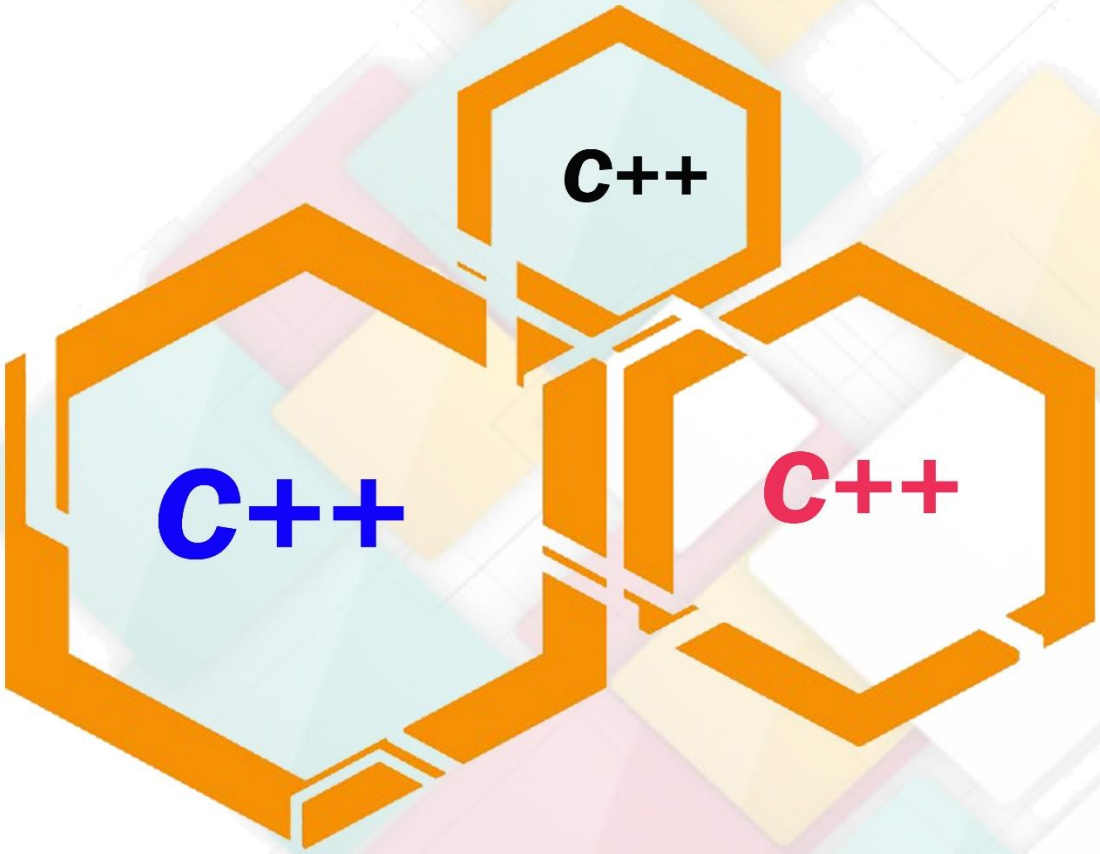
CV. KASIH INOVASI TEKNOLOGI

GENERATION
TEKNIK INFORMATIKA

Konsep Dasar
ALGORITMA PEMROGRAMAN
Menggunakan C++



Chapter#1

Patmi Kasih
Umi Mahdiyah

CV. KASIH INOVASI TEKNOLOGI

Konsep Dasar Algoritma Pemrograman Menggunakan C++

Chapter#1

Patmi Kasih
Umi Mahdiyah

Konsep Dasar Algoritma Pemrograman Menggunakan C++ Chapter#1

Oleh :
Patmi Kasih
Umi Mahdiyah
ISBN : 978-602-51918-4-8

Editor
Danar Putra Pamungkas
Penyunting
Julian Sahertian
Desain Cover
Ahmad Bagus Setiawan

CV. KASIH INOVASI TEKNOLOGI

Jl. KH. Hasyim Asyari Gg.1 Nusa Indah No.74,
Kota Kediri
Telp. +628563533234
Email : kasihinovasiteknologi@gmail.com

Cetakan Pertama, Oktober 2018

Kata Pengantar

Syukur Alhamdulillah kehadirat Allah SWT atas karunia dan berkah-Nya sehingga penulis dapat menyelesaikan penyusunan buku Konsep Dasar Algoritma Pemrograman Menggunakan C++ Chapter#1 ini. Tekad untuk memberikan bahan belajar yang mudah untuk dipahami bagi mahasiswa tingkat I jurusan Teknik informatika di Universitas Nusantara PGRI Kediri, membuat penulis bersemangat untuk menyelesaikan buku Konsep Dasar Algoritma Pemrograman Menggunakan C++ Chapter#1 ini. Pembelajaran Algoritma dan Pemrograman sangatlah penting bagi jurusan teknik informatika, karena sebagian besar mata kuliah dalam jurusan teknik informatika membutuhkan pemahaman algoritma yang kuat, oleh karena itu kemampuan dasar yang kuat mengenai algoritma dan pemrograman sangat dibutuhkan.

Sebagai mata kuliah dasar dan inti, dalam penyampaiannya diperlukan cara yang mudah diterima dan dipahami oleh mahasiswa yang notabene berasal dari latar belakang jurusan sekolah yang beragam. Bahasa C dipilih sebagai bahasa bantu dalam mata kuliah Algoritma dan Pemrograman karena bahasa C merupakan bahasa pemrograman dasar yang terstruktur dan mendukung pemrograman berorientasi objeck. Bahasa C merupakan bahasa pemrograman yang sudah tidak diragukan lagi kehandalannya dan banyak digunakan untuk membuat program-program dalam berbagai bidang, termasuk pembuatan compiler dan sistem operasi. Sampai saat ini, C masih tetap menjadi bahasa populer dan berwibawa dalam kancah pemrograman. Bahasa C telah menjadi inspirasi bagi kelahiran bahasa-bahasa pemrograman baru, Java, dan juga yang lainnya. Bahasa C sehingga dari sisi sintak control programnya, ketiga bahasa ini bisa dikatakan sama. Bahasa pemrograman C sangatlah fleksibel dan portabel, sehingga dapat ditempatkan dan dijalankan di dalam beragam sistem operasi. Pada umumnya, C banyak digunakan untuk melakukan interfacing antar perangkat keras (*hardware*) agar dapat berkomunikasi satu sama lainnya.

Dalam buku dan modul pembelajaran ini dibahas konsep-konsep dasar penyusunan algoritma dan juga konsep dasar yang mutlak diperlukan dalam pemrograman menggunakan bahasa C sehingga dapat mengimplementasikannya langsung ke dalam kasus-kasus pemrograman yang dihadapi. Dengan penyajian contoh program yang cukup banyak dalam setiap bab dan sub bab diharapkan bisa banyak membantu memberikan banyak isnpirasi bagi mahasiswa dalam mempelajari pemrograman C secara mudah dan cepat.

Terakhir, penulis menyampaikan banyak terima kepada kepada Pak Ahmad Bagus Setiawan beserta rekan-rekan Tim Generation UN PGRI Kediri yang telah mempercayai

penulis untuk menyusun modul pembelajaran ini dan telah banyak membantu dalam proses pencetakan dan penerbitan modul pembelajaran ini. Terima kasih sepenuh hati kepada Suami dan Putraku tercinta atas cinta dan dukungannya. Semoga Apa yang kita niatkan dengan kebaikan membawa keberkahan bagi kita semua…aamiin.

Kediri, September 2018

Penulis

Daftar Isi

BAB 1
ALGORITMA & PEMROGRAMAN

1.1 Algoritma

1.1.1 Konsep Algoritma

Algoritma berasal dari kata *algoris* dan *ritmis* yang pertama kali diungkapkan oleh Abu Ja'far Mohammad Ibn Musa Al Khowarizmi (825M) dalam buku Al-Jabr Wa-al Muqobla. Dalam pemrograman algortima berarti suatu metode khusus yang tepat dan terdiri dari serangkaian langkah-langkah yang terstruktur dan dituliskan secara sistematis yang akan dikerjakan untuk menyelesaikan masalah dengan bantuan komputer.Secara sederhanya algoritma adalah langkah-langkah, urutan langkah-langkah, atau tahap-tahap. Algoritma merupakan urutan langkah-langkah *logis* yang disusun secara sistematis untuk menyelesaikan suatu masalah. Kata logis (logika) merupakan kunci dan syarat utama dalam algoritma, dimana langkah-langkah yang disusun dalam algoritma harus logis (masuk akal), dan dapat di nalar oleh akal pikiran manusia. Hasil dari algoritma harus dapat ditentukan bernilai benar atau salah. Karena langkah yang salah akan memberikan hasil yang salah.

Disadari atau tidak setiap kegiatan/ aktifitas sehari-hari kita dipenuhi dengan algoritma. Contoh sederhana adalah langkah-langkah dalam menggosok gigi, dimulai dari mengambil sikat gigi, mengambil pasta gigi, membuka tutup pasta, menuangkan pasta pada sikat gigi, mengambil air, berkumur dengan air, menggosok gigi dengan sikat gigi yang telah diberi pasta, berkumur untuk membersihkan mulut. Kegiatan ini kita dilakukan setiap hari, dan tanpa kita sadari bahwa kita telah melakukan apa yang disebut dengan *algoritma.*

Contoh algoritma:

Menukarkan isi dua buah gelas, dimana gelas A berisi cairan biru, gelas B berisi cairan merah. Maka, untuk menyelesaikan permasalahan tersebut dapat dilakukan cara, misalnya:

- **Algoritma 1:**
  a. Tuangkan isi gelas A ke gelas B
  b. Tuangkan isi gelas B ke gelas A
- **Algoritma 2:**
  a. Tuangkan isi gelas A ke gelas C
  b. Tuangkan isi gelas B ke gelas A
  c. Tuangkan isi gelas C ke gelas B

Dari jawaban kedua algoritma diatas, dapat di analisis:

- Algoritma 1 tidak salah, akan tetapi tidak memberikan penyelesaian yang tepat dan tidak menghasilkan pertukaran yang benar. Karena cairan pada kedua gelas bisa saja tercampur.
- Jawaban pada Algoritma 2 sudah tepat. Dengan menggunakan bantuan sebuah gelas C, maka kedua isi gelas dapat ditukarkan tanpa tercampur.

d. Cara Penyajian Algoritma

Secara umum algoritma disusun untuk menggambarkan langkah-langkah penyelesaian suatu masalah. Dalam membuat suatu algoritma harus memenuhi syarat, terstruktur, masuk akal dan dituliskan secara sistematis, untuk menyelsaikan suatu masalah dengan bantuan komputer. Algoritma dalam penyajiannya dapat dibuat dalam 3 (tiga) bentuk, yaitu:

1. Deskriptif Algoritma

Algoritma dengan kalimat deskriptif dilakukan dengan cara menuliskan instruksi-instruksi yang harus dilaksanakan dalam bentuk untaian kalimat deskriptif dengan bahasa yang jelas. Namun, agar notasi algoritma mudah ditranslasi ke dalam notasi bahasa pemrograman, maka sebaiknya notasi algoritma tersebut berkoresponden dengan notasi bahasa pemrograman umumnya. Kata kerja adalah jenis kata yang biasa digunakan dalam penulisan bahasa deskriptif, contoh tulis, baca, hitung, tampilkan, ulangi, bandingkan, dll.

Notasi jenis ini cocok untuk algoritma yang pendek. Tapi untuk masalah algoritma yang panjang, notasi ini kurang efektif. Pada dasarnya teks algoritma dengan bahasa deskriptif disusun oleh tiga bagian utama yaitu:

a) Bagian judul (header)
b) Bagian deklarasi (kamus)
c) Bagian deskripsi

Disarankan setiap bagian disertai dengan komentar untuk memperjelas maksud teks yang dituliskan. Komentar adalah kalimat yang diapit pasangan tanda kurung kurawal (‘{’ ‘}’).

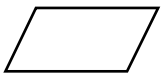
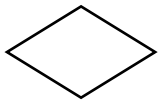
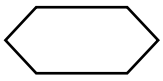
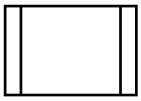
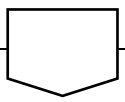
Salah satu contoh sederhana adalah algoritma untuk menentukan bilangan mana yang lebih besar dari input dua buah bilangan. Jadi diberikan input dua buah bilangan dari user, dan program akan secara otomatis menentukan dan memberikan output bilangan mana yang lebih besar. Maka penyelesaian yang bisa dibuat adalah:

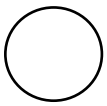
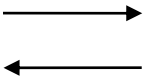
1) Inisialisasi
2) Input dua buah bilangan, masukkan masing-masing dalam variabel, misalkan bil_1 dan bil_2.
3) Apakah bil_1 lebih besar dari bil_2 ?
   - Jika ya, maka bil_1 adalah bilangan yang lebih besar dari bil_2.
   - Jika tidak, maka pasti bahwa bil_2 adalah bilangan yang lebih besar dari bil_1.
4) Tampilkan bilangan terbesar (nilai dari bil_1, atau nilai dari bil_2).
5) Selesai.

2. Flowchart Algoritma

Flowchart merupakan representasi secara diagram dari urutan langkah-langkah untuk mendapatkan suatu hasil. Urutan langkah-langkah ini dilakukan dengan menggambarkan tahap-tahap pemecahan masalah dengan merepresentasikan simbol-simbol tertentu yang mudah dimengerti, mudah digunakan dan standar.

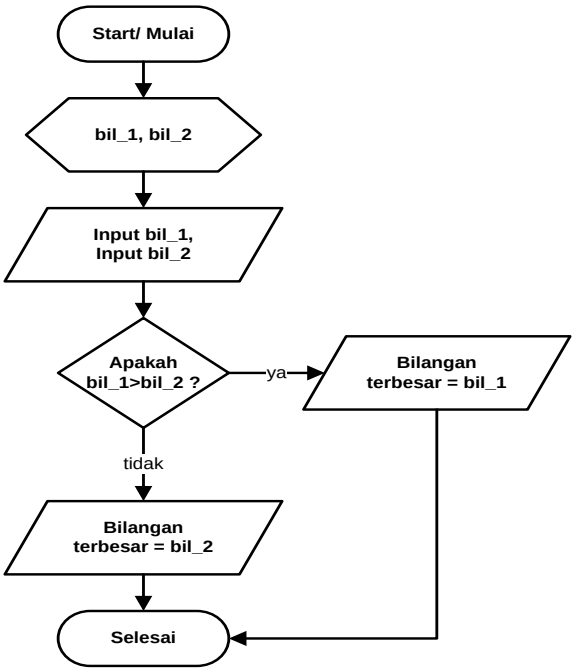
Simbol-simbol dalam flowchart program adalah:

a) 	a) Simbol Terminal, simbol yang digunakan untuk menyatakan awal atau akhir suatu program.
b) 	b) Simbol Input/ Output, untuk menunjuk-kan operasi masukan atau keluaran.
c) 	c) Simbol Proses, untuk menggambarkan proses pengolahan data.
d) 	d) Simbol Keputusan/ Decision, untuk menyatakan suatu pilihan berdasarkan suatu kondisi tertentu.
e) 	e) Simbol persiapan/ Preparation, digunakan untuk memberikan nilai awal pada suatu variabel atau pencacah.
f) 	f) Simbol proses terdefinisi (predefined process), digunakan untuk proses yang detilnya dijelaskan terpisah, misal dalam bentuk Subroutine.
g) 	g) Simbol Penghubung ke halaman lain, untuk menghubungkan

	bagian diagram alir pada halaman yang berbeda.
h) 	h) Simbol Penghubung ke halaman yang sama, untuk menghubungkan bagian diagram alir pada halaman yang sama.
i) 	i) Simbol Arah aliran, simbol yang digunakan untuk menunjukkan arah aliran proses

Contoh penggunaan flowchart program:

Algoritma untuk menentukan bilangan mana yang lebih besar dari input dua buah bilangan.



Gambar 2. Flowchart Algoritma Menentukan Bilangan Yang Terbesar dari Dua Buah Bilangan

3. Pseudocode Algoritma

Pseudo artinya semu atau tidak sebenarnya. Pseudo-code berisikan langkah-langkah untuk menyelesaikan suatu permasalahan (hampir sama dengan algoritma), hanya saja bentuknya sedikit berbeda dari algoritma. Pseudocode menggunakan bahasa yang hampir menyerupai bahasa pemrograman. Selain itu biasanya menggunakan bahasa yang mudah dipahami secara universal dan juga lebih ringkas dari pada algoritma. Dalam penulisan pseudocode disarankan untuk menggunakan keyword yang umum digunakan dalam bahasa pemrograman, seperti : if, then, else, while, do, repeat, for, dan lainnya.

Contoh soal dengan Pseudocode Algoritma:

1) Buatlah pseudocode algoritma untuk menghitung luas dan keliling lingkaran !.

   Jawab: Pseudocode untuk menghitung Luas dan Keliling Lingkaran:

```
(1) Start
(2) phi(π)=3.14,
(3) read jari_jari (r)
(4) Luas = π x r x r
(5) Keliling = 2π x r
(6) cout (“Luas lingkaran : “) ← Luas
(7) cout (“Keliling lingkaran : “) ←Keliling
(8) End
```

2) Buatlah pseudocode algoritma untuk menentukan apakah bilangan yang di input adalah bilangan ganjil atau bilangan genap.

   Jawab:

```
Read bil
if (bil mod 2 = 0) then
  cout (“Output Genap”)
else
  cout (“Output Ganjil” )
```

   **Catatan : Mod adalah sisa hasil bagi (modulus).*

1.2 Pemrograman

Dalam menjalankan komputer, hal terpenting yang harus diketahui dan dipahami adalah program. Dalam pemrograman dikenal beberapa bahasa pemrograman, seperti juga manusia mengenal bahasa-bahasa yang digunakan untuk berkomunikasi. Manusia dalam berkomunakasi menggunakan kata atau karakter sedangkan komputer dengan kode 0 dan 1. Untuk mempermudah manusia berkomunikasi dengan komputer, maka diciptakan bahasa pemrograman. Dengan adanya bahasa pemrograman ini, tidak harus menerjemahkan ke dalam 0 dan 1 untuk berkomunikasi dengan komputer.

1.2.1 Istilah-Istilah Dasar

Dalam berkomunikasi dengan komputer, banyak istilah yang harus dipahami sebelum mendalami lebih jauh tentang pemrograman. Istilah-istilah dasar tersebut adalah:

a. Program

   Program adalah kata, ekspresi, pernyataan atau kombinasi yang disusun dan dirangkai menjadi satu kesatuan prosedur yang menjadi urutan langkah untuk menyesuaikan masalah yang diimplementasikan dengan bahasa pemrograman.

b. Bahasa Pemrograman

   Bahasa pemrograman merupakan prosedur atau tata cara penulisan program dalam bahasa pemrograman, ada dua faktor penting yaitu *sintaksis* dan *semantik*. Sintak adalah aturan-aturan gramatikal yang mengatur tata cara penulisan kata, ekspresi dan pernyataan. Semantik adalah aturan-aturan untuk menyatakan suatu arti. Contoh : Write, Read.

c. Pemrograman

   Pemrograman merupakan proses mengimplementasikan urutan langkah-langkah untuk menyelesaikan suatu masalah dengan bahasa pemrograman.

d. Pemrograman Terstruktur

   Pemrograman Terstruktur merupakan proses mengimplementasikan urutan langkah-langkah untuk menyelesaikan suatu masalah dalam bentuk program yang memiliki rancang bangun yang terstruktur dan tidak berbelit-belit sehingga mudah ditelusuri, dipahami dan dikembangkan oleh siapa saja.

1.2.2 Memahami dan menguraikan permasalahan dalam pemrograman

Study kasus atau permasalahan yang disajikan dalam bentuk cerita/ deskriptif dan atau narasi. Atau bisa berbentuk uraian dari suatu aktivitas atau permasalahan yang ingin diselesaikan. Permintaan penyelesaian masalah dengan syarat dan ketentuan yang dituliskan. Langkah-langkah memahami dan menguraikan suatu permasalahan :

1) Baca soal secara keseluruhan.
2) Ulangi membaca soal kalimat demi kalimat hingga jelas dan paham.
3) Temukan/ identifikasi variabel yang ada, (sesuatu yang diminta dalam soal) tuliskan.
4) Temukan proses yang merupakan solusi. Susun urutan proses tersebut.
5) Hasil akhir apa yang ingin ditampilkan (diminta oleh soal).

1.3 Pemrograman Terstruktur

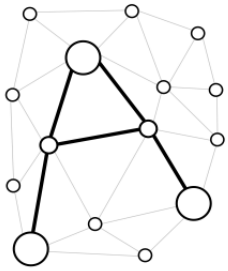
Sebagai langkah awal untuk mempelajari bahasa pemrograman adalah memulai dengan memahami konsep pemrograman terstruktur. Pemrograman terstruktur merupakan suatu tindakan untuk mengorganisasikan dan membuat kode-kode program supaya mudah dimengerti, mudah dites, dan mudah dimodifikasi.

Ciri-ciri teknik pemrograman terstruktur :

1) Mengandung teknik pemecahan masalah tepat dan benar.
2) Memiliki algoritma pemecahan masalah yang bersifat sederhana, standar dan efektif dalam menyelesaikan masalah.
3) Teknik penulisan program memiliki struktur logika yang benar dan mudah dipahami.
4) Program terdiri dari tiga struktur yaitu *sequence structure, selection structure* dan *looping structure*.
5) Menghindarkan penggunaan instruksi *GOTO* (peralihan proses tanpa syarat tertentu) yang menjadikan program tidak terstruktur lagi.
6) Membutuhkan biaya testing yang rendah.
7) Memiliki dokumentasi yang baik.
8) Membutuhkan biaya perawatan dan pengembangan yang rendah.

Latihan Praktikum - 1

1. Buatlah algoritma sederhana cara membuat telur dadar dengan kalimat deskriptif. Gunakan struktur bahasa Indonesia yang jelas dan mudah dimengerti!.
2. Buatlah flowchart algoritma untuk menghitung luas segitiga:
   a. Segitiga Siku
   b. Segitiga Sama Kaki
3. Buatlah Pseudocode algoritma untuk menghitung keliling persegi panjang!.



BAB 2
STRUKTUR DASAR ALGORITMA

Dalam membangun sebuah algoritma sebagai langkah-langkah penyelesaian masalah, haruslah sesuai dengan kaidah penyusunan algoritma atau struktur dasar algoritma yang terdiri dari:

2.1 Struktur Urut (*sequence*)

Struktur urut adalah suatu struktur program dimana setiap baris program akan dikerjakan secara urut dari atas ke bawah sesuai dengan urutan penulisannya.

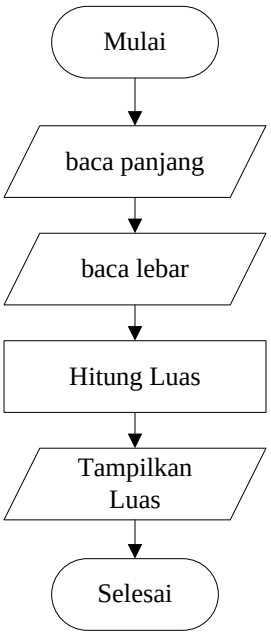
Contoh :

Algoritma untuk menghitung luas pesegi panjang yang diketahui panjang dan lebarnya, adalah sebagai berikut :

- Dalam bentuk pseudocode algoritma dapat dituliskan:

```
1) MulaiMengerti dan memahami bentuk struktur
2) Baca panjang
3) Baca lebar
4) Hitung luas = panjang * lebar
5) Tampilkan luas
6) Selesai
```

- Dalam bentuk flowchart Algoritmanya adalah:



Gambar 2.1 Algoritma Luas Pesegi Panjang

2.2 Struktur Pemilihan (*selection*/ Penyeleksian Kondisi)

Pada struktur pemilihan tidak setiap baris program akan dikerjakan. Baris program yang dikerjakan hanya yang memenuhi syarat saja. Struktur pemilihan adalah struktur program yang melakukan proses pengujian untuk mengambil suatu keputusan apakah suatu baris atau blok instruksi akan diproses atau tidak. Pengujian kondisi dilakukan untuk memilih satu dari beberapa alternatif yang tersedia. Seleksi kondisi, haruslah dapat menghasilkan nilai benar (*true*) atau nilai salah (*false*).

Penulisan program untuk struktur pemilihan dapat diimplementasikan dengan dua cara:

2.2.1 Instruksi *if*

Struktur pemrograman dengan instruksi *if* terdiri dari:

1) ***if* sederhana,** dengan bentuk dasar:

```
if (kondisi)
  pernyataan;
```

Contoh 2.1:

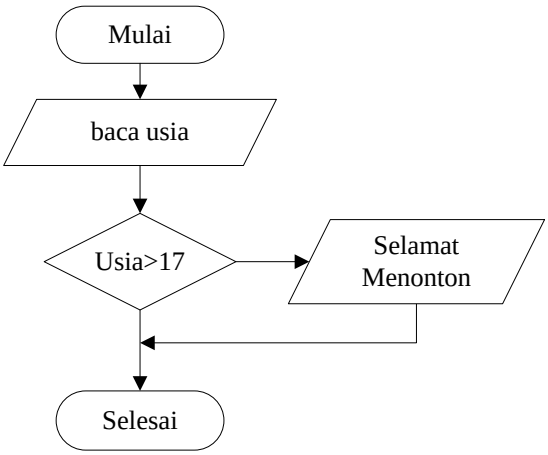
Algoritma menonton_film,

- Dengan pseudocode algoritma **:**

```
1) mulai
2) baca usia
```

3) jika (usia>17), maka kerjakan langkah 4 selain itu selesai.
4) Cetak "Selamat Menonton"
5) Selesai

- Dengan struktur flowchart:



Gambar 2.2 Struktur Flowchart *if sederhana*

2) if … then ….else …

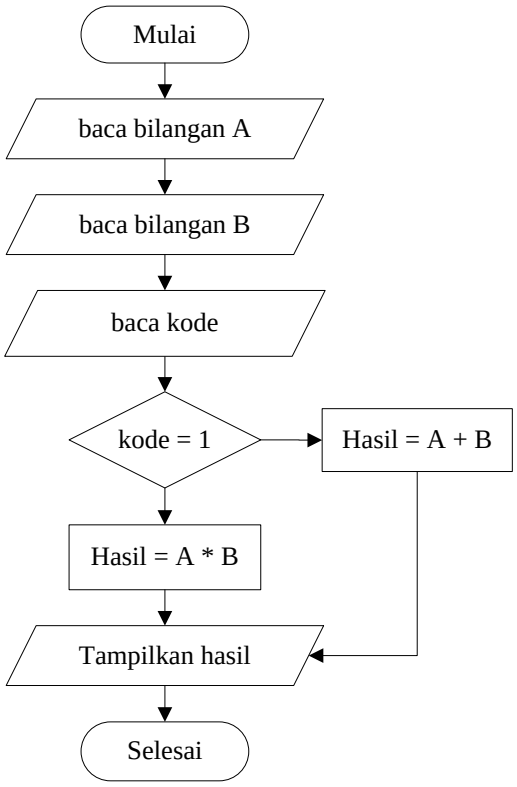
bentuk dasarnya:

```
if (kondisi)
  pernyataan1;
else
  pernyataan2;
```

Contoh 2.2:

Algoritma untuk melakukan operasi dua bilangan berdasarkan kode operasi, misalkan bilangan A dan bilangan B. Jika kode bernilai 1 maka lakukan penjumlahan dua bilangan, dan apabila kode berisi selain 1 maka lakukan perkalian dua bilangan.

- Dalam flowchart algoritma dapat digambarkan seperti berikut:



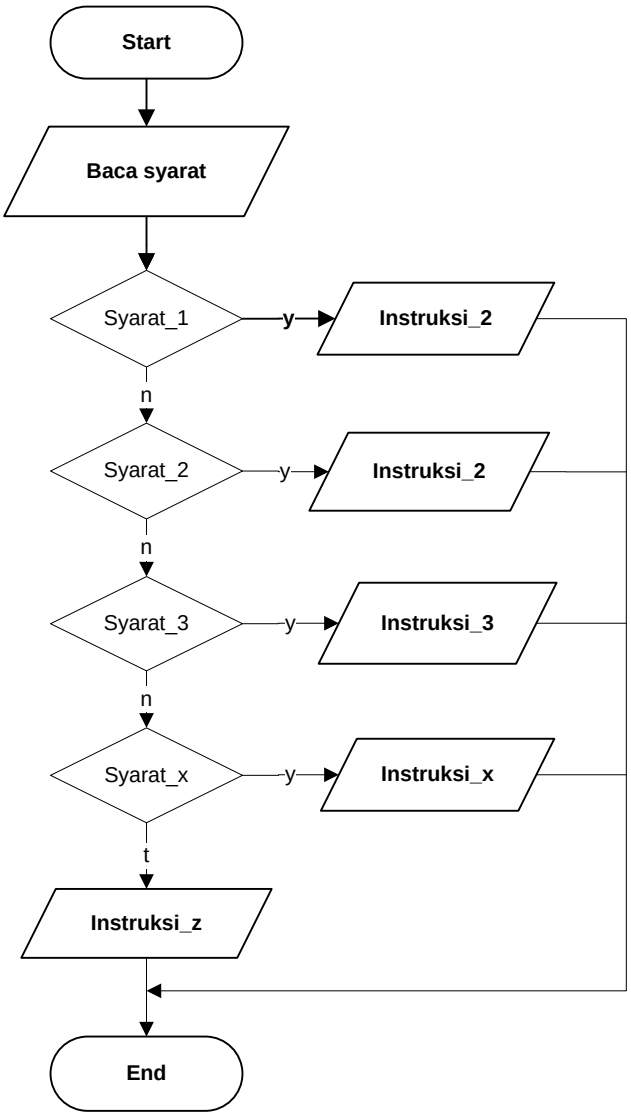
Gambar 2.3 Algoritma Pemilihan Operasi Dua Bilangan

- Dalam pseudocode algoritma dapat dituliskan:

```
1) mulai
2) Baca bilangan A
3) Baca bilangan B
4) Baca kode operasi
5) Jika (kode=1), kerjakan langkah. Jika (kode=2/selain 1),
   kerjakan langkah 7.
6) Hasil = A + B
7) Hasil = A * B
8) Tampilkan hasil
9) Selesai
```

3) ***nested if***

Struktur pemilihan dalam bentuk *nested if* mempunyai bentuk flowchart sebagai berikut:

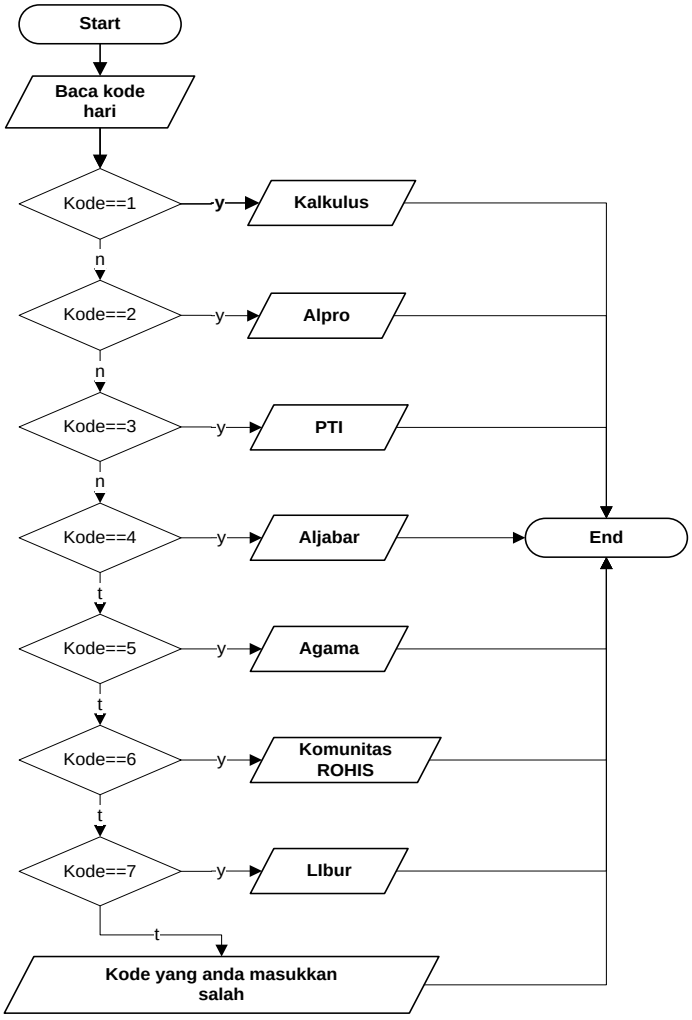


Gambar 2.4 Algoritma Pemilihan Nested If

Contoh penggunaan *nested if :*

Buatlah sebuah flowchart algoritma yang menampilkan jadwal kegiatan kuliah anda dalam satu minggu. Dimana inputan berupa angka (1-7) mewakili kode hari dalam satu minggu, dan kode hari selain angka (1-7) adalah kode yang salah.

Maka bentuk flowchart yang bisa dibuat adalah:



Gambar 2.5 Contoh Flowchart dengan *nested if*

2.2.2 Stetement Switch

Statemen ***switch*** digunakan untuk melakukan pemilihan terhadap ekspresi atau kondisi yang memiliki nilai-nilai konstan (tetap/ angka). Ekspresi yang didefinisikan harus menghasilkan nilai yang bertipe bilangan bulat atau karakter. Untuk mendefinisikan nilai-nilai konstan tersebut digunakan kata kunci case. Hal yang perlu diperhatikan dalam melakukan pemilihan menggunakan statamen switch ini adalah harus menambahkan statemen break pada setiap nilai yang didefinisikan. Pernyataan ini juga mirip dengan **nested if**.

Bentuk umum dari statemen switch adalah:

```
switch (ungkapan)
{
  case ungkapan1:
       pernyataan1;
       break;
```

```
    case ungkapan2:
         pernyataan2;
         break;
    .......
    default:                    //optional
         pernyataan_x;          //optional
}
```

Latihan Praktikum - 2

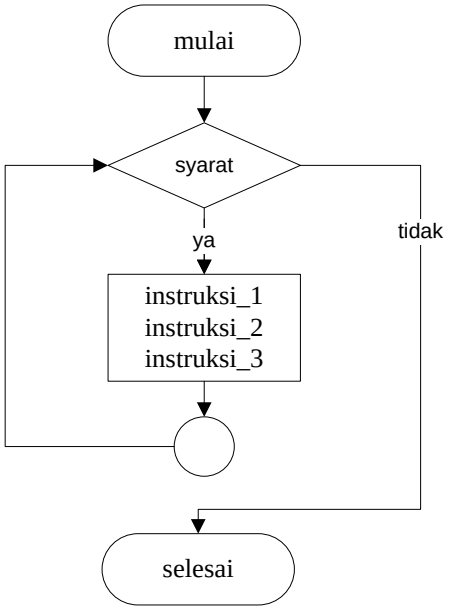
1. Dengan konsep pemilihan, buat flowchart algoritma untuk mencari nilai P dan Q, dimana P = X + Y. Jika P bernilai positif, maka Q = X * Y, sedangkan jika P bernilai negative maka nilai Q = X/Y.
2. Buatlah sebuah flowchart untuk melakukan operasi hitung dari 2 buah bilangan, missal bilangan A dan B. Sebagai dasar pemilihan digunakan kode_hitung, jika kode=1, maka lakukan penjumlahan bilangan, jika kode=2 maka lakukan perkalian, jika kode=3 lakukan pembagian, jika kode=4 pengurangan, selain itu tidak ada operasi hitung
3. Buatlah Pseudocode algoritma untuk menghitung nilai akhir dari siswa dengan ketentuan: nama siswa, nilai tugas, nilai UTS, dan nilai UAS, diinput. Nilai Akhir = (nilai tugas+UTS+UAS)/3. Berikan predikat kelulusan untuk mata kuliah tersebut. Jika nilai_akhir<65 maka Tidak Lulus, jika nilai_akhir> 65 maka Lulus. Konversikan nilai akhir dalam bentuk huruf. huruf hasil konfersi dengan aturan :
   Jika nilai_angka >= 80 maka nilai huruf sama dengan A
   Jika nilai_angka >= 70 maka nilai huruf sama dengan B
   Jika nilai_angka >= 60 maka nilai huruf sama dengan C
   Jika nilai_angka >= 50 maka nilai huruf sama dengan D
   Jika nilai_angka < 50 maka nilai huruf sama dengan E
4. Buatlah flowchart algoritma untuk menghitung luas dan keliling dari lingkaran, segitiga, dan trapesium. Kode bangun 1= lingkaran, 2= segitiga, 3=trapesium !.

2.3 Struktur Pengulangan (*repetition*)

Struktur pengulangan merupakan struktur yang melakukan pengulangan terhadap satu baris atau satu blok baris program beberapa kali sesuai dengan persyaratan yang diberikan. Struktur pengulangan mempunyai beberapa bentuk :

a) *Struktur While*

Struktur pengulangan dengan instruksi `while` digunakan untuk mengulang satu baris instruksi (satu blok) selama syarat yang diberikan masih terpenuhi. Pada pengulangan `while`, syarat akan uji terlebih dahulu sebelum instruksi yang akan diulang dikerjakan, yang berarti syarat akan diuji didepan, sehingga ada kemungkinan baris instruksi yang akan diulang tidak dikerjakan sama sekali (syarat tidak terpenuhi).

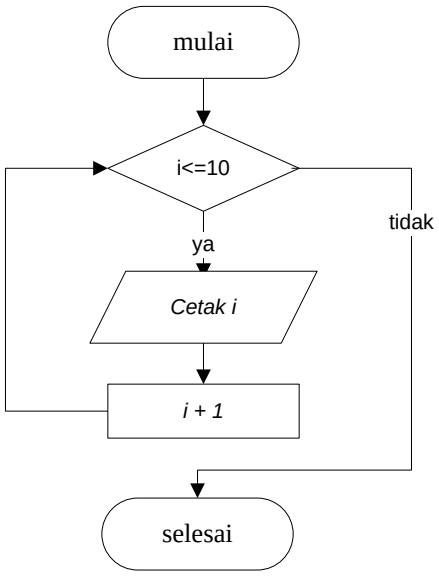


Gambar 2.6 Flowchart Struktur *while*

Contoh 2.3:

Algoritma untuk cetak angka 1-10 dengan while yang bisa dibuat adalah:

- Dengan flowchart algoritma dapat digambarkan:



Gambar 2.7 Flowchart cetak angka 1-10 dengan while

- *Dengan pseudocode algoritma dapat dituliskan:*

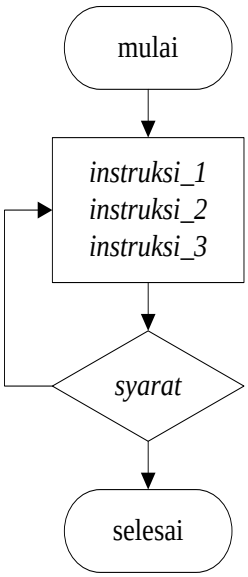
```
1) Mulai
2) i = 1
3) selama i <= 10 kerjakan langkah 4 sampai langkah 5
4) cetak i
5) i = i + 1
6) selesai
```

Dari flowchart dapat dijelaskan pertama kali ***i*** bernilai 1, kemudian diuji apakah ***i*** lebih kecil atau sama dengan 10, jika benar dicetak nilai ***i***, kemudian nilai ***i*** dinaikkan sebesar 1, kemudian nilai ***i*** diuji kembali apakah masih lebih kecil atau sama dengan 10 jika benar maka dicetak nilai ***i***, begitu seterusnya. Perulangan akan berhenti jika nilai ***i*** lebih besar 10.

b) Struktur *do … while …*

Struktur pengulangan dengan instruksi *do…while* digunakan untuk mengulang satu baris instruksi (satu blok) sampai syarat tidak terpenuhi. Ciri utama pengulangan *do…while* adalah syarat akan uji setelah instruksi yang akan diulang dikerjakan, dengan kata lain syarat akan diuji dibelakang, sehingga baris instruksi yang masuk dalam blok *do…while* minimal akan dikerjakan satu kali meskipun kondisi syarat bernilai salah.

Dengan struktur flowchart:



Gambar 2.8 Flowchart cetak angka 1-10 dengan while

Dari bentuk flowchart diatas dapat dituliskan bentuk dasar struktur *do ... while ...* :

```
do
{
   pernyataan1;
   pernyataan2;
   ...
   pernyataanN;
} while (ungkapan)
```

Algoritma Cetak_Angka_do_while :

```
1) mulai
2) i = 0
3) i = i + 1          //bagian dimulainya do …
4) cetak i
5) jika i < 10 kerjakan langkah 3 sampai langkah 4
6) selesai
```

c) Struktur *for*

Struktur pengulangan dengan intruksi *for* digunakan untuk mengulang satu baris instruksi atau satu blok instruksi sampai jumlah perulangan yang disyaratkan terpenuhi. Ciri utama pengulangan *for* adalah terdapat nilai awal dan nilai akhir yang menunjukkan banyaknya pengulangan yang akan dilakukan. Dengan bentuk dasar:

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

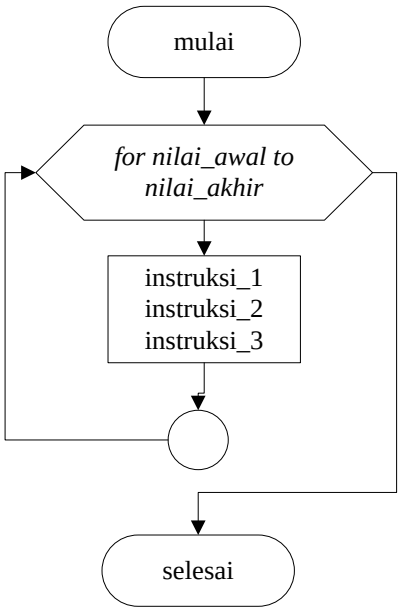
Bentuk dasar *struktur for* tersebut sama dengan bentuk dasar struktur while, yaitu:

```
ungkapan1;
while (ungkapan2)
{
    pernyataan;
    ungkapan3;
}
```

dimana:

- *ungkapan1* merupakan pernyataan inisialisasi sebelum masuk ke while.

- *ungkapan2* berlaku sebagai kondisi yang menentukan pengulangan terhadap pernyataan atau tidak.
- *ungkapan3* digunakan sebagai pengatur variabel yang digunakan di dalam *ungkapan1*.



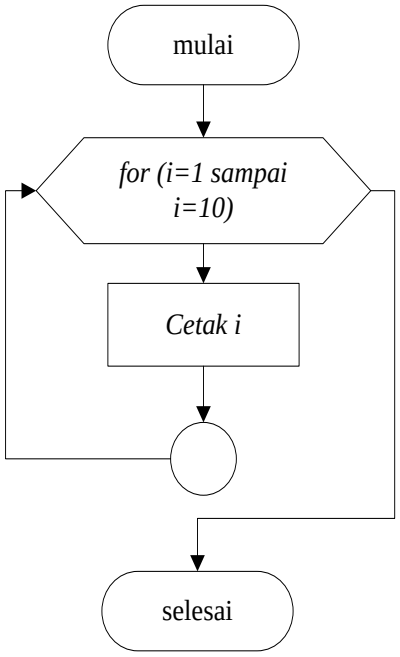
Gambar 2.9 Flowchart Struktur *for*

Dari gambar flowchart diatas dapat dijelaskan bahwa `instruksi1, instruksi2, instruksi3` akan dikerjakan berulang yang dimulai dari nilai_awal sampai nilai_akhir yang diberikan. Jika pengulangan sudah sampai pada kondisi nilai_akhir yang diberikan maka pengulangan akan berhenti.

Contoh:

Buatlah algoritma untuk mencetak angka 1 sampai 10 menggunakan perulangan *for. :*

- Dengan flowchart algoritma dapat digambarkan seperti sebagai berikut:



Gambar 2.10 flowchart struktur *for* untuk mencetak angka

- Dengan pseudocode algoritma, flowchart diatas dapat dideskripsikan sebagai berikut:

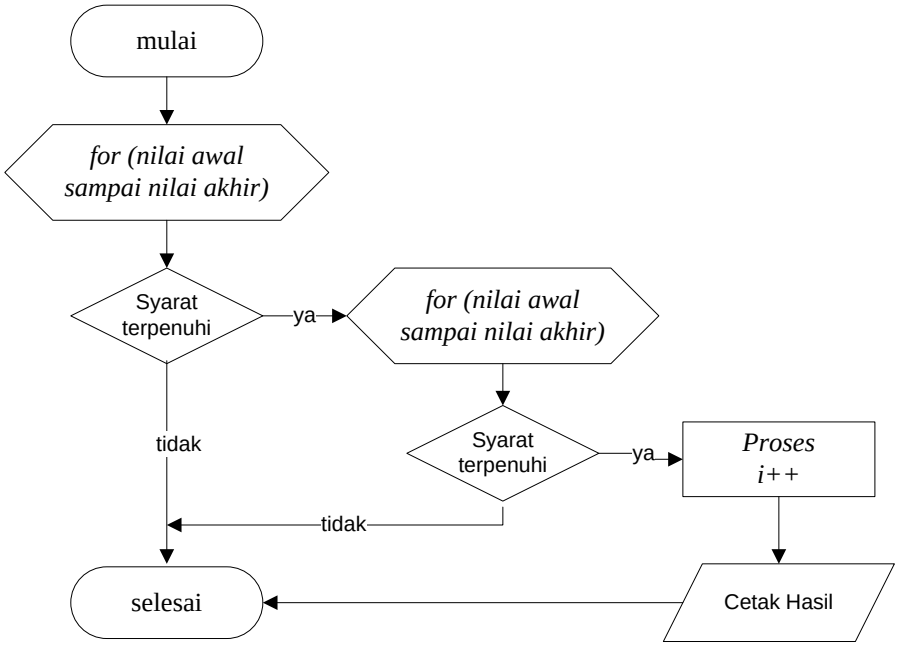
  Algoritma Cetak_Angka_for

  ```
  1) mulai
  2) kerjakan langkah 3 mulai i = 1 sampai i = 10
  3) cetak i
  4) selesai
  ```

Dari algoritma diatas dapat dijelaskan bahwa nilai i pertama akan berisi 1, kemudian dicetak nilai i, dalam perulangan for nilai variabel i akan bertambah secara otomatis sehingga nilai variabel i sekarang menjadi 2, kemudian dicetak nilai i, begitu seterusnya sampai nilai i berisi 10, maka proses pengulangan selesai.

d) Struktur *for bersarang*

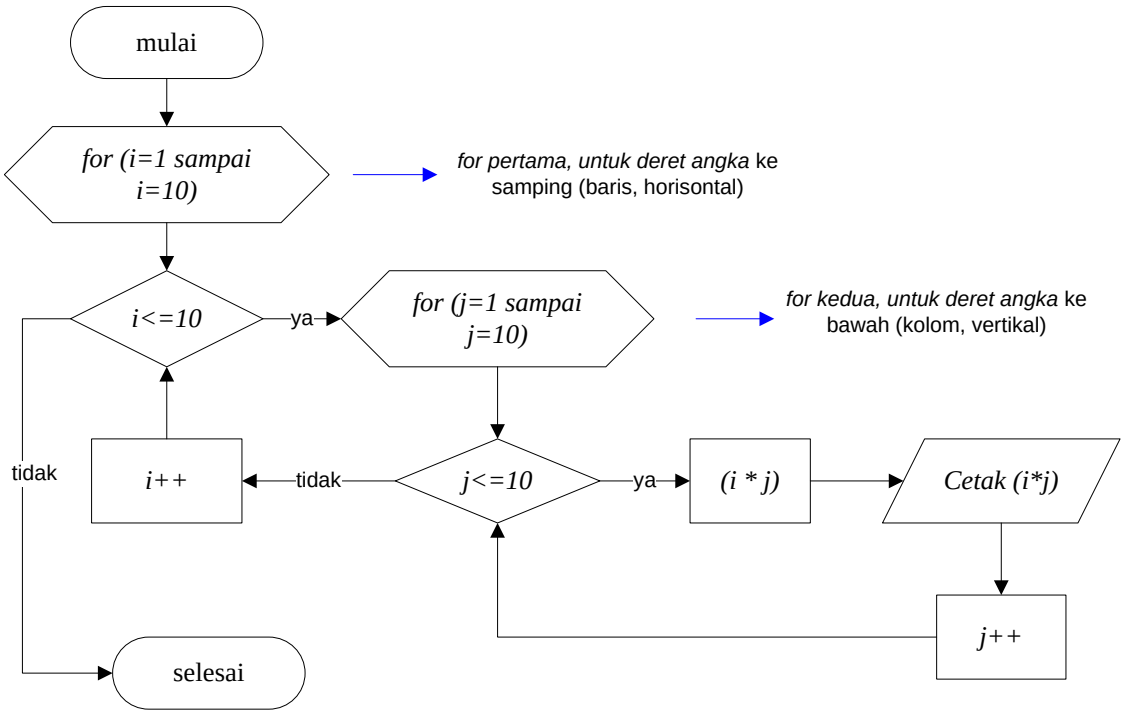
Proses pengulangan dapat dilakukan juga di dalam strukur pengulangan yang lain, atau disebut dengan pengulangan di dalam pengulangan. Struktur program seperti ini dapat dilakukan dengan implementasi *for dalam for*, dengan bentuk flowchart sebagai berikut:



Gambar 2.11 Flowchart Struktur *for dalam for*

Contoh 2.4:

Penggunaan struktur *for dalam for* adalah algoritma untuk menampilkan hasil perkalian angka 1 sampai 10.



Gambar 2.12 Flowchart Struktur *for dalam for*

Dari flowchart algoritma diatas dapat dideskripsikan sebagai berikut:

```
1)  Mulai
2)  Kerjakan langkah 3 mulai dari i=1 sampai i=10
3)  Jika i<=10, kerjakan langkah 4
4)  Kerjakan langkah 5 mulai dari j=1 sampai j=10
5)  Jika j<=10 kerjakan langkah 6
6)  i*j
7)  cetak (i+j)
8)  j=j+1
9)  kembali ke langkah 5, sampai j>10 (langkah 5 bernilai
    salah)kerjakan langkah 10
10) i=i+1
11) kembali ke langkah 3 sampai 10, sampai i>10 (langkah 3
    bernilai salah)
12) selesai
```

Hasil algoritma perkalian angka 1 sampai 10 adalah:

```
 1   2   3   4   5   6   7   8   9  10
 2   4   6   8  10  12  14  16  18  20
 3   6   9  12  15  18  21  24  27  30
 4   8  12  16  20  24  28  32  36  40
 5  10  15  20  25  30  35  40  45  50
 6  12  18  24  30  36  42  48  54  60
 7  14  21  28  35  42  49  56  63  70
 8  16  24  32  40  48  56  64  72  80
 9  18  27  36  45  54  63  72  81  90
10  20  30  40  50  60  70  80  90 100
```

Untuk pemrograman dengan konsep perulangan bersarang *(nested repetition)* dapat juga dibuat dengan kombinasi antara struktur *for* dan *while.* Flowchart algoritma untuk kombinasi perulangan dengan *for* dan *while* sama bentuknya seperti flowchart *nested for*. Contoh sederhana dengan struktur bahasa pseudocode:

```
1)  Mulai
2)  Kerjakan langkah 3 mulai dari i=1 sampai i=10 //for
3)  Jika i<=10, kerjakan langkah 4
4)  j = 1  //while
5)  selama j <= 10 kerjakan langkah 5 sampai langkah 8
```

```
6)  i*j
7)  cetak (i+j)
8)  j=j+1
9)  kembali ke langkah 5 sampai 8, sampai j>10 (langkah 5
    bernilai salah)kerjakan langkah 10
10) i=i+1  //kembali ke for
11) kembali ke langkah 3 sampai 10, sampai i>10 (langkah 3
    bernilai salah)
12) selesai
```

Latihan Praktikum - 3

1. Buatlah flowchart algoritma dan pseudocode algoritma dengan struktur bahasa indonesia untuk mencetak bilangan genap sampai batas nilai tertentu (nilai batas diinputkan) dengan struktur pengulangan!
2. Buatlah algoritma dengan struktur bahasa Indonesia (pseudocode) dan flowchart untuk mencari angka terbesar dan angka terkecil dari sejumlah angka yang diinputkan dengan menggunakan pengulangan. Bila diperlukan berikan batas jumlah angka yang diinputkan.
3. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menghitung total pembayaran pembelian. Input berupa nama pembeli, nama barang dengan jumlah barang yang dibeli, jumlah dan harga barang. Barang yang dibeli jumlahnya bisa banyak tergantung pembelian dari konsumen.

Tugas (1)

1) Buatlah flowchart algoritma untuk mencari luas dan volume tabung!
2) Buatlah pseudocode dan flowchart algoritma untuk mengetahui bilangan terbesar dari input 3 (tiga) buah bilangan!
3) Dengan struktur pengulangan buatlah flowchart/ pseudocode algoritma untuk menuliskan deret angka kelipatan 3 sampai jumlah/ batas tertentu!.
4) Buatlah flowchart dan pseudocode algoritma untuk menghitung gaji yang harus dibayarkan diakhir bulan kepada pegawai harian dengan data inputan nama pegawai,

jumlah hari kerja, dan jumlah jam lembur. Gaji per hari Rp.40.000,upah lembur per jam sebesar Rp.5000. Gaji pegawai dihitung dari gaji per hari*jumlah hari kerja. Total Upah lembur didapat dari upah lembur*jumlah jam lembur. Uang transport akan diberikan kepada pegawai yang lembur lebih dari 10 jam dalam sebulan sebesar 10% dari total upah lembur.

BAB 3
DASAR PEMROGRAMAN BAHASA C++

3.1 Bahasa C++

C merupakan bahasa pemrograman dasar terstruktur yang berkekuatan tinggi dan fleksibel. Bahasa C dikembangkan di laboratorium Bell AT&T di Murray, New Jersey sebagai bahasa pemrograman untuk sistem operasi Unix yang bertujuan untuk membuat antara muka yang bersifat *user friendly*. Pencipta C adalah pengembang sistem bernama Dennis M. Ritchie dan Brian W. Kernighan pada sekitar tahun 1971-1972. Bahasa C adalah pengembangan dari bahasa sebelumnya yang dinilai masih terkesan lambat, yaitu bahasa B pada tahun 1970, oleh seorang pengembang sistem dari laboratorium yang sama bernama Ken Thompson. Bahasa C dikembangkan dengan tujuan untuk menulis ulang dan menutupi kelemahan-kelemahan yang ada pada sistem operasi Unix sebelumnya. Sejak itu bahasa C terus digunakan untuk memelihara sitem operasi Unix. Sampai akhirnya pada tahun 90-an, bahasa C ini digunakan untuk mengembangkan sistem operasi Windows dan sekarang ini digunakan untuk mengembangkan sistem operasi Linux. Selain untuk menulis program yang merupakan *embedded system*, di kalangan industry hiburan, bahasa C banyak digunakan dalam mengembangkan perangkat lunak untuk permainan (*game*). Hal ini yang menyebabkan Bahasa C menjadi bahasa yang sangat populer di kalangan industri perangkat lunak.

Saat ini banyak sekali terdapat bahasa permograman tingkat tinggi (*high level language*) seperti Pascal, BASIC, COBOL dan lainnya. Walaupun demikian, sebagian besar dari para programmer profesional masih tetap memilih bahasa C sebagai bahasa yang lebih unggul, dengan pertimbangan bahwa bahasa C mempunyai kelebihan-kelebihan antara lain:

- Bahasa C++ tersedia hampir di semua jenis komputer.
- Kode bahasa C/C++ sifatnya adalah portable dan fleksibel untuk semua jenis komputer.
- Proses executable program bahasa C/C++ lebih cepat.
- Dukungan pustaka yang banyak.
- C adalah bahasa yang terstruktur.

- C++ sudah mendukung OOP (Object Oriented Programming).

Sedangkan kekurangan Bahasa C/C++ yang

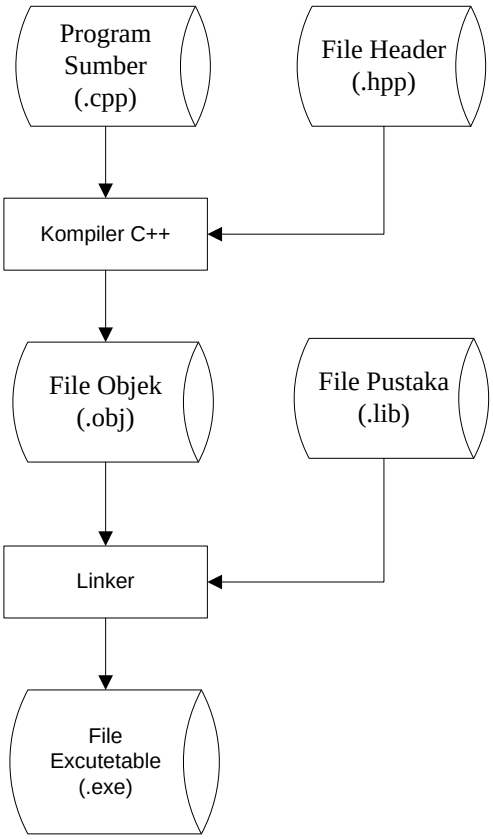
- Banyaknya Operator serta fleksibilitas penulisan program kadang–kadang membingungkan pemakai.
- Bagi pemula pada umumnya akan kesulitan menggunakan pointer dan penerapan konsep OOP (objeck Oriented Programming).

Editor Bahasa C/C++

Untuk memulai membuat program, tersedia berbagai editor yang dapat digunakan ,diantaranya : Turbo C++, Borland C++, C++ builder, Microsoft Visual C++, dan lainnya.

3.2 Dasar Pemrograman C++

C adalah bahasa pemrograman terstruktur yang membagi program dalam bentuk sejumlah blok. Tujuannya adalah untuk memudahkan dalam pembuatan dan pengembangan program. Program C++ biasa ditulis dengan nama ekstensi .CPP (dari kata C plus plus). Agar program ini dapat dijalankan (dieksekusi), program harus dikompilasi lebih dulu dengan menggunakan kompiler C++. Agar program dapat dieksekusi, program harus dikompilasi dahulu menggunakan compiler C++. Proses kompilasi file sumber (.cpp) bersama dengan file-file header (.h) akan diterjemahkan oleh kompiler C++ menjadi kode objek (.obj). File objek ini dalam format biner (berkode 0 dan 1). Selanjutnya file objek bersama file objek lain serta file pustaka (.lib) dikaitkan menjadi satu oleh linker. Hasilnya file *Executable*.



Gambar 3.1 Proses Pembentukan File Executable

3.2.1 Struktur Program C++

Struktur dasar pemrograman C++ adalah:

```
#include <nama_file>
void main()
{
   <blok_pernyataan>
}
```

Dari kerangka program diatas, masing-masing bagian dapat dijelaskan:

1. **#include**

   Adalah pengarah preprosesor yang berfungsi menginstruksikan kepada kompiler untuk menyisipkan file lain saat program dikompilasi (yaitu file-file yang berisi stream dan fungsi-fungsi input/output). Fungsi input/output merupakan pustaka/ library. Pustaka ini telah ada di header **stdio.h** dan **iostream.h**.

2. **Void**, terletak didepan **main()**, untuk menyatakan bahwa fungsi main() tidak memiliki nilai balik (nilai yang dikirim ke fungsi/ bagian program yang lain)

3. **main()** menjadi awal dan akhir eksekusi program C++, sehingga sebuah program dalam C++ mengandung sebuah fungsi main().

   main → nama judul fungsi
   { →awal tubuh fungsi/awal eksekusi program
      →tubuh fungsi/blok
   } →akhir tubuh fungsi/akhir eksekusi program

   Tanda **()** digunakan untuk mengapit argumen fungsi, yaitu nilai yang akan dilewatkan ke fungsi.

4. **Blok Pernyataan**

   Merupakan satu atau beberapa buah statemen/pernyataan yang pada setiap akhir baris pernyataan diakhiri dengan titik koma (;).

Setiap program yang ditulis dengan C++ harus mempunyai fungsi utama, yang bernama **main()**. Fungsi inilah yang akan dipanggil pertama kali pada saat proses eksekusi program. Artinya apabila kita mempunyai fungsi lain selain fungsi utama, maka fungsi lain tersebut baru akan dipanggil pada saat digunakan. Fungsi main() ini dapat mengembalikan nilai 0 ke sistem operasi yang berarti bahwa program tersebut berjalan dengan baik tanpa adanya kesalahan. Berikut dua bentuk kerangka fungsi main() dalam bahasa C yang dapat digunakan.

1. Bentuk Pertama (*tanpa pengembalian nilai ke sistem operasi*)

```
void main(void)
{
  Statemen_ yang_akan_dieksekusi;
  ...
}
```

Kata kunci void di atas bersifat opsional, dalam arti bisa dituliskan dan bisa juga tidak.

2. Bentuk Kedua (*dengan mengembalikan nilai 0 ke sistem operasi*)

```
int main(void)
{
  Statemen_yang_akan_dieksekusi;
  ...
  return 0;
}
```

Kata kunci void di atas juga bersifat opsional. Namun, para programmer C pada umumnya menuliskan kata kunci tersebut di dalam fungsi yang tidak memiliki parameter, juga fungsi yang tidak memiliki nilai balik.

Dari kedua bentuk diatas, kerangka lengkap program bahasa C dapat dituliskan:

```
#include <nama_header_file>
...          //header-header yang lain
/* Prototipe fungsi */
tipe_data nama_fungsi1(parameter1, parameter2, ...);
tipe_data nama_fungsi2(parameter1, parameter2, ...);
...
int main(void)    //Fungsi utama
{
  Statemen_yang_akan_dieksekusi;
  ...
  return 0;
}
//Implementasi fungsi
tipe_data nama_fungsi1(parameter1, parameter2, ...)
{
  Statemen_yang_akan_dieksekusi;
  ...
}
tipe_data nama_fungsi1(parameter1, parameter2, ...)
{
  Statemen_yang_akan_dieksekusi
  ...
}
```

Bahasa C merupakan bahasa prosedural yang menerapkan konsep runtunan/terurut (program dieksekusi per baris dari atas ke bawah secara berurutan), maka apabila kita menuliskan fungsi-fungsi lain tersebut di bawah fungsi utama, maka kita harus menuliskan bagian prototipe (*prototype*), hal ini dimaksudkan untuk mengenalkan terlebih dahulu kepada kompilator daftar fungsi yang akan digunakan di dalam program. Namun apabila kita

menuliskan fungsi-fungsi lain tersebut di atas atau sebelum fungsi utama, maka kita tidak perlu lagi untuk menuliskan bagian prototipe di atas.

3.2.2 Elemen Dasar C++

Elemen-elemen dasar C++ adalah elemen (objek-objek) yang selalu ada dalam pemrograman C++. Elemen dasar tersebut adalah:

1) Himpunan Karakter

Himpunan karakter pada C++ terdiri dari huruf, digit maupun simbol-simbol lainnya (termasuk spasi, karakter kontrol).

Huruf : ABCDEFGHIJKLMNOPQRSTUVWXYZ,
abcdefghijklmnopqrstuvwxyz

Digit : 0123456789

Simbol : _ - + * dan sebagainya.

2) Pengenal (*Identifier*)

Pengenal adalah suatu nama yang biasa dipakai dalam pemrograman untuk menyatakan menyatakan variabel, konstanta, tipe data, fungsi, label, obyek serta hal-hal lain yang dibuat oleh pemrogram. Suatu pengenal merupakan kombinasi dari huruf, angka dan garis bawah (_). Penamaan pengenal harus berawalan dengan huruf atau garis bawah dan menggunakan kata yang mudah dipahami dan dapat mewakili fungsi dari pengenal yang dibuat. Pengenal dalam C++ bersifat ***case sensitive*** atau dibedakan antara huruf kecil dan huruf besar. Misalkan pengenal **NAMA**, **nama**, **Nama** merupakan buah tiga pengenal yang berbeda.

Pemberian nama pengenal haruslah berupa satu atau beberapa karakter yaitu : huruf, digit, garis bawah (_) dan berawalan dengan huruf atau garis bawah. Disarankan agar pemberian nama pengenal menggunakan nama yang berarti dan mudah dibaca. Misalnya: **gaji_pegawai** yang menyatakan gaji pegawai lebih baik, mudah diingat daripada **g** saja. Berikut ini contoh pengenal yang boleh dan tidak boleh.

Boleh	Tidak Boleh
nama	2semester (tidak boleh diawali dengan angka)
NAMA	nama-barang (tanda – tidak diperkenankan)

nama_barang	#barang (simbol # tidak diperkenankan)
kuartal_2	nama barang (tidak boleh mengandung spasi)

Pemberian nama pengenal tidak boleh menggunakan kata-kata yang merupakan *keyword* (kata kunci) dari bahasa pemrograman, yaitu adalah pengenal sistem yang mempunyai makna khusus bagi kompiler. Contoh: do, else, class, for, if, delete dan yang lainnya.

3) Tipe Data

Berikut tipe data dasar pada C++ beserta ukurannya:

Tipe Data	Ukuran Memori	Jangkauan Nilai	Jumlah Digit Presisi
char	1 byte	-128 hingga +127	-
int	2 byte	-32.768 hingga +32.767	-
Short	2 byte	-32.768 hingga +32.767	-
long	4 byte	-2.147.438.648 hingga 2.147.438.647	-
float	4 byte	3.4×10^{-38} hingga $3.4 \times 10^{+38}$	6-7
double	8 byte	1.7×10^{-308} hingga $1.7 \times 10^{+308}$	15-16
long double	10 byte	3.4×10^{-4932} hingga $1.1 \times 10^{+4932}$	19

Tipe data yang berhubungan dengan bilangan bulat adalah char, int, short dan long. Sedangkan yang lainnya berhubungan dengan bilangan pecahan. Pemilihan tipe variabel disesuaikan dengan data yang akan disimpan. Jika untuk bilangan bulat maka gunakan tipe data bilangan bulat, seperti : **int**, **long**. Jika untuk bilangan pecahan maka gunakan tipe data bilangan pecahan, seperti : **float**. Selain itu jangkauan tipe data juga harus disesuaikan dengan kemungkinan data yang akan disimpan dalam variabel.

Perlu diketahui bahwa agar sedapat mungkin menggunakan variabel dengan tipe data yang berukuran memori lebih kecil.

4) Variabel & Konstanta

Penggunaan variabel dan konstanta dalam pemrograman tidak terlepas dari tipe data. Dalam pemrograman komputer, tipe data digunakan untuk mendefinisikan suatu variabel atau konstanta. Variabel adalah suatu memori yang dialokasikan dengan nama

(pengenal) tertentu dan hanya bisa menampung data sesuai dengan tipe yang ditentukan. Sifat dari variabel adalah nilai yang dikandung akan mudah diubah sesuai dengan proses yang terjadi seperti contoh dibawah ini. Konstanta adalah suatu memori yang dialokasikan dengan nama tertentu yang berisi suatu nilai yang memiliki sifat tetap yang tidak akan bisa berubah.

a) Mendeklarasikan & Mendefinisikan Variabel

Sebelum variabel digunakan, maka variabel tersebut harus didefinisikan terlebih dahulu. Pendefinisian variabel dapat dimana saja sebelum variabel itu digunakan dengan bentuk :

```
<tipedata> <namavariabel>;
```

atau:

```
<tipedata> <daftarvariabel>;
```

Contoh:

```
int jumlah;
float harga_satuan, total_harga;
```

Untuk mendeklarasikan konstanta digunakan bentuk:

```
conts <tipedata> <namakonstanta> = <nilaikonstanta>;
```

Contoh:

```
conts float phi= 3.14;
```

b) Memberikan Nilai ke Variabel

Bentuk pernyataan yang digunakan memberikan nilai ke variabel yang telah dideklarasikan :

```
nama_variabel = nilai;
```

Contoh:

```
jumlah = 15;
harga_satuan=7.500;
```

Pada C++, pendefinisian variabel bisa dilakukan dimana saja dalam program. Misalnya bentuk seperti berikut tetap diperkenankan :

```
int i = 10;
cout<<"Nilai i = "<<i<<endl;
int j = 15;
cout<<"Nilai j = "<<j<<endl;
```

c) Variabel String

String pada dasarnya merupakan array (deret) dari karakter. String sangat memudahkan pemrogram. Dengan string kita dapat menampilkan prompt, pesan kesalahan dan berbagai informasi lainnya. Seperti halnya tipe data lain, string juga dapat berupa konstanta atau variabel.

Konstanta string merupakan deretan karakter yang diawali dan diakhiri dengan tanda petik ganda (“). Misalnya “Hai, selamat belajar C++”. Konstanta string berbeda dengan konstanta karakter, “a” tidaklah sama dengan ‘a’.

Contoh program 3.1:

```
//program string
#include <conio.h>
#include <iostream.h>
void main()
{   char teks[20];
    cout<<"Masukkan kata : ";
    cin>>teks;
    cout<<"Kata yang anda masukkan : "<<teks;
    getch();
}
```

Program di atas dapat berjalan dengan normal jika diinputkan sebuah kata. Namun jika terdapat spasi (lebih dari satu kata) maka kata sesudah spasi tidak ditampung dalam variabel string. Ini disebabkan karena operator >> pada **cin** hanya bisa membaca masukan hingga terdapat spasi, tab atau enter. Untuk mengatasi hal tersebut kita dapat menggunakan fungsi getline().

Penggunaan fungsi ***getline*** dapat dituliskan dalam bentuk:

- Untuk data bertipe 'char'
  cara definisinya -----> `char alamat[50];`
  cara menulis di input --------> `cin.getline(alamat,sizeof(alamat));`
- Untuk data bertipe “string”
  cara definisi --------> `string nama;`
  cara menulis di input ---->`getline(cin, nama);`

d) Lingkup Variabel

Lingkup variabel merupakan jangkauan berlakunya suatu variabel. Lingkup variabel ditentukan oleh tempat di mana variabel dideklarasikan. Menurut lingkupnya, variabel dibedakan menjadi dua, yaitu variabel global dan variabel lokal.

1) Variabel Global

Variabel global adalah variabel yang dideklarasikan di luar fungsi, baik fungsi utama maupun fungsi pendukung lainnya. Sifat-sifat variabel global : \

- Dikenal (dapat diakses) oleh semua fungsi.\
- Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
- Dideklarasikan dengan menambahkan kata "**extern**" (opsional).

2) Variabel Lokal

Variabel lokal adalah variabel yang dideklarasikan di dalam fungsi.

Sifat-sifat variabel lokal :

- Secara otomatis akan diciptakan ketika fungsi dipanggil dan akan lenyap ketika proses eksekusi terhadap fungsi berakhir.
- Hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Tidak ada inisialisasi secara otomatis (saat variabel diciptakan nilainya random).
- Dideklarasikan dengan menambahkan kata "**auto**" (opsional).

Berikut ini contoh penggunaan variabel global dan lokal dalam program.

Contoh program 3.2:

```
#include <stdio.h>
// Mendeklarasikan variabel global x
int x = 10;     //memberikan nilai 10 ke dalam variabel x

void NilaiX(void) //Mendefinisikan fungsi lain - NilaiX()
{
  int a=25; //deklarasi variabel lokal
  cout<<"Nilai x dari fungsi  NilaiX() : "<<x<<endl;
  cout<<"Nilai a dari fungsi  NilaiX() : "<<a<<endl;
}
int main(void) //Fungsi utama
{
 int b = 100; //variabel local untuk fungsi utama
```

```
 cout<<"Nilai b dari fungsi main : "<<b<<endl;
 cout<<"Nilai x dari fungsi main : "<<x<<endl;
return 0;
}
```

Hasil eksekusi:

```
Nilai b dari fungsi NilaiX(): 10
Nilai x dari fungsi NilaiX(): 25

Nilai b dari fungsi main : 100
Nilai x dari fungsi main : 25
```

Dari hasil tersebut terlihat bahwa variabel x bersifat global, artinya dia akan dikenali dan dapat diakses oleh setiap fungsi yang terdapat di dalam program (*dalam hal ini fungsi main() dan nNilaiX()*). Pendefinisian fungsi di atas mungkin sedikit membingungkan karena belum pernah ada pembahasan pada bab sebelumnya. Pembahasan lebih lanjut mengenai pendefinisian fungsi selain fungsi utama akan di bahas lebih detil pada Algoritma dan Pemrograman 2.

3.2.3 Mengenal Cout dan Cin

Seperti bahasa pemrograman lainnya, bahasa C juga mempunyai operasi dasar keluaran dan masukkan.

1. ***Cout***

   Pengenal ***cout*** (baca : **c out**) adalah operator C++ yang digunakan sebagai operator keluaran, fungsinya untuk mengarahkan ***data*** ke standar output (layar). Atau dapat juga di tuliskan dengan ***printf.*** Tanda << (dua tanda kurang dari berurutan) adalah operator "penyisipan/ peletakan" yang akan mengarahkan operand (data) yang terletak di sebelah kanannya ke objek yang terletak di sebelah kirinya.

2. ***Cin***

   Pengenal ***cin*** (baca: **c in)**, adalah operator masukkan dalam C untuk membaca data dari standard input (normalnya keyboard). Data masukkan adalah data yang dimasukkan dalam program, diterima dan diproses oleh program.

3. Pustaka input/ output merupakan pustaka yang berisi stream dan fungsi-fungsi input/output. Pustaka ini telah ada di header **stdio.h** dan **iostream.h**. Dalam bahasa C pustaka I/O yang digunakan adalah **stdio.h** yang berisi fungsi-fungsi seperti **printf** (cout) dan **scanf**. Scanf (cin) digunakan untuk memasukkan sebuah nilai ke variable dan printf(cout) digunakan untuk mencetak suatu nilai dari variable maupun konstanta.

Contoh program 3.3 :

```
#include <iostream.h>
void main()
{
   clrscr(); //membersihkan layar
   cout << "Selamat Belajar C++\n";
}
```

Hasil eksekusi :

```
Selamat Belajar C++
```

Pada contoh 3.3 di atas:

"Selamat Belajar C++\n" diarahkan ke cout, yang memberikan hasil berupa tampilan string tersebut ke layar. \ **n**, dapat juga dituliskan **<<endl;** adalah perintah untuk pointer/ karakter untuk pindah baris (new line).

4. Fungsi **getch()** dan **getche()**

Fungsi **getch()** dan **getche()** digunakan untuk membaca tanpa perlu menekan enter, selain itu fungsi ini juga dapat membaca tombol Spasi dan Enter.

Bentuk :

karakter = **getch();**

karakter = **getche();**

Apabila fungsi **getch()** dan **getche()** disertakan maka header conio.h perlu disertakan. Fungsi **getch()** tidak menampilkan karakter dari tombol yang ditekan, sedangkan **getche()** menampilkan karakter dari tombol yang ditekan.

Contoh program 3.4

```
/* contoh penggunaan getch() dan geche()*/
#include<iostream.h>
```

```
#include<conio.h>
void main()
 {
   char kar1, kar2;
   cout<<"Tekan sembarang karakater : ";
   kar1 = getch();
   cout<<"\nTekan sembarang karakater : ";
   kar2 = getche();
   cout<<"\n\nKarakter pertama : "<<kar1<<'\n';
   cout<<"Karakter kedua   : "<<kar2<<'\n';
   getch();
 }
```

Hasil eksekusi:

```
Tekan sembarang karakater :
Tekan sembarang karakater : b

Karakter pertama : a
Karakter kedua   : b
```

Contoh program 3.5

```
[1]   #include <iostream.h>
[2]   #include <stdio.h>
[3]   //program contoh penggunaan cout dan cin
[4]   /*mohon diperhatikan dengan teliti*/
[5]   void main()
[6]   {
[7]   int bil_x;
[8]   float bil_y;
[9]   cout<<"Masukkan bilangan bulat : ";
[10]  cin>>bil_x;
[11]  cout<<"Masukkan bilangan pecahan : ";
[12]  cin>>bil_y;
[13]  cout<<"Bilangan bulat = "<<bil_x<<endl;
```

```
[14]  cout<<"Bilangan pecahan = "<<bil_y<<endl;
[15]  getch();
[16]  }
```

5. Komentar

Perhatikan baris ke 3 dan ke 4 pada contoh program 3.2 diatas. Baris tersebut yang disebut sebagai **komentar**. Komentar diperlukan untuk menjelaskan mengenai program atau bagian-bagian dalam program, atau menuliskan sesuatu sebagai pengingat dalam program.

Pada C++ komentar diawali dengan dua tanda garis miring (//). Ini digunakan untuk komentar pada satu baris. Selain itu, komentar dapat juga diawali dengan tanda /* dan diakhiri dengan tanda */. Ini digunakan untuk komentar yang terdiri dari beberapa baris.

Isi komentar dapat berupa:

- Tujuan/ fungsi program
- Saat program dibuat/direvisi
- Keterangan-keterangan lain tentang kegunaan sejumlah pernyataan dalam program.

Latihan Praktikum - 4

1. Buatlah sebuah program sederhana yang menampilkan data identitas pribadi anda dengan operator keluaran saja (cout)!.

   Nama : (isikan nama anda)
   Tempat Lahir : (isikan kota asal)
   Tanggal lahir : (isikan tanggal lahir)
   Alamat : (alamat rumah)
2. Buatlah program yang menampilkan data identitas pribadi anda dengan operator keluaran dan masukkan!. Anda bisa melakukan save-as dari program no. 1 dan lakukan modifikasi.

BAB 4

OPERATOR & UNGKAPAN

4.1 Operator

Operator merupakan suatu simbol yang digunakan untuk melakukan suatu operasi atau manipulasi, seperti operator penjumlahan, pengurangan, perkalian, pembagian dan operator lainnya. Fungsi operator-operator ini sangat penting, karena tanpa adanya operator kita tidak bisa membuat program perhitungan, program perbandingan dan lain-lain. Dalam operasi setiap operator selalu melibatkan operad. Operand adalah nilai asal yang digunakan didalam proses operasi atau variabel yang akan diproses oleh operator.

Operator-operator pada pemrograman C++ memiliki sifat sebagai berikut:

Sifat	Keterangan	Contoh
Unary	Operator ini hanya melibatkan 1 operand	-1
Binary	Operator ini melibatkan 2 operand	1 + 2
Ternary	Operator ini melibatkan 3 operand	(a>b)?a:b

Macam-macam operator yang di kenal dalam pemrograman C++ adalah:

4.1.1 Operator Aritmatika

Seperti dalam matematika pada umumnya operator artimatika adalah operator yang berfungsi untuk perhitungan matematika seperti pembagian, perkalian, penambahan, pengurangan. Operator aritmatika yang dipakai dalam C++ adalah:

Operator	Deskripsi	Contoh
+	Menambahkan dua operan	A + B
-	untuk operasi aritmatika pengurangan	B - A
*	untuk operasi aritmatika perkalian	A * B
/	untuk operasi aritmatika pembagian	B / A

%	Modulus Operator dan sisa setelah pembagian integer	B % A

Contoh Program 4.1

```
//program contoh penggunaan operator aritmatika
#include <iostream.h>
#include <conio.h>

void main()
{
  clrscr();
  int a = 25;
  int b = 10;

  cout << "a + b = ";
  cout << (a + b) << endl;
  cout << "a - b = ";
  cout << (a – b) <<endl;
  cout << "a x b = ";
  cout << (a * b) <<endl;
  cout << "a / b = ";
  cout << (a / b) <<endl;
  cout << "a % b =”;
  cout << (a % b) <<endl;
  getch();
}
```

Hasil program:

```
  a + b = 35
  a - b = 15
  a x b = 250
  a / b = 2
  a % b = 5
```

4.1.2 Operator Relasional

Operator relasional atau operator relasi merupakan sebuah operator yang bernilai true dan false. Untuk mengevaluasi antara 2 ekspresi, dapat digunakan operator Relasional. Hasil dari operator ini adalah nilai bool yaitu hanya berupa true atau false, atau dapat juga dalam nilai int, 0 untuk mereprensentasikan "false" dan 1 untuk merepresentasikan "true".

Operator	Deskripsi	Contoh (A=25, B=10)
==	digunakan untuk memeriksa apakah kedua nilai atau tidak. jika sama maka kedua kondisi benar	(A==B) false atau (0)
!=	digunakan untuk memerika apakah kedua sama atau tidak. jika tidak sama maka kedua kondisi benar	(A!=B) true atau (1)
>	Memeriksa apakah nilai operan kiri lebih besar dari nilai operan kanan, jika ya maka kondisi menjadi benar.	(A>B) true atau(1)
<	Memeriksa apakah nilai operan kiri kurang dari nilai operan kanan, jika ya maka kondisi menjadi benar.	(A<B) false atau (0)
>=	Memeriksa apakah nilai operan kiri lebih besar dari atau sama dengan nilai operan kanan, jika ya maka kondisi menjadi benar.	(A>=B) true atau (1)
<=	Memeriksa apakah nilai operan kiri kurang dari atau sama dengan nilai operan kanan, jika ya maka kondisi menjadi benar.	(A<=B) false atau (0)

Contoh program 4.2

```
//program contoh penggunaan operator relasi
#include <iostream.h>
#include <conio.h>
void main()
{
  int nilai;
  nilai = 3 > 2;
  cout<<"Nilai = "<<nilai<<endl;
  nilai = 2 > 3;
  cout<<"Nilai = "<<nilai<<endl;
  getch();
}
```

Hasil eksekusi:

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

4.1.3 Operator Logika

Operator Logika adalah operator yang digunakan untuk membandingkan dua nilai variabel atau lebih. Operator ini juga biasa digunakan untuk menghubungkan dua buah ungkapan kondisi menjadi sebuah ungkapan kondisi. Hasil dari operasi ini adalah nilai boolean true atau false.

Operator	Keterangan	Contoh a = true, b = false, c = true
&&	Operator dan/ AND Jika semua operand bernilai benar (TRUE) maka kondisi bernilai benar.	a && b hasilnya false a && c hasilnya true
||	Operator atau/ OR Jika salah satu dari operand bernilai benar (TRUE) maka kondisi bernilai benar.	a || b hasilnya true a || c hasilnya true
!	Operator BUKAN/ NOT Digunakan untuk membalik kondisi. Jika kondisi benar (TRUE) maka akan berubah menjadi salah (FALSE), begitu pula sebaliknya	!a hasilnya false !b hasilnya true !(b && a) hasilnya true

Contoh program 4.3

```
//program contoh penggunaan operator logika
#include <iostream.h>
#include <conio.h>
void main()
{
  int x = 200;
  int nilai;
  nilai = (x > 1) && (x <= 50);
  cout<<"Nilai = "<<nilai<<endl;
  nilai = (x > 1) || (x <= 50);
  cout<<"Nilai = "<<nilai<<endl;
  getch();
}
```

Hasil eksekusi:

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

4.1.4 Operator Bitwase

Operator bitwise digunakan untuk menyelesaikan operasi-operasi bilangan dalam bentuk biner yang dilakukan bit demi bit. Operator bitwise ini berfungsi untuk melakukan manipulasi bit. Operasi ini merupakan hal penting apabila program yang dibuat akan melakukan interaksi dengan perangkat keras (*hardware*). Meski bahasa pemrograman lebih bersifat *data-oriented*, namun perangkat keras masih bersifat *bit-oriented*, sehingga perangkat keras menginginkan input dan output data yang dilakukan terhadapnya tetap dalam bentuk bit tersendiri. Perlu ditekankan di sini bahwa operasi pemanipulasian bit ini hanya dapat dilakukan pada bilangan-bilangan yang bertipe ***char*** dan ***int*** saja, karena keduanya dapat berkoresponden dengan tipe ***byte*** dan ***word*** di dalam bit.

Adapun yang termasuk ke dalam operator bitwise di dalam bahasa C :

Operator	Jenis Operasi	Contoh
&	Bitwise AND	1 & 1 = 1
|	Bitwise OR	1 | 0 = 1
^	Bitwise XOR (Exclusive OR)	1 ^ 1 = 0
~	Bitwise Complements (NOT)	~ 1 = 1
>>	Shift Right (geser kanan)	4 << 1 = 8
<<	Shift Left (geser kiri)	4 >> 1 = 2

Fungsi dari operator &, | dan ~ di atas sebenarnya sama dengan fungsi operator logika &&, || dan !. Perbedaannya hanya operator *bitwise* ini melakukan operasinya bit demi bit, sedangkan operator logika melakukan operasi pada nilai totalnya.

Contoh program 4.4

```
// contoh penggunaan operator bitwise
#include<iostream.h>
#include<conio.h>
void main()
{
  unsigned long a,b,x;
```

```
  a = 50;
  b = 3;
  x = a << b;
  cout<<"Hasil "<<a<<" << "<<b<<" = "<<x<<'\n';
  x = a >> b;
  cout<<"Hasil "<<a<<" >> "<<b<<" = "<<x<<'\n';
  x = a & b;
  cout<<"Hasil "<<a<<" & "<<b<<"  = "<<x<<'\n';
  x = a | b;
  cout<<"Hasil "<<a<<" | "<<b<<"  = "<<x<<'\n';
  x = a ^ b;
  cout<<"Hasil "<<a<<" ^ "<<b<<"  = "<<x<<'\n';
  x = ~a;
  cout<<"Hasil ~"<<a<<"     = "<<x<<'\n';
  getch();
}
```

Hasil eksekusi:

```
  Hasil 50 << 3 = 400
  Hasil 50 >> 3 = 6
  Hasil 50 & 3  = 2
  Hasil 50 | 3  = 51
  Hasil 50 ^ 3  = 49
  Hasil ~50     = 4294967245
```

4.1.5 Operator Assignment

Operator Assignment disebut operator **penugasan,** berupa simbol sama dengan (=). Operator penugasan ini berguna untuk memberikan nilai ke suatu variabel.

Contoh :

```
  a = 5;
```

yang berarti memberikan integer 5 ke variabel a. Sisi kiri dari operator disebut ***Lvalue (Left Value)*** dan sisi kanan disebut ***Rvalue (Right Value)***. Lvalue harus selalu berupa variabel dan sisi kanan dapat berupa konstanta, variabel, hasil dari suatu operasi atau kombinasi dari semuanya. Contoh lain :

```
a = 2 + (b = 5);
```

sama dengan :

```
b = 5;
a = 2 + b;
```

Berikut beberapa bentuk penggunaan operator penugasan dalam struktur pemrograman C++:

Operator	Deskripsi
=	operator penugasan sederhana, untuk menetapkan nilai dari sisi operand kanan ke operand kiri
+=	tambahkan AND operator penugasan, menambahkan operand yang benar untuk operand kiri dan menetapkan hasil untuk operand kiri
-=	kurangi AND operator penugasan, itu mengurangi operand kanan dari operand kri dan menetapkan hasil untuk operand kiri
*=	kalikan AND operator penugasan, mengalihkan operand kanan dengan operand kiri dan menetapkan hasil untuk operand kiri
/=	Bagikan AND operator penugasan, membagi operan kiri dengan operand kanan dan menetapkan hasil untuk operand kiri
%=	Modulus AND operator penugasan, sisa bagi dengan menggunakan dua operand dan menetapkan hasil untuk operand kiri
<<=	shift kiri AND operator penugasan
>>=	shift kanan AND operator penugasan
&=	Bitwise AND operator penugasan
^=	bitwise XOR dan operator penugasan
|=	bitwise inclusive OR dan operator penugasan

Contoh penggunaan yang jelas untuk jenis operator poenugasan ini adalah operator majemuk.

4.1.6 Operator Majemuk

Operator majemuk adalah operator yang digunakan untuk memendekkan operasi penugasan, misalnya:

```
x = x + 2;
y = y * 4;
```

dapat dituliskan dengan bentuk:

```
x += 2;
y *= 4;
```

bentuk seperti ini berlaku untuk semua operator yang bersifat binary.

4.1.7 Operator Precedence

Operator Precedence merupakan operator yang didahulukan, menentukan derajat (prioritas) pengerjaan operator, mana perhitungan yang ingin dilakukan terlebih dahulu. Ada urutan standar/ default prioritas perhitungan, tetapi kita dapat menentukan operasi mana yang akan didahulukan dengan mengubah urutan tersebut dengan menggunakan tanda kurung.

Missal :

```
a = 4 + 8 / 2
```

Maka nilai a adalah 8. Mengapa bukan 6 ?, karena pada C++ pengerjaan operasi di lakukan dari level yang tinggi ke level yang lebh rendah. Untuk mendapatkan nilai 6, kita dapat mengubah urutan prioritas sebagai berikut:

```
a = 4 + (8 / 2)
```

Berikut ini adalah prioritas operator dari tinggi ke rendah :

Kategori	Operator	Associativity
Postfix	() [] -> . ++ - -	kiri ke kanan
Unary	+ - ! ~ ++ - - (type)* & sizeof	kanan ke kiri
Multiplicative	* / %	kiri ke kanan
Additive	+ -	kiri ke kanan
Shift	<< >>	kiri ke kanan
Relational	< <= > >=	kiri ke kanan
Equality	== !=	kiri ke kanan
Bitwise AND	&	kiri ke kanan
Bitwise XOR	^	kiri ke kanan
Bitwise OR	|	kiri ke kanan
Logical AND	&&	kiri ke kanan
Logical OR	||	kiri ke kanan
Conditional(kondisi)	?:	kanan ke kiri
Assignment (penugasan)	= += -= *= /= %=>>= <<= &= ^= |=	kanan ke kiri
Comma(koma)	,	kiri ke kanan

Contoh lain:

```
5 + 20 * 2 - 20 / 2 + (10 - 2)
```

Maka:

- Ekspresi numerik yang terdapat di dalam kurung buka tutup akan dikerjakan pertama kali karena mempunyai prioritas paling tinggi dibandingkan dengan operator perkalian (*), pembagian (/), penjumlahan (+) dan pengurangan (-) sehingga ekspresi numerik akan menjadi 5 + 20 * 2 – 20 / 2 + 8.
- Operator perkalian (*) dan pembagian (/) akan dikerjakan berikutnya mulai dari kiri ke kanan karena memiliki prioritas yang sama dan lebih tinggi dari operator penjumlahan (+) dan pengurangan (-) sehingga ekspresi numerik menjadi 5 + 40 – 10 + 8.
- Operator penambahan dan pengurangan mempunyai prioritas yang sama dan akan dikerjakan terakhir mulai dari kiri ke nanan sehingga menghasilkan nila 43.

4.1.8 Operator Penaikan dan Penurun

Operator penaikan dan penurunan merupakan operator yang digunakan untuk melakukan penaikkan atau penurunan nilai secara otomatis saat program dijalankan. Operator ini digunakan pada operand bertipe bilangan bulat, yaitu:

- Operator ++ untuk penaikan
  Digunakan untuk menaikkan nilai variabel sebesar satu.
- Operator -- untuk penurunan
  Digunakan untuk menurunkan nilai variabel sebesar satu.
- Contoh:
  x++; atau ++x; ➔ identik dengan x = x + 1;
  y--; atau --y; ➔ identik dengan y = y – 1;a

Bentuk operator penaikan dan penurunan:

- ***Post Increment***

  S = 10 + R++;

  identik dengan : S = 10 + R;
  R = R + 1;

  S = 10 + R--;

  identik dengan : S = 10 + R;
  R = R - 1;

- ***Pree Increment***

  S = 10 + ++R;

  identik dengan : R = R + 1;

S = 10 + R;

S = 10 + --R;

identik dengan : R = R - 1;

S = 10 + R;

4.1.9 Operator Misc

Operator misc adalah operator tambahan yang sering dipakai dalam pemrograman, terutama pada pemrograman array. Misalnya operator ***sizeof*** dan ***pointer (*).***

Operator	Description
sizeof	size of operator digunakan untuk mengetahui ukuran dari memori
Condition ? X : Y	operator kondisi Jika Kondisi ini benar? maka kembali nilai X: jika nilai Y
,	Nilai seluruh ekspresi koma adalah nilai ekspresi terakhir dari daftar dipisahkan koma
Cast	digunakan untuk mengkonversi type data
&	pointer & digunakan untuk mengembalikan alamat dari variabel
*****	opinter * digunakan untuk mengetahui alamat memori

Contoh penggunaan operator misc adalah operator kondisi. Operator kondisi biasa dipakai untuk mendapatkan sebuah nilai dari dua buah kemungkinan, berdasarkan suatu kondisi. Ada tiga ungkapan yang dilibatkan. Oleh karena itu operator **?:** tergolong operator ternary. Bentuk pemakaian operator kondisi adalah:

```
ungkapan1? Ungkapan2 : ungkapan3;
```

hasil dari ungkapan tersebut adalah:

- Jika ungkapan1 bernilai benar (1) maka hasilnya adalah ungkapan2.
- Jika ungkapan1 bernilai salah (0) maka hasilnya adalah ungkapan3.

Contoh program 4.5

```
//contoh program penggunaan operator kondisi
#include <iostream.h>
#include <conio.h>
void main()
{
```

```
    int bil1, bil2, minim;
    bil1 = 53;
    bil2 = 6;
    minim = bil1 < bil2 ? bil1 : bil2;
    cout<<"Bilangan terkecil = "<<minim;
    getch();
}
```

Latihan Praktikum - 5

1. Buatlah flowchart algoritma untuk menampilkan deret bilangan 1 – 100 yang habis dibagi dengan 2 dan habis dibagi dengan 3.
2. Buatlah pseudocode algoritma dari soal no. 1 diatas!

4.2 Ungkapan

Ungkapan disebut juga dengan ekspresi. Dalam C++ ungkapan dapat berupa pengenal, konstanta, atau diantara kombinasi pengenal dan konstanta dengan operator.

Contoh ungkapan:

```
a = b + c – 2;
```

maka: a, b, c merupakan operand, simbol =, + ,- adalah operator. Dalam hal ini variabel **a** diisi hasil penjumlahan **b** dan **c** dikurangi **2**. Selanjutnya nilai **a** menyatakan nilai ungkapan.

Operator Aritmatika dalam ungkapan:

Sifat	Operator	Keterangan	Contoh
Unary	-	Tanda Minus	-2
	+	Tanda Plus	**+4**
Binary	*	Perkalian	2 * 3
	/	Pembagian	7 / 2
	%	Sisa Pembagian (modulus)	7 % 2
	+	Penjumlahan	2 + 5
	-	Pengurangan	10 - 5

Seperti halnya pada matematika, dalam pemrograman operator juga mempunyai derajat prioritas yang sama. Operator dengan prioritas tinggi akan diutamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki proritas lebih rendah.

Ungkapan Kondisi

Ungkapan kondisi adalah ungkapan yang menjadi dasar bagi pernyataan berkondisi (misalnya **if**). Ungkapan ini memberikan nilai benar atau salah.

Hasil ungkapan berupa :

0 kalau ungkapan bernilai salah

1 kalau ungkapan bernilai benar

Adapun elemen yang dapat membentuk ungkapan ini operator relasi dan operator logika.

4.3 Manipulator

Manipulator adalah fungsi pembantu yang memungkinkan untuk mengontrol input/ output stream. Manipulator pada umumnya digunakan untuk mengatur tampilan layar. Contohnya untuk mengatur supaya suatu nilai ditampilkan dengan lebar 8 karakter dan diatur rata kiri terhadap lebar tersebut. Dalam C++, terdapat beberapa manipulator yang merupakan fitur baru, yang baru ditambahkan. Hal ini karena compiler C++ klasik (belum distandarisasi) tidak mendukung adanya manipulator. Adapun manipulator yang dimaksud disini adalah seperti yang terlihat pada tabel dibawah ini:

Manipulator	Kegunaan	Operasi
boolalpha	Mengaktifkan flag	Input / Output
dec	Mengaktifkan flag dec	Input / Output
endl	Menampilkan baris baru dan membuat stream	Output
ends	Menampilkan null	Output
fixed	Mengaktifkan flag fixed	Output
flush	Membuang stream	Output
hex	Mengaktifkan flag hex	Input / Output
internal	Mengaktifkan flag internal	Output
left	Mengaktifkan flag left	Output
noboolalpha	Mematikan flag boolalpha	Input / Output
noshowbase	Mematikan flag showbase	Output

noshowpoint	Mematikan flag showpoint	Output
noshowpos	Mematikan flag showpos	Output
noskipws	Mematikan flag skipws	Input
nounitbuf	Mematikan flag unitbuf	Output
nouppercase	Mematikan flag uppercase	Output
oct	Mematikan flag oct	Input / Output
resetiosflags (fmtflags f)	Mematikan flag yang dituliskan (f)	Input / Output
right	Mengaktifkan flag right	Output
scientific	Mengaktifkan flag scientific	Output
setbase (int base)	Mengaktifkan basis base	Input / Output
setfill (int ch)	Mengisi karakter dengan ch	Output
setiodflags (fmtflags f)	Mengaktifkan flag yang dituliskan (f)	Input / Output
setprecision (int p)	Menentukan presisi digit	Output
setw (int w)	Menentukan lebar kolom	Output
showbase	Mengaktifkan flag show base	Output
showpoint	Mengaktifkan flag show point	Output
showpos	Mengaktifkan flag showpos	Output
skipws	Mengaktifkan flag skipws	Input
unitbuf	Mengaktifkan flag unitbuf	Output
uppercase	Mengaktifkan flag uppercase	Output
ws	Mengabaikan karakter white-space	Input

Untuk menggunakan manipulator yang tercantum diatas harus menyertakan file header <iomanip> dan <ionsteam>. Berikut ini penjelasan mengenai beberapa manipulator yang disediakan oleh C++ :

1. Manipulator ***endl***

   Manipulator endl digunakan untuk menyisipkan karakter ganti baris (*newline*). Manipulator ini identik dengan karakter '\n'.

Contoh program 4.6

```
/* contoh penggunaan endl */
#include<iostream.h>
#include<conio.h>
```

```
void main()
{
  int angka1 = 22;
  int angka2 = 555;
  float total = 12345.67;
  cout<<"Isi angka 1 : "<<angka1<<endl;
  cout<<"Isi angka 2 : "<<angka2<<endl;
  cout<<"Isi angka 2 : "<<total<<endl;
  getch();
}
```

Hasil program:

```
Isi angka 1 = 22
int angka 2 = 555
Isi angka 2 = 12345.67
```

2. Manipulator ***setw()***

Manipulator **setw()** digunakan untuk mengatur lebar tampilan data pada layar. Jika anda akan mengatur tampilan data pada layar maka perlu disertakan header **iomanip.h.**

Contoh program 4.7

```
/* contoh penggunaan setw()*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  int angka = 7775;
  cout<<setw(0)<<angka<<endl;
  cout<<setw(5)<<angka<<endl;
  cout<<setw(7)<<angka<<endl;
  getch();
}
```

Hasil program:

```
7775
 7775
   7775
```

3. Manipulator ***setfill()***

Manipulator **setfill()** digunakan untuk mengatur karakter yang diisikan pada *field* yang ditentukan oleh **setw()** yang tidak digunakan untuk menampilkan data.

Contoh program 4.8

```
/* contoh penggunaan setw() dan setfill()*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
    int angka = 7775;
    cout<<setw(0)<<setfill('*')<<angka<<endl;
    cout<<setw(4)<<setfill('*')<<angka<<endl;
    cout<<setw(6)<<setfill('*')<<angka<<endl;
    cout<<setw(8)<<setfill('*')<<angka<<endl;
    getch();
}
```

Hasil program:

```
7775
*7775
**7775
****7775
```

4. Manipulator ***dec***, ***oct*** dan ***hex***

Manipulator ***dec, oct*** dan ***hex*** digunakan untuk menampilkan data dalam bentuk desimal (basis 10), oktal (basis 8), dan heksadesimal (basis 16).

Contoh program 4.9

```
/* contoh penggunaan dec, oct dan hex*/
```

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  int x = 100;
  cout<<"Nilai x : "<<x<<endl;
  cout<<"Nilai x dalam oktal        : "<<oct<<x<<endl;
  cout<<"Nilai x dalam heksadesimal : "<<hex<<x<<endl;
  cout<<"Nilai x dalam desimal      : "<<dec<<x<<endl;
  getch();
}
```

Hasil eksekusi :

```
Nilai x                   : 100
Nilai x dalam octal       : 144
Nilai x dalam heksadesimal : 64
Nilai x dalam decimal     : 100
```

5. Manipulor ***setiosflags()***

Manipulor setiosflags() merupakan manipulator yang dapat dipakai untuk mengontrol sejumlah tanda format sebagai berikut :

Tanda Format	**Keterangan**
ios::left	: untuk set rata kiri terhadap lebar *filed* yang diatur melalui **setw()**
ios::right	: untuk rata kanan terhadap lebar *filed* yang diatur melalui **setw()**
ios::scientific	: untuk format keluaran dalam notasi eksponensial
ios::fixed	: untuk format keluaran dalam bentuk notasi desimal
ios::dec	: untuk format keluaran dalam basis 10 (desimal)
ios::oct	: untuk format keluaran dalam basis 8 (oktal)
ios::hex	: untuk format keluaran dalam basis 16 (heksadesimal)
ios::uppercase	: untuk format huruf pada notasi heksadesimal dalam bentuk kapital
ios::showbase	: untuk menampilkan awalan 0x untuk bilangan heksadesimal atau 0 (nol) untuk bilangan octal

ios::showpoint : untuk menampilkan titik desimal pada bilangan pecahan yang tidak memiliki bagian pecahan

ios::showpos : Untuk menampilkan tanda + pada bilangan positif

Contoh program 4.10

```
/* contoh penggunaan setiosflags()*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
 int x = 520;
 cout<<"x ditampilkan dengan ios::left "<<endl;
 cout<<setiosflags(ios::left)<<setw(10)<<x<<endl<<endl;
 cout<<"x ditampilkan dengan ios::right "<<endl;
 cout<<setiosflags(ios::right)<<setw(10)<<x<<endl<<endl;

 float y = 123.456;
 cout<<"y ditampilkan dengan ios::fixed "<<endl;
 cout<<setiosflags(ios::fixed)<<y<<endl<<endl;
 cout<<resetiosflags(ios::fixed);

 cout<<"y ditampilkan dengan ios::scientific "<<endl;
 cout<<setiosflags(ios::scientific)<<y<<endl<<endl;
 cout<<resetiosflags(ios::scientific);

 int bil = 51;
 cout<<"Tanpa ios::showbase : "<<endl;
 cout<<oct<<bil<<endl;
 cout<<dec<<bil<<endl;
 cout<<hex<<bil<<endl;
 cout<<"Dengan ios::showbase : "<<endl;
 cout<<setiosflags(ios::showbase);
 cout<<oct<<bil<<endl;
 cout<<hex<<bil<<endl;
```

```
    cout<<dec<<bil<<endl;

    cout<<"\nDengan ios::uppercase untuk heksadesimal: "<<endl;
    cout<<setiosflags(ios::uppercase)<<bil<<endl;
    cout<<resetiosflags(ios::showbase);
    cout<<resetiosflags(ios::uppercase);

    float a = 234.00;
    cout<<"\nTanpa ios::showpoint : "<<endl;
    cout<<a<<endl<<endl;

    cout<<"Dengan ios::showpoint : "<<endl;
    cout<<setiosflags(ios::showpoint)<<a<<endl;
    cout<<resetiosflags(ios::showpoint);

    int b = 27;
    cout<<"\nTanpa ios::showpos : "<<endl;
    cout<<b<<endl<<endl;

    cout<<"Dengan ios::showpos : "<<endl;
    cout<<setiosflags(ios::showpos)<<b<<endl;
    cout<<resetiosflags(ios::showpos);
    getch();
}
```

Hasil eksekusi :

```
x ditampilkan dengan ios::left
520
x ditampilkan dengan ios::right
520
y ditampilkan dengan ios::fixed
123.456001
y ditampilkan dengan ios::scientific
1.234560e+02
Tanpa ios::showbase :
```

```
63
51
33
Dengan ios::showbase :
063
0x33
51
Dengan ios::uppercase untuk heksadesimal :
51
Tanpa ios::showpoint :
234
Dengan ios::showpoint :
234.000
Tanpa ios::showpos :
27
Dengan ios::showpos :
+27
```

6. Manipulator ***resetiosflags()***

Manipulator resetiosflags() digunakan untuk mengembalikan ke bentuk format awal setelah adanya penggunaan manipulator, misalnya :

```
setiosflags(ios::left)
```

telah digunakan, maka untuk kembali ke bentuk awal bisa digunakan :

```
resetiosflags(ios::left)
```

Contoh program 4.11

```
//contoh penggunaan resetiosflags()
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
 int x = 12345;
 cout<<"Ditampilkan dengan ios::right "<<endl;
```

```
 cout<<setiosflags(ios::left)<<setw(10)<<x<<endl<<endl;
 cout<<"Setelah dilakukan resetiosflags(ios::right)"<<endl;
 cout<<resetiosflags(ios::left);
 cout<<setw(10)<<x<<endl<<endl;
 getch();
}
```

Hasil eksekusi :

```
Ditampilkan dengan ios::right
12345
Setelah dilakukan resetiosflags(ios::right)
     12345
```

7. Manipulator *setprecision()*

Manipulator *setprecision()* digunakan untuk mengatur jumlah digit pecahan yang akan ditampilkan pada bilangan pecahan.

Bentuk :

```
setprecision(n)
```

dengan *n* adalah jumlah digit yang ingin ditampilkan

Contoh program 4.12

```
//contoh penggunaan setprecision()
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
 float y = 12.340;
 cout<<setiosflags(ios::fixed);
 cout<<setprecision(0)<<y<<endl;
 cout<<setprecision(1)<<y<<endl;
 cout<<setprecision(2)<<y<<endl;
 cout<<setprecision(3)<<y<<endl;
 cout<<setprecision(4)<<y<<endl;
```

```
    cout<<setprecision(5)<<y<<endl;
    getch();
}
```

Hasil eksekusi :

```
12
12.3
12.34
12.340
12.3400
12.34000
```

Latihan Praktikum - 6

1. Buatlah program dari hasil algoritma yang anda buat pada tugas (1) untuk no. 2 dan No. 3. Tampilkan hasilnya dengan baik
2. Buatlah sebuah program yang menampilkan data mahasiswa kelas anda yang terdiri dari npm, nama, alamat asal, sekolah_asal. Gunakan fungsi *getline*. Tampilkan data dalam format tabel dengan fungsi-fungsi manipulator.

BAB 5
PERNYATAAN DASAR DALAM C++

Pernyataan *(statement)* adalah baris perintah atau instruksi yang digunakan untuk melakukan suatu tindakan. Pernyataan bisa terdiri dari satu baris atau lebih (satu blok) yang disebut blok pernyataan. Blok pernyataan merupakan sekumpulan baris program (instruksi) yang berada di dalam kurung kurawal. Blok pernyataan sering juga disebut sebagai pernyataan majemuk. Contohnya:

```
{
 a = 15;
 b = a + 20;
 c = b – 5;
}
```

adalah blok pernyataan yang dapat diperlakukan sebagai blok tunggal atau blok yang setingkat. Jika dalam suatu blok pernyataan didefinisikan suatu pengenal maka pengenal tersebut hanya dikenal didalam blok itu saja dan blok yang ada didalamnya. Perngenal tidak akan dikenal diblok diluar definisi dari pengenalnya.

Contoh program 5.1

```
//contoh penggunaan blok_penyataan/pernyataan majemuk)
#include <iostream.h>
void main()
{  int a = 5;
   cout<<"a = "<<a<<endl;
   {
      int a;   //a hanya dikenal di blok ini
```

```
        a = 20;
        cout<<"a = "<<a<<endl;
      }
      cout<<"a = "<<a<<endl;
   getch();
  }
```

Macam-macam pernyataan dalam pemrogramaan C++, yaitu:

5.1 Pernyataan Ungkapan dan Pernyataan Deklarasi (Definisi)

Pernyataan ungkapan terdiri dari sebuah ungkapan dan diakhiri dengan tanda titik koma (;). Biasanya berupa penugasan nilai terhadap variabel atau pemanggilan fungsi. Contoh :

```
bil = 10;
x++;
jumlah = hasil * 15;
```

Sedangkan pernyataan deklarasi/ definisi digunakan untuk memperkenalkan nama variabel ataupun pengenal yang lain beserta tipe datanya. Contoh :

```
int nilai;
float discount;
```

baris diatas merupakan contoh pendefinisian variabel *nilai* dengan tipe **int,** dan variabel *discount* dengan tipe float.

5.2 Pernyataan goto dan Pernyataan Berlabel

Pernyataan goto adalah pernyataan yang digunakan untuk mengarahkan eksekusi program ke pernyataan berlabel. Pernyataan berlabel yaitu pernyataan yang mengandung nama label dan titik dua (:).Dalam hal ini, *label* berupa suatu pengenal yang penamaanya juga mengikuti aturan nama pengenal.

Contoh:

```
goto nama_label;
……
nama_label:
 pernyataan;
```

Contoh program 5.2

```
//program contoh penggunaan pernyataan goto dan label
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
   cout<<"Tes goto"<<endl;
   goto selesai;
   cout<<"Pernyataan ini tidak ditampilkan"<<endl;
   selesai: //nama label
   cout<<"Selesai..."<<endl;
   getch();
}
```

Hasil eksekusi:

Tes goto

Selesai …

Pemakaian goto sebisa mungkin dihindari karena pernyataan ini cenderung membuat program menjadi rumit dan sulit dipahami. Penggunaan goto sering kali menyulitkan dalam logika program dan jika terjadi kesalahan penelusuran kesalahan akan menjadi rumit, apalagi kalau program yang dibuat besar.

5.3 Pernyataan Penyeleksian Kondisi

Pernyataan seleksi kondisi melakukan pengujian untuk memilih satu dari beberapa alternatif yang tersedia. Seleksi kondisi, haruslah dapat menghasilkan nilai benar (*true*) atau nilai salah (*false*). Jika hasil seleksi kondisi bernilai benar, maka suatu pernyataan baru akan dikerjakan.

5.3.1 Pernyataan If ...

Pernyataan if digunakan untuk mengambil keputusan berdasarkan suatu kondisi yang diuji, merupakan syarat pengambilan keputusan. Ada 3 (tiga) bentuk if dalam C++.

1. **Pernyataan if sederhana**

   Pada bentuk ini, pernyataan if hanya mempunyai satu kondisi atau satu kemungkinan pernyataan yang akan dikerjakan jika kondisi (syarat) yang diuji bernilai benar.

   Bentuk pernyataan if sederhana :

```
if (kondisi)
 pernyataan;
```

Contoh program 5.3:

```
//program contoh penggunaan if sederhana
#include <iostream.h>
#include <conio.h>
void main()
{
   int usia;
   cout<<"Berapa usia anda ? ";
   cin>>usia;
   if (usia < 17)
       cout<<"Anda tidak boleh menonton."<<endl;
   getch();
}
```

Hasil eksekusi:

```
Berapa Usia anda ? 15
Anda tidak boleh menonton.
```

5.3.2 Pernyataan if ... else ...

Pernyataan seleksi kondisi dengan bentuk if … else … mempunyai dua kemungkinan pernyataan yang akan dikerjakan berdasarkan hasil pengujian kondisi. Bentuk pernyataannya:

```
if <kondisi>
   pernyataan1;
else
   pernyataan2;
```

Pada penyataan ini, <*kondisi*> digunakan untuk menentukan hasil pengujian, jika <*kondisi*> bernilai benar (*true*) maka *pernyataan1* akan dikerjakan, jika <*kondisi*> bernilai salah (*false*) maka *pernyataan2* yang akan dikerjakan.

Dalam bentuk algoritma dapat dicontohkan:

Pseudocode peserta_pemilih;

```
  //seorang warga Negara boleh ikut pemilu jika usia >=17
  //jika usia<17 maka tidak boleh memilih
  //deklarasi
  int usia_warga;
  //deskripsi
  {
    baca(usia_warga;
    if (usia_warga >=17)
     cetak("silahkan Memilih")
    else
     cetak("Anda belum bisa memilih")
  }
```

Contoh program 5.4:

```
//contoh program yang mengandung pernyataan if...else
#include<iostream.h>
#include<conio.h>
void main()
{
   int usia_warga;
   cout<<"Masukkan Usia Warga : "; cin>>usia_warga;
   if (usia_warga>=17)
      cout<<"Silahkan Memilih…"<<endl;
   else
      cout<<"Anda belum bisa memilih"<<endl;
   getch();
}
```

Hasil eksekusi :

```
Masukkan nilai : 25
Silahkan Memilih…

Masukkan nilai : 15
Anda belum bisa memilih
```

5.3.3 Pernyataan *nested if* (if bersarang)

Pada bentuk *nested if*, pernyataan **if** memiliki banyak kemungkinan pernyataan dan memiliki banyak pengujian kondisi untuk mengerjakan pernyataan.

Bentuk umum pernyataan **nested if :**

```
if <kondisi1>
pernyataan1;
else if <kondisi2>
   pernyataan2;
else if <kondisiM>
   pernyataanM;
else
   pernyataanN;
```

Pada pernyataan ini <*kondisi1*> diuji, jika hasil pengujian bernilai benar maka pernyataan1 dikerjakan, jika hasil pengujian <*kondisi1*> bernilai salah maka akan diuji <*kondisi2*>, jika hasil pengujian bernilai benar maka *pernyataan2* akan dikerjakan, jika hasil pengujian <*kondisi2*> bernilai salah maka <*kondisiM*> akan diuji, jika hasil pengujian bernilai benar maka <*pernyataanM*> akan dikerjakan, jika hasil pengujian <*kondisiM*> bernilai salah maka akan mengerjakan <*pernyataanN*> yang merupakan alternatif terakhir jika semua kondisi yang diuji tidak terpenuhi.

Pseudocode bangun_datar

```
//ditampilkan menu pilihan untuk hitung luas bangun datar
//dilakukan dengan menggunakan nested if
//deklarasi
char pilih
int sisi, panjang, lebar, alas, tinggi;
float luas

//deskripsi
{
  cetak (" Hitung Luas Bangun Datar")
  cetak (" ========================")
```

```
cetak ("1. Luas Kubus             ")
cetak ("2. Luas Persegi Panjang   ")
cetak ("3. Luas Segitiga          ")
cetak ("4. Selesai                ")
cetak (" Pilihan Anda[1..4] :     ")
baca(pilih)
if(pilih=='1')
{
  cetak("Kubus")
  baca(sisi)
  luas = sisi * sisi
}
else if(pilih=='2')
     {
       cetak("Persegi Panjang")
       baca(panjang)
       baca(lebar)
       luas = panjang * lebar
     }
     else if(pilih=='3')
          {
            cetak("Segitiga)
            baca(alas)
            baca(tinggi)
            luas = (0.5) * alas * tinggi
          }
          else
          {
             cetak("Selesai")
          }
  cetak(luas)
}
```

Contoh program 5.5

```
//program contoh penggunaan nested if
//untuk hitung luas bangun datar
#include <iostream.h>
#include <conio.h>
void main()
{
  char pilih;
  int sisi, panjang, lebar, alas, tinggi;
  float luas;

  cout<<" Hitung Luas Bangun Datar "<<endl;
  cout<<" ======================== "<<endl;
  cout<<"1. Luas Kubus             "<<endl;
  cout<<"2. Luas Persegi Panjang   "<<endl;
  cout<<"3. Luas Segitiga          "<<endl;
  cout<<"4. Selesai                "<<endl;
  cout<<" Pilihan Anda[1..4]  :    "<<endl;
  cin>>pilih;

  if(pilih=='1')
  {
    cout<<"Kubus"<<endl;
    cout<<"Masukkan sisi kubus : "; cin>>sisi;
    luas = sisi * sisi ;
  }
  else if(pilih=='2')
       {
         cout<<"Persegi Panjang"<<endl;
         cout<<"Masukkan Panjang : "; cin>>panjang;
         cout<<"Masukkan Lebar : "; cin>>lebar;
         luas = panjang * lebar;
       }
```

```
    else if(pilih=='3')
         {
           cout<<"Segitiga"<<endl;
           cout<<"Masukkan Alas : "; cin>>alas;
           cout<<"Masukkan Tinggi     : "; cin>>tinggi;
           luas = (0.5) * alas * tinggi;
         }
         else
         {
            cetak("Selesai"<<endl;
         }
  cetak(luas)
}
```

5.3.4 Pernyataan Swicth

Pernyataan **switch** digunakan untuk menjalankan salah satu pernyataan dari beberapa kemungkinan pilihan. Permilihan pada pernyataan **switch** berdasarkan nilai dari ungkapan dan nilai dari penyeleksi.

Bentuk pernyataan **switch** :

```
switch (ungkapan)
{
   case ungkapan1 :
        pernyataan1;
        break;
   case ungkapan2 :
        pernyataan2;
        break;
   ...
   default :
        pernyataanX;
}
```

Pada pernyataan ***switch***, *(ungkapan)* dapat berupa konstanta atau variabel, sedangkan *ungkapan1*, *ungkapan2*, dapat berupa konstanta bertipe ***int*** atau ***char***. Proses pencocokan

(ungkapan) dengan *ungkapan1*, *ungkapan2* dilakukan berurutan mulai *ungkapan1*, *ungkapan2* dan seterusnya. Jika cocok maka pernyataan yang mengikuti ***case*** akan dikerjakan. Eksekusi akan berakhir jika ditemukan pernyataan ***break***. Pernyataan ***default*** akan dikerjakan jika ungkapan dalam **case** tidak ada yang cocok dengan ungkapan dalam ***switch***.

Pseudocode Operasi_duaBilangan

```
//ditampilkan menu pilihan untuk operasi hitung dua bilangan
//deklarasi
  char pilih
  int bil_A, int bil_B
  float hasil
//deskripsi
{
  cetak (“   Menu Pilihan Operasi Hitung     ”)
  cetak (“ ====================== “)
  cetak (“1. Penjumlahan Bilangan       “)
  cetak (“2. Perkalian Bilangan         “)
  cetak (“3. Pengurangan Bilangan       “)
  cetak (“4. Pembagian Bilangan         “)
  cetak (“ 5. Selesai                   “)
  cetak (“ Pilihan Anda [ 1 .. 5]  :    “)
  baca (pilih)
  switch (pilih)
  case 1:
     baca bil_A
     baca bil_B
     hasil = bil_A + bil_B
  case 2:
     baca bil_A
     baca bil_B
     hasil = bil_A * bil_B
  case 3:
     baca bil_A
     baca bil_B
```

```
     hasil = bil_A - bil_B
  case 4:
     baca bil_A
     baca bil_B
     hasil = bil_A / bil_B
  case 5: break;
}
```

Contoh program 5.6

```
//program contoh penggunaan switch
#include <iostream.h>
#include <conio.h>
void main()
{
  char pilih;
  int bil_A, int bil_B
  float hasil
{
  cout<<"  Menu Pilihan Operasi Hitung"<<endl;
  cout<<" ============================"<<endl;
  cout<<"1. Penjumlahan Bilangan   "<<endl;
  cout<<"2. Perkalian Bilangan     "<<endl;
  cout<<"3. Pengurangan Bilangan   "<<endl;
  cout<<"4. Pembagian Bilangan     "<<endl;
  cout<<" 5. Selesai               "<<endl;
  cout<<" Pilihan Anda [ 1 .. 5] : "<<endl;
  cin>>pilih;
  switch (pilih)
  {
    case '1':
    {
     cout<<"Masukkan Bilangan pertama : "; cin>bil_A;
     cout<<"Masukkan Bilangan pertama : "; cin>>bil_B;
```

```
     hasil = bil_A + bil_B;
     cout<<"Hasil penjumlahan = "<<hasil<<endl;
     break;
    }
    case '2':
     {
     cout<<"Masukkan Bilangan pertama : "; cin>bil_A;
     cout<<"Masukkan Bilangan pertama : "; cin>>bil_B;
     hasil = bil_A * bil_B;
     cout<<"Hasil perkalian = "<<hasil<<endl;
     break;
    }
    case '3':
     {
     cout<<"Masukkan Bilangan pertama : "; cin>bil_A;
     cout<<"Masukkan Bilangan pertama : "; cin>>bil_B;
     hasil = bil_A - bil_B;
     cout<<"Hasil pengurangan = "<<hasil<<endl;
     break
    }
    case '4':
     {
     cout<<"Masukkan Bilangan pertama : "; cin>bil_A;
     cout<<"Masukkan Bilangan pertama : "; cin>>bil_B;
     hasil = bil_A / bil_B;
     cout<<"Hasil pembagian = "<<hasil<<endl;
     break;
    }
    case '5':  break;
}
getch();
}
```

Tugas (2) – Fungsi if

Buatlah program untuk menghitung biaya sewa gedung dengan data inputan nama penyewa, kategori_gedung, lama sewa (hari), kelebihan jam. Dimana terdapat 3 kategori gedung:

a. Gedung Pertemuan (Rapat) dengan harga sewa per hari Rp.3.000.000, harga sewa lebih jam sebesar Rp.150.000/jam.
b. Gedung Pentas Seni, harga sewa per hari Rp.850.000, harga sewa lebih jam sebesar Rp.100.000.
c. Gedung Pernikahan, per hari Rp.2.000.000, sewa jam sebesar Rp.200.000.

Pajak Sewa sebesar 2% di tanggung oleh penyewa. Potongan biaya sewa akan diberikan kepada penyewa yang lebih dari 3 hari sebesar sebesar 10% dari total sewa. Total Sewa dihitung dari harga_sewa per hari*jumlah hari sewa. Biaya lebih jam didapat dari harga lebih jam*jumlah jam. Gunakan constanta untuk biaya sewa per hari dan biaya lebih jam.

Tampilkan data dalam bentuk tabel dengan fungsi-fungsi manipulator.

Tugas (3) - fungsi switch

Buatlah program untuk menghitung pembayaran pembeli pada agen mie instan. Dengan data inputan nama pembeli, jenis/merk mie, jumlah_beli. Dimana terdapat 3 merk/jenis mie yang dijual dengan kode (a,b,c):

a. Indomie, harga per karton untuk mie goreng Rp. 52.000. dan untuk mie kuah Rp.45.000.
b. Sedap Mie, harga per karton untuk mie Rp. 50.000. dan untuk mie kuah Rp.42.500.
c. Sarimi, harga per karton untuk mie Rp. 48.500. dan untuk mie kuah Rp.40.000.

Total harga dihitung dari hargamie*jumlah_beli. Potongan harga diberikan jika membeli lebih dari 20 karton, sebesar 10% dari total harga. Bonus tas cantik diberikan kepada pembeli yang membayar lebih dari 2 juta, dan bonus jas hujan bagi total bayar lebih dari 1 juta. Gunakan constanta untuk harga.

tampilan program yang diminta:

```
==========================================
      *****  HITUNG TOTAL BAYAR  *****
==========================================
  NAMA PEMBELI          : .............
  KODE MIE              : ……...
  MERK MIE              : .............
  JUMLAH MIE GORENG     : ..... KARTON
```

JUMLAH MIE KUAH : ..... KARTON

==========

TOTAL HARGA : Rp. .....

POTONGAN : Rp. .....

==========

TOTAL BAYAR : Rp. .....

BONUS : .....

==========

5.4 Pernyataan Pengulangan

Pernyataan pengulangan merupakan perintah program yang melakukan pengulangan terhadap satu baris/ satu blok baris program beberapa kali sesuai dengan persyaratan yang diberikan.

5.4.1 Pernyataan While

Pernyataan **while** berfungsi untuk mengulang satu atau beberapa pernyataan sebanyak syarat yang diberikan dengan cara syarat diuji terlebih dahulu baru mengerjakan pernyataan. Bentuk pernyataan **while** :

```
while(ungkapan)
    pernyataan;
```

Pada pengulangan dengan bentuk ***while***, *pernyataan* yang mengikuti **while** akan dikerjakan jika *ungkapan* yang diuji bernilai benar (sama dengan 1). Pada pernyataan **while**, *ungkapan* akan diuji terlebih dahulu sebelum *pernyataan* dikerjakan, sehingga ada kemungkinan bagian *pernyataan* pada **while** tidak akan dikerjakan sama sekali jika *ungkapan* yang diuji bernilai salah (sama dengan nol).

Psoudocode Cetak_angka

```
// Akan dicetak angka 1 sampai 10 dengan perulangan while
//deklarasi
   int i

//deskripsi
{
   i = 1
```

```
while(i<=10)
{
   cetak(i)
   i++
}
```

Contoh program 5.7

//program contoh pengulangan dengan while

```
#include <iostream.h>
#include <conio.h>
void main()
{
   clrscr();
   int i;       //sebagai variabel pencacah
   i = 0;       //mula-mula diisi nilai 0
   while (i < 10)
   {
     cout<<"i ke - "<<i<<endl;
          i++; //menaikkan pencacah sebesar 1
   }
      getch();
 }
```

Hasil Eksekusi:

```
i ke - 0
i ke - 1
i ke - 2
i ke - 3
i ke - 4
i ke - 5
i ke - 6
i ke - 7
i ke - 8
i ke - 9
```

Pseudocode Cetak_bil_ganjil

```
/* Akan dicetak bilangan ganjil dengan batas tertentu
menggunakan pengulangan while*/
//deklarasi
   int batas, i

//deskripsi
{
  baca(batas)
  i = 1
  while(i<=batas)
  {
    if(i%2=1)
       cetak(i)
        i++
     }
```

Contoh program 5.8

```
//program contoh pengulangan cetak bilangan ganjil dengan while
#include <iostream.h>
#include <conio.h>
void main()
{
     clrscr();
     int batas;
     int i;    //sebagai variabel pencacah
     i = 0;          //mula-mula diisi nilai 0
     cout<<"Masukkan batas bilangan : "; cin>>batas;
     while (i <= batas)
     {
          if (i%2==1)
          cout<<i<<" ";
          i++; //menaikkan pencacah sebesar 1
```

```
    }
    getch();
}
```

Hasil Eksekusi:

```
Masukkan batas bilangan : 33
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33
```

5.4.2 Pernyataan do...while

Pernyataan **do...while** berfungsi untuk mengulang satu atau beberapa pernyataan sebanyak syarat yang diberikan dengan cara syarat diuji setelah pernyataan dikerjakan. Bentuk pernyataan **do...while** :

```
do
   pernyataan;
while (ungkapan);
```

Pada pengulangan ini *pernyataan* yang mengikuti **do...while** akan dikerjakan terlebih dahulu, setelah itu baru diuji ungkapan dalam **while,** jika *ungkapan* yang diuji bernilai benar (sama dengan 1), maka pernyataan akan dikerjakan kembali, begitu seterusnya sampai ungkapan yang diuji bernilai salah (sama dengan nol).

Pseudocode cetak_bil_genap

```
/* Akan dicetak bilangan genap dengan batas tertentu
menggunakan pengulangan while*/
//deklarasi
   int batas, i

//deskripsi
{
  baca(batas)
  i = 1
  do
  {
   if(i%2=0)
```

```
      cetak(i)
      i++
  }while(i<=batas)
```

Contoh program 5.9

```
/*program contoh pengulangan cetak bilangan genap dengan
do_while */
#include <iostream.h>
#include <conio.h>
void main()
{
  clrscr();
  int batas;
  int i;  //sebagai variabel pencacah
  i = 0;       //mula-mula diisi nilai 0
  cout<<"Masukkan batas bilangan : "; cin>>batas;
  do {
     if (i%2==0)
     cout<<i<<" ";
     i++; //menaikkan pencacah sebesar 1
  }while(i<=batas);
  getch();
}
```

Hasil Eksekusi:

```
Masukkan batas bilangan : 30
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

5.4.3 Pernyataan for

Pernyataan for juga berguna untuk mengulang eksekusi terhadap satu atau sejumlah pernyatan (baris program). Bentuknya adalah sebagai berikut :

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

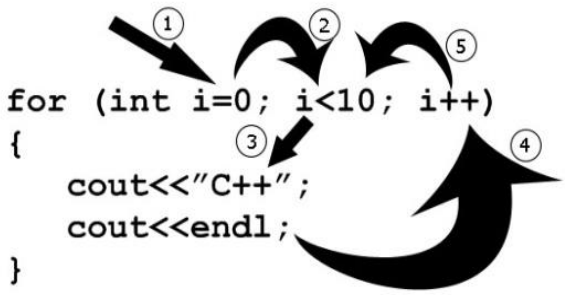
Pernyataan ini identik dengan pernyataan :

```
ungkapan1;
while (ungkapan2)
{
  pernyataan;
  ungkapan3;
}
```

Dimana

- *ungkapan1* merupakan pernyataan inisialisasi sebelum masuk ke while.
- *ungkapan2* berlaku sebagai kondisi yang menentukan pengulangan terhadap pernyataan atau tidak.
- *ungkapan3* digunakan sebagai pengatur variabel yang digunakan di dalam *ungkapan1*.

Bentuk pernyataan for dapat diilustrasikan :



Pseudocode cetak_angka_for

```
// Akan dicetak angka 1 sampai 10 dengan perulangan for
//deklarasi
   int i;
//deskripsi
{
   for(i=0; i<=10; i++)
   {
      cetak(i)
   }
}
```

Contoh program 5.10

```
//program contoh penggunaan pernyataan for - cetak bilangan
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr();
 int i;
 for(i=1; i<=10; i++)
 {
     cout<<i<<" ";
 }
 getch();
}
```

Hasil eksekusi:

```
     1 2 3 4 5 6 7 8 9 10
```

Contoh program 5.11

```
/*program contoh penggunaan pernyataan for - cetak bilangan
genap dengan batas tertentu*/
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr();
 int i, batas;
 cout<<"Masukkan bilangan batas : "; cin>>batas;
 for(i=1; i<=batas; i++)
 {
     if(i%2==0)
     cout<<i<<" ";
 }
 getch();
}
```

Hasil eksekusi:

```
Masukkan batas bilangan : 30
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

5.4.4 Pernyataan for bersarang

Pernyataan for bersarang (**nested for)** merupakan pernyataan for yang berada dalam pernyataan for. Pernyatan ini juga berguna untuk mengulang eksekusi terhadap satu atau sejumlah pernyatan (baris program). Bentuknya adalah sebagai berikut :

```
for1 (ungkapan1; ungkapan2; ungkapan3;)
{
  for2 ((ungkapan1; ungkapan2; ungkapan3;)
  {
      Pernyataan
  }
}
```

Dalam bentuk ini, *pernyataan* akan dikerjakan sebanyak syarat pada *ungkapan2* pada *for2.* Selanjutnya seluruh pernyataan yang ada di dalam *for2* akan diulang sebanyak syarat pada *ungkapan2* pada *for1*.

Pseudocode baris_bintang

```
//akan dicetak barisan karakter * dengan for bersarang
//deklarasi
int tinggi, lebar;
int i,j;

//deskripsi
cetak("Masukkan tinggi : ")
baca(tinggi)
cetak("Masukkan lebar : ")
```

```
baca(lebar)
for(i=1; i<=tinggi; i++)
 {
   for(j=1; j<=lebar; j++)
   {
    cetak("*")
   }
  }
```

Contoh Program 5.12

```
//program contoh penggunaan nested for
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr();
 int tinggi, lebar;
 int i,j;
 cout<<"Masukkan tinggi : "; cin>>tinggi;
 cout<<"Masukkan lebar : "; cin>>lebar;
 cout<<endl;
 cout<<"Barisan Bintang: "<<endl;
for(i=1; i<=tinggi; i++) //pengulangan untuk cetak tinggi
{
  for(j=1; j<=lebar; j++) //pengulangan untuk cetak lebar
```

```
    {
      cout<<"*";
    }
  }
  getch();
  }
```

Hasil eksekusi:

```
  Masukkan tinggi baris : 4
  Masukkan lebar        : 5

  Barisan Bintang
  *****
  *****
  *****
  *****
```

Pseudocode Segitiga_Bintang

```
// Akan dicetak karakter * yang membentuk bangun segitiga
 //DEKLARASI
   int tinggi, i, j
 //DESKRIPSI
{
   baca(tinggi)
   for(i=1;i<=tinggi;i++)
      for(j=1;j<=i;j++)
          cetak('*')
      endl
 }
```

Contoh Program 5.13

```
  // contoh penggunaan nested for untuk segitiga bintang
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int tinggi; //untuk tinggi segitiga
 clrscr();
 cout<<"Masukkan Tinggi Segitiga = "; cin>>tinggi;
 cout<<endl;
 for(int baris=1; baris<=tinggi; baris++)
 {
   for(int kolom=1; kolom<=baris; kolom++)
   {
    cout<<"*";
   }
    cout<<endl;
  }
  getch();
}
```

Hasil eksekusi:

```
Masukkan Tinggi Segitiga = 8

*
**
***
****
*****
******
*******
********
```

Contoh Program 5.14

```
/*program contoh penggunaan nested for untuk segitiga sama
kaki karakter */
```

```
#include <iostream.h>
int main()
{
    int tinggi;
    cout << "Tinggi segitiga = "; cin >>tinggi;
    for (int kolom=1; kolom<=tinggi; kolom++)
    {
          int baris =kolom-1;
          for (int turun=1; turun<=baris; turun++)
          {
                cout << " ";
          }
          baris = tinggi - kolom + 1;
          for ( turun = 1; turun<=baris; turun++)
          {
                cout << " *";
          }
          cout << endl;
    }
}
```

Hasil eksekusi:

```
Tinggi segitiga = 8
* * * * * * * *
 * * * * * * *
  * * * * * *
   * * * * *
    * * * *
     * * *
      * *
       *
```

Latihan Praktikum – 7

1. Buatlah program untuk menampilkan deret bilangan yang habis dibagi 3 dan 5 sampai batas bilangan tertentu!
2. Sebuah LPK ternama di Kediri, mengadakan test Program komputer yang akan diadakan tiga kali test (terdiri dari program Ms. Word, Ms. Excel, Bahasa C). Test ini didakan untuk menentukan Grade dan besarnya biaya kursus bila siswa tersebut ingin melanjutkan ke tingkat yang lebih tinggi. Sebagai inputan adalah Jumlah Siswa dan Nilai dari ketiga test tersebut masing-masing siswa. Rata-rata Nilai adalah nilai akhir dari siswa. Tampilkan data dan Grade serta Jumlah Bayar

Keterangan : Gunakan fungsi perulangan untuk input data siswa, f**ungsi if d**an fungsi **switch** untuk menentukan **grade.**

Ketentuan Nilai :

Nilai	Grade	Besar Biaya
91 - 100	**A**	**50.000**
76 - 90	**B**	**150.000**
60 - 75	**C**	**250.000**
40 - 59	**D**	**350.000**
01 - 39	**E**	**450.000**
0	**0**	**550.000**

Tugas (4)

1. Sebagai seorang analis sistem anda diminta membuat flowchart algoritma dan program untuk menghitung gaji karyawan. Data inputan nama pegawai, kategori bagian, jumlah hari kerja, dan jumlah jam lembur. Jika terdapat 3 kategori karyawan:
   a. Karyawan bagian Produksi dengan Gaji Rp.160.000, upah lembur per jam sebesar Rp.20.000.
   b. Karyawan bagian Packing dengan Gaji Rp.145.000, upah lembur per jam sebesar Rp.15.000.
   c. Karyawan bagian Distribusi dengan Gaji Rp.155.000, upah lembur per jam sebesar Rp.17.500.

   dengan ketentuan:

- Tunjangan istri diberikan sebesar 15% dari gaji pokok, Tunjangan anak akan diberikan sampai anak ke 3, sebesar 10% dari total gajiPOKOK.
- Uang transport akan diberikan kepada pegawai yang lembur lebih dari 10 jam dalam sebulan sebesar 10% dari total upah lembur.
- Gaji pegawai dihitung dari gaji per hari*jumlah hari kerja. Total Upah lembur didapat dari upah lembur*jumlah jam lembur. Gunakan constanta untuk tunjangan istri dan anak.

Tampilkan hasil dalam bentuk tabel yang rapi. Manfaatkan fungsi manipulator.

DAFTAR PUSTAKA

Abdul Kadir, *Pemrograman C++ Membahas Pemrograman Berorientasi Objek Menggunakan Turbo C++ dan Borland C++*, Andi Offset, Yogyakarta

Budi Sutedjo, Michel An, *Algoritma dan teknik pemrograman,* Andi offset, yogyakarta

Eko Nugroho, *Pemrograman Terstruktur Dengan Pascal,* Andi Offset, Yogyakarta

I Made Joni, Budi Raharjo, *Cara Mudah mempelajari Algoritma dan Implementasinya,* Informatika, Bandung.

Jogiyanto, *Analisis dan Desain Sistem informasi Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis,* Andi Offset, Yogyakarta

Rinaldi Munir, *Algoritma dan Pemrograman dalam Bahasa Pascal dan C buku 1 dan 2,* Informatikan, Bandung

Solichin, Achmad (2003). *Pemrograman Bahasa C dengan Turbo C*. IlmuKomputer.Com.

Wahono, Romi Satria(2003). *Cepat Mahir Algoritma dalam Bahasa C*. IlmuKomputer.Com.