

MODUL

PEMROGRAMAN TERSTRUKTUR

MENGGUNAKAN DEV-C++

- NUR ALAMSYAH, M. KOM
- M.EDYA ROSADI, M.KOM

FAKULTAS TEKNOLOGI INFORMASI



DAFTAR ISI

| | Halaman |
|--|----------------|
| DAFTAR ISI..... | ii |
| DAFTAR GAMBAR..... | v |
| DAFTAR TABEL | vi |
| BAB I ALGORITMA DAN PEMROGRAMAN TERSTRUKTUR..... | 1 |
| 1.1 Konsep Algoritma..... | 1 |
| 1.2 Konsep Dasar Pemrograman Terstruktur..... | 4 |
| 1.3 Tahapan membuat Program | 4 |
| 1.4 Struktur Dasar Algoritma dalam Pemrograman Terstruktur | 5 |
| 1.4.1 Flowchart | 6 |
| 1.4.2 Pseudocode | 28 |
| 1.5 Mengenal Bahasa Pemrograman..... | 30 |
| 1.5.1 Istilah – istilah Dasar dalam bahasa Pemrograman | 31 |
| 1.5.2 Klasifikasi Bahasa Pemrograman | 31 |
| BAB II PENGENALAN C++..... | 35 |
| 2.1 Apa itu C++ ?..... | 35 |
| 2.2 Sejarah Singkat Lahirnya C++..... | 35 |
| 2.3 Hubungan antara C dan C++ | 36 |
| 2.4 C++ Standar : C++98, C++03, dan C++11 (C++0x)..... | 37 |
| 2.5 Proses Pembentukan Program dalam C++..... | 37 |
| 2.6 Kerangka Kode Program dalam C++..... | 39 |
| 2.7 Program yang ditulis dalam bahasa C..... | 41 |
| 2.8 Pengertian dan kegunaan File Header (File.h)..... | 41 |
| 2.9 Software yang dibutuhkan | 42 |
| 2.10 Instalasi dan Konfigurasi Software Dev-C++..... | 43 |
| 2.11 Membuat Program “I Love C++” | 47 |

| | |
|--|-----------|
| 2.12 Kompilasi dan Eksekusi Program | 48 |
| BAB III KOMENTAR, IDENTIFIER, DAN TIPE DATA..... | 50 |
| 3.1 Komentar Program..... | 50 |
| 3.2 Jenis Identifier..... | 51 |
| 3.2.1 Konstanta | 53 |
| 3.2.2 Variabel..... | 55 |
| 3.3 Tipe Data..... | 58 |
| 3.3.1 Tipe Data Dasar | 58 |
| 3.3.2 Tipe Data Bentuk | 61 |
| BAB IV OPERATOR | 65 |
| 4.1 Jenis-jenis Operator dalam bahasa C++..... | 65 |
| 4.2 Operator Assignment | 65 |
| 4.3 Operator Unary | 67 |
| 4.4 Operator Binary | 70 |
| 4.5 Operator Ternary..... | 76 |
| BAB V PERCABANGAN | 78 |
| 5.1 Struktur <code>if</code> Satu Kondisi..... | 78 |
| 5.2 Struktur <code>if</code> Dua Kondisi..... | 80 |
| 5.3 Struktur <code>if</code> Tiga Kondisi atau Lebih..... | 82 |
| 5.4 Percabangan Menggunakan Statemen <code>switch</code> | 84 |
| BAB VI PENGULANGAN..... | 86 |
| 6.1 Struktur <code>for</code> | 86 |
| 6.2 Struktur <code>while</code> | 90 |
| 6.3 Struktur <code>do-while</code> | 92 |
| BAB VII ARRAY | 94 |
| 7.1 Apa itu Array | 94 |
| 7.2 Mengisikan Nilai ke dalam Elemen Array..... | 94 |

| | |
|--|------------|
| 7.3 Menampilkan Nilai yang terdapat pada Array | 95 |
| 7.4 Melakukan Inisialisasi Array | 96 |
| 7.5 Melakukan Pencarian pada Elemen Array | 97 |
| 7.6 Mengurutkan Elemen Array | 98 |
| 7.7 Array Multidimensi..... | 101 |
| DAFTAR PUSTAKA..... | 103 |

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 1. 1 Simbol flowcahrt..... | 6 |
| Gambar 1. 2 Algoritma dan Pemrograman | 30 |
| Gambar 1. 3 Proses Kerja Interpreter | 32 |
| Gambar 1. 4 Proses kerja Compiler | 33 |
| Gambar 2. 1 Pengelompokan bahasa Pemrograman..... | 36 |
| Gambar 2. 2 Proses pembentukan Program | 38 |
| Gambar 2. 3 Proses Pembentukan Program..... | 42 |

DAFTAR TABEL

| | |
|---|----|
| Tabel 1. 1 Perbedaan Interpreter dan Compiler | 34 |
|---|----|

BAB I

ALGORITMA DAN PEMROGRAMAN TERSTRUKTUR

1.1 Konsep Algoritma

Algoritma merupakan pondasi yang harus dikuasai oleh setiap mahasiswa yang ingin menyelesaikan suatu masalah secara berstruktur, efektif dan efisien.

Definisi Algoritma :

1. Teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas.
2. Suatu prosedur yang jelas untuk menyelesaikan suatu persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya.

Catatan Sejarah

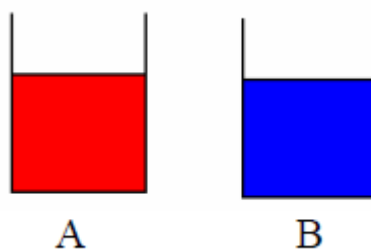
Abu Ja'far Muhammad Ibnu Musa Al-Kwarizmi, penulis buku “Aljabar wal muqabala” beberapa abad yang lalu (pada abad IX), dianggap sebagai pencetus pertama Algoritma karena didalam bukunya menjelaskan langkah-langkah dalam menyelesaikan berbagai persoalan aritmatika (aljabar), kemungkinan besar kata “Algoritma” diambil dari kata “Al-Kwarizmi” yang kemudian berubah menjadi “Algorism”, selanjutnya menjadi “Algorithm”.

Contoh Algoritma:

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 menukarkan 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

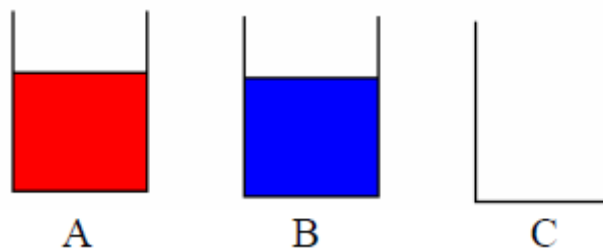


Algoritma diatas tidak menghasilkan pertukaran yang benar, langkah-langkahnya tidak logis, karena yang terjadi bukan pertukaran tetapi percampuran antara air di gelas A dengan air di gelas B. Sehingga algoritma Tukar_Isi_Gelas diatas salah. Dari permasalahan diatas algoritma yang benar adalah bahwa untuk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

Algoritma Tukar_Isi_Gelas

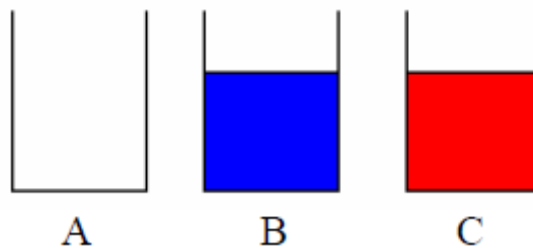
1. Tuangkan air dari gelas A ke gelas C
2. Tuangkan air dari gelas B ke gelas A
3. Tuangkan air dari gelas C ke gelas B

Keadaan awal sebelum pertukaran

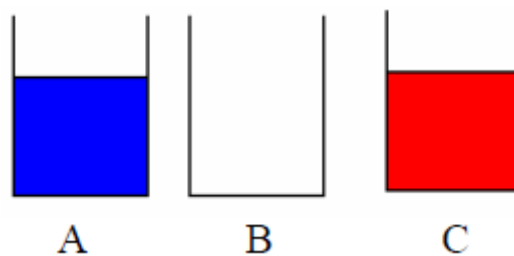


Proses pertukaran :

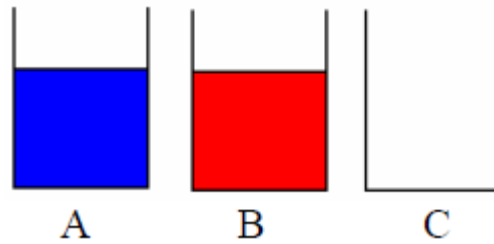
1. Tuangkan air dari gelas A ke gelas C



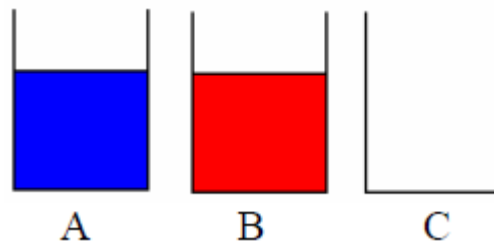
2. Tuangkan air dari gelas B ke gelas A



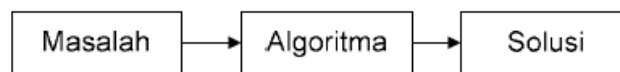
3. Tuangkan air dari gelas C ke gelas B



Hasil 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 :

1. Algoritma memiliki logika perhitungan atau metode yang tepat dalam menyelesaikan masalah.
2. Menghasilkan output yang tepat dan benar dalam waktu yang singkat.
3. Algoritma ditulis dengan bahasa yang standar secara sistematis dan rapi sehingga tidak menimbulkan arti ganda (ambiguous).

4. Algoritma ditulis dengan format yang mudah dipahami dan mudah diimplementasikan ke dalam bahasa pemrograman.
5. Semua operasi yang dibutuhkan terdefinisi dengan jelas.
6. Semua proses dalam algoritma harus berakhir setelah sejumlah langkah dilakukan.

1.2 Konsep Dasar Pemrograman Terstruktur

Program adalah deretan instruksi yang digunakan untuk mengendalikan komputer, sehingga komputer dapat melakukan tindakan sesuai dengan yang dikehendaki pembuatnya. Sebuah program bisa dikatakan baik jika algoritmanya jelas terstruktur dan mudah dibaca oleh orang lain. Sedangkan **Algoritma** yaitu langkah-langkah untuk menyelesaikan sesuatu masalah. Adapun pengertian dari **Pemrograman Terstruktur** yaitu metode untuk mengorganisasikan dan membuat kode-kode program supaya mudah untuk dimengerti, mudah di test dan di modifikasi. Dimana programnya dapat dipergunakan oleh pengguna secara mudah dan dapat dimengerti tentang proses yang sedang dilakukan oleh program tersebut. Serta dapat mengatur kebutuhan akan piranti masukan dan keluaran.

Ciri-ciri teknik pemrograman terstruktur :

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

1.3 Tahapan membuat Program

Pemrograman adalah sebuah rangkaian instruksi-instruksi dalam bahasa komputer yang disusun secara logis dan sistematis. Proses pemrograman komputer bertujuan untuk memecahkan suatu masalah dan membuat mudah pekerjaan dari user atau pengguna komputer. Adapun tahapan pemrograman yang kompleks sebagai berikut :

1. Definisi Masalah

Menentukan model /rancangan apa yang akan dibuat untuk penyelesaian masalah.

2. Analisa Kebutuhan

Menentukan data untuk masukan dan keluaran yang diminta, bahasa pemrograman yang digunakan serta tipe komputer apa yang dibutuhkan.

3. Pembuatan Algoritma/Desain algoritma

Membuat susunan langkah-langkah / intruksi penyelesaian masalah

Hal yang dapat dilakukan dengan 2 cara:

- a. Menggunakan Flowchart
- b. Menggunakan bahasa semu (pseudocode)

4. Pemrograman (dengan bahasa Pemrograman)

Pembuatan program dengan menggunakan bahasa pemrograman.

5. Pengujian Program

Dapat dilakukan melalui 2 tahap :

- a. Pengujian Tahap Debuging

Untuk mengecek kesalahan program, Baik sintaksis maupun logika.

- b. Pengujian tahap profiling.

Untuk menentukan waktu tempuh dan banyak nya memori program yang digunakan. Setelah program bebas dari kesalahan sehingga dapat dilakukan proses excute program.

6. Dokumentasi yang digunakan untuk file backup

7. Pemeliharaan

Upaya yang dilakukan dengan Menghindari kerusakan atau hilangnya suatu program baik hardware maupun Human Error.

1.4 Struktur Dasar Algoritma dalam Pemrograman Terstruktur

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

Sebelum melangkah jauh ke pembahasan sequensial, seleksi dan perulangan, alangkah baiknya kita mengenal dulu tentang Flowchart dan Pseudocode, Mengapa ? Karena Flowchart akan selalu dibutuhkan seorang programmer jika ingin membuat sebuah listing program yang lebih kompleks, karena flowchart berfungsi sebagai gambaran algoritma (urutan langkah-langkah

logis penyelesaian suatu masalah program) yang akan digunakan. Sedangkan pseudocode (bahasa semu) merupakan kode yang mirip dengan pemograman sebenarnya. Pseudocode berasal dari kata Pseudo yang berarti imitasi, mirip, atau menyerupai dengan kode bahasa pemograman.

1.4.1 Flowchart

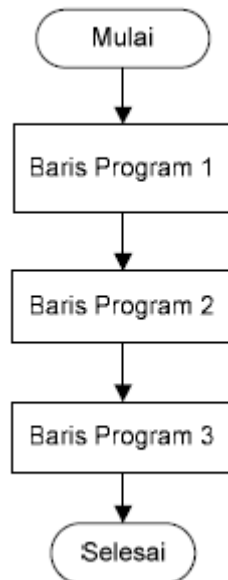
Flowchart adalah sebuah gambar atau bagan yang terbentuk dari simbol dan garis penghubung, yang menunjukkan urutan langkah yang benar, dan hubungan antara proses beserta instruksinya. Jadi, simbol melambangkan proses, sedangkan bubungan antar proses-proses digambarkan dengan garis penghubung. Agar tidak semakin bingung perhatikan beberapa contoh simbol flowchart beserta penggunaanya di bawah ini :

| Simbol | Keterangan |
|---|---|
|  | Terminator. Tanda mulai atau selesai |
|  | Input/output. Tanda masukan atau keluaran |
|  | Proses. Tanda proses komputasi |
|  | Keputusan. Tanda pengambilan keputusan/validasi |
|  | Konektor. Tanda penghubung flowchart bila pindah halaman |
|  | Proses terdefinisi. Tanda prosedur atau fungsi (subalgoritma) |

Gambar 1. 1 Simbol flowcahrt

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.



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 :

a. Algoritma Luas Persegi Panjang

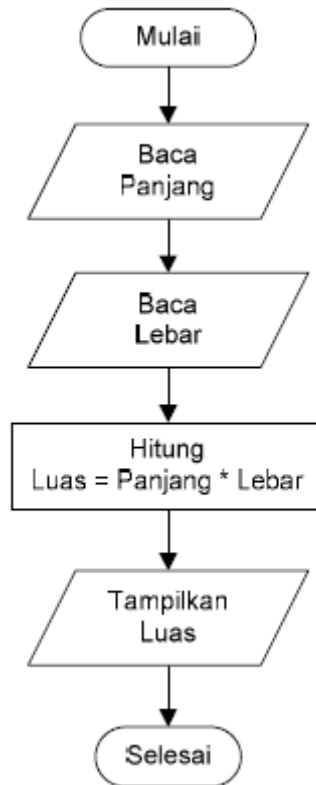
Akan dihitung luas persegi panjang yang diketahui panjang dan lebarnya, maka algoritmanya sebagai berikut :

Diketahui sebuah persegi 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 :



b. Algoritma Isi Tabung

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

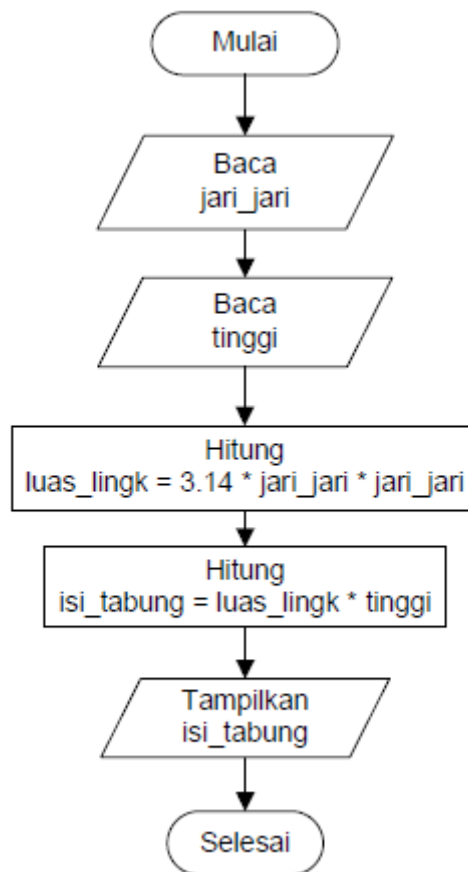
Tabung 1

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 :



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.

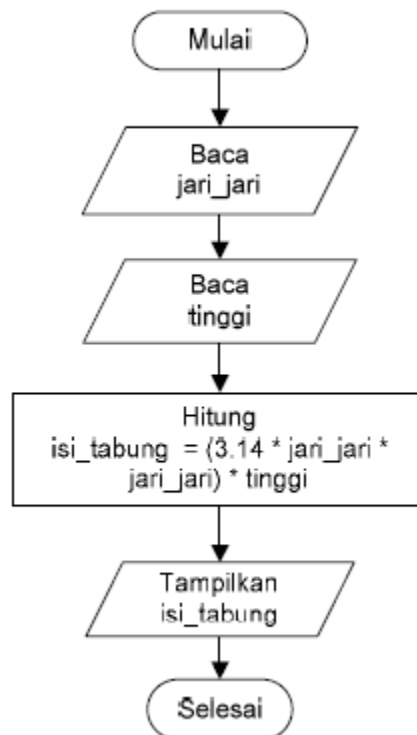
Tabung 2

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 :



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 algoritma 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 program yang dikerjakan hanya yang memenuhi syarat saja. Struktur pemilihan adalah struktur program yang melakukan proses pengujian untuk mengambil suatu keputusan apakah suatu baris atau blok instruksi akan diproses atau tidak. Pengujian kondisi 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.

Contoh :

- $7 = 7$ (Benilai benar, sebab 7 sama dengan 7)
- $5 = 9$ (Bernilai salah, sebab 5 tidak sama dengan 9)
- $4 > 2$ (Bernilai benar, sebab 4 lebih besar dari pada 2)
- $3 <> 8$ (Bernilai benar, sebab 3 tidak sama dengan 8)
- $X = 10$ (Dapat benilai benar atau salah, tergantung isi variabel X)
- $(X > 3) \text{ And } (Y < 12)$ (Dapat benilai benar atau salah, tergantung isi variabel X dan Y) Struktur pemilihan dalam penulisan program diimplementasikan dengan instruksi **IF**.

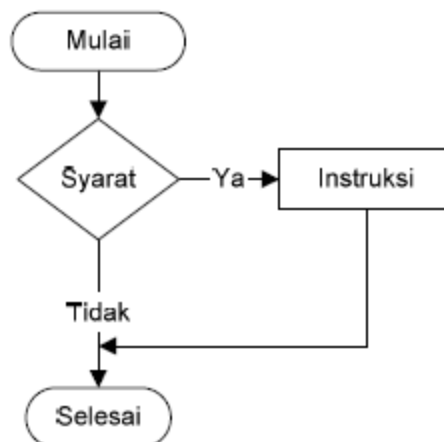
Macam-macam struktuf IF :

IF SEDERHANA

Bentuk IF sederhana adalah :

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

Bentuk flowchart :



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 :

a. Algoritma Kelulusan Siswa 1

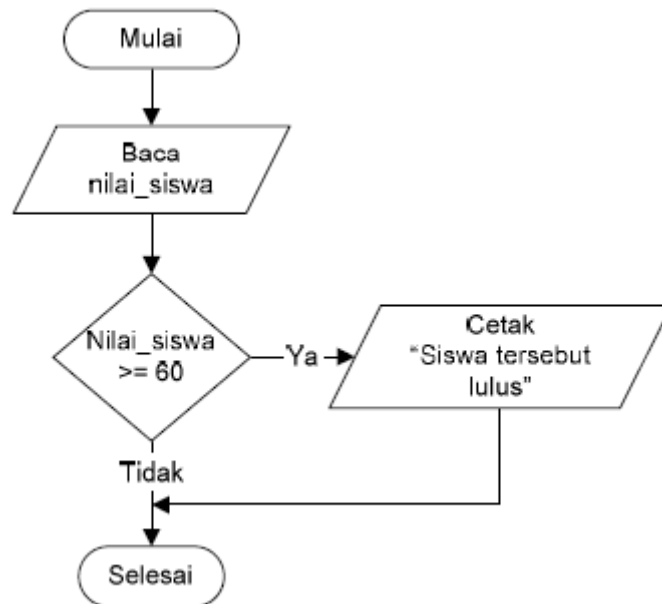
Diketahui seorang siswa dikatakan lulus jika nilainya ≥ 60 .

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_Siswa1 :



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.

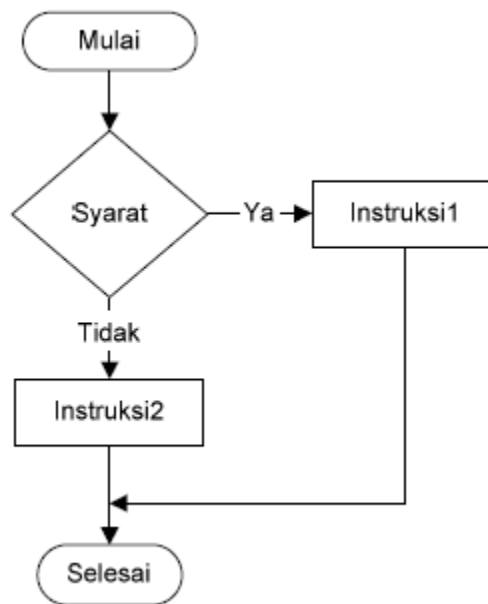
IF ... THEN ... ELSE ...

Bentuk umum :

```
IF < syarat > THEN
< instruksi1 >
ELSE
< instruksi2 >
```

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

Bentuk flowchart :



b. Algoritma Kelulusan Siswa 2

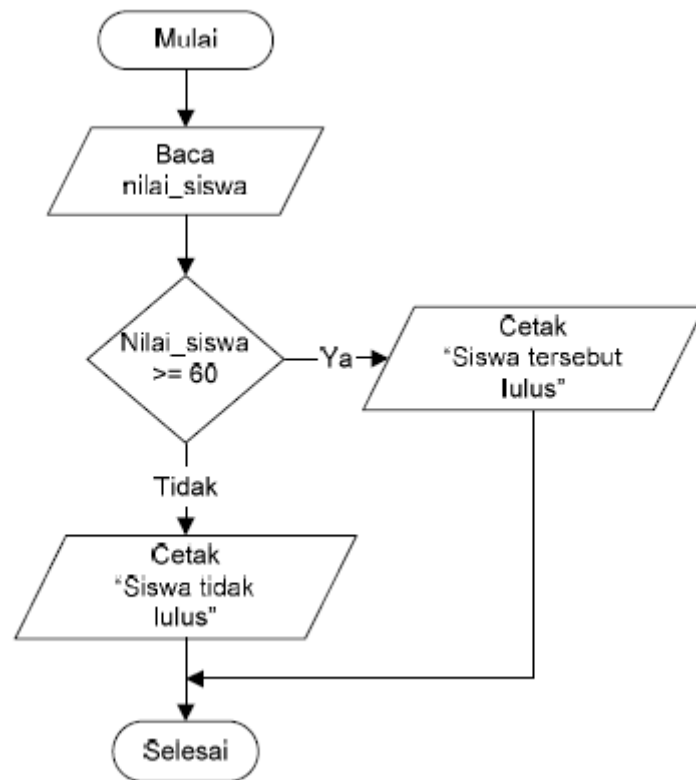
Dibuat suatu aturan kelulusan seorang siswa yang diketahui dari hasil nilainya dalam bentuk angka. Seorang siswa dikatakan lulus jika nilai lebih besar atau sama dengan 60, dan jika nilainya lebih kecil dari 60 maka siswa tidak lulus.

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_Siswa2 :



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 :

c. Algoritma Pembayaran Gaji

Buatlah algoritma dan flowchart untuk menghitung jumlah pembayaran 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.

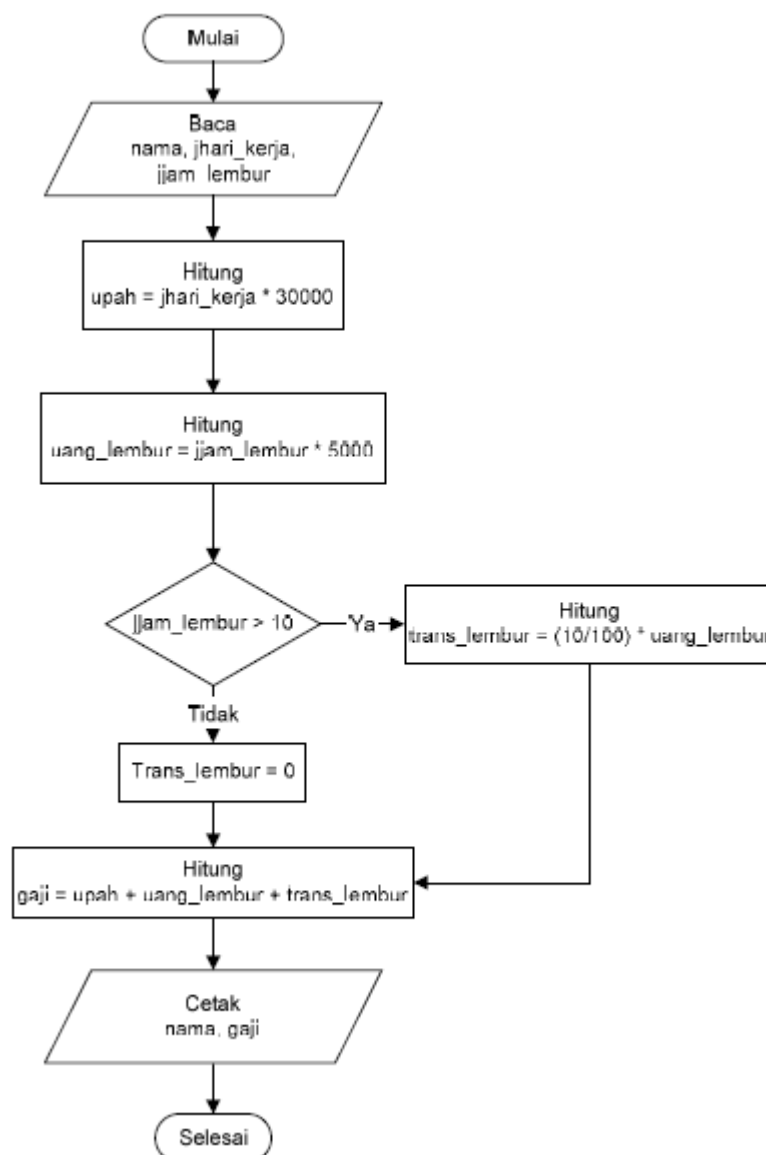
Diketahui input data nama, jumlah hari kerja dan jumlah jam lembur, tarif per hari kerja Rp. 30.000, tarif per jam lembur Rp. 5.000, jika jumlah jam lembur lebih dari 10 jam maka akan mendapatkan tambahan uang transport lembur 10% dari jumlah uang lembur.

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_lembur > 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 :



Dari flowchart diatas dapat dijelaskan bahwa setelah nama, jhari_kerja, jjam_lembur dimasukkan maka akan dihitung besarnya upah, kemudian dihitung

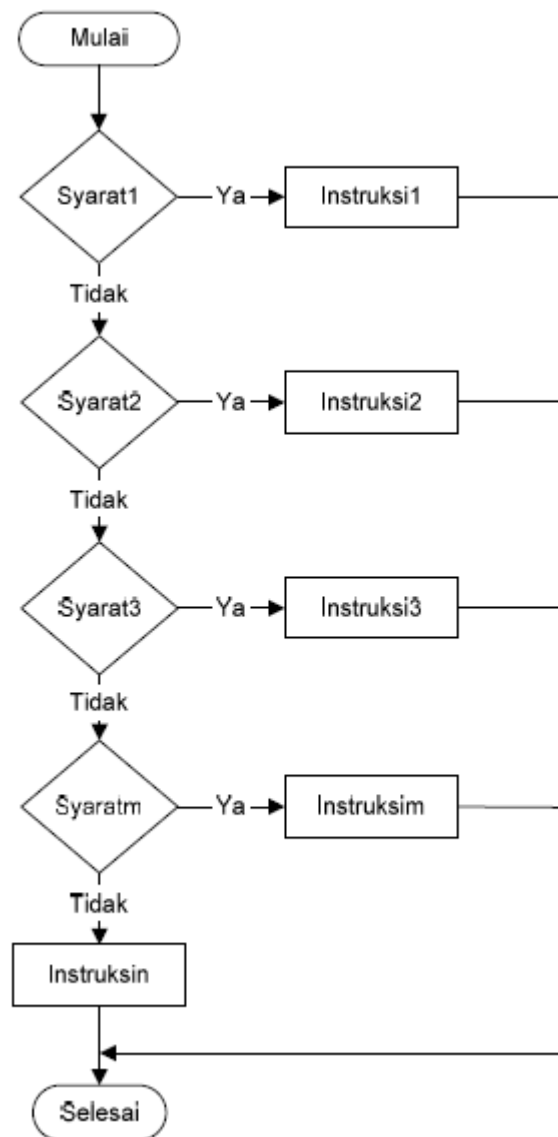
besarnya uang_lembur, kemudian diuji apakah jjam_lembur > 10, jika benar maka dihitung trans_lembur 10% dari uang_lembur, jika salah maka trans_lembur = 0, kemudian dihitung besar gaji yang diperoleh. Terakhir dicetak berupa nama dan gaji, kemudian selesai.

IF BERSARANG (NESTED IF)

Bentuk umum :

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

Bentuk flowchart :



Pada bentuk ini terdapat banyak kemungkinan pilihan yang akan dikerjakan berdasarkan hasil pengujian, proses pengujiannya adalah :

jika syarat1 yang diuji bernilai 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 :

d. Algoritma Konversi Nilai

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 :

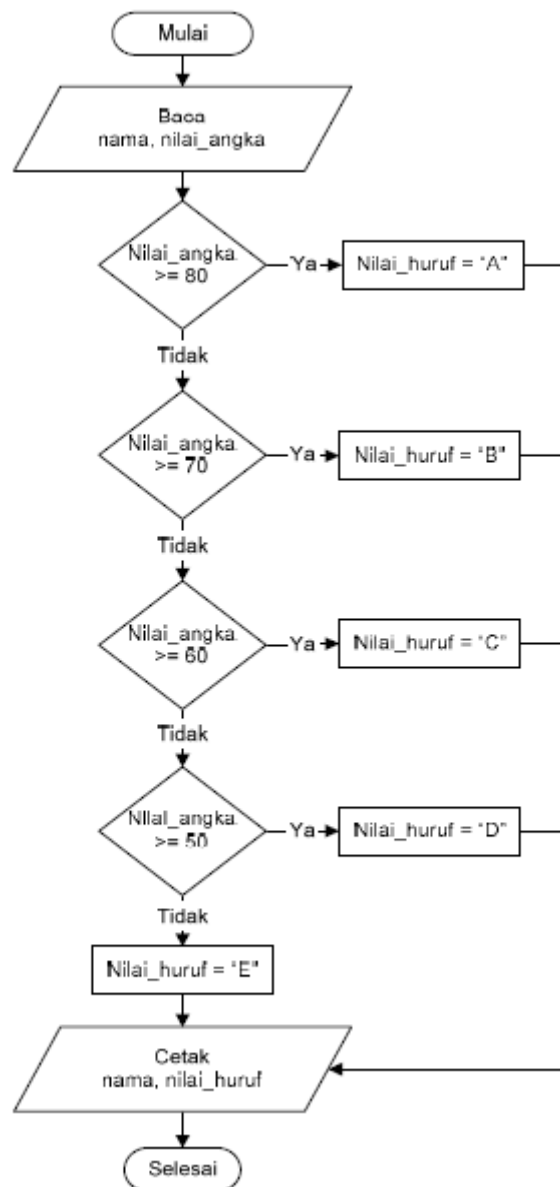
- Jika nilai_angka ≥ 80 maka nilai huruf sama dengan A
- Jika nilai_angka ≥ 70 maka nilai huruf sama dengan B
- Jika nilai_angka ≥ 60 maka nilai huruf sama dengan C
- Jika nilai_angka ≥ 50 maka nilai huruf sama dengan D
- Jika nilai_angka < 50 maka nilai huruf sama dengan E

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 :



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

3. STRUKTUR PENGULANGAN (REPETITION)

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

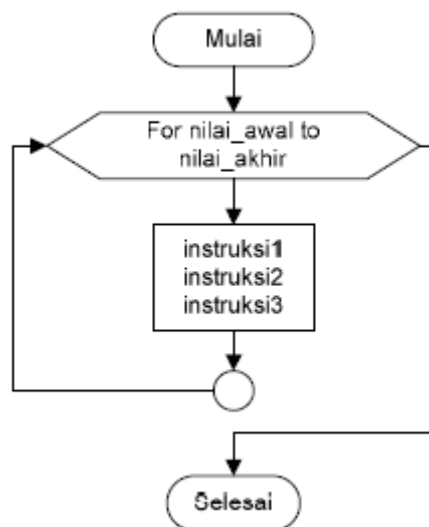
yang diberikan.

Struktur pengulangan mempunyai beberapa bentuk :

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** :



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:

a. Algoritma Cetak Angka FOR

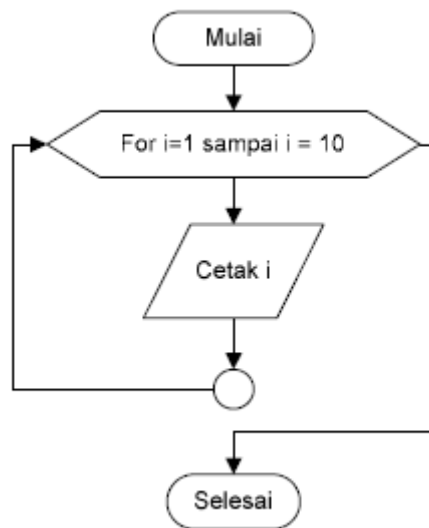
Akan dicetak angka 1 sampai 10 dengan menggunakan perulangan `for`

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 :



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 :

b. Algoritma Cetak Bilangan Genap FOR

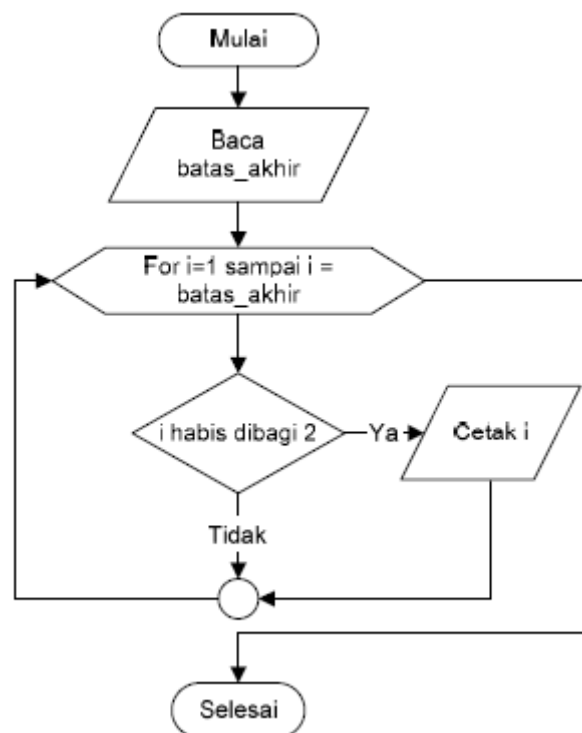
Akan dicetak bilangan genap mulai dari 0 dengan batas akhir diinputkan dari keyboard dengan menggunakan pengulangan for.

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 :

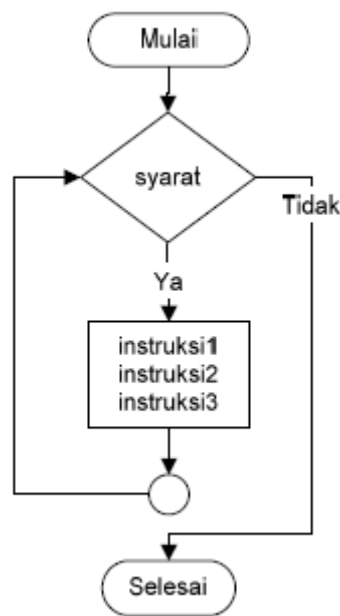


Dari gambar flowchart diatas dapat dijelaskan pertama kali dibaca `batas_akhir` perulangan, kemudian nilai `i` pertama kali akan berisi 1, kemudian akan diuji apakah nilai `i` habis dibagi dua, jika benar maka dicetak nilai `i`, kemudian pengulangan dilanjutkan dengan nilai `i` menjadi 2, jika tidak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

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** :



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 :

c. Algoritma Cetak Angka WHILE

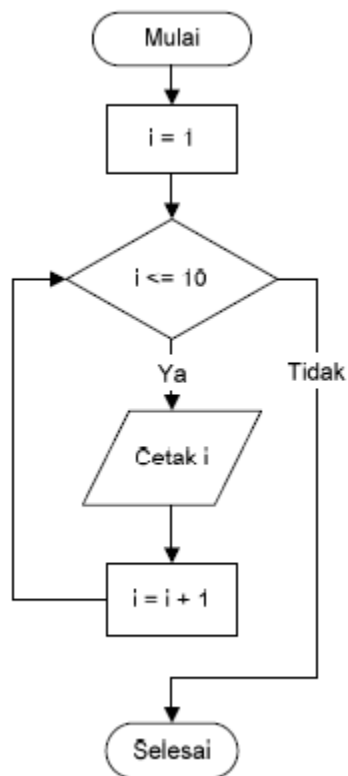
Akan dibuat contoh diatas dengan menggunakan while.

Dicetak angka 1 sampai 10 dengan perulangan while.

Deskripsi :

1. mulai
2. $i = 1$
3. selama $i \leq 10$ kerjakan langkah 4 sampai langkah 5
4. cetak i
5. $i = i + 1$
6. selesai

Flowchart **Cetak_Angka** :



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.

d. Algoritma Cetak Bilangan Genap WHILE

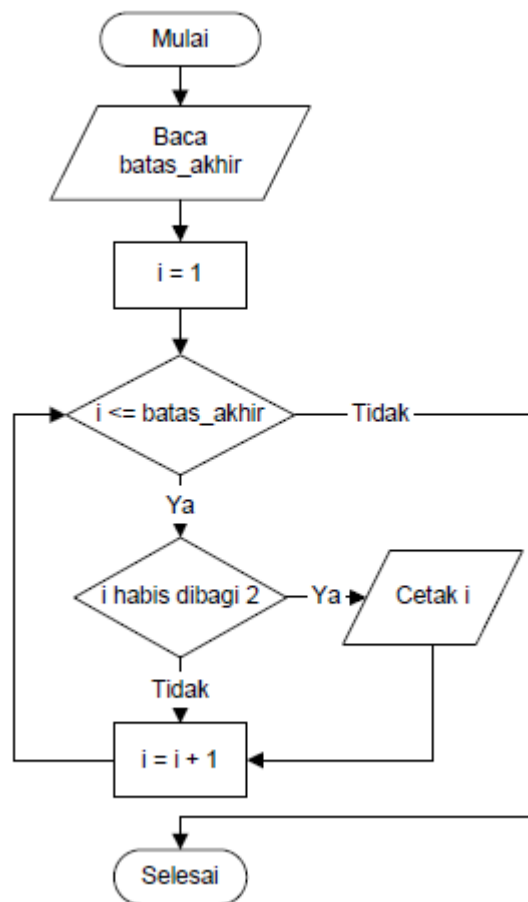
Akan dibuat contoh diatas dengan menggunakan while.

Dicetak bilangan genap dengan batas akhir diinputkan dengan menggunakan while.

Deskripsi :

1. mulai
2. Baca batas_akhir
3. $i = 1$
4. selama $i \leq \text{batas_akhir}$ 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 **Cetak_Angka_Genap** :



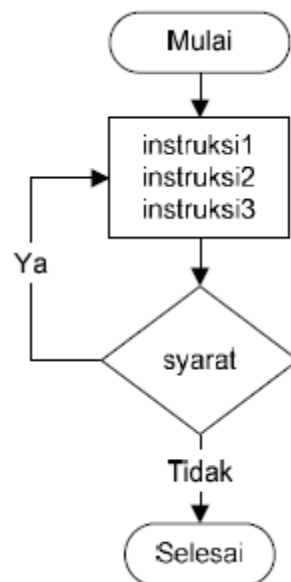
Dari gambar flowchart diatas dapat dijelaskan pertama kali dibaca `batas_akhir` perngulangan, kemudian `i` diberi nilai 1, kemudian diuji apakah `i` lebih kecil atau sama dengan `batas_akhir`, jika benar maka diuji apakah nilai `i` habis dibagi 2, jika benar maka dicetak nilai `i`, kemudian nilai `i` 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.

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** :



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 :

e. Algoritma Cetak Angka DO ... WHILE

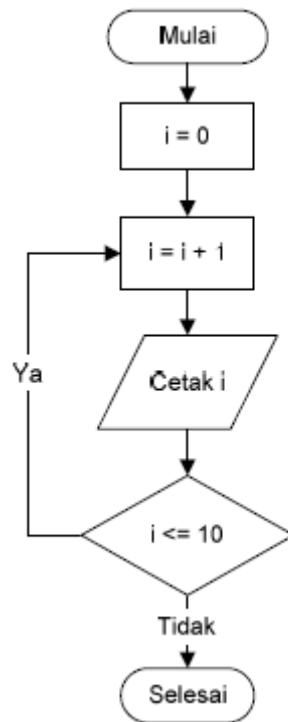
Akan dibuat contoh diatas dengan menggunakan `do...while`.

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** :



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 :

f. Algoritma Cetak Bilangan Genap DO ... WHILE

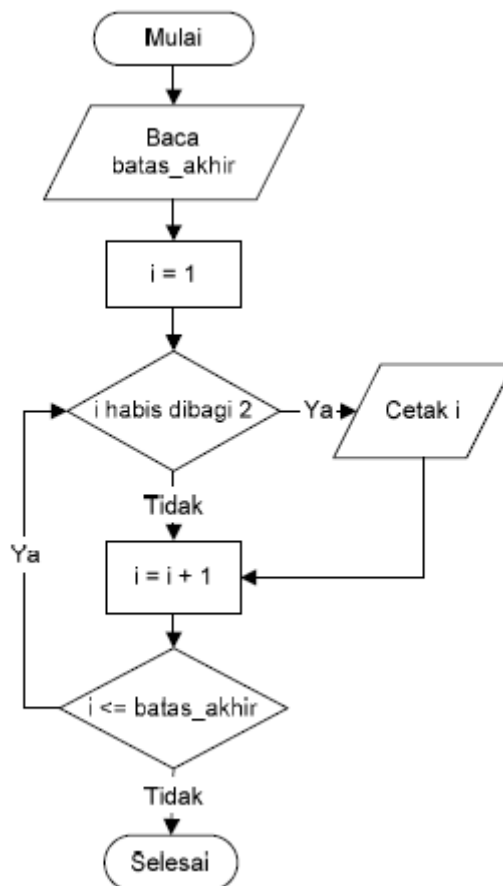
Akan dibuat contoh diatas dengan menggunakan do...while

Dicetak bilangan genap dengan batas akhir diinputkan dengan menggunakan do..while.

Deskripsi :

1. mulai
2. Baca batas_akhir
3. $i = 1$
4. Selama $i \leq \text{batas_akhir}$ 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 **Cetak_Angka_Genap** :



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.

1.4.2 Pseudocode

Menurut wikipedia, pengertian *pseudocode* adalah deskripsi tingkat tinggi informal dan ringkas atas algoritma pemrograman komputer yang menggunakan konvensi struktural atas suatu bahasa pemrograman, dan ditujukan untuk dibaca oleh manusia dan bukan oleh mesin.

Pseudocode merupakan kode yang mirip dengan pemrograman sebenarnya. Pseudocode berasal dari kata Pseudo yang berarti imitasi, mirip, atau menyerupai dengan kode bahasa pemrograman.

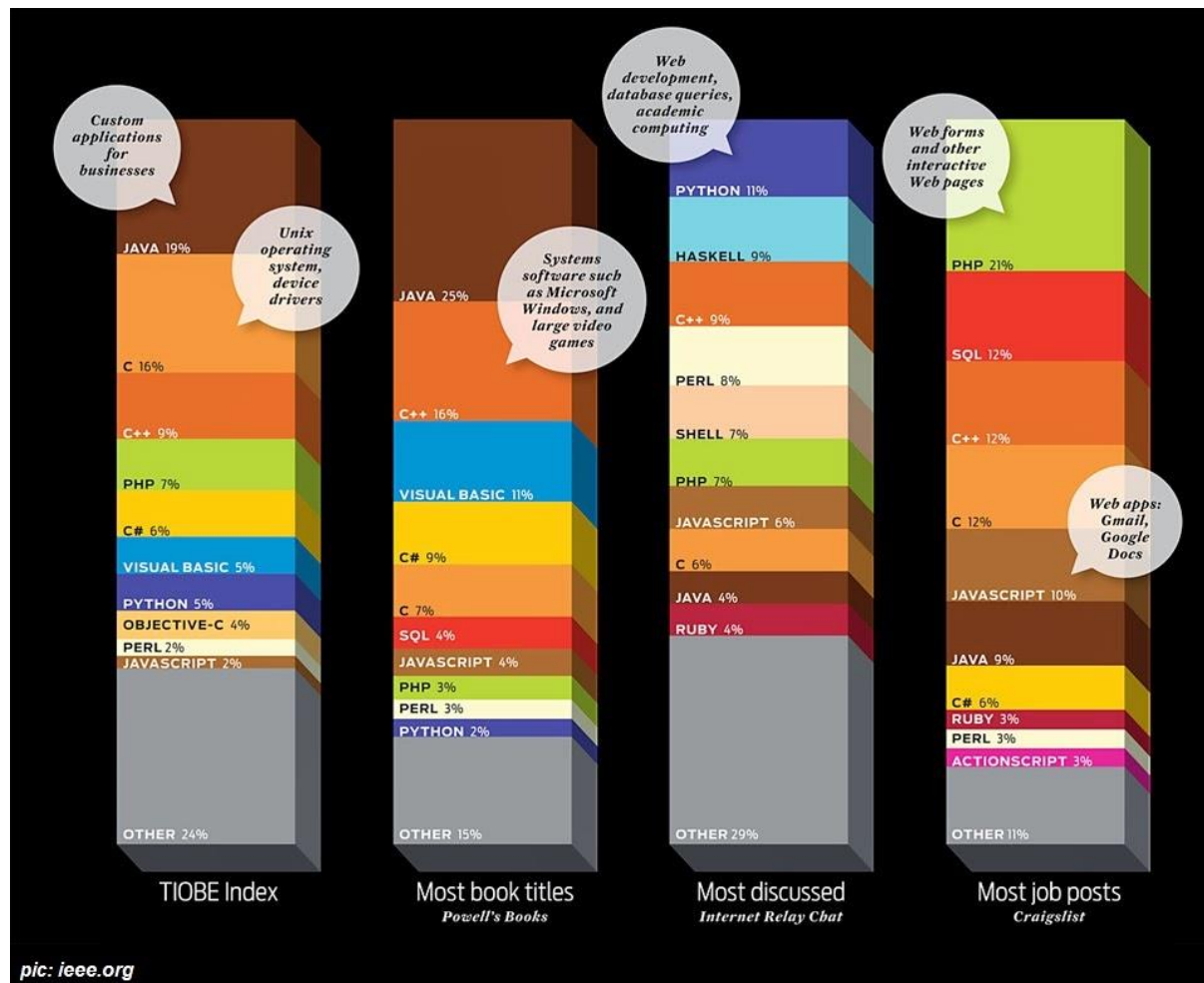
Dalam penulisan pseudocode tidak ada aturan yang baku, oleh karena itu pseudocode biasanya ditulis berbasiskan bahasa pemrograman yang akan digunakan, misalnya Basic, pascal, C++ dan lain-lain. Sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada programmer.

Ciri-ciri Pseudocode

Pseudocode adalah kode / tanda / notasi yang menyerupai atau merupakan penjelasan cara menyelesaikan suatu masalah

1. Pseudocode sering digunakan oleh seseorang untuk menuliskan algoritma dari suatu permasalahan
2. Pseudocode berisikan langkah-langkah untuk menyelesaikan suatu masalah (hampir sama dengan algoritma), hanya saja bentuknya sedikit berbeda dari algoritma
3. Pseudocode menggunakan bahasa yang hampir menyerupai bahasa pemrograman. Selain itu biasanya pseudocode menggunakan bahasa yang mudah dipahami secara universal dan juga lebih ringkas dari pada algoritma.
4. Tidak ada aturan baku yang mengikat tentang penulisan pseudocode

1.5 Mengenal Bahasa Pemrograman



Gambar 1. 2 Algoritma dan Pemrograman

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 berkomunikasi menggunakan kata atau karakter sedangkan komputer dengan kode 0 dan 1.

Oleh sebab itu para pakar dan ilmuwan menciptakan bahasa pemrograman yang tujuannya untuk mempermudah manusia berkomunikasi dengan komputer. Dengan adanya bahasa pemrograman ini, maka bila manusia ingin berkomunikasi dengan komputer tidak harus menerjemahkan ke dalam 0 dan 1. Bila hal itu dilakukan betapa rumitnya suatu program.

1.5.1 Istilah – istilah Dasar dalam bahasa Pemrograman

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

1.5.2 Klasifikasi Bahasa Pemrograman

Secara umum bahasa pemrograman dibagi menjadi empat kelompok :

- **Bahasa Aras Rendah (Low Level Language)**, Merupakan bahasa yang berorientasi pada mesin. Pemrogram dengan bahasa ini harus berpikir berdasarkan logika mesin, sehingga bahasa ini kurang fleksibel dan sulit dipahami. Contoh : Bahasa mesin, Bahasa rakitan (assembly).
- **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, B, Fortran.
- **Bahasa Aras Tinggi (High 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.
- **Bahasa Berorientasi Objek (Object Oriented Programming)**, merupakan kolaborasi antara GUI dengan bahasa pemrograman berorientasi objek, sehingga kita tidak perlu

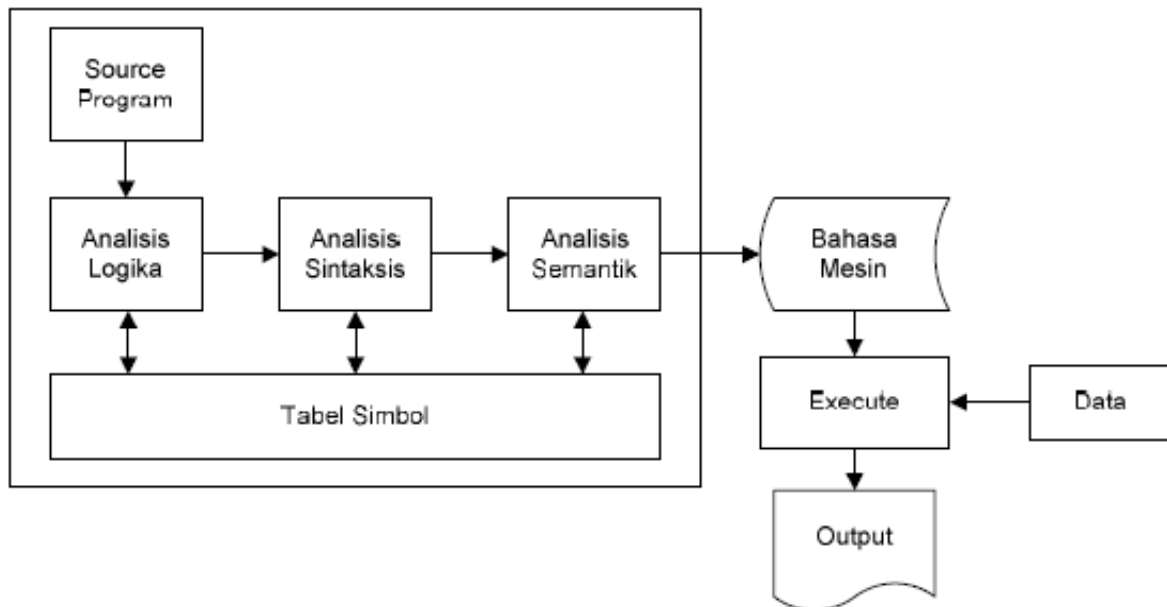
menuliskan secara detail semua pernyataan dan ekspresi seperti bahasa aras tinggi, melainkan cukup dengan memasukkan kriteria-kriteria yang dikehendaki saja. Contoh : Delphi, Visual Basic, C++.

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

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.

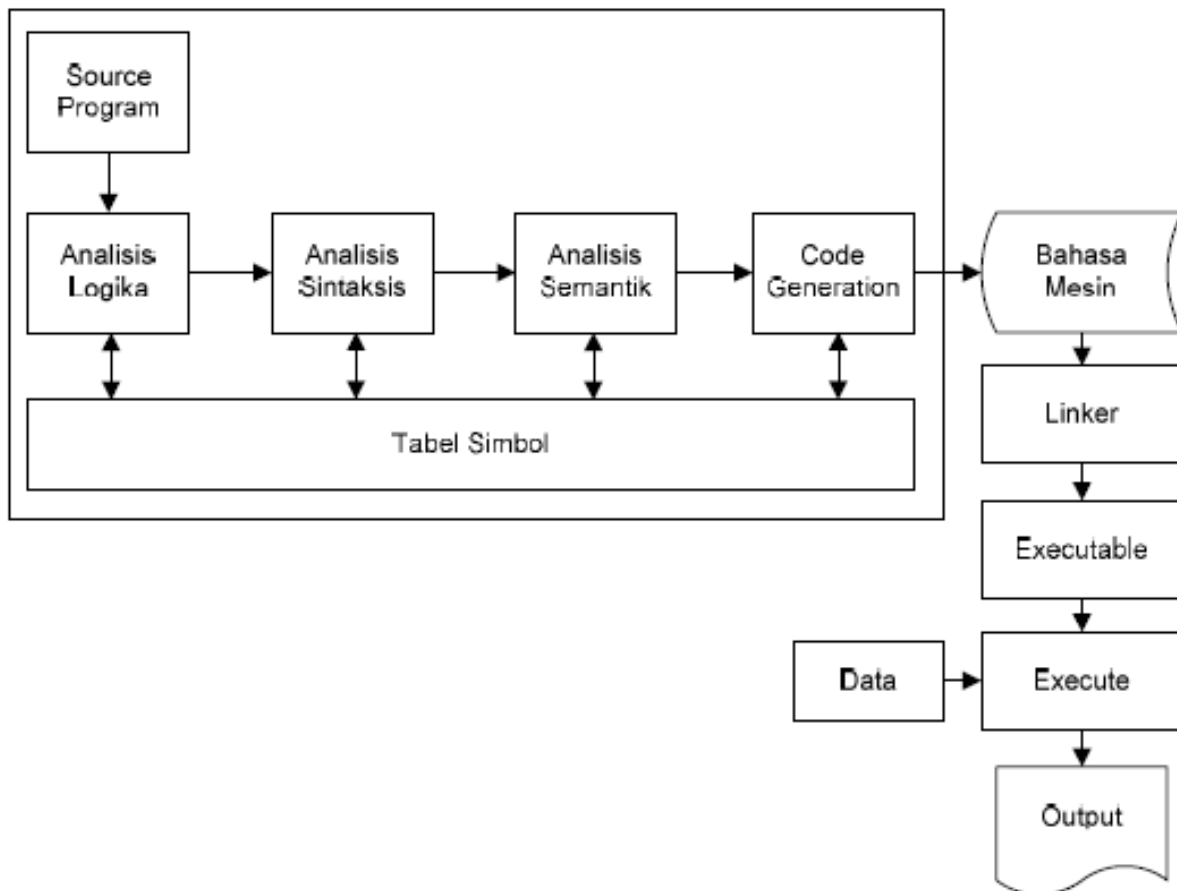
Pada saat menerjemahkan interpreter akan memeriksa sintaksis (sintak program), semantik(arti perintah), dan kebenaran logika. Jika ditemukan kesalahan sintaksis (syntak error) maka interpreter akan menampilkan pesan kesalahan dan eksekusi program langsung terhenti.



Gambar 1. 3 Proses Kerja Interpreter

COMPILER

Berasal dari kata to compile yang berarti menyusun, mengumpulkan atau menghimpun. Compiler merupakan penyusun/penghimpun instruksi-instruksi dalam satu kesatuan modul yang telah berhasil melalui tahap interpreter sehingga menjadi 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 1. 4 Proses kerja Compiler

PERBEDAAN INTERPRETER DAN COMPILER

Tabel 1. 1 Perbedaan Interpreter dan Compiler

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

BAB II PENGENALAN C++

2.1 Apa itu C++ ?

C++ adalah [bahasa pemrograman komputer](#) yang di buat oleh Bjarne Stroustrup, yang merupakan perkembangan dari bahasa C dikembangkan di [Bell Labs](#) ([Dennis Ritchie](#)) pada awal tahun 1970-an, Bahasa itu diturunkan dari bahasa sebelumnya, yaitu B, Pada awalnya, bahasa tersebut dirancang sebagai bahasa pemrograman yang dijalankan pada sistem [Unix](#), Pada perkembangannya, versi ANSI (American National Standard Institute) [Bahasa pemrograman C](#) menjadi versi dominan, Meskipun versi tersebut sekarang jarang dipakai dalam pengembangan sistem dan jaringan maupun untuk sistem embedded, Bjarne Stroustrup pada Bell Labs pertama kali mengembangkan C++ pada awal [1980](#)-an. Untuk mendukung fitur-fitur pada C++, dibangun efisiensi dan sistem support untuk pemrograman tingkat rendah (low level coding).

Pada C++ ditambahkan konsep-konsep baru seperti class dengan sifat-sifatnya seperti inheritance dan overloading.^{[[butuh rujukan](#)]} Salah satu perbedaan yang paling mendasar dengan bahasa C adalah dukungan terhadap konsep pemrograman berorientasi objek ([Object Oriented Programming](#)). [1]

C++ adalah salah satu bahasa pemrograman populer yang sudah terbukti banyak digunakan oleh para praktisi dan ilmuwan untuk mengembangkan program-program (aplikasi) berskala besar seperti games (program permainan dikomputer), program untuk penelitian dibidang sains, embedded system dan lain-lain. [2]

2.2 Sejarah Singkat Lahirnya C++

Tahun 1978, Brian W. Kernighan & Dennis M. Ritchie dari AT & T Laboratories mengembangkan bahasa B menjadi bahasa C. Bahasa B yang diciptakan oleh Ken Thompson sebenarnya merupakan pengembangan dari bahasa BCPL (Basic Combined Programming Language) yang diciptakan oleh Martin Richard.

Sejak tahun 1980, bahasa C banyak digunakan pemrogram di Eropa yang sebelumnya menggunakan bahasa B dan BCPL. Dalam perkembangannya, bahasa C menjadi bahasa paling populer diantara bahasa lainnya, seperti PASCAL, BASIC, FORTRAN.

Tahun 1989, dunia pemrograman C mengalami peristiwa penting dengan dikeluarkannya standar bahasa C oleh American National Standards Institute (ANSI). Bahasa C yang diciptakan Kernighan & Ritchie kemudian dikenal dengan nama ANSI C.

Mulai awal tahun 1980, Bjarne Stroustrup dari AT & T Bell Laboratories mulai mengembangkan bahasa C. Pada tahun 1985, lahirlah secara resmi bahasa baru hasil pengembangan C yang dikenal dengan nama C++. Symbol ++ merupakan operator C untuk operasi penaikan, muncul untuk menunjukkan bahwa bahasa baru ini merupakan versi yang lebih canggih dari C. Sebenarnya

bahasa C++ mengalami dua tahap evolusi. C++ yang pertama, dirilis oleh AT&T Laboratories, dinamakan cfront. C++ versi kuno ini hanya berupa kompiler yang menterjemahkan C++ menjadi bahasa C.

Borland International merilis compiler Borland C++ dan Turbo C++. Kedua compiler ini sama-sama dapat digunakan untuk mengkompilasi kode C++. Bedanya, Borland C++ selain dapat digunakan dibawah lingkungan DOS, juga dapat digunakan untuk pemrograman Windows. Pada evolusi selanjutnya, Borland International Inc. mengembangkan kompiler C++ menjadi sebuah kompiler yang mampu mengubah C++ langsung menjadi bahasa mesin (assembly).

Sejak evolusi ini, mulai tahun 1990 C++ menjadi bahasa berorientasi obyek (OOP) yang digunakan oleh sebagian besar pemrogram professional. Selain Borland International, terdapat beberapa perusahaan lain yang juga merilis compiler C++, seperti Topsispeed C++ dan Zortech C++.

Atau tonton langsung di <https://youtu.be/UnlC7Xp4KxI>

2.3 Hubungan antara C dan C++

C++ merupakan bentuk perluasan dari bahasa C. C dan C++ seperti layaknya saudara kandung, kakak-adik. Dalam bahasa C/C++. Tanda ++ merupakan increment, yaitu proses penambahan nilai dengan 1. Dengan demikian, C++ bearti C+1. Nilai 1 disini melambangkan dukungan terhadap pemrograman berorientasi objek. Dengan demikian, C++ adalah bahasa C yang ditambah dengan kemampuan atau dukungan terhadap pemrograman berorientasi objek. Kedua bahasa ini (C dan C++) merupakan bahasa yang sangat populer dalam dunia pengembangan perangkat lunak. Keduanya digolongkan kedalam bahasa tingkat menengah (middle level language). Berikut pengelompokan tingkatan dari bahasa pemrograman:

| | |
|-------------------------|-----------------|
| Bahasa Tingkat Tinggi | Ada |
| | Modula-2 |
| | Pascal |
| | COBOL |
| | FORTRAN |
| | BASIC |
| | |
| Bahasa Tingkat Menengah | Java |
| | C++ |
| | C |
| | FORTH |
| | |
| Bahasa Tingkat Rendah | Macro-Assembler |
| | Assembler |

Gambar 2. 1 Pengelompokan bahasa Pemrograman

Dari tabel tersebut dapat kita lihat bahwa bahasa pemrograman yang terdapat pada bagian paling atas merupakan bahasa pemrograman yang paling mudah untuk difahami. Sebagai contoh, C adalah bahasa yang lebih sulit dibandingkan C++ dan C++ adalah bahasa yang lebih sulit dibandingkan dengan bahasa Java, dan seterusnya.

2.4 C++ Standar : C++98, C++03, dan C++11 (C++0x)

C++ Mulai dikembangkan sejak tahun 1979 oleh Bjarne Stroustrup di laboratorium Bell. C++ dibentuk dari bahasa C dengan penambahan fitur-fitur baru seperti kelas, fungsi virtual, operator overloading dan multiple inheritance, yang sebelumnya tidak dimiliki oleh bahasa C.

Pada awal kemunculannya, bahasa ini disebut sebagai “C with Classes”. Selanjutnya, pada tahun **1983** diganti menjadi C++, yang artinya **C + kelas**.

Setelah beberapa tahun pengembangan C++ yang terus berlanjut, akhirnya pada tahun **1998** bahasa pemrograman distandarisasi oleh ISO (badan standarisasi internasional) dengan nama standar ISO/IEC 14882:1998 atau dikenal dengan istilah **C++98**. Dengan fitur baru seperti template, namespace, exception handling.

Pada tahun **2003**, C++ kembali mengalami proses standarisasi dengan nama standar ISO/IEC 14882:2003 dikenal dengan istilah **C++03**.

Terakhir, tahun 2011, tepatnya tanggal 12 Agustus, C++ kembali revisi dan standarisasi ulang, dengan nama standar ISO/IEC 14882:2011, dikenal dengan istilah C++11 atau C++0x.

Jadi Bahasa C++ yang beredar sebelum proses standarisasi (sebelum 1998) sering disebut C++ tradisional atau klasik. Maka dari itu, compiler lama seperti Turbo C++ dan Borland C++ tidak dapat digunakan untuk melakukan kompilasi terhadap kode-kode program yang ditulis dalam C++ baru (C++ standar).

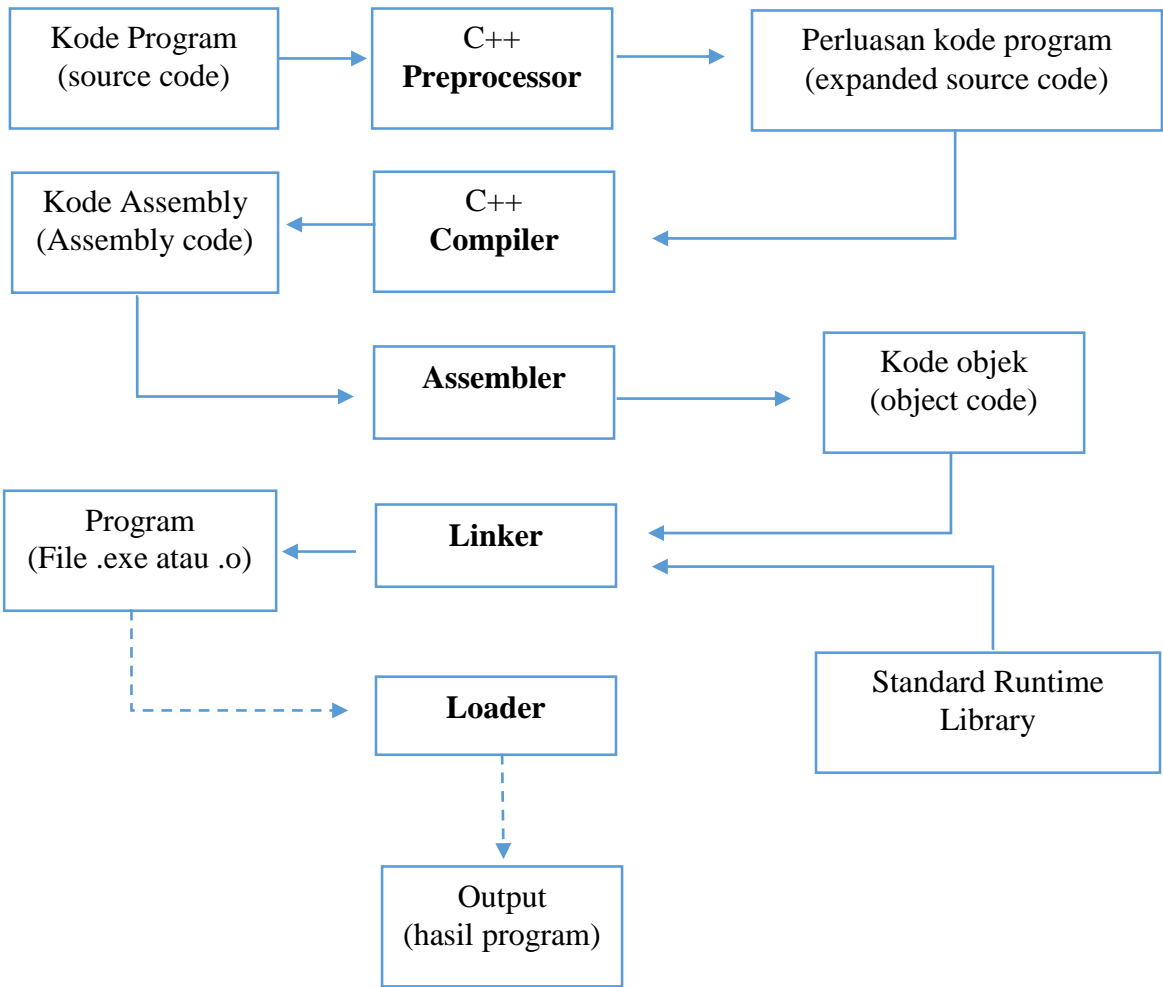
2.5 Proses Pembentukan Program dalam C++

Terdapat beberapa tahapan yang terjadi pada saat proses pembentukan program, dari kode program menjadi file biner yang dapat dieksekusi. Dalam sistem operasi Windows, file program akan berekstensi .exe. dalam Linux, biasanya berekstensi .o. Masing-masing tahapan dilakukan oleh program lain yang berbeda-beda, yaitu:

- C++ Preprocessor
- C++ Compiler
- Assembler
- Linker

Setelah program (file.exe atau .o) terbentuk, file tersebut dapat dieksekusi atau dijalankan (Run). Program yang melakukan ekstensi file ini disebut Program Loader.

Berikut ini gambar yang mengilustrasikan proses pembentukan program di dalam C++ (berlaku juga untuk program-program yang ditulis dalam bahasa C)



Gambar 2. 2 Proses pembentukan Program

Program C++ Preprocessor

Pada pembentukan program yang ditulis dalam bahasa C++, mula-mula code program akan diproses oleh program C++ Preprocessor. Program ini akan menggabungkan kode program dengan preprocessor directive yang digunakan didalam code.

Preprocessor directive adalah statemen program yang diawali dengan tanda #.

Pada tahap ini, statemen #include, #define, dan lain-lain akan diubah kedalam bentuk kode aktual. Sebagai contoh, apabila didalam code program terdapat statemen #include <iostream> maka seluruh file “iostream” akan dimasukkan kedalam code program. Selanjutnya, bentuk kode yang sudah diperluas ini kemudian siap dikirim ke program C++ Compiler.

Program C++ Compiler

Tahap berikutnya setelah tahap pemrosesan awal code program adalah tahap penerjemahan kode program menjadi code assembly. Proses pada tahap ini dilakukan oleh program C++ Compiler. Kode yang sudah berubah ke bentuk kode assembly ini kemudian siap dikirim ke program Assembler.

Program Assembler

Setelah program Assembler menerima kiriman berupa code assembly dari C++ Compiler, program ini akan langsung menerjemahkan kode=kode yang terdapat didalamnya kedalam bentuk kode objek (bahasa mesin). Selanjutnya, kode objek ini akan dikirim ke program Linker.

Program Linker

Apabila kode program yang kita tulis lebih dari satu file, maka kode objek yang dihasilkan juga akan sesuai dengan jumlah file kode program. Pada tahap ini, program Linker akan menghubungkan semua file kode objek tersebut dengan standard Runtime Library yang sudah disediakan oleh C++, kemudian mengubahnya ke dalam satu file .exe (atau .o). file inilah yang disebut dengan program. sampai tahap ini, pembentukan program sudah berakhir.

Program Loader

Program Loader adalah program yang digunakan untuk mengeksekusi program yang sudah kita buat. Sebagai contoh, dalam aplikasi Dev-C++, ketika kita menggunakan menu Compile maka Dev-C++ akan melakukan proses pembentukan program.

Disini yang akan dipanggil adalah program *C++ Preprocessor*, *C++ Compiler*, *Assembler*, dan *Linker*. Akan tetapi, pada saat menggunakan menu Run, maka loader akan dipanggil untuk mengeksekusi program yang telah kita buat sehingga hasil (output) program akan ditampilkan dilayar monitor (console).

2.6 Kerangka Kode Program dalam C++

Setiap program yang ditulis dalam bahasa C/C++ pasti akan memiliki sebuah fungsi utama dengan nama **main()**.

Dibawah ini akan diberikan kerangka umum dari kode program yang ditulis dalam bahasa C dan C++.

Berikut ini kerangka umum dari program yang ditulis dalam bahasa C.

```
#include <stdio.h>

// Prototipe fungsi
tipe_data nama_fungsi1(parameter1, parameter2, ....);
tipe_data nama_fungsi2(parameter1, parameter2, ...);
....

// Fungsi Utama
int main(void) {
    Statemen_yang_akan_dilakukan;
    ...
    return 0;
}
```

Perhatikan kata kunci **void** diatas, didalam bahasa C kata tersebut masih banyak dijumpai, namun didalam program C++ sebaiknya dihindari penggunaanya.

```
void main () {  
    Statemen_yang_akan_dilakukan;  
    ...  
}
```

Bentuk diatas berlaku untuk bahasa C dan C++ versi lama (klasik).

```
#include <iostream>  
  
Using namespace std;  
  
// Prototipe fungsi  
tipe_data nama_fungsi1(parameter1, parameter2, ....);  
tipe_data nama_fungsi2(parameter1, parameter2, ...);  
....  
  
// Fungsi Utama  
int main() {  
    Statemen_yang_akan_dilakukan;  
    ...  
    return 0;  
}
```

File header yang digunakan didalam bahasa C++ untuk proses input/output (I/O) satandar adalah **<iostream>**, sedangkan didalam bahasa C file header yang digunakan untuk keperluan yang sama adalah **<stdio.h>**.

Using namespace std;

Statemen diatas menunjukkan bahwa kita menggunakan bahasa C++ standar didalam kode program yang kita tulis.

2.7 Program yang ditulis dalam bahasa C

Pada contoh ini, kita akan membuat program yang meminta user untuk memasukkan bilangan bulat. Selanjutnya, program akan menampilkan kembali bilangan yang telah dimasukkan.

Kode Program 1-1

```
#include <stdio.h>

int main(void){
    int x;

    /* Menampilkan teks untuk informasi */
    printf ("Masukkan bilangan bulat: ");

    /* Membaca nilai dari keyboard
       dan menyimpannya kedalam variabel x */
    scanf ("%d", &x);

    /* Menampilkan nilai yang telah dimasukkan */
    printf ("Bilangan yang telah dimasukkan adalah %d", x);

    return 0;
}
```

2.8 Pengertian dan kegunaan File Header (File.h)

File Header (file dengan ekstensi .h) adalah file yang berisi deklarasi, baik berupa konstanta, fungsi, kelas, namespace, dan sebagainya.

Kode Program 1-2

```
#include <iostream>

using namespace std;

int main(){
    int x;

    /* Menampilkan teks untuk informasi */
    cout<<"Masukkan bilangan bulat: ";

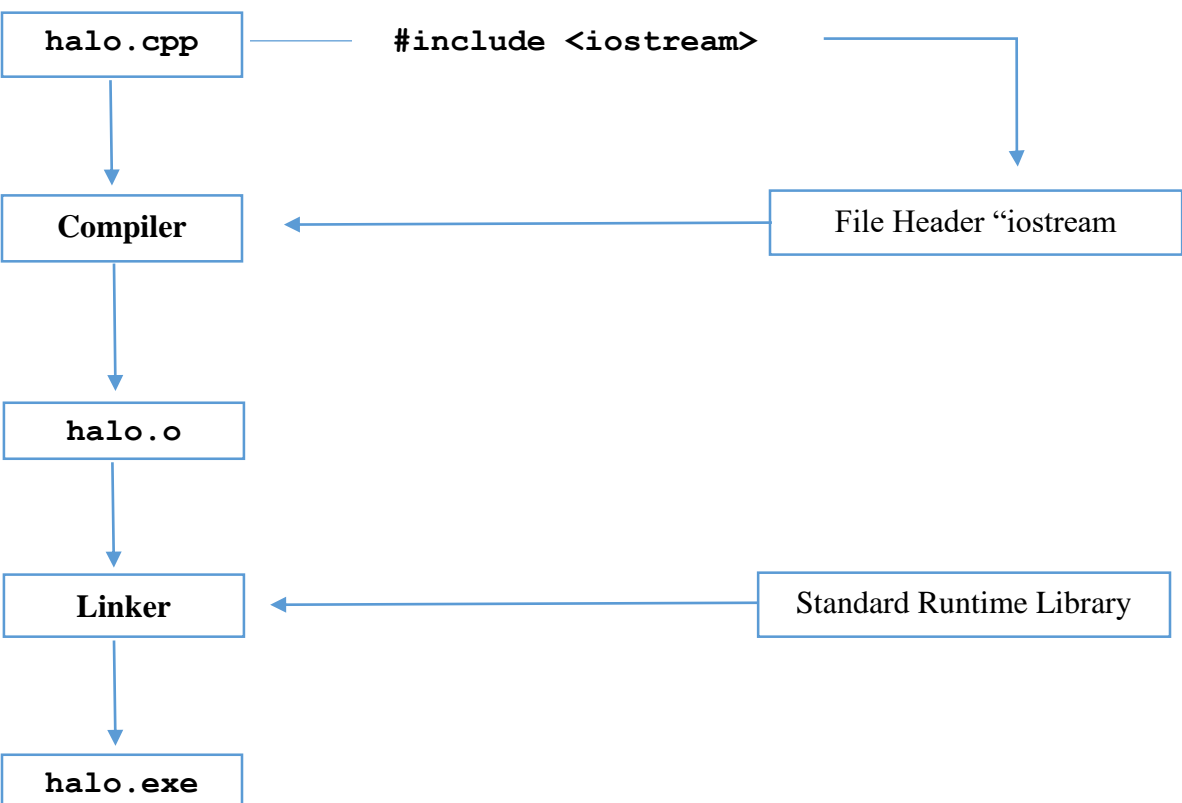
    /* Membaca nilai dari keyboard
       dan menyimpannya kedalam variabel x */
    cin >> x;

    /* Menampilkan nilai yang telah dimasukkan */
    cout<<"Bilangan yang telah dimasukkan adalah" << x;

    return 0;
}
```

C++ menyediakan banyak file header untuk berbagai macam keperluan. Perlu diingat bahwa sebagian besar file header dalam C++ standar sudah tidak memiliki ekstensi .h, misalnya <iostream>, <string>, <cstring>, <cstdlib>, dan sebagainya. Dalam C++ klasik atau tradisional, ekstensi .h wajib disertakan, misalnya <iostream.h>, <string.h> dan sebagainya.

Sebagai contoh, asumsikan kita memiliki kode program yang disimpan ke dalam file `halo.cpp`. dalam kode tersebut kita menggunakan file header “`iostream`”. Pada situasi ini, proses pembentukan program dapat digambarkan seperti berikut:



Gambar 2. 3 Proses Pembentukan Program

Gambar diatas merupakan gambar yang sudah disederhanakan tanpa adanya program C++ Preprocessor dan Assembler.

C++ juga akan mengenal semua file header milik bahasa C. hanya saja, semua nama file header tersebut sudah berubah. Dalam bukunya, *The C++ Programming Language*, Bjarne Stroustrup (pencipta C++) menyatakan bahwa setiap file header dalam C++ yang diawali huruf ‘c’ adalah sama dengan file header milik bahasa C.

Sebagai contoh, file header `<stdio.h>`, `<string.h>` dan `<stdlib.h>` milik bahasa C. Dalam bahasa C++ akan dirubah menjadi `<cstdio>`, `<cstring>`, dan `<cstdlib>`. Dengan demikian, file header lain yang tidak diawali dengan huruf ‘c’ merupakan library murni milik C++ dan tidak dikenal dalam bahasa C, misal `<iostream>`, `<string>`, `<new>`, `<list>`, dan sebagainya.

2.9 Software yang dibutuhkan

Untuk membuat program yang ditulis dalam bahasa C++, kita memerlukan dua software:

1. Text Editor, digunakan untuk menuliskan kode-kode program.
2. C++ Compiler, digunakan untuk menterjemahkan kode C++ menjadi program.

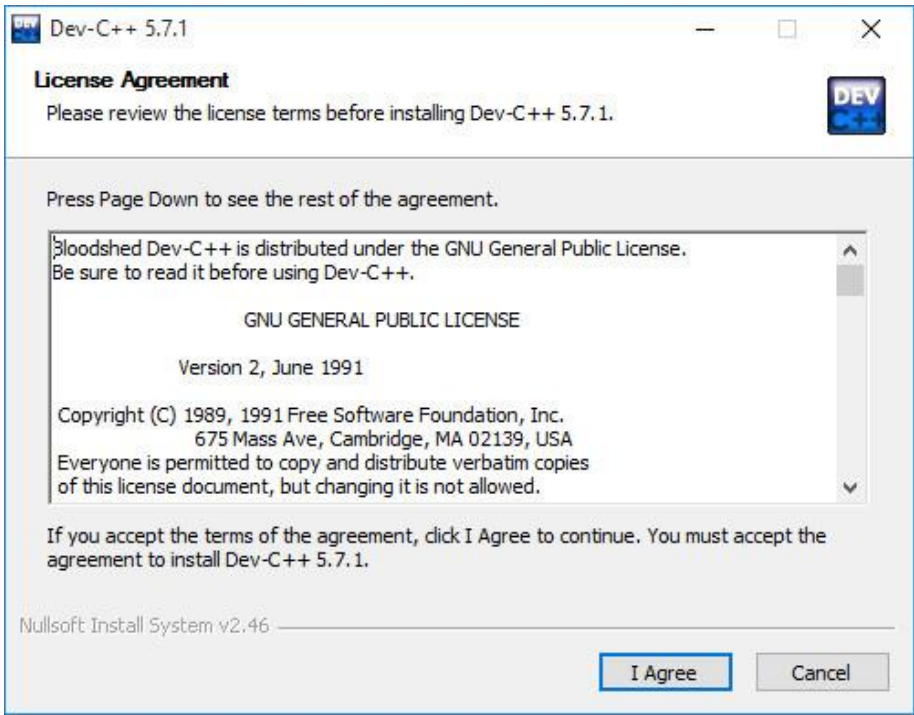
Pada modul ini kita menggunakan Dev-C++ merupakan sebuah C/C++ dalam IDE (Integrated Development Environment) yang sudah dilengkapi MinGW C/C++ Compiler.

Anda dapat memperoleh informasi dan mengunduh De-C++ versi terbaru melalui alamat web berikut: <http://sourceforge.net/projects/orwelldevcpp/files/Compilers/MinGW/>

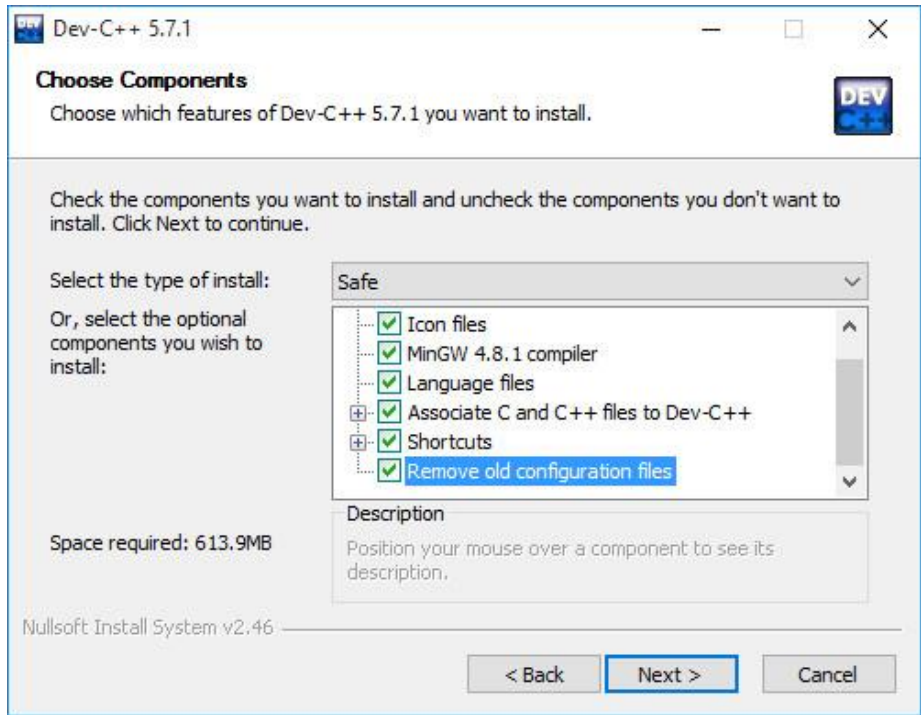
2.10 Instalasi dan Konfigurasi Software Dev-C++

Instalasi software Dev-C++ sangat mudah dilakukan, berikut ini adalah tahapan dan screenshot dari proses instalasi serta konfigurasi Dev-C++ :

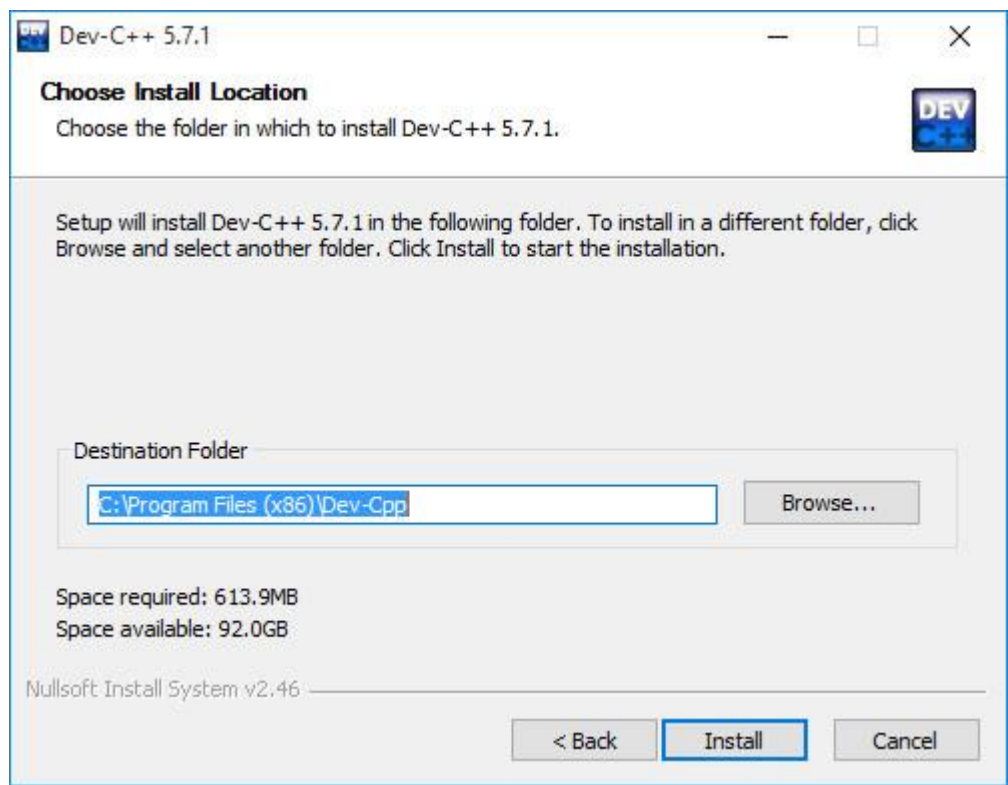
1. Jalankan file installer Dev-C++



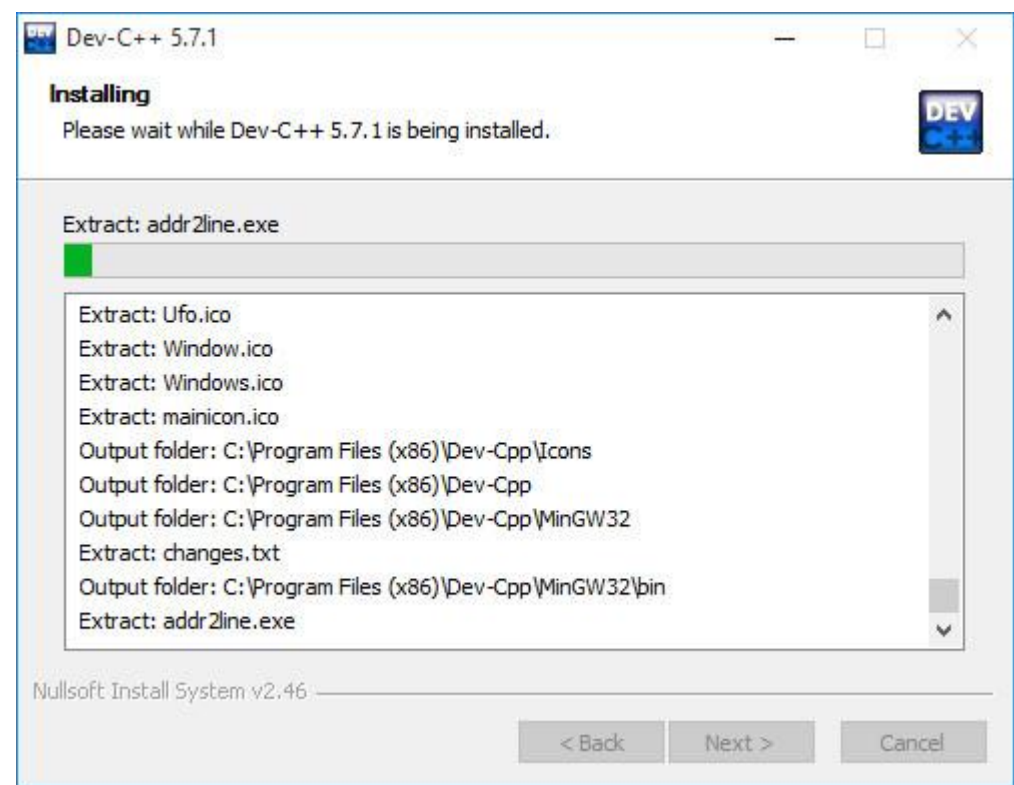
2. Klik **I Agree**



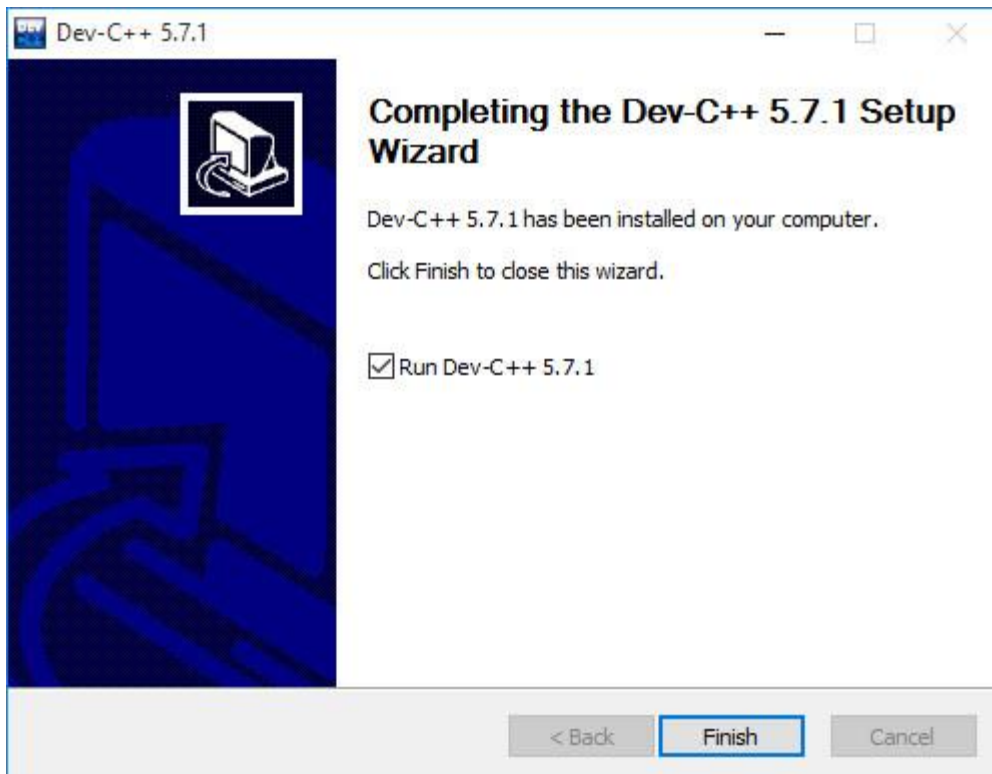
3. Pilih komponen-komponen yang akan di-install, lalu klik **Next**



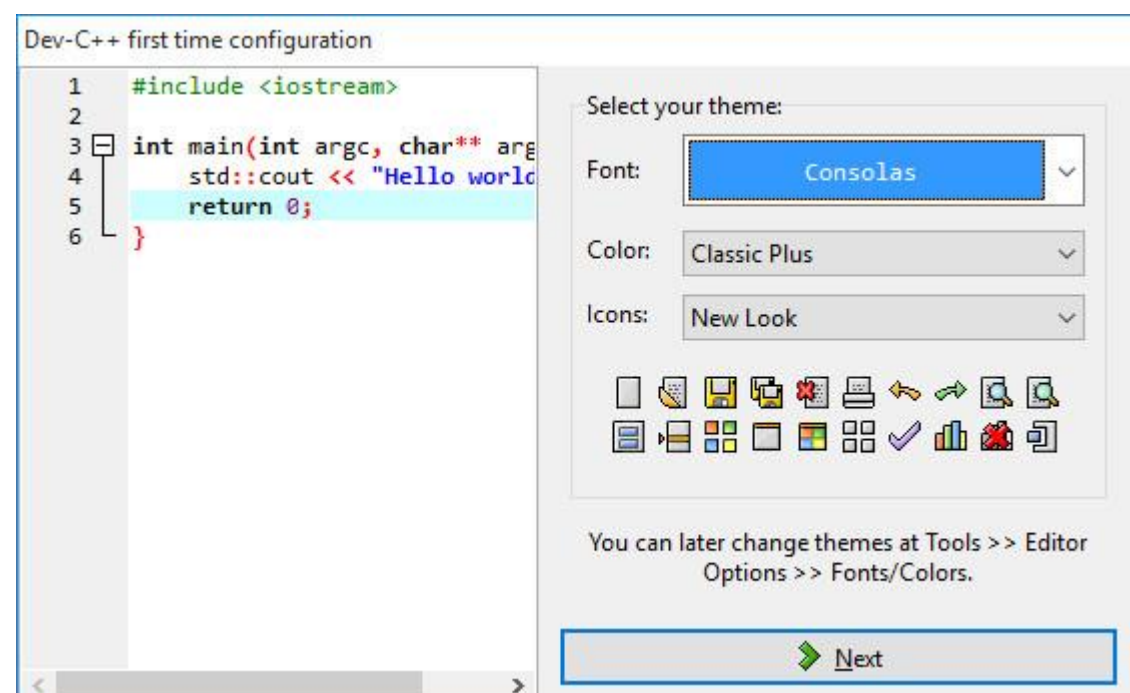
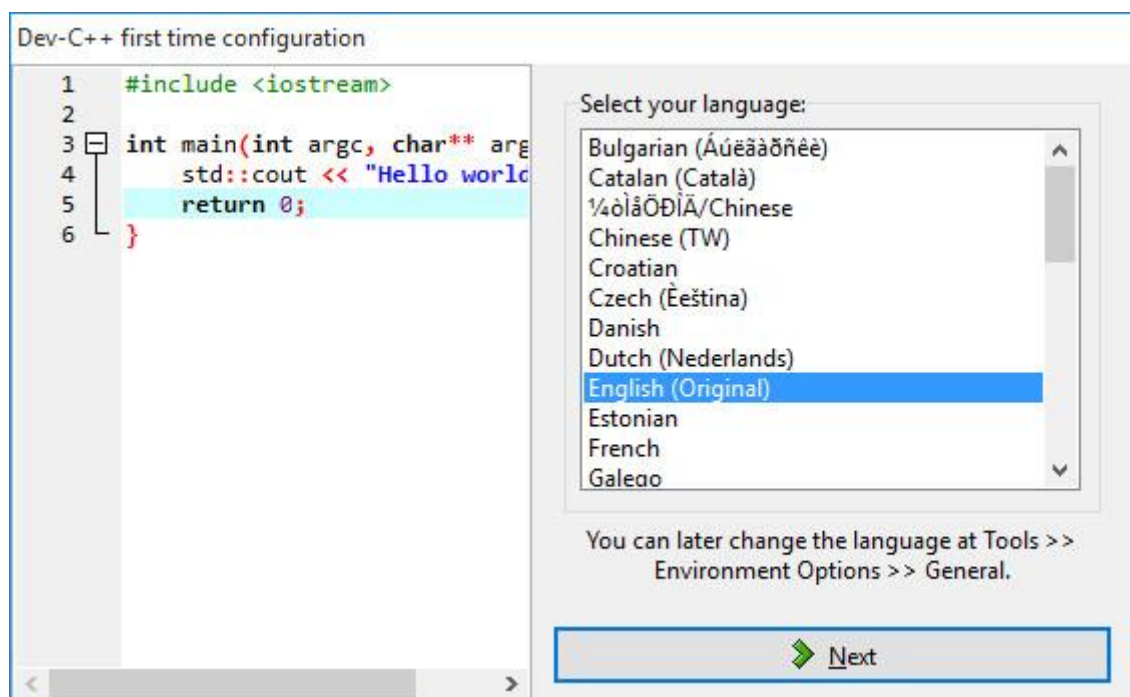
4. Tentukan direktori tujuan instalasi, lalu klik **Install**

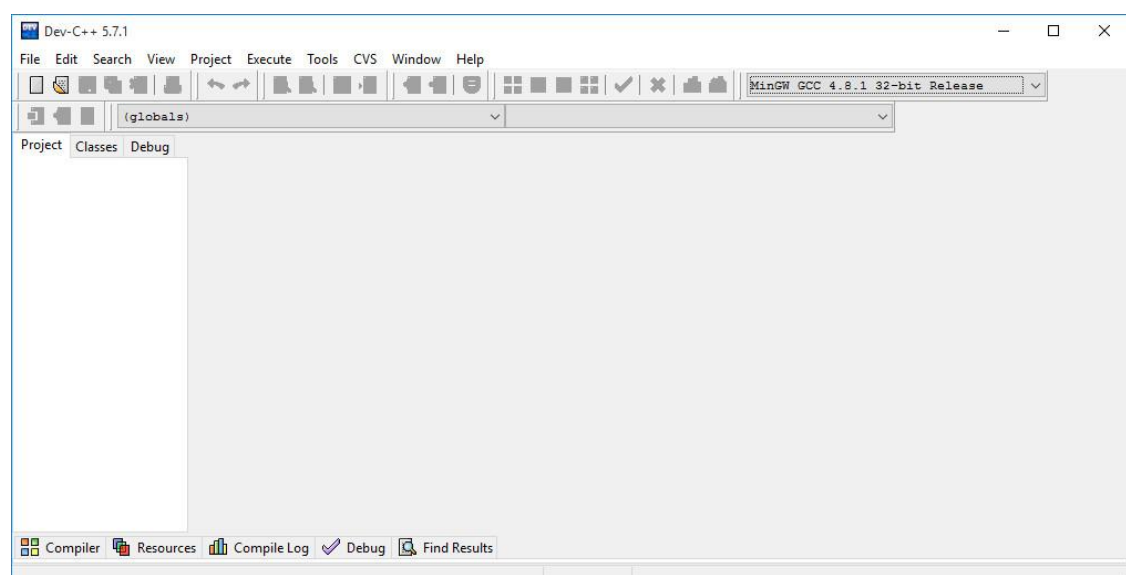
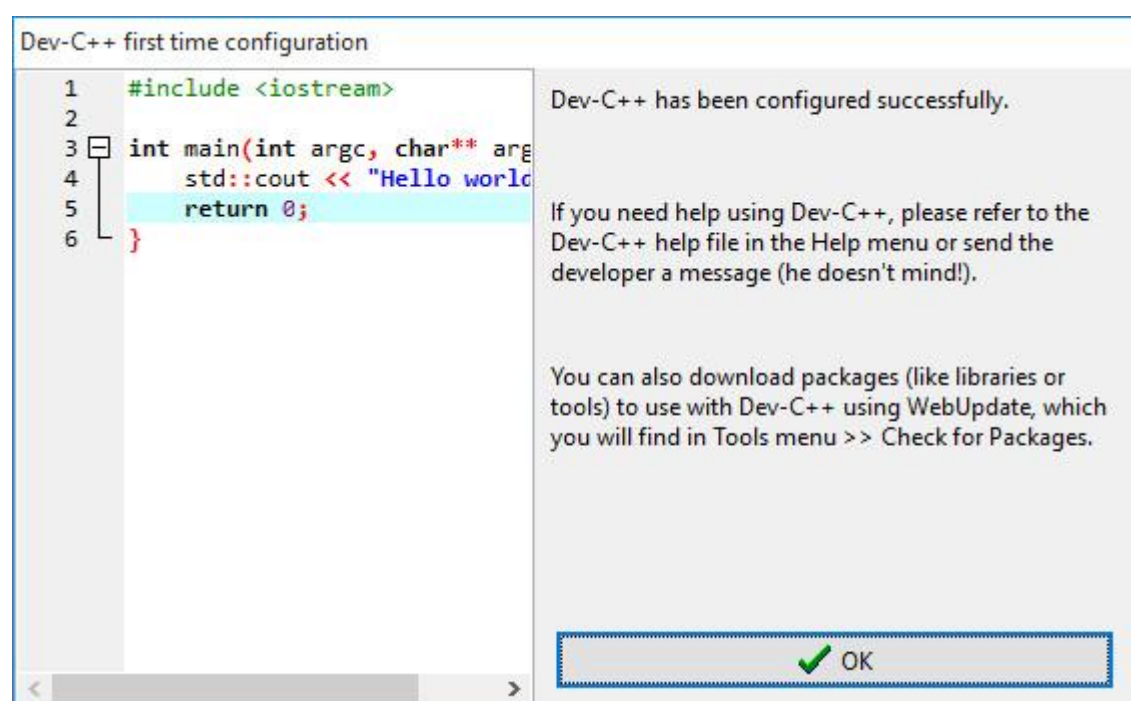
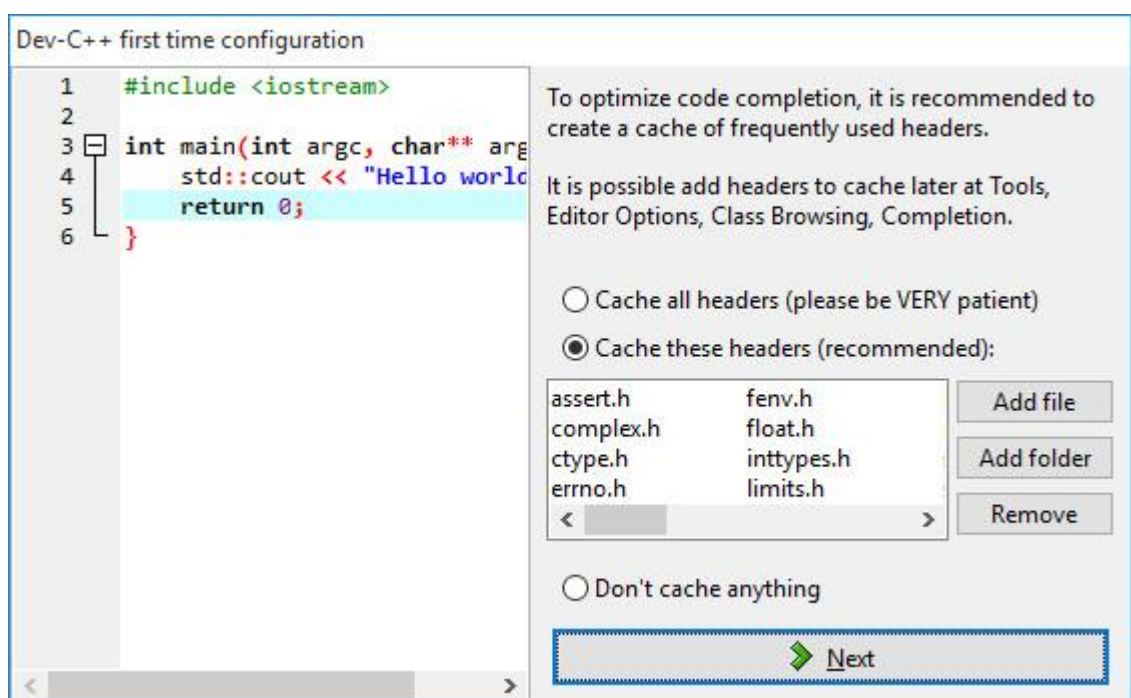


5. Tunggu beberapa saat sampai proses instalasi selesai.

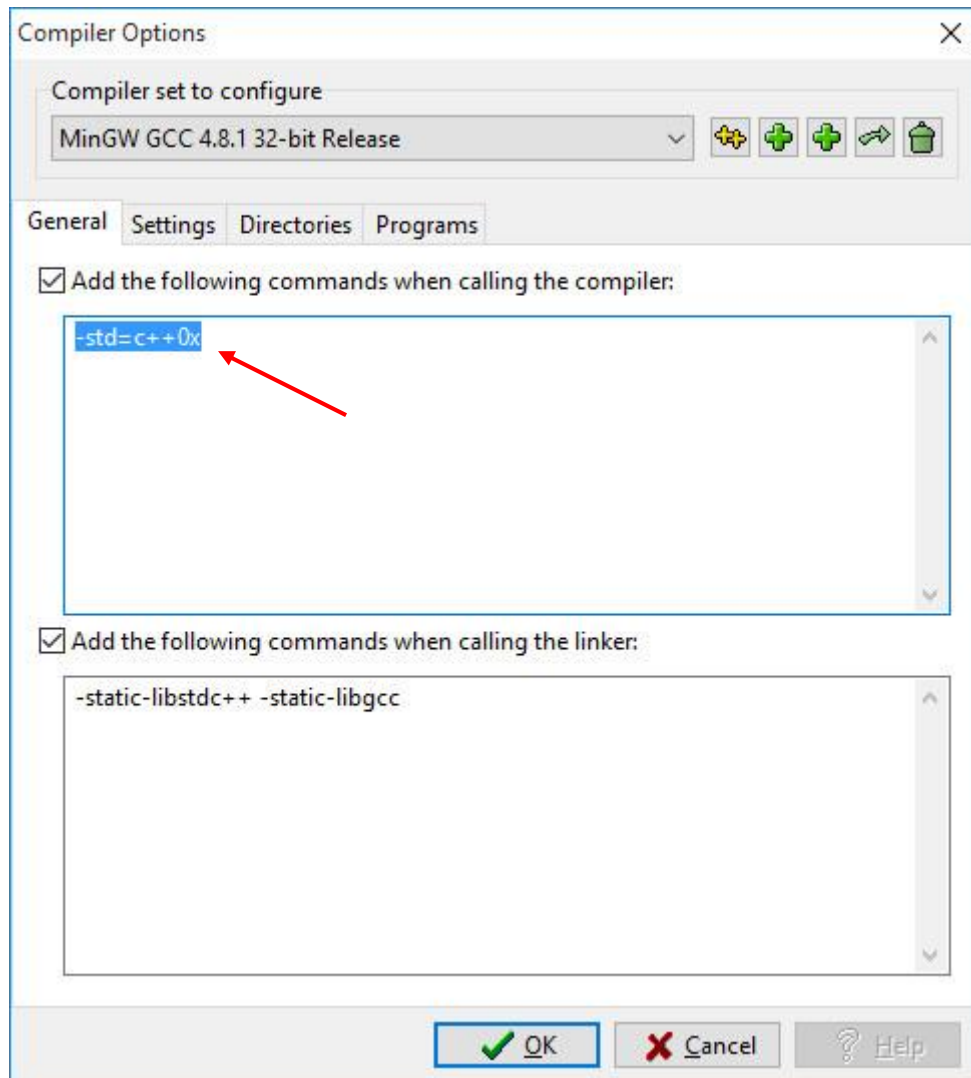


6. Klik **Finish** untuk mengakhiri proses instalasi dan menjalankan aplikasi Dec-C++





7. Agar Dev-C++ dapat mendukung C++11 (c++0x), pilih menu **Tools | Compiler Options**



8. Beri tanda centang (checklist) pada bagian “Add the Following commands when calling the compiler:”, lalu isikan teks berikut;
-std=c++0x
9. Klik OK untuk mengakhiri konfigurasi

2.11 Membuat Program “I Love C++”

Untuk memulai program dalam C++ menggunakan Dev-C++, silahkan ikuti langkah-langkah berikut:

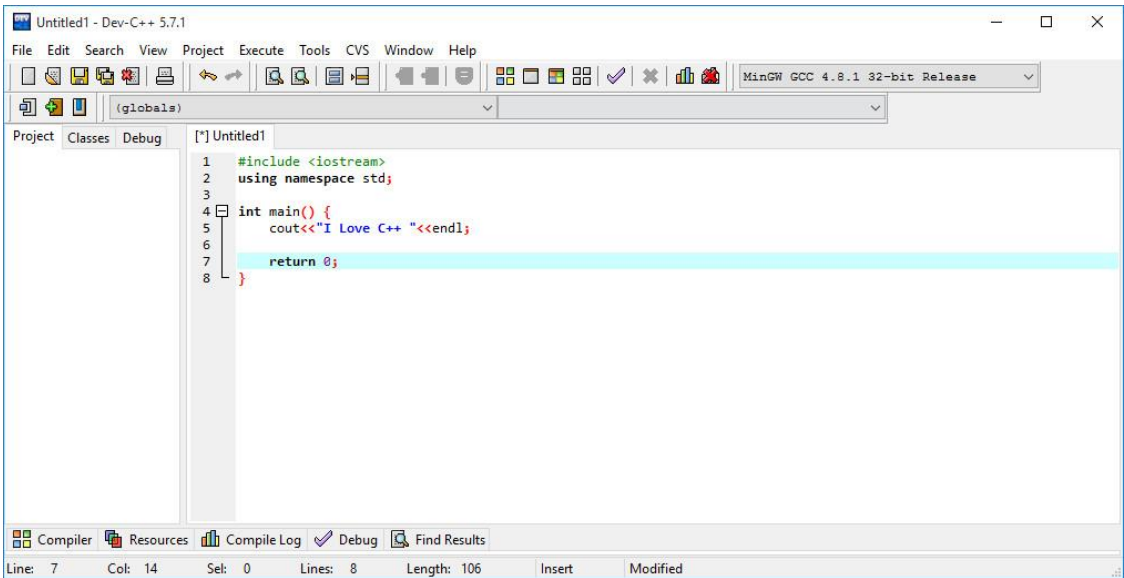
1. Jalankan aplikasi Dev-C++
2. Pilih menu **File | New | Source File**
3. Dalam *Code Editor*, tuliskan kode program berikut:

Kode Program 1-3

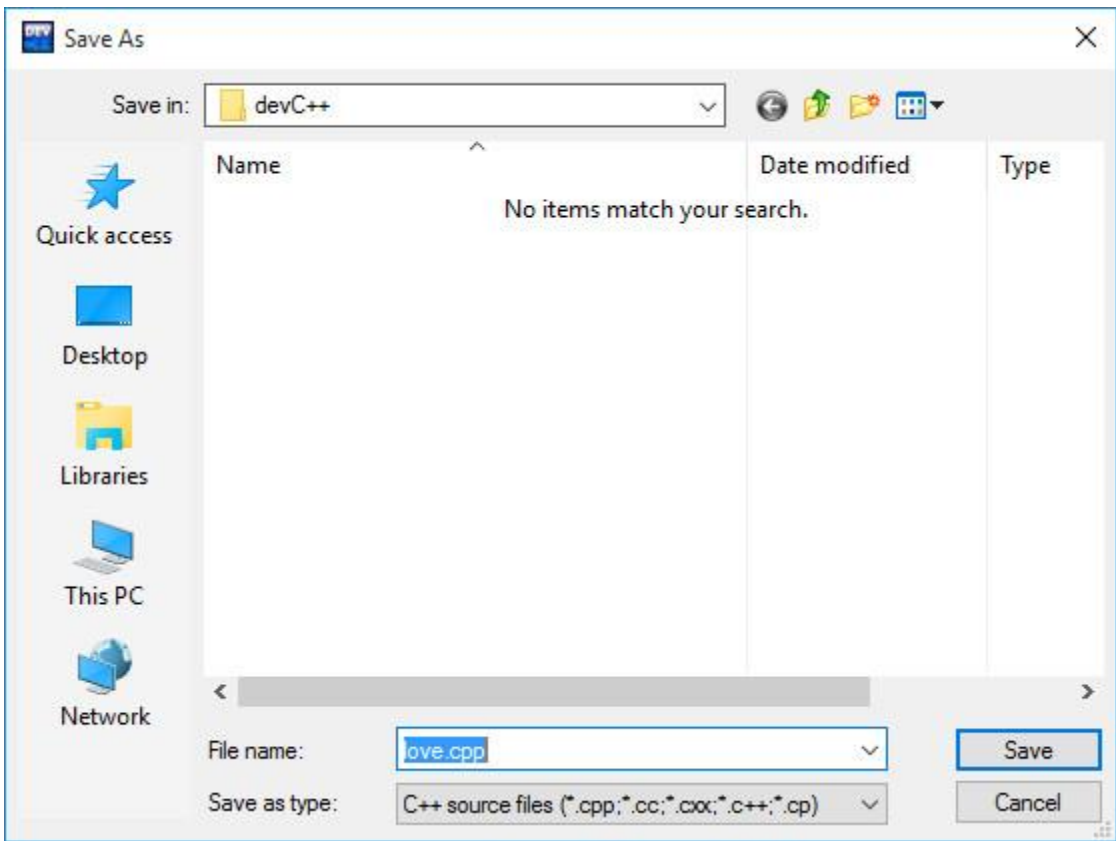
```
#include <iostream>
using namespace std;

int main() {
    cout<<"I Love C++"<<endl;

    return 0;
}
```



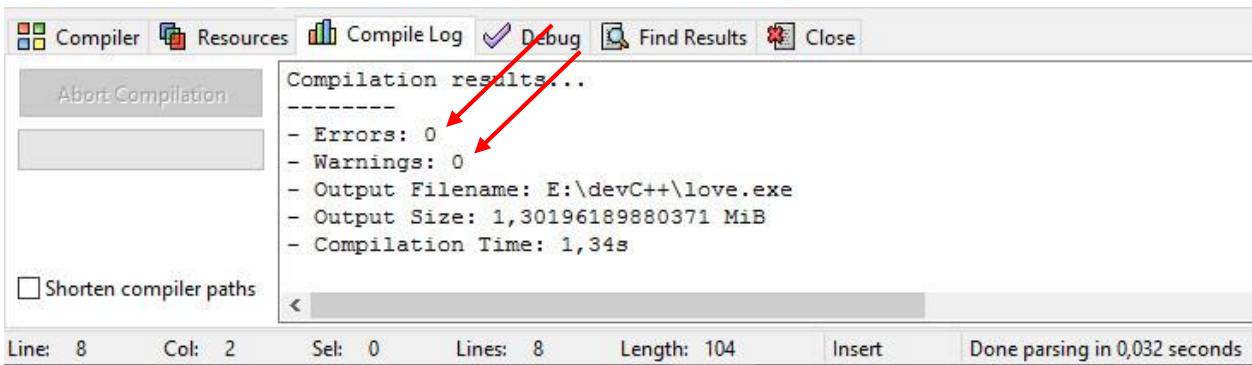
4. Pilih menu File | Save untuk melakukan penyimpanan file kode program



5. Isi nama file dengan love.cpp, lalu simpan didirektori kerja anda

2.12 Kompilasi dan Eksekusi Program

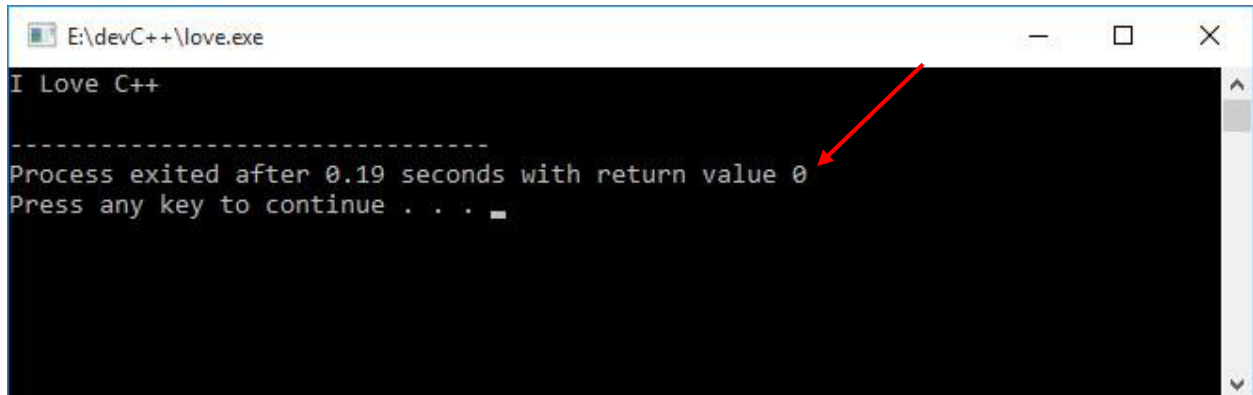
Setelah kode program terbuat dan tersimpan didirektori kerja, langkah selanjutnya adalah melakukan kompilasi terhadap kode tersebut. Dalam Dev-C++, anda hanya perlu memilih menu **Execute | Compile (F9)** untuk melakukan kompilasi



Pastikan bagian Errors dan Warnings bernilai 0. Hal ini menunjukkan kode yang kita tulis betul-betul benar, tidak ada kesalahan maupun peringatan.

Selanjutnya, eksekusi atau jalankan program dengan menu **Execute | Run (F10)**.

Hasilnya sebagai berikut:



```
E:\devC++\love.exe
I Love C++
-----
Process exited after 0.19 seconds with return value 0
Press any key to continue . . .
```

Nilai 0 yang tampil pada hasil diatas menunjukkan bahwa program dihentikan secara normal (tanpa kesalahan).

BAB III

KOMENTAR, IDENTIFIER, DAN TIPE DATA

Variabel dan tipe data merupakan komponen terpenting dalam pemrograman. Setiap program yang menggunakan nilai pasti akan membutuhkan variabel sebagai penampung nilai tersebut. Nilai yang akan digunakan dalam program dapat berupa numerik, karakter, maupun teks. Nilai selanjutnya akan disimpan kedalam memori komputer. Karena itulah kita memerlukan tipe data untuk memberitahukan kepada compiler jenis data apa yang akan disimpan kedalam memori komputer.

Untuk program sederhana yang tidak membutuhkan nilai didalam memori komputer, sebagai contoh:

Kode Program 2-1

```
#include <iostream>

using namespace std;

int main()
{
    cout<<"Teks ini hanya sebagai Contoh"<<endl;
    cout<<2+5;

    return 0;
}
```

- Pada contoh diatas, kita sama sekali tidak memiliki variabel karena nilai langsung ditampilkan dilayar monitor.
- Dalam bahasa C++, perintah yang digunakan untuk menampilkan teks atau bilangan ke layar adalah `cout` (dibaca: si out). Perintah ini akan dikenal oleh compiler apabila kita menggunakan namespace `std`.
- Namespace ini tersimpan atau dideklarasikan didalam file header `<iostream>` sehingga kita perlu mendaftarkan file header tersebut menggunakan perintah `#include`.
- Perintah lain dalam program diatas adalah `endl` (end line atau akhir baris), yang juga tersimpan dalam namespace `std`. perintah ini digunakan untuk membuat baris baru. (memindahkan kursor ke baris baru).

3.1 Komentar Program

Dalam proses pengembangan sebuah program, tentunya kita akan disibukkan dengan penulisan kode-kode yang begitu banyak dan “tampak” rumit sehingga akan sulit difahami orang lain. Untuk menangani masalah ini, sebagai programmer kita sebaiknya menambahkan komentar untuk menjelaskan algoritma dan keterangan-keterangan yang diperlukan dalam program.

Komentar adalah bagian dari kode program yang tidak ikut dibaca pada saat proses kompilasi. Dengan kata lain, komentar tidak akan mempengaruhi jalannya program.

Dalam bahasa C++, komentar dibagi emnjadi dua jenis yaitu komentar yang hanya terdiri dari satu baris dan komentar yang terdiri dari beberapa baris.

Menggunakan Tanda //

Tanda ini digunakan untuk menulis komentar yang banyaknya hanya satu baris.

```
// Ini adalah komentar untuk satu baris
```

Berikut contoh penggunaan bentuk komentar satu baris

```
// Nama file      : I love C++.cpp
// Oleh           : Uniskalam
// Dibuat          : September 2015
// Deskripsi       : Kelengkapan mengajar pemrograman terstruktur
```

Menggunakan Tanda /* ... */

Berbeda dengan tanda //, untuk membuat komentar yang lebih dari satu baris menggunakan tanda /* sampai ditemukan tanda */

Contoh penggunaannya adalah sebagai berikut:

```
/* ini adalah komentar yang banyaknya satu baris */
/* ini adalah komentar yang panjang
   Yang banyaknya lebih dari satu baris
   Bisa panjang dan diakhiri dengan tanda seperti ini */
```

Atau sering juga seperti ini :

```
/*
*****
Nama file : I love C++.cpp
Oleh      : Uniskalam
Dibuat     : September 2015
Deskripsi  : Kelengkapan mengajar pemrograman terstruktur
*****
*/
```

3.2 Jenis Identifier

Identifier adalah suatu pengenalan atau pengidentifikasi yang kita deklarasikan agar compiler dapat mengenalinya.

Identifier dapat berupa nama variabel, konstanta, fungsi, kelas, templete, maupun namespace. Pada bagian ini kita akan membahas tentang identifier yang berperan sebagai variabel dan konstanta saja.

Identifier yang berperan sebagai variabel dan konstanta berfungsi untuk menampung sebuah nilai yang digunakan dalam program. dilakukan untuk mempermudah proses penanganan data atau nilai, misalnya untuk memasukkan dan menampilkan nilai, sebagai gambaran, dibawah ini adalah contoh program yang menggunakan dua buah identifier didalamnya.

Kode Program 2-2

```
#include <iostream>

using namespace std;

int main()
{
    char Teks[20];
    int X;
    cout<<"Masukkan sebuah kata : "; cin>>Teks;
    cout<<"Masukkan sebuah angka : "; cin>>X;
    cout<<Teks<<endl; // bisa ditulis dengan cout<<X<<'\n';
    cout<<X;

    return 0;
}
```

Pada program diatas kita mempunyai dua buah *identifier*, yaitu Teks dan X. Pada saat program dijalankan, identifier tersebut akan digunakan untuk menyimpan nilai yang dimasukkan dari keyboard.

Dalam C++, proses penyimpanan nilai seperti ini dinyatakan dengan perintah `cin` (dibaca: si in). Berbeda dengan perintah `cout` yang menggunakan operator `<<`, pada perintah `cin` operator yang digunakan adalah operator `>>`.

Dalam menentukan atau membuat *identifier* dalam program, harus memperhatikan hal-hal berikut:

- Karena bahasa C++ bersifat *case sensitive*, maka C++ akan membedakan variabel yang ditulis dengan huruf KAPITAL dan huruf kecil.

Misalnya variabel **A** akan berbeda dengan variabel **a**.

- Identifier tidak boleh berupa angka atau diawali dengan karakter yang berupa angka

Contoh:

```
long 1000; // SALAH karena identifier berupa angka
long 2X;   // SALAH karena identifier diawali oleh
           //oleh karakter berupa angka
long X2;   // BENAR karena identifier tidak diawali angka
```

- Identifier tidak boleh mengandung spasi

Contoh :

```
int Bilangan Bulat; // SALAH, karena mengandung spasi
int Bilangan_Bulat // BENAR
int BilanganBulat  // BENAR
int _BilanganBulat // BENAR
```

- Identifier tidak boleh menggunakan karakter-karakter simbol (#, @, ?, !, \$, dll)

```
long !satu; // SALAH
long dua@;  // SALAH
long ti#ga; // SALAH
```

- Identifier tidak boleh menggunakan kata kunci (keyword) yang terdapat pada C++

```
long break;    // SALAH
               // karena menggunakan kata kunci break
long return;   // SALAH
               // karena menggunakan kata kunci return
```

- Nama identifier sebaiknya disesuaikan dengan kebutuhannya, artinya jangan sampai orang lain bingung hanya karena salah dalam penamaan identifier.

Berdasarkan jenisnya identifier sendiri dibagi menjadi dua bagian yaitu konstanta dan variabel.

3.2.1 Konstanta

Konstanta adalah jenis identifier yang bersifat konstan atau tetap, artinya nilai dari konstanta didalam program tidak dapat diubah. Konstanta berguna untuk menentukan nilai yang merupakan tetapan, misalnya nilai pi (π), kecepatan cahaya dan lainnya.

Dalam bahasa C++, terdapat dua buah cara untuk membuat konstanta, yaitu:

1. Dengan menggunakan *preprocessor directive* `#define`
2. Menggunakan kata kunci `const`.

Menggunakan Preprocessor Directive #define

Kita dapat menggunakan makro untuk mendefinisikan sebuah konstanta, yaitu dengan menggunakan *preprocessor directive* `#define`. Sebagai contoh perhatikan program dibawah ini:

Kode Program 2-3

```
#include <iostream>

using namespace std;

int main() {
    int A[5];
    for (int C=0; C<5; C++) {
        // mengisi nilai ke dalam A[C]
        A[C] = C * 10;
        // menampilkan nilai A[C]
        cout<<A[C]<<endl;
    }
    return 0;
}
```

Anda bingung??? Apabila anda belum memahami maksud dari sintaks program diatas, itu ‘wajar’ dan tak perlu panik atau cemas karena itu semua akan kita pelajari dibab-bab berikutnya.

Namun, yang perlu diperhatikan disini adalah bilangan **5**, bilangan tersebut muncul dua kali dalam program. Hal ini akan merepotkan jika ternyata kita ingin melakukan perubahan terhadap bilangan tersebut misalnya menjadi **10**, jika kita menggunakan kode diatas maka kita harus merubah semua bilangan menjadi **10**, berikut program perbaikannya

Kode Program 2-4

```
#include <iostream>

using namespace std;

#define MAX 5

int main() {
    int A[MAX];

    // MAX = 10; // SALAH,
    // nilai konstanta tidak dapat ubah

    for (int C=0; C<MAX; C++) {
        // mengisi nilai ke dalam A[C]
        A[C] = C * 10;
        // menampilkan nilai A[C]
        cout<<A[C]<<endl;
    }

    return 0;
}
```

Perlu diperhatikan bahwa penggunaan #define tidak diakhiri dengan tanda titik koma (;).

Menggunakan kata kunci *const*

Bentuk umumnya adalah sebagai berikut:

```
const tipe_data nama_konstanta=nilai-tetapan;
```

contoh pendeklarasian konstanta-konstanta adalah:

```
const double PI =3.14;
const int NILAI_MAX =100;
const char MyChar ='A';
```

untuk lebih memahami, berikut sebuah kode program yang merupakan implementasi dari penggunaan kata kunci const.

Kode Program 2-5

```
#include <iostream>

using namespace std;

const int MAX = 5;

int main() {
    int A[MAX];
    for (int C=0; C<MAX; C++) {
        // mengisi nilai ke dalam A[C]
        A[C] = C * 10;
        // menampilkan nilai A[C]
        cout<<A[C]<<endl;
    }

    return 0;
}
```

Kali ini, konstanta MAX didefinisikan menggunakan kata kunci const. tidak seperti #define, penggunaan kata kunci const harus diakhiri tanda titik koma (;).

3.2.2 Variabel

Berbeda dengan konstanta yang mempunyai nilai tetap, variabel adalah sebuah identifier yang mempunyai nilai dinamis. Arti kata 'dinamis' disini bermaksud bahwa nilai variabel tersebut dapat kita ubah sesuai kebutuhan dalam program.

Berikut bentuk umum pendeklarasian variabel dalam C++.

```
tipe_data nama_variabel;
```

Contoh:

```
int A;
```

Pada contoh diatas, kita mendeklarasikan sebuah variabel bertipe `int` dengan nama `A`, melalui cara seperti ini, variabel tersebut sudah dapat digunakan untuk menampung nilai-nilai berupa bilangan bulat (tipe data `int`)

Apabila kita mendeklarasikan beberapa variabel dengan bertipe sama, maka kita dapat menyingkat penulisan dengan bentuk:

```
tipe_data nama_variabel1, nama_variabel2, nama_variabel3;
```

Contoh:

```
int A, B, C;
```

Kali ini kita mendeklarasikan tiga buah variabel `int`, yaitu `A`, `B` dan `C`.

Inisialisasi Variabel

Dalam konteks ini, inisialisasi dapat didefinisikan sebagai proses pengisian nilai awal (nilai default) kedalam suatu variabel. Dalam C++, pengisian nilai dilakukan dengan menggunakan operator sama dengan (`=`).

Bentuk umum yang digunakan untuk melakukan inisialiasasi variabel adalah sebagai berikut:

```
tipe_data nama_variabel=nilai awal;
```

atau

```
tipe_data nama_variabel1=nilai awal1, nama_variabel2=nilai_awal2, ....;
```

Contoh:

```
int A=9;
```

Pada contoh diatas, kita melakukan inisialisasi terhadap variabel A dengan nilai 9, apabila kita menginisialisasi lebih dari satu variabel, maka sintaksnya dirubah seperti ini:

```
int A=10, B=15, C=25;
```

Atau bisa juga seperti berikut:

```
int A, B=15, C;
```

Untuk memahami konsep inisialisasi variabel, perhatikan contoh program berikut

Kode Program 2-6

```
#include <iostream>

using namespace std;

int main() {
    int X;

    // Menampilkan nilai X sebelum
    // dilakukan pengisian nilai (assignment)
    cout<<"Nilai X sebelum assignment: "<<X<<endl;

    // mengisi nilai 10 ke dalam variabel X
    X = 10;
    // Menampilkan nilai X setelah
    // dilakukan pengisian nilai (assignment)
    cout<<"Nilai X sesudah assignment: "<<X<<endl;

    return 0;
}
```

Hasil yang diberikan oleh program diatas adalah sebagai berikut:

```
Nilai X sebelum assignment: 0
Nilai X setelah assignment: 10
```

Kode Program 2-7

```
#include <iostream>

using namespace std;

int main() {
    // Inisialisasi X dengan nilai 5
    int X = 5;

    // Menampilkan nilai X sebelum
    // dilakukan pengisian nilai (assignment)
    cout<<"Nilai X sebelum assignment: "<<X<<endl;

    // mengisi nilai 10 ke dalam variabel X
    X = 10;
    // Menampilkan nilai X setelah
    // dilakukan pengisian nilai (assignment)
    cout<<"Nilai X sesudah assignment: "<<X<<endl;

    return 0;
}
```

Kali ini, hasil yang diberikan oleh program diatas adalah:

```
Nilai X sebelum assignment: 5
Nilai X setelah assignment: 10
```

Variabel Global vs Variabel Lokal

Berdasarkan ruang lingkupnya, variabel dibedakan menjadi dua: global dan lokal. Penentuan variabel untuk dijadikan sebagai variabel global atau lokal tergantung dari kasus program yang dihadapi.

Variabel Global

Kita telah mengetahui bahwa dalam bahasa C++ selalu terdapat fungsi utama main(). Apabila kita mendeklarasikan sebuah variabel diluar fungsi main() (atau fungsi lain), maka dengan sendirinya compiler akan menganggap variabel tersebut sebagai variabel global.

Kode Program 2-8

```
#include <iostream>

using namespace std;

int A; // Variabel A adalah variabel global
      // karena dideklarasikan di luar fungsi

// Membuat fungsi test()
void test() {
    // Mengisikan (assign) nilai ke dalam variabel A
    A = 20;
    cout<<"Nilai A di dalam fungsi test(): "<<A<<endl;
}

// Membuat fungsi main() atau fungsi utama
int main() {
    // Mengisikan (assign) nilai ke dalam variabel A
    A = 10;
    cout<<"Nilai A di dalam fungsi main(): "<<A<<endl;

    // Memanggil fungsi test()
    test();

    return 0;
}
```

Hasil dari program diatas adalah sebagai berikut:

```
Nilai A di dalam fungsi main(): 10
Nilai A di dalam fungsi test(): 20
```

Variabel Lokal

Berebeda dengan variabel global, variabel lokal adalah variabel yang hanya dikenal oleh suatu fungsi saja. Proses deklarasi variabel lokal dilakukan didalam lingkup fungsi yang dimaksud.

Kode Program 2-9

```
#include <iostream>

using namespace std;

// Membuat fungsi test()
void test() {
    int A;    // A bersifat lokal
              // dan hanya dikenal oleh fungsi test()
    A = 20;
    cout<<"Nilai A di dalam fungsi test(): "<<A<<endl;
}

// Membuat fungsi main() atau fungsi utama
int main() {
    // A tidak dapat digunakan oleh fungsi main()
    // A = 10;    // SALAH
    // cout<<"Nilai A di dalam fungsi main(): "
    //         <<A<<endl; // SALAH

    // Memanggil fungsi test()
    test();

    return 0;
}
```

Hasil yang akan diberikan oleh program diatas adalah sebagai berikut:

```
Nilai A di dalam fungsi test(): 20
```

3.3 Tipe Data

Tipe data berfungsi untuk mempresentasikan jenis dari nilai yang terdapat dalam program. sebagai contoh kita mempunyai suatu data dengan nilai **2**, maka **2** termasuk ke dalam tipe bilangan bulat. Begitu juga dengan data seperti ini “**Rahasia C++**”, maka data tersebut dikategorikan kedalam tipe teks (dalam bahasa pemrograman teks disebut *string*).

Dalam bahasa C++, tipe data dapat dikelompokkan menjadi dua, yaitu:

1. Tipe Data Dasar
2. Tipe Data Bentukan

3.3.1 Tipe Data Dasar

Dalam bahasa C++ terdapat beberapa tipe data dasar yang telah didefinisikan, yaitu yang digolongkan kedalam:

- Tipe bilangan bulat (*integer*),
- Tipe bilangan riil (*floating-point*)
- Tipe logika (*boolean*)
- Tipe karakter

Tipe Bilangan Bulat (Integer)

Digunakan untuk mempresentasikan data numerik yang berupa bilangan bulat, yaitu bilangan yang tidak mengandung angka dibelakang koma.

Contoh bilangan bulat adalah 20, 45,123 dan sebagainya atau dapat dilihat pada tabel dibawah ini:

| Tipe Data | Ukuran (dalam bit) | Rentang |
|--------------------|-----------------------|--|
| int | 16 atau 32 | -32.768 sampai 32.767 atau -2.147.483.648 sampai 2.147.483.647 |
| unsigned int | 16 atau 32 | 0 sampai 65.535 atau 0 sampai 4.294.967.295 |
| signed int | 16 atau 32 | Sama seperti int |
| short int | 16 | -32.768 sampai 32.767 |
| unsigned short int | 16 | 0 sampai 65.535 |
| signed short int | 16 | Sama seperti short int |
| long int | 32 | -2.147.483.648 sampai 2.147.483.647 |
| signed long int | 32 | Sama seperti long int |
| unsigned long int | 32 | 0 sampai 4.294.967.295 |

Contoh program yang menggunakan tipe bilangan bulat.

Kode Program 2-10

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel x dengan tipe data int
    int x;

    // Melakukan assigment terhadap variabel x
    x = 3;
    cout<<"Nilai x: "<<x;

    return 0;
}
```

Hasil yang diperoleh

Nilai x: 3

Tipe Bilangan Riil

Tipe yang mempresentasikan data-data bilangan yang menagndung angka dibelakang koma, misalnya 7.66,36.25 atau perhatikan tabel berikut:

| Tipe Data | Ukuran (dalam bit) | Rentang | Presisi |
|-------------|-----------------------|----------------------------|----------|
| float | 32 | 1.2E-38 sampai 3.4E+38 | 6 digit |
| double | 64 | 2.3E-308 sampai 1.7E+308 | 16 digit |
| long double | 80 | 3.4E-4932 sampai 1.1E+4932 | 19 digit |

Berikut contoh programnya

Kode Program 2-11

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel y
    // dengan tipe data double
    double y;

    // Melakukan assignment terhadap variabel y
    y = 222.134;
    cout<<"Nilai y: "<<y;

    return 0;
}
```

Hasil yang diperoleh

```
Nilai y: 222.134
```

Tipe Logika

Mempresentasikan data-data yang hanya mengandung dua buah nilai yaitu nilai logika (boolean) terdiri dari nilai benar (biasanya 1) dan salah (biasanya 0). Dalam bahasa C++ standar, tipe ini telah dinyatakan dalam tipe `bool`, yang nilainya dapat berupa `true` (benar) atau `false` (salah).

Kode Program 2-12

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel y
    // dengan tipe data bool
    bool benar;

    // Meminta input dari user
    int a, b;
    cout<<"Masukkan nilai a: "; cin>>a;
    cout<<"Masukkan nilai b: "; cin>>b;

    // Melakukan assignment terhadap variabel benar
    benar = a > b;

    if (benar == true) {    // bisa ditulis: if (benar) {
        cout<<"a lebih besar dari b";
    } else {
        cout<<"a lebih kecil dari atau sama dengan b";
    }

    return 0;
}
```

Hasil yang diperoleh

```
Masukan nilai a: 19
Masukan nilai b: 5
A lebih besar dari b
```

Tipe Karakter

Tipe ini digunakan untuk mempresentasikan data bertipe karakter seperti 'A' , 'a' , '9' , '&' dan sebagainya. Dalam bahasa C++, tipe karakter dinyatakan dalam bentuk char. Untuk karakter Unicode (wide character) dinyatakan dalam wchar_t .

Tabel berikut menunjukkan ukuran tipe data karakter:

| Tipe Data | Ukuran (dalam bit) | Rentang |
|---------------|--------------------|-----------------------------------|
| char | 8 | -128 sampai 127 atau 0 sampai 255 |
| Unsigned char | 8 | 0 sampai 255 |
| Signed char | 8 | 128 sampai 127 |
| Wchart_t | 16 atau 32 | 0 sampai 65.535 |

Dalam Bahasa C++, tipe karakter selalu diapit oleh petik tunggal. Satu karakter yang diapitg oleh tanda petik ganda akan dianggap sebagai string .

Contoh: 'A' adalah karakter; sedangkan "A" adalah string yang terdiri dari satu karakter.

Kode Program 2-13

```
#include <iostream>

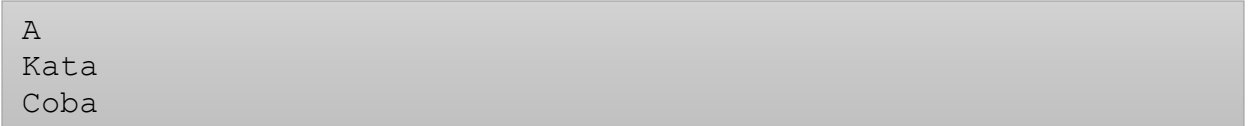
using namespace std;

int main() {
    // Mendeklarasikan variabel Karakter
    // dan mengisinya dengan nilai 'A'
    char Karakter = 'A';

    // Mendeklarasikan variabel Teks
    // dan mengisinya dengan nilai "Kata"
    char* Teks = (char*) "Kata";
    char TEKS[10] = "Coba";
    cout<<Karakter<<endl;
    cout<<Teks<<endl;
    cout<<TEKS<<endl;

    return 0;
}
```

Hasil yang diperoleh



3.3.2 Tipe Data Bentukan

Adalah tipe data yang dibuat sendiri sesuai kebutuhan dalam program yang kita buat. Tipe ini lebih dikenal dengan sebutan *user defined types*.

Adapun yang termasuk kedalam tipe bentukan adalah array (larik), struktur dan enumerasi, termasuk juga string. Atau dalam bahasa Pascal dikenal dengan istilah rekaman (*record*).

Tipe Struktur

Adalah tipe data bentukan yang menyimpan lebih dari satu variabel bertipe sama maupun berbeda. Untuk membuat tipe data struktur dalam bahasa C++, kita harus mendeklarasikannya dengan menggunakan kata kunci struct.

Berikut bentuk umum pendeklarasian tipe data struktur didalam C++

```
Struct nama_struktur {  
    tipe_data variabel1;  
    tipe_data variabel2;  
    ...  
};
```

Ingat, dalam pendeklarasian struktur kita harus selalu mengakhirinya dengan tanda titik koma atau semicolon (;).

Kode Program 2-14

```
#include <iostream>  
#include <cstring>  
  
using namespace std;  
  
int main() {  
  
    struct SISWA {  
        char NIS[9];  
        char Nama[25];  
        char Alamat[20];  
        char Kota[15];  
    };  
  
    SISWA A;           // Mendeklarasikan variabel A  
                        // yang bertipe SISWA  
  
    strcpy(A.NIS, "10299999");  
    strcpy(A.Nama, "Nur Alamsyah");  
    strcpy(A.Alamat, "Kayutangi");  
    strcpy(A.Kota, "Banjarmasin");  
  
    // Menampilkan nilai yang diisikan ke layar  
    cout<<A.NIS<<endl;  
    cout<<A.Nama<<endl;  
    cout<<A.Alamat<<endl;  
    cout<<A.Kota<<endl;  
  
    return 0;  
}
```

Hasil yang diperoleh

```
10299999  
Nur Alamsyah  
Kayutangi  
Banjarmasin
```

Dalam bahasa C++, pendeklarasian variabel yang bertipe struktur dapat dituliskan langsung pada saat pendefinisian tipe struktur yang bersangkutan. Misalnya kita akan mendeklarasikan variabel A, B, dan C yang bertipe SISWA, maka penulisannya dapat dilakukan seperti dibawah ini:

```
// Membuat tipe struktur dengan nama SISWA
// dan menggunakan langsung
// mendeklarasikan variabel A, B dan C

struct SISWA {
    char NIS [9];
    char Nama [25];
    char Alamat [20];
    char Kota [5];
} A, B, C;
```

Tipe Array

Tipe ini digunakan untuk mempresentasikan kumpulan data sejenis atau yang bertipe sama. Banyaknya data dengan menyebutkan nilai didalam bracket ([]).

Contoh:

```
int nomorhari[7];
char nama[25];
```

Tipe Enumerasi

Adalah tipe data yang nilainya terbatas pada nilai-nilai yang telah didefinisikan saja. Membentuk tipe data yang nilainya bersifat pasti. Misalnya mendefinisikan tipe jenis kelamin, nama hari, warna primer dan lain sebagainya. Kata kunci nya enum.

Berikut adalah bentuk umum untuk mendefinisikan tipe enumerasi

```
enum nama_tipe {nilai_1, nilai_2, ...};
```

Contoh:

```
enum JENIS_KELAMIN {Pria, Wanita};
enum HARI {Minggu, Senin, Selasa, Rabu, Kamis, Jumat, Sabtu};
enum WARNA_PRIMER {Merah, Biru, Hijau};
```

Kode Program 2-15

```
#include <iostream>
#include <cstring>

using namespace std;

// Mendefinisikan tipe JENIS_KELAMIN
enum JENIS_KELAMIN { Pria, Wanita };

int main() {

    struct SISWA {
        char NIS[9];
        char Nama[25];
        JENIS_KELAMIN gender;
    } A;

    strcpy(A.NIS, "10899999");
    strcpy(A.Nama, "Dewi Aulia");
    A.gender = Wanita;

    cout<<"NIS      : "<<A.NIS<<endl;
    cout<<"Nama      : "<<A.Nama<<endl;
    cout<<"Gender    : "<<A.gender<<endl;

    return 0;
}
```

Hasil yang diperoleh

```
NIS      : 10899999
Nama      : Dewi Aulia
Gender    : 1
```

Tipe String

Tipe ini digunakan untuk mempresentasikan data yang berupa teks (kumpulan Karakter). Contoh data string adalah “Uniska”, “C++” dan sebagainya.

Contoh:

```
char namadepan[15];    // string yang berupa array
char *namabelakang;    // string yang berupa pointer
```

BAB IV
OPERATOR

Operator adalah tanda yang digunakan untuk melakukan operasi-operasi tertentu didalam program. dengan operator, kita dapat melakukan operasi perhitungan, perbandingan, manipulasi bit, dan lain-lain.

C++ merupakan salah satu bahasa pemrograman yang banyak menyediakan operator, dengan klasifikasi menjadi empat kelompok:

- 1. Operator assignment
- 2. Operator unary
- 3. Operator binary
- 4. Operator ternary

Misal ada statemen seperti berikut:

```
c = 5 + 7
```

Maka:

c disebut variabel
= disebut operator assignment
5 dan 7 disebut operand
5 + 7 disebut ekspresi
+ disebut operator aritmatika (penambahan)
c = 5 + 7 disebut statemen aritmatika.

4.1 Jenis-jenis Operator dalam bahasa C++

C++ mendukung tiga jenis operator, yaitu:

| Jenis Operator | Pengertian | Contoh |
|----------------|---|--|
| Unary | Digunakan dalam operasi yang hanya melibatkan satu buah operand | x++; a = -b; |
| Binary | Digunakan dalam operasi yang melibatkan dua buah operand | x = y + z; a = 2 * 0; |
| Ternary | Digunakan dalam operasi yang melibatkan tiga buah operand | x = (x > 0) ? x : -x; maks=(maks <=a)? a: maks; |

4.2 Operator Assignment

Operator assignment (pengisian) adalah operator yang digunakan untuk memasukkan atau mengisikan nilai kedalam suatu variabel. Dalam C++, operator yang digunakan untuk keperluan ini adalah operator = (sama dengan).

```
int a, b;  
a = 88;  
b = 99;
```

Memasukan nilai 88 kedalam variabel a dan 99 kedalam variabel b .

Operator = dapat digunakan untuk mengisi nilai dari berbagai macam tipe data, bisa bilangan (bulat dan riil), karakter, boolean, string, maupun tipe data bentukan lainnya.

Contoh:

```
// deklarasi variabel
int i;
double d;
char c;
char *s;

// mengisi nilai ke dalam variabel
i = 88;
d = 4.675;
c = 'c';
s = (char *) "Contoh string";
```

Berikut contoh programnya:

Kode Program 3-1

```
#include <iostream>

using namespace std;

int main()
{
    // deklarasi variabel
    int i;
    double d;
    char c;
    char *s;

    // mengisi nilai ke dalam variabel
    i = 88;
    d = 4.675;
    c = 'C';
    s = (char *) "Contoh string";

    // menampilkan nilai variabel
    cout<<"Nilai i: "<<i<<endl;
    cout<<"Nilai d: "<<d<<endl;
    cout<<"Nilai c: "<<c<<endl;
    cout<<"Nilai s: "<<s<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
nilai i: 88
nilai d: 4.675
nilai c: C
nilai s: Contoh string
```


4.3 Operator Unary

Dalam ilmu matematika yang disebut dengan operator Unary adalah operator yang hanya melibatkan sebuah operand.

| Operator | Jenis Operasi | Contoh |
|----------|-----------------------|--------|
| + | Membuat nilai positif | +7 |
| - | Membuat nilai negatif | -7 |
| ++ | Increment | C++ |
| -- | Decrement | C-- |

Berikut contoh programnya menggunakan operator plus (+) dan minus (-).

Kode Program 3-2

```
#include <iostream>

using namespace std;

int main() {
    int X;
    float Y;

    X = +5; // Dapat ditulis dengan X = 5,
           // yang berarti memasukkan nilai positif 5
    Y = -2.12; // Memasukkan nilai negatif 2.12

    // Menampilkan nilai
    // yang disimpan dalam variabel X dan Y
    cout<<"Nilai X : "<<X<<endl;
    cout<<"Nilai Y : "<<Y<<endl;

    X = -X; // Mengubah nilai X menjadi negatif
    Y = -Y;

    // Menampilkan kembali nilai yang disimpan
    // dalam variabel X dan Y
    cout<<"Nilai X : "<<X<<endl;
    cout<<"Nilai Y : "<<Y<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
nilai x: 5
nilai y: -2.12

nilai x: -5
nilai y: 2.12
```

Increment

Adalah suatu penambahan nilai yang terjadi pada sebuah variabel. Operator yang digunakan adalah operator ++. Operator ini akan menambahkan nilai dari suatu variabel dengan nilai 1.

Kode Program 3-3

```
#include <iostream>

using namespace std;

int main() {

    int C;    // Mendeklarsikan variabel C

    // Mengisikan nilai ke dalam variabel C
    // dengan nilai 5
    C = 5;

    // Melakukan pre-increment
    cout<<"Nilai C awal    : "<<C<<endl;
    cout<<"Nilai ++C    : "<<++C<<endl;
    cout<<"Nilai C akhir  : "<<C<<endl;
    cout<<'\n';

    // Mengubah nilai yang terdapat dalam variabel C
    // dengan nilai 10
    C = 10;

    // Melakukan post-increment
    cout<<"Nilai C awal    : "<<C<<endl;
    cout<<"Nilai C++   : "<<C++<<endl;
    cout<<"Nilai C akhir  : "<<C<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
nilai C awal      : 5
nilai ++ C        : 6
nilai C akhir     : 6

nilai C awal      : 10
nilai C ++        : 10
nilai C akhir     : 11
```

Decrement

Merupakan kebalikan dari proses increment, yaitu menurunkan (mengurangi) nilai dari suatu variabel.

Kode Program 3-4

```
#include <iostream>

using namespace std;

int main() {
    int C;    // Mendeklarsikan variabel C

    // Mengisikan nilai ke dalam variabel C
    // dengan nilai 5
    C = 5;

    // Melakukan pre-increment
    cout<<"Nilai C awal    : "<<C<<endl;
    cout<<"Nilai --C    : "<<--C<<endl;
    cout<<"Nilai C akhir   : "<<C<<endl;
    cout<<'\n';

    // Mengubah nilai yang terdapat dalam variabel C
    // dengan nilai 10
    C = 10;

    // Melakukan post-decrement
    cout<<"Nilai C awal    : "<<C<<endl;
    cout<<"Nilai C--    : "<<C--<<endl;
    cout<<"Nilai C akhir   : "<<C<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
nilai C awal      : 5
nilai -- C        : 6
nilai C akhir     : 6

nilai C awal      : 10
nilai C --        : 10
nilai C akhir     : 11
```

4.4 Operator Binary

Operator binary adalah operator yang digunakan dalam operasi yang melibatkan dua buah operand. Dalam bahasa C++, operator binary ini dikelompokkan menjadi 4 jenis, yaitu:

- 1. Operator Aritmatika,
- 2. Operator Logika,
- 3. Operator Relasional dan
- 4. Operator Bitwise.

1. Operator Aritmetika

Operator aritmetika adalah operator yang digunakan untuk melakukan operasi-operasi aritmetika seperti penjumlahan, pengurangan, dan sebagainya.

| Operator | Jenis Operasi | Contoh |
|----------|---------------------|---------------------|
| + | Penjumlahan | $2 + 3 = 5$ |
| - | Pengurangan | $5 - 3 = 2$ |
| * | Perkalian | $2 * 3 = 6$ |
| / | Pembagian | $10.0/3.0 = 3.3333$ |
| % | Sisa bagi (modulus) | $10\%3 = 1$ |

- Menggunakan Operator Plus (+)

Kode Program 3-5

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel X (diisi nilai 10)
    // dan Y (diisi nilai 3)
    int X = 10, Y = 3;
    // Mendeklarasikan variabel Z sebagai penampung
    // nilai hasil operasi
    int Z;

    // Melakukan operasi penjumlahan
    Z = X + Y;

    // Menampilkan hasil penjumlahan
    cout<<X<<" + "<<Y<<" = "<<Z;

    return 0;
}
```

Hasil yang diperoleh:

```
10 + 3 = 13
```

- *Menggunakan Operator Minus (-)*

Kode Program 3-6

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel X (diisi nilai 10)
    // dan Y (diisi nilai 3)
    int X = 10, Y = 3;
    // Mendeklarasikan variabel Z sebagai penampung
    // nilai hasil operasi
    int Z;

    // Melakukan operasi pengurangan
    Z = X - Y;

    // Menampilkan hasil pengurangan
    cout<<X<<" - "<<Y<<" = "<<Z;

    return 0;
}
```

Hasil yang diperoleh:

```
10 - 3 = 7
```

- *Menggunakan Operator **

Kode Program 3-7

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel X (diisi nilai 10)
    // dan Y (diisi nilai 3)
    int X = 10, Y = 3;
    // Mendeklarasikan variabel Z sebagai penampung
    // nilai hasil operasi
    int Z;

    // Melakukan operasi perkalian
    Z = X * Y;

    // Menampilkan hasil perkalian
    cout<<X<<" * "<<Y<<" = "<<Z;

    return 0;
}
```

Hasil yang diperoleh:

```
10 * 3 = 30
```

- *Menggunakan Operator /*

Kode Program 3-8

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel X, Y, dan Z
    // bertipe int
    int X = 10, Y = 3;
    int Z;

    // Mendeklarasikan variabel A, B, dan C
    // bertipe float
    float A = 10.0, B = 3.0;
    float C;

    // Melakukan operasi pembagian
    // pada bilangan bulat
    Z = X / Y;

    // Melakukan operasi pembagian
    // pada bilangan riil (floating-point)
    C = A / B;

    // Menampilkan hasil pembagian
    cout<<X<<" / "<<Y<<" = "<<Z<<endl;
    cout<<A<<" / "<<B<<" = "<<C;

    return 0;
}
```

Hasil yang diperoleh:

```
10 / 3 = 3
```

- *Menggunakan Operator %*

Kode Program 3-9

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan variabel X (diisi nilai 10)
    // dan Y (diisi nilai 3)
    int X = 10, Y = 3;

    int Z;    // Mendeklarasikan variabel Z sebagai
              // penampung nilai hasil operasi

    // Melakukan operasi pembagian dan menyimpan
    // sisa baginya ke dalam Z
    Z = X % Y;

    // Menampilkan sisa bagi
    // dari operasi pembagian X / Y
    cout<<X<<" % "<<Y<<" = "<<Z;

    return 0;
}
```

Hasil yang diperoleh:

```
10 % 3 = 1
```


2. Operator Logika

Operator Logika adalah operator yang digunakan untuk melakukan operasi dimana nilai yang dihasilkan dari operasi tersebut hanya berupa nilai benar (true) dan salah (false). Nilai ini disebut dengan nilai boolean. Boolean sendiri ditemukan oleh seorang matematikawan Inggris yang bernama George Bool.

Dalam bahasa C++, nilai benar dipresentasikan dengan bilangan selain 0 (biasanya nilai 1). Sedangkan nilai salah dipresentasikan dengan nilai 0.
Tapi dalam bahasa C++ modern yang telah mendukung tipe bool, nilai benar dipresentasikan dengan nilai true dan salah dengan nilai false.

Adapun yang termasuk kedalam operator logika dalam C++ adalah seperti berikut:

| Operator | Jenis Operasi | Contoh |
|----------|---------------|------------|
| && | AND (dan) | 1 && 1 = 1 |
| | OR (atau) | 1 0 = 1 |
| ! | NOT (negasi) | ! 0 = 1 |

• Operator && (AND)

Operasi AND hanya akan menghasilkan nilai 1 (benar) jika semua operandnya bernilai benar. Namun jika tidak, maka operasi tersebut akan menghasilkan nilai 0 (salah).
Berikut tabel yang menunjukkan hasil dari operasi AND.

| X | Y | X&&Y |
|---|---|------|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 1 | 0 |

Sebagai bukti dari pernyataan tersebut, perhatikan program berikut:

Kode Program 3-10

```
#include <iostream>

using namespace std;

int main() {
    cout<<"1 && 1 = "<<(1 && 1)<<endl;
    cout<<"1 && 0 = "<<(1 && 0)<<endl;
    cout<<"0 && 0 = "<<(0 && 0)<<endl;
    cout<<"0 && 1 = "<<(0 && 1)<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
1 && 1 = 1
1 && 0 = 0
0 && 0 = 0
0 && 1 = 0
```

Apabila kita rubah ke tipe bool, bisa seperti berikut ini:

Kode Program 3-11

```
#include <iostream>

using namespace std;

int main() {
    // mendeklarasikan variabel boolean
    bool benar=true, salah=false;

    cout<<"true && true   = "
        <<(benar && benar)<<endl;
    cout<<"true && false   = "
        <<(benar && salah)<<endl;
    cout<<"false && false   = "
        <<(salah && salah)<<endl;
    cout<<"false && true    = "
        <<(salah && benar)<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
true && true      = 1
true && false      = 0
false && false     = 0
false && true      = 0
```

• Operator || (OR)

Operator OR hanya akan menghasilkan nilai 0 (salah) jika semua operator nya bernilai salah, namun jika tidak, maka operator tersebut akan menghasilkan nilai 1 (benar).

Berikut Tabel yang menunjukkan hasil dari operasi OR

| X | Y | X Y |
|---|---|--------|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |

Berikut Program yang akan membuktikan

Kode Program 3-12

```
#include <iostream>

using namespace std;

int main() {
    cout<<"1 || 1 = "<<(1 || 1)<<endl;
    cout<<"1 || 0 = "<<(1 || 0)<<endl;
    cout<<"0 || 0 = "<<(0 || 0)<<endl;
    cout<<"0 || 1 = "<<(0 || 1)<<endl;

    return 0;
}
```


Hasil yang diperoleh:

```
1 | 1 = 1
1 | 0 = 0
0 | 0 = 0
0 | 1 = 0
```

• **Operator ! (NOT)**

Nilai yang dihasilkan dari operator NOT adalah kebalikan dari nilai yang dikandung didalamnya.

| X | !X |
|---|----|
| 1 | 0 |
| 0 | 1 |

Berikut programnya

Kode Program 3-13

```
#include <iostream>

using namespace std;

int main() {
    cout<<"!1 = "<<!1<<endl;
    cout<<"!0 = "<<!0<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
!1 = 0
!0 = 1
```

3. **Operator Relasional**

Operator Relasional adalah operator yang digunakan untuk menentukan relasi atau hubungan dari dua buah operand. Operator ini ditempatkan didalam sebuah ekspresi, yang kemudian akan menentukan benar atau tidaknya sebuah eksperesi.

| Operator | Jenis Operasi | Contoh |
|----------|------------------------------|--------------|
| > | Lebih besar | (5 > 2) = 1 |
| < | Lebih kecil | (5 < 2) = 0 |
| >= | Lebih besar atau sama dengan | (5 >= 5) = 1 |
| <= | Lebih kecil atau sama dengan | (5<= 2) = 0 |
| == | Sama dengan | (5 == 2) = 0 |
| != | Tidak sama dengan | (5 != 2) = 1 |

4. Operator Bitwise

Tidak seperti kebanyakan bahasa pemrograman lainnya. C/C++ mendukung penuh operator-operator bitwise.

Operator bitwise berguna untuk melakukan operasi-operasi yang berhubungan dengan manipulasi Bit. Hanya bisa dilakukan pada operand yang bertipe char dan int saja karena ini berkoresponden dengan tipe byte atau word didalam bit.

| Operator | Jenis Operasi | Contoh |
|----------|--------------------|-------------|
| & | AND | 1 & 0 = 0 |
| | OR | 1 0 = 1 |
| ^ | Exclusive OR (XOR) | 1 ^ 1 = 0 |
| ~ | NOT | ~ 1 = 0 |
| >> | Shift Right | 5 << 1 = 10 |
| << | Shift Left | 10 >> = 5 |

Fungsi dari operator &, | dan ~ diatas adalah sama dengan operator &&, || dan ! pada operator logika. Bedanya operator bitwise ini bekerja bit demi bit. Sedangkan operator logika bekerja untuk setiap nilai.

4.5 Operator Ternary

Operator Ternary adalah operator yang digunakan dalam operasi yang melibatkan tiga buah operand. Adapun operator yang digunakan untuk menyatakannya adalah operator ?:. konsep yang mendasari operasi ini adalah suatu percabangan (pemilihan) yang didasarkan atas kondisi tertentu.

Bentuk umum dari penggunaan operator ternary:

```
Ekspresi 1 ? Ekspresi2 : Ekspresi3;
```

Jika Ekspresi1 bernilai benar, maka prograsm akan mengeksekusi Ekspresi2. Sedangkan jika Ekspresi1 bernilai salah maka yang dieksekusi adalah Ekspresi3.

Berikut contoh programnya:

Kode Program 3-14

```
#include <iostream>

using namespace std;

int main() {

    int X;

    // Meminta user untuk memasukkan nilai X
    // dari keyboard
    cout<<"Masukkan nilai X : "; cin>>X;
    cout<<"\n";

    // Melakukan pemeriksaan terhadap nilai X
    X = (X < 0) ? -X : X;

    // Menampilkan nilai X
    // setelah proses pemeriksaan
    cout<<"| X | = "<<X;

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan nilai x : -15
| X | = 15
```

BAB V PERCABANGAN

Salah satu permasalahan yang pasti akan dijumpai dalam pembuatan program adalah suatu percabangan. Percabangan adalah suatu pemilihan statemen yang akan dieksekusi dimana pemilihan tersebut berdasarkan atas kondisi tertentu.

Dalam Bahasa C++, terdapat dua buah jenis struktur (blok program) yang digunakan untuk mengimplentasikan suatu percabangan, yaitu dengan menggunakan struktur `if` dan struktur `switch`.

Statemen- statemen yang terdapat dalam sebuah blok percabangan akan dieksekusi hanya jika kondisi yang didefinisikan terpenuhi (bernilai benar). Artinya jika kondisi tidak terpenuhi (bernilai salah), maka statemen-statemen tersebut juga tidak ikut dieksekusi atau dengan kata lain akan diabaikan oleh compiler.

Untuk memahami konsep percabangan, perhatikan kalimat dibawah ini:

“Jika Hanan lulus ujian, maka Hanan akan dibelikan sepeda motor oleh ayahnya.”

Coba anda amati, pada kalimat diatas yang merupakan kondisi adalah lulus ujian. Pada kasus ini sepeda motor hanya akan dibeli jika Hanan lulus ujian. Sebaliknya, jika tidak lulus, maka sepeda motor pun tidak akan dibeli.

Suatu percabangan yang dibuat menggunakan statemen `if` dapat terdiri dari satu kondisi maupun lebih.

5.1 Struktur `if` Satu Kondisi

Struktur ini merupakan struktur yang paling sederhana karena hanya melibatkan satu buah ekspresi yang akan diperiksa.

Bentuk umumnya dari struktur percabangan yang memiliki satu kondisi adalah sebagai berikut:

// Jika terdapat lebih dari satu statemen

```
If (kondisi) {  
    Statemen1;  
    Statemen2;  
    ...  
}
```

// Jika hanya terdapat satu statemen, dapat ditulis seperti dibawah ini

```
If (kondisi) Statemen;
```

Sebagai contoh untuk menerapkan konsep yang terdapat pada bagian ini, berikut ini beberapa contoh program.

Kode Program 5-1

```
#include <iostream>

using namespace std;

int main() {
    int nilai;

    // Memberikan informasi agar user memasukkan
    // sebuah bilangan bulat
    cout<<"Masukkan sebuah bilangan bulat: ";

    // Membaca nilai yang dimasukkan dari keyboard
    // dan disimpan ke variabel nilai
    cin>>nilai;

    // Menampilkan sebuah teks
    // jika nilai yang tersimpan
    // lebih besar dari nol
    if (nilai > 0)
        cout<<"Nilai yang Anda masukkan "
            <<"adalah bilangan positif";

    return 0;
}
```

Kita juga bisa menggunakan operator `| |` dan `&&` dalam menentukan sebuah ekspresi. Contoh programnya adalah sebagai berikut:

Kode Program 5-2

```
#include <iostream>

using namespace std;

int main() {
    int bilangan;
    char huruf;

    // Meminta user untuk memasukkan sebuah bilangan
    cout<<"Masukkan sebuah bilangan bulat: ";
    cin>>bilangan;

    if ((bilangan > 0) && (bilangan < 10))
        cout<<bilangan
            <<" lebih besar dari nol"
            <<" dan lebih kecil dari sepuluh";

    // Meminta user untuk memasukkan sebuah huruf
    cout<<endl<<endl;
    cout<<"Masukkan sebuah huruf: ";
    cin>>huruf;

    if ((huruf == 'A') || (huruf == 'a') ||
        (huruf == 'I') || (huruf == 'i') ||
        (huruf == 'U') || (huruf == 'u') ||
        (huruf == 'E') || (huruf == 'e') ||
        (huruf == 'O') || (huruf == 'o')) {
        cout<<huruf
            <<" adalah salah satu huruf vokal";
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan sebuah bilangan bulat : 6
6 lebih besar dari nol dan lebih kecil dari sepuluh
Masukkan sebuah huruf: A
A adalah salah satu huruf vokal
```

5.2 Struktur `if` Dua Kondisi

Struktur percabangan jenis ini sedikit lebih kompleks bila dibandingkan dengan struktur yang hanya memiliki satu buah kondisi.

Bentuk umum dari struktur percabangan dua kondisi adalah sebagai berikut:

```
If(kondisi) {
    Statemen_jika_kondisi_terpenuhi;
} else{
    Statemen_Jika_kondisi_tidak_terpenuhi;
}
```

Contoh programnya:

Kode Program 5-3

```
#include <iostream>

using namespace std;

int main() {
    int bilangan;

    cout<<"Masukkan bilangan bulat "
    <<"yang akan diperiksa: ";
    cin>>bilangan;

    // Melakukan pengecekan bilangan apakah habis
    // dibagi dua atau tidak
    if (bilangan % 2 == 0) {
        cout<<bilangan<<" adalah bilangan genap";
    } else {
        cout<<bilangan<<" adalah bilangan ganjil";
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan bilangan bulat yang akan diperiksa: 11
11 adalah bilangan ganjil
```

Contoh lainnya:

```

#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    double X, Y, Z;

    // Meminta user untuk memasukkan bilangan
    cout<<"Masukkan bilangan yang akan dibagi : ";
    cin>>X;
    cout<<"Masukkan bilangan pembagi          : ";
    cin>>Y;

    // Menghindari terjadinya pembagian dengan 0
    if (Y == 0) {
        cout<<"Kesalahan: "
            <<"Bilangan pembagi tidak boleh NOL";
    } else {
        // Melakukan pembagian bilangan X dengan Y
        Z = X/Y;

        // Menentukan presisi dari hasil bagi
        // dengan nilai dua angka
        // di belakang koma
        cout.setf(ios::fixed, ios::floatfield);
        cout.precision(2);

        // Menampilkan hasil bagi
        cout<<"Hasil bagi = "<<Z;
    }

    return 0;
}

```

Hasil yang diperoleh:

```

Masukkan bilangan bulat yang akan dibagi: 10
Masukkan bilangan pembagi          : 3
Hasil Bagi = 3.33

```

5.3 Struktur if Tiga Kondisi atau Lebih

Struktur jenis ini merupakan struktur percabangan yang biasanya membingungkan para programmer pemula, percabangan jenis ini merupakan perluasan dari struktur yang memiliki dua kondisi diatas, yaitu dengan menyisipkan (menambahkan) satu atau lebih kondisi didalamnya.

Bentuk umum dari struktur percabangan yang memiliki lebih dari dua kondisi adalah :

```
If (kondisi1) {
    Statemen_jika_kondisi1_terpenuhi;
} else if (kondisi2) {
    Statemen_Jika_kondisi2_terpenuhi;
} else if (kondisi3) {
    Statemen_Jika_kondisi3_terpenuhi;
}
...
Else {
    Statemen_Jika_semua_kondisi_diatas_tidak_terpenuhi;
}
```

Contoh programnya:

Kode Program 5-5

```
#include <iostream>

using namespace std;

int main() {
    int bil;

    cout<<"Masukkan sebuah bilangan bulat "
    <<"yang akan diperiksa: ";
    cin>>bil;

    if (bil > 0) {
        cout<<bil<<" adalah bilangan POSITIF";
    } else if (bil < 0) {
        cout<<bil<<" adalah bilangan NEGATIF";
    } else {
        cout<<"Anda memasukkan bilangan NOL";
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan bilangan bulat yang akan diperiksa: -6
-6 adalah bilangan NEGATIF
```

Contoh lainnya:

Kode Program 5-6

```
#include <iostream>
// Memasukkan header <cmath>
// untuk memanggil fungsi sqrt()
#include <cmath>

using namespace std;

int main() {
    int a, b, c;
    float D, x1, x2;
    int flag;

    // Meminta user untuk menentukan
    // koefisien persamaan kuadrat
    cout<<"Masukkan nilai a: "; cin>>a;
    cout<<"Masukkan nilai b: "; cin>>b;
    cout<<"Masukkan nilai c: "; cin>>c;
    cout<<endl;

    // Menghitung nilai determinan
    D = (b*b) - (4*a*c);

    // Menentukan akar-akar persamaan kuadrat
    if (D > 0) {
        x1 = ((-b) + sqrt(D)) / (2*a);
        x2 = ((-b) - sqrt(D)) / (2*a);
        flag = 1;
    } else if (D == 0) {
        x1 = ((-b) + sqrt(D)) / (2*a);
        x2 = x1;
        flag = 1;
    } else {
        flag = 0;
    }

    // Menampilkan akar-akar persamaan kuadrat
    // dapat ditulis dengan if (flag) {
    if (flag == 1) {
        cout<<"x1 = "<<x1<<endl;
        cout<<"x2 = "<<x2<<endl;
    } else {
        cout<<"x1 dan x2 imajiner";
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan nilai a: 1
Masukkan nilai b: -5
Masukkan nilai c: 6

x1 = 3
x2 = 2
```

Sebagai contoh terakhir dari penggunaan statemen if bersarang ini, disini kita akan membuat program yang dapat menentukan nilai indeks (A, B, C, D, dan E) dari nilai akhir yang didapatkan oleh seorang mahasiswa.

Misalkan kita punya ketentuan penilaian seperti ini:

- A : nilai ≥ 85
- B : $70 \leq \text{nilai} < 85$
- C : $55 \leq \text{nilai} < 75$

- D : $40 \leq \text{nilai} < 55$
E : $\text{nilai} < 40$

Dengan ketentuan diatas dapat ditulis kode programnya:

Kode Program 5-7

```
#include <iostream>

using namespace std;

int main() {
    double nilai;
    char indeks;

    // Meminta user untuk menentukan nilai
    // yang diperoleh dalam bentuk bilangan
    cout<<"Masukkan nilai yang diperoleh: ";
    cin>>nilai;

    // Melakukan konversi nilai numerik
    // menjadi nilai indeks
    if (nilai >= 85) {
        indeks = 'A';
    } else if (nilai >= 70) {
        indeks = 'B';
    } else if (nilai >= 55) {
        indeks = 'C';
    } else if (nilai >= 40) {
        indeks = 'D';
    } else {
        indeks = 'E';
    }

    // Menampilkan nilai indeks yang didapatkan
    cout<<"Nilai indeks dari nilai "
        <<nilai<<" adalah "<<indeks;

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan nilai yang diperoleh : 76
Nilai indeks dari nilai 76 adalah B
```

5.4 Percabangan Menggunakan Statemen switch

Selain menggunakan statemen if, C++ juga menawarkan kepada kita untuk dapat melakukan percabangan (pemilihan) dengan menggunakan statemen switch.

Bentuk umum penggunaan statemen switch adalah sebagai berikut:

```
Switch (ekspresi) {
    Case nilai_konstan1 : Statemen_statemen; break;
    Case nilai_konstan2 : Statemen_statemen; break;
    ....
    Case nilai_konstanN: Statmen-statemen;berak;
    Default:
    Statemen_statemen_alternatif;
}
```

Contoh programnya:

Kode Program 5-8

```
#include <iostream>

using namespace std;

int main() {
    int nohari;

    cout<<"Masukkan nomor hari (1..7): ";
    cin>>nohari;

    switch (nohari) {
        case 1:
            cout<<"Hari ke-"<<nohari<<": adalah MINGGU";
            break;
        case 2:
            cout<<"Hari ke-"<<nohari<<": adalah SENIN";
            break;
        case 3:
            cout<<"Hari ke-"<<nohari<<": adalah SELASA";
            break;
        case 4:
            cout<<"Hari ke-"<<nohari<<": adalah RABU";
            break;
        case 5:
            cout<<"Hari ke-"<<nohari<<": adalah KAMIS";
            break;
        case 6:
            cout<<"Hari ke-"<<nohari<<": adalah JUMAT";
            break;
        case 7:
            cout<<"Hari ke-"<<nohari<<": adalah SABTU";
            break;
        default:
            cout<<"Tidak terdapat nama hari ke-"<<nohari;
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Masukkan sebuah bilangan (1...7) : 3
Hari ke-3 adalah SELASA
```

BAB VI PENGULANGAN

Pengulangan adalah suatu proses yang melakukan statemen-statemen dalam sebuah program secara terus menerus sampai terdapat kondisi untuk menghentikannya. Struktur pengulangan akan sangat membantu dalam efisiensi program. perhatikan program berikut:

Kode Program 6-1

```
#include <iostream>

using namespace std;

int main() {

    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;
    cout<<"Saya sangat menyukai C++"<<endl;

    return 0;
}
```

Dalam bahasa C++ terdapat tiga buah jenis struktur pengulangan, yaitu :

1. Struktur for
2. Struktur while
3. Struktur do-while

Sebagai programmer C++, kita perlu memahami dengan baik konsep dan penggunaan ketiga jenis struktur pengulangan tersebut.

6.1 Struktur for

Struktur pengulangan jenis ini biasanya digunakan untuk melakukan pengulangan yang telah diketahui banyaknya. Kita harus memiliki variabel sebagai indeksinya.

Untuk diperhatikan bahwa tipe data dari variabel yang akan digunakan sebagai indeks haruslah tipe data yang mempunyai urutan yang teratur. Misalnya tipe data int (0,1,2,...) atau char ('a', 'b', 'c',....)

Adapun bentuk umum dari struktur for adalah seperti dibawah ini:

```
// Untuk pengulangan yang sifatnya menaik (increment)
For (variabel =nilai_awal;kondisi;variabel++) {
    Statemen_yang_akan_diulang;
}

// Untuk pengulangan yang sifatnya menurun (decrement)
For (variabel =nilai_awal;kondisi;variabel - - ) {
    Statemen_yang_akan_diulang;
}
```

Berikut contoh program menggunakan struktur for yang sifatnya menaik (increment)

Kode Program 6-2

```
#include <iostream>

using namespace std;

int main() {
    int C;
    for (C=0; C<10; C++) {
        cout<<"Saya sangat menyukai C++"<<endl;
    }

    return 0;
}
```

Struktur for yang sifatnya menurun (decrement)

Kode Program 6-3

```
#include <iostream>

using namespace std;

int main() {
    int C;
    for (C=10; C>0; C--) {
        cout<<"Saya sangat menyukai C++"<<endl;
    }

    return 0;
}
```

Kedua program diatas akan menghasilkan hasil yang sama.

Hasil yang diperoleh:

```
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
```

Apabila masih merasa bingung tentang perbedaan antara pengulangan yang sifatnya menaik dan menurun, coba perhatikan program dibawah ini:

Kode Program 6-4

```
#include <iostream>

using namespace std;

int main() {
    cout<<"PENGULANGAN MENAIK"<<endl;
    for (int C=0; C<10; C++) {
        cout<<C+1<<endl;
    }

    // Membuat spasi vertikal
    cout<<'\n'; // dapat ditulis cout<<endl;
    cout<<"PENGULANGAN MENURUN"<<endl;
    for (int J=10; J>0; J--) {
        cout<<J<<endl;
    }

    return 0;
}
```

Hasil yang diperoleh:

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

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

Struktur for dengan banyak variabel

Struktur for didalam C++ dapat juga melibatkan lebih dari satu variabel, namun yang jelas satu diantaranya akan digunakan sebagai indeks pengulangan. Berikut contoh program struktur for yang melibatkan tiga buah variabel yaitu A, B, C.

Kode Program 6-5

```
#include <iostream>

using namespace std;

int main() {
    char A; // variabel A (bertipe char) akan digunakan
            // sebagai indeks pengulangan
    int B; // variabel B akan digunakan untuk
           // menampung nilai penjumlahan
    int C; // variabel C akan digunakan untuk
           // menampung nilai perkalian

    for (A='a', B=0, C=1; A<='e'; A++, B=B+5, C=C*10) {
        cout<<"Nilai A = "<<A<<endl;
        cout<<"Nilai B = "<<B<<endl;
        cout<<"Nilai C = "<<C<<endl;
        cout<<endl;
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Nilai A = a
Nilai B = 0
Nilai C = 1

Nilai A = b
Nilai B = 5
Nilai C = 10

Nilai A = c
Nilai B = 10
Nilai C = 100

Nilai A = d
Nilai B = 15
Nilai C = 1000

Nilai A = e
Nilai B = 20
Nilai C = 10000
```

Struktur for Bersarang

Sama halnya seperti pada percabangan, pada struktur pengulangan juga dapat diterapkan pengulangan bersarang (nested looping).

Bentuk umumnya dari struktur for bersarang adalah

```
for (variabel1=nilai_awal;kondisi1;variabel1++) {
    for (variabel2=nilai_awal;kondisi2;variabel2++) {
        for (variabel3=nilai_awal;kondisi3;variabel3++) {
            Statemen_statemen_yang_akan_diulang;
        }
    }
}
```

Contoh programnya:

Kode Program 6-6

```
#include <iostream>

using namespace std;

int main() {
    for (int j=1; j<=10; j++) {
        for (int k=1; k<=j; k++) {
            cout<<k*j<<' ';
        }
        cout<<'\n';
    }

    return 0;
}
```

Hasil yang diperoleh:

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

6.2 Struktur while

Struktur pengulangan jenis ini adalah pengulangan yang melakukan pemeriksaan kondisi diawal blok struktur.

Kita tahu bahwa pengulangan hanya akan dilakukan jika kondisi yang didefinisikan didalamnya terpenuhi (bernilai benar). Hal ini berarti jika kondisi yang didefinisikan tidak terpenuhi (bernilai salah) maka statemen-statemen yang terdapat dalam blok pengulangan pun tidak akan pernah dieksekusi program.

Bentuk umum dari struktur pengulangan while adalah sebagai berikut:

```
While (kondisi) {
    Statemen_statemen_yang_akan_diulang;
}
```

Berikut contoh programnya:

Kode Program 6-7

```
#include <iostream>

using namespace std;

int main() {
    int C;    // Mendeklarasikan variabel C sebagai
              // indeks pengulangan

    C = 0;    // Melakukan inisialisasi nilai terhadap
              // variabel C

    while (C<10) {
        cout<<"Saya sangat menyukai C++"<<endl;
        C++;  /* Statemen ini berguna untuk menaikkan nilai,
               dan setelah bernilai 10,
               maka pengulangan akan dihentikan */
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
```

Berikut contoh program yang akan melakukan pengulangan terus menerus karena tidak adanya statemen untuk menghentikan pengulangan.

Kode Program 6-8

```
// Contoh Program yang SALAH

#include <iostream>

using namespace std;

int main() {

    int C;    // Mendeklarasikan variabel C
              // sebagai indeks pengulangan

    C = 0;    // Melakukan inisialisasi nilai
              // terhadap variabel C

    while (C<10) {
        cout<<"Saya sangat menyukai C++"<<endl;
        /* Pada baris ini tidak terdapat statemen increment
           sehingga C akan terus bernilai 0
           dan kondisi C<10 akan selalu bernilai benar.
           Akibatnya, pengulangan akan terus dilakukan */
    }

    return 0;
}
```

Struktur while Bersarang

Berikut contoh while bersarang

Kode Program 6-9

```
#include <iostream>

using namespace std;

int main() {
    int J = 10;
    int K;

    while (J >= 1) {
        K = 1;
        while (K <= J) {
            cout<<K*J<<' ';
            K++;
        }
        cout<<'\n';
        J--;
    }

    return 0;
}
```

Hasil yang diperoleh:

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

6.3 Struktur do-while

Berbeda dengan struktur while yang melakukan pemeriksaan kondisi diawal blok perulangan, pada struktur do-while kondisi justru ditempatkan dibagian akhir. Hal ini menyebabkan struktur pengulangan ini minimal akan melakukan satu kali proses walaupun kondisi yang didefinisikan tidak terpenuhi (bernilai salah).

Berikut bentuk umum dari struktur do-while

```
do {
    Statemen_yang_akan_diulang;
} while (kondisi);
```

Berikut contoh programnya

Kode Program 6-10

```
#include <iostream>

using namespace std;

int main() {

    int C = 0;

    do {
        cout<<"Saya sangat menyukai C++"<<endl;
        C++;
    } while (C < 10);

    return 0;
}
```

Hasil yang diperoleh:

```
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
Saya sangat menyukai C++
```

Struktur do - while Bersarang

Adapun kode programnya dapat dilihat dibawah ini:

Kode Program 6-11

```
#include <iostream>

using namespace std;

int main() {
    int J = 10;
    int K;

    do {
        K = 1;
        do {
            cout<<K*J<<' ';
            K++;
        } while (K <= J);
        cout<<'\n';
        J--;
    } while (J >= 1);

    return 0;
}
```

Hasil yang diperoleh:

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

BAB VII
ARRAY

Array (larik) merupakan hal pundamental yang sering dijumpai dalam banyak kasus di dunia pemrograman. Makanya perlu pemahaman yang baik konsep array dan mampu mengimplementasikannya kedalam kasus-kasus yang akan dihadapi.

7.1 Apa itu Array

Array adalah sebuah variabel yang menyimpan sekumpulan data yang memiliki tipe sama. Setiap data tersebut menempati lokasi atau alamat memori yang berbeda-beda dan selanjutnya disebut elemen array.

Elemen array itu kemudian dapat diakses melalui indeks yang terdapat didalamnya. Perlu diperhatikan dalam bahasa C++ indeks array selalui dimulai dari 0, bukan 1.

Berikut ini gambar ilustrasi sebuah array

| | | | | |
|-------------|-------------|-------|-------------|----------------------------|
| Nilai ke-1 | Nilai ke-2 | | Nilai ke-N | -----> Nilai elemen array |
| Alamat ke-1 | Alamat ke-2 | | Alamat ke-N | -----> Alamat elemen array |
| 0 | 1 | | N | -----> Indeks elemen array |

Untuk mendeklarasikan sebuah array dalam bahasa C++, kita harus menggunakan tanda [] (bracket). Adapun bentuk umum dari pendeklarasiannya adalah sebagai berikut:

Tipe_data nama_array[jumlah_elemen];

Sebagai contoh, apabila kita ingin mendeklarasikan sebuah array (misal dengan nama dengan nama LARIK) yang memiliki 25 elemen dengan tipe data int, maka bentuk deklarasinya adalah seperti dibawah ini:

```
int LARIK [25];
```

Ruang memori yang dibutuhkan untuk deklarasi array tersebut adalah 100 byte, yang berasal dari 25 x 4 byte (4 merupakan ukuran tipe data int).

7.2 Mengisikan Nilai ke dalam Elemen Array

Untuk mengisikan nilai kedalam elemen-elemen array, dapat langsung melakukannya untuk setiap elemen, misalnya

```
A[0] = 10;
B[1] = 20;
C[2] = 30;
...
dst
```

Namun cara ini tidak direkomendasikan karena tidak efisien. Cara yang banyak digunakan oleh programmer untuk mengisi array adalah dengan menggunakan pengulangan (looping).

Sebagai contoh, apabila kita ingin melakukan pengisian 25 elemen array, maka dapat dituliskan kode seperti ini:

```
for (int C=0; C<25; C++){
    cout<<"A["<<C<<"]= "; cin>>A[C];
}
```

Berikut contoh programnya dalam proses pengisian array menggunakan proses pengulangan.

Kode Program 7-1

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan array A
    // dengan 5 buah elemen bertipe int
    int A[5];

    // Memasukkan nilai ke dalam elemen array
    for (int C=0; C<5; C++) {
        cout<<"A["<<C<<"] = "; cin>>A[C];
    }

    return 0;
}
```

7.3 Menampilkan Nilai yang terdapat pada Array

Berikut program yang akan mengisi dan menampilkan nilai yang terdapat pada elemen array.

Kode Program 7-2

```
#include <iostream>

using namespace std;

int main() {

    // Mendeklarasikan array A
    // dengan 5 buah elemen bertipe int
    int A[5];

    // Mengisi nilai ke dalam elemen array
    cout<<"Masukkan nilai yang diinginkan"<<endl;
    for (int C=0; C<5; C++) {
        cout<<"A["<<C<<"] = "; cin>>A[C];
    }

    cout<<"\n";

    // Menampilkan nilai yang terdapat dalam elemen array
    cout<<"Menampilkan nilai yang telah dimasukkan"<<endl;
    for (int J=0; J<5; J++) {
        cout<<"Nilai yang terdapat pada elemen ke-";
        cout<<J+1<<": "<<A[J]<<endl;
    }
}
```

7.4 Melakukan Inisialisasi Array

Pada saat kita mendeklarasikan sebuah array, kita dapat langsung melakukan inisialisasi nilai terhadap elemen-elemen array didalamnya. Hal ini dimaksudkan untuk mengisi nilai awal (default) pada elemen array sehingga jika elemen bersangkutan tidak diisi dengan nilai baru, maka nilai yang digunakan adalah nilai yang sudah ada.

Bentuk umum dari proses inisialisasi array adalah sebagai berikut:

```
tipe_data nama_array[N] = { nilai1, nilai2,...,nilai N};
```

berikut contoh program yang menunjukkan proses inisialisasi nilai pada elemen-elemen array.

Kode Program 7-3

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan array
    // dan langsung menginisialisasi nilainya
    int A[5] = { 10, 20, 30, 40, 50 };

    // Menampilkan nilai
    // yang terdapat pada elemen array
    cout<<"Sebelum dilakukan perubahan nilai"<<endl;
    cout<<"A[0] = "<<A[0]<<endl;
    cout<<"A[1] = "<<A[1]<<endl;
    cout<<"A[2] = "<<A[2]<<endl;
    cout<<"A[3] = "<<A[3]<<endl;
    cout<<"A[4] = "<<A[4]<<endl;

    // Mengubah elemen ke-1 dan ke-2
    A[0] = 12;
    A[1] = 25;

    // Menampilkan kembali nilai
    // yang terdapat pada elemen array
    cout<<"Setelah dilakukan perubahan nilai"<<endl;
    cout<<"A[0] = "<<A[0]<<endl;
    cout<<"A[1] = "<<A[1]<<endl;
    cout<<"A[2] = "<<A[2]<<endl;
    cout<<"A[3] = "<<A[3]<<endl;
    cout<<"A[4] = "<<A[4]<<endl;

    return 0;
}
```

Hasil yang diperoleh:

```
Sebelum dilakukan perubahan nilai
10
20
30
40
50

Setelah dilakukan perubahan nilai
12
25
30
40
50
```


7.5 Melakukan Pencarian pada Elemen Array

Salah satu permasalahan yang sering muncul pada saat kita menggunakan array adalah tuntutan untuk melakukan pencarian elemen array.

Contoh-contoh pencarian adalah pencarian data mahasiswa disalah satu jurusan dari perguruan tinggi tertentu, pencarian data nasabah bank, pencarian nilai terbesar atau terkecil dari suatu kumpulan data bilangan.

Berikut ini contoh program yang akan menunjukkan cara melakukan pencarian nilai dari sekumpulan data yang bertipe int.

Kode Program 7-4

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan array
    // dengan melakukan inisialisasi nilai ke dalamnya
    int A[10] = { 12, 24, 14, 25, 10,
                 13, 21, 20, 15, 18 };
    int BIL;    // Variabel untuk menampung
               // nilai yang akan dicari

    // Menampilkan nilai yang terdapat
    // pada elemen-elemen array di atas
    for (int C=0; C<10; C++) {
        cout<<"A["<<C<<"]: "
             <<A[C]<<endl;
    }
    cout<<endl;

    // Memasukkan nilai yang akan dicari
    cout<<"Masukkan nilai yang akan dicari: ";
    cin>>BIL;

    // Melakukan pencarian data
    for (int J=0; J<10; J++) {
        if (A[J] == BIL) {
            cout<<"Nilai yang dicari "
                 <<"terdapat pada indeks ke-"<<J;
            break;
        }
    }

    return 0;
}
```

Hasil yang diperoleh:

```
A[0]: 12
A[1]: 24
A[2]: 14
A[3]: 25
A[4]: 10
A[5]: 13
A[6]: 21
A[7]: 20
A[8]: 15
A[9]: 18
```

```
Masukkan nilai yang akan dicari: 13
Nilai yang dicari terdapat pada indeks ke-5
```

Sebenarnya nilai 13 ditemukan pada elemen array ke-6, namun karena indeksnya dimulai dari 0, maka indeks yang akan diberikan untuk data tersebut adalah 5.

Contoh program lainnya:

Kode Program 7-5

```
#include <iostream>

using namespace std;

int main() {
    // Mendeklarasikan array
    // dengan melakukan inisialisasi nilai ke dalamnya
    int A[10] = { 12, 24, 14, 25, 10,
                 13, 21, 20, 15, 18 };

    int BIL;    // Variabel untuk menampung
                // nilai yang akan dicari

    // Menampilkan nilai yang terdapat
    // pada elemen-elemen array di atas
    for (int C=0; C<10; C++) {
        cout<<"A["<<C<<"] : "
            <<A[C]<<endl;
    }
    cout<<endl;

    // Memasukkan nilai yang akan dicari
    cout<<"Masukkan nilai yang akan dicari: ";
    cin>>BIL;

    // Melakukan pencarian data
    for (int J=0; J<10; J++) {
        if (A[J] == BIL) {
            cout<<"Nilai yang dicari "
                <<"terdapat pada indeks ke-"<<J;
            break;
        }
    }

    return 0;
}
```

Hasil yang diperoleh:

```
Nilai maksimum : 98
Nilai minimum  : 10
```

7.6 Mengurutkan Elemen Array

Untuk mengurutkan elemen array dapat menggunakan dua buah metode yaitu:

1. Metode Gelembung
2. Metode maksimum-minimum.

Salah satu dari kegunaan suatu pengurutan data adalah untuk mempermudah dan mempercepat proses pencarian data. Untuk lebih memahaminya, perhatikan dua buah program dibawah ini:

1. Metode Gelembung(bubble sort).

Contoh program dibawah ini akan menunjukkan cara mengurutkan array menggunakan metode gelembung (bubble sort).

Kode Program 7-6

```
#include <iostream>

using namespace std;

int main() {

    // Mendeklarasikan array dengan 7 buah elemen
    // yang bertipe int
    int A[7];

    // Mendeklarasikan variabel-variabel bantu
    // yang diperlukan
    int j, k, C, temp;

    // Memasukkan nilai array
    cout<<"Masukkan nilai pada elemen array:"
    <<endl;
    for (C=0; C<7; C++) {
        cout<<"A["<<C<<" ] = "; cin>>A[C];
    }

    // Menampilkan nilai sebelum diurutkan
    cout<<"\nNilai elemen array sebelum diurutkan:"
    <<endl;
    for (C=0; C<7; C++) {
        cout<<"A["<<C<<" ] = "<<A[C]<<endl;
    }

    // Melakukan pengurutan elemen array
    // dengan metode gelembung
    for (j=0; j<6; j++) {
        for (k=7; k>0; k--) {
            if (A[k] < A[k-1]) {
                temp = A[k];
                A[k] = A[k-1];
                A[k-1] = temp;
            }
        }
    }

    // Menampilkan nilai setelah diurutkan
    cout<<"\nNilai elemen array setelah diurutkan:"
    <<endl;
    for (C=0; C<7; C++) {
        cout<<"A["<<C<<" ] = "<<A[C]<<endl;
    }

    return 0;
}
```

2. Metode Maksimum-Minimum.

Kode Program 7-7

```
#include <iostream>

using namespace std;

int main() {

    // Mendeklarasikan array dengan 7 buah elemen
    // yang bertipe int
    int A[7];

    // Mendeklarasikan variabel-variabel bantu
    // yang diperlukan
    int j, k, C, temp;

    // Memasukkan nilai array
    cout<<"Masukkan nilai pada elemen array:"
        <<endl;
    for (C=0; C<7; C++) {
        cout<<"A["<<C<<" ] = "; cin>>A[C];
    }

    // Menampilkan nilai sebelum diurutkan
    cout<<"\nNilai elemen array sebelum diurutkan:"
        <<endl;
    for (C=0; C<7; C++) {
        cout<<"A["<<C<<" ] = "<<A[C]<<endl;
    }

    // Melakukan pengurutan elemen array
    // dengan metode maksimum-minimum
    int jmaks, U=6;

    for (j=0; j<6; j++) {
        jmaks = 0;
        for (k=1; k<=U; k++) {
            if (A[k] > A[jmaks]) {
                jmaks = k;
            }
        }
        temp = A[U];
        A[U] = A[jmaks];
        A[jmaks] = temp;
        U--;
    }

    // Menampilkan nilai setelah diurutkan
    cout<<"\nNilai elemen array setelah diurutkan:"
        <<endl;
    for (C=0; C<7; C++) {
        cout<<"A["<<C<<" ] = "<<A[C]<<endl;
    }

    return 0;
}
```

7.7 Array Multidimensi

Array Multidimensi yaitu array yang terdiri dari beberapa subskrip. Sebagai contoh, array 2 dimensi adalah array yang mempunyai 2 subskrip, 3 dimensi mempunyai 3 subskrip, dan seterusnya. Array seperti ini sering digunakan untuk pemrosesan matrik.

ARRAY DUA DIMENSI

Adalah array yang mempunyai dua buah subskrip, yaitu baris dan kolom. Bentuk umum pendeklarasian sebuah array dua dimensi didalam C++ adalah sebagai berikut:

```
Tipe_data nama_array[jumlah_elemen_baris][jumlah_elemen_kolom];
```

Sebagai contoh, apabila kita melakukan penjumlahan 2 buah matrik ordo 3 x 2, maka contoh programnya adalah sebagai berikut:

Kode Program 7-8

```
#include <iostream>

using namespace std;

int main() {

    // Mendefinisikan tipe data
    // yang berupa array dua dimensi
    typedef int MATRIK32 [3][2];

    // Mendeklarasikan array A
    // sebagai array dua dimensi
    MATRIK32 A, B, C;

    int j, k; // Mendeklarasikan variabel
              // untuk indeks pengulangan

    // Mengisikan nilai
    // ke dalam elemen-elemen array A
    for (j=0; j<3; j++) {
        for (k=0; k<2; k++) {
            cout<<"A["<<j<<"["<<k<<" = ";
            cin>>A[j][k];
        }
    }
    cout<<endl;
```

```

// Mengisikan nilai
// ke dalam elemen-elemen array B
for (j=0; j<3; j++) {
    for (k=0; k<2; k++) {
        cout<<"B["<<j<<"] ["<<k<<"] = ";
        cin>>B[j][k];
    }
}
cout<<endl;

// Melakukan penjumlahan A dan B
// dan menyimpan hasilnya ke dalam array C
for (j=0; j<3; j++) {
    for (k=0; k<2; k++) {
        C[j][k] = A[j][k] + B[j][k];
    }
}

// Menampilkan hasil penjumlahan
for (j=0; j<3; j++) {
    for (k=0; k<2; k++) {
        cout<<"C["<<j<<"] ["<<k<<"] = "
            <<C[j][k]<<endl;
    }
}

return 0;
}

```

Inisialisasi pada Array Multidimensi

Dapat dilakukan inisialisasi nilai ke dalam elemen-elemennya. Caranya seperti berikut:

```
int A[3][3] = { 1,2,3,4,5,6,7,8,9 };
```

Namun untuk memudahkan proses inisialisasi,C++ mengizinkan kita untuk melakukan pengelompokkan untuk setiap baris, yaitu dengan kode seperti ini

```
int A[3][3] = { {1,2,3},{4,5,6},{7,8,9} };
```

Untuk membuktikan hal tersebut, perhatikan contoh programnya

Kode Program 7-9

```

#include <iostream>

using namespace std;

int main() {

    // Melakukan inisialisasi nilai
    // ke dalam elemen-elemen array dua dimensi
    int A[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };

    // Menampilkan nilai yang tersimpan
    // dalam elemen array
    for (int j=0; j<3; j++) {
        for (int k=0; k<3; k++) {
            cout<<"A["<<j<<"] ["<<k<<"] = "
                <<A[j][k]<<endl;
        }
        cout<<endl;
    }

    return 0;
}

```

DAFTAR PUSTAKA

Budi Raharjo., 2014, *Pemrograman C++ Mudah dan cepat menjadi Master C++*, menggunakan Dev-C++. Penerbit INFORMATIKA Bandung.

Abdul Kadir., 2012, *Buku Pintar C++ untuk Pemula*, Menggunakan CodeBlocks. Penerbit MediaKom. Yogyakarta.

Hanif Al Fatta., 2006, *Dasar Pemrograman C++ Disertai dengan pengenalan pemrograman Berorientasi Objek*. Penerbit CV ANDI OFFSET. Yogyakarta.