Nama MK : Algoritma Dan Pemrograman
Kode MK :
SKS : 3
Prodi : Teknik Informatika
No HP :
Email : thoelet@yahoo.com

SILABUS

- Bahasa pemrograman
- Algoritma dan pemrograman terstruktur
- Penyajian algoritma
- Struktur dasar algoritma (runtunan, pemilihan, pengulangan)
- Teknik penulisan program yang baik
- Struktur program
- Mempelajari Bahasa pemrograman

Referensi :

Budi sutedjo, Michel An, *Algoritma dan teknik pemrorgaman,* Andi offset, yogyakarta
Rinaldi Munir, *Algoritma dan pemrograman dalam bahasa pascal dan C buku 1 dan 2,* Informatikan Bandung
Antony Pranata, *Algortima dan pemrograman* J&J Learning, Yogyakarta
Eko Nugroho, *Pemrograman Terstruktur Dengan Pascal,* Andi Offset, Yogyakarta
Jogiyanto, *Analisis dan Desain Sistem informasi Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis,* Andi Offset, Yogyakarta
Abdul Kadir, *Pemrograman C++ membahas Pemrograman Berorientasi objek Menggunakan Turbo C++ dan Borland C++*, Andi Offset, Yogyakarta

Penilaian :

UTS + UAS + Tugas + Presensi
30% 40% 25% 5%

Range Nilai :
A : 81 – 100
B : 71 – 80
C : 61 – 70
D : 51 – 60
E : <=50

Komplain nilai akhir hanya dilayani maksimal dua minggu setelah nilai diumumkan.

Ketentuan lain :

- Kehadiran kuliah minimal 75% jika kurang dari itu maka tidak boleh mengikuti Ujian Akhir Semester (UAS) atau nilai maksimal D. Jika sebelum masa UAS kehadiran < 75% maka mahasiswa bisa menghubungi dosen untuk melengkapi kehidirannya yang kurang (bisa dengan tugas, presentasi, dll)
- Jika tidak bisa kuliah karena berhalang memberi kabar kepada dosen dengan SMS dan memberitahu kelompoknya.
- Dibentuk Kelompok Diskusi dengan anggota maksimal 5 orang

Fungsi Kelompok Diskusi :

- Sebagai media belajar bersama-sama
- Jika ada anggota yang tidak hadir kuliah maka kelompok harus tahu →jika tidak tahu ada sangsi

- Tugas-tugas bisa diberikan hanya kepada ketua kelompok atau anggota, bisa langsung maupun melalui email
- Setiap tugas yang diberikan harus dipahami oleh setiap anggota kelompok, jika ada anggota kelompok yang tidak paham maka anggota yang lain harus menjelaskan kepada anggota yang tidak paham
- Tugas yang diberikan bisa saja dipresentasikan oleh salah satu anggota kelompok

Teknik Pembelajaran :

- Pemberian materi oleh dosen
- Pemberian tugas kelompok atau individu (di kelas maupun di rumah)
- Presentasi kelompok atau individu
- Tanya jawab / diskusi

BAB 1
BAHASA PEMROGRAMAN

I. PENDAHULUAN

Hal terpenting dalam menjalankan komputer 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, bila manusia ingin berkomunikasi dengan komputer tidak harus menerjemahkan ke dalam 0 dan 1. Bila hal itu dilakukan betapa rumitnya suatu program.

II. Istilah-Istilah Dasar

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, terdapat dua faktor penting yaitu sintaksis dan semantik. Sintak adalah aturan-aturan gramatikal yang mengatur tata cara penulisan kata, ekspresi dan pernyataan sedangkan 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.

III. Bahasa Pemrograman

Secara umum bahasa permrograman dibagi menjadi empat kelompok :

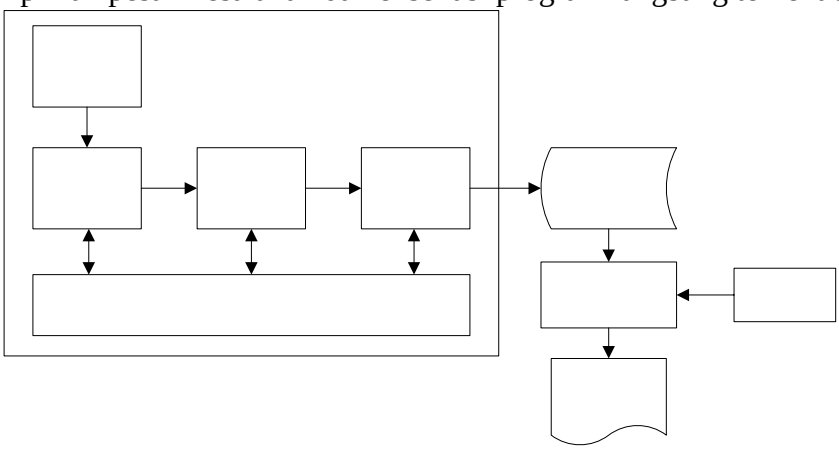
a. Bahasa Aras Rendah (*Low Level Language*)
   Merupakan bahasa yang berorientasi pada mesin. Pemrogram dengan bahasa ini harus berpikir berdasarkan logika mesin berpikir, sehingga bahasa ini kurang fleksibel dan sulit dipahami.
   Contoh : Bahasa mesin, Bahasa rakitan (assembly)
b. Bahasa Aras Menengah *(Middle Level Language)*
   Merupakan bahasa pemrograman yang menggunakan aturan-aturan gramatikal dalam penulisan *ekspresi* atau pernyataan dengan standar yang mudah dipahami manusia serta memiliki instruksi-instruksi tertentu yang langsung bisa diakses oleh komputer.
   Contoh : Bahasa C
c. Bahasa Aras Tinggi *(Hight Level Language)*
   Merupakan bahasa pemrograman yang menggunakan aturan-aturan gramatikal dalam penulisan *ekspresi* atau pernyataan dengan standar bahasa yang langsung dapat dipahami oleh manusia.
   Contoh : Bahasa Pascal, Basic, COBOL
d. Bahasa Berorientasi Objek (*Object Oriented Programming*)
   Dengan bahasa berorientasi objek kita tidak perlu menuliskan secara detail semua pernyataan dan *ekspresi* seperti bahasa aras tinggi, melainkan cukup dengan memasukkan kriteria-kriteria yang dikehendaki saja.

Contoh : Delphi, Visual Basic, C++

Agar komputer memahami program yang disusun dengan bahasa pemrograman, maka dibutuhkan suatu penerjemah yaitu *Interpreter* dan *Compiller*.

A. Interpreter

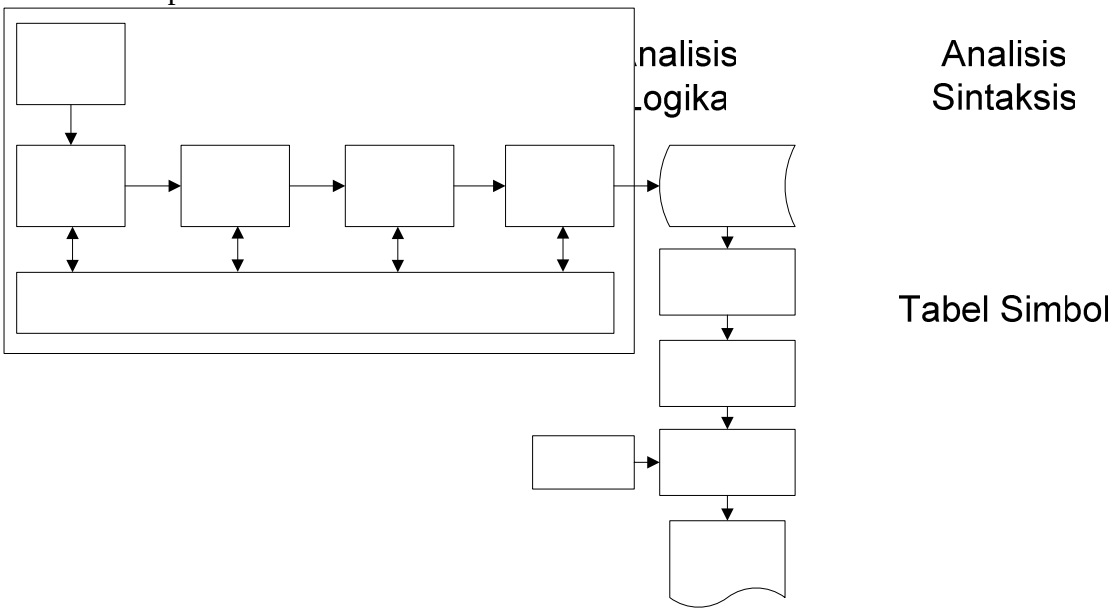
Interpreter berasal dari kata *to interpret* yang berarti menerjemahkan atau mengartikan. *Interpreter* merupakan penerjemah bahasa pemrograman yang menerjemahkan instruksi demi instruksi pada saat eksekusi program. Pada saat penerjemahan interpreter akan memeriksa sintaksis (sintak program), semantik (arti perintah), dan kebenaran logika. Jika ditemukan kesalahan sintaksis (*syntak error*) maka interpreter akan menampilkan pesan kesalahan dan eksekusi program langsung terhenti.



Gambar 1. Proses Kerja Interpreter

B. Compiler

Berasal dari kata *to compile* yang berarti menyusun, mengumpulkan atau menghimpun. *Compiler* merupakan penerjemah bahasa pemrograman yang menerjemahkan instruksi-instruksi dalam satu kesatuan modul ke dalam bahasa mesin (*objek program*), kemudian objek program akan mengalami *linking* yang berfungsi untuk menggabungkan modul-modul tersebut dengan modul-modul lain yang berkaitan seperti data tentang karakteristik mesin, file-file pustaka atau objek program lainnya yang berkaitan dengan objek lainnya menghasilkan file *Executable* program yang akan dieksekusi oleh komputer.



Gambar 2. Proses kerja Compiler

C. Perbedaan Interpreter dan Compiler

	Interpreter		Compiler
1	Menerjemahkan instruksi per instruksi	1	Menerjemahkan secara keseluruhan sekaligus
2	Bila tejadi kesalahan kompilasi, dapat langsung dibetulkan secara interaktif	2	Bila tejadi kesalahan kompilasi, *Source* program harus dibenarkan dan proses kompilasi diulang kembali
3	Tidak menghasilkan objek program	3	Menghasilkan objek program
4	Tidak menghasilkan *executable* program karena langsung dijalankan pada saat program diinterprestasi	4	Menghasilkan *executable* program, sehingga dapat dijalankan di keadaan prompt sistem
5	Proses interprestasi terasa cepat, karena tiap-tiap instruksi langsung dikerjakan dan output langsung dilihat hasilnya	5	Proses kompilasi lama karena sekaligus menterjemahkan seluruh instruksi program
6	Source program terus dipergunakan karena tidak dihasilkan *executable* program	6	Source program sudah tidak dipergunakan lagi untuk mengerjakan program
7	Proses pengerjaan program lebih lambat karena setiap instruksi dikerjakan harus diinterprestasikan ulang kembali	7	Proses mengerjakan program lebih cepat, karena *executable* program sudah dalam bahasa mesin
8	Keamanan dari program kurang terjamin, karena yang selalu digunakan adalah *source* program	8	Keamanan dari program lebih terjamin, karena yang dipergunakan *executable* program

BAB 2
ALGORITMA DAN PEMROGRAMAN TERSTRUKTUR

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.

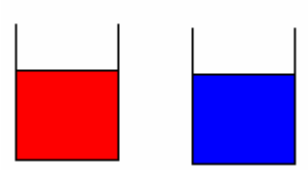
   Secara sederhanya algoritma dapat didefinisikan urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis.

   Kata logis berarti bahwa nilai kebenarannya harus dapat ditentukan, benar atau salah. Langkah-langkah yang tidak benar dapat memberikan hasil yang salah.

   Contoh :

   Misalkan terdapat dua buah gelas, gelas A dan gelas B. Gelas A berisi air berwarna merah dan gelas B berisi air berwarna biru, kita ingin menukartan isi air kedua gelas tersebut, sehingga gelas A berisi air berwarna biru dan gelas B berisi air berwarna merah.

```
Algoritma Tukar_Isi_Gelas
1. Tuangkan air dari gelas A ke gelas B
2. Tuangkan air dari gelas B ke gelas A
```



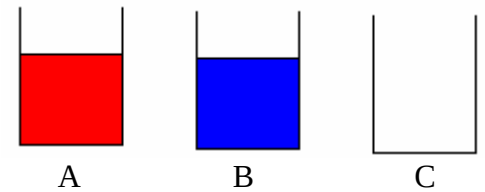
A B

   Algoritma diatas tidak menghasilkan pertukaran yang benar, langkah-langkahnya tidak logis, karena yang terjadi bukan pertukaran tetapi percampuran antara air di gelas A dengan air di gelas B. Sehingga algoritma Tukar_Isi_Gelas diatas salah.

   Dari permasalahan diatas algoritma yang benar adalah bahwa untuk menukarkan isi air pada gelas A dengan isi air pada gelas B maka dibutuhkan sebuah gelas bantuan yang dipakai untuk menampung salah satu air dalam gelas tersebut misalkan gelas C. Sehingga algoritma yang benar dari permasalah diatas adalah :

```
Algoritma Tukar_Isi_Gelas
1. Tuangkan air dari gelas A ke gelas C
2. Tuangkan air dari gelas B ke gelas A
3. Tuangkan air dari gelas C ke gelas B
```

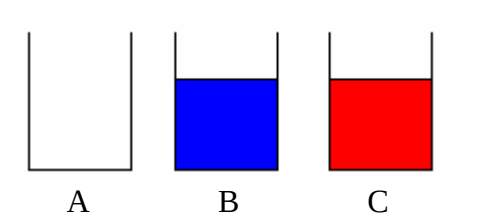
   Keadaan awal sebelum pertukaran



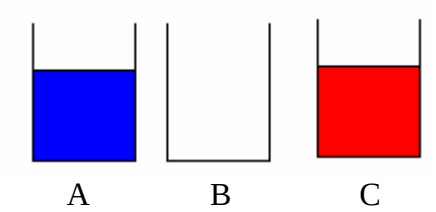
A B C

Proses pertukaran :

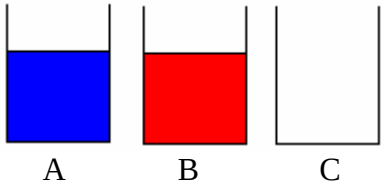
1. Tuangkan air dari gelas A ke gelas C



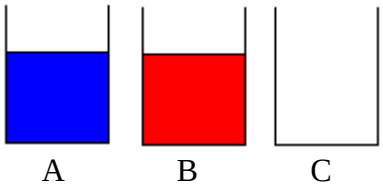
2. Tuangkan air dari gelas B ke gelas A



3. Tuangkan air dari gelas C ke gelas B

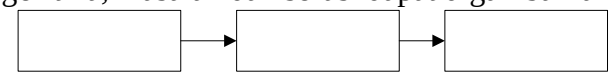


Keadaan setelah pertukaran



Sekarang algoritma Tukar_Isi_Gelas diatas sudah diperbaiki, sehingga isi air pada gelas A dan isi air pada gelas B dapat dipertukarkan dengan benar.

Hubungan antara algoritma, masalah dan solusi dapat digambarkan sebagai berikut :



Tahap pemecahan masalah adalah Proses dari masalah hingga terbentuk suatu algoritma. **Tahap implementasi** adalah proses penerapan algoritma hingga menghasilkan solusi. **Solusi** yang dimaksud adalah suatu program yang merupakan implementasi dari algoritma yang disusun.

Ciri algoritma yang baik adalah :

a. Algoritma memiliki logika perhitungan atau metode yang tepat dalam menyelesaikan masalah.
b. Menghasilkan output yang tepat dan benar dalam waktu yang singkat.
c. Algortima ditulis dengan bahasa yang standar secara sistematis dan rapi sehingga tidak menimbulkan arti ganda (*ambiguous*).
d. Algortima ditulis dengan format yang mudah dipahami dan mudah diimplementasikan ke dalam bahasa pemrograman.
e. Semua operasi yang dibutuhkan terdefinisi dengan jelas.
f. Semua proses dalam algoritma harus berakhir setelah sejumlah langkah dilakukan.

2. 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 :

a. Mengandung teknik pemecahan masalah yang tepat dan benar.
b. Memiliki algoritma pemecahan masalah yang bersifat sederhana, standar dan efektif dalam menyelesaikan masalah.
c. Teknik penulisan program memiliki struktur logika yang benar dan mudah dipahami.
d. Program semata-mata terdiri dari tiga struktur yaitu *sequence structure, looping structure* dan *selection structure.*
e. Menghindarkan penggunaan instruksi GOTO (peralihan proses tanpa syarat tertentu) yang menjadikan program tidak terstruktur lagi.
f. Membutuhkan biaya testing yang rendah.
g. Memiliki dokumentasi yang baik.
h. Membutuhkan biaya perawatan dan pengembangan yang rendah.

BAB 3
PENYAJIAN ALGORITMA

Algoritma dapat disajikan dengan dua teknik yaitu teknik tulisan dan teknik gambar. Teknik tulisan biasanya menggunakan metode *structure english* dan *pseudocode*, sedangkan teknik gambar biasanya menggunakan diagram alir (*flow chart).*

A. *Structure English* dan *Pseudocode*

Structure English merupakan alat yang cukup efisien untuk menggambarkan suatu algoritma. Basis dari structure english adalah bahasa inggris, tetapi juga bisa digunakan bahasa indonesia, sedangkan *pseudocode* berarti kode yang mirip dengan kode pemrograman sebenarnya. *Pseudocode* berasal dari kata *pseudo* yang berarti imitasi/mirip/menyerupai dan *code* yang berarti program. *Pseudocode* berbasis pada kode program yang sesungguhnya seperti Pascal, C, C++. *Pseudocode* lebih rinci dari structure english misalnya dalam menyatakan tipe data yang digunakan.

Contoh struktur Indonesia

```
Baca data jam_kerja
Hitung gaji adalah jam_kerja dikalikan tarif
Tampilkan gaji
```

Pseudocode dengan Pascal :

```
Read jam_kerja
Gaji := jam_kerja * tarif
Write gaji
```

Aturan Penulisan Teks Algoritma

Langkah-langkah penyelesaian masalah dalam teks algoritma dapat ditulis dalam notasi apapun, dengan syarat bahwa langkah-langkah tersebut mudah dipahami dan dimengerti. Tidak ada notasi yang baku dalam teks algoritma sebagaimana notasi dalam bahasa pemrograman (notasi dalam algoritma disebut dengan notasi algoritmik). Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Berkaitan hal itu untuk memudahkan translasi notasi algoritmik ke dalam bahasa pemrograman, sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum. Sebagai contoh :

```
Tulis nilai X dan Y
```

Dalam notasi algoritmik menjadi :

```
Write(X,Y)
```

Notasi `write` ini berarti nilai `X` dan `Y` dicetak ke piranti keluaran. Notasi `write` ini berkorespondensi dengan `write` atau `writeln` dalam bahasa pascal, `printf` dalam bahasa C, `cout` dalam bahasa C++. Jadi, translasi `write(X,Y)` dalam masing-masing bahasa tersebut adalah :

```
writeln(X,Y);              { dalam bahasa pascal }
printf("%d %d", X,Y);      /* dalam bahasa C */
cout<<X<<Y;                /* dalam bahasa C++ */
```

Perhatikan bahwa setiap bahasa pemrograman mempunyai aturan sendiri dalam menggunakan perintah penulisan.

Contoh lain :

```
Isikan nilai X ke dalam max
```

Ditulis dalam notasi algoritmik menjadi :

```
max ← X
```

Notasi "←" berarti mengisi (*assign*) peubah (*variable*) `max` dengan nilai peubah `X`. Translasi notasi "←" kedalam bahasa Pascal adalah ":=", dalam bahasa C adalah "=", dalam bahasa C++ adalah "=". Translasi `max←X` dalam masing-masing bahasa adalah :

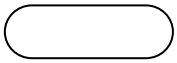
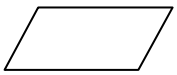
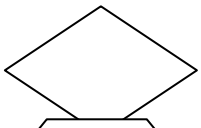
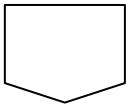
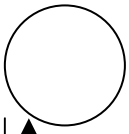
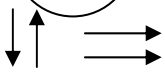
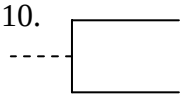
```
max := X;       { dalam bahasa Pascal }
max = X;        /* dalam bahasa C */
max = X;        /* dalam bahasa C++ */
```

B. *Flowchart*

Dalam structure English / struktur Indonesia digambarkan tahap-tahap penyelesaian masalah dengan menggunakan kata-kata (teks). Kelemahan cara ini adalah dalam penyusunan algoritma sangat dipengaruhi oleh tata bahasa pembuatnya, sehingga kadang-kadang orang lain sulit memahaminya. Oleh sebab itu kemudian dikembangkan metode yang menggambarkan tahap-tahap pemecahan masalah dengan merepresentasikan simbol-simbol tertentu yang mudah dimengerti, mudah digunakan dan standar. Salah satu penulisan simbol tersebut adalah dengan menggunakan *flowchart*. *Flowchart* terdiri dari dua macam yaitu :

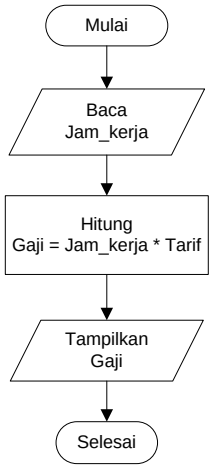
1. *Flowchart Program*

   Bagan alir program adalah suatu bagan yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir. Bagan alir program merupakan alat yang berguna bagi programmer untuk mempersiapkan program yang rumit. Bagan alir terdiri dari simbol-simbol yang mewakili fungsi-fungsi langkah program dan garis alir (*flow lines*) menunjukan urutan dari simbol yang akan dikerjakan.

1.  Simbol Terminal, simbol yang digunakan untuk menyatakan awal atau akhir suatu program.
2.  Simbol Input/Output, simbol yang digunakan untuk menunjukkan operasi masukan atau keluaran
3.  Simbol Proses, simbol yang digunakan untuk menggambarkan proses pengolahan data
4.  Simbol Keputusan, simbol yang digunakan untuk menyatakan suatu pilihan berdasarkan suatu kondisi tertentu
5.  Simbol persiapan (Preparation), simbol yang digunakan untuk memberikan nilai awal pada suatu variabel atau pencacah
6.  Simbol proses terdefinisi (predefined process symbol), simbol yang digunakan untuk proses yang detilnya dijelaskan terpisah, misal dalam bentuk subroutine
7.  Simbol Penghubung ke halaman lain, simbol yang digunakan untuk menghubungkan bagian diagram alir pada halaman yang berbeda
8.  Simbol Penghubung ke halaman yang sama, simbol yang digunakan untuk menghubungkan bagian diagram alir pada halaman yang sama
9.  Simbol Arah aliran, simbol yang digunakan untuk menunjukkan arah aliran proses
10.  Annotation simbol, simbol yang digunakan untuk memberikan keterangan-keterangan untuk memperjelas simbol-simbol lain

Gambar 3. Simbol-simbol flowchart program

Contoh penggunaan flowchart program :



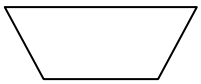
Pedoman membuat flowchart :

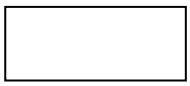
1. Flowchart dibuat dari atas ke bawah dimulai dari bagian kiri suatu halaman.
2. Kegiatan dalam flowchart harus ditunjukkan dengan jelas.
3. Kegiatan dalam flowchart harus jelas dimana akan dimulai dan dimana akan berakhir.
4. Kegiatan yang ada dalam flowchart digunakan kata yang mewakili pekerjaan.
5. Kegiatan dalam flowchart harus sesuai dengan urutannya.
6. Kegiatan yang terpotong dihubungkan dengan simbol penguhubung.
7. Simbol-simbol yang digunakan flowchart adalah simbol-simbol standar.

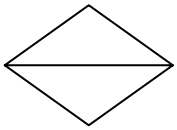
2. Flowchart system

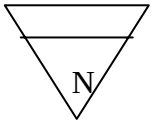
Bagan alir sistem berbeda dengan bagan alir program. Bagan alir program sifatnya lebih terperinci tentang langkah-langkah proses di dalam program dari awal sampai akhir. Bagan alir sistem hanya menggambarkan arus data dari sistem. Simbol-simbol yang digunakan pada bagan alir sistem ada yang sama dan ada yang berbeda dengan simbol-simbol yang digunakan pada bagan alir program.

1.  Simbol Dokumen
   Simbol yang menunjukkan dokumen yang digunakan untuk input dan output baik secara manual, mekanik maupun komputerisasi.

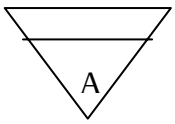
2.  Simbol operasi Manual
   Simbol yang menunjukkan pekerjaan yang dilakukan secara manual.

3.  Simbol Proses
   Simbol yang menunjukkan kegiatan proses operasi program komputer.

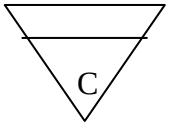
4.  Simbol pengurutan
   Simbol yang menunjukkan proses pengurutan dokumen di luar komputer.

5. 

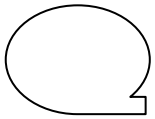
   Simbol Offine Storage
   Simbol yang menunjukkan file non komputer yang diarsip urut angka (numeric).

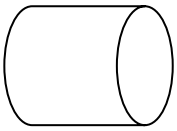
6. 

   Simbol Offine Storage
   Simbol yang menunjukkan file non komputer yang diarsip urut huruf (Alphabetic).

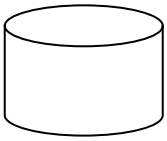
7. 

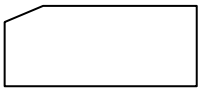
Simbol Offine Storage
Simbol yang menunjukkan file non komputer yang diarsip urut tanggal (Chronological).

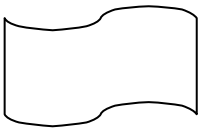
8. 
Simbol Magnetic tape
Simbol yang menunjukkan Input Output yang menggunakan pita magnetic.

9. 
Simbol Magnetic Drum
Simbol yang menunjukkan Input Output yang menggunakan Drum magnetic.

10. 
Simbol Magnetic Storage
Simbol yang menunjukkan Input Output yang menggunakan Diskette.

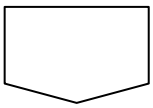
11. 
Simbol Hard Disk Storage
Simbol yang menunjukkan Input Output yang menggunakan Hard Disk.

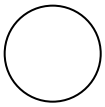
12. 
Simbol Punched Card
Simbol yang menunjukkan Input Output yang menggunakan Kartu Plong.

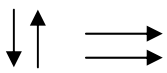
13. 
Simbol Punched tape
Simbol yang menunjukkan Input Output yang menggunakan kertas berlubang.

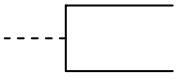
14. 
Simbol Keyboard
Simbol yang menunjukkan Input Output yang menggunakan on line keyboard

15. 
Simbol Display
Simbol yang menunjukkan Output yang ditampilkan dilayar terminal

16. 
Simbol Penghubung ke halaman lain, simbol yang digunakan untuk menghubungkan bagian diagram alir pada halaman yang berbeda

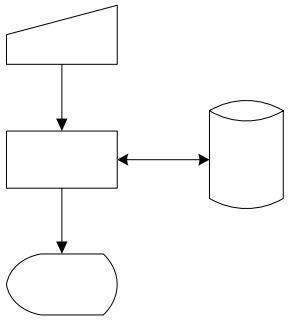
17. 
Simbol Penghubung ke halaman yang sama, simbol yang digunakan untuk menghubungkan bagian diagram alir pada halaman yang sama

18. 
Simbol Arah aliran, simbol yang digunakan untuk menunjukkan arah aliran proses

19. 
Annotation simbol, simbol yang digunakan untuk memberikan keterangan-keterangan untuk memperjelas simbol-simbol lain

Gambar 4. Simbol-simbol bagan alir

Contoh penggunaan flowchart sistem :



Gambar Contoh penerapan sistem flowchart

Keyboar

CPU

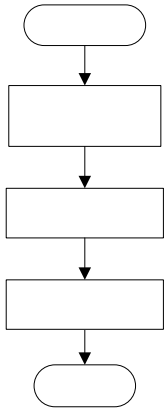
Monitor

BAB 4
STRUKTUR DASAR ALGORITMA

Dalam sebuah algoritma langkah-langkah penyelesaian masalahnya dapat berupa struktur urut (*sequence*), struktur pemilihan (*selection)*, dan struktur pengulangan (*repetition*). Ketiga jenis langkah tersebut membentuk konstrukti suatu algoritma.

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.



Gambar Flowchart struktur urut

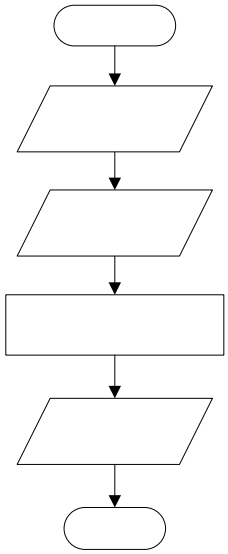
Dari flowchart diatas mula-mula pemroses akan melaksanakan instruksi baris program 1, instruksi baris program 2 akan dikerjakan jika instruksi baris program 1 telah selesai dikerjakan. Selanjutnya instruksi baris program 3 dikerjakan setelah instruksi baris program 2 selesai dikerjakan. Setelah instruksi baris program 3 selesai dilaksanakan maka algoritma berhenti.

Contoh 1 :

Akan dihitung luas pesegi panjang yang diketahui panjang dan lebarnya, maka algoritmanya sebagai berikut :

```
Algoritma Luas_Pesegi_Panjang
Diketahui sebuah pesegi panjang yang memiliki panjang dan
lebar.
Deskripsi :
1. mulai
2. Baca panjang
3. Baca lebar
4. Hitung luas = panjang * lebar
5. Tampilkan luas
6. selesai
```

Flowchart Luas_Pesegi_Panjang :



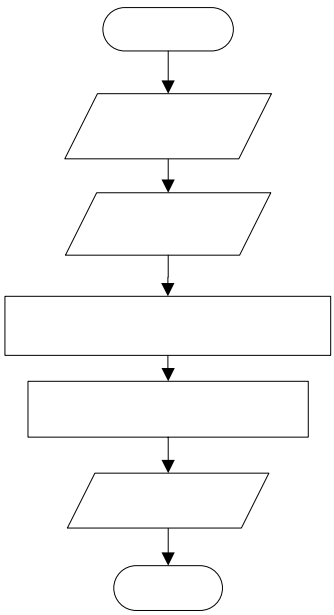
Gambar Flwochart menghitung luas pesegi panjang

Contoh 2 :

Akan dihitung isi sebuah tabung yang diketahui jari-jari lingkaran dan tinggi tabung.

```
Algoritma Isi_Tabung1
Diketahui sebuah tabung yang diketahui jari-jari tabung dan
tinggi tabung.
Deskripsi :
1. mulai
2. Baca jari_jari
3. Baca tinggi
4. Hitung luas_lingk = 3.14 * jari_jari * jari_jari
5. Hitung isi_tabung = luas_lingk * tinggi
6. Tampilkan isi_tabung
7. selesai
```

Flowchart Isi_Tabung1 :



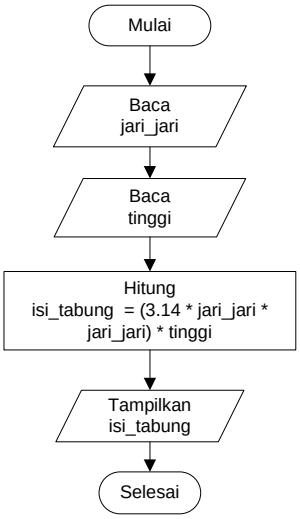
Gambar flowchart menghitung isi tabung

Perhatikan bahwa algoritma Isi_Tabung1 diatas memiliki 5 baris intruksi yang harus dikerjakan sebelum algoritma selesai. Pada algoritma diatas bisa disederhanakan lagi sehingga baris prosesnya lebih sedikit.

```
Algoritma Isi_Tabung2
```

```
Diketahui sebuah tabung yang diketahui jari-jari tabung dan
tinggi tabung.
Deskripsi :
1. mulai
2. Baca jari_jari
3. Baca tinggi
4. Hitung isi_tabung = (3.14 * jari_jari * jari_jari) * tinggi
5. Tampilkan isi_tabung
6. selesai
```

Flowchart Isi_Tabung2 :



Gambar flowchart menghitung isi tabung

Dari kedua algoritma dan flowchart diatas terlihat bahwa algoritma yang kedua lebih sedikit baris intruksinya, sehingga menyebabkan pemrosesan menjadi lebih cepat selesai dengan hasil yang sama dengan algoritma pertama. Pada algorima yang kedua jika diimplementasikan dalam program kebutuhan variabelnya juga lebih sedikit sehingga menghemat penggunaan memori.

2. Struktur Pemilihan (*selection*) atau Penyeleksian Kondisi

Pada struktur pemilihan tidak setiap baris program akan dikerjakan. Baris prorgam 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 ini dilakukan untuk memilih salah satu dari beberapa alternatif yang tersedia.

Pada pemrograman penyeleksian dilakukan pada suatu pernyataan *boole*, yang dapat menghasilkan nilai benar (*true*) atau nilai salah (*false*). Biasanya sebuah pernyataan pemilihan terdiri dari *operand-operand* yang dihubungkan dengan operator relasi dan digabungkan dengan operator logika.

Contohnya :

1. 7 = 7 (Benilai benar, sebab 7 sama dengan 7)
2. 5 = 9 (Bernilai salah, sebab 5 tidak sama dengan 9)
3. 4 > 2 (Bernilai benar, sebab 4 lebih besar dari pada 2)
4. 3 <> 8 (Bernilai benar, sebab 3 tidak sama dengan 8)
5. X = 10 (Dapat benilai benar atau salah, tergantung isi variabel X)
6. (X > 3) And (Y < 12)
   (Dapat benilai benar atau salah, tergantung isi variabel X dan Y)

Struktur pemilihan dalam penulisan program diimplementasikan dengan instruksi **IF**.

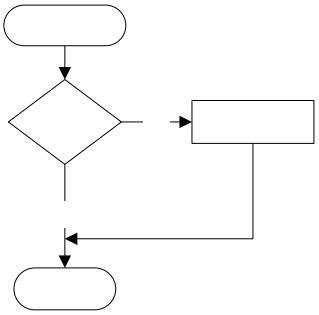
Macam-macam struktuf IF :

1. IF sederhana

Bentuk IF sederhana adalah :

```
IF <syarat> THEN
    <instruksi>
```

Bentuk flowchart :



Gambar Flowchart IF sederhana

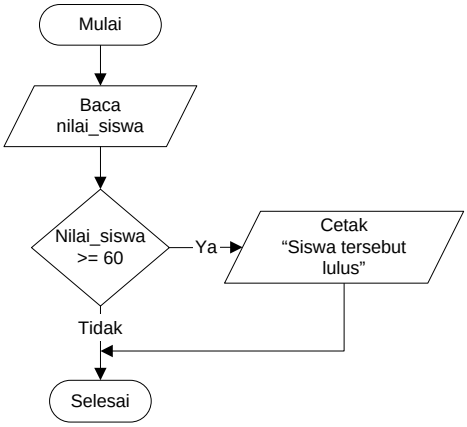
Pada bentuk IF sederhana ini, `intruksi` akan dikerjakan jika `syarat` yang diuji benilai benar (*true*). Jika `syarat` yang diuji benilai salah (*false*) maka tidak ada instruksi yang dikerjakan.

Contoh 1 :

Dibuat aturan untuk menentukan kelulusan seorang siswa yang diketahui dari hasil nilainya. Seorang siswa dikatakan lulus jika nilai lebih besar atau sama dengan 60.

```
Algoritma Kelulusan_Siswa
Diketahui seorang siswa dikatakan lulus jika nilainya >=
60.
Deskripsi :
1. mulai
2. Baca nilai_siswa
3. Jika nilai_siswa >= 60 maka kerjakan langkah 4
4. Cetak “Siswa tersebut lulus”
5. selesai
```

Flowchart Kelulusan_Siswa :



Gambar Flowchart kelulusan siswa dengan satu pilihan

Dari flowchart diatas dapat dijelaskan bahwa setelah `nilai_siswa` dimasukkan maka akan diuji apakah `nilai_siswa` lebih besar atau sama dengan `60`? Jika benar maka akan dicetak “`Siswa tersebut lulus`” kemudian selesai, jika tidak maka selesai.

2. IF … THEN … ELSE …

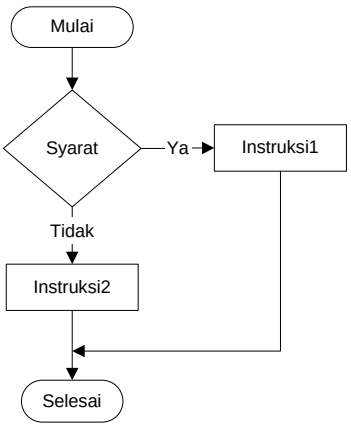
Bentuk :

```
IF <syarat> THEN
    <instruksi1>
```

```
ELSE
    <instruksi2>
```

Pada bentuk ini terdapat dua kemungkinan pilihan yang akan dikerjakan berdasarkan hasil pengujian, jika `syarat` yang diuji benilai benar maka `instruksi1` yang dikerjakan, dan jika `syarat` yang diuji bernilai salah maka `instruksi2` yang dikerjakan.

Flowchart :



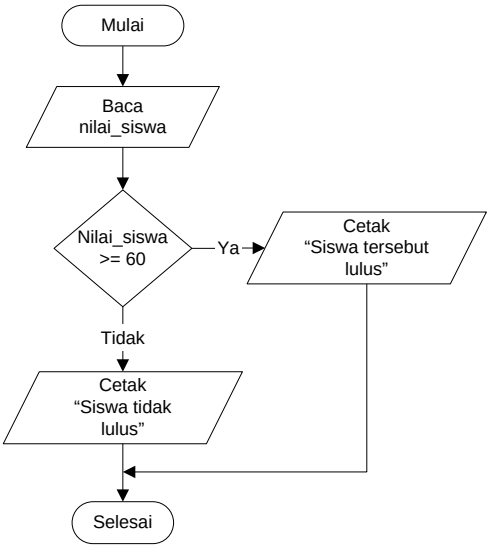
Gambar Flowchart If … Then … Else …

Contoh 1 :

Dibuat suatu aturan kelulusan seorang siswa yang diketahui dari hasil nilainya dalam bentuk angka. Seorang siswa dikatakan lulus jika nilai lebih besar atau sama dengan 60, dan jika nilainya lebih kecil dari 60 maka siswa tidak lulus.

```
Algoritma Kelulusan_Siswa
Diketahui seorang siswa dikatakan lulus jika nilainya >=
60, dan jika nilainya < 60 maka siswa tidak lulus.
Deskripsi :
1. mulai
2. Baca nilai_siswa
3. Jika nilai_siswa >= 60 maka kerjakan langkah 4, selain
   itu kerjakan langkah 5
4. Cetak “Siswa tersebut lulus”
5. Cetak “Siswa tidak lulus”
6. selesai
```

Flowchart Kelulusan_Siswa :



Gambar Flowchart kelulusan siswa dengan dua pilihan

Dari flowchart diatas dapat dijelaskan bahwa setelah `nilai_siswa` dimasukkan maka akan diuji apakah `nilai_siswa` lebih besar atau sama dengan `60`? Jika benar maka akan dicetak "`Siswa tersebut lulus`" kemudian selesai, jika tidak maka akan dicetak "Siswa tidak lulus" kemudian selesai.

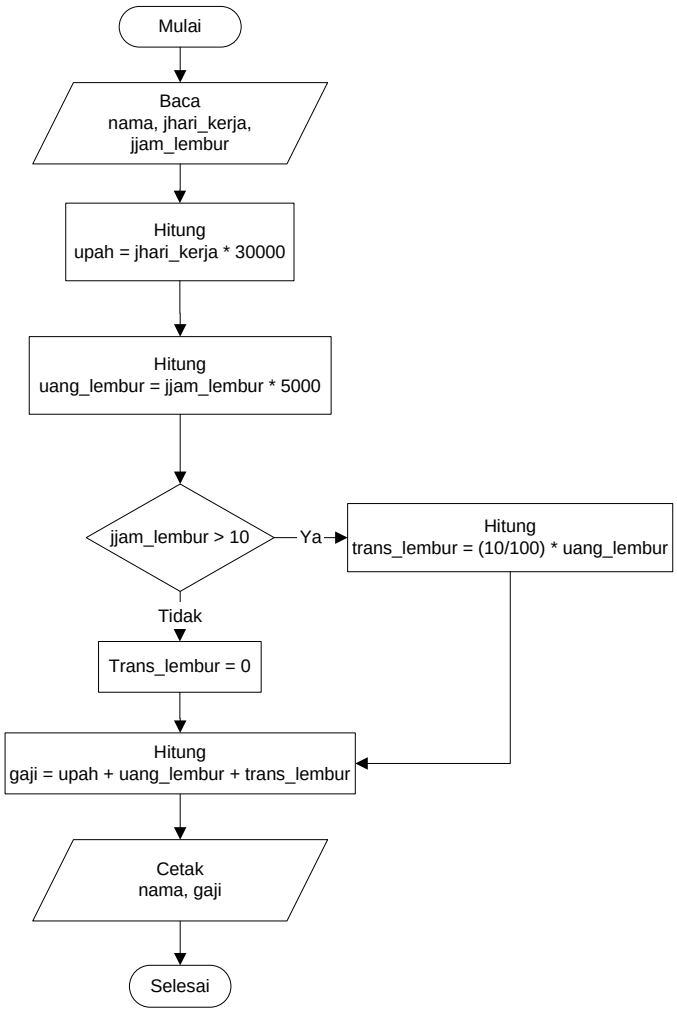
Contoh 2 :

Buatlah algoritma dan flowchart untuk menghitung jumlah pembayan gaji dengan input nama, jumlah hari kerja dan jumlah jam lembur. Tarif untuk hari kerja adalah Rp. 30.000,- per hari, sedangkan tarif perjam lembur adalah Rp. 5.000,-. Jika seorang karyawan jam lemburnya lebih dari 10 jam maka akan mendapatkan tambahan transport lembur sebesar 10% dari jumlah uang lembur, jika tidak maka tidak mendapatkan transport lembur.

Algoritma Pembayaran_Gaji

```
Diketahui input data nama, jumlah hari kerja dan jumlah
jam lembur, tarif per hari kerja Rp. 30.000, tarif per jam
lembur Rp. 5.000, jika jumlah jam lembur lebih dari 10 jam
maka akan mendapatkan tambahan uang transport lembur 10%
dari jumlah uang lembur.
Deskripsi :
1. mulai
2. Baca nama
3. Baca jhr_kerja
4. Baca jjam_lembur
5. Hitung upah = jhr_kerja * 30000
6. Hitung uang_lembur = jjam_lembur * 5000
7. Jika jjam_lebur > 10 maka kerjakan langkah 8 selain itu
   kerjakan langkah 9
8. Hitung trans_lembur = (10/100) * uang_lembur
9. trans_lembur = 0
10.Hitung gaji = upah + uang_lembur + trans_lembur
11.Tampilkan gaji
12.selesai
```

Flowchart Pembayaran_Gaji :



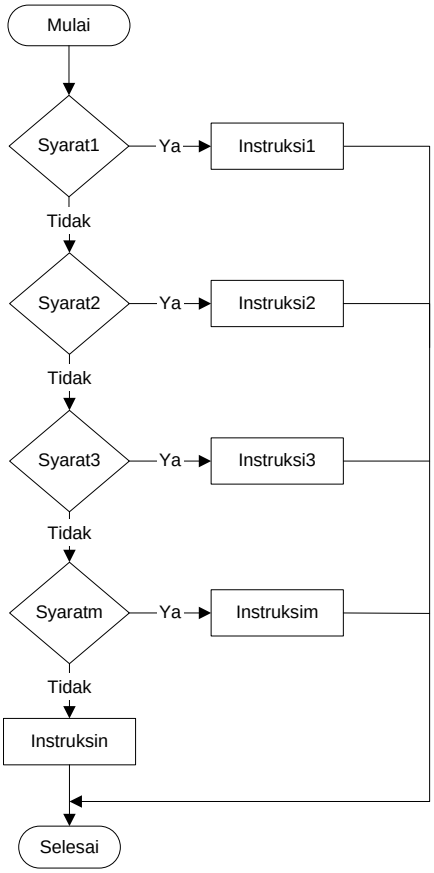
Gambar flowchart perhitungan gaji

Dari flowchart diatas dapat dijelaskan bahwa setelah `nama, jhari_kerja,jjam_lembur` dimasukkan maka akan dihitung besarnya `upah`, kemudian dihitung besarnya `uang_lembur`, kemudian diuji apakah `jjam_lembur > 10`, jika benar maka dihitung `trans_lembur` 10% dari `uang_lembur`, jika salah maka `trans_lembur = 0`, kemudian dihitung besar `gaji` yang diperoleh. Terakhir dicetak berupa `nama` dan `gaji`, kemudian selesai.

3. IF Bersarang (*Nested* IF)
   Bentuk :

```
IF <syarat1> THEN
    <instruksi1>
ELSE IF <syarat2> THEN
    <instruksi2>
ELSE IF <syarat3> THEN
    <instruksi3>
ELSE IF <syaratm> THEN
    <instruksim>
ELSE
    <Instruksin>
```

Flowchart :



Gambar Flowchart bersarang

Pada bentuk ini terdapat banyak kemungkinan pilihan yang akan dikerjakan berdasarkan hasil pengujian, proses pengujiannya adalah :
jika `syarat1` yang diuji benilai benar maka `instruksi1` yang dikerjakan, jika `syarat1` yang diuji bernilai salah maka `syarat2` diuji, jika `syarat2` bernilai benar maka `instruksi2` yang dikerjakan, jika `syarat2` bernilai salah maka `syarat3` yang diuji, jika `syarat3` bernilai benar maka `instruksi3` yang dikerjakan, jika `syarat3` bernilai salah maka `syaratm` yang diuji, jika `syaratm` bernilai benar maka `instruksim` yang dikerjakan, begitu seterusnya, jika tidak ada syarat yang terpenuhi maka `instruksin` yang dikerjakan.

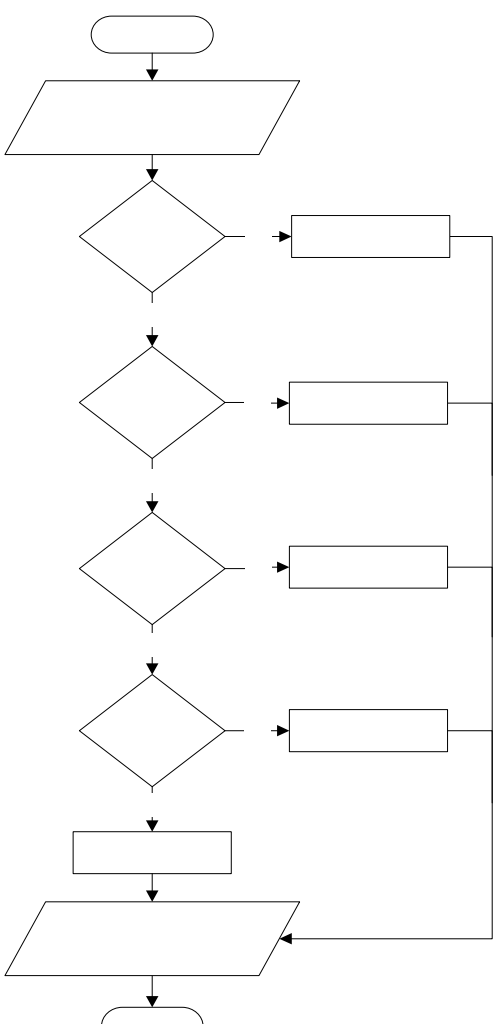
Contoh :

Buatlah algoritma dan flowchart untuk menghitung konfersi nilai siswa, input berupa nama siswa dan nilai berupa nilai angka. Hasilnya akhir adalah berupa nilai 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

Algoritma Konfersi_Nilai
Diketahui nilai angka seorang siswa yang akan dikonfersikan ke nilai huruf.
Deskripsi :
1. mulai
2. Baca nama_siswa
3. Baca nilai_angka
4. Jika nilai_angka >= 80 maka nilai_huruf = “A”, selain itu

```
5. jika nilai_angka >= 70 maka nilai_huruf = “B”, selain
   itu
6. jika nilai_angka >= 60 maka nilai_huruf = “C”, selain
   itu
7. jika nilai_angka >= 50 maka nilai_huruf = “D” selain
   itu
8. nilai_huruf = “E”
9. Cetak nama_siswa dan nilai_huruf
10.selesai
```

Flowchart Konfersi_Nilai :



Gambar Flowchart konfersi nilai angka ke nilai huruf

Pada bentuk IF bersarang ini yang perlu diperhatikan adalah bahwa jika suatu syarat sudah terpenuhi maka syarat lain yang ada dibawahnya tidak akan diuji lagi. Pada contoh diatas misalkan nilai_angka yang diinputkan 75 maka nilai hurufnya adalah B (lihat bentuk flowchartnya), sehingga pengujian tidak dilanjutkan lagi untuk kondisi dibawahnya. Dengan kata lain input nilai_angka 75 tidak akan diujikan untuk apakah nilai_angka >= 60, apakah nilai_angka >=50 atau apakah nilai_angka <50.

Latihan struktur urut dan pemilihan:

1. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menukarkan isi dua buah nilai variabel yang diinputkan.
2. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menentukan nilai terbesar diantara dua buah input.

3. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menentukan nilai terbesar diantara tiga buah input.
4. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menentukan input bilangan bulat termasuk bilangan genap atau ganjil atau atau nol.
5. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menentukan input bilangan bulat termasuk bilangan genap positif atau ganjil positif atau genap negatif atau ganjil negatif atau nol.

ii. Struktur Pengulangan (*repetition*)

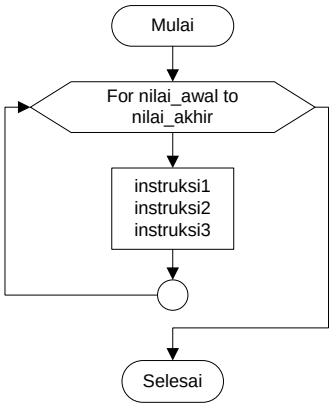
Struktur pengulangan merupakan struktur yang melakukan pengulangan terhadap satu baris atau satu blok baris program beberapa kali sesuai dengan persyaratan yang diberikan.

Struktur pengulangan mempunyai beberapa bentuk :

1. 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.

   Flowchart struktur `for`



Gambar flowchart struktur `for`

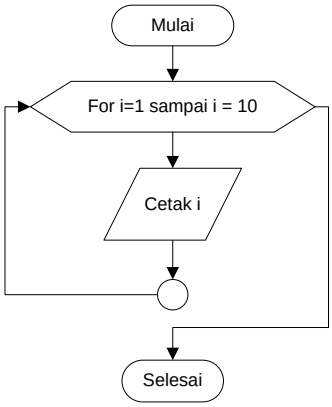
Dari gambar flowchart diatas dapat dijelaskan bahwa `instruksi1, instruksi2, instruksi3` akan dikerjakan berulang yang dimulai dari `nilai_awal` sampai `nilai_akhir` yang diberikan. Jika pengulangan sudah sampai pada kondisi `nilai_akhir` yang diberikan maka pengulangan akan berhenti.

Contoh 1:

Akan dicetak angka 1 sampai 10 dengan menggunakan perulangan for

```
Algoritma Cetak_Angka_for
Dicetak angka 1 sampai 10 dengan perulangan for.
Deskripsi :
1. mulai
2. kerjakan langkah 3 mulai i = 1 sampai i = 10
3. cetak i
4. selesai
```

Flowchart `Cetak_Angka dengan for`



Gambar flowchart cetak angka dengan `for`

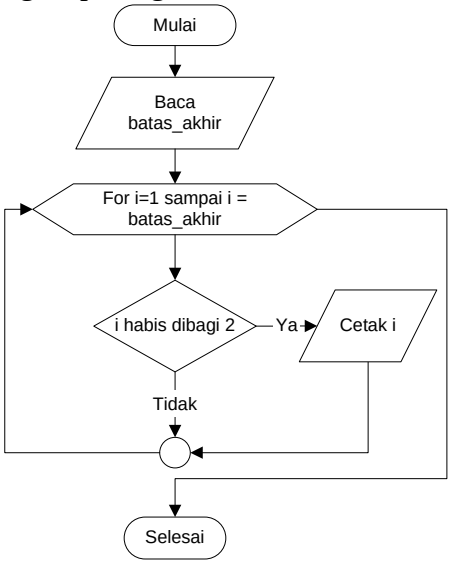
Dari gambar flowchart diatas dapat dijelaskan bahwa nilai i pertama akan berisi 1, kemudian dicetak nilai i, dalam perulangan for nilai variabel i akan bertambah secara otomatis sehingga nilai variabel i sekarang menjadi 2, kemudian dicetak nilai i, begitu seterusnya sampai nilai i berisi 10, maka proses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

Contoh 2 :

Akan dicetak bilangan genap mulai dari 0 dengan batas akhir diinputkan dari keyboard dengan menggunakan pengulangan for.

```
Algoritma Cetak_bilangan_genap_for
Dicetak bilangan genap dengan batas akhir diinputkan
dengan menggunakan for.
Deskripsi :
1. mulai
2. Baca batas_akhir
3. Kerjakan langkah 4 sampai langkah 5 mulai i = 1 sampai
   i = batas_akhir
4. jika i habis dibagi 2 maka kerjakan langkah 5
5. cetak i
6. selesai
```

Flowchart cetak bilangan genap dengan for :



Gambar flowchart cetak bilangan genap dengan `for`

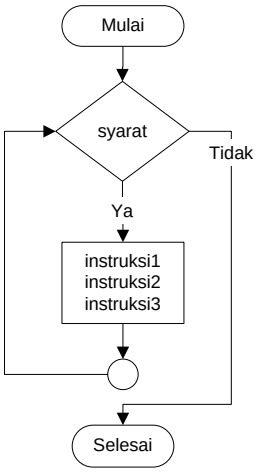
Dari gambar flowchart diatas dapat dijelaskan pertama kali dibaca `batas_akhir` perulangan, kemudian nilai `i` pertama kali akan berisi 1, kemudian akan diuji apakah nilai `i` habis dibagi dua, jika benar maka dicetak nilai `i`, kemudian pengulangan dilanjutkan dengan nilai `i` menjadi 2, jika tidak maka pengulangan

akan dilanjutkan dengan nilai `i` menjadi 2, begitu seterusnya sampai nilai `i` lebih besar `batas_akhir`.

2. Struktur `while`

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

   Flowchart struktur `while`



Gambar flowchart struktur `while`

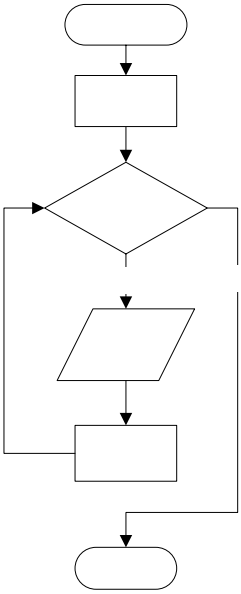
Dari gambar diatas dapat dijelaskan bahwa syarat akan diuji terlebih dahulu sebelum masuk blok yang diulang. Jika syarat yang diuji bernilai benar maka `instruksi1, instruksi2, instruksi3` akan dikerjakan, setelah mengerjakan `instruksi1, instruksi2, instruksi3` maka syarat akan diuji lagi. Jika syarat yang diuji bernilai benar maka `instruksi1, instruksi2, instruksi3` akan dikerjakan lagi, pengulangan akan berhenti jika syarat yang diuji bernilai salah.

Contoh :

Akan dibuat contoh diatas dengan menggunakan while

```
Algoritma Cetak_Angka_while
Dicetak angka 1 sampai 10 dengan perulangan while.
Deskripsi :
1. mulai
2. i = 1
3. selama i <= 10 kerjakan langkah 4 sampai langkah 5
4. cetak i
5. i = i + 1
6. selesai
```

Flowchart `Cetak_Angka`



Gambar flowchart cetak angka dengan while

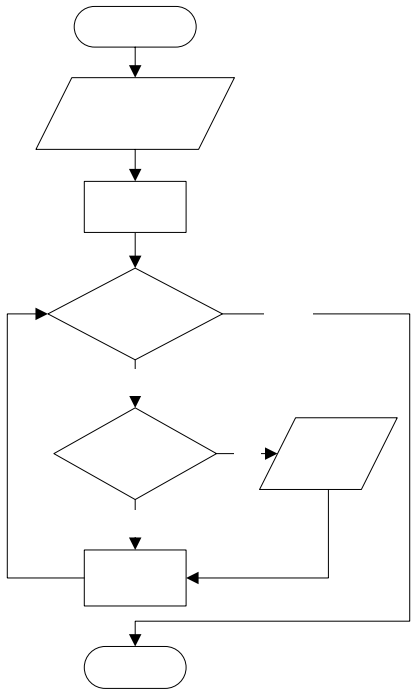
Dari gambar flowchart diatas dapat dijelaskan pertama kali i bernilai 1, kemudian diuji apakah i lebih kecil atau sama dengan 10, jika benar maka dicetak nilai i, kemudian nilai i dinaikkan sebesar 1, kemudian nilai i diuji kembali apakah masih lebih kecil atau sama dengan 10 jika benar maka dicetak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

Contoh :

Akan dibuat contoh diatas dengan menggunakan while

```
Algoritma Cetak_bilangan_genap_while
Dicetak bilangan genap dengan batas akhir diinputkan
dengan menggunakan while.
Deskripsi :
1. mulai
2. Baca batas_akhir
3. i = 1
4. selama i <= batas_akhir kerjakan langkah 5 sampai
   langkah 7
5. jika i habis dibagi 2 kerjakan langkah 6
6. cetak i
7. i = i + 1
8. selesai
```

Flowchart :



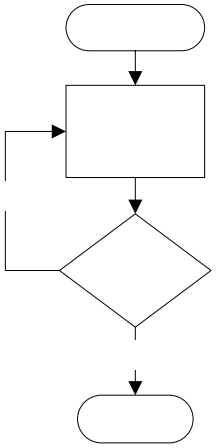
Gambar flowchart cetak bilangan genap dengan while

Dari gambar flowchart diatas dapat dijelaskan pertama kali dibaca `batas_akhir` perngulangan, kemudian `i` diberi nilai 1, kemudian diuji apakah `i` lebih kecil atau sama dengan `batas_akhir`, jika benar maka diuji apakah nilai `i` habis dibagi 2, jika benar maka dicetak nilai `i`, kemudian nilai `i` dinaikkan sebesar 1 sehingga nilai `i` menjadi 2, jika tidak maka nilai `i` langsung dinaikkan 1 sehingga nilai `i` menjadi 2, kemudian nilai `i` diuji kembali apakah masih lebih kecil atau sama dengan `batas_akhir` jika benar maka diuji apakah nilai `i` habis dibagi 2, jika benar maka dicetak nilai `i`, kemudian nilai `i` dinaikkan sebesar 1 menjadi 3, jika tidak maka nilai i langsung dinaikkan 1 menjadi 2, begitu seterusnya sampai nilai i lebih besar `batas_akhir` sehingga perulangan berakhir.

3. Struktur `do ... while`

   Struktur pengulangan dengan instruksi `do...while` digunakan untuk mengulang satu baris instruksi atau satu blok baris instruksi sampai syarat tidak terpenuhi. Ciri utama pengulangan `do...while` adalah syarat akan uji setelah instruksi yang akan diulang dikerjakan, dengan kata lain dalam instruksi `do...while` syarat akan diuji dibelakang, sehingga baris instruksi yang masuk dalam blok `do...while` minimal akan dikerjakan satu sekali.

   Flowchart struktur `do...while`



Gambar flowchart struktur `do...while`

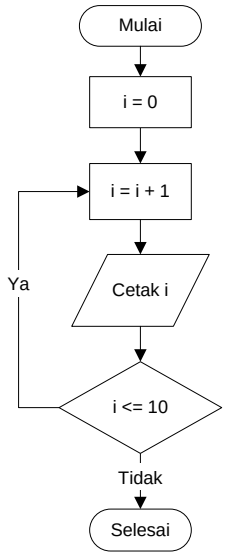
Dari gambar diatas dapat dijelaskan bahwa `instruksi1, instruksi2, instruksi3` akan dikerjakan terlebih dahulu baru syarat diuji. Jika syarat yang diuji bernilai benar maka `instruksi1, instruksi2, instruksi3` akan dikerjakan lagi, setelah itu syarat diuji lagi, pengulangan akan berhenti jika syarat yang diuji bernilai salah.

Contoh :

Akan dibuat contoh diatas dengan menggunakan do…while

```
Algoritma Cetak_Angka_do_while
Dicetak angka 1 sampai 10 dengan perulangan while.
Deskripsi :
1. mulai
2. i = 0
3. i = i + 1
4. cetak i
5. jika i < 10 kerjakan langkah 3 sampai langkah 4
6. selesai
```

Flowchart `Cetak_Angka`



Gambar flowchart cetak angka dengan do…while

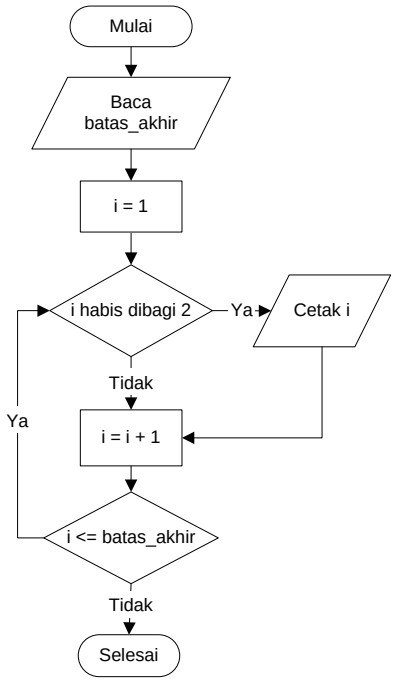
Dari gambar flowchart diatas dapat dijelaskan pertama kali i diberi nilai awal 0, kemudian nilai i dinaikkan sebesar 1 sehingga nilai i menjadi 1, kemudian nilai i dicetak. Setelah dicetak nilai i diuji apakah i lebih kecil atau sama dengan 10, jika banar maka nilai i dinaikkan 1, sehingga i menjadi 2, kemudian nilai i dicetak. Setelah itu nilai i diuji lagi apakah i lebih keci atau sama dengan 10, begitu seterusnya sampai nilai i lebih besar 10 maka perulangan akan berhenti.

Contoh :

Akan dibuat contoh diatas dengan menggunakan do…while

```
Algoritma Cetak_bilangan_genap_do_while
Dicetak bilangan genap dengan batas akhir diinputkan
dengan menggunakan do..while.
Deskripsi :
1. mulai
2. Baca batas_akhir
3. i = 1
4. Selama i <= batas_akhir kerjakan langkah 5 sampai
   langkah 7
5. jika i habis dibagi 2 kerjakan langkah 6
6. cetak i
7. i = i + 1
8. selesai
```

Flowchart :



Gambar flowchart cetak bilangan genap dengan do…while

Dari gambar flowchart diatas dapat dijelaskan pertama kali dibaca batas_akhir perngulangan, kemudian i diberi nilai awal 1, setelah itu diuji apakah nilai habis dibagi 2, jika benar maka cetak nilai i, kemudian nilai i dinaikkan 1 sehingga i menjadi 2, jika tidak maka nilai i langsung dinaikkan sebesar 1, sehingga nilai i menjadi 2. Setelah diuji apakah nilai i lebih kecil atau sama dengan batas_akhir, jika benar maka kembali diuji setelah itu diuji apakah nilai habis dibagi 2, jika benar maka cetak nilai i, kemudian nilai i dinaikkan 1 sehingga i menjadi 3, jika tidak maka nilai i langsung dinaikkan sebesar 1, sehingga nilai i menjadi 3. Setelah diuji apakah nilai i lebih kecil atau sama dengan batas_akhir, begitu seterusnya sampai nilai i lebih besar batas_akhir sehingga perulangan selesai.

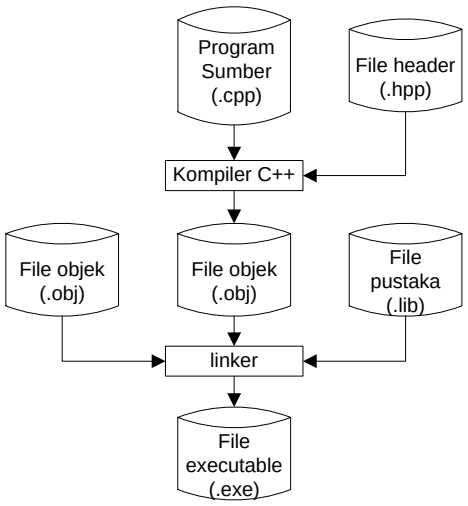
Latihan Pengulangan :

1. Buatlah algoritma dengan struktur indonesia dan flowchart untuk mencari rata-rata dari sejumlah angka yang diinputkan dengan menggunakan pengulangan.
2. Buatlah algoritma dengan struktur indonesia dan flowchart untuk mencari angka terbesar dan angka terkecil dari sejumlah angka yang diinputkan dengan menggunakan pengulangan.
3. Buatlah algoritma dengan struktur indonesia dan flowchart untuk menghitung total pembayaran pembelian. Input berupa nama pembeli, dan data barang dengan jumlah barang yang dibeli berupa input nama barang, jumlah dan harga barang. Barang yang dibeli jumlahnya bisa banyak tergantung pembelian dari konsumen.

BAB 6
DASAR BAHASA PEMROGRAMAN BORLAND C++

1. Proses kompilasi

Program C++ ditulis dengan ekstensi .cpp. 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 Proses pembentukan file executable

2. Struktur program C++

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

#include adalah pengarah praprosesor yang berfungsi menginstruksikan kepada kompiler untuk menyisikan file lain saat program dikompilasi. Biasanya file-file yang disisipkan adalah file-file header

void didepan main() dipakai untuk menyatakan bahwa fungsi main() tidak memiliki nilai balik.

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.

Blok pernyataan

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

Contoh program :

```
/*
   Program yang mengandung blok pernyataan
*/

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

Hasil eksekusi :

```
Selamat Belajar C++
```

Untuk mengkompile program dengan menggunakan borland C++ dapat dilakukan dengan menekan tombol Ctrl + F9 atau tekan tombol pada toolbar.

Mengenal cout

Pengenal cout (baca : c out) merupakan objek dalam C++ yang digunakan untuk mengarahkan data ke standar output (layar). 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.

Pada contoh di atas

"Selamat Belajar C++\n" diarahkan ke cout, yang memberikan hasil berupa tampilan string tersebut ke layar. \ n adalah karakter pindah baris (*new line*).

#include <iostream.h>

`#include <iostream.h>` menginstruksikan kepada kompiler untuk menyisipkan file `iostream.h` pada saat program dikompilasi tanpa diakhiri titik koma. File `iostream.h` perlu disertakan pada program yang melibatkan `cout`. Tanpa `#include <iostream.h>` akan terjadi kesalahan saat program dikompilasi. Sebab file `iostream.h` berisi deklarasi yang diperlukan oleh `cout` dan berbagai objek yang berhubungan dengan operasi masukan–keluaran.

clrscr();

Pernyataan yang diperlukan untuk menghapus layar. Apabila menggunakan pernyataan ini maka harus disertakan file header `conio.h`.

Komentar

Komentar diperlukan untuk menjelaskan mengenai program atau bagian-bagian dalam program. Isi penjelasan berupa:

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

Tanda awal komentar dalam program C++ ada dua cara:

1. Diawali tanda // (dua tanda garis miring)

Semua tulisan setelah tanda // dianggap sebagai komentar dan tidak akan dieksekusi oleh C++.

Contoh program :

```
/*
   Program yang mengandung komentar
*/

#include <iostream.h>
#include <conio.h>
void main()
{
   clrscr();//membarsihkan layar
   // teks ini sampai akhir baris tidak akan pernah dieksekusi

   cout << "Selamat Belajar C++\n";
   getch();
}
```

2. Diawali tanda /* blok komentar dan diakhiri tanda */

Bentuk ini bermanfaat untuk mengabaikan sejumlah baris pernyataan yang telah dibuat karena suatu alasan.

Contoh program :

```
/*
   Program yang mengandung komentar
*/

#include <iostream.h>
#include <conio.h>
void main()
{
   clrscr();      //membersihkan layar
   /*
    mulai blok komentar pada baris ini tidak akan dieksekusi
    sampai ditemui akhir blok komentar
   */

   cout << "Selamat Belajar C++\n";
   getch();
}
```

3. Elemen Dasar Dalam C++

3.1 Pengenal (Identifier)

Pengenal merupakan nama yang akan digunakan dalam pemrogrman yang biasa digunakan untuk 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 *sensitive case* atau dibedakan antara huruf kecil dan huruf besar. Misalkan pengenal gajipokok, GajiPokok, GAJIPOKOK, merupakan tiga buah pengenal yang berbeda.

3.2 Tipe Data Dalam C++

Tipe data	Ukuran Memori	Jangkauan Nilai	Keterangan
char	1 byte	-128 sampai 128	Digunakan untuk menampung data berupa sebuah karakter. Penulisan data dalam bentuk karakter diapit dengan tanda petik tunggal.
int	2 byte	-32.768 sampai 32.768	Digunakan untuk menampung bilangan bulat dan tidak mengandung digit desimal.
Short	2 byte	-32.768 sampai 32.767	Digunakan untuk menampung bilangan bulat dan tidak mengandung digit desimal.
long	4 byte	-2.147.438.648 sampai 2.147.438.647	Digunakan untuk menampung bilangan bulat yang jangkaunnya lebih besar dari tipe **int**.
float	4 byte	3.4×10^{-38} sampai 3.4×10^{38}	Digunakan untuk menampung bilangan pecahan yang mengandung digit desimal.
double	8 byte	1.7×10^{-308} sampai 1.7×10^{308}	Digunakan untuk menampung bilangan pecahan yang jangkaunnya lebih besar dari tipe **float**.
long double	10 byte	3.4×10^{-4932} sampai 1.1×10^{4932}	Digunakan untuk menampung bilangan pecahan yang jangkaunnya lebih besar dari tipe **double**.

Contoh program :

```
/*
  program yang mengandung tipe data dalam c++
*/

#include <iostream.h>
#include <conio.h>
void main()
{
   clrscr();
   char huruf;
   huruf = 'B';
   cout<<"Isi huruf bertipe char        = "<<huruf<<'\n';
   int angka;
   angka = 123;
   cout<<"Isi angka bertipe int         = "<<angka<<'\n';
   long jumlah;
   jumlah = 12345678;
   cout<<"Isi jumlah bertipe long       = "<<jumlah<<'\n';
   float nilai;
   nilai = 234.543;
   cout<<"Isi nilai bertipe float       = "<<nilai<<'\n';
   double cacah;
   cacah = 3453.345;
   cout<<"Isi cacah bertipe double      = "<<cacah<<'\n';
   long double total;
   total = 23456.3945;
   cout<<"Isi total bertipe long double = "<<total<<'\n';
```

```
   getch();
}
```

Hasil eksekusi :

```
Isi huruf bertipe char        = B
Isi angka bertipe int         = 123
Isi jumlah bertipe long       = 12345678
Isi nilai bertipe float       = 234.543
Isi cacah bertipe double      = 3453.34
Isi total bertipe long double = 23456.4
```

3.3 Variabel dan Konstanta

Dalam proses pemrograman tipe data biasa digunakan untuk mendefinisikan suatu variabel atau konstanta. Variabel adalah suatu memori yang dialokasikan dengan nama 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. Sedangkan konstanta adalah suatu memori yang dialokasikan dengan nama tertentu yang berisi suatu nilai yang memiliki sifat tetap yang tidak akan bisa berubah.

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

<tipevariabel> <namavariabel>;

Untuk memasukkan nilai kedalam variabel digunakan bentuk :

variabel = nilai;

Contoh program :

```
/*
  program yang mengandung pemasukkan nilai pada variabel
*/

#include <iostream.h>
#include <conio.h>
void main()
{
   int bilangan;    //Mendefinisikan variabel bilangan bertipe int
   bilangan = 10;   //Memberikan nilai 10 pada variabel bilangan
   clrscr();
   cout<<"Isi bilangan = "<<bilangan<<'\n';
   bilangan = 50;
   cout<<"Isi bilangan = "<<bilangan<<'\n';
   getch();
}
```

Hasil eksekusi :

```
Isi bilangan = 10
Isi bilangan = 50
```

Untuk mendeklarasikan konstanta digunakan bentuk :

const <tipedata><namakonstanta> = <nilaikonstanta>

Contoh program :

```
/*
  program yang mengandung konstanta
```

```
*/

#include <iostream.h>
#include <conio.h>
void main()
{
   const float phi = 3.14;
   int jari = 11;
   float luas;
   clrscr();
   luas = phi * jari * jari;
   cout<<"Luas lingkaran = "<<luas<<'\n';
   getch();
}
```

Hasil eksekusi :

```
Luas lingkaran = 379.94
```

Dalam C++ dikenal adanya pemodifikasian variabel menggunakan **unsigned** dan **signed.** Variabel yang ditambahi **unsigned** akan menyebabkan nilai yang dikandung variabel benilai positif, sedangkan **signed** tidak menyebabkan perubahan nilai dari data yang dikandung (sama dengan nilai data dasar).

Pemodifikasian Tipe	Persamaan	Jangkauan Nilai
unsigned char	Tidak ada	0 s/d 255
unsigned int	unsigned	0 s/d 65.535
unsigned short int	unsigned short	0 s/d 65.535
unsigned long int	unsigned long	0 s/d 4.294.967.295
signed char	char	-128 s/d 127
signed int	int	-32.768 s/d 32.767
Signed short int	short, signed short	-32.768 s/d 32.767
Signed long int	long, long int, signed long	-2.147.483.648 s/d 2.147.483.687

3.4 Operator dan ungkapan

Operator merupakan simbol yang akan digunakan untuk melakukan suatu operasi atau manipulasi. Untuk menghasilkan suatu nilai operator harus menerima operand. Sedangkan ungkapan merupakan gabungan antara operator dengan operand yang menghasilkan nilai ungkapan.

Contoh :

6 + 10 – 4

Pada ungkapan diatas terdiri dari dua operator yaitu + (penjumlahan) dan - (pengurangan) dan tiga operand yaitu 6, 10 dan 4, nilai ungkapan diatas adalah 12.

Operator aritmatika

Operator	Keterangan	Contoh
*	Perkalian	10 * 5
/	Pembagian	20 / 4
%	Sisa pembagian (modulus)	22 % 3
+	Penjumlahan	5 + 6
-	Pengurangan	8 - 4

Contoh program :

```
/*
      contoh penggunaan operator aritmatika
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int d,b,a,c;
  b = 7;
  a = 5;
  c = 2;
  d = b * b - 4 * a * c;
  cout<<"Nilai d = "<<d<<'\n';

  //mencari sisa bagi
  int sisa;
  sisa = 20 % 3;
  cout<<"Sisa 20 % 3 = "<<sisa<<'\n';
  getch();
}
```

Hasil eksekusi :

```
Nilai d = 9
Sisa 20 % 3 = 2
```

Operator penugasan

Operator penugasan menggunakan simbol sama dengan (=) yang berfungsi untuk memberikan nilai pada suatu variabel. Contoh penggunaan operator penugasan dalam C++ :

Contoh	Keterangan
y = 4 + (a = 7);	Variabel a diberi nilai 7, kemudian y diisi dengan nilai ungkapan 4 + 7
a = b = c = d = e = 10;	Pertama e diisi dengan 10, kemudian d diisi dengan nilai e, kemudian c diisi dengan nilai d, kemudian b diisi dengan nilai c, kemudian a diisi dengan nilai b.

Contoh program :

```
/*
      contoh penggunaan operator penugasan
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int a,b,c,d,e;
  a = b = c = d = e = 100;
  cout<<"Isi a = "<<a<<'\n';
  cout<<"Isi b = "<<b<<'\n';
  cout<<"Isi c = "<<c<<'\n';
  cout<<"Isi d = "<<d<<'\n';
  cout<<"Isi e = "<<e<<'\n';
  int x,y;
```

```
  x = 10 * (y = 3);
  cout<<"Isi x = "<<x<<'\n';
  getch();
}
```

Hasil eksekusi :

```
Isi a = 100
Isi b = 100
Isi c = 100
Isi d = 100
Isi e = 100
Isi y = 3
Isi x = 30
```

Operator penaikkan (*increment*) dan penurunan (*decrement*)

Digunakan untuk menaikkan dan menurunkan nilai variabel yang bertipe bilangan bulat. Bentuk operator ini adalah ++ (penaikkan) dan -- (penurunan). Sebagai contoh :

```
x = x + 1;
y = y - 1;
```

Dapat ditulis :

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

atau

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

Contoh :

```
/*
      contoh penggunaan operator penaikkan dan penurunan
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int a,b,c;
  a = 15;

  //variabel a dinaikkan 1 baru dijumlahkan dengan 10
  b = 10 + ++a;

  cout<<"Isi b = "<<b<<'\n';

  //variabel b dijumlahkan dengan 10 baru b dinaikkan 1
  c = 10 + b++;
  cout<<"Isi b = "<<b<<'\n';
  cout<<"Isi c = "<<c<<'\n';
  getch();
}
```

Hasil eksekusi :

```
Isi b = 26
Isi b = 27
Isi c = 36
```

Operator bitwise

Digunakan untuk memanipulasi bilangan biner

Operator	Keterangan	Contoh
<<	Geser bit ke kiri	10 << 3
>>	Geser bit ke kanan	10 >> 3
&	Bitwise and	10 & 3
|	Bitwise or	10 | 3
^	Bitwise xor	10 ^ 3
~	Bitwise not	~10

Prioritas operator bitwise

Prioritas	Operator
Tertinggi	~
	<< >>
	&
	^
Terendah	|

Contoh program :

```
/*
      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
```

Operator majemuk

Operator majemuk digunakan untuk memendekkan penulisan suatu penugasan.

x = x + 5;
y = y – 8;

Dapat ditulis :

x += 5;
y -= 8;

Contoh program :

```
/*
      contoh penggunaan operator majemuk
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int a;

  a = 20;
  cout<<"Nilai a sekarang = "<<a<<'\n';

  a += 5;
  cout<<"Nilai a += 5 sekarang = "<<a<<'\n';

  a -= 3;
  cout<<"Nilai a -= 3 sekarang = "<<a<<'\n';

  a *= 2;
  cout<<"Nilai a *= 2 sekarang = "<<a<<'\n';

  a %= 3;
  cout<<"Nilai a %= 3 sekarang = "<<a<<'\n';

  a <<= 1; //a = a << 1;
  cout<<"Nilai a <<= 1 sekarang = "<<a<<'\n';

  a >>= 1;  //a = a >> 1;
  cout<<"Nilai a >>= 1 sekarang = "<<a<<'\n';

  getch();
}
```

Hasil eksekusi :

```
Nilai a sekarang = 20
Nilai a += 5 sekarang = 25
Nilai a -= 3 sekarang = 22
Nilai a *= 2 sekarang = 44
Nilai a %= 3 sekarang = 2
Nilai a <<= 1 sekarang = 4
Nilai a >>= 1 sekarang = 2
```

Operator relasi

Operator relasi digunakan untuk membandingkan dua buah nilai, yang hasilnya berupa nilai 1 jika benar dan nalai 0 jika salah.

Operator	Keterangan
= =	Sama dengan (bukan penugasan)
!=	Tidak sama dengan
<	Lebih kecil
>	Lebih besar
<=	Lebih kecil atau sama dengan
>=	Lebih besar atau sama dengan

Contoh program :

```
/*
      contoh penggunaan operator relasi
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int a,b,hasil;

  a = 45;
  b = 25;

  hasil = a == b;
  cout<<"Hasil relasi "<<a<<" == "<<b<<" = "<<hasil<<'\n';

  hasil = a != b;
  cout<<"Hasil relasi "<<a<<" != "<<b<<" = "<<hasil<<'\n';

  hasil = a > b;
  cout<<"Hasil relasi "<<a<<" > "<<b<<" = "<<hasil<<'\n';

  hasil = a < b;
  cout<<"Hasil relasi "<<a<<" < "<<b<<" = "<<hasil<<'\n';

  hasil = a >= b;
  cout<<"Hasil relasi "<<a<<" >= "<<b<<" = "<<hasil<<'\n';

  hasil = a <= b;
  cout<<"Hasil relasi "<<a<<" <= "<<b<<" = "<<hasil<<'\n';

  getch();
}
```

Hasil eksekusi :

```
Hasil relasi 45 == 25 = 0
Hasil relasi 45 != 25 = 1
Hasil relasi 45 > 25 = 1
Hasil relasi 45 < 25 = 0
Hasil relasi 45 >= 25 = 1
Hasil relasi 45 <= 25 = 0
```

Operator logika

Operator logika digunakan untuk menghubungkan dua buah kondisi menjadi sebuah kondisi.

Operator	Keterangan	Contoh
&& (dan)	Akan bernilai benar jika dua buah kondisi yang dihubungkan bernilai benar. Selain itu bernilai salah	(5>2) && (4>3) → 1 && 1 hasil 1 (5<2) && (4>3) → 0 && 1 hasil 0 (5>2) && (4<3) → 1 && 0 hasil 0 (5<2) && (4<3) → 0 && 0 hasil 0
|| (atau)	Akan bernilai salah jika kedua kondisi yang dihubungkan bernilai salah, selain itu bernilai benar.	(5>2) || (4>3) → 1 || 1 hasil 1 (5<2) || (4>3) → 0 || 1 hasil 1 (5>2) || (4<3) → 1 || 0 hasil 1 (5<2) || (4<3) → 0 || 0 hasil 0
! (bukan)	Akan menghasilkan kebalikan dari operand yang diberikan. Jika operand bernilai salah maka hasilnya benar dan jika operand bernilai benar maka hasilnya salah.	!(5>4) → !(1) hasil 0 !(5<4) → !(0) hasil 1

Contoh program :

```
/*
      contoh penggunaan operator logika
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int hasil;

  cout<<"Operator logika && (dan)"<<'\n';
  hasil = (5 > 2) && (4 > 3);
  cout<<"Hasil logika (5 > 2) && (4 > 3)= "<<hasil<<'\n';
  hasil = (5 < 2) && (4 > 3);
  cout<<"Hasil logika (5 < 2) && (4 > 3)= "<<hasil<<'\n';
  hasil = (5 > 2) && (4 < 3);
  cout<<"Hasil logika (5 > 2) && (4 < 3)= "<<hasil<<'\n';
  hasil = (5 < 2) && (4 < 3);
  cout<<"Hasil logika (5 > 3) && (4 > 3)= "<<hasil<<'\n';

  cout<<"\nOperator logika || (atau)"<<'\n';
  hasil = (5 > 2) || (4 > 3);
  cout<<"Hasil logika (5 > 2) || (4 > 3)= "<<hasil<<'\n';
  hasil = (5 < 2) || (4 > 3);
  cout<<"Hasil logika (5 < 2) || (4 > 3)= "<<hasil<<'\n';
  hasil = (5 > 2) || (4 < 3);
  cout<<"Hasil logika (5 > 2) || (4 < 3)= "<<hasil<<'\n';
  hasil = (5 < 2) || (4 < 3);
  cout<<"Hasil logika (5 > 3) || (4 > 3)= "<<hasil<<'\n';
```

```
  cout<<"\nOperator logika !"<<'\n';
  hasil = !(5 > 2);
  cout<<"Hasil logika !(5 > 2) = "<<hasil<<'\n';
  hasil = !(5 < 2);
  cout<<"Hasil logika !(5 < 2) = "<<hasil<<'\n';

  getch();
}
```

Hasil eksekusi :

```
Operator logika && (dan)
Hasil logika (5 > 2) && (4 > 3)= 1
Hasil logika (5 < 2) && (4 > 3)= 0
Hasil logika (5 > 2) && (4 < 3)= 0
Hasil logika (5 > 3) && (4 > 3)= 0

Operator logika || (atau)
Hasil logika (5 > 2) || (4 > 3)= 1
Hasil logika (5 < 2) || (4 > 3)= 1
Hasil logika (5 > 2) || (4 < 3)= 1
Hasil logika (5 > 3) || (4 > 3)= 0

Operator logika !
Hasil logika !(5 > 2) = 0
Hasil logika !(5 < 2) = 1
```

Operator kondisi

Operator kondisi digunakan untuk mendapatkan sebuah nilai dari dua kemungkinan berdasarkan suatu kondisi. Bentuk operator kondisi :

Ungkapan1?ungkapan2:ungkapan3

Jika *ungkapan1* diuji bernilai benar maka hasilnya adalah *ungkapan2*, jika salah maka hasilnya adalah *ungkapan3*.

Contoh :

```
/*
      contoh penggunaan operator kondisi
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int angka1, angka2, maks;
  angka1 = 70;
  angka2 = 90;

  maks = (angka1 > angka2)?angka1:angka2;
  cout<<"Angka terbesar = "<<maks<<'\n';

  getch();
}
```

Hasil eksekusi :

```
Angka terbesar = 90
```

3.5 Input dalam C++

1. cout (baca C out)

cout merupakan obyek dalam C++ yang berfungsi untuk menampilkan data ke standar output (layar). Objek cout sudah banyak digunakan pada contoh-contoh didepan yang digunakan untuk menampilkan hasil ke layar.

Bentuk :

cout << *var*;

2. cin (baca C in)

cin merupakan obyek dalam C++ yang berfungsi untuk membaca data dari keyboard.

Bentuk :

cin >> *var*;

membaca data dari keyboard dan memasukkan dalam variabel bernama *var*.

Contoh program :

```
/*
      contoh penggunaan cin
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int angka;
  char huruf;

  cout<<"Masukkan angka        : ";
  cin>>angka;

  cout<<"Masukkan sebuah huruf : ";
  cin>>huruf;

  cout<<"\nAngka yang anda masukkan : "<<angka<<'\n';
  cout<<"Huruf yang anda masukkan : "<<huruf<<'\n';

  getch();
}
```

Hasil eksekusi :

```
Masukkan angka        : 125
Masukkan sebuah huruf : A

Angka yang anda masukkan : 125
Huruf yang anda masukkan : A
```

cin >> tidak dapat digunakan untuk membaca karakter Spasi atau Tab, jika akan dibaca tombol tersebut dunakan fungsi **getch()**.

3. 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 :

```
/*
      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
```

3.6 Manipulator

Manipulator digunakan untuk mengatur tampilan data pada layar. Beberapa manipular dalam C++ :

1. Manipulator **endl**

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

Contoh program :

```
/*
      contoh penggunaan endl
*/

#include<iostream.h>
#include<conio.h>
void main()
{
  int angka1 = 100;
  int angka2 = 5000;
  float total = 12323.8;


  cout<<"Isi angka 1 : "<<angka1<<endl;
  cout<<"Isi angka 2 : "<<angka2<<endl;
  cout<<"Isi angka 2 : "<<total<<endl;
  getch();
}
```

Hasil eksekusi :

```
Isi angka 1 : 100
Isi angka 2 : 5000
Isi angka 2 : 12323.8
```

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 :

```
/*
      contoh penggunaan setw()
*/

#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  int angka = 1005;

  cout<<setw(0)<<angka<<endl;
  cout<<setw(4)<<angka<<endl;
  cout<<setw(7)<<angka<<endl;
  getch();
}
```

Hasil eksekusi :

```
1005
1005
   1005
```

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 :

```
/*
      contoh penggunaan setw() dan setfill()
*/
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  int angka = 1005;

  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 eksekusi :

```
1005
1005
**1005
****1005
```

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 :

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

#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
  int x = 150;

  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 : 150
Nilai x dalam oktal        : 226
Nilai x dalam heksadesimal : 96
Nilai x dalam desimal      : 150
```

5. **Manipulor setiosflags()**

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

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

	(nol) untuk bilangan oktal
ios::showpoint	Menampilkan titik desimal pada bilangan pecahan yang tidak memiliki bagian pecahan
ios::showpos	Untuk menampilkan tanda + pada bilangan positif

Contoh program :

```
/*
      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 :

```
/*
      contoh penggunaan resetiosflags()
*/

#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
```

```
   int x = 12340;

   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
12340

Setelah dilakukan resetiosflags(ios::right)
     12340
```

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 :

```
/*
      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
```

BAB 7
PERNYATAAN DASAR DALAM C++

Blok Pernyataan

Blok pernyataan merupakan sekumpulan baris program yang berada di dalam kurung kurawal. Contoh :

```
{
   x = 10;
   x = x + 1;
}
```

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 progam :

```
/*
      contoh pendefnisian variabel dalam blok
*/

#include<iostream.h>
#include<conio.h>
void main()
{
   int nilai = 350;

   cout<<"Isi nilai : "<<nilai<<endl;
   {
      float nilai = 234.34;
      cout<<"Isi nilai : "<<nilai<<endl;
      {
         char nilai = 'A';
         cout<<"Isi nilai : "<<nilai<<endl;
      }
      cout<<"Isi nilai : "<<nilai<<endl;
   }
   cout<<"Isi nilai : "<<nilai<<endl;
   getch();
}
```

Hasil eksekusi :

```
Isi nilai : 350
Isi nilai : 234.34
Isi nilai : A
Isi nilai : 234.34
Isi nilai : 350
```

Pada contoh program diatas terdapat tiga buah variabel nilai yang didefinisikan bertipe int, float dan char. Misalkan variaber nilai yang bertipe char tidak ada maka blok tersebut akan mengenal variabel nilai yang bertipe float.

Pernyataan goto dan Pernyataan Berlabel

Pernyataan goto digunakan untuk mengarahkan eksekusi program ke pernyataan berlabel. Pernyataan berlabel adalah pernyataan yang mengandung nama label dan titik dua (:).

Bentuk pernyataan goto dan pernyataan berlabel :

goto *nama_label*;

nama_label:
pernyataan;

Contoh progam :

```
/*
      contoh program yang mengandung goto dan label
*/

#include<iostream.h>
#include<conio.h>
void main()
{
   cout<<"Teks ini diawal program"<<endl<<endl;

   goto akhir;
   cout<<"Saya ngak pernah tampil dilayar..."<<endl<<endl;

   tengah:
   cout<<"Teks ini ditengah program"<<endl;
   cout<<"Kalau Saya tampil dilayar"<<endl;
   goto selesai;

   akhir:
   cout<<"Teks ini diakhir program"<<endl;
   cout<<"Lompat ke tengah dulu ya..."<<endl<<endl;
   goto tengah;

   selesai:
   getch();
}
```

Hasil eksekusi :

```
Teks ini diawal program

Teks ini diakhir program
Lompat ke tengah dulu ya...

Teks ini ditengah program
Kalau Saya tampil dilayar
```

Sebaiknya didalam program diminimal pemakaian pernyataan **goto**, karena dapat menyebabkan program kita menjadi rumit, sehingga menyulitkan dalam logika program dan jika terjadi kesalahan penelusuran kesalahan akan menjadi rumit, apalagi kalau program yang dibuat besar.

Struktur Urut

Pada struktur ini baris program akan dikerjakan secara urut dari atas kebawah sesuai dengan penulisannya.

Algoritmik

```
Pseudocode Luas_Pesegi_panjang
// Diketahui sebuah pesegi panjang yang memiliki panjang
// dan lebar
```

```
//DEKLARASI
   int panjang, lebar, luas

//DESKRIPSI
{
   read(panjang)
   read(lebar)
   luas = panjang * lebar
   write(luas)
}
```

C++

```
/*
      contoh program menghitung luas pesegi panjang
*/

#include<iostream.h>
#include<conio.h>

void main()
{
   int panjang, lebar, luas;
   cout<<"Masukkan panjang : ";
   cin>>panjang;
   cout<<"Masukkan lebar   : ";
   cin>>lebar;
   luas = panjang * lebar;
   cout<<"Luasnya          : "<<luas<<endl;;
   getch();
}
```

Hasil eksekusi :

```
Masukkan panjang : 15
Masukkan lebar   : 10
Luasnya          : 150
```

Algoritmik

```
Pseudocode Isi_Tabung
// Diketahui sebuah tabung yang diketahui jari-jari tabung
// dan tinggi tabung

//DEKLARASI
   int jari_jari, tinggi
   float luas_lingk, isi_tabung

//DESKRIPSI
{
   read(jari_jari)
   read(tinggi)
   luas_lingk = 3.14 * jari_jari * jari_jari
   isi_tabung = luas_lingk * tinggi
   write(isi_tabung)
}
```

C++

```
/*
      contoh program menghitung isi tabung
*/

#include<iostream.h>
#include<conio.h>
void main()
{
   int jari_jari, tinggi;
   float luas_lingk, isi_tabung;

   cout<<"Masukkan jari-jari tabung : ";
   cin>>jari_jari;
   cout<<"Masukkan Tinggi tabung    : ";
   cin>>tinggi;
   luas_lingk = 3.14 * jari_jari * jari_jari;
   isi_tabung = luas_lingk * tinggi;
   cout<<"Isi tabung                : "<<isi_tabung<<endl;;
   getch();
}
```

Hasil eksekusi :

```
Masukkan jari-jari tabung : 10
Masukkan Tinggi tabung    : 5
Isi tabung                : 1570
```

Pernyataan Penyeleksian Kondisi

Pernyataan if

Pernyataan **if** digunakan untuk mengambil keputusan berdasarkan kondisi yang diuji. Bentuk **if** dalam C++ terdiri dari 3 bentuk yaitu :

1. Pernyataan if sederhana

Pada bentuk ini pernyataan **if** hanya memiliki satu kemungkingan pernyataan yang akan dikerjakan jika kondisi yang diuji bernilai benar.

Bentuk pernyataan **if** sederhana :

```
if <kondisi>
   pernyataan;
```

<*kondisi*> digunakan untuk menentukan hasil pengujian, jika <*kondisi*> bernilai benar (*true*) maka pernyataan akan dikerjakan, jika <*kondisi*> bernilai salah (*false*) maka tidak ada pernyataan yang dikerjakan.

Algoritmik

```
Pseudocode Kelulusan_Siswa
// Seorang siswa dikatakan lulus jika nilainya >= 60

//DEKLARASI
   int nilai_siswa

//DESKRIPSI
{
   read(nilai_siswa)
   if (nilai_siswa >= 60)
      write('Siswa tersebut lulus')
}
```

C++

```
/*
      contoh program yang mengandung pernyataan if sederhana
*/

#include<iostream.h>
#include<conio.h>
void main()
{
   int nilai_siswa;

   cout<<"Masukkan nilai : ";
   cin>>nilai_siswa;
   if (nilai_siswa >= 60)
      cout<<"Siswa tersebut lulus"<<endl;

   getch();
}
```

Hasil eksekusi :

```
Masukkan nilai : 80
Siswa tersebut lulus
```

2. Pernyataan if ... else

Pada bentuk ini pernyataan **if** memiliki dua kemungkinan pernyataan yang akan dikerjakan berdasarkan hasil pengujian kondisi.

Bentuk pernyataan **if ... else**

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

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

Algoritmik

```
Pseudocode Kelulusan_Siswa
// Seorang siswa dikatakan lulus jika nilainya >= 60
// jika < 60 maka siswa tidak lulus

//DEKLARASI
   int nilai_siswa

//DESKRIPSI
{
   read(nilai_siswa)
   if (nilai_siswa >= 60)
      write('Siswa tersebut lulus')
   else
      write('Siswa tersebut tidak lulus')
}
```

C++

```
/*
      contoh program yang mengandung pernyataan if...else
*/

#include<iostream.h>
#include<conio.h>
void main()
{
   int nilai_siswa;
   cout<<"Masukkan nilai : ";
   cin>>nilai_siswa;
   if (nilai_siswa >= 60)
      cout<<"Siswa tersebut lulus"<<endl;
   else
      cout<<"Siswa tersebut tidak lulus"<<endl;
   getch();
}
```

Hasil eksekusi :

```
Masukkan nilai : 85
Siswa tersebut lulus

Masukkan nilai : 50
Siswa tersebut tidak lulus
```

Algoritmik

```
Pseudocode Pembayaran_Gaji
// Akan dihitung gaji yang diterima pegawai berdasarkan jumlah hari
// kerja dan jumlah jam lembur dengan tarif tertentu

//DEKLARASI
   char nama[20]
   int jhr_kerja
   int jjam_lembur
   float upah
   float uang_lembur
   float trans_lembur
   float gaji

//DESKRIPSI
{
   read(nama)
   read(jhr_kerja)
   read(jjam_lembur)
   upah = jhr_kerja * 30000
   uang_lembur = jjam_lembur * 5000
   if (jjam_lembur >= 10)
      trans_lembur = 0.1 * uang_lembur
   else
      trans_lembur = 0
   gaji = upah + uang_lembur + trans_lembur
   write(upah)
   write(uang_lembur)
   write(trans_lembur)
   write(gaji)
}
```

C++

```
/*
      contoh program perhitungan pembayaran gaji
*/

#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
   char nama[20];
   int jhr_kerja;
   int jjam_lembur;
   float upah;
   float uang_lembur;
   float trans_lembur;
   float gaji;

   cout<<"Masukkan Nama           : ";
   cin>>nama;
   cout<<"Masukkan jml hari kerja : ";
   cin>>jhr_kerja;
   cout<<"Masukkan jml jam lembur : ";
   cin>>jjam_lembur;
   upah = jhr_kerja * 30000;
   uang_lembur = jjam_lembur * 5000;

   if (jjam_lembur >= 10)
      trans_lembur = 0.1 * uang_lembur;
   else
      trans_lembur = 0;

   gaji = upah + uang_lembur + trans_lembur;

   cout<<endl;
   cout<<"Jumlah upah        : "<<setw(10)<<upah<<endl;
   cout<<"Jumlah uang lembur : "<<setw(10)<<uang_lembur<<endl;
   cout<<"Transfort lembur   : "<<setw(10)<<trans_lembur<<endl;
   cout<<"                     -------------- + "<<endl;
   cout<<"Gaji yang diterima : "<<setw(10)<<gaji<<endl;
   getch();
}
```

Hasil eksekusi :

```
Masukkan Nama           : Joko
Masukkan jml hari kerja : 23
Masukkan jml jam lembur : 15

Jumlah upah        :     690000
Jumlah uang lembur :      75000
Transfort lembur   :       7500
                     -------------- +
Gaji yang diterima :     772500
```

3. Pernyataan nested if (if bersarang)

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

Bentuk pernyataan **if ... else**

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

<*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.

Algoritmik

```
Pseudocode Konfersi_Nilai
// Diketahui nilai angka seorang siswa yang akan
// dikonfersikan ke nilai huruf


//DEKLARASI
   char nama[20]
   int nilai_angka
   char huruf

//DESKRIPSI
{
   read(nama)
   read(nilai_angka)
   if (nilai_angka >= 80)
      huruf = 'A'
   else if (nilai_angka >= 70)
           huruf = 'B'
        else if (nilai_angka >= 60)
                huruf = 'C'
             else if (nilai_angka >= 50)
                     huruf = 'D'
                  else
                     huruf = 'E'
   write(nama)
   write(huruf)
}
```

C++

```
/*
      contoh program yang mengandung pernyataan nested if
*/

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

```
void main()
{
   char nama[20];
   int nilai_angka;
   char huruf;
   cout<<"Masukkan nama        : ";
   cin>>nama;
   cout<<"Masukkan nilai angka : ";
   cin>>nilai_angka;
   if (nilai_angka >= 80)
      huruf = 'A';
   else if (nilai_angka >= 70)
           huruf = 'B';
        else if (nilai_angka >= 70)
                huruf = 'C';
             else if (nilai_angka >= 70)
                     huruf = 'D';
                  else
                     huruf = 'E';
   cout<<endl;
   cout<<"Nama     : "<<nama<<endl;
   cout<<"Nilainya : "<<huruf<<endl;
   getch();
}
```

Hasil eksekusi :

```
Masukkan nama        : Rahmad
Masukkan nilai angka : 75

Nama     : Rahmad
Nilainya : B
```

4. Pernyataan switch

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;
}
```

(ungkapan)dalam pernyataan **switch** 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**.

Algoritmik

```
Pseudocode Menu_Pilihan
// Ditampilkan menu pilihan untuk mengakses perhitungan yang
// akan dilakukan dengan menggunakan switch()


//DEKLARASI
   char pilih
   int rusuk, isi_kubus, alas, tinggi
   float luas_segi3

//DESKRIPSI
{
   write(“Menu Pilihan”)
   write("============")
   write("1. Isi Kubus")
   write("2. Luas Segitiga")
   write("3. Selesai")
   write("Pilihan anda [1..3] : ")
   read(pilih)

   switch(pilih)
   case 1 :
            read(rusuk)
            isi_kubus = rusuk * rusuk * rusuk
            write(isi_kubus)

   case 2 :
            read(alas)
            read(tinggi)
            luas_segi3 = 0.5 * alas * tinggi
            write(luas_segi3)

   case 3 : break

}
```

C++

```
/*
    contoh penggunaan pernyataan switch
*/

#include <iostream.h>
#include <conio.h>
void main()
{
   char pilih;
   int rusuk, isi_kubus, alas, tinggi;
   float luas_segi3;
   clrscr();
   cout<<"Menu Pilihan"<<endl;
   cout<<"============"<<endl;
   cout<<"1. Isi Kubus"<<endl;
   cout<<"2. Luas Segitiga"<<endl;
   cout<<"3. Selesai"<<endl;
   cout<<"Pilihan anda [1..3] : ";
   cin>>pilih;

   switch(pilih)
   {
    case '1' :
```

```
         {
            cout<<"\nPanjang rusuk : ";
            cin>>rusuk;
            isi_kubus = rusuk * rusuk * rusuk;
            cout<<"Isi kubus : "<<isi_kubus<<endl;
            break;
         }
    case '2' :
         {
            cout<<"\nPanjang alas segitiga : ";
            cin>>alas;
            cout<<"Tinggi segitiga       : ";
            cin>>tinggi;
            luas_segi3 = 0.5 * alas * tinggi;
            cout<<"Luas segitiga         : "<<luas_segi3<<endl;
            break;
         }
    case '3' : break;
   }
   getch();
}
```

Hasil eksekusi :

```
Menu Pilihan
============
1. Isi Kubus
2. Luas Segitiga
3. Selesai
Pilihan anda [1..3] : 2

Panjang alas segitiga : 10
Tinggi segitiga       : 15
Luas segitiga         : 75
```

Pernyataan Pengulangan

1. Pernyataan for

Pernyataan **for** berfungsi untuk mengulang satu atau beberapa pernyataan sebanyak syarat yang diberikan.

Bentuk pernyataan **for** :

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

Keterangan :

- *ungkakapan1* merupakan pernyataan inisialisasi awal dari perulangan **for**
- *ungkakapan2* merupakan kondisi yang menentukan pengulangan terhadap *pernyataan* atau tidak
- *ungkakapan3* digunakan sebagai pengatur variabel yang digunakan dalam *ungkakapan1*

Algoritmik

```
Pseudocode Cetak_Angka_For
// Akan dicetak angka 1 sampai 10 dengan perulangan for


//DEKLARASI
   int i
```

```
//DESKRIPSI
{
   for(i=1; i<=10; i++)
   {
      write(i)
   }
}
```

C++

```
/*
   Contoh program perulangan for
*/

#include <iostream.h>
#include <conio.h>

void main()
{
   for (int i=1; i<=10; i++)
   {
      cout<<"Isi i = "<<i<<endl;
   }
   getch();
}
```

Hasil eksekusi :

```
Isi i = 1
Isi i = 2
Isi i = 3
Isi i = 4
Isi i = 5
Isi i = 6
Isi i = 7
Isi i = 8
Isi i = 9
Isi i = 10
```

Algoritmik

```
Pseudocode Cetak_bilangan_genap_for
// Akan dicetak bilangan genap dengan batas akhir diinputkan


//DEKLARASI
   int batas_akhir

//DESKRIPSI
{
   read(batas_akhir)
   for(i=1; i<=batas_akhir; i++)
   {
      if( i%2 = 0)
         write(i)
   }
}
```

C++

```
/*
```

```
   Mencetak bilangan genap perulangan for
*/

#include <iostream.h>
#include <conio.h>

void main()
{
   int batas_akhir;
   cout<<"Jumlah perulangan : ";
   cin>>batas_akhir;
   for (int i=1; i<=batas_akhir; i++)
   {
      if ( i % 2 == 0)
         cout<<i<<"  ";
   }
   getch();
}
```

Hasil eksekusi :

```
Jumlah perulangan : 30
2  4  6  8  10  12  14  16  18  20  22  24  26  28  30
```

for bersarang (Nested for)

for bersarang merupakan pernyataan for yang berada dalam pernyataan **for** (**for** dalam **for**). Sebagai contoh misalkan akan dibentuk segitiga yang tersusun dari karakter * dengan **for** bersarang.

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

Algoritmik

```
Pseudocode Segitiga_Bintang
// Akan dicetak karakter * yang membentuk bangun segitiga


//DEKLARASI
   int tinggi

//DESKRIPSI
{
   read(tinggi)
   for(i=1;i<=tinggi;i++)
      for(j=1;j<=i;j++)
          write('*')
      endl
}
```

C++

```
/*
   Contoh program for bersarang
```

```
*/

#include <iostream.h>
#include <conio.h>

void main()
{
   int tinggi;
   cout<<"Tinggi segi3 : ";
   cin>>tinggi;
   for(int i = 1; i <= tinggi; i++)
   {
      for (int j = 1; j <= i; j++)
           cout<<'*';
      cout<<endl;
   }
   getch();
}
```

Hasil eksekusi :

```
Tinggi segi3 : 7
*
**
***
****
*****
******
*******
```

2. 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;
```

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).

Algoritmik

```
Pseudocode Cetak_Angka_While
// Akan dicetak angka 1 sampai 10 dengan perulangan while

//DEKLARASI
   int i

//DESKRIPSI
{
   i = 1
   while(i<=10)
   {
      write(i)
      i++
   }
```

```
}
```

C++

```
/*
   Contoh program perulangan while
*/

#include <iostream.h>
#include <conio.h>

void main()
{
   int i;
   i = 1;
   while ( i <= 10)
   {
      cout<<"Isi i = "<<i<<endl;
      i++;
   }
   getch();
}
```

Hasil eksekusi :

```
Isi i = 1
Isi i = 2
Isi i = 3
Isi i = 4
Isi i = 5
Isi i = 6
Isi i = 7
Isi i = 8
Isi i = 9
Isi i = 10
```

Algoritmik

```
Pseudocode Cetak_bilangan_genap_while
// Akan dicetak bilangan genap dengan batas akhir diinputkan

//DEKLARASI
   int batas_akhir

//DESKRIPSI
{
   read(batas_akhir)
   i = 1
   for(i; i<=10; i++)
   {
      if( i%2 = 0)
         write(i)
   }
}
```

C++

```
/*
   Mencetak bilangan genap perulangan while
```

```
*/

#include <iostream.h>
#include <conio.h>

void main()
{
   int batas_akhir;
   int i = 1;
   cout<<"Jumlah perulangan : ";
   cin>>batas_akhir;
   while (i<=batas_akhir)
   {
      if ( i % 2 == 0)
         cout<<i<<"  ";
      i++;
   }
   getch();
}
```

Hasil eksekusi :

```
Jumlah perulangan : 30
2  4  6  8  10  12  14  16  18  20  22  24  26  28  30
```

3. 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);
```

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).

Algoritmik

```
Pseudocode Cetak_Angka_do_While
// Akan dicetak angka 1 sampai 10 dengan perulangan do…while

//DEKLARASI
   int i

//DESKRIPSI
{
   i = 1
   do
   {
      write(i)
      i++
   } while(i<=10)
}
```

C++

```
/*
```

```
   Contoh program perulangan do...while
*/

#include <iostream.h>
#include <conio.h>

void main()
{
   int i;
   i = 1;
   do
   {
      cout<<"Isi i = "<<i<<endl;
      i++;
   }while ( i <= 10);
   getch();
}
```

Hasil eksekusi :

```
Isi i = 1
Isi i = 2
Isi i = 3
Isi i = 4
Isi i = 5
Isi i = 6
Isi i = 7
Isi i = 8
Isi i = 9
Isi i = 10
```

Algoritmik

```
Pseudocode Cetak_bilangan_genap_do_while
// Akan dicetak bilangan genap dengan batas akhir diinputkan

//DEKLARASI
   int batas_akhir

//DESKRIPSI
{
   read(batas_akhir)
   i = 1
   do
   {
      if( i%2 = 0)
         write(i)
   } while ( i <= batas_akhir)
}
```

C++

```
/*
   Mencetak bilangan genap perulangan do...while
*/

#include <iostream.h>
#include <conio.h>

void main()
```

```
{
   int batas_akhir;
   int i = 1;
   cout<<"Jumlah perulangan : ";
   cin>>batas_akhir;
   do
   {
      if ( i % 2 == 0)
         cout<<i<<"  ";
      i++;
   }while (i<=batas_akhir);
   getch();
}
```

Hasil eksekusi :

```
Jumlah perulangan : 30
2  4  6  8  10  12  14  16  18  20  22  24  26  28  30
```

Latihan Pengulangan :

1. Buatlah pseudocode dan program untuk mencari rata-rata dari sejumlah angka yang diinputkan dengan menggunakan pengulangan.
2. Buatlah pseudocode dan program untuk mencari angka terbesar dan angka terkecil dari sejumlah angka yang diinputkan dengan menggunakan pengulangan.
3. Buatlah pseudocode dan program untuk mencetak angka 1 sampai 1000 dengan menggunakan perulangan. Angka dicetak per 10 kolom dan 10 baris, pada baris pertama jika angka sudah dicetak 10 kolom maka dilakukan pergantian baris, jika sudah 10 baris maka ditampilkan pesan “Tekan enter untuk melanjutkan”. Begitu seterusnya sampai angka ke 1000.

```
    1    2    3    4    5    6    7    8    9   10
   11   12   13   14   15   16   17   18   19   20
   21   22   23   24   25   26   27   28   29   30
   31   32   33   34   35   36   37   38   39   40
   41   42   43   44   45   46   47   48   49   50
   51   52   53   54   55   56   57   58   59   60
   61   62   63   64   65   66   67   68   69   70
   71   72   73   74   75   76   77   78   79   80
   81   82   83   84   85   86   87   88   89   90
   91   92   93   94   95   96   97   98   99  100
Tekan enter untuk melanjutkan...
```

4. Buatlah pseudocode dan program untuk menghitung total pembayaran pembelian. Input berupa nama pembeli, dan data barang dengan jumlah barang yang dibeli berupa input nama barang, jumlah dan harga barang. Barang yang dibeli jumlahnya bisa banyak terganng pembelian dari konsumen.

   Output:

no	Nama barang	Jml beli	Hg satuan	Total hg

DAFTAR PUSTAKA

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

Antony Pranata, *Algortima dan Pemrograman* J&J Learning, Yogyakarta

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

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

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