SD

2020

Modul Praktikum

# Pemrograman Dasar

Nama :

Nim :

PROGRAM STUDI SARJANA SAINS DATA

Universitas Teknologi Yogyakarta

# MODUL PRAKTIKUM
# PEMROGRAMAN DASAR

Disusun Oleh :
Agus Suhendar
Farida Ardiani

Universitas Teknologi Yogyakarta
2020

## KATA PENGANTAR

Dengan memanjatkan Puji dan syukur kehadirat Tuhan Yang Maha Esa, atas keridhoannya penulis mampu menyelesaikan penyusunan Modul Praktikum Pemograman Dasar ini. Adapun modul praktikum ini bertujuan sebagai bahan pelaksanaan praktikum bagi mahasiswa Program Studi Sarjana Sains Data Universitas Teknologi Yogyakarta dan sebagai bahan belajar mandiri bagi mahasiswa.

Tersusunnya buku ini tentu bukan dari usaha penulis seorang. Dukungan moral dan material dari berbagai pihak sangatlah membantu tersusunnya modul ini. Untuk itu, penulis ucapkan terima kasih kepada keluarga, sahabat, rekan-rekan, dan pihak-pihak lainnya yang membantu secara moral dan material bagi tersusunnya modul ini. Harapan kami semoga modul praktikum ini dapat menambah pengetahuan dan pengalaman bagi para pembaca dan praktikan laboratorium komputer Universitas Teknologi Yogyakarta dan mampu memperdalam pemahaman pembaca dan praktikan mengenai materi praktikum yang dibahas.

Modul yang tersusun ini tentu masih jauh dari kata sempurna. Untuk itu, kritik dan saran yang membangun sangat diperlukan agar buku ini bisa lebih baik nantinya. Kritik dan saran bisa dikirim ke alamat e-mail penulis agus.suhendar@staff.uty.ac.id dan farida.ardiani@staff.uty.ac.id.

Yogyakarta, Juni 2020

Penulis

## DAFTAR ISI

**KATA PENGANTAR** ..... **3**

**DAFTAR ISI** ..... **4**

**BAB I DASAR PEMROGRAMAN BAHASA C** ..... **8**

**1.1. Capaian Pembelajaran** ..... **8**

**1.2. Indikator Keberasilan** ..... **8**

**1.3. Uraian Materi** ..... **8**

**1.4. Latihan** ..... **12**

**1.5. Tugas** ..... **12**

**BAB II OPERATOR** ..... **14**

**2.1. Capaian Pembelajaran** ..... **14**

**2.2. Indikator Keberasilan** ..... **14**

**2.3. Uraian Materi** ..... **14**

**2.4. Latihan** ..... **22**

**2.5. Tugas** ..... **23**

**BAB III PERCABANGAN (SELECTION) DENGAN IF** ..... **24**

**3.1. Capaian Pembelajaran** ..... **24**

**3.2. Indikator** ..... **24**

**3.3. Uraian Materi** ..... **24**

**3.4. Latihan** ..... **31**

**3.5. Tugas** ..... **33**

**BAB IV PERCABANGAN (SELECTION) IF BERTINGKAT** ..... **34**

**4.1. Capaian Pembelajaran** ..... **34**

**4.2. Indikator** ..... **34**

**4.3. Uraian Materi** ..... **34**

**4.4. Latihan** ..... **37**

**4.5. Tugas ..... 38**

**BAB V PERCABANGAN (SELECTION) DENGAN SWITCH ..... 39**

**5.1. Capaian Pembelajaran ..... 39**

**5.2. Indikator ..... 39**

**5.3. Uraian Materi ..... 39**

**5.4. Latihan ..... 41**

**5.5. Tugas ..... 43**

**BAB VI PERULANGAN (LOOPING) DENGAN FOR ..... 44**

**6.1. Capaian Pembelajaran ..... 44**

**6.2. Indikator ..... 44**

**6.3. Uraian Materi ..... 44**

**6.4. Latihan ..... 46**

**6.5. Tugas ..... 47**

**BAB VII PERULANGAN (LOOPING) DENGAN WHILE ..... 48**

**7.1. Capaian Pembelajaran ..... 48**

**7.2. Indikator ..... 48**

**7.3. Uraian Materi ..... 48**

**7.4. Latihan ..... 53**

**7.5. Tugas ..... 54**

**BAB VIII FOR BERSARANG (NESTED FOR ) ..... 55**

**8.1. Capaian Pembelajaran ..... 55**

**8.2. Indikator ..... 55**

**8.3. Uraian Materi ..... 55**

**8.4. Latihan ..... 55**

**8.5. Tugas ..... 56**

**BAB IX KOMBINASI SELEKSI DAN PERULANGAN ..... 57**

**9.1. Capaian Pembelajaran ..... 57**

**9.2. Indikator ..... 57**

**9.3. Uraian Materi ..... 57**

**9.4. Latihan ..... 61**

**9.5. Tugas ..... 62**

**BAB X ARRAY SATU DIMENSI ..... 63**

**10.1. Capaian Pembelajaran ..... 63**

**10.2. Indikator ..... 63**

**10.3. Uraian Materi ..... 63**

**10.4. Latihan ..... 65**

**10.5. Tugas ..... 66**

**BAB XI ARRAY DUA DIMENSI ..... 68**

**11.1. Capaian Pembelajaran ..... 68**

**11.2. Indikator ..... 68**

**11.3. Uraian Materi ..... 68**

**11.4. Latihan ..... 70**

**11.5. Tugas ..... 72**

**BAB XII FUNGSI DENGAN DAN TANPA NILAI BALIK ..... 73**

**12.1. Capaian Pembelajaran ..... 73**

**12.2. Indikator ..... 73**

**12.3. Uraian Materi 1 – Dengan Nilai Balik ..... 73**

**12.4. Latihan ..... 75**

**12.5. Tugas ..... 76**

**12.6. Uraian Materi 2 – Tanpa Nilai Balik..... 77**

**12.7. Latihan ..... 78**

**12.8. Tugas ..... 79**

**BAB XIII FUNGSI DENGAN PARAMETER..... 80**

**13.1. Capaian Pembelajaran ..... 80**

**13.2. Indikator ..... 80**

**13.3. Uraian Materi ..... 80**

**13.4. Latihan ..... 81**

**13.5. Tugas ..... 82**

**MODUL XIV CHALLENGES ..... 83**

**14.1. Capaian Pembelajaran ..... 83**

**14.2. Indikator ..... 83**

**14.3. Latihan ..... 83**

**RESPONSI - 1 ..... I**

**RESPONSI - 2 ..... II**

# BAB I
# DASAR PEMROGRAMAN BAHASA C

## 1.1. Capaian Pembelajaran

- Mengetahui dasar-dasar pemrograman meliputi struktur dasar bahasa pemrograman.
- Mengetahui penulisan komentar, pengenal, konstanta, dan variabel pada bahasa pemrograman.
- Mengetahui perintah-perintah dasar untuk menampilkan tampilan pada monitor pada bahasa pemrograman.
- Mampu mengimplementasikan cara penulisan di dalam program menggunakan Bahasa C, serta menerapkan perhitungan rumus matematika ke dalam bahasa pemrograman.

## 1.2. Indikator Keberasilan

- Mampu menjelaskan dasar-dasar bahasa pemrograman.
- Mampu menuliskan komentar, pengenal, konstanta, dan variabel pada suatu program.
- Mampu membuat program sederhana untuk menampilkan tulisan pada monitor

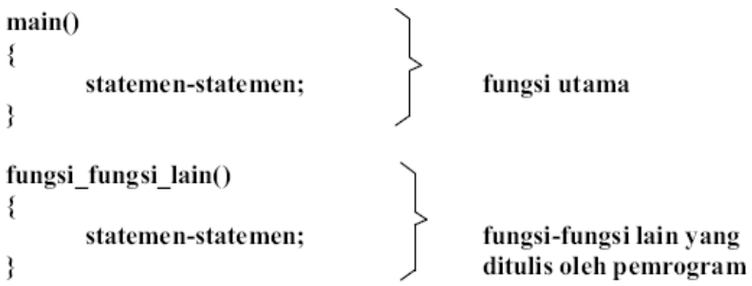
## 1.3. Uraian Materi

### A. Struktur Penulisan Program C

Untuk dapat memahami bagaimana suatu program ditulis, maka struktur dari program harus dimengerti terlebih dahulu. Tiap bahasa komputer mempunyai struktur program yang berbeda. Struktur program memberikan gambaran secara luas, bagaimana bentuk program secara umum.

Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Fungsi pertama yang harus ada dalam program C dan sudah ditentukan namanya adalah main(). Setiap fungsi terdiri atas satu atau beberapa pernyataan, yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus. Bagian pernyataan fungsi (sering disebut tubuh fungsi) diawali dengan tanda kurung kurawal buka ({) dan diakhiri dengan tanda kurung kurawal tutup (}). Diantara kurung itu dapat dituliskan statemen-statemen program C. Namun pada kenyataannya, suatu

fungsi bisa saja tidak mengandung pernyataan sama sekali. Walaupun fungsi tidak memiliki pernyataan, kurung kurawal haruslah tetap ada. Sebab kurung kurawal mengisyaratkan awal dan akhir definisi fungsi. Berikut ini adalah struktur dari program C.




Bahasa C dikatakan sebagai bahasa pemrograman terstruktur karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagiannya (*subroutine*). Fungsi-fungsi yang ada selain fungsi utama (*main()*) merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi utama atau diletakkan di file pustaka (*library*). Jika fungsi-fungsi diletakkan di file pustaka dan akan dipakai di suatu program, maka nama file judulnya (*header file*) harus dilibatkan dalam program yang menggunakannya dengan *preprocessor directive* berupa *#include*.

## B. Pengenalan Fungsi-Fungsi Dasar Bahasa C

### a. Fungsi *main()*

Pada program C, *main()* merupakan fungsi yang istimewa. Fungsi *main()* harus ada pada program, sebab fungsi inilah yang menjadi titik awal dan titik akhir Eksekusi program. Tanda { di awal fungsi menyatakan awal tubuh fungsi dan sekaligus awal Eksekusi program, sedangkan tanda } di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus adalah akhir Eksekusi program. Jika program terdiri atas lebih dari satu fungsi, fungsi *main()* biasa ditempatkan pada posisi yang paling atas dalam pendefinisian fungsi. Hal ini hanya merupakan kebiasaan. Tujuannya untuk memudahkan pencarian terhadap program utama bagi pemrogram. Jadi bukanlah merupakan suatu keharusan.

### b. Fungsi *printf()*

Fungsi *printf()* merupakan fungsi yang umum dipakai untuk menampilkan suatu keluaran pada layar peraga. Untuk menampilkan tulisan

Selamat belajar bahasa C

misalnya, pernyataan yang diperlukan berupa:

```
printf("Selamat belajar bahasa C");
```

Pernyataan di atas berupa pemanggilan fungsi *printf()* dengan argumen atau parameter berupa string. Dalam C suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik-ganda (“). Perlu juga diketahui pernyataan dalam C selalu diakhiri dengan tanda titik koma (;). Tanda titik koma dipakai sebagai tanda pemberhentian sebuah pernyataan dan bukanlah sebagai pemisah antara dua pernyataan.

Tanda \ pada string yang dilewatkan sebagai argumen *printf()* mempunyai makna yang khusus. Tanda ini bisa digunakan untuk menyatakan karakter khusus seperti karakter baris-baru ataupun karakter *backslash* (miring kiri). Jadi karakter seperti *n* sebenarnya menyatakan sebuah karakter. Contoh karakter yang ditulis dengan diawali tanda \ adalah:

| | |
|---|---|
| \” | menyatakan karakter petik-ganda |
| \\ | menyatakan karakter backslash |
| \t | menyatakan karakter tab |

Dalam bentuk yang lebih umum, format printf()

```
printf("string kontrol", daftar argumen);
```

dengan string kontrol dapat berupa satu atau sejumlah karakter yang akan ditampilkan ataupun berupa penentu format yang akan mengatur penampilan dari argumen yang terletak pada daftar argumen. Mengenai penentu format di antaranya berupa:

| | |
|---|---|
| %d | untuk menampilkan bilangan bulat (integer) |
| %f | untuk menampilkan bilangan titik-mengambang (pecahan) |
| %c | untuk menampilkan sebuah karakter |
| %s | untuk menampilkan sebuah string |

**Contoh:**

```
#include <stdio.h>
main( )
{
printf("No      : %d\n", 10);
printf("Nama : %s\n", "Ali");
```

```
printf("Nilai: %f\n",80.5);
printf("Huruf   : %c\n",'A');
}
```

**c. Pengenalan Praprosesor #include**

*#include* merupakan salah satu jenis pengarah praprosesor (*preprocessor directive*). Pengarah praprosesor ini dipakai untuk membaca file yang di antaranya berisi deklarasi fungsi dan definisi konstanta. Beberapa file judul disediakan dalam C. File-file ini mempunyai ciri yaitu namanya diakhiri dengan ekstensi **.h**. Misalnya pada program *#include <stdio.h>* menyatakan pada kompiler agar membaca file bernama *stdio.h* saat pelaksanaan kompilasi.

Bentuk umum #include:

```
#include "namafile"
```

Bentuk pertama (*#include* <namafile>) mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file include. Sedangkan bentuk kedua (*#include* "namafile") menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

Kebanyakan program melibatkan file *stdio.h* (file-judul I/O standard, yang disediakan dalam C). Program yang melibatkan file ini yaitu program yang menggunakan pustaka I/O (input-output) standar seperti *printf()*.

**d. Komentar dalam Program**

Untuk keperluan dokumentasi dengan maksud agar program mudah dipahami, biasanya pada program disertakan komentar atau keterangan mengenai program. Dalam C, suatu komentar ditulis dengan diawali dengan tanda /* dan diakhiri dengan tanda */.

**Contoh :**

```
/*Tanda ini adalah komentar
tidak masuk dalam Eksekusi program */
#include <stdio.h>
main()
{
printf("Coba\n");    //Ini adl program pertama
}
```

### 1.4.Latihan

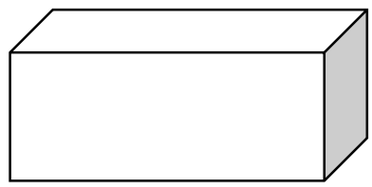
1. Tulislah program berikut ini :

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

2. Program Biodata

```
#include<stdio.h> /* preprosesor */
int Umur;
int main()
{
char Nama[10];
printf("Latihan Modul I: ");
scanf("%s",&Nama);
printf("Nama : %s \n",Nama);
scanf("%d",&Umur);
printf("Umur : %d \n",Umur);
}
```

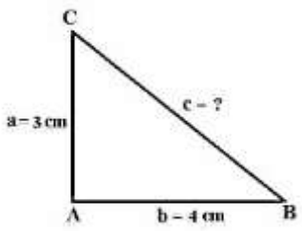
3. Program dibawah ini adalah untuk menghitung luas persegi panjang serta volume dari persegi panjang tersebut. Masukan dari program ini adalah panjang 9 cm, lebar 3 cm dan tinggi 3 cm. Untuk masukan nilai tersebut dibuat sebagai inputan dari keyboard.



### 1.5.Tugas

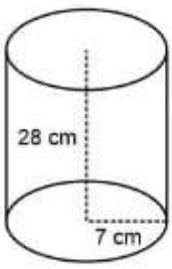
1. Buatlah sebuah program untuk menampilkan sebuah kalimat sebagai berikut :
   Universitas Teknologi Yogyakarta, HEBAT !!!
   Jurusan Teknik Informatika

2. Hitunglah sisi miring dari segitiga siku-siku berikut ini :




   Silahkan menggunakan teori theorema phytagoras!

3. Buatlah program untuk menghitung volume tabung berikut ini.




4. Sebuah bis berangkat dari kota A ke kota B dengan kecepatan rata-rata 50 km/jam dalam waktu 4 jam 30 menit. Berapa jarak antara kota A dengan kota B?

# BAB II
# OPERATOR

## 2.1. Capaian Pembelajaran

- Mengetahui tipe data dan operator pada C.
- Mengetahui perbedaan penggunaan tipe data dan operator.
- Mampu menerapkan perhitungan skala jarak dalam peta vs jarak sesungguhnya, serta penerapan rumus matematika ke dalam bahasa pemrograman.

## 2.2. Indikator Keberasilan

- Mampu menjelaskan tipe data dan operator.
- Mampu menggunakan tipe data dan operator sesuai dengan kebutuhan.

## 2.3. Uraian Materi

### A. Pengantar Operator

Operator simbol khusus yang biasa dilibatkan dalam pembuatan program untuk melakukan suatu operasi matematika maupun manipulasi logika. Pada dasarnya, ada tiga jenis operator dalam pemrograman C++:

1. **Operator Unary** - operator yang dikenakan untuk satu buah nilai (operand)
2. **Operator Binary** - operator yang dikenakan untuk dua buah nilai (operand)
3. **Operator Ternary** - operator yang dikenakan untuk tiga buah nilai (operand)

Sebagian operator C tergolong sebagai operator binary, yaitu operator yang dikenakan terhadap dua buah nilai (operand).

**Contoh :**

**a - b**

simbol “ **-** “ merupakan operand untuk melakukan pengurangan dari a dan b, karena operator pengurangan melibatkan dua operand, operator pengurangan tergolong sebagai operator binary.

Contoh lain :

**-c**

simbol “ - “ (tanda minus, bukan tanda kurang/ pengurangan) merupakan unary, karena hanya memiliki sebauh operand (yaitu c pada contoh diatas).

Ungkapan (ekspresi) dalam C dapat berupa :

- Pengenal
- Konstanta
- Diantara kombinasi elemen diatas dengan operator

Contoh ungkapan :

**10 - 5 + 1**

Pada ungkapan di atas, 10, 5 dan 1 merupakan operand dan simbol " - " serta " + " adalah operator. Nilai ungkapan sendiri adalah hasil pengurangan 10 dan 5, ditambah 1.

## B. Operator Aritmatika

Operator untuk aritmatika yang tergolong sebagai operator binary. Contoh penggunan operator aritmatika misalnya untuk memperoleh nilai diskriminan dari suatu persamaan kuadrat.

$$d = b^2 - 4ac$$

untuk mengimplementasikan contoh diatas adalah seperti berikut :

```
d = b * b – 4 * a * c ;
```

Contoh program pemakaian operator Aritmatika

```
#include <stdio.h>
int main(void)
{
 int a, b, c, d;
 a = 10;
 b = 60;
 c = 5;
 d = (b*b)-(4*a*c);
 printf("d = %d \n",d);
 return 0;
}
```

Hasil Eksekusi :

```
d = 3400
```

Operator aritmatika mempunyai prioritas pengerjaan. Prioritas yang tinggi akan diuatamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki prioritas yang lebih rendah. Urutan prioritas dapat dilihat dalam tabel berikut ini :

| Operator | Prioritas |
|---|---|
| + -- (Khusus yang berkedudukan sebagai awalan) | Tertinggi |
| - ( Unary Minus ) | |
| * / % | |
| + - | Terendah |

Jika operator memiliki prioritas yang sama, operator sebelah kiri akan diutamakan untuk dikerjakan terlebih dahulu. Tanda kurung biasa digunakan untuk merubah urutan pengerjaan.

Misalnya : x = ( 8 + 5 ) * 2 ;

akan memberikan nilai x = 26, sebab 8 + 5 dikerjakan terlebih dahulu dan hasilnya baru dikalikan dengan 2.

Contoh program penggunaan kurung untuk mengatur prioritas pengerjaan terhadap suatu operasi

```
#include <stdio.h>
int main(void)
{
 int x ;
 x = 4+3*2;
 printf("x = %d \n",x);
 x = (4+3)*2;
 printf("x = %d \n",x);
 return 0;
}
```

Hasil Eksekusi :

```
10
14
```

## C. Operator Sisa Pembagian

Operator sisa pembagian (operator modulus) yang berupa %. Operator ini diterapkan pada operand bertipe integer. Lebih jelasnya perhatikan contoh berikut :

| | |
|---|---|
| 7 % 2 → 1 | 7:2=3, Sisa pembagian adalah 1 |
| 6 % 2 → 0 | 6:2=3, Sisa pembagian adalah 0 |
| 8 % 3 → 2 | 8:3=2, Sisa pembagian adalah 2 |

Contoh program melihat sisi pembagian dengan menggunakan operator %

```
#include <stdio.h>
int main(void) {
int a, b, c;
 a = 7 % 7;
 b = 8 % 7;
 c = 9 % 7;
 printf("Isi  a: %d \n",a);
 printf("Isi  b: %d \n",b);
 printf("Isi  c: %d \n",c);
 return 0;
 }
```

Hasil Eksekusi :

```
Isi a: 0
Isi b: 1
Isi c: 2
```

Kegunaan operator % diantaranya bisa dipakai untuk menentukan suatu bilangan bulat termasuk ganjil atau genap.

## D. Operator Penurunan dan Penaikan

Kedua operator ini digunakan pada operand bertipe bilangan bulat. Operator penaikan digunakan untuk menaikan nilai variabel sebesar satu, sedangkan operator penurunan dipakai untuk menurunkan nilai variabel sebesar satu. Sebagai contoh :

x = x + 1 ;

y = y – 1 ;

bisa ditulis menjadi :

```
++ x ;
-- y ;
```

atau :

```
x ++ ;
y -- ;
```

| | | |
|---|---|---|
| Pre-increment | ++x | Tambah x sebanyak 1 angka, lalu tampilkan hasilnya |
| Post-increment | x++ | Tampilkan nilai x, lalu tambah x sebanyak 1 angka |
| Pre-decrement | --y | Kurangi y sebanyak 1 angka, lalu tampilkan hasilnya |
| Post-decrement | y-- | Tampilkan nilai y, lalu kurangi y sebanyak 1 angka |

**Kenaikan nilai di belakang**

Efek peletakkan tanda ++ dibelakang variabel ditunjukkan pada program berikut :

Contoh program pemakaian operator kenaikan di belakang variabel

```
#include <stdio.h>
int main(void)
{
int t = 10, f;
    f = 10+t++ ;
printf("t: %d \n",t);
printf("f: %d \n",f);
return 0;
}
```

Hasil Eksekusi :

```
t = 11
f = 20
```

Pada contoh di atas f diisi dengan penjumlahan nilai 10 dan t. Dengan demikian f akan bernilai 20. setelah f diisi dengan 20, nilai t baru dinaikan karena operator ++ ditulis dibelakang t. Disebut post-increment yang artinya dinaikkan dibelakang setelah penjumlahan anatara r dan 10 dilaksanakan.

**Kenaikan nilai di depan**

Efek peletakkan tanda ++ di depan variabel ditunjukkan pada program berikut ini :

Contoh program Pemakaian operator kenaikan nilai di depan variabel

```
#include <stdio.h>
int main(void)
{
 int t = 10, f;
     f = 10+ ++t ;
 printf("t: %d \n",t);
 printf("f: %d \n",f);
 return 0;
 }
```

Hasil Eksekusi :

```
t = 11
f = 21
```

Pada contoh ini, nilai t mula-mula dinaikan terlebih dahulu karena operator ++ ditempatkan didepan t. Disebut pre-increment kemudian nilainnya dijumlahkan dengan 10 dan diberikan ke f. Dengan demikian f bernilai 21 dan t sama dengan 11.

## E. Operator Majemuk

Operator majemuk digunakan untuk memperpendek penulisan operasi penugasan, contoh:

x = x + 2 ;

y = y * 4 ;

menjadi :

x += 2;

y *= 4;

Contoh program penggunaan operator majemuk

```
#include <stdio.h>
int main(void)
{
int x = 2;
printf("x= %d \n",x);
x+=3;
printf("Setelah x += 3, x= %d \n",x);
```

```
x*=2;
printf("Setelah x *= 2, x= %d \n",x);
return 0;
}
```

Hasil Eksekusi :

```
x = 2
Setelah x += 3, x = 5
Setelah x *= 2, x = 10
```

## F. Operator Kondisi

Operator kondisi biasa dipakai untuk mendapatkan sebuah nilai dari dua buah kemungkinan, berdasarkan suatu kondisi. Format pemakaiannya :

```
ungkapan1 ? ungkapan 2 : ungkapan 3
```

Contoh program penggunaan operator kondisi untuk memperoleh bilangan terkecil diantara dua buah bilangan

```
#include <stdio.h>
int main(void) {
int bil1, bil2, kecil;
bil1 = 32;
bil2 = 27;
kecil = bil1<bil2?bil1:bil2;
printf("Bilangan terkecil= %d \n",kecil);

return 0;
}
```

Hasil Eksekusi :

```
Bilangan terkecil= 27
```

kecil = bil1 < bil2 ? bil1 : bil2;

akan menyebabkan kecil bernilai bil1 kalau ungkapan :

bil1 < bil2

bernilai benar. Untuk keadaan sebaliknya, kecil akan bernilai bil2.

## G. Ungkapan Kondisi

Ungkapan adalah ungkapan yang menjadi dasar bagi pernyataan berkondisi (misalnya **if** ). Hasil ungkapan berupa 1 kalau ungkapan bernilai benar dan ungkapan berupa 0 kalau ungkapan bernilai salah.

Operator Relasi

Operator biasa digunakan untuk membandingkan dua buah nilai. Macam operator relasi dapat dilihat dalam tabel berikut :

| Operator | Keterangan |
|---|---|
| == | Sama dengan (bukan penugasan) |
| != | Tidak sama dengan |
| > | Lebih dari |
| < | Kurang dari |
| >= | Lebih dari atau sama dengan |
| <= | Kurang dari atau sama dengan |

Agar tidak salah dalam menuliskan suatu ungkapan, pengetahuan tentang prioritas operator perlu diketahui.

Contoh 1 :

```
a = b = c
```

pada pernyataan diatas, operator yang dilibatkan ( = ) mempunyai sifat pengerjaan dimulai dari kanan. Berarti :

```
b = c
```

akan dikerjakan terlebih dahulu, barulah kemudian mengerjakan :

```
a = b
```

Contoh 2 :

```
x = 2 * 3 * 4 ;
```

pada pernyataan diatas, 2 * 3 akan dikerjakan terlebih dahulu, barulah kemudian mengerjakan perkalian hasil 6 dengan 4. Adapun prioritas = lebih rendah dari *, maka 2 * 3 * 4 dikerjakan lebih dahulu. Selanjutnya hasilnya baru diberikan ke x.

## H. Fungsi Pustaka

Pada Bahasa C, operasi akar kuadrat ataupun memperoleh logaritma alamiah dari suatu nilai memang tidak terdapat operator-operator yang khusus untuk melaksanakan operasi-operasi seperti itu. Tetapi tidak berarti hal ini tidak dapat dilakukan. Bahasa C

menyediakan sejumlah fungsi pustaka (*library fuctions*) yang dirancang untuk memenuhi solusi dari berbagai persoalan.

Jika program ingin menggunakan fungsi pustaka, perlulah untuk mencatumkan deklarasi dari funsi bersangkutan. Untuk keperluan ini program mesti menyertakan baris :

**#include <nama_file>**

degan *nama_file* adalah nama *header*, yaitu file yang berakhiran **.h**. sebagai contoh program diatas menyertakan **#include <math.h>** disebabkan file *header* tersebut berisi deklarasi (prototipe) dari fungsi **sqrt()**.

## 2.4.Latihan

1. Contoh menghitung sebuah akar kuadrat, digunakan fungsi **sqrt()**. Seperti contoh program berikut :

```
#include<stdio.h>
#include<math.h> // sertakan untuk menggunakan fungsi sqrt
int main(void)
{
int x,y;
    x = 4;
    y = sqrt(x);
printf("y= %d \n",y);
return 0;
}
```

Hasil Eksekusi :

```
y= 2
```

2. Contoh program untuk menunjukkan nilai hasil ungkapan kondisi

```
#include <stdio.h>
int main(void) {
int nilai;
nilai = 3 > 2 ; // hasil ungkapan  : benar
printf("Nilai= %d \n",nilai);
nilai = 2 > 3 ; // hasil ungkapan  : salah
printf("Nilai= %d \n",nilai);
return 0;
}
```

Hasil Eksekusi :

```
Nilai= 1
Nilai= 0
```

**2.5.Tugas**

1. (simpan program untuk A -> tugas2A.c dan B -> tugas2B.c)
   Skala peta adalah 1: 250.000. Jika jarak pada peta 5 cm dari kota T menuju kota F, jarak sebenarnya kedua kota tersebut adalah…
2. Pipa air mengeluarkan air dengan volume 20 liter tiap menitnya. Berapakah kira-kira debit air yang keluar dari pipa dengan satuan liter per detik?

# BAB III
# PERCABANGAN (SELECTION) DENGAN IF

## 3.1. Capaian Pembelajaran

- Mengetahui dasar-dasar percabangan.
- Mengetahui penulisan percabangan menggunakan IF.
- Mengetahui pemilihan jenis percabangan yang tepat sesuai dengan kebutuhannya.
- Mampu mengimplementasikan pengkategorian kasus yang sedang terjadi yaitu, covid19 ke dalam bahasa pemrograman.
- Mampu mengimplementasikan pengkategorian data kelulusan mata kuliah dan konversi nilai ke dalam bahasa pemrograman.

## 3.2. Indikator

- Mampu menjelaskan dasar-dasar percabangan.
- Mampu menuliskan contoh program menggunakan percabangan.
- Mampu memilih jenis percabangan sesuai kasus yang sedang dikerjakan.

## 3.3. Uraian Materi

### A. Pengantar Kondisi

### a. Operator Kondisi

Banyak persoalan yang diperlukan untuk membuat keputusan. Contoh yang sederhana berupa cara mengatur agar komputer bisa menyimpulkan bahwa suatu bilangan merupakan bilangan genap atau bilangan ganjil. Untuk keperluan pengambilan keputusan semacam itu, C menyediakan beberapa jenis pernyataan, berupa

- Pernyataan *if*
- Pernyataan *if-else*, dan
- Pernyataan *switch*

Pernyataan-pernyataan tersebut memerlukan suatu kondisi, sebagai basis dalam pengambilan keputusan. Kondisi umum yang dipakai berupa keadaan benar dan salah. Oleh karena itu pembahasan pada bab ini akan diawali dengan pengenalan operator yang membentuk kondisi benar dan salah.

Operator yang digunakan untuk menghasilkan kondisi benar dan salah, bisa berupa operator relasi dan bisa juga berupa operator logika. Berikut ini dibahas masing-masing

jenis operator serta tabel prioritas masing-masing operator.

**b. Operator Relasi**

Operator relasi biasa dipakai untuk membandingkan dua buah nilai. Hasil pembandingan berupa keadaan benar atau salah. Keseluruhan operator relasi pada C ditunjukkan pada Tabel 4.1.

Tabel 4.1. Operasi relasi

| Operator | Makna |
|---|---|
| > | Lebih dari |
| >= | Lebih dari atau sama dengan |
| < | Kurang dari |
| <= | Kurang dari atau sama dengan |
| == | Sama dengan |
| != | Tidak sama dengan |

Khususnya untuk operator relasi sama dengan (==) harap dibedakan dengan operator (=) yang merupakan operator penugasan (assignment).

**Contoh :**

| **Pembandingan** | | **Hasil** |
|---|---|---|
| 1 > 2 |  | Salah |
| 1 < 2 |  | Benar |
| A == 1 |  | Benar, jika A bernilai 1<br>Salah, jika A tidak bernilai 1 |
| 'A' < 'B' |  | Benar, karena kode ASCII untuk karakter ‘A’ kurang dari kode ASCII untuk karakter ‘B’ *) |
| kar == 'Y' |  | Benar, jika kar berisi 'Y'<br>Salah, jika kar tidak berisi 'Y' |

*) Dalam daftar ASCII standar, kode untuk karakter ‘A’ = 65 sedangkan karakter ‘B’ = 66, ‘C’ = 67, ‘D’ = 68 dan seterusnya sampai dengan karakter ‘Z’ = 90.

### c. Operator Logika

Operator logika biasa dipakai untuk menghubungkan ekspresi relasi. Keseluruhan operator logika ditunjukkan pada tabel 4.2.

Tabel 4.2. Operator logika

| Operator | Makna |
|---|---|
| && | dan (AND) |
| \|\| | atau (OR) |
| ! | tidak (NOT) |

Bentuk pemakaian operator && dan || adalah

**operand1 operator operand2**

Baik **operand1** maupun **operand2** dapat berupa ekspresi relasi ataupun ekspresi logika. Hasil ekspresi bisa bernilai benar atau salah. Pada C nilai hasil dari sebuah ekspresi relasi atau ekspresi logika jika dinyatakan dengan angka adalah :

Salah $\rightarrow$ nilai = *0*

Benar $\rightarrow$ nilai != *0* (misalnya nilai = *1*)

Tabel 4.3 memberikan penjelasan hasil operasi ekspresi logika yang menggunakan operator && maupun || untuk berbagai kemungkinan keadaan operand-nya.

Tabel 4.3. Kemungkinan pada operasi logika && dan ||

| Operand1 | Operand2 | Hasil | |
|---|---|---|---|
| | | \|\| | && |
| Salah | Salah | 0 | 0 |
| Salah | Benar | 1 | 0 |
| Benar | Salah | 1 | 0 |
| Benar | Benar | 1 | 1 |

Tampak bahwa operator **atau** (||) menghasilkan nilai *1* jika ada operand yang benar. Hasil berupa *0* jika semua operand adalah salah. Adapun operator logika **dan** (&&) memberikan hasil *1* hanya jika kedua operand adalah benar.

Beberapa contoh ekspresi logika di antaranya :

- (kar > 'A') && (kar < 'Z')

  Hasil operasi logika && adalah benar hanya jika kar > 'A' dan kar < 'Z'

(dalam hal ini yang diperbandingkan adalah kode ASCII dari karakter tsb).

- (pilihan == 'Y') || (pilihan == 'y')

Hasil operasi logika || adalah benar jika pilihan berupa *'Y'* atau *'y'*

Sedangkan bentuk pemakaian operator logika ! adalah :

**!operand**

dengan **operand** dapat berupa ekspresi logika ataupun ekspresi relasi.

Hasil operasi ! bernilai :

- *1* jika **operand** bernilai salah
- *0* jika **operand** bernilai benar

Perhatikan contoh potongan program di bawah ini :

```
if (!sudah_benar)
printf("Masukan Anda salah!\n");
```

Pada contoh potongan program di atas, dilakukan pengecekan kondisi terhadap nilai dari variabel **sudah_benar**. Jika variabel **sudah_benar** bernilai *0*, maka kondisi **!sudah_benar** akan bernilai benar (*true*) sehingga instruksi :

```
printf("Masukan Anda salah!\n");
```

akan diproses. Penjelasan lebih rinci tentang pengecekan kondisi dengan pernyataan if dibahas pada sub lainnya.

## B. Prioritas Operator Logika dan Relasi

Tabel 4.4 memberikan penjelasan singkat mengenai prioritas di antara berbagai operator logika dan operator relasi.

Tabel 4.4 Prioritas operator logika dan relasi

| | |
|---|---|
| **Tertinggi** : | ! |
| | > >= < <= |
| | = = != |
| | && |
| **Terendah** : | \|\| |

Berdasarkan prioritas yang ditunjukkan pada tabel 3-4, maka ekspresi seperti

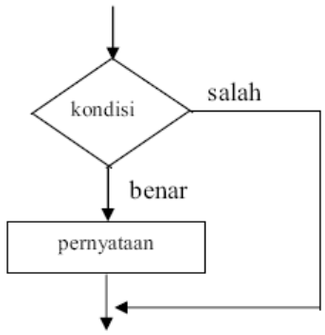
```
(kar > 'A') && (kar < 'Z')
```

sama saja kalau ditulis menjadi

```
kar > 'A' && kar < 'Z'
```

Hanya saja penulisan dengan menggunakan tanda kurung akan lebih memberikan kejelasan.

**a. Pernyataan *if***




Gambar 4.1. Diagram alir *if*

Pernyataan if mempunyai bentuk umum :

**if (kondisi)**

**pernyataan;**

Bentuk ini menyatakan :

- Jika kondisi yang diseleksi adalah benar (bernilai logika = *1*), maka pernyataan yang mengikutinya akan diproses.
- Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = *0*), maka pernyataan yang mengikutinya tidak akan diproses.

Mengenai kodisi harus ditulis diantara tanda kurung, sedangkan pernyataan dapat berupa sebuah pernyataan tunggal, pernyataan majemuk atau pernyataan kosong. Diagram alir dapat dilihat seperti Gambar 4.1.

Contoh penggunaan pernyataan if, misalkan untuk menentukan besarnya potongan harga yang diterima oleh seorang pembeli, berdasarkan kriteria:

- Tidak ada potongan harga jika total pembelian kurang dari Rp. 100.000 (dalam hal ini potongan harga diinisialisasi dengan nol).
- Bila total pembelian lebih dari atau sama dengan Rp. 100.000, potongan harga yang diterima dirubah menjadi sebesar 5% dari total pembelian.

Contoh penggunaan if untuk menghitung nilai discount

```
#include <stdio.h>
main()
{
double total_pembelian, discount = 0;
/* discount diinisialisasi dengan nilai 0 */
printf("Total pembelian = Rp ");
scanf("%lf", &total_pembelian);
```

```
if(total_pembelian >= 100.000)
discount = 0.05 * total_pembelian;
printf("Besarnya discount = Rp %.2lf\n", discount);
}
```

Hasil Eksekusi :

```
Total pembelian     = Rp 200000
Besarnya discount = Rp 10000.00
```

Untuk pernyataan ***if*** yang diikuti dengan pernyataan majemuk, bentuknya adalah sebagai berikut :

```
if (kondisi)
{ /* tanda awal pernyataan majemuk*/
      pernyataan-1;
      pernyataan-2;
      .
      .
      .
      pernyataan-n;
} /* tanda akhir pernyataan majemuk */
```

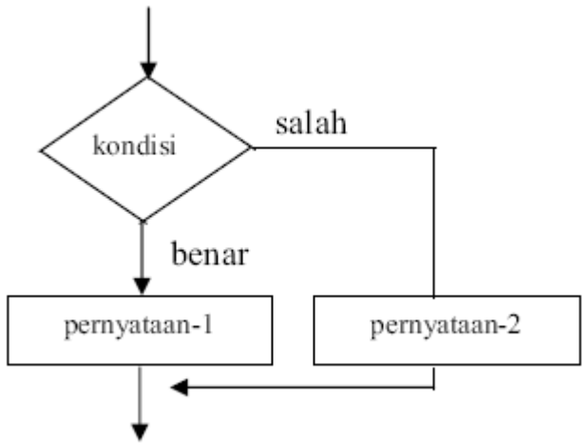
Pernyataan-pernyataan yang berada dalam tanda kurung *{* dan *}* akan dijalankan hanya bila kondisi *if* bernilai benar.

**b. Pernyataan *if-else***

Pernyataan *if-else* memiliki bentuk :

```
if (kondisi)
      pernyataan-1;
else
      pernyataan-2;
```

Diagram alir dapat dilihat seperti gambar 4.2.




Gambar 4.2. Diagram alir *if-else*

Arti dari pernyataan *if-else* :

- Jika kondisi benar, maka **pernyataan-1** dijalankan.
- Sedangkan bila kondisi bernilai salah, maka **pernyataaan-2** yang dijalankan.

Masing-masing **pernyataan-1** dan **pernyataan-2** dapat berupa sebuah pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong.

Contoh penggunaan pernyataan *if-else*, misalkan untuk menyeleksi nilai suatu bilangan pembagi. Jika nilai bilangan pembagi adalah nol, maka hasil pembagian dengan nilai nol akan mendapatkan hasil tak berhingga. Jika ditemui nilai pembaginya nol, maka proses pembagian tidak akan dilakukan.

Contoh Pemakaian if-else untuk menyeleksi bilangan pembagi

```
#include <stdio.h>
main()
{
float a, b;
printf("Masukkan nilai a : ");
scanf("%f", &a);
printf("Masukkan nilai b : ");
scanf("%f", &b);
if (b == 0)
printf("\n%g dibagi dengan nol = TAK BERHINGGA\n", a);
else
printf("\n%g dibagi dengan %g = %g\n", a, b, a/b);
}
```

Hasil Eksekusi :

```
Masukkan nilai a : 5
Masukkan nilai b : 0
dibagi dengan nol = TAK BERHINGGA
```

### 3.4. Latihan

1. Program konveksi nilai dengan if

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

main(){
int nilai;
/*Membuat program konversi nilai huruf dari nilai yang dinputkan
user*/
printf("Nilai: "); scanf("%d", &nilai);
printf("\nNilai yang diperoleh: ");
if(nilai >=95){
printf("A+");
}
if(nilai >=80 && nilai < 95){
printf("A");
}
if(nilai>=65 && nilai < 80){
printf("B");
}
if(nilai>=50 && nilai < 65){
printf("C");
}
```

Hasil Eksekusi :

```
Nilai : 80
Nilai yang diperoleh : A
```

2. Program pilihan operasi matematika

```
#include<stdio.h>

int main()
{
int pilihan, nilai1, nilai2, hasil;
int tambah, kurang, kali;
printf("Masukkan Nilai Pertama : ");
scanf("%d", &nilai1);
printf("Masukkan Nilai Kedua : ");
scanf("%d", &nilai2);
}
printf("\nMau Melakukan Operasi apa ?");//\n untuk enter 1x
printf("\n\t1. Tambah\n\t2. Kurang\n\t3. Kali");//\t untuk tab 1x
printf("\nTentukan Pilihan : ");
scanf("%d", &pilihan);
if(pilihan == 1) // dibelakang if tidak ada tanda tiitk koma ";"
hasil = nilai1 + nilai2;
else if(pilihan == 2)
hasil = nilai1 - nilai2;
else
hasil = nilai1 * nilai2;
printf("Hasil : %d\n", hasil);
return 0;
```

Hasil Eksekusi :

```
Masukan nilai pertama : 3
Masukan nilai kedua : 2


Mau Melakukan Operasi apa ?
1. Tambah
2. Kurang
3. Kali
Tentukan Pilihan :
Hasil :6
```

### 3.5.Tugas

1. Buatlah program menentukan kategori pasien covid-19 akan dikarantina atau isolasi mandiri. Dengan ketentuan jika suhu pasien lebih dari 37 °C maka berkategori karantina rumah sakit, jika suhu dibawah suhu tersebut maka pasien diharuskan melakukan karantina mandiri dirumah.
2. Buatlah program untuk menentukan sebuah nilai kelulusan dari mata kuliah pemrograman dasar, dengan syarat kelulusan.
   Jika nilai ≥ 75 maka lulus
   Jika nilai ≤ 75 maka mengulang
3. Berkelanjutan dari program diatas silahkan konversi nilai angka tersebut menjadi nilai huruf.
   Misal :
   80 → A
   70 → B
   60 → C

# BAB IV
# PERCABANGAN (SELECTION) IF BERTINGKAT

## 4.1. Capaian Pembelajaran

- Mengetahui penulisan percabangan menggunakan IF.
- Mampu mengimplementasikan pengkategorian data manajemen penerimaan pasien rumah sakit menggunakan bahasa pemrograman.

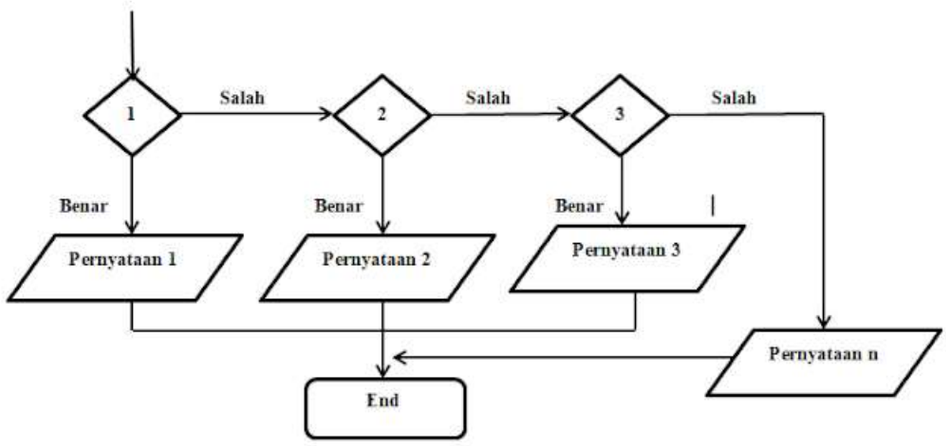
## 4.2. Indikator

- Mampu membuat program menggunakan nested IF.

## 4.3. Uraian Materi

### A. Pernyataan if else bertingkat

Pada percabangan bahasa C ada yang disebut sebagai struktur if seperti yang telah kita pelajari pada BAB III, struktur if ini terbagi untuk berbagai kondisi. Kondisi tersebut biasanya digunakan jika kondisi berjumlah satu, kondisi berjumlah dua dan kondisi yang berjumlah lebih dari dua. Kondisi yang berjumlah lebih dari dua tersebut disebut dengan if else bertingkat. if else bertingkat memiliki flowchart sebagai berikut:




Sesuai dengan struktur dan flowchart di atas, maka dalam hal ini jika kondisi 1 bernilai benar maka pernyataan 1 akan di kerjakan (dicetak program), apabila kondisi 1 salah maka kondisi 2 yang akan dilihat atau di cek. Jika kondisi 2 benar maka pernyataan 2 yang akan dikerjakan, apabila kondisi 2 salah maka kondisi 3 yang akan dilihat atau di cek. Jika kondisi 3 benar maka pernyataan 3 akan dikerjakan, apabila jika kondisi 3 masih juga salah , maka pernyataan terakhir yang dikerjakan yaitu pernyataan n.

Contoh program penggunaan if else bertingkat

```
#include<stdio.h>
int main() {
int makanan;
printf("Menu Makanan\n");
printf("============\n");
printf("1.Nasi Goreng\n");
printf("2.Pecel Lele\n");
printf("3.Ikan Bakar\n");
printf("Masukkan pilihan anda (1-3)? ");
scanf("%d", &makanan);
if(makanan==1){
printf("Menu makanan pilihan anda adalah nasi goreng \n");
              }
else if(makanan==2){
printf("Menu makanan pilihan anda adalah pecel lele \n");
                   }
else if(makanan==3){
printf("Menu makanan pilihan anda adalah ikan bakar\n");
                     }
else{
printf("Menu makanan tidak ada\n");
}
return 0;
}
```

Hasil Eksekusi :

```
Menu Makanan
============
1.Nasi Goreng
2.Pecel Lele
3.Ikan Bakar
Masukkan pilihan anda (1-3)?
(User memasukkan angka 3 menggunakan keyboard)
Menu makanan pilihan anda adalah ikan bakar
```

**B. Pernyataan if dalam if (Nested IF)**

Selain if else bertingkat yang bisa digunakan dengan beberapa kondisi, nested if juga bisa juga bisa digunakan jika banyak kondisi. Tetapi nested if berbeda dengan if else

bertingkat, untuk nested if memiliki if yang di dalamnya terdapat if juga . Secara umum struktur bentuk dari pernyataan nested if adalah sebagai berikut:

```
if(kondisi 1){
      if(kondisi a)
  pernyatan;

  else if(kondisi n)
  pernyataan;

        else
      }

  else if (kondisi 2){ // else IF kondisi 1 tidak terpenuhi maka
  proses lompat ke sini
  if(kondisi a)
  pernyatan;

  else if(kondisi n)
  pernyataan;

        else
   }

else {
        pernyataan;
     }
```

Kondisi yang akan diseleksi pertama kali adalah yang terluar atau kondisi 1. Jika kondisi 1 bernilai salah, maka statement else yang diluar (pasangan dari if kondisi 1) yang akan di proses, yaitu kondisi 2. Tetapi jika kondisi 1 bernilai benar, maka kondisi berikutnya yang berada di dalam kondisi 1, yaitu kondisi a dst yang akan diproses atau dicek. Tetapi jika kondisi 1 bernilai salah, maka statement else pasangan if kondisi 2 akan diproses. Tetapi jika kondisi 2 bernilai salah, maka penyeleksian akan dilakukan sampai dengan kondisi n.

### 4.4. Latihan

1. Contoh program pemakaian nested if

```
#include<stdio.h>
int main() {
int jk,tinggi;
printf("Program Seleksi Penerimaan Model Brand Cloht\n");
printf("=======================================\n");
printf("==jenis kelamin==\n");
printf("1.Laki-Laki\n");
printf("2.Perempuan\n");
printf("Pilih jenis kelamin anda (1 atau 2): ");
scanf("%d", &jk);
printf("\n\n==Tinggi badan==\n");
printf("Masukkan tinggi badan anda :");
scanf("%d", &tinggi);
if(jk==1)
{
  if(175 <= tinggi)
  printf("Selamat anda lolos seleksi ");
  else if(175 > tinggi)
  printf("Maaf , anda tidak lolos seleksi");
  else("Data yang anda masukkan tidak sesuai");
}
else if(jk==2)
{
  if(165 <= tinggi)
  printf("Selamat anda lolos seleksi");
  else if(165 > tinggi )
  printf("Maaf, anda tidak lolos seleksi");
  else
  printf("Data yang anda masukkan tidak sesuai");
}
else
{
printf("Masukkan dengan benar");
}
}
```

Hasil Eksekusi :

```
Program Seleksi Penerimaan Model Brand Cloht
========================================
==jenis kelamin==
1.Laki-Laki
2.Perempuan
 Pilih jenis kelamin anda (1 atau 2): 2 (User memasukkan angka 2 menggunakan keyboard)

==Tinggi badan==
Masukkan tinggi badan anda : 170 (User memasukkan angka 170 menggunakan keyboard)
Selamat anda lolos seleksi
```

### 4.5. Tugas

1. Sebuah rumah sakit menerima pasien dengan dua jenis kartu, BPJS dan Umum. Buatlah program menggunakan nested if, dengan ketentuan yang bisa anda cari di internet, apa saja administrasi yang harus dilengkapi jika menggunakan BPJS atau saat pasien tersebut mendaftar sebagai pasien umum. Selain itu tidak diterima sebagai pasien

# BAB V
# PERCABANGAN (SELECTION) DENGAN SWITCH

## 5.1. Capaian Pembelajaran

- Mengetahui penulisan percabangan menggunakan SWITCH.
- Mampu mengimplementasikan perhitungan koordinat secara keilmuan matematika, serta konversi numerik, ke dalam bahasa pemrograman.

## 5.2. Indikator

- Mampu membuat program menggunakan nested SWITCH.

## 5.3. Uraian Materi

### A. Pengantar Switch

Pernyataan *switch* merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah alternatif, misalnya untuk menggantikan pernyataan *if* bertingkat.

Bentuk umum pernyataan *switch* adalah :

```
switch (ekspresi)
{
case konstanta-1:
pernyataan-1;
......
break;
case konstanta-2:
.
.
.
case konstanta-n:
pernyataan-n;
.......
break;
default:
.......
.......
break;
}
```

dengan **ekspresi** dapat berupa ekspresi bertipe integer atau bertipe karakter. Demikian

juga **konstanta-1**, **konstanta-2**, …, **konstanta-n** dapat berupa konstanta integer atau karakter. Setiap pernyataan-i (**pernyataan-1**, …, **pernyataan-n**) dapat berupa pernyataan tunggal ataupun pernyataan jamak. Dalam hal ini urutan penulisan pernyataan *case* tidak berpengaruh. Proses penyeleksian berlangsung sebagai berikut :

- Pengujian pada *switch* akan dimulai dari **konstanta-1**. Kalau nilai **konstanta-1** cocok dengan ekspresi maka **pernyataan-1** dijalankan. Kata kunci *break* harus disertakan di bagian akhir setiap pernyataan *case*, yang akan mengarahkan Eksekusi ke akhir *switch*.
- Kalau ternyata **pernyataan-1** tidak sama dengan nilai **ekspresi**, pengujian dilanjutkan pada **konstanta-2**, dan berikutnya serupa dengan pengujian pada **konstanta-1**.
- Jika sampai pada pengujian *case* yang terakhir ternyata tidak ada kecocokan, maka pernyataan yang mengikuti kata kunci *default* yang akan diEksekusi. Kata kunci *default* ini bersifat opsional.
- Tanda kurung kurawal tutup (*}*) menandakan akhir dari proses penyeleksian kondisi *case*.

Di bawah ini contoh program pemakaian pernyataan *switch* untuk menggantikan *if-else* bertingkat pada program **kalkulator1.c** di atas.

Contoh penggunaan pernyataan switch untuk mengimplementasikan kalkulator sederhana

```
#include <stdio.h>
main(){
int valid_operator = 1;
char operator;
float number1, number2, result;
printf("Masukkan 2 buah bilangan dan sebuah operator\n");
printf("dengan format : number1 operator number2\n\n");
scanf("%f %c %f", &number1, &operator, &number2);
switch(operator) {
case '*' : result = number1 * number2; break;
case '/' : result = number1 / number2; break;
case '+' : result = number1 + number2; break;
case '-' : result = number1 - number2; break;
default : valid_operator = 0;
      }
      if(valid_operator)
printf("%g %c %g is %g\n", number1, operator, number2,result);
else
```

```
printf("Invalid operator!\n");
}
```

Contoh Eksekusi :

```
Masukkan 2 buah bilangan dan sebuah operator
Dengan format : number1 operator number2
23.2 = 12
invalid operator !
```

### 5.4. Latihan

1. Program menampilkan nilai

```
#include <stdio.h>
 int main(void)
{
char nilai;
printf("Input Nilai Anda (A - E): ");
scanf("%c",&nilai);

switch (nilai) {
case 'A':
printf("Pertahankan! \n");
break;
case 'B':
printf("Harus lebih baik lagi \n");
break;
case 'C':
printf("Perbanyak belajar \n");
break;
case 'D':
printf("Jangan keseringan main \n");
break;
case 'E':
printf("Kebanyakan bolos... \n");
break;
default:
printf("Maaf, format nilai tidak sesuai \n");
}
return 0;
}
```

Hasil Eksekusi :

```
Input Nilai Anda (A - E): A
Pertahankan!

Input Nilai Anda (A - E): D
Jangan keseringan main

Input Nilai Anda (A - E): E
Kebanyakan bolos...

Input Nilai Anda (A - E): F
Maaf, format nilai tidak sesuai
```

2. Program Konveksi nilai angka menjadi huruf

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

int main(int argc, char *argv[])
{
  int nilai;
printf("Masukkan nilai ujian anda: ");
scanf("%d",&nilai);
if((nilai>100)||(nilai<0))
printf("Maaf, Masukan salah!\nnilai diluar jangkauan !!");
else if(nilai==100) printf("Nilai = A");
else {
switch (nilai/10){
case 9: printf("Nilai = A");
break;
case 8: printf("Nilai = B");
break;
case 7: printf("Nilai = C");
break;
case 6: printf("Nilai = D");
break;
default: printf("Nilai = F");
break;
```

```
  }
  }
    printf("\n");
    system("PAUSE");
    return 0;
  }
```

Hasil Eksekusi :

```
Masukkan nilai ujian anda:80
B
```

### 5.5. Tugas

1. Buatlah program untuk menentukan posisi suatu titik pada koordinat cartesius berada pada kuadran ke berapa. Masukan dari program ini adalah sumbu X dan sumbu Y. Untuk menentukan berada pada kuadran berapa didasarkan pada :
   Jika sumbu x positif dan sumbu Y positif maka berada pada kuadrat I jika sumbu X negatif dan sumbu Y positif maka berada pada kuadtan II jika sumbu X negatif dan sumbu Y negatif maka berada pada kuadrat III jika sumbu X positif dan sumbu Y negatif maka berada pada kuadran IV.
2. Buatlah program untuk mengkonversi suatu angka numerik menjadi suatu bilangan. Misalnya jika dimasukan angka 1200 maka akan ditampilkan tulisan "seribu dua ratus".

# BAB VI
# PERULANGAN (LOOPING) DENGAN FOR

## 6.1. Capaian Pembelajaran

- Mengetahui dasar-dasar perulangan.
- Mengetahui penulisan perulangan dengan FOR.
- Mampu menerapkan proses input data pasien berkali-kali dalam contoh kasus data pasien rumah sakit ke dalam bahasa pemrograman.

## 6.2. Indikator

- Mampu menjelaskan dasar-dasar perulangan.
- Mampu menuliskan contoh program menggunakan perulangan FOR

## 6.3. Uraian Materi

### A. Pernyataan for

Pernyataan for berguna untuk menggulang pengEksekusian terhadap satu atau sejumlah pernyataan.

Bentuk format :

**for** (*ungkapan1*; *ungkapan2*; *ungkapan3*)
*pernyataan*;

Contoh program menampilkan bilangan genap yang nilainya kurang atau sama dengan n dan ditampilkan dari terbesar sampai nol

```
#include <stdio.h>
int main(){
int n;
printf("Menampilkan bilangan genap yang nilainya kurang atau sama
dengan n\n");
printf("Masukkan nilai n = ");
scanf("%d", &n);
// Jika n ganjil, maka dikurangi 1
if ( n % 2)
     n --;
// tampilkan deret bilangan genap dari besar ke kecil
for (   ; n >= 0; n -= 2 )
printf("%d ",n);
```

```
return 0;
}
```

Hasil Eksekusi :

```
Menampilkan bilangan genap yang nilainya
kurang atau sama dengan n
Masukkan nilai n =  9
8  6  4  2  0
```

Pada program diatas terdapat :

n --; ungkapan kosong

for ( ; n >= 0; n -= 2 )

sama artinya dengan :

for (n -- ; n >= 0 ; n - = 2 )

**B. Pernyataan Continue**

Kegunaan dari **continue** dipakai untuk mengarahkan Eksekusi ke putaran atau iterasi berikutnya pada pernyataan pengulangan. Efek dari dari perintah ini pada **for**, **while** dan **do-while** :

- Pada **for** :
  Ungkapan ke tiga pada **for** (ungkapan terkanan yang terletak didalam ( ) pada **for** ) akan dijalankan dan kemudian ungkapan ke dua diuji lagi.
- Pada **while** dan **do-while** :
  Pengujian terhadap ungkapan pada **while** dilakkan kembali.

Dalam kode program ini, di baris 5 terdapat kondisi if (i == 3) { continue; }. Jika kondisi ini terpenuhi (saat variabel counter i berisi angka 3), maka jalankan perintah continue.

Hasilnya, perintah printf di baris 8 akan dilompati dan perulangan langsung lompat ke iterasi berikutnya, yakni variabel i akan berisi angka 4. Dalam tampilan akhir bisa terlihat bahwa 5 + 5 = 10 tidak ada di daftar penambahan.

**C. Menghentikan Program dengan exit()**

Suatu Eksekusi program dapat dihentikan melalui pemanggilan funsi **exit()**. Hal ini dapat dilakukan jika dalam sebuah program ada suatu Eksekusi kondisi yang tidak dikehendaki.

Bentuk pemakaian **exit()** : **exit** (*nilai_keluar*);

*nilai_keluar* dapat disi dengan dengan 0 sampai dengan 255. Umumnya jika program dapat melakukan tugasnya dengan bauk maka nilain keluarnya adalah 0. nilai keluar tidak sama dengan nol untuk menyatakan suatu kesalahan.

### 6.4. Latihan

1. Contoh program penggunaan exit()

```
#include <stdio.h>
#include <stdlib.h>
main(){
 char status;
 cetak:
 printf("\nlatihan exit\n");
 printf("tampilkan lagi? (y/n) :");
 status = getche();
 if(toupper(status)=='Y')
 goto cetak;
 else
 exit(0);
}
```

Hasil Eksekusi:

```
latihan exit
tampilkan lagi? (y/n) : y
latihan exit
tampilkan lagi? (y/n) :
```

2. Contoh menunjukan efek **continue** pada **for** :

```
#include <stdio.h>
int main(void) {
int i;
  for (i = 1; i <= 5; i++) {
    if (i == 3) {
      continue;
    }
printf("%i + %i = %i \n",i,i,i+i);
}
  return 0;
```

```
}
```

Hasil Eksekusi :

```
1+1=2
2+2=4
4+4=8
5+5=10
```

### 6.5. Tugas

1. Berdasarkan tugas Modul IV (Sebuah rumah sakit menerima pasien dengan dua jenis kartu, BPJS dan Umum. Buatlah program menggunakan nested if, dengan ketentuan yang bisa anda cari di internet, apa saja administrasi yang harus dilengkapi jika menggunakan BPJS atau saat pasien tersebut mendaftar sebagai pasien umum. Selain itu tidak diterima sebagai pasien.), tambahkan perintah for untuk memasukkan data pasien sebanyak 3 kali inputan

# BAB VII
# PERULANGAN (LOOPING) DENGAN WHILE

## 7.1. Capaian Pembelajaran

- Mengetahui dasar-dasar perulangan.
- Mengetahui penulisan perulangan dengan WHILE.
- Mampu menerapkan perhitungan kelipatan dan menghitung jumlah bilangan ke dalam bahasa pemrograman.

## 7.2. Indikator

- Mampu menjelaskan dasar-dasar perulangan.
- Mampu menuliskan contoh program menggunakan perulangan WHILE

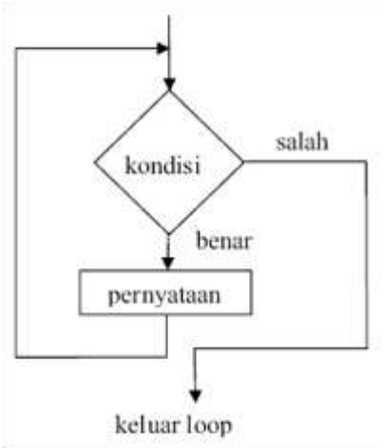
## 7.3. Uraian Materi

### A. Pengantar while

Pada pernyataan *while*, pengecekan terhadap *loop* dilakukan di bagian awal (sebelum tubuh *loop*). Lebih jelasnya, bentuk pernyataan *while* adalah sebagai berikut.

```
while (kondisi)
    pernyataan;
```

Pernyataan dapat berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Proses pengulangan terhadap pernyataan dijelaskan pada Gambar 5.2. Pada gambar tersebut, tampak bahwa ada kemungkinan pernyataan yang merupakan tubuh *loop* tidak dijalankan sama sekali, yaitu kalau hasil pengujian kondisi *while* yang pertama kali ternyata bernilai salah.




Gambar 7.2. Diagram alir *while*

Bagian pernyataan yang mengikuti **while** akan diEksekusi selama *ungkapan* pada **while** bernilai benar (tidak sama dengan nol). Pengujian terhadap ungkapan **while** dilakukan sebelum bagian pernyataan.

Contoh pemakaian *while* misalnya untuk mengatur agar tombol yang ditekan oleh pemakai program berupa salah satu diantara 'Y','y', 'T' atau 't'. Impelementasinya :

```
#include <stdio.h>
main()
{
char pilihan;
while(!sudah_benar)
{
pilihan = getchar(); /* baca tombol */
sudah_benar = (pilihan == 'Y') || (pilihan == 'y') || (pilihan ==
'T') ||  (pilihan == 't');
}
/* memberi keterangan tentang pilihan */
switch(pilihan)
{
case 'Y': case 'y':
puts("\nPilihan anda adalah Y"); break;
case 'T': case 't':
puts("\nPilihan anda adalah T");
    }
}
```

Hasil Eksekusi :

```
Pilihlah Y atau T
Pilihan anda adalah Y
```

Inisialisasi terhadap variabel sudah_benar yang akan dijalankan pada kondisi *while* dengan memberi nilai awal bernilai *false* (*sudah_benar = 0*) dimaksudkan agar tubuh *loop*. dijalankan minimal sekali.

```
{
pilihan = getchar( ); /* baca tombol */ sudah_benar = (pilihan ==
'Y') || (pilihan== 'y')|| (pilihan == 'T') || (pilihan == 't');
}
```

Contoh lain pemakaian *while* dapat dilihat pada program yang digunakan untuk menghitung banyaknya karakter dari kalimat yang dimasukkan melalui keyboard (termasuk karakter spasi). Untuk mengakhiri pemasukan kalimat, tombol ENTER (*'\n'*) harus ditekan. Karena itu, tombol ENTER inilah yang dijadikan kondisi penghitungan jumlah spasi maupun karakter seluruhnya. Lengkapnya, kondisi yang dipakai dalam *while* berupa :

```
while((kar = getchar()) != '\n')
```

Ungkapan di atas mempunyai arti :

- Bacalah sebuah karakter dan berikan ke variabel **kar**
- Kemudian bandingkan apakah karakter tersebut = *'\n'* (ENTER)

  Ungkapan menghasilkan nilai benar jika tombol yang ditekan bukan ENTER.

Pada program, kalau tombol yang ditekan bukan ENTER , maka :

- Jumlah karakter dinaikkan sebesar satu melalui pernyataan : *jumkar++;*
- Kalau karakter berupa SPASI, maka jumlah spasi dinaikkan sebesar satu, melalui pernyataan : if (kar == ' ') jumspasi++;

```
#include <stdio.h>
main()
{
char kar;
int jumkar = 0, jumspasi = 0;
puts("Masukkan sebuah kalimat dan akhiri dgn ENTER.\n");
puts("Saya akan menghitung jumlah karakter "); puts("pada kalimat
tersebut.\n");
while((kar = getchar()) != '\n')
{
jumkar++;
if (kar == ' ') jumspasi++;
}
printf("\nJumlah karakter  = %d", jumkar);
printf("\nJumlah SPASI     = %d\n\n", jumspasi);
}
```

Hasil Eksekusi :

```
Masukkan sebuah kalimat, akhiri dgn ENTER.
Saya akan menghitung jumlah karakter pada kalimat tersebut.


Belajar bahasa C sangat menyenangkan
```

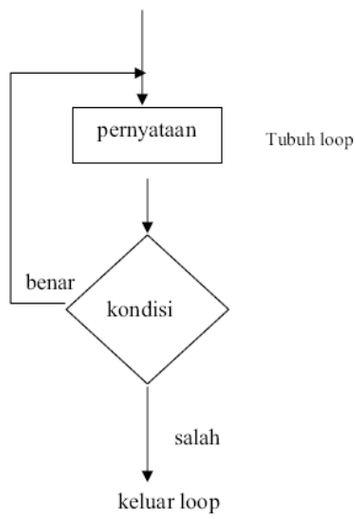
```
Jumlah karakter = 36
Jumlah SPASI = 4
```

## B. Pernyataan do-while

Bentuk pernyataan *do-while*

```
do
    pernyataan;
while (kondisi)
```

Pada pernyataan *do-while*, tubuh *loop* berupa pernyataan, dengan pernyataan bisa berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Pada pernyataan *do*, mula-mula pernyataan dijalankan. Selanjutnya, kondisi diuji. Sendainya kondisi bernilai benar, maka pernyataan dijalankan lagi, kemudian kondisi diperiksa kembali, dan seterusnya. Kalau kondisi bernilai salah pada saat dites, maka pernyataan tidak dijalankan lagi. Untuk lebih jelasnya dapat dilihat pada Gambar 7.3. Berdasarkan Gambar 7.3 terlihat bahwa tubuh *loop* minimal akan dijalankan sekali.




Gambar 7.3. Diagram alir do-while

Program berikut memberikan contoh pemakaian do-while untuk mengatur penampilan tulisan "BAHASA C" sebanyak sepuluh kali.

```
i = 0;
do
{
puts("BAHASA C");
i++;
} while(i<10);
```

Pada program di atas, variabel pencacah dipakai untuk menghitung jumlah tulisan yang sudah ditampilkan pada layar. Selama nilai pencacah kurang dari 10, maka perintah

```
puts("BAHASA C");
```

akan dilaksanakan kembali

Penanganan pembacaan tombol pada contoh program yang memakai *while* di atas, kalau diimplementasikan dengan memakai *do-while* dapat dibuat seperti berikut.

```
#include <stdio.h>
main()
{
char pilihan;
int sudah_benar;
printf("Pilihlah Y atau T.\n");
/* program dilanjutkan kalau tombol Y,y,T atau t ditekan */
do
{
pilihan = getchar( ); /* baca tombol */
sudah_benar = (pilihan == 'Y') || (pilihan== 'y')||(pilihan ==
'T') || (pilihan == 't');
} while(! sudah_benar);
/* memberi keterangan tentang pilihan */
switch(pilihan)
{
case 'Y':
case 'y':
puts("\nPilihan anda adalah Y");
break;
case 'T':
case 't':
puts("\nPilihan anda adalah T");
}
}
```

Mula-mula tombol dibaca dengan menggunakan *getchar()* dan kemudian diberikan ke variabel pilihan. Sesudah itu, variabel **sudah_benar** akan diisi dengan nilai benar (1) atau salah (0) tergantung dari nilai pilihan. Kalau pilihan berisi salah satu diantara ‘Y’,‘y’, ‘T’ atau ‘t’, maka sudah berisi salah satu diantara ‘Y’, ‘y’, ‘T’ atau ‘t’, maka **sudah_benar** akan berisi benar. Nilai pada vaiabel **sudah_benar** ini selanjutnya dijadikan sebagai kondisi *do-while*. Pengulangan terhadap pembacaan tombol akan dilakukan kembali selama **sudah_benar** benilai salah.

### 7.4. Latihan

1. Program perulangan while :

```
#include <stdio.h>

int main(void)
{
  int i = 10;
  while (i > 5){
printf("Hello World %i \n",i);
    i--;
  }
return 0;
}
```

Hasil Eksekusi :

```
Hello World 10
Hello World 9
Hello World 8
Hello World 7
Hello World 6
```

2. Program perulangan dengan do-while

```
#include <stdio.h>

int main(void)
{
  int i = 1;
  do {
printf("%i ",i*3);
    i++;
  }
  while (i <= 5);
return 0;
}
```

Hasil Eksekusi :

```
3   6 9 12 15
```

## 7.5. Tugas

1. Buatlah program untuk menampilkan bilangan kelipatan 5 antara 125 sampai dengan 200, menggunakan while dan do..while
2. Program untuk menghitung jumlah bilangan dari bilangan 1 sampai dengan N (N= Masukan dari user), while dan do..while

# BAB VIII
# FOR BERSARANG (NESTED FOR )

## 8.1. Capaian Pembelajaran

- Mengetahui penulisan perulangan bersarang (NESTED FOR).
- Mampu membuat sebuah pola tertentu ke dalam bahasa pemrograman.

## 8.2. Indikator

- Mampu membuat program menggunakan perulangan bersarang (NESTED FOR).

## 8.3. Uraian Materi

### A. Pengantar Nested FOR

Nested for adalah perulangan di dalam perulangan sehingga membentuk banyak perulangan yang harus di proses.

Syntax yang digunakan untuk nested for:

```
for(initiation; condition; increment){
      for(initiation; condition; increment);
      //statement of inside loop;
}
    //statement of outer loop;
}
```

Nested for bekerja dengan membaca for terluar, dilanjutkan for di dalamnya. Sebelum perulangan ke-2 terluar dimulai, diselesaikan terlebih dahulu for di dalam sesuai kondisi nilai akhirnya. Jika for di dalam sudah diEksekusi sesuai increment yang ada, maka perulangan for terluar dilanjutkan kembali. Struktur perulangan tersebut akan berlanjut terus-menerus sampai kondisi dari for terluar terpenuhi.

## 8.4. Latihan

Contoh program membuat dua kolom tiga baris angka :

```
#include <stdio.h>
int main(void) {
int i, j;
     for (i=1; i<=3; i++){  //loop terluar
         for (j=1; j<=3; j++){  //loop didalam
```

```
    printf("%i      %i\n",i,j);
            }
    printf("\n");
        }
}
```

Hasil Eksekusi:

```
1           1
1           2
1           3

2           1
2           2
2           3

3           1
3           2
3           3
(for luar)    (for dalam)
```

### 8.5. Tugas

1. Buatlah sebuah program menggunakan Bahasa C sehingga menampilkan segitiga sebagai berikut:

   1
   1 2
   1 2 3
   1 2 3 4
   1 2 3 4 5
   1 2 3 4 5 6

2. Buatlah sebuah program menggunakan Bahasa C sehingga menampilkan segitiga sebagai berikut:

   * * * * *
     * * * *
       * * *
         * *
           *

# BAB IX
# KOMBINASI SELEKSI DAN PERULANGAN

## 9.1. Capaian Pembelajaran

- Mengetahui penulisan program yang menggunakan kombinasi seleksi dan perulangan.
- Mampu menerapkan perhitungan berdasarkan sebuah kondisi, dengan keluaran hasil berdasarkan kondisi yang telah ditetapkan sebelumnya, ke dalam bahasa pemrograman.

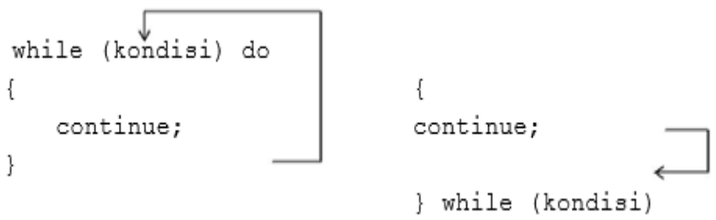
## 9.2. Indikator

- Mampu membuat program menggunakan kombinasi seleksi dan perulangan.

## 9.3. Uraian Materi

### A. Pengantar Continue

Pernyataan *continue* digunakan untuk mengarahkan Eksekusi ke iterasi (proses) berikutnya pada *loop* yang sama. Pada *do-while* dan *while*, pernyataan *continue* menyebabkan Eksekusi menuju ke kondisi pengujian pengulangan, seperti yang dilukiskan pada Gambar 5.5. Pada *loop for*, pernyataan *continue* menyebabkan bagian penaik variabel pengendali *loop* dikerjakan (ungkapan3 pada struktur *for*) dan kondisi untuk keluar dari *loop* for (ungkapan2 pada struktur *for*) diuji kembali.




Gambar 7.5 Pengaruh *continue* pada *while* dan *do-while*

Program ini dapat digunakan untuk memasukkan data yang harus diulangi dan hal ini dikendalikan dengan *continue*. Untuk mengakhiri pemasukan data, data yang dimasukkan harus bernilai 0. Perlu diketahui kondisi bernilai 1, menyatakan bahwa kondisi selalu dianggap benar. Untuk keluar dari *loop*, pernyataan yang digunakan berupa *break*.

Pengaruh *continue* pada *loop for* diperlihatkan pada program berikut ini. Program ini

dipakai untuk menampilkan bilangan ganjil yang terletak antara 7 sampai dengan 25, kecuali 15.

```
#include <stdio.h>
main()
{
int x;
for (x = 7; x <= 25; x += 2)
{
if (x == 15)
continue;
printf("%4d", x);
}
printf("\n");
}
```

Hasil Eksekusi :

```
9 11 13 17 19 21 23 25
```

Pada program di atas, untuk menghindari agar nilai 15 tidak ditampilkan ke layar, pernyataan yang digunakan berupa

```
if (x == 15)
continue;
```

Artinya, jika kondisi *x == 15* bernilai benar, pernyataan *continue* menyebabkan pernyataan sisanya yaitu

```
printf(“%d”,x);
```

diabaikan dan Eksekusi diarahkan kepada ungkapan :

```
x += 2
```

dan kemudian menguji kondisi :

```
x <= 25
```

Pada program di atas, pernyataan :

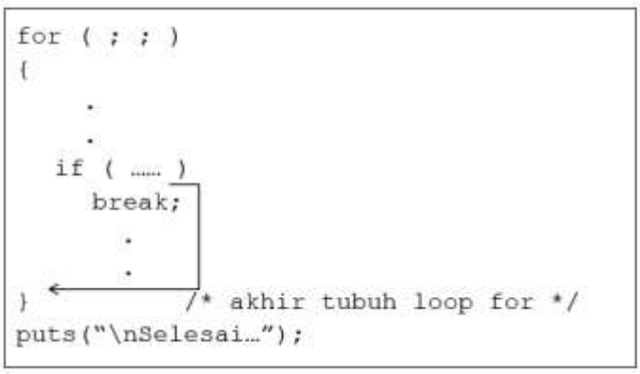
```
for (x = 7; x <= 25; x += 2)
{
if (x == 15)
continue;
printf("%4d", x);
}
```

dapat ditulis dalam bentuk lain sebagai berikut :

```
for (x = 7; x <= 25; x += 2)
if (x != 15)
printf("%4d", x);
```

**B. Pernyataan break**

Pernyataan break sesungguhnya telah diperkenalkan pada pernyataan switch. Pernyataan ini berfungsi untuk keluar dari loop for, do-while dan while. Sedangkan pada switch yaitu untuk menuju ke akhir (keluar dari) struktur switch. Sebagai contoh dapat dilihat pada gambar 5.4. Kalau pernyataan break dijalankan maka Eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir tubuh loop for.




Gambar 5.4 Ilustrasi pengaruh *break*

Pada contoh potongan program berikut, pembacaan dan penampilan terhadap tombol yang ditekan akan berakhir kalau tombol yang ditekan adalah ENTER (‘\n’). Pernyataan yang digunakan untuk keperluan ini :

```
if (kar == '\n')
break;    /* keluar dari loop for */
```

Potongan program ini menyatakan “Jika tombol yang ditekan berupa ENTER, maka keluarlah dari *loop for*”. Untuk lebih jelasnya, perhatikan program berikut.

Berikut Program pemakaian break untuk keluar dari looping.

```
#include <stdio.h>


main()
{
char kar;
printf("Ketik sembarang kalimat");
printf(" dan akhiri dengan ENTER\n\n");
```

```
for ( ; ; )
{
      kar = getchar(); if(kar == '\n') break;
}
printf("Selesai\n");
}
```

Hasil Eksekusi :

```
Ketik sembarang kalimat dan akhiri dengan ENTER :
Menulis apa saja
Selesai
```

Jika pernyataan *break* berada dalam *loop* yang bertingkat (*nested loop*), maka pernyataan *break* hanya akan membuat proses keluar dari *loop* yang bersangkutan (tempat *break* dituliskan), bukan keluar dari semua *loop*.

## C. Loncatan Goto

Pernyataan *goto* merupakan intruksi untuk mengarahkan Eksekusi ke pernyataan yang diawali dengan suatu label. Label sendiri berupa suatu pengenal (*identifier*) yang diikuti dengan tanda titik dua (:)

Contoh pemakaian *goto* ditujukan pada program dibawah ini:

```
Pernyataan
goto cetak;
```

Mengisyaratkan agar Eksekusi dilanjutkan ke pernyataan yang diawali dengan label

```
cetak:
Pernyataan
if (++pencacah <= 10)
goto cetak;
```

Mempunyai arti :

- Naikkan nilai **pencacah** sebesar 1
- Kemudian, jika **pencacah** kurang dari atau sama dengan 10 maka Eksekusi menuju ke label **cetak**.

Penerapan *goto* biasanya dilakukan pada *loop* di dalam *loop* (*nested loop*), dengan tujuan memudahkan untuk keluar dari *loop* terdalam menuju ke pernyataan yang terletak di luar *loop* terluar.

### D. Menghentikan Program dengan exit()

Suatu Eksekusi program dapat dihentikan (secara normal) melalui pemanggilan fungsi *exit()*. Hal ini biasa dilakukan, jika di dalam suatu Eksekusi terdapat suatu kondisi yang tak dikehendaki. Prototipe dari fungsi *exit()* didefinisikan pada file **stdlib.h**, yang memiliki deklarasi sebagai berikut :

```
void exit(int status);
```

Menurut kebiasaan, nilai nol diberikan pada argumen *exit()* untuk menunjukkan penghentian program yang normal. Sedangkan untuk menunjukkan kesalahan, nilai yang diberikan pada argumen fungsi diisi dengan nilai bukan-nol. Pada contoh program berikut, Eksekusi program akan dihentikan hanya jika tombol *'X'* ditekan.

## 9.4. Latihan

1. Contoh program Goto

```
#include <stdio.h>
main(){
int i=2;
cetak:
 printf("%d\t",i);
loop:
 i++;
if(i==2 || i==3 || i==5 || i==7)
 goto cetak;
else if(i%2 && i%3 && i%5 && i%7)
 goto cetak;
else if(i < 100 )
 goto loop;
}
```

2. Contoh program exit:

```
#include <stdio.h> #include <stdlib.h>
main()
{
    char kar;
printf("Tekanlah X untuk menghentikan program.\n");
for ( ; ;)
{
while ((kar = getchar()) == 'X') exit(0);
}
}
```

## 9.5. Tugas

1. Buatlah program untuk suatu kondisi sebagai berikut :

   Jika nilai = 81 - 90,  nilai = A, keterangan = lulus

   Jika nilai = 61 – 80, nilai = B, keterangan = lulus

   Jika nilai = 41 – 60, nilai = C, keterangan = lulus

   Jika nilai = 21 – 40, nilai = D, keterangan = ulang

   Jika nilai = 5 – 20, nilai = E, keterangan = ulang

# BAB X
# ARRAY SATU DIMENSI

## 10.1. Capaian Pembelajaran

- Mengetahui dasar-dasar array 1 dimesi.
- Mengetahui penulisan array 1 dimensi.
- Mengetahui penggunaan array 1 dimensi untuk aplikasi.
- Mampu menerapkan perhitungan berat badan ideal dan menetapkan suhu ke dalam bahasa pemrograman.

## 10.2. Indikator

- Mampu menjelaskan dasar-dasar array 1 dimensi.
- Mampu menuliskan penulisan array 1 dimensi.
- Mampu menggunakan array 1 dimensi untuk aplikasi.

## 10.3. Uraian Materi

### A. Pengantar Array satu dimensi

Array atau bisa disebut sebagai larik adalah koleksi data atau kumpulan data yang memiliki nama variable dan jenis tipe data yang sama dan memiliki index. Index tersebut digunakan untuk mengakses (baca/tulis) elemen, atau isi array tersebut.
Array satu dimensi adalah sekelompok data yang memiliki nama variable dan tipe data yang sama yang dapat diakses menggunakan 1 buah index saja.

Pada dunia pemrograman array sangat dibutuhkan sekali dengan tujuan untuk mempersingkat dan mempermudah proses penulisan kode program yang melibatkan banyak sekali variable yang memiliki tipe data yang sama. Adanya array, variable bisa ditulis lebih sederhana jika dibandingkan tanpa menggunakan array.

Misalnya seorang pegawai diminta untuk membuat sebuah program yang dapat menghitung angka yang diinput oleh pengguna sebanyak 50 buah, jika tanpa menggunakan array, maka terpaksa pegawai tersebut harus menyediakan variable sebanyak 50 buah, dengan nama variable berbeda untuk menampung input pengguna. Sedangkan jika menggunakan array, penulisan variable tersebut dapat dibuat lebih singkat dan praktis hanya cukup satu baris saja dengan satu nama variable dan tipe data

yang sama, yang memiliki index sebanyak 50, sehingga penulisan program lebih cepat, lebih sederhana, memiliki sedikit baris dan lebih mudah dalam pembacaan.
Bentuk umum array:

tipe_data nama_array [jumlah_elemen];

Sebagai contoh sebuah pendeklarasian array satu dimensi (misal dengan nama array LARIK) yang mempunyai 10 elemen dengan tipe data int, maka penulisan arraynya sebagai berikut:
int LARIK [10];

Jika array LARIK tersebut akan diberi nilai saat pendeklarasian (inisialisasi), maka contoh penulisannya sebagai berikut:
int LARIK [5] = {1, 6, 2, 4, 3};

Pendeklarasian sekaligus inisialisasi array LARIK di atas, dapat diartikan bahwa kita telah memesan tempat pada memori komputer sebanyak 5 tempat, dengan indeks dari 0 sampai 4. Nilai-nilai akan dimasukan dalam elemen array secara berturut-turut, mulai dari indeks 0 akan di isi dengan nilai '1' sampai indeks 4 yang di isi dengan nilai '3'. Lebih jelasnya perhatikan table di bawah ini:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 6 | 2 | 4 | 3 | Nilai Elemen Array |
| Alamat memory ke-1 | Alamat memory ke-2 | Alamat memory ke-3 | Alamat memory ke-4 | Alamat memory ke-5 | Alamat Elemen Array |
| 0 | 1 | 2 | 3 | 4 | Index Elemen Array |

Contoh program sederhana array satu dimensi

```
#include <stdio.h>
main(void) {
char huruf[5] = {'a', 'b', 'c', 'd', 'e'};
printf("Huruf: %c\n", huruf[3]);
}
```

Hasil Eksekusi:

```
Huruf: d
```

Program ini menampilkan bahwa index elemen ke-3 adalah huruf d. meminta pemakai untuk memasukkan 5 buah data temperatur dari keyboard. Kelima data tersebut disimpan pada array bernam suhu. Selanjutnya data yang ada pada array tersebut ditampilkan ke layar.

### 10.4. Latihan

1. Contoh program sederhana array satu dimensi

```
#include <stdio.h>
int main(){
// membuat array kosong
int i, nilai[5];

// mengisi array
nilai[0] = 27;
nilai[1] = 32;
nilai[2] = 93;
nilai[3] = 87;
nilai[4] = 20;

// mencetak isi array dengan perulangan
for(i=0; i < 5; i++){
    printf("Nilai: %d\n", nilai[i]);
}
}
```

Hasil Eksekusi:

```
Nilai: 27
Nilai: 32
Nilai: 93
Nilai: 87
Nilai: 20
```

## 10.5. Tugas

1. Buatlah sebuah program mencari berat badan rata-rata dengan mengunakan array satu dimensi. Data diisi oleh 10 data berat badan wanita
2. Berdasarkan program di bawah ini:

```
#include <stdio.h>
int main() {
// Array dengan elemen 5 bertipe float
float suhu[5];
int i;

// Membaca data dari keyboard dan meletakkan array
printf("Masukkan 5 buah data suhu: \n");
    for (i=0; i<5; i ++){
printf(……);
scanf(……);
}
```

// Menampilkan isi array ke layar

```
printf("Data suhu yang anda masukkan: \n");
        for ( i=0; i<5; i ++ )
        scanf(……);
}
```

Program di atas dimaksudkan memiliki pernyataan demikian:

- **float suhu[5];** menyatakan array **suhu** dapat menyimpan 5 (lima) buah data bertipe **float**.
- **suhu[i]** menyatakan elemen suhu dengan *subscript* sama dengan **i**.
- **scanf suhu[i];** membaca data dari keyboard dan meletakkan ke elemen nomor **i** pada array **suhu**.
- **printf suhu[i];** akan menampilkan elemen bernomor i pada array **suhu**.
-

Lengkapi program tersebut sehingga memiliki hasil Eksekusi sebagai berikut:

```
Masukkan 5 buah data suhu :
1 : 30.5
2 : 41
3 : 28.5
4 : 23.2
5 : 32
```

Data suhu yang anda masukkan :

30.5

41

28.5

23.3

32

# BAB XI
# ARRAY DUA DIMENSI

## 11.1. Capaian Pembelajaran

- Mengetahui dasar-dasar array 2 dimesi.
- Mengetahui penulisan array 2 dimensi.
- Mengetahui penggunaan array 2 dimensi untuk aplikasi.
- Mampu mengimplementasikan data jumlah mahasiswa per tahun ke dalam bahasa pemrograman.

## 11.2. Indikator

- Mampu menjelaskan dasar-dasar array 2 dimensi.
- Mampu menuliskan penulisan array 2 dimensi.
- Mampu menggunakan array 2 dimensi untuk aplikasi.

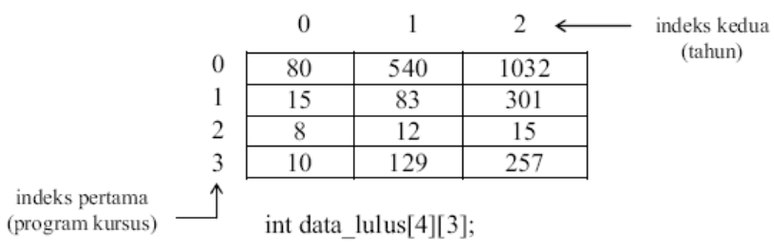
## 11.3. Uraian Materi

### A. Array Berdimensi Dua

Pendeklarasian yang diperlukan untuk menyimpan data kelulusan siswa pada Gambar 7.2 adalah:

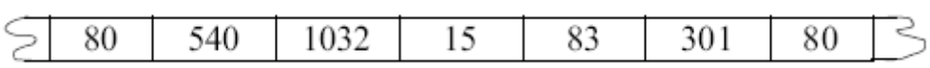
int data_lulus[4][3];

Nilai 3 untuk menyatakan banyaknya tahun dan 4 menyatakan banyaknya program kursus. Gambar 7.2 memberikan ilustrasi untuk memudahkan pemahaman tentang array berdimensi dua.




Gambar 7.2 Array berdimensi dua

Sama halnya pada array berdimensi satu, data array aka ditempatkan pada memori yang berurutan. Perhatikan Gambar 6.3.



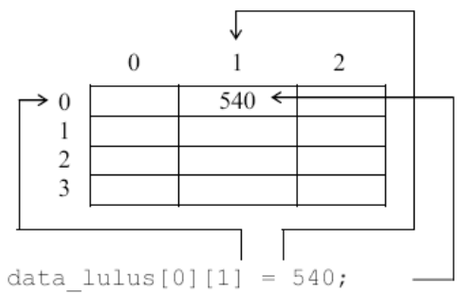

Gambar 7.3 Model penyimpanan array dimensi dua pada memori

## B. Mengakses Elemen Array Berdimensi Dua

Array seperti **data_lulus** dapat diakses dalam bentuk

**data_lulus[indeks pertama, indeks kedua]**

Contoh :




Gambar 7.4. Pemberian nilai ke array berdimensi dua

(1) data_lulus[0][1] = 540;

merupakan instruksi untuk memberikan nilai 540 ke array **data_lulus** untuk indeks pertama = 0 dan indeks kedua bernilai 1.

(2) printf("%d",data_lulus[2][0]);

merupakan perintah untuk menampilkan elemen yang memiliki indeks pertama = 2 dan indeks kedua = 0.

## C. Inisialisasi Array Berdimensi Dua

Gambar berikut memberikan penjelasan tentang inisialisasi yang dilakukan terhadap array berdimensi dua :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Gambar 7.5 Representasi inisialisasi array berdimensi dua

Untuk itu, deklarasi dan inisialisasi yang dilakukan berupa :

```
int huruf_A[8][8] = {
        { 0, 1, 1, 1, 1, 1, 0, 0 } ,
        { 0, 1, 0, 0, 0, 1, 0, 0 } ,
        { 0, 1, 0, 0, 0, 1, 0, 0 } ,
```

```
{ 1, 1, 1, 1, 1, 1, 1, 0 } ,
{ 1, 1, 0, 0, 0, 0, 1, 0 } ,
{ 1, 1, 0, 0, 0, 0, 1, 0 } ,
{ 1, 1, 0, 0, 0, 0, 1, 0 } ,
{ 0, 0, 0, 0, 0, 0, 0, 0 }
};
```

atau bisa juga ditulis sebagai berikut :

```
int huruf_A[8][8] =
    { 0, 1, 1, 1, 1, 1, 0, 0,
      0, 1, 0, 0, 0, 1, 0, 0,
      0, 1, 0, 0, 0, 1, 0, 0,
      1, 1, 1, 1, 1, 1, 1, 0,
      1, 1, 0, 0, 0, 0, 1, 0,
      1, 1, 0, 0, 0, 0, 1, 0,
      1, 1, 0, 0, 0, 0, 1, 0,
      0, 0, 0, 0, 0, 0, 0, 0
    };
```

### 11.4. Latihan

1. Program data kelulusan

```
#include <stdio.h>
main( )
{
int tahun, kode_program;
int data_lulus[4][3] ;

/* Memberikan data ke array */
data_lulus[0][0] = 80;
data_lulus[0][1] = 540;
data_lulus[0][2] = 1032;
data_lulus[1][0] = 15;
data_lulus[1][1] = 83;
data_lulus[1][2] = 301;
data_lulus[2][0] = 8;
data_lulus[2][1] = 12;
data_lulus[2][2] = 15;
data_lulus[3][0] = 10;
data_lulus[3][1] = 129;
```

```
data_lulus[3][2] = 257;
/* proses utk memperoleh informasi jml siswa yg lulus */
printf("Masukkan tahun dr data yg ingin anda ketahui ");
printf("(1998..2000) : ");
scanf("%d", &tahun);
printf("Masukkan kode program kursus yang ingin anda ketahui");
printf("(1 = INTRO, 2 = BASIC, 3 = PASCAL, 4 = C) : ");
scanf("%d", &kode_program);
printf("\nTotal kelulusan program tsb = %d\n",
data_lulus[kode_program - 1][tahun - 1998] );
}
```

2. Program Huruf "A"

```
#include <stdio.h>
main(){
int i,j;
int huruf_A[8][8] = {
{ 0, 1, 1, 1, 1, 1, 0, 0 } ,
{ 0, 1, 0, 0, 0, 1, 0, 0 } ,
{ 0, 1, 0, 0, 0, 1, 0, 0 } ,
{ 1, 1, 1, 1, 1, 1, 1, 0 } ,
{ 1, 1, 0, 0, 0, 0, 1, 0 } ,
{ 1, 1, 0, 0, 0, 0, 1, 0 } ,
{ 1, 1, 0, 0, 0, 0, 1, 0 } ,
{ 0, 0, 0, 0, 0, 0, 0, 0 }
};
for(i = 0; i < 8; i++)
{
for(j = 0; j < 8; j++)
if(huruf_A[i][j] !=0 )
putchar('\xDB\n');
else
putchar (' '); /* spasi */
putchar('\n');
}
}
```

## 11.5. Tugas

1. Diketahui data kelulusan mahasiswa sebagai berikut :

| Tahun Prodi | Teknik Komputer | Teknik Elektro | Rata-rata kelulusan per tahun semua prodi |
|---|---|---|---|
| 2015 | 40 | 60 | 50 |
| 2016 | 50 | 90 | 70 |
| Rata-rata pertahun per prodi | 45 | 75 | |

Buatlah program yang bisa menampilkan data dan informasi seperti diatas.

# BAB XII
# FUNGSI DENGAN DAN TANPA NILAI BALIK

## 12.1. Capaian Pembelajaran

- Mengetahui fungsi dengan dan tanpa nilai balik.
- Mengetahui penulisan fungsi dengan dan tanpa nilai balik.
- Mengetahui penggunaan dan manfaat fungsi dengan dan tanpa nilai balik untuk aplikasi.
- Mampu menampilkan posisi data tertentu dalam sebuah deret data ke dalam bahasa pemrograman.
- Mampu meningkatkan kreatifitas dalam membuat sebuah program dengan kondisi tertentu sesuai kriteria.

## 12.2. Indikator

- Mampu menjelaskan fungsi dengan dan tanpa nilai balik.
- Mampu menuliskan program menggunakan fungsi dengan dan tanpa nilai balik.
- Mampu menggunakan fungsi dengan dan tanpa nilai balik untuk aplikasi.

## 12.3. Uraian Materi 1 – Dengan Nilai Balik

### A. Pengantar Fungsi dengan Nilai Balik

Berbeda dengan fungsi bertipe void, fungsi ini berguna untuk melakukan suatu proses yang dapat mengembalikan sebuah nilai. Dalam membuat fungsi ini kita harus mendefisinikan tipe data dari nilai yang akan dikembalikan.

```
Contoh definisi kuadrat() :
// Prototipe funsi
long kuadrat (long 1);
-------------------------------------
// Definisi fungsi
long kuadrat(long 1)
{
   return(1 * 1);
}
```

Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya. Elemen utama dari

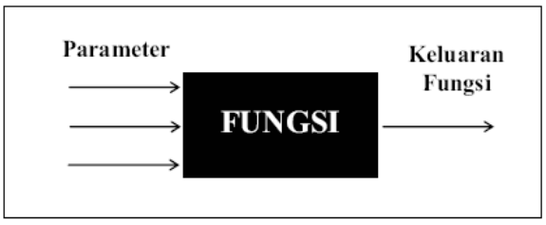
program bahasa C berupa fungsi-fungsi, dalam hal ini program dari bahasa C dibentuk dari kumpulan fungsi pustaka (standar) dan fungsi yang dibuat sendiri oleh pemrogram. Fungsi banyak digunakan pada program C dengan tujuan :

a. Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Dengan memisahkan langkah-langkah detail ke satu atau lebih fungsi-fungsi, maka fungsi utama (*main()*) menjadi lebih pendek, jelas dan mudah dimengerti.
b. dapat mengurangi pengulangan (duplikasi) kode. Langkah-langkah program yang sama dan dipakai berulang-ulang di program dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi-fungsi. Selanjutnya bagian program yang membutuhkan langkah-langkah ini tidak perlu selalu menuliskannya, tetapi cukup memanggil fungsi-fungsi tersebut.

Fungsi standar C yang mengemban tugas khusus contohnya adalah ;

- *printf()* , yaitu untuk menampilkan informasi atau data ke layar.
- *scanf()* , yaitu untuk membaca kode tombol yang diinputkan.

Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai argumen. Nilai masukan ini akan diolah oleh fungsi. Hasil akhir fungsi berupa sebuah nilai (disebut sebagai *return value* atau nilai keluaran fungsi). Oleh karena itu fungsi sering digambarkan sebagai "kotak gelap" seperti ditunjukkan pada Gambar 6.1




Gambar 6.1 Fungsi sebagai sebuah kotak gelap

Penggambaran sebagai kotak gelap di antaranya menjelaskan bahwa bagian dalam fungsi bersifat pribadi bagi fungsi. Tak ada suatu pernyataan di luar fungsi yang bisa mengakses bagian dalam fungsi, selain melalui parameter (atau variabel eksternal yang akan dibahas belakangan). Misalnya melakukan *goto* dari pernyataan di luar fungsi ke pernyataan dalam fungsi adalah tidak diperkenankan.

Bentuk umum dari definisi sebuah fungsi adalah sebagai berikut ;

```
tipe-keluaran-fungsi nama-fungsi (deklarasi argumen)
{
      tubuh fungsi
}
```

Keterangan :

- **tipe-keluaran-fungsi**, dapat berupa salah satu tipe data C, misalnya *char* atau *int*. Kalau penentu tipe tidak disebutkan maka dianggap bertipe *int* (secara *default*).
- **tubuh fungsi** berisi deklarasi variabel (kalau ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan. Tubuh fungsi ini ditulis di dalam tanda kurung kurawal buka dan kurung kurawal tutup.

Sebuah fungsi yang sederhana bisa saja tidak mengandung parameter sama sekali dan tentu saja untuk keadaan ini deklarasi parameter juga tidak ada.

Contoh ;

```
int inisialisasi()
{
return(0);
}
inisialisasi()
{
return(0);
}
```

Pernyataan **return** di dalam fungsi digunakan untuk memeberikan nilai balik fungsi

## 12.4. Latihan

1. Program membuat fungsi tambah

```
#include "stdio.h"

/* prototype fungsi tambah(), ada titik koma */
float tambah(float x, float y);

main(){
float a, b, c;
printf("A = "); scanf("%f", &a);
printf("B = "); scanf("%f", &b);
c = tambah(a, b); /* pemanggilan fungsi tambah() */
printf("A + B = %.2f", c);
}

/* Definisi fungsi , tanpa titik koma */
float tambah(float x, float y){
    return (x+y); /* Nilai balik fungsi */
}
```

2. Program penjumlahan 3 variabel

```
#include <stdio.h>

int hasilTambah(int x,int y, int z){
 int penjumlahan;
 penjumlahan = x + y + z;
 return penjumlahan;
}

int hasilKurang(int x,int y, int z){
 int pengurangan;
 pengurangan = x - y - z;
 return pengurangan;
}

main(){

 int a,b,c,hasil;

 a = 1;  b = 2;  c = 3;

 hasil = hasilTambah(a, b, c);
 printf("Hasil penjumlahan : %d\n", hasil);

 hasil = hasilKurang(a, b, c);
 printf("Hasil pengurangan : %d\n", hasil);

}
```

## 12.5. Tugas

1. Diketahui deklarasi fungsi sebagai berikut :

```
int luas (int a, int b)
{
   return (0.5*a*t);
}
```

   Buatlah program utama yang menggunakan fungsi diatas.

2. Buat program untuk pemasukkan data disertai keterangan data ke-.

   Contoh :

   Banyak data = 3

   Masukkan Nilai ke-1 : 28 ↵

Masukkan Nilai ke-2 : 11 ↵

Masukkan Nilai ke-3 : 1982 ↵

Data ke-1 = 28

Data ke-2 = 11

Data ke-3 = 1982

## 12.6. Uraian Materi 2 – Tanpa Nilai Balik

### A. Pengantar Fungsi Tanpa Nilai Balik

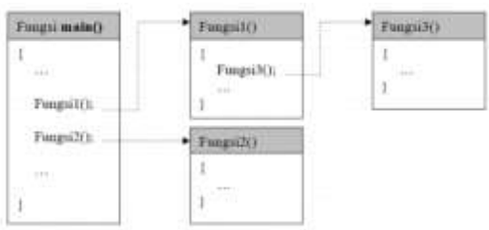
Fungsi adalah sub-program yang bisa digunakan kembali baik di dalam program itu sendiri, maupun di program yang lain, yang bertujuan:

1. Program jadi lebih struktur
2. Menghemat ukuran program
3. Memudahkan pembuatan program

Cara membuat fungsi sebagai berikut:

```
void nama_fungsi(parameter1, parameter2,…) {
  Statemen_yang_akan_dieksekusi;
  ...
}
```

Cara pemanggilan fungsi sebagai berikut:




Fungsi biasanya akan mengembalikan sebuah nilai dari hasil prosesnya, dengan demikian, harus ditentukan tipe data untuk nilai yang akan dikembalikan.
Apabila fungsi tersebut tidak memiliki nilai kembalian, maka kita harus menggunakan tipe “void” untuk menyatakan kalau fungsi tersebut tidak akan mengembalikan nilai apa-apa.

Contoh penulisan fungsi:

```
void nama_fungsi(){
    printf("Saya sedang belajar fungsi tanpa nilai balik\n");
}
```

## 12.7. Latihan

1. Contoh program tanpa nilai balik

```
#include <stdio.h>

// membuat fungsi say_hello()
void say_hello(){
printf("Hello saya sedang belajar fungsi tanpa nilai balik)\n");
}

void main(){
// memanggil fungsi say_hello()
say_hello();
}
```

Hasil Eksekusi:

```
Hello saya sedang belajar fungsi tanpa nilai balik
```

2. Contoh program tulis 10 kali

```
#include <stdio.h>
// Mendefinisikan sebuah fungsi dengan nama Tulis10Kali
void Tulis10Kali(){
int j;
 for (j=0; j<10; j++) {
 printf("Saya sedang belajar fungsi dalam bahasa C\n");
 }
}
int main(void){
Tulis10Kali();
return 0;
}
```

Hasil Eksekusi:

```
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
```

```
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
Saya sedang belajar fungsi dalam bahasa C
```

## 12.8. Tugas

1. Buatlah sebuah program yang di dalamnya melibatkan penggunaan perulangan dan fungsi tanpa nilai balik. Buat program sekreatifitas mungkin.

# BAB XIII
# FUNGSI DENGAN PARAMETER

## 13.1. Capaian Pembelajaran

- Mengetahui fungsi dengan parameter.
- Mengetahui penulisan fungsi dengan parameter.
- Mengetahui penggunaan dan manfaat fungsi dengan parameter untuk aplikasi.
- Mampu menerapkan perhitungan rumus fisika dengan parameter ke dalam bahasa pemrograman.

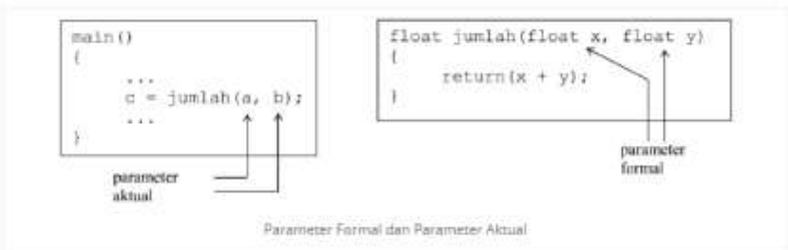
## 13.2. Indikator

- Mampu menjelaskan fungsi dengan parameter.
- Mampu menuliskan program menggunakan fungsi dengan parameter.
- Mampu menggunakan fungsi dengan parameter untuk aplikasi.

## 13.3. Uraian Materi

### A. Pengantar fungsi dengan parameter

Parameter adalah suatu variabel yang menyertai fungi pada saat deklarasi maupun saat fungsi dipanggil. Terdapat dua parameter, yaitu parameter aktual dan parameter formal.



Parameter Formal dan Parameter Aktual

### B. Parameter Aktual

Parameter aktual adalah parameter yang berupa variabel atau konstanta yang dipakai saat suatu fungsi dipanggil. Contoh:

1. **namavariabel-fungsi(variabel)**
   hitung_luas_persegi_panjang(panjang,lebar)
2. **namavariabel-fungsi(konstanta)**
   hitung_luas_lingkaran(9.3)

### C. Parameter Formal

Parameter formal adalah parameter yang hanya berupa variabel saat ketika suatu fungsi dideklarasikan. Dalam parameter formal kita bisa menggunakan satu ataupun banyak variabel di dalamnya yang mewakili suatu fungsi tersebut. Contoh:

**namavariabel-fungsi(variabel)**

hitung_luas_persegi_panjang(float panjang, float lebar)

## 13.4. Latihan

1. Contoh program menggunakan paramater

```
void add(int a, int b){
 printf("%d * %d = %d\n", a, b, a*b);
}

void main(){
    add(4, 5);
    add(8, 3);
    add(2, 2);
}
```

Hasil Eksekusi :

```
4 * 5 = 20
8 * 3 = 24
2 * 2 = 4
```

2. Contoh program luas persegi dan keliling lingkaran

```
#include <stdio.h>
 //parameter formal berupa variabel: panjang, lebar dan diameter
 int hitung_luas_persegi_panjang(int panjang, int lebar);
 float hitung_keliling_lingkaran(float diameter);

 int main () {
  int panjang = 20;
  int lebar = 4;
  int luas;
  float keliling;

  //parameter aktual berupa variabel
  luas = hitung_luas_persegi_panjang(panjang,lebar);
  printf("Luas persegi panjang = %d \n",luas);
```

```
 //parameter aktual berupa konstanta
 keliling = hitung_keliling_lingkaran(9.5);
 printf("Keliling lingkaran = %f \n",keliling);
 return 0;
}

//definisi fungsi
//parameter formal berupa variabel: panjang, lebar dan diameter
int hitung_luas_persegi_panjang(int panjang, int lebar)
{
 return (panjang * lebar);
}

float hitung_keliling_lingkaran(float diameter)
{
 return (2 * 3.14 * (diameter/2));
}
```

Hasil Eksekusi :

```
Luas persegi panjang = 80
Keliling lingkaran = 29.830000
```

### 13.5. Tugas

1. Buat sebuah program yang melibatkan fungsi dengan parameter. Program tersebut berisi tentang salah satu rumus fisika (rumus fisika bebas). Uraikan jawaban anda
2. Buatlah sebuah program yang di dalamnya melibatkan If, For dan Fungsi dengan Parameter.

# MODUL XIV
# CHALLENGES

## 14.1. Capaian Pembelajaran

- Mampu membuat sebuah program berdasarkan seluruh materi yang telah dijelaskan dalam modul ini, penerapan diambil dari kasus dalam dunia nyata.

## 14.2. Indikator

- Mampu menyelesaikan challenges yang diberikan.

## 14.3. Latihan

**Buatlah sebuah sistem ATM sederhana yang mengimplementasikan beberapa menu seperti berikut:**

**1. Input Nasabah**

Input :

- Rekening
- Nama
- Alamat
- Jumlah Saldo Awal

Masukan No. Rekening "WAJIB" 13 digit. kalau tidak 13, tolak pendaftaran.
Minimal saldo awal 100.000 , kurang dari 100.000 tolak pendaftaran

**2. Deposit / Penyetoran**

Input :

- Rekening
- Jumlah Saldo Setoran

Minimal deposit 100.000 keatas, kurang dari 100 rb auto tolak.

**3. Cetak Daftar Nasabah (seluruh data yang sudah diinputkan)**

Output :

- Rekening
- Nama

- Alamat
- Total Saldo (hasil seluruh kalkulasi Saldo awal dan setoran ditambah bunga 15%)

**4. Keluar**

| | | |
|---|---|---|
| **RESPONSI - 1** | Nama | |
| | Nim | |

| | | | |
|---|---|---|---|
| Diperiksa Tanggal | | Paraf Dosen/Asisten | |

| | | |
|---|---|---|
| **RESPONSI - 2** | Nama | |
| | Nim | |
| | | |
| Diperiksa Tanggal | | Paraf Dosen/Asisten | |