

## **Kegiatan Belajar 11**

### **FUNGSI**

#### **A. Pendahuluan**

##### **1. Deskripsi Singkat**

Secara umum, tujuan kegiatan belajar 11 ini adalah untuk memberikan pengetahuan, keterampilan, dan sikap mahasiswa tentang konsep dasar teknologi informasi.

Secara khusus, tujuan kegiatan belajar 11 ini adalah agar peserta mampu: (1) menjelaskan konsep array satu dimensi dan multidimensi, (2) merancang variabel array satu dimensi dan multi dimensi terhadap array, (3) membuat program yang menerapkan operasi terhadap array satu dimensi dan dua dimensi dalam pemrograman C.

##### **2. Panduan Belajar**

Proses pembelajaran untuk materi modul 3 kegiatan belajar 8 dapat berjalan dengan lancar apabila Anda mengikuti langkah-langkah belajar sebagai berikut:

1. Pahami dulu kegiatan penting dalam program pelatihan ini dengan memperhatikan isi capaian pembelajaran setiap kegiatan belajar.
2. Lakukan kajian terhadap setiap materi dalam kegiatan belajar, agar memudahkan proses pembelajaran.
3. Pelajari dahulu kegiatan belajar 5 yang setiap akhir kegiatan belajar menyelesaikan tugas yang harus dikerjakan secara langsung.
4. Keberhasilan program pembelajaran ini tergantung dengan kesungguhan Anda dalam mengerjakan setiap tugas dalam kegiatan belajar.
5. Bila Anda menemukan kesulitan, silahkan hubungi instruktur pembimbing atau fasilitator yang mengajar modul ini.

#### **B. Inti**

##### **1. Capaian Pembelajaran Matakuliah**

Setelah mengikuti seluruh tahapan pada kegiatan belajar, peserta menerapkan berbagai alur kontrol perulangan dalam program.

##### **2. Pokok Materi**

- a. Fungsi

### **3. Uraian Materi**

#### **1. Fungsi**

Secara umum fungsi mengacu pada sub program yang berisi serangkaian pernyataan yang dilaksanakan tersendiri. Dalam algoritma fungsi dapat berupa procedure dan function. Procedure adalah sub program yang tidak mengembalikan nilai apapun setelah mengerjakan tugasnya (pelaksanaan pernyataan dalam prosedur) sedangkan function merupakan sub program yang mengembalikan suatu nilai setelah mengerjakan tugasnya.

Dalam Bahasa C (dan juga Bahasa lainnya seperti C++, C#, Java dan lain-lain), kedua jenis sub program tersebut sama-sama disebut fungsi. Sehingga dalam C, fungsi dapat dibedakan menjadi fungsi dengan nilai balik dan fungsi tanpa nilai balik.

Selain itu, fungsi dapat memiliki passing parameter atau tidak memiliki passing parameter. Fungsi yang memiliki passing parameter pun dapat dibedakan menjadi dua jenis passing parameter, yaitu, fungsi dengan passing parameter by value dan fungsi dengan passing parameter by reference.

#### **2. Parameter**

Dalam bahasa pemrograman C, "parameter" merujuk pada variabel yang digunakan dalam deklarasi atau definisi fungsi atau prosedur untuk menerima data atau argumen dari pemanggil fungsi. "Passing parameter" adalah proses mengirimkan nilai atau argumen ke dalam fungsi atau prosedur tersebut. Parameter digunakan untuk mengirim data ke dalam fungsi atau prosedur agar dapat diolah di dalamnya. Berikut penjelasan lebih lanjut tentang parameter dan passing parameter dalam bahasa C:

##### **Parameter dalam Deklarasi Fungsi:**

- Parameter didefinisikan dalam deklarasi fungsi atau prosedur sebagai bagian dari tipe data fungsi.
- Parameter digunakan untuk menunjukkan jenis dan jumlah data yang diterima oleh fungsi.
- Parameter dinyatakan dalam kurung kurawal setelah nama fungsi, dan mereka memiliki nama dan tipe data.
- Contoh deklarasi fungsi dengan parameter:

```
c
int tambah(int a, int b);
```

Dalam contoh ini, a dan b adalah parameter yang akan menerima dua bilangan bulat untuk dijumlahkan.

#### 🚦 Passing Parameter:

- Ketika Anda memanggil fungsi atau prosedur, Anda harus memberikan nilai yang sesuai untuk setiap parameter yang dideklarasikan dalam fungsi tersebut.
- Nilai yang diberikan saat pemanggilan fungsi disebut "argumen" atau "nilai aktual".
- Argumen yang diberikan harus sesuai dengan tipe data dan urutan parameter yang didefinisikan dalam fungsi.
- Contoh pemanggilan fungsi dengan argumen:

```
c
int hasil = tambah(5, 3);
```

Dalam contoh ini, fungsi tambah dipanggil dengan dua argumen, yaitu 5 dan 3, yang akan diterima oleh parameter a dan b.

#### 🚦 Penggunaan Parameter dalam Fungsi:

- Parameter dapat digunakan di dalam fungsi untuk melakukan operasi atau perhitungan.
- Nilai dari parameter dapat diakses dan digunakan seperti variabel dalam tubuh fungsi.
- Contoh penggunaan parameter dalam fungsi:

```
c
int tambah(int a, int b) {
    int hasil = a + b;
    return hasil;
}
```

#### ✚ **Passing Parameter dengan Referensi (by Reference):**

- Dalam C, parameter biasanya "dilewatkan" dengan nilai (by value), yang berarti fungsi menerima salinan nilai argumen.
- Anda juga dapat mengirim parameter dengan referensi (by reference) menggunakan pointer. Dengan cara ini, fungsi menerima akses ke variabel asli yang dilewatkan, bukan salinan nilainya.
- Ini memungkinkan fungsi untuk mengubah nilai variabel yang dilewatkan.
- Contoh penggunaan parameter dengan referensi:

```
c
void ubahNilai(int *x) {
    *x = *x + 1;
}
```

Dalam contoh ini, fungsi `ubahNilai` menerima pointer ke sebuah integer, dan ini memungkinkan fungsi untuk mengubah nilai integer yang diketahui oleh pemanggil.

#### ✚ **Passing Parameter dengan Value (by value):**

"Passing parameter by value" adalah teknik pengiriman argumen ke dalam fungsi atau prosedur dalam pemrograman di mana fungsi atau prosedur menerima salinan nilai argumen yang dilewatkan, bukan mengakses variabel asli yang dikirim. Dalam konteks ini, "by value" berarti bahwa nilai yang dikirimkan ke fungsi atau prosedur tidak dapat diubah di dalam fungsi tersebut, dan perubahan yang mungkin terjadi hanya berlaku di dalam fungsi itu sendiri.

```

c

#include <stdio.h>

// Fungsi yang menerima dua parameter dengan cara "by value"
int tambah(int a, int b) {
    int hasil = a + b;
    return hasil;
}

int main() {
    int angka1 = 5;
    int angka2 = 3;

    int hasilTambah = tambah(angka1, angka2); // Memanggil fungsi tambah

    printf("Hasil penambahan: %d\n", hasilTambah);

    return 0;
}

```

Dalam contoh di atas, fungsi tambah menerima dua argumen (a dan b) dengan cara "by value". Ini berarti bahwa ketika Anda memanggil fungsi tambah dengan angka1 dan angka2, fungsi tersebut akan bekerja dengan salinan nilai dari angka1 dan angka2. Oleh karena itu, perubahan yang mungkin terjadi pada a dan b di dalam fungsi tambah tidak akan memengaruhi nilai angka1 dan angka2 di dalam main.

### 3. Prosedur dan Fungsi

Fungsi dan prosedur adalah dua konsep yang berbeda dalam bahasa pemrograman C, meskipun keduanya digunakan untuk mengelompokkan blok kode yang dapat dipanggil dari tempat lain dalam program. Berikut adalah perbedaan antara fungsi dan prosedur dalam bahasa C beserta beberapa keyword yang terkait:

## 1. Fungsi (Function):

- Fungsi adalah blok kode yang dapat menerima argumen (input) dan mengembalikan nilai (output).
- Mereka dapat mengembalikan nilai dengan menggunakan pernyataan `return`.
- Fungsi dapat mengembalikan satu nilai.
- Keyword yang digunakan untuk mendefinisikan fungsi adalah `int`, `float`, `char`, `void`, dll., yang menentukan tipe data kembalian.
- Contoh deklarasi fungsi:

```
c
int tambah(int a, int b);
```

## 2. Prosedur (Procedure):

- Prosedur adalah blok kode yang dapat menerima argumen (input) tetapi tidak mengembalikan nilai (output).
- Mereka tidak menggunakan pernyataan `return` untuk mengembalikan nilai.
- Prosedur digunakan ketika Anda hanya ingin menjalankan serangkaian instruksi tanpa perlu mengembalikan nilai.
- Keyword yang digunakan untuk mendefinisikan prosedur adalah `void`.
- Contoh deklarasi prosedur:

```
c
void cetakPesan(char *pesan);
```

Perlu diingat bahwa dalam praktiknya, fungsi dan prosedur sering disebut dengan istilah "fungsi" secara umum, terutama dalam bahasa pemrograman C. Namun, perbedaan antara keduanya penting ketika Anda bekerja dengan bahasa pemrograman yang lebih canggih atau tipe data yang lebih kompleks. Dalam C, penggunaan `void` untuk menunjukkan bahwa suatu fungsi adalah prosedur adalah konvensi yang baik agar kode menjadi lebih mudah dipahami.

#### **4. Hubungan Fungsi dan Prosedur dengan Pemrograman Modular**

Dalam pemrograman modular, fungsi dan prosedur memiliki peran penting dalam membangun struktur modular yang efisien dan terorganisir. Kedua konsep ini bekerja sama untuk mengelompokkan kode program ke dalam unit-unit yang lebih kecil, yang dapat meningkatkan keterbacaan, pemeliharaan, dan reusabilitas kode. Berikut adalah hubungan antara fungsi dan prosedur dalam pemrograman modular:

##### **1. Fungsi sebagai Komponen Modular:**

- Fungsi adalah unit dasar pemrograman modular yang sering digunakan untuk mengelompokkan kode yang memiliki tujuan atau tugas tertentu.
- Fungsi dapat menerima argumen (parameter) dan mengembalikan nilai (return value) sesuai dengan kebutuhan.
- Setiap fungsi seharusnya memiliki tugas yang jelas dan spesifik dalam program.
- Fungsi digunakan untuk memecah program menjadi bagian-bagian yang lebih kecil, sehingga Anda dapat fokus pada satu tugas pada satu waktu.

##### **2. Prosedur (atau Fungsi Void) dalam Modularitas:**

- Prosedur (fungsi yang mengembalikan `void`) juga merupakan komponen modular yang penting dalam pemrograman modular.
- Seperti fungsi, prosedur digunakan untuk mengelompokkan kode yang terkait, tetapi mereka tidak mengembalikan nilai.
- Proses-proses yang tidak perlu mengembalikan nilai biasanya diimplementasikan sebagai prosedur.
- Prosedur membantu memisahkan logika program dari efek samping (side effects) yang mungkin terjadi.

##### **3. Hubungan dan Interaksi:**

- Fungsi dan prosedur sering berinteraksi satu sama lain dalam program modular. Misalnya, sebuah fungsi dapat memanggil satu atau lebih prosedur untuk melakukan tugas tertentu.
- Dengan mengatur kode ke dalam fungsi dan prosedur, Anda dapat membuat kode menjadi lebih terstruktur dan mudah dipahami.
- Fungsi dapat menerima argumen yang kemudian diteruskan ke prosedur yang lebih khusus.

Contoh sederhana hubungan antara fungsi dan prosedur dalam pemrograman modular:

```
c Copy code

// Fungsi untuk menghitung total harga dengan diskon
float hitungTotal(float hargaAwal, float diskon) {
    float hargaDiskon = hargaAwal - (hargaAwal * diskon);
    return hargaDiskon;
}

// Prosedur untuk mencetak struk belanja
void cetakStruk(float totalHarga) {
    printf("Total belanja: %.2f\n", totalHarga);
    // Proses mencetak struk lainnya
}

int main() {
    float hargaBarang = 100.0;
    float diskon = 0.1;

    float total = hitungTotal(hargaBarang, diskon); // Memanggil fungsi
    cetakStruk(total); // Memanggil prosedur

    return 0;
}
```

Dalam contoh di atas, `hitungTotal` adalah fungsi yang menghitung total harga dengan diskon, sementara `cetakStruk` adalah prosedur yang mencetak struk belanja. Kedua komponen ini bekerja bersama dalam program modular untuk mencapai tujuan yang diinginkan.

## 5. Berbagai Konstruksi Fungsi

Fungsi dapat memiliki berbagai bentuk implementasi. Dalam bahasa C, bentuk fungsi tersebut dapat dibedakan atas :

Tabel 1. Tabel Jenis Fungsi

No	Jenis Fungsi	Keterangan
1.	Fungsi tanpa parameter dan tanpa nilai balik	fungsi ini tidak memerlukan argumen apapun saat pemanggilannya serta tidak



		memberikan nilai balik kepada pemanggil fungsi tersebut.
2.	Fungsi memiliki parameter dan tanpa nilai balik	Fungsi jenis ini membutuhkan argumen atau parameter saat pemanggilannya dan tidak memberikan nilai balik kepada pemanggil fungsi tersebut
3.	Fungsi tidak memiliki parameter dan memberikan nilai balik	Fungsi ini tidak memerlukan argumen apapun saat pemanggilannya dan setelah mengerjakan tugasnya, fungsi ini memberikan suatu nilai balik
4.	Fungsi memiliki parameter dan memberikan nilai balik	Fungsi jenis ini membutuhkan argumen atau parameter ketika dipanggil lalu setelah mengerjakan tugasnya, fungsi tersebut memberikan nilai balik kepada pemanggilnya

### 1. Fungsi tanpa parameter dan tanpa nilai balik

Dalam algoritma dan beberapa bahasa, fungsi yang tidak mengembalikan apapun (tanpa nilai balik) selain hanya melakukan tugas yang diberikan, disebut prosedur (procedure). Sehingga definisi dari fungsi (atau prosedur) seperti ini berbentuk seperti berikut:

***Procedure namaProsedur()***

***Deklarasi***

***Konstanta:***

***Daftar konstanta yang digunakan***

***Variable:***

***Daftar variable yang digunakan***

***Deskripsi***

***Start***

***Aksi-aksi***

***end***

Dalam bahasa C, bentuk fungsi tanpa parameter dan tanpa nilai balik memiliki cara penulisan seperti berikut:

```
void namaFungsi ()  
{  
//deklarasi dari konstanta dan variable yg digunakan  
Aksi-aksi;  
}
```

Penjelasan:

Keyword void, berarti bahwa fungsi tersebut tidak memiliki nilai balik apapun.

namaFungsi, merupakan nama dari fungsi tersebut yang digunakan saat fungsi ini dipanggil dari bagian lain program.

1. Contoh 1: Buatlah suatu fungsi yang bertugas menampilkan judul program.

Jawab:

Implementasi fungsi tersebut hanya mencetak ke layar beberapa informasi terkait suatu judul program. Misalnya dapat terdiri dari informasi sebagai berikut:

- Menampilkan judul program
- Menampilkan identitas programmer
- Menampilkan keterangan tentang program

Algoritma fungsi dapat berupa:

```
procedure judul()  
deskripsi  
start  
write('PROGRAM RAMALAN BINTANG')  
write('Programmer: Nama Programmer')  
write('Program ini akan meminta anda mengetikkan tanggal kelahiran')  
write('Berdasarkan tanggal, bulan dan tahun kelahiran, program akan')  
write('Menentukan bintang kelahiran dan ramalan tentang Anda')  
end
```

Catatan: dalam algoritma dan beberapa Bahasa pemrograman lain, fungsi yang tidak mengembalikan nilai disebut prosedur sedangkan dalam Bahasa C, C++, C#, Java dan

banyak Bahasa lain, semua sub program disebut fungsi.

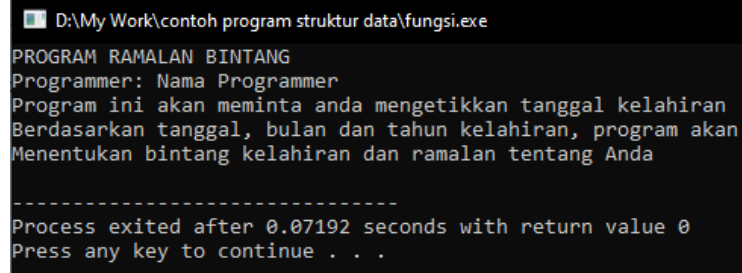
Salah satu bentuk implementasi dalam C dapat berupa:

```
void judul()  
{  
    printf("PROGRAM RAMALAN BINTANG\n");  
    printf("Programmer: Nama Programmer\n");  
    printf("Program ini akan meminta anda mengetikkan tanggal kelahiran\n");  
    printf("Berdasarkan tanggal, bulan dan tahun kelahiran, program akan\n");  
    printf("Menentukan bintang kelahiran dan ramalan tentang Anda\n");  
}
```

Sebagai contoh pemakaian fungsi tersebut, perhatikan program berikut :

```
1  #include <stdio.h>  
2  
3  //deklarasikan dulu fungsinya  
4      void judul();  
5  
6  main(){  
7      //panggil fungsi judul()  
8      judul();  
9  }  
10  
11 //definisi fungsi judul()  
12 void judul()  
13 {  
14     printf("PROGRAM RAMALAN BINTANG\n");  
15     printf("Programmer: Nama Programmer\n");  
16     printf("Program ini akan meminta anda mengetikkan tanggal kelahiran\n");  
17     printf("Berdasarkan tanggal, bulan dan tahun kelahiran, program akan\n");  
18     printf("Menentukan bintang kelahiran dan ramalan tentang Anda\n");  
19 }
```

Apabila program tersebut dijalankan, maka pada screen (layar) akan tampil output berupa:



```
D:\My Work\contoh program struktur data\fungsi.exe  
PROGRAM RAMALAN BINTANG  
Programmer: Nama Programmer  
Program ini akan meminta anda mengetikkan tanggal kelahiran  
Berdasarkan tanggal, bulan dan tahun kelahiran, program akan  
Menentukan bintang kelahiran dan ramalan tentang Anda  
-----  
Process exited after 0.07192 seconds with return value 0  
Press any key to continue . . .
```

2. Contoh 2: Buatlah suatu fungsi yang bertugas membaca 3 buah input berupa nilai uts, nilai uas dan nilai tugas. Berdasarkan tiga data tersebut fungsi akan menghitung nilai akhir dengan mengacu pada aturan, bahwa nilai akhir diperoleh dari 30% uts ditambah 30% uas ditambah 40% tugas.

Jawab:

Fungsi yang akan dibuat memiliki Langkah-langkah sebagai berikut:

- Membaca 3 buah input berupa nilai uts, nilai uas dan nilai tugas
- Menghitung nilai akhir dengan formula:
  - o  $\text{Nilai akhir} \leftarrow 30\% * \text{nilai uts} + 30\% * \text{nilai uas} + 40\% * \text{nilai tugas}$
- Menampilkan informasi nilai akhir ke layar

Algoritma fungsi dapat berupa:

```
procedure hitungNilaiAkhir()
deklarasi
variable:
    uts, uas, tugas, nilaiAkhir: real
deskripsi
start
    write('Ketikkan nilai UTS, UAS, dan TUGAS')
    write('Pisahkan setiap nilai dengan spasi: ')
    read(mid, uas, tugas)

    nilaiAkhir <- 0.30 * uts + 0.30 * uas + 0.40 * tugas
    write('Nilai Akhir: ', nilaiAkhir)
end
```

Implementasi fungsi dalam C dapat berupa sebagai berikut:

```
void hitungNilaiAkhir()
{
    double uts, uas, tugas, nilaiAkhir;

    printf("Ketikkan nilai UTS, UAS, dan TUGAS \n ");
    printf("Pisahkan setiap nilai dengan spasi: ");
    scanf("%lf %lf %lf", &mid, &uas, &tugas);

    nilaiAkhir = 0.30 * uts + 0.30 * uas + 0.40 * tugas;
    printf("Nilai Akhir: %0.1lf\n", nilaiAkhir);
}
```

```

1  #include <stdio.h>
2
3  void hitungNilaiAkhir(); //ini merupakan deklarasi fungsi
4
5  main()
6  {
7      //memanggil fungsi hitungNilaiAkhir()
8      hitungNilaiAkhir();
9  }
10
11 //berikut adalah definisi dari fungsi hitungNilaiAkhir()
12
13 void hitungNilaiAkhir()
14 {
15     double uts, uas, tugas, nilaiAkhir;
16
17     printf("Ketikkan nilai UTS, UAS, dan TUGAS \n ");
18     printf("Pisahkan setiap nilai dengan spasi: ");
19     scanf("%lf %lf %lf", &uts, &uas, &tugas);
20     public int __cdecl scanf (const char * __restrict __Format, ...)
21     nilaiAkhir = 0.30 * uts + 0.30 * uas + 0.40 * tugas;
22     printf("Nilai Akhir: %0.1lf\n", nilaiAkhir);
23 }

```

Coba jalankan program tersebut dan amati hasilnya.

```

D:\My Work\contoh program struktur data\fungsi.exe
Ketikkan nilai UTS, UAS, dan TUGAS
Pisahkan setiap nilai dengan spasi: 90 87 56
Nilai Akhir: 75.5

-----
Process exited after 6.966 seconds with return value 18
Press any key to continue . . .

```

- Contoh 3: Buatlah fungsi yang dapat mengkonversi nilai akhir menjadi nilai huruf dengan ketentuan sebagai berikut :

<i>Rentang nilai</i>	<i>Nilai Huruf</i>
$81.0 \leq \text{nilaiAkhir} \leq 100.0$	<i>A</i>
$66.0 \leq \text{nilaiAkhir} < 81.0$	<i>B</i>
$56.0 \leq \text{nilaiAkhir} < 66.0$	<i>C</i>
$45.0 \leq \text{nilaiAkhir} < 56.0$	<i>D</i>
$\text{nilaiAkhir} < 45.0$	<i>E</i>

Untuk Menyusun fungsi yang bertugas mengkonversi nilai akhir menjadi nilai huruf

dapat memiliki Langkah-langkah sebagai berikut:

- Membaca satu buah input berupa nilai akhir
- Mengkonversi nilai akhir menjadi nilai huruf dengan aturan sebagai berikut:

*Jika  $81.0 \leq \text{nilaiAkhir} \leq 100.0$  maka nilai huruf adalah A*

*Jika  $66.0 \leq \text{nilaiAkhir} < 81.0$  maka nilai huruf adalah B*

*Jika  $56.0 \leq \text{nilaiAkhir} < 66.0$  maka nilai huruf adalah C*

*Jika  $45.0 \leq \text{nilaiAkhir} < 56.0$  maka nilai huruf adalah D*

*Jika  $\text{nilaiAkhir} < 45.0$  maka nilai huruf adalah E*

- Menampilkan informasi nilai huruf ke layar

Algoritma fungsi dapat dibuat seperti berikut:

*procedure konversiNilai()*

*deklarasi*

*variable:*

*nilaiAkhir: real*

*nilaiHuruf: karakter*

*deskripsi*

*start*

*printf("Ketikkan nilai yang akan dikonversi: ");*

*scanf("%lf", &nilaiAkhir);*

*if((nilaiAkhir >= 81.0) && (nilaiAkhir <= 100.0))*

*nilaiHuruf = 'A';*

*else if(nilaiAkhir >= 66.0)*

*nilaiHuruf = 'B';*

*else if(nilaiAkhir >= 56.0)*

*nilaiHuruf = 'C';*

*else if(nilaiAkhir >= 45.0)*

*nilaiHuruf = 'D';*

*else nilaiHuruf = 'E';*

*printf("Nilai Akhir: %0.1lf Nilai Huruf: %c\n", nilaiAkhir, nilaiHuruf);*

*end*

Implementasi fungsi konversi nilai akhir menjadi nilai huruf dalam C dapat berupa:

*void konversiNilai()*

*{*

*double nilaiAkhir;*

*char nilaiHuruf;*

*printf("Ketikkan nilai yang akan dikonversi: ");*

```

scanf("%lf", &nilaiAkhir);

if((nilaiAkhir >= 81.0) && (nilaiAkhir <= 100.0))
    nilaiHuruf = 'A';
else if(nilaiAkhir >= 66.0)
    nilaiHuruf = 'B';
else if(nilaiAkhir >= 56.0)
    nilaiHuruf = 'C';
else if(nilaiAkhir >= 45.0)
    nilaiHuruf = 'D';
else nilaiHuruf = 'E';

printf("Nilai Akhir: %0.1lf Nilai Huruf: %c\n", nilaiAkhir, nilaiHuruf);
}

```

Fungsi tersebut dapat digunakan dalam suatu program seperti berikut :

```

1  #include <stdio.h>
2
3  void konversiNilai(); //deklarasi fungsi
4  main()
5  {
6      printf("Program ini akan mengkonversi nilai angka ke nilai huruf\n");
7      konversiNilai(); //memanggil fungsi konversiNilai()
8  }
9
10 //definisi fungsi
11 void konversiNilai()
12 {
13     double nilaiAkhir;
14     char nilaiHuruf;
15
16     printf("Ketikkan nilai yang akan dikonversi: ");
17     scanf("%lf", &nilaiAkhir);
18
19     if((nilaiAkhir >= 81.0) && (nilaiAkhir <= 100.0))
20         nilaiHuruf = 'A';
21     else if(nilaiAkhir >= 66.0)
22         nilaiHuruf = 'B';
23     else if(nilaiAkhir >= 56.0)
24         nilaiHuruf = 'C';
25     else if(nilaiAkhir >= 45.0)
26         nilaiHuruf = 'D';
27     else nilaiHuruf = 'E';
28
29     printf("Nilai Akhir: %0.1lf Nilai Huruf: %c\n", nilaiAkhir, nilaiHuruf);
30 }

```

```
D:\My Work\contoh program struktur data\fungsi.exe
Program ini akan mengkonversi nilai angka ke nilai huruf
Ketikkan nilai yang akan dikonversi: 76.3
Nilai Akhir: 76.3  Nilai Huruf: B

-----
Process exited after 5.095 seconds with return value 34
Press any key to continue . . .
```

4. Contoh 4: Buatlah program yang menggunakan fungsi membaca data dan memasukkannya ke variable array serta fungsi untuk mendisplaykan isi array le layar.

Untuk menyelesaikan masalah di atas, kita perlu menguraikan langkah-langkah penyelesaian dan melakukan analisis kebutuhan.

Langkah-langkah yang dilakukan:

- Siapkan variabel array bersifat global yang dapat menyimpan data, misalnya, 10 buah data bertipe integer
- Siapkan fungsi untuk membaca data array
- Siapkan fungsi untuk menuliskan data array
- Dari fungsi utama, atau main(), panggil fungsi membaca data
- Dari fungsi utama, atau main(), panggil fungsi menampilkan isi array

Kebutuhan Konstanta

- Konstanta untuk menetapkan banyaknya data, beri nama  $N$

Kebutuhan Variabel

- Variabel untuk menyimpan sementara data yang dibaca, beri nama *datanya* bertipe int, bersifat lokal pada fungsi
- Variabel untuk konter perulangan (banyaknya membaca dan menuliskan data), beri nama  $i$ , bertipe int. Variabel ini bersifat lokal pada masing-masing fungsi.

Kebutuhan fungsi

- Fungsi untuk membaca data, beri nama *bacaData()*

Langkah-langkah pada fungsi *bacaData()*

- o Siapkan variabel konter bernama  $i$  dan bertipe int
- o Siapkan variabel untuk menyimpan data yang dibaca bernama *datanya* dan bertipe int
- o Mulai data pertama, lakukan
  - baca data simpan pada *datanya*



- simpan datanya pada array data[]
- ulangi hingga data terakhir

**Algoritma fungsi bacaData() dapat berbentuk:**

*Procedure bacaData()*

*Deklarasi*

*Variable:*

*I, datanya : integer*

*Deskripsi*

*Start*

*Write('Membaca data: ')*

*For I <- 0 to N step 1 do*

*Write('Ketikan data: ')*

*Read(datanya)*

*Data[i] = datanya*

*endFor*

*end*

Implementasi fungsi dalam C :

**void bacaData()**

**{**

**int i;**

**int datanya;**

**printf("Membaca data:\n");**

**for(i=0;i<N;i++){**

**printf("Data ke-%d: ", i+1);**

**scanf("%d", &datanya); fflush(stdin);**

**//masukkan datanya ke array**

**data[i] = datanya;**

**}**

**printf("Entri data selesai...\n");**

```

    getch();
}

```

- Fungsi untuk menuliskan data, beri nama tulisInfo()

Langkah-langkah pada fungsi tulisInfo()

- Siapkan variabel konter bernama *I* dan bertipe *int*
- Siapkan variabel untuk menyalin isi array bernama *datanya* dan bertipe *int*
- Mulai elemen ke 1 array, lakukan
  - salin data pada element *data[I]* ke *datanya*
  - tuliskan isi *datanya* ke layar
  - ulangi hingga elemen terakhir

**Algoritma fungsi tulisInfo() dapat berbentuk:**

**Procedure tulisInfo()**

**Deklarasi**

**Variable:**

*I, datanya : integer*

**Deskripsi**

**Start**

*Write('Menampilkan isi array ke layar: ')*

*For I <- 0 to N step 1 do*

*Datanya = data[i]*

*Write('Data ke-', i+1), datanya)*

*endFor*

*write('Akhir dari data array')*

**end**

Implementasi fungsi dalam C:

```

void tulisInfo()

```

```

{

```

```

    int I;

```

```

    int datanya;

```

```

    printf("Menampilkan isi array ke layar:\n");

```

```

    for(i=0;i<N;i++){

```

```

        datanya = data[i];

```

```

        Printf("data ke-%d: %d\n", i+1, datanya);
    }
}

```

```

    }
    printf("\nAkhir dari data array...\n");
    getch();
}

```

Sekarang kita buat program lengkapnya sebagai berikut:

```

/*
    contoh fungsi tanpa parameter tanpa nilai balik untuk membaca data untuk
    mengisi array dan fungsi untuk menuliskan isi array ke layar.
    Nama File: isidata.c
*/
#include <stdio.h>
#include <stdlib.h>
//konstanta
#define N 10 //menetapkan banyaknya data sekaligus ukuran array

//deklarasi fungsi untuk mengisi data array dan menuliskan isi array
void bacaData();
void tulisInfo();

//deklarasi variabel global
int data[N];
main()
{
    Printf("Program untuk mendemonstrasikan pembacaan data dan\n");
    Printf("Mengisikannya ke array data, lalu menuliskan isi array tersebut ke
    layar\n");
    //panggil fungsi bacaData()
    bacaData();
    //panggil fungsi tulisInfo()
    tulisInfo();
    printf("Selesai... Tekan tombol Enter!");
    getch();
}
//definisi fungsi
void bacaData()
{
    int I;
    int datanya;

    printf("Membaca data:\n");
    for(i=0;i<N;i++){

```

```

        printf("Data ke-%d: ", I+1);
        scanf("%d", &datanya); fflush(stdin);
        //masukkan datanya ke array
        data[I] = datanya;
    }
    printf("Entri data selesai...\n");
    getch();
}

void tulisInfo()
{
    int I;
    int datanya;

    printf("Menuliskan isi array ke layar:\n");
    for(i=0;i<N;i++){
        datanya = data[I];
        Printf("data ke-%d: %d\n", i+1, datanya);
    }
    printf("\nAkhir dari data array...\n");
    getch();
}

```

## 2. Fungsi dengan nilai balik tanpa passing parameter

Bentuk umum penulisan algoritma untuk fungsi yang memiliki nilai balik namun tidak mempunyai passing parameter berupa:

**Function** *namaFungsi()*: *tipe nilaiBalik*

**Deklarasi**

**Constanta:**

*//daftar konstanta jika ada*

**Variable:**

*//daftar variable jika ada*

**Start**

*Pernyataan\_aksi-aksi*

*namaFungsi <- nilaiBalik*

**end**

Fungsi seperti ini dalam C secara umum adalah seperti berikut:

```

Nilai_balik nama_fungsi()
{
    //badan fungsi
    return nilai_balik;
}

```

Sebagai contoh, misalnya kita membuat fungsi untuk menghitung nilai faktorial sebuah bilangan integer, dimana hasil berupa nilai faktorialnya dijadikan sebagai nilai balik dari fungsi tersebut. Perhatikan cara menuliskan algoritma fungsi dimaksud sebagai berikut:

*Function factorial(): integer*

*deklarasi*

*variable*

*i, fak, n: integer*

*deskripsi*

*start*

*write('Ketikkan nilai n: ')*

*read(n)*

*fak <- 1*

*for i<-1 to n step 1 do*

*fak = fak \* i;*

*endFor*

*factorial <- fak*

*end*

Implementasi algoritma tersebut dalam C adalah sebagai berikut:

```

int factorial()

```

```

{

```

```

    int i, fak, n;

```

```

    printf("Ketikkan nilai n: ");

```

```

    scanf("%d", &n);

```

```

    fak = 1;
    for(i=1;i<=n;i++)
    {
        fak = fak * i;
    }
    return fak;
}

```

Contoh penggunaan dari fungsi di atas dapat berupa:

```

#include <stdio.h>

int factorial();

main()
{
    int fakt;

    printf("Menghitung nilai faktorial suatu bilangan\n ");
    fakt = factorial();
    printf("Nilai faktorial = %d\n", fakt);
}

Int factorial()
{
    int i, fak, n;

    printf("Ketikkan nilai n: ");
    scanf("%d", &n);
    fak = 1;
    for(i=1;i<=n;i++)
    {
        fak = fak * i;
    }
    return fak;
}

```

Atau bisa juga variabel yang digunakan berupa variabel global, perhatikan program berikut:

```

#include <stdio.h>

```

```

//daftar fungsi
int factorial();

int i, fak, n;//global variable

main()
{

    printf("Ketikkan nilai n: ");
    scanf("%d", &n);

    fak = faktorial();
    printf("\nNilai faktorial %d = %d\n", n, fak);
}

int faktorial()
{
    fak = 1;
    for(i=1;i<=n;i++)
    {
        fak = fak * i;
    }
    return fak;
}

```

### 3. Fungsi tanpa nilai balik dengan passing parameter

Bentuk lain fungsi berupa fungsi yang tidak memiliki nilai balik dan memiliki passing parameter, dalam algoritma disebut prosedur, dan bentuk penulisan algoritma seperti berikut :

*procedure nama\_prosedur(passing\_parameter)*

*deklarasi*

*Konstanta:*

*//daftar konstanta jika ada*

*Variable:*

*//daftar variable jika ada deskripsi*

*start*

*//badan fungsi*

*end*

Penulisan fungsi yang tidak memiliki nilai balik dan memiliki passing parameter

dalam C berbentuk:

```
void nama_fungsi(passing_parameter)  
{  
//badan fungsi  
}
```

Sebagai contoh, kita ubah fungsi untuk menghitung nilai faktorial sebelumnya menjadi fungsi faktorial() yang memiliki passing parameter tapi fungsi tersebut tidak memiliki nilai balik, penulisan algoritmanya dapat seperti berikut:

*procedure faktorial(n: integer)*

*deklarasi*

*variable:*

*i, fak: integer*

*deskripsi*

*start*

*fak <- 1*

*for I <- 1 to n step 1 do*

*fak <- fak \* i*

*endFor*

*write('Nilai Faktorial dari ', n, '= ', fak)*

*end*

implementasi dalam C dapat berbentuk seperti berikut:

*void faktorial(int n)*

*{*

*int i, fak;*

*fak = 1;*

*for(i=1;i<=n;i++)*

*{*

*fak = fak \* i;*

*}*



```
    printf("Nilai Faktorial dari %d! = %d\n",n, fak);  
}
```

Cara memanggil fungsi faktorial() adalah sbb:

```
faktorial(5);
```

Sebagai contoh penggunaan fungsi faktorial() yang memiliki passing parameter tersebut, perhatikan program berikut ini:

```
#include <stdio.h>  
void faktorial();  
  
main()  
{  
    int n;  
  
    printf("Menghitung nilai faktorial suatu bilangan\n ");  
    printf("Ketikkan sebuah bilangan integer: ");  
    scanf("%d", &n);  
    faktorial(n); //panggil fungsinya dan kirim n sebagai parameter fungsi  
}  
  
void faktorial(int n)  
{  
    int i, fak;  
  
    fak = 1;  
  
    for(i=1; i<=n; i++)  
    {  
        fak = fak * i;  
    }  
    printf("Nilai Faktorial dari %d! = %d\n",n, fak);  
}
```

Berikut ini adalah contoh lain tentang fungsi. Cobalah Anda perhatikan dan pelajari, apa yang dilakukan program tersebut?

```
#include <stdio.h>

void deret(int p, int n);

main()
{
    int a, b;

    printf("Ketikkan angka kelipatan: ");
    scanf("%d", &a);
    printf("Ketikkan banyaknya suku : ");
    scanf("%d", &b);

    deret(a, b);
}

void deret(int p, int n)
{
    int i, hasil;

    hasil = 0;
    for(i=1; i<=n;++i)
    {
        hasil = p * i;
        printf("%d ", hasil);
    }
}
```

#### 4. Fungsi dengan nilai balik dan passing parameter

Bentuk lain lagi dari fungsi adalah fungsi yang memiliki nilai balik dan juga memiliki passing parameter. Perhatikan bagaimana bentuk algoritma dari fungsi tersebut sebagai berikut:

*Function nama\_fungsi(parameter): tipe\_nilai\_balik*

*Deklarasi*

*Konstanta:*

*//daftar konstanta*

*Variable:*

*//daftar variabel*

*Deskripsi*

*Start*

*//badan fungsi*

*Nama\_fungsi <- nilai\_balik*

*end*

Bentuk definisi fungsi dalam C adalah seperti berikut

**Tipe\_Nilai\_balik nama\_fungsi(parameter)**

```
{  
    //badan fungsi  
    return nilai_balik;  
}
```

Misalnya, kita akan merubah bentuk fungsi factorial() sebelumnya ke dalam bentuk fungsi yang memiliki nilai balik dan juga memiliki passing parameter. Penulisan algoritma dapat berbentuk seperti berikut:

*function factorial(n: integer): integer*

*deklarasi*

*variable:*

*i, fak: integer*

*deskripsi*

*start*

```

    fak <- 1
    for I <- 1 to n step 1 do
        fak <- fak * i
    endFor
    factorial <- fak
end

```

Definisi fungsi tersebut dalam C adalah sebagai berikut:

```

int factorial(int n)
{
    int i, fak;

    fak = 1;
    for(i=1; i<=n; i++)
    {
        fak = fak * i;
    }
    return fak;
}

```

Contoh pemakaian fungsi tersebut dalam program dapat berupa:

```

#include <stdio.h>

int factorial(int n);

main()
{
    int n, fak;

    printf("Ketikkan sebuah bilangan: ");
    scanf("%d", &n);

    fak = factorial(n);
    printf("Nilai faktorial dari %d = %d\n", n, fak);
}

int factorial(int n)

```

```

{
    int i, fak;

    fak = 1;
    for(i=1; i<=n; i++)
    {
        fak = fak * i;
    }
    return fak;
}

```

Contoh lainnya, misal kita ingin mengkonversikan suatu nilai angka menjadi nilai huruf.

Tentu saja kita harus menentukan kriteria konversi, misal:

$80.0 \leq \text{nilai angka} \leq 100.0$	<i>maka nilai hurufnya adalah A</i>
$70.0 \leq \text{nilai angka} < 80.0$	<i>maka nilai hurufnya adalah B</i>
$60.0 \leq \text{nilai angka} < 70.0$	<i>maka nilai hurufnya adalah C</i>
$50.0 \leq \text{nilai angka} < 60.0$	<i>maka nilai hurufnya adalah D</i>
$\text{nilai angka} < 50.0$	<i>maka nilai hurufnya adalah E</i>

Untuk menyelesaikan kasus ini, analisis kebutuhan yang dilakukan meliputi:

- Kebutuhan output atau nilai balik fungsi  
fungsi dapat menghasilkan nilai balik berupa nilai huruf dengan tipe karakter.
- Kebutuhan passing parameter fungsi  
Passing parameter yang dibutuhkan berupa nilai akhir bertipe real dan diberi nama nilaiAkhir.
- Kebutuhan proses  
Proses untuk mengkonversi nilai akhir mengacu pada ketentuan yang diuraikan pada soal terkait kriteria penilaian, dimana nilai akhir dievaluasi dalam rentang nilai tertentu yang menentukan nilai huruf.

Algoritma fungsi :

Berikut adalah algoritma untuk fungsi konversi nilai akhir menjadi nilai huruf:

***Function konversiNilai(nilaiAkhir: real)***

***Deklarasi***

*Variable:*

*nilaiHuruf: karakter*

*deskripsi*

*start*

*if 80.0 <= nilaiAkhir <= 100.0*

*then nilaiHuruf <- 'A'*

*else if 80.0 <= nilaiAkhir <= 80.0*

*then nilaiHuruf <- 'B'*

*else if 60.0 <= nilaiAkhir < 70.0*

*then nilaiHuruf <- 'C'*

*else if 45.0 <= nilaiAkhir < 60.0*

*then nilaiHuruf <- 'D'*

*else nilaiHuruf <- 'E'*

*nedIf*

*konversiNilai <- nilaiHuruf*

*end*

implementasi dalam C, fungsi konversiNilai() dapat berbentuk seperti berikut:

*char konversNilai(double nilaiAkhir)*

*{*

*char nilaiHuruf;*

*if((nilaiAkhir >= 80.0) && (nilaiAkhir <= 100.0))*

*nilaiHuruf = 'A';*

*else if((nilaiAkhir >= 70.0) && (nilaiAkhir < 80.0))*

*nilaiHuruf = 'B';*

*else if((nilaiAkhir >= 60.0) && (nilaiAkhir < 70.0))*

*nilaiHuruf = 'C';*

*else if((nilaiAkhir >= 50.0) && (nilaiAkhir < 60.0))*

*nilaiHuruf = 'D';*

*else nilaiHuruf = 'E';*

*return nilaiHuruf;*

*}*

## 4. Forum Diskusi

```
1  #include <stdio.h>
2  #include <string.h>
3
4  char konversiNilai(double angka);
5
6  main()
7  {
8      double angka;
9      char huruf;
10
11     printf("Ketikkan suatu nilai: ");
12     scanf("%lf", &angka);
13     huruf = konversiNilai(angka);
14     printf("Nilai angka = %0.1lf dan nilai huruf = %c\n", angka, huruf);
15 }
16
17 char konversNilaii(double nilaiAkhir)
18 {
19     char nilaiHuruf;
20     if((nilaiAkhir >= 80.0) && (nilaiAkhir <= 100.0))
21         nilaiHuruf = 'A';
22     else if((nilaiAkhir >= 70.0) && (nilaiAkhir < 80.0))
23         nilaiHuruf = 'B';
24     else if((nilaiAkhir >= 60.0) && (nilaiAkhir < 70.0))
25         nilaiHuruf = 'C';
26     else if((nilaiAkhir >= 50.0) && (nilaiAkhir < 60.0))
27         nilaiHuruf = 'D';
28     else nilaiHuruf = 'E';
29
30     return nilaiHuruf;
31 }
```

Perhatikan, bahwa Ketika fungsi `konversiNilai()` didefinisikan, parameter dari fungsi berupa satu buah variable Bernama `nilaiAkhir` bertipe `double`, sedangkan pada saat pemanggilan fungsi tersebut, passing parameter (disebut juga `argument`) berupa variable Bernama `nilai`, yang kebetulan diperoleh dari input pada fungsi `main()`. Mengapa hal ini bisa dilakukan?

### Contoh Kasus Program Menghitung Segitiga

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

// Deklarasi fungsi-fungsi

float hitungLuas(float a, float b, float c);
float hitungKeliling(float a, float b, float c);
void tampilkanInformasiSegitiga(float a, float b, float c);
void inputs(float *sisi1, float *sisi2, float *sisi3);

int main() {

    float sisi1, sisi2, sisi3;

    inputs(&sisi1, &sisi2, &sisi3);
```

```

    tampilkanInformasiSegitiga(sisi1, sisi2, sisi3);

    return 0;
}

void inputs(float *sisi1, float *sisi2, float *sisi3){

    printf("Masukkan panjang sisi 1: ");
    scanf("%f", sisi1);
    printf("Masukkan panjang sisi 2: ");
    scanf("%f", sisi2);
    printf("Masukkan panjang sisi 3: ");
    scanf("%f", sisi3);
}

// Implementasi fungsi untuk menghitung luas segitiga
float hitungLuas(float a, float b, float c) {
    float s = (a + b + c) / 2;
    return sqrt(s * (s - a) * (s - b) * (s - c));
}

// Implementasi fungsi untuk menghitung keliling segitiga
float hitungKeliling(float a, float b, float c) {
    return a + b + c;
}

// Implementasi prosedur untuk menampilkan informasi segitiga
void tampilkanInformasiSegitiga(float a, float b, float c) {
    float luas = hitungLuas(a, b, c);
    float keliling = hitungKeliling(a, b, c);

    printf("\nInformasi Segitiga:\n");
    printf("Panjang Sisi 1: %.2f\n", a);
    printf("Panjang Sisi 2: %.2f\n", b);
    printf("Panjang Sisi 3: %.2f\n", c);
    printf("Luas: %.2f\n", luas);
    printf("Keliling: %.2f\n", keliling);

    if (a == b && b == c) {
        printf("Segitiga ini adalah segitiga sama sisi.\n");
    } else if (a == b || b == c || a == c) {
        printf("Segitiga ini adalah segitiga sama kaki.\n");
    } else {
        printf("Segitiga ini adalah segitiga sembarang.\n");
    }
}

```



```

Masukkan panjang sisi 1: 5
Masukkan panjang sisi 2: 5
Masukkan panjang sisi 3: 5

Informasi Segitiga:
Panjang Sisi 1: 5.00
Panjang Sisi 2: 5.00
Panjang Sisi 3: 5.00
Luas: 10.83
Keliling: 15.00
Segitiga ini adalah segitiga sama sisi.

-----
Process exited after 2.744 seconds with return value 0
Press any key to continue . . .

```

## C. Penutup

### 1. Rangkuman

No	Jenis Fungsi	Keterangan
1.	Fungsi tanpa parameter dan tanpa nilai balik	fungsi ini tidak memerlukan argumen apapun saat pemanggilannya serta tidak memberikan nilai balik kepada pemanggil fungsi tersebut.
2.	Fungsi memiliki parameter dan tanpa nilai balik	Fungsi jenis ini membutuhkan argumen atau parameter saat pemanggilannya dan tidak memberikan nilai balik kepada pemanggil fungsi tersebut
3.	Fungsi tidak memiliki parameter dan memberikan nilai balik	Fungsi ini tidak memerlukan argumen apapu saat pemanggilannya dan setelah mengerjakan tugasnya, fungsi ini memberikan suatu nioai balik
4.	Fungsi memiliki parameter dan memberikan nilai balik	Fungsi jenis ini membutuhkan argumen atau parameter ketika dipanggil lalu setelah mengerjakan tugasnya, fungsi tersebut memberikan nilai balik kepada pemanggilnya

## Tugas

1. Buatlah program modular untuk menampilkan daftar kode morse
2. Buatlah program modular untuk menampilkan bilangan prima Batas

## Daftar Pustaka

### Utama:

- ✓ Rinaldi Munir. 2016. *Algoritma dan Pemrograman*. Bandung. Informatika ITB
- ✓ Noel Kalicharan. 2015. *Learn to Program with C*. New York, Springer-Science
- ✓ Harry H. Chaudhary. 2014. *C Programming Step by Step*. LLC USA. Amazon Inc.

### Pendukung:

- ✓ Mike McGrath. 2015. *Coding for Beginners*. Leamington Spa. Easy Step Limited.
- ✓ Dan Gookin. 2014. *Beginning Programming with C for Dummies*. New Jersey. John Wiley & Sons.
- ✓ [www.tutorialspoint.com](http://www.tutorialspoint.com)
- ✓ [www.javatpoint.com](http://www.javatpoint.com)
- ✓ [www.programiz.com](http://www.programiz.com)