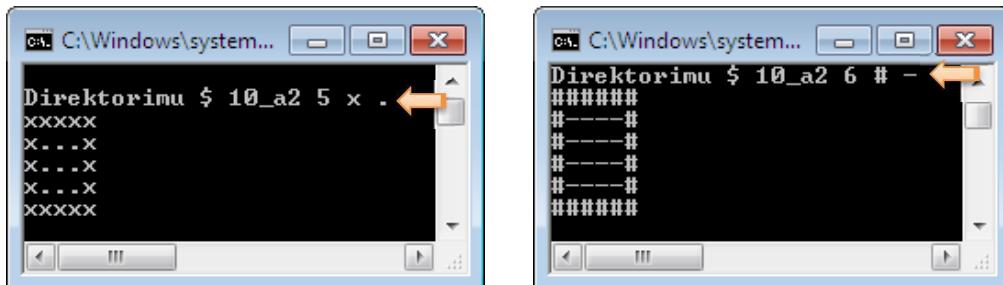
## Latihan Modul 10

1. Modifikasi **Contoh 2 Modul X** menjadi:

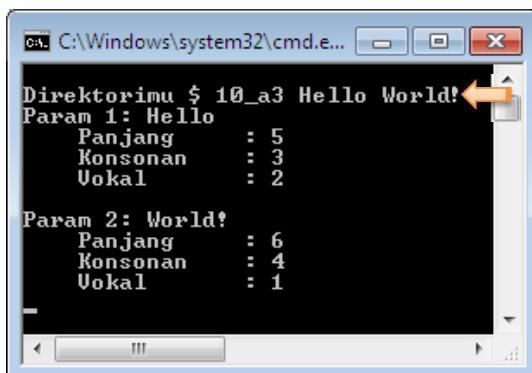
- 1 : menambahkan seluruh nilai parameter berikutnya,  
Misal: param1 + param2 + param3 + ...
- 2 : mengurangi seluruh nilai parameter berikutnya,  
Misal: param1 - param2 - param3 - ...
- 3 : mengalikan seluruh nilai parameter berikutnya,  
Misal: param1 \* param2 \* param3 \* ...

Jumlah param tergantung dari jumlah parameter yang dimasukkan melalui command line.  
Simpan dengan nama '10\_a1.c'

2. Buatlah program yang menerima tiga parameter, berupa angka, karakter1, dan karakter2. Informasi tersebut digunakan untuk membentuk pola seperti berikut dan simpan dengan nama '10\_a2.c'



3. Buatlah program yang menerima parameter berupa sebuah string kemudian string tersebut akan dianalisis berupa panjang string, jumlah konsonan, dan jumlah vokal. Hasilkan keluaran seperti berikut dan simpan dengan nama '10\_a3.c'



## Modul XI

### Struct

#### Objektif

Memahami penggunaan struct dan implementasi dalam bahasa C.

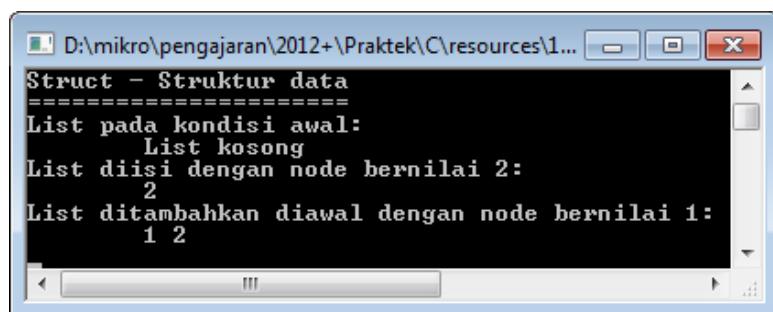
#### Catatan

- Struct pada bahasa C memiliki bentuk:

```
struct nama_struct {
    type nama_var1;
    type nama_var2;
    ...
};
```

- Struktur pada struct dapat berupa struct lain atau dirinya sendiri

#### Contoh



```
D:\mikro\pengajaran\2012+\Praktek\C\resources\1...
Struct - Struktur data
=====
List pada kondisi awal:
List kosong
List diisi dengan node bernilai 2:
2
List ditambahkan diawal dengan node bernilai 1:
1 2
```

Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '11\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*
2   * Nama File: 11_1.c
3   */
4
5  #include<stdio.h>
6
7
```

```

8  struct node {
9      int data;
10     struct node *next;
11 };

```

Baris 8-11: membuat struktur node yang berisi data dan pointer ke node.

```

12
13 struct node *head = NULL;

```

Baris 13: deklarasi pointer head bertipe node dan langsung diisi dengan nilai NULL.

```

14
15 void init(int v){
16     struct node *ptr = (struct node*) malloc(sizeof(struct
node));

```

Baris 16: deklarasikan pointer ptr dan alokasikan memori untuk ptr.

```

17     if (ptr == NULL)
18         printf("Node gagal dibuat");

```

Baris 17&18: menampilkan node gagal dibuat jika ptr masih NULL

```

19     ptr->data = v;

```

Baris 19: mengisi pointer data dengan v

```

20     ptr->next = NULL;

```

Baris 20: mengisi nilai next dengan NULL.

```

21     head = ptr;

```

Baris 21: mengubah pointer head menjadi yang ditunjuk oleh ptr.

```

22 }
23
24 void addFirstNode(int v){
25     struct node *ptr = (struct node*) malloc(sizeof(struct
node));
26     if (ptr == NULL)
27         printf("Node gagal dibuat");
28     ptr->data = v;
29     ptr->next = head;

```

Baris 29: mengubah next dari ptr menjadi head.

```
30     head = ptr;
```

Baris 30: mengubah pointer head menjadi yang ditunjuk oleh ptr.

```
31 }
32
33 void showList(){
34     struct node *ptr = head;
```

Baris 34: deklarasikan ptr dan diinisialisasi dengan head

```
35     printf("\t");
36     if (ptr == NULL)
37         printf("List kosong");
38     while (ptr!=NULL){
39         printf("%d ", ptr->data);
40         ptr = ptr->next;
41     }
42     printf("\n");
43 }
```

Baris 38-42: melakukan penelusuran pada list, sampai ptr menunjuk ke NULL.

Jika ptr tidak NULL, maka tampilkan nilai yang diisi oleh ptr, kemudian ubah nilai ptr ke pointer yang ditunjuk oleh next.

```
44
45 int main(){
46     printf("Struct - Struktur data\n");
47     printf("=====\\n");
48     printf("List pada kondisi awal: \\n");
49     showList();
50     printf("List diisi dengan node bernilai 2:\\n");
51     init(2);
52     showList();
53     printf("List ditambahkan diawal dengan node bernilai
54     1:\\n");
55     addFirstNode(1);
56     showList();
```

Baris 46-55: mengimplementasikan prosedur yang telah dibuat sebelumnya.

```
57     getch();
58     return 0;
59 }
```

## Latihan Modul 11

1. Modifikasi program **Contoh Modul XI** dengan menambahkan informasi pada struct node dengan warna, misal 1, 2, 3, dan lain-lain. Sesuaikan kembali parameter prosedur untuk mengikuti perubahan informasi struct node. Simpan perubahan dengan nama '11\_a1.c'
2. Berdasarkan **Contoh Modul XI**, tambahkan prosedur addLastNode(int v). Fungsi akan membuat node baru dan menambahkannya pada akhir List. Simpan perubahan dengan nama '11\_a2.c'
3. Berdasarkan **Soal no 2**, tambahkan fungsi int Find(int v). Fungsi akan mencari nilai v pada list, jika tidak ditemukan, fungsi akan mengembalikan nilai -1, jika ditemukan, fungsi akan mengembalikan posisi indeks pada list. Misal: Find(7) = 3



Simpan perubahan dengan nama '11\_a3.c'

4. Buatlah program struct barang yang berisi nama barang dan harga barang. Deklarasikan variabel Barang berupa array dari struct barang, kemudian tambahkanlah tiga data barang secara bebas. Sebaiknya buat prosedur addbarang (struct barang b[10], int indeks, char \*nama, long harga). Simpan dengan nama '11\_a4.c'

## Modul XII

### File

#### Objektif

Memahami kegunaan file dan implementasinya dalam bahasa C.

#### Catatan

- File pada bahasa C memiliki bentuk:

```
FILE* variabel;
```

- Bahasa C memiliki dua jenis file antara lain file teks dan file biner.
- Operasi dasar file:

- Deklarasi variable File
- Buka File
- Operasi File (baca/tulis)
- Tutup File

#### Contoh 1



Langkah penggeraan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama '12\_1.c'.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1 /*  
2      Nama File: 12_1.c  
3 */
```

```

4 #include<stdio.h>
5
6 void addData(char namaFile[100], char nim[10], double nilai){
7     FILE *vf = fopen(namaFile, "a");
8     if (vf != NULL) {
9         fprintf(vf, "%s %lf\n", nim, nilai);
10        fclose(vf);
11    }
12 }

```

Baris 7-13: fungsi menambahkan data mahasiswa pada file teks.

- ð fopen merupakan fungsi yang digunakan untuk membuka file.
- ð Paramater “a” pada fungsi fopen berarti data akan ditambahkan diakhir file teks.
- ð fprintf merupakan fungsi pada library stdio.h yang digunakan untuk menulis data teks pada file. Fungsi ini memiliki bentuk yang mirip dengan fungsi printf, dengan penambahan parameter variabel file.
- ð fclose digunakan untuk menutup file yang sudah dibuka sebelumnya.

```

13 void showAllData(char namaFile[100]){
14     FILE *vf = fopen(namaFile, "r");
15     if (vf != NULL) {
16         while (!feof(vf)){
17             char nim[10];
18             double nilai;
19             fscanf(vf, "%s", nim);
20             fscanf(vf, "%lf\n", &nilai);
21             printf("%s %0.2lf\n", nim, nilai);
22         }
23         fclose(vf);
24     }
25 }

```

Baris 11-17: fungsi menambahkan data mahasiswa pada file teks.

- ð Paramater “r” pada fungsi fopen berarti data akan ditambahkan diakhir file teks.
- ð fscanf merupakan fungsi pada library stdio.h yang digunakan untuk membaca data pada file teks. Fungsi ini memiliki bentuk yang mirip dengan fungsi scanf, dengan penambahan parameter variabel file.

```

26 int main(){
27     printf("Demo File Teks\n");
28     printf("-----\n");
29     addData("datafile.txt", "121110010", 87.5);
30     addData("datafile.txt", "121110011", 80.5);

```

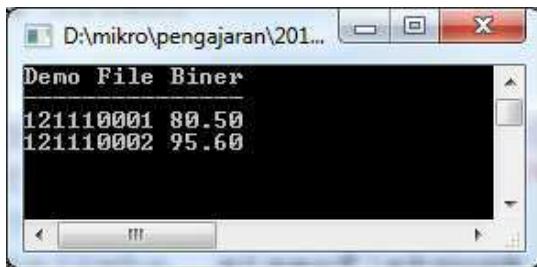
Baris 30-31: menambahkan dua data pada file teks.

```
31     showAllData("datateks.txt");
```

Baris 32: menampilkan seluruh data pada file teks “datateks.txt”.

```
32
33     getch();
34     return 0;
35 }
```

## Contoh 2



Langkah pengerjaan kode program:

1. Buatlah lembar kerja baru (**Ctrl+N**) dan simpan (**Ctrl+F12**) dengan nama ‘12\_2.c’.
2. Masukkan kode program pada lembar kerja editor yang telah tersimpan:

```
1  /*
2   * Nama File: 12_2.c
3   */
4
5 #include<stdio.h>
6 struct mahasiswa{
7     char nim[10];
8     double nilai;
9 };
```

Baris 6-9: mendefinisikan struct mahasiswa yang berisi nim dan nilai.

```
10
11 void addDataBiner(char namaFile[100], struct mahasiswa mhs) {
12     FILE *vf = fopen(namaFile, "ab");
13     if (vf != NULL) {
14         fwrite(&mhs, sizeof(struct mahasiswa), 1, vf);
15         fclose(vf);
16     }
17 }
```

Baris 11-17: fungsi menambahkan data mahasiswa pada file biner.

- ð Paramater “ab” pada fungsi fopen berarti data akan ditambahkan diakhir file biner.

- ð fwrite merupakan fungsi pada library stdio.h yang digunakan untuk menulis data biner pada file.
- ð fclose digunakan untuk menutup file yang sudah dibuka sebelumnya.

```

18 void showAllDataBiner(char namaFile[100]){
19     FILE *vf = fopen(namaFile,"rb");
20
21     if (vf != NULL){
22         while (!feof(vf)){
23             struct mahasiswa mhs;
24             fread(&mhs,sizeof(struct mahasiswa),1,vf);
25             if (!feof(vf))
26                 printf("%s %0.2lf\n", mhs.nim, mhs.nilai);
27         }
28     }
29 }
30 }
```

Baris 18-30: fungsi menampilkan data mahasiswa pada file biner.

- ð Paramater “rb” pada fungsi fopen berarti data akan dibaca dari file biner.
- ð fread merupakan fungsi pada library stdio.h yang digunakan untuk membaca data biner pada file biner.
- ð Fungsi feof akan bernilai 0 jika belum mencapai akhir dari file.

```

31 int main(){
32     printf("Demo File Biner\n");
33     printf("-----\n");
34     struct mahasiswa mhs = {"121110001",80.5};
35     struct mahasiswa mhs2 = {"121110002",95.6};
36     addDataBiner("databiner.txt",mhs);
37     addDataBiner("databiner.txt",mhs2);
```

Baris 34-37: mendeklarasikan dan inisialisasi dua data mahasiswa dan menyimpannya pada file biner “databiner.txt”

```
38     showAllDataBiner("databiner.txt");
```

Baris 38: memanggil fungsi untuk menampilkan seluruh data pada file biner “databiner.txt”

```

39     getch();
40     return 0;
41 }
```

## Latihan Modul 12

1. Modifikasi program **Contoh 1 dan Contoh 2 Modul XII** dengan menambahkan fungsi search yang mengembalikan nilai dari mahasiswa:
  - a. double searchData(char namaFile[100], char nim[10])
  - b. struct mahasiswa searchDataBiner(char namaFile[100], char nim[10])
2. Modifikasi program **Contoh 1 dan Contoh 2 Modul XII** dengan menambahkan fungsi yang mengembalikan data mahasiswa pada posisi tertentu:
  - a. struct mahasiswa searchData(char namaFile[100], int posisi)
  - b. struct mahasiswa searchDataBiner(char namaFile[100], int posisi)

Nb: Tambahkan definisi struct mahasiswa pada file teks!

Posisi dimulai dari angka satu(1) yang menandai awal dari data file.

Jika posisi tidak ditemukan, maka kembalikan nim: "" dan nilai 0.
3. Ubah program **Soal 10\_a2 Modul X** dengan menampilkan hasil pola pada file "pola.txt" dan simpan dengan nama 12\_a3.c

## LAMPIRAN

### ESCAPE CHARACTER

CHARACTER	ASCII REPRESENTATION	ESCAPE CHARACTER
NewLine	NL (LF)	\n
Horizontal tab	HT	\t
Vertical tab	VT	\v
Backspace	BS	\b
Carriage return	CR	\r
Formfeed	FF	\f
Alert	BEL	\a
Backslash	\	\\\
Question mark	?	\?
Single quotation mark	'	\'
Double quotation mark	"	\"
Octal Number	ooo	\ooo
Hexadecimal number	hhh	\xhhh
Null character	NUL	\0

### TABEL ASCII

0	24	↑	48	0	72	H	96	`	120	x	144	É	168	¸	192	Ł	216	±	240	☰
1	25	↓	49	1	73	I	97	a	121	y	145	æ	169	ѓ	193	ł	217	„	241	±
2	26	→	50	2	74	J	98	b	122	z	146	Œ	170	ѓ	194	ł	218	„	242	≥
3	27	←	51	3	75	K	99	c	123	{	147	ô	171	ќ	195	ł	219	„	243	≤
4	28	„	52	4	76	L	100	d	124		148	ö	172	ќ	196	—	220	„	244	„
5	29	„	53	5	77	M	101	e	125	)	149	ð	173	ќ	197	+	221	„	245	„
6	30	^	54	6	78	N	102	f	126	~	150	û	174	«	198	F	222	„	246	÷
7	31	▼	55	7	79	O	103	g	127	Δ	151	ù	175	»	199		223	„	247	„
8	32	„	56	8	80	P	104	h	128	Ҫ	152	ü	176	‰	200		224	α	248	◦
9	33	!	57	9	81	Q	105	i	129	ü	153	Ӯ	177	‰	201		225	ڦ	249	·
10	34	“	58	:	82	R	106	j	130	é	154	Ü	178	‰	202		226	Gamma	250	·
11	35	#	59	:	83	S	107	k	131	â	155	ც	179	‰	203		227	Π	251	„
12	36	\$	60	<	84	T	108	l	132	ä	156	£	180	‰	204		228	Σ	252	„
13	37	%	61	=	85	U	109	m	133	à	157	¥	181	‰	205	=	229	σ	253	²
14	38	&	62	>	86	U	110	n	134	å	158	₱	182	‰	206		230	μ	254	■
15	39	·	63	?	87	W	111	o	135	ç	159	ƒ	183	‰	207	±	231	γ	255	a
16	40	(	64	@	88	X	112	p	136	ê	160	á	184	‰	208		232	૧		
17	41	)	65	A	89	Y	113	q	137	ë	161	í	185	‰	209		233	θ		
18	42	*	66	B	90	Z	114	r	138	è	162	ó	186	‰	210		234	Ω		
19	43	+	67	C	91	[	115	s	139	ï	163	ú	187	‰	211		235	δ		
20	44	,	68	D	92	\	116	t	140	î	164	ñ	188	‰	212		236	ω		
21	45	-	69	E	93	]	117	u	141	í	165	Ñ	189	‰	213	F	237	∅		
22	46	.	70	F	94	^	118	v	142	Ä	166	¤	190	‰	214		238	€		
23	47	/	71	G	95	_	119	w	143	å	167	º	191	‰	215		239	∏		

## KEYWORD BAHASA C

auto	break	case	char	const	continue	Default	do
double	else	enum	extern	float	for	Goto	if
int	long	register	return	short	signed	Sizeof	static
struct	switch	typedef	union	unsigned	void	Volatile	while

## FORMAT TIPE DATA

Specifier	Data Type
%c	Character
%d	Decimal integer
%o	Octal Integer
%x	Hexadecimal Integer
%u	Unsigned decimal integer
%ld	Long int
%f	Floating point
%lf	Double or long double
%e	Exponential floating point
%s	Character string