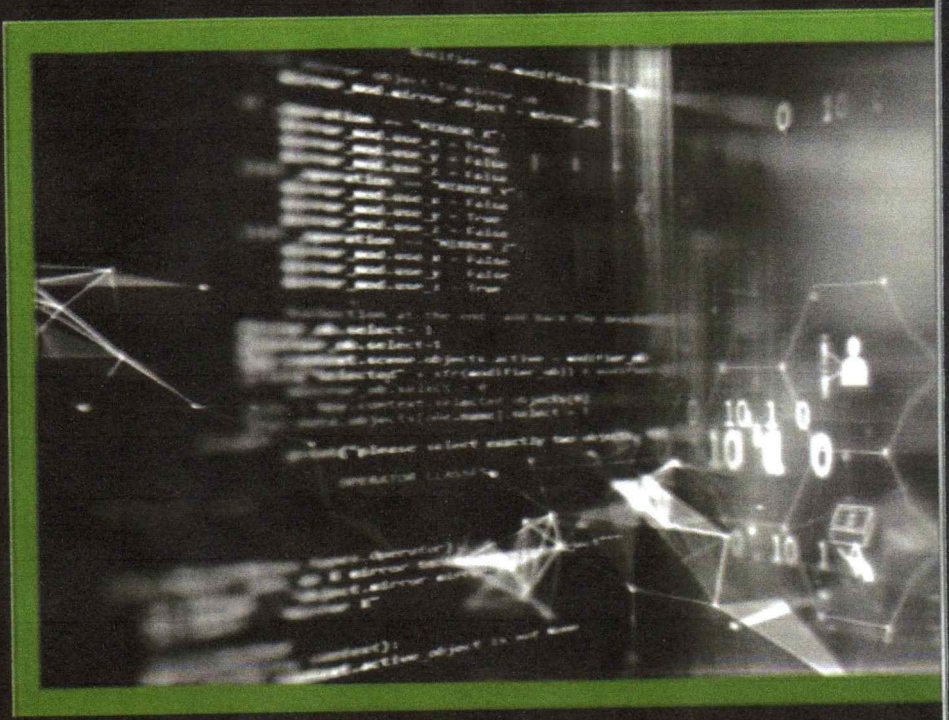
Pemrograman Dasar



Anita Qoiriah
Rina Harimurti
Andi Iwan Nurhidayat
Asmunin

Zifatama
JAWARA

PEMOGRAMAN DASAR

PENULIS :

Anita Qoiriah
Rina Harimurti
Andi Iwan Nurhidayat
Asmunin

PEMOGRAMAN DASAR

Penulis : Anita Qoiriah
Rina Harimurti
Andi Iwan Nurhidayat
Asmunin

Diterbitkan Oleh:

Penerbit
Zifatama Jawara
Jl. Taman Pondok Jati J4,
Taman - Sidoarjo
Telp : 031-99786278
Email : zifatama1@gmail.com
Anggota IKAPI No. 149/JTI/2014

Cetakan Pertama, November 2019
Ukuran/ Jumlah hal: 15,5 x 23 mm / 162 hlm

ISBN : 978-602-5815-80-5

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah S.W.T atas rahmat dan hidayah yang dilimpahkan, sehingga buku ini dapat diselesaikan dengan baik.

Buku ini merupakan pengantar untuk belajar pemrograman. Dengan panduan langkah demi langkah diharapkan akan membantu kegiatan yang rumit seperti pemrograman. Di akhir setiap bab terdapat latihan untuk dicoba serta berbagai macam tugas pemrograman, mulai dari latihan sederhana sampai masalah pada aplikasi realistis. Buku ini menggunakan bahasa pemrograman C++ sebagai wahana untuk memperkenalkan pemrograman komputer.

Buku disusun sedemikian rupa sehingga memudahkan untuk memahami bagaimana menyusun program dengan menggunakan bahasa pemrograman C++. Bab 1 membahas bagaimana menyelesaikan permasalahan dengan menggunakan diagram alir. Bab 2 membahas elemen dasar pemrograman C++. Bab 3 membahas tentang fungsi-fungsi input output. Bab 4 membahas struktur kondisional. Bab 5 membahas struktur pengulangan. Bab 6 membahas array. Bab 7 membahas fungsi.

Penulis menyadari buku ini tidak terlepas dari kekurangan. Penulis mengharapkan masukan yang membangun untuk perbaikan buku ini di masa akan datang. Ucapan terima kasih yang dalam ditujukan kepada orang-orang yang telah memberikan masukan dan bantuan sehingga buku ini dapat tersusun.

Nopember 2019

Penulis

DAFTAR ISI

1

Pengantar Pemrograman

Dalam bab ini akan dibahas mengenai bahasa pemrograman, jenis eksekusi bahasa pemrogrman, bagaimana menganalisa masalah dan membuat algoritma pemrograman. Selain itu juga penjelasan bagaimana menggambarkan algoritma dalam bentuk flowchart, beberapa bentuk aliran flowchart beserta contoh-contohnya

Pendahuluan

Komputer adalah suatu mesin. Dan seperti mesin lainnya, komputer harus dinyalakan dan kemudian dikendalikan untuk melakukan tugas. Jika pada mobil, pengemudi yang mengarahkan mobil dan memberikan kontrol. Di komputer, program komputer yang memberikan kontrol.

Program komputer adalah kombinasi data dan instruksi yang terstruktur yang digunakan untuk mengoperasikan komputer untuk menghasilkan hasil yang spesifik. Komputer beroperasi dengan mengikuti serangkaian instruksi. Namun instruksi yang dapat dimengerti adalah rangkaian bilangan biner berupa angka 1 dan 0 yang disusun untuk mengkodekan instruksi dan data. Untuk memudahkan penyusunan instruksi maka dikembangkan bahasa pemrograman.

Instruksi dasar yang dilakukan pada komputer adalah input (mendapatkan data), output (menghasilkan tampilan), penyimpanan, serta operasi aritmatika dan logika.

Bahasa Pemrograman

Program harus dibuat dalam bahasa yang dimengerti oleh komputer supaya instruksi yang ditulis oleh programmer dapat dilaksanakan oleh komputer. Ada bermacam-macam bahasa pemrograman yang digunakan untuk menulis program. Agar dapat dimengerti oleh komputer maka suatu bahasa pemrograman harus dieksekusi terlebih dahulu kedalam bahasa mesin. Eksekusi bahasa pemrograman tersebut ada dua macam yaitu dengan:

1. Interpreter
   Pada interpreter, suatu instruksi diterjemahkan kedalam kode mesin baris demi baris. Bahasa pemrogram seperti BASIC merupakan jenis interpreter. Umumnya bahasa jenis ini lebih lambat waktu eksekusinya.

2. Kompiler
   Pada kompailer, seluruh program diterjemahkan terlebih dahulu ke dalam kode mesin sebelum dijalankan. Contoh bahasa pemrograman jenis ini adalah C, Pascal, Java.

Bahasa pemrograman diklasifikasikan berdasarkan tingkatannya, semakin tinggi tingkatannya berarti semakin jauh dari bahasa komputer itu sendiri dan semakin dekat dengan bahasa manusia. Visual Basic, C, C ++, dan Java adalah contoh dari bahasa tingkat tinggi. Program akhir yang ditulis dalam bahasa ini dapat dijalankan pada berbagai jenis komputer, seperti yang diproduksi oleh IBM, Apple, dan Hewlett-Packard. Sebaliknya, bahasa tingkat rendah menggunakan instruksi yang terikat pada satu jenis komputer. Meskipun program yang ditulis dalam bahasa tingkat rendah terbatas, karena hanya dapat berjalan pada jenis komputer yang digunakan untuk menulisnya, tetapi mempunyai akses langsung fitur khusus internal perangkat keras dengan cara yang tidak mungkin dengan bahasa tingkat tinggi. Dan juga dapat dijalankan lebih cepat daripada program yang ditulis dalam bahasa tingkat tinggi. Berdasarkan tingkatannya, bahasa pemrograman dapat dibedakan:

1. Bahasa Mesin (Mnemonic Code)
   Bahasa mesin adalah bahasa yang berisi kode-kode mesin yang hanya dapat diinterpretasikan langsung oleh mesin komputer. Bahasa ini merupakan bahasa level terendah dan berupa kode numerik 0 dan 1.

2. Bahasa Assembly
   Bahasa assembly adalah bahasa simbol dari bahasa mesin. Setiap kode bahasa mesin memiliki simbol sendiri dalam bahasa assembly. Misalnya ADD untuk penjumlahan, MUL untuk perkalian, SUB untuk pengurangan, dan lain-lain.

3. Bahasa Tingkat Tinggi (High Level Language)
   Bahasa tingkat tinggi adalah bahasa pemrograman yang lebih tinggi daripada bahasa assembly. Bahasa ini lebih dekat dengan bahasa manusia dan lebih dipahami manusia. Contoh: Pascal, Delphi, Python, C++, Java, dll. Bahasa generasi ini disebut juga bahasa generasi ke-3 (3rd Generation Programming Language).

4. Bahasa yang berorientasi pada masalah spesifik
   Bahasa ini adalah bahasa yang digunakan langsung untuk memecahkan suatu masalah tertentu. Misalnya SQL untuk database. Bahasa ini juga masuk ke bahasa tingkat tinggi. Bahasa ini disebut juga bahasa generasi ke-4 (4th Generation Programmming Language).

Program

Program adalah kumpulan langkah-langkah instruksi yang mengatur komputer untuk mengerjakan tugas yang diinginkan dan menghasilkan hasil yang diinginkan. Untuk membuat sebuah program diperlukan bahasa pemrograman, yaitu sekumpulan aturan untuk memberitahukan komputer operasi apa yang harus dikerjakan. Bahasa pemrograman inilah yang akan menerjemahkan dari bahasa tingkat tinggi yang mudah dikenali manusia ke bahasa mesin.

Bahasa pemrograman mempunyai instruksi dasar pada komputer sebagai berikut:

1. Membaca masukan.
2. Mengerjakan pekerjaan secara berurutan.
3. Melakukan perhitungan.
4. Menyimpan data.
5. Melakukan perbandingan dan percabangan perintah.
6. Iterasi atau perulangan.
7. Menuliskan hasil.

Aliran instruksi dalam program dapat dieksekusi secara:

1. Sekuensial

   Eksekusi program dilakukan secara urut dari awal sampai akhir, tidak ada instruksi yang melompat atau berulang. Instruksi seperti ini hanya dapat menanganni beberapa situasi tertentu.

2. Kondisional/Pengambilan Keputusan

   Instruksi pengambilan keputusan dilakukan ketika terdapat pilihan apakah suatu operasi akan dikerjakan atau tidak, juga dilakukan ketika pengerjaan suatu operasi tertentu bergantung pada kondisi operasi yang lain.

   Pemilihan situasi dilakukan berdasarkan ekspresi nilai Boolean yang menggunakan operator relasional dan logika. Semua perbandingan dilakukan terhadap dua kondisii. Jika terdapat lebih dari dua perbandingan kondisi maka harus dilakukan multiple comparison, dimana masing-masing dilakukan antara sepasang kondisi.

3. Iterasi/Perulangan

   Dalam sebuah program sering terdapat pekerjaan yang harus dilakukan berulang. Untuk melakukan itu perlu untuk menyalin kode sebanyak perulangan yang diinginkan. Untuk mengatasi masalah ini terdapat instruksi iterasi yang biasa dinamakan looping untuk memerintahkan komputer agar melakukan pengulangan dengan kondisi tertentu.

   Dalam pernyataan perulangan, sebuah pernyataan dapat dieksekusi berkali-kali sampai ditemukan situasi yang mengharuskan keluar dari iterasi. Sama seperti pada kondisional, pemilihan situasi dilakukan berdasar ekspresi nilai Boolean yang

Pemrograman adalah proses pemecahan masalah. Orang yang berbeda menggunakan teknik yang berbeda untuk menyelesaikan masalah. Beberapa teknik diuraikan dengan baik dan mudah diikuti. Teknik-teknik tersebut tidak hanya memecahkan masalah, tetapi juga memberikan wawasan tentang bagaimana solusi itu dicapai.

Teknik pemecahan masalah ini dapat dengan mudah dimodifikasi jika domain masalah berubah. Untuk menjadi pemecah masalah yang baik dan programmer yang baik, harus mengikuti teknik pemecahan masalah yang baik. Salah satu teknik pemecahan masalah yang umum adalah menganalisis masalah, menguraikan persyaratan masalah, dan merancang langkah-langkah, yang disebut algoritma, untuk menyelesaikan masalah.

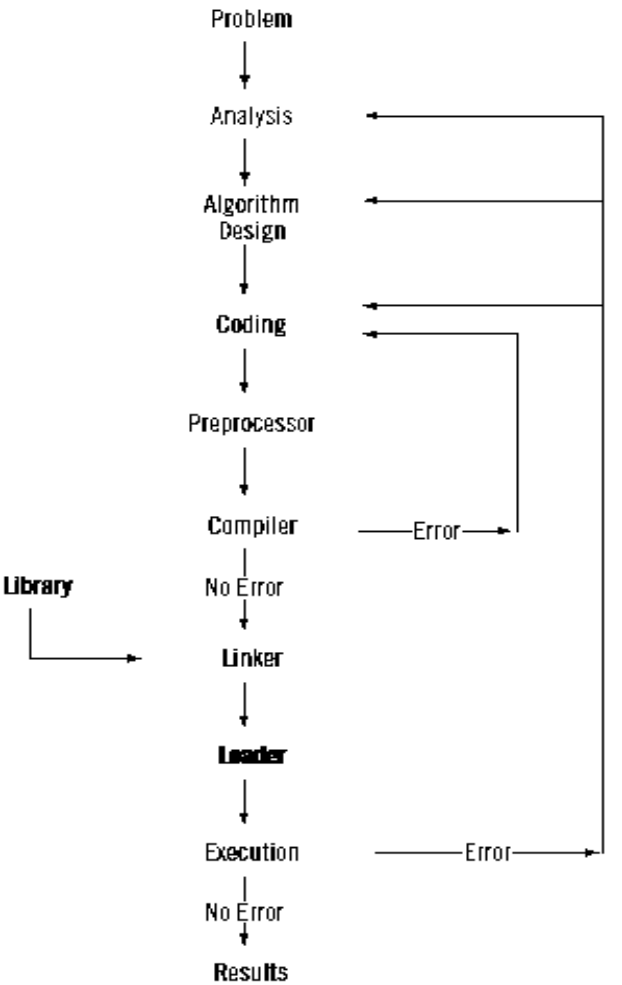
Dalaam pemrograman, proses pemecahan masalah memerlukam 3 langkah :

1. Menganalisa masalah, membuat garis besar masalah dan pemecahannya dan mendesain algoritma untuk memecahkan masalah. Langkah awal yang harus dilakukan adalah mengidentifikasi masalah antara lain tujuan dari pembuatan program, parameter-parameter yang digunakan, fasilitas apa saja yang akan disediakan oleh program. Kemudian menentukan metode atau algoritma apa yang akan diterapkan untuk menyelesaikan masalah tersebut dan terakhir menentukan bahasa program yang digunakan untuk pembuatan program.

2. Mengimplementasikan algoritma kedalam bahasa pemrograman seperti c++ dan memverifikasi apakah algoritma dapat dijalankan.

3. Merawat program dengan menggunakan dan memodifikasinya jika domain masalah berubah

Pada gambar 1.1 merupakan langkah-langkah pembuatan program



Gambar 1.1 Alur pembuatan dan eksekusi program

Dalam mengembangkan program untuk memecahkan masalah, dimulai dengan menganalisis masalah. Dilanjutkan dengan merancang algoritma, menulis instruksi program dalam bahasa tingkat tinggi, atau kode program dan menjalankan program dalam sistem komputer.

Menganalisis masalah adalah langkah pertama dan paling penting. Langkah ini mengharuskan untuk melakukan hal berikut:

1. Memahami masalahnya dengan seksama.
2. Memahami kebutuhan masalah. Seperti, apakah program memerlukan interaksi dengan pengguna, apakah perlu untuk memanipulasi data, apakah menghasilkan output dan sepeerti apa outputnya. Jika program manipulasi data, programmer harus mengetahui data apa dan bagaimana menyajikannya. Karena itu diperlukan sampel data. Jika program menghasilkan output, harus diketahui bagaimana format output dan bagaimana dihasilkannya.
3. Jika permasalahan cukup kompleks, masalah dibagi kedalam sub problem, dab diulangi langkah 1 & 2. Karena dengan masalah yang kompleks, maka perlu untuk menganalisa masing-masing sub problem dan memahami kebutuhan sub problem.

Setelah menganalisis masalah dengan cermat, langkah berikutnya adalah merancang algoritma untuk menyelesaikan masalah. Jika memecahkan masalah menjadi submasalah, diperlukan merancang algoritma untuk setiap submasalah. Setelah merancang suatu algoritma, perlu memeriksa kebenarannya. Menguji kebenaran suatu algoritma dengan menggunakan data sampel. Di lain waktu, Anda mungkin perlu melakukan beberapa analisis matematika untuk menguji kebenaran algoritme.

Secara umum, struktur suatu program terdiri dari beberapa bagian yaitu :

1. Input

   Bagian ini merupakan proses untuk memasukkan data ke komputer melalui piranti yang ada misalnya keyboard, mouse,

scanner. Program melakukan proses membaca data yang akan diolah dari piranti tersebut.

2. Output

   Bagian ini merupakan proses untuk menampilkan data yang telah diolah, melaporkan hasil pegolahan data melalui piranti seperti monitor, printer. Program melakukan proses mencetak data ke piranti tersebut.

3. Proses Pengolahan Data

   Bagian ini merupakan proses mengolah data yang dimasukkan dengan nenerapkan metode dan algoritma yang ada. Proses ini menghasilkan data output yang akan dikeluarkan ke pengguna program.

4. Penyimpanan Data

   Bagian ini merupakan proses menyimpan data dalam memori atau piranti penyimpanan data seperti disket, harddisk, CD.

Algoritma

Pengertian Algoritma

Dari sudut pandang prosedural, sebelum menulis suatu program, seorang programmer harus memahami dengan jelas data yang akan digunakan, hasil yang dimaksudkan, dan prosedur yang digunakan untuk menghasilkan hasil ini. Prosedur ini disebut sebagai algoritma. Lebih tepatnya, suatu algoritma adalah urutan langkah-demi-langkah dari instruksi yang menjelaskan bagaimana melakukan komputasi.

Setelah memahami dengan jelas data yang digunakan dan algoritma (langkah-langkah spesifik untuk mendapatkan hasilnya) maka dapat ditulis menjadi program. Algoritma yang dipilih diterjemahkan ke dalam program komputer dengan menggunakan bahasa pemrograman, seperti C ++.

Komputer tidak dapat diberi tahu misalnya untuk menambahkan angka dari 1 hingga 100. Sebagai gantinya, perlu diberikan urutan instruksi langkah demi langkah yang secara

kolektif membentuk suatu algoritma. Misalnya, urutan instruksi berikut untuk menentukan jumlah angka dari 1 hingga 100:

Set n sama dengan 100
Set a sama dengan 1
Set b sama dengan 100
Hitung sum = n (a + b)/2
Cetak sum

Instruksi ini bukanlah program komputer. Tidak seperti program, yang harus ditulis dalam bahasa yang dapat direspon komputer, algoritma dapat ditulis dengan berbagai cara. Ketika dituliskan dalam bahasa manusia untuk menggambarkan langkah-langkah dalam suatu algoritma, seperti dalam contoh sebelumnya, deskripsi disebut pseudocode. Ketika persamaan matematika digunakan, deskripsi disebut rumus. Ketika diagram dengan simbol digunakan, deskripsi disebut *flowchart*.

Algoritma merupakan metode efektif yang diekspresikan sebagai rangkaian instruksi-instruksi terbatas yang didefinisikan dengan baik untuk melakukan sebuah fungsi. Dimulai dari sebuah kondisi awal dan input awal (mungkin kosong), instruksi-instruksi tersebut menjelaskan suatu komputasi. Dan apabila dieksekusi, diproses lewat sejumlah urutan kondisi terbatas yang terdefinisi dengan baik, akan menghasilkan "keluaran" dan berhenti di kondisi akhir.

Contoh : Buat algoritma untuk menentukan apakah suatu bilangan merupakan bilangan ganjil atau bilangan genap.
Algoritmanya :

1. Bagi bilangan dengan bilangan 2.
2. Hitung sisa hasil bagi pada langkah 1.
3. Bila sisa hasil bagi sama dengan 0 maka bilangan itu adalah bilangan genap tetapi bila sisa hasil bagi sama dengan 1 maka bilangan itu adalah bilangan ganjil.

Perbedaan Algoritma dan Program

Agar dapat dilaksanakan oleh komputer, algoritma harus ditulis dalam notasi bahasa pemrograman sehingga dinamakan program. Jadi program adalah perwujudan atau implementasi

teknis Algoritma yang ditulis dalam bahasa pemrogaman tertentu sehingga dapat dilaksanakan oleh komputer.

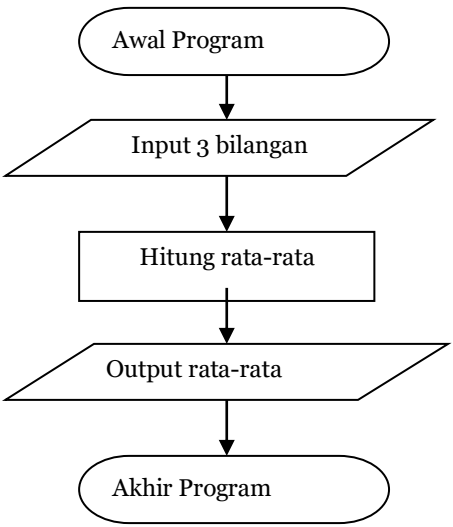
Algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemrograman, terutama dalam komputasi numeris. Tanpa algoritma yang dirancang baik maka proses pemrograman akan menjadi salah, rusak, atau lambat dan tidak efisien.

Aturan Notasi Algoritma

Penulisan algoritma tidak tergantung dari spesifikasi bahasa pemrograman dan komputer yang mengeksekusinya, melainkan bersifat umum. Tetapi notasi-notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.

Notasi algoritma dapat berupa :

- *Logic Flowchart*
  Menggambarkan aliran logika dalam program dan membantu pemrogram melihat desain program.



Gambar 1.2. Contoh Flowchart Sederhana

- *Pseudocode*

  Merupakan garis besar program dan lebih mengutamakan pada logika program. Pseudocode berbeda dengan perintah dalam bahasa pemrograman. Contoh pseudocode untuk menuliskan algoritma mencari nilai terbesar dari dua buah bilangan, maka pseudocode dapat dituliskan sebagai berikut:

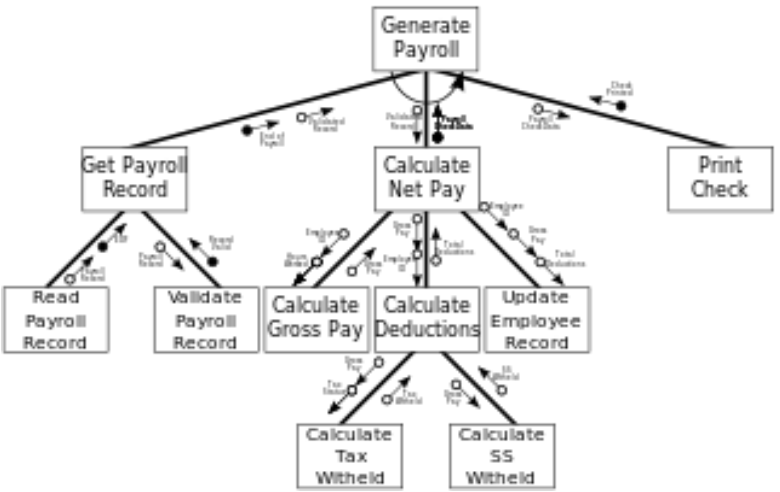
  Jika angka1 lebih besar dari angka2
  Cetak " angka 1 lebih besar"
  Jika tidak
  Cetak " angka 2 lebih besar"

  Sebaiknya tidak dituliskan:

  If angka1>angka2
  Print "angka1"
  Else
  Print "angka2"

- *Stucture Chart*

  Menggambarkan struktur program dengan menunjukkan langkah-langkah hirarki secara independen. Program dibagi dalam informasi-informasi yang lebih kecil. Contoh structure chart seperti gambar 2.1

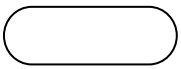
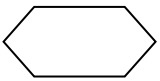
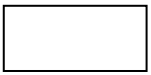
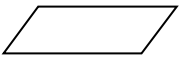
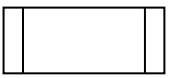
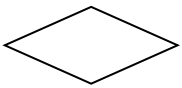


Gambar 1.3. Contoh Structure Chart

Flowchart

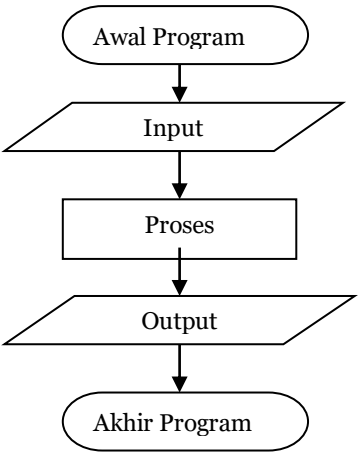
Flowchart adalah algoritma dalam suatu program, yang menyatakan arah alur program tersebut. Beberapa simbol yang digunakan untuk menggambar suatu *flowchart* seperti pada tabel 1.1.

Tabel 1.1 Simbol-simbol Flowchart

SIMBOL	NAMA	FUNGSI
	Terminator	Permulaan/akhir program
	Garis Alir (*Flow Line*)	Arah aliran program
	Preparation	Proses inisialisasi/pemberian harga awal
	Proses	Proses perhitungan/proses pengolahan data
	Input/Output Data	Proses input/output data, parameter, informasi
	*Predefined Process* (Sub Program)	Permulaan sub program/proses menjalankan sub program
	*Decision*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	*On Page Connector*	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	*Off Page Connector*	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Struktur Sederhana Flowchart

Bentuk sederhana sebuah program yang digambarkan oleh *flowchart* pada gambar 1.4. Sebuah flowchart harus mempunyai awal dan akhir program sebagai tanda dimana algoritma dimulai dan dimana diakhiri. Semua bentuk flowchart harus mempunyai aliran yang mengarah ke akhir. Tidak boleh terdapat simbol flowchart yang menggantung.



Gambar 1.4. Contoh alur program sederhana

Contoh : Konversikan jarak dari satuan mile menjadi km.

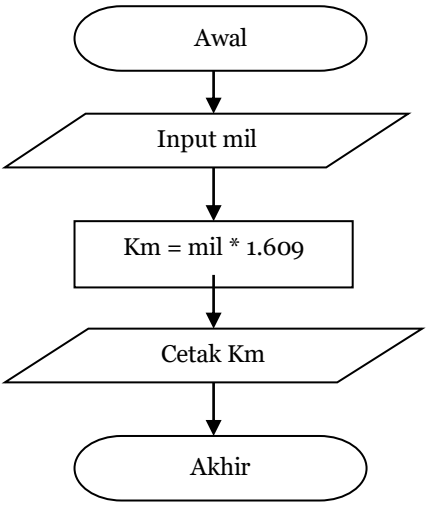
Data & rumus yang diperlukan :

Input : mil
Output : km
Rumus: 1mil = 1,609 km

Algoritma:

1. Jarak dalam mil
2. Konversikan ke km
3. Menampilkan jarak dalam km

Flowchart:

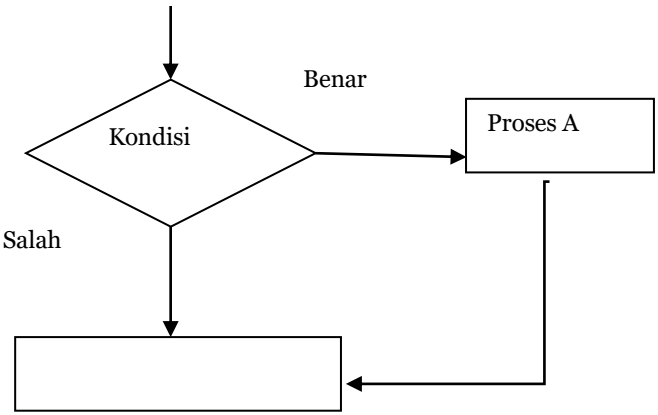


Gambar 1.5. Flowchart konversi jarak

Struktur Kondisi

Struktur kondisi dari *flowchart* diperlukan ketika terdapat suatu kondisi yang mempunyai dua pilihan operasi. Kondisi tersebut akan bernilai benar atau salah. Berikut adalah beberapa bentuk struktur dari struktur kondisi:

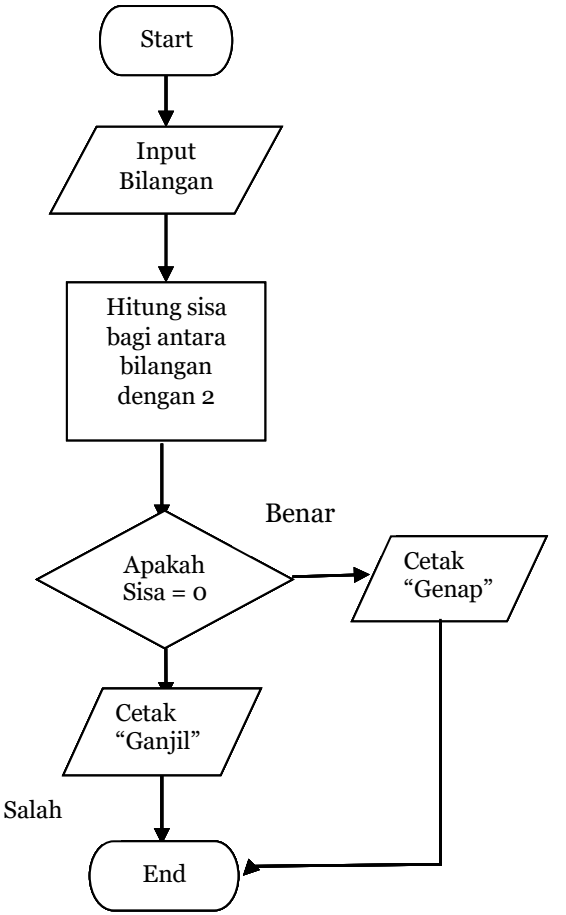
1.



Gambar 1.6. Flowchart Kondisi untuk kondisi benar saja

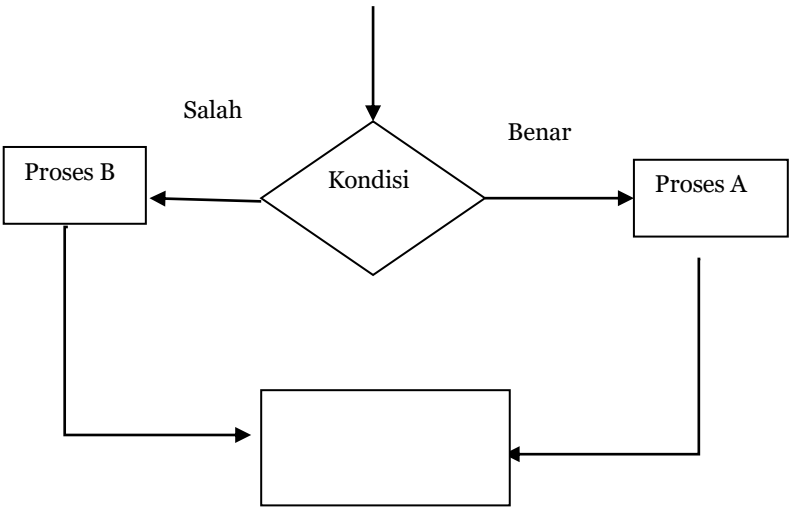
Struktur ini terjadi ketika terdapat sebuah proses yang dilakukan jika suatu kondisi tertentu dipenuhi. Tetapi jika kondisi tidak terpenuhi maka alur program langsung menuju ke pernyataan berikutnya

Contoh:
Menentukan apakah suatu bilangan adalah bilangan ganjil atau bilangan genap.



Gambar 1.7. Flowchart menentukan bilangan ganjil genap

2. Struktur kondisi dimana terdapat suatu kondisi ketika kondisi bernilai benar maka ada sebuah proses tertentu yang harus dilakukan, jika bernilai salah maka terdapat proses lain yang harus dilakukan.



Gambar 1.8. Flowchart Kondisi untuk kondisi benar dan salah

Contoh:
Menentukan apakah seorang mahasiswa lulus atau tidak berdasar syarat kelulusan mahasiswa nilai > 50.

Data & rumus yang diperlukan :

Input : nilai

Output : hasil seleksi

Rumus: nilai > 50 lulus

Desain:

Algoritma:

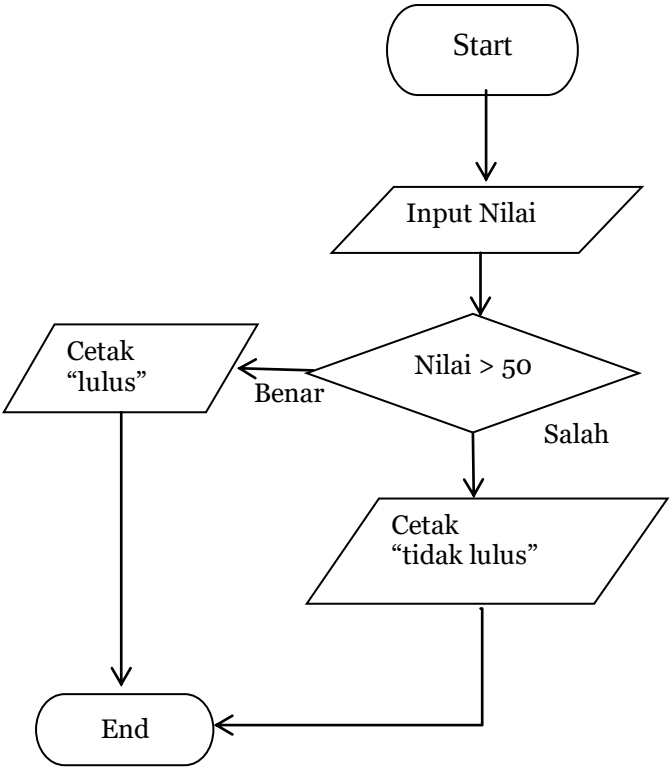
Input nilai

Seleksi apakah nilai > 50

Jika benar cetak lulus

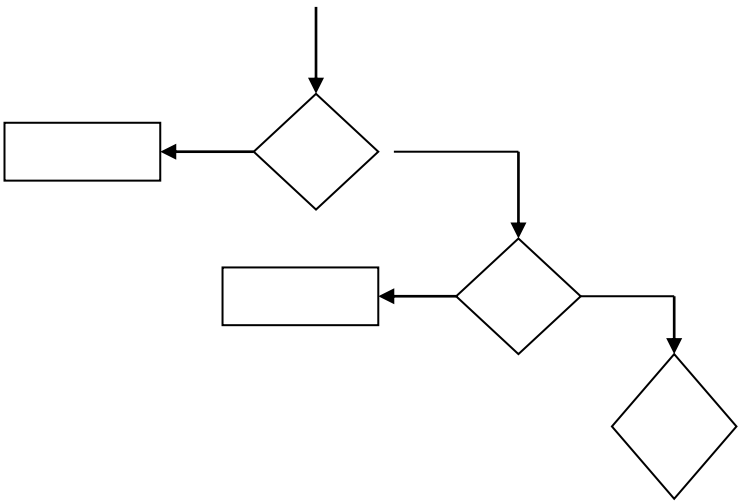
Jika salah cetak tidak lulus

Flowchart:



Gambar 1.9. Flowchart Kondisi syarat kelulusan

3. Dalam suatu pemecahan masalah dapat terjadi dimana terdapat kondisi yang bertingkat, hal ini terutama jika terdapat banyak situasi yang harus dibandingkan untuk mengerjakan suatu proses tertentu. Salah satu bentuk *flowcha*rt adalah sebagai berikut:



Gambar 1.10. Flowchart Kondisi Bersarang

Contoh:

Mencari nilai terbesar dari 3 bilangan pada variabel A, B, C

Data & rumus yang diperlukan :

Input : 3 nilai

Output : nilai terbesar

Rumus: bandingkan masing masing nilai

Desain:

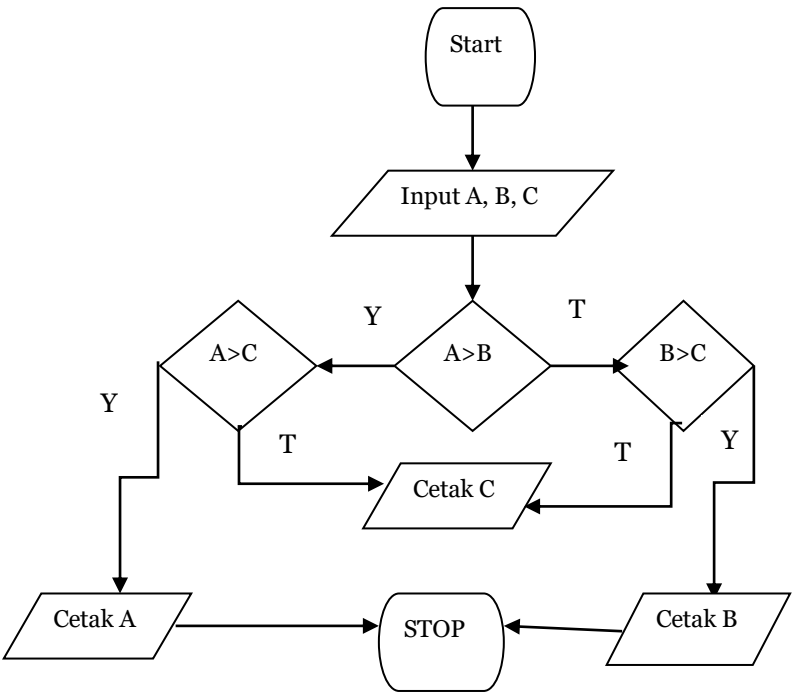
Algoritma:

1. Input 3 buah nilai
2. Bandingkan nilai ke 1 dan ke 2
   a. jika nilai ke 1 lebih besar, bandingkan nilai ke 1 dan ke 3 (A>B)

- jika nilai ke 1 lebih besar maka nilai ke 1 adalah nilai terbesar (A>B, A>C)
- jika nilai ke 3 lebih besar maka nilai ke 3 terbesar (A>B, C>A□ C>A>B)

b. Jika nilai ke 2 lebih besar, bandingkan nilai ke 2 dan ke 3 (B>A)
   - jika nilai ke 2 lebih besar maka nilai ke 2 adalah nilai terbesar (B>A, B>C)
   - jika nilai ke 3 lebih besar maka nilai ke 3 terbesar (B>A, C>B→C>B>A)

Flowchart

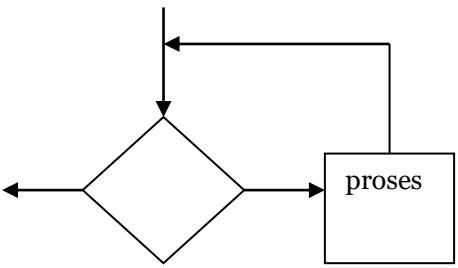


Gambar 1.11. Flowchart mencari nilai terbesar

Struktur Perulangan

Struktur berulang merupakan struktur ketika terdapat kondisi tertentu maka perlu dilakukan pengulangan terhadap suatu operasi. Berikut adalah 2 bentuk dari struktur berulang:

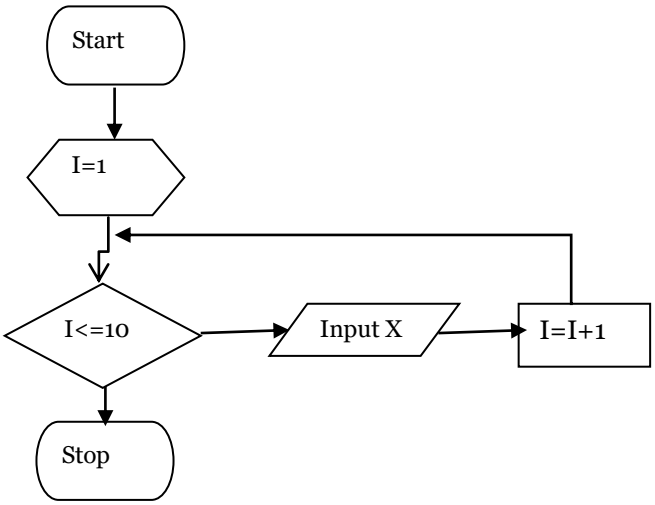
1.



Gambar 1.12. Flowchart perulangan kondisi di awal

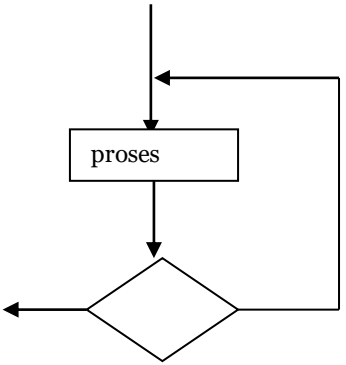
Pada bentuk ini, pemeriksaan kondisi dilakukan diawal sebelum operasi.Jika kondisi terpenuhi maka operasi boleh dikerjakan dan berulang sampai kondisi bernilai FALSE. Jika diawal kondisi sudah bernilai FALSE maka operasi tidak akan pernah dikerjakan

Contoh : Flowchart untuk memasukkan 10 bilangan.



Gambar 1.13. Flowchart input 10 bilangan kondisi diawal

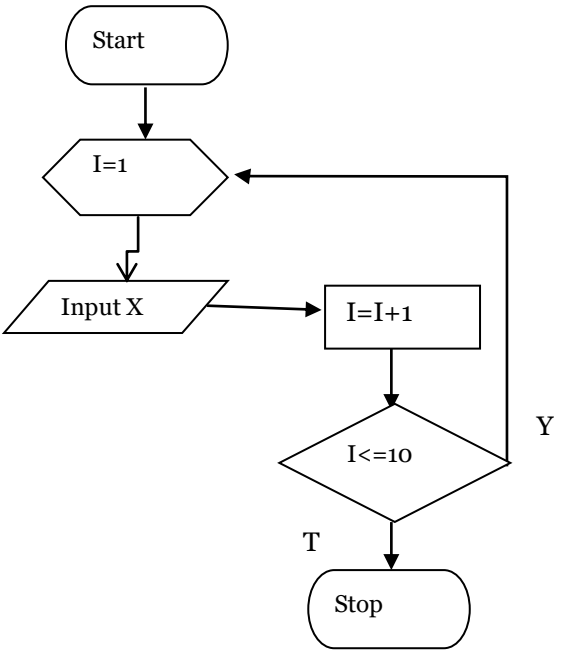
2.



Gambar 1.14. Flowchart perulangan kondisi di akhir

Pada bentuk ini, pemeriksaan kondisi dilakukan setelah proses dikerjakan. Proses akan berulang selama kondisi terpenuhi. Pada bentuk ini proses pasti dikerjakan minimal 1 kali.

Contoh: Flowchart untuk memasukkan 10 bilangan.

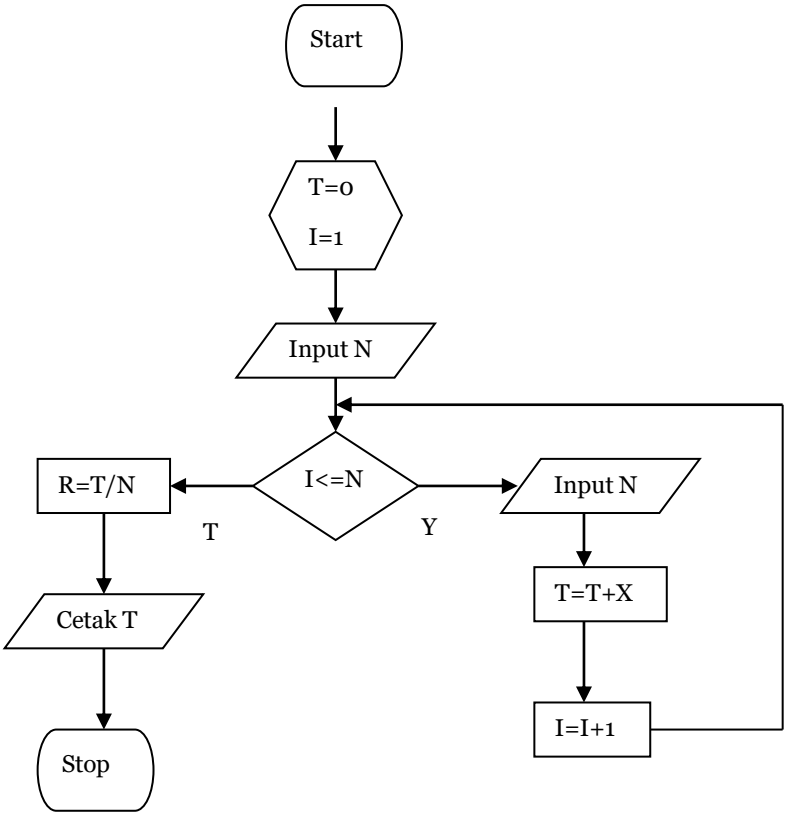


Gambar 1.15. Flowchart input 10 bilangan kondisi di akhir

Contoh :

Menghitung nilai rata-rata dari sekelompok data yang dimasukkan.
Variabel yang digunakan:
X = nilai data
N = banyaknya data
T = total nilai data
R = rata-rata
I = menghitung banyaknya pengulangan

Maka flowchartnya dapat digambarkan sebagai berikut:



Gambar 1.16. Flowchart nilai rata-rata

Rangkuman

Program adalah kumpulan langkah-langkah instruksi yang mengatur komputer untuk mengerjakan tugas yang diinginkan. Untuk membuat sebuah program diperlukan bahasa pemrograman yang merupakan sekumpulan aturan untuk memberitahu komputer tentang operasi apa yang harus dilakukan. Program dapat memberi instruksi komputer agar membaca input, mengerjakan tugas secara berurutan, menghitung, menyimpan data dan menulis output. Diagram alir adalah salah satu metode penggambaran atau mempresentasikan pemecahan masalah.

Latihan

1. Tuliskan algoritma untuk menentukan kelulusan seorang mahasiswa matakuliah Pemrograman Dasar berdasarkan nilai ujian yang didapat. Syarat lulus adalah nilai harus diatas 50.

2. Buatlah flowchart untuk mengkonversikan ukuran panjang dalam centimeter menjadi meter dan kelebihannya dalam centimeter (misalnya panjang 165 cm, maka hasil konversi adalah 1 m 65 cm).

3. Buat flowchart untuk membuat deret fibonacci sebanyak n input.
   Cont : jika n = 4 maka deret = 1 1 2 3

4. Buat flowchart untuk mencari deret bilangan yang habis dibagi 5 dengan batas yang diinputkan.
   Contoh : input bilangan : 19 maka deret : 5 10 15

5. Untuk memperkirakan biaya mengecat sebuah rumah, seorang tukang cat perlu mengetahui luas permukaan yang akan di cat. Buat algoritma untuk menghitung luas permukaan tersebut. Input adalah lebar, panjang, dan tinggi rumah, jumlah jendela dan pintu, dan ukurannya. (Asumsikan jendela dan pintu memiliki ukuran seragam.)

2

Elemen Dasar C++

Dalam bab ini akan dibahas mengenai struktur dasar bahasa pemrograman C++, variable, tipe data serta operasi aritmetika, operasi relasiona dan operasi logika. Diberikan juga contoh sederhana penggunaan tipe data, variabel, operasi-operasi aritmetik, relasional serta logika.

Struktur Pemrograman C++

Pemrograman dalam Bahasa C++ adalah program yang terdiri dari satu / lebih fungsi. Dan ada satu fungsi yang harus ada, yaitu fungsi `main()`. Fungsi `main()` merupakan fungsi utama, dimana eksekusi program dimulai. Susunan perintah diberikan pada badan fungsi. Badan fungsi ditandai oleh `{ }` dan berisi serangkaian perintah. Setiap perintah harus diakhiri dengan `;`. Berikut adalah bentuk umum dari program bahasa C++ :

```
# include file

# define var konstan

 deklarasi fungsi

  main()
 {
        deklarasi variabel global
        :
        <pernyataan - pernyataan>
        :
}

nama fungsi (arg1, arg2,...)
{
        deklarasi variabel lokal
        :
        < pernyataan - pernyataan>
        :
 }
```

Keterangan:

#include file :

merupakan nama file library yang fungsinya akan digunakan dalam program.

#define var konstan :

mendefinisikan sebuah nilai sebagai konstanta, dimana dalam program nilainya selalu tetap.

Deklarasi fungsi:

mendeklarasikan fungsi apa saja yang akan dipakai dalam program, selain fungsi main().

main() :

merupakan fungsi induk yang harus ada dalam program.

Deklarasi variabel:

sebelum sebuah variabel digunakan dalam program maka harus dideklarasikan yang maksudnya memesan tempat dulu pada memori. Banyaknya tempat yang dipesan tergantung dari tipe data apa yang digunakan oleh variabel tersebut. Karena itu didalam deklarasi variabel terdapat nama variabel dan tipenya.

Contoh program C sederhana seperti berikut :

```
#include <iostream>
using namespace std;
// main() merupakan awal eksekusi program.
int main()
{
      cout << "Hello World"; // cetak Hello World
      return 0;
}
```

Baris pertama pada program diatas adalah header `<iostream>`. Bahasa pemrograman C++ mendefinisikan beberapa header yang berisi informasi yang berguna untuk program. Baris berikutnya, `using namespace std;` memberi perintah pada kompailer untuk menggunakan namespace `std`. Dalam ANSI / ISO Standard C ++, perintah cin dan cout yang merupakan perintah input dan output dinyatakan dalam file header iostream, tetapi didalam namespace. Nama namespacenya adalah std.

Selanjutnya `// main() merupakan awal eksekusi program.` adalah komentar dalam C++. Komentar dengan tanda // digunakan untuk komentar dalam satu baris. `int main()` adalah fungsi utama yang harus ada pada program, dimana eksekusi program dimulai. Berikutnya, perintah `cout << "Hello World";` mencetak tulisan “hello world” dilayar. Baris terakhir perintah `return 0;` mengembalikan 0 dan mengakhiri fungsi utama.

File Header

File header berisi definisi Fungsi dan Variabel, yang diimpor atau digunakan ke program C ++ dengan menggunakan pernyataan #include pra-prosesor. File header memiliki ekstensi ".h" yang berisi deklarasi fungsi C ++ dan definisi makro.

Setiap file header berisi informasi (atau deklarasi) untuk kelompok fungsi tertentu. Seperti file header `stdio.h` berisi deklarasi fungsi input dan output standar yang tersedia di C ++ yang digunakan untuk mendapatkan input dan mencetak output. Demikian pula, file header math.h berisi deklarasi fungsi matematika yang tersedia di C ++. Terdapat 2 tipe file header, yaitu : *System header files* merupakan file header dari kompailer dan *User header files* yang ditulis oleh programer.

File header digunakan ketika program memerlukan fungsi yang ada pada C++. Untuk mengimpor fungsi dari *library* yang ada pada C++ diperlukan file header. File header dituliskan di awal program dengan menggunakan `#include`. Contoh, jika ingin menggunakan `clrscr()` dalam program C++ maka diperlukan file header `conio.h`. Hal ini karena definisi fungsi `clrscr()` dituliskan dalam file header `conio.h`.

Berikut contoh beberapa file header dalam C++:

1. `include "iostream.h"`

Beberapa fungsi yang terdapat dalam iostream.h adalah :

a. `cout` merupakan fungsi keluaran yang digunakan untuk menampilkan data ataupun informasi.
b. `cin` merupakan fungsi masukkan yang digunakan untuk menyimpan data dalam suatu variable.
c. `endl` , fungsi yang digunakan untuk berpindah baris.

Contoh program :

```
#include "iostream.h"
void main(){
    char nama[30];
    cout<<"Masukkan nama : ";  cin>>nama;
    cout<<"Nama anda adalah "<<nama;
}
```

2. include "stdio.h"

Terdiri dari perintah-perintah berikut :

a. printf merupakan fungsi keluaran untuk menampilkan informasi
b. scanf merupakan fungsi masukkan yang digunakan menyimpan data pada suatu variable.
c. gets merupakan fungsi masukkan seperti scanf, namun fungsi ini dapat membaca karakter spasi tidak seperti scanf.

Contoh Program :

1)
```
#include "stdio.h"
void main()
{
    char nama[30];
    printf("\nMasukkan nama: ");
    scanf("%s",nama);
    printf("Nama anda adalah %s",&nama);
}
```

2)
```
#include " stdio.h"
void main()
{
    char nama[30];
    printf("\nMasukkan nama: ");
    gets(nama);
    printf("Nama anda adalah %s",nama);
}
```

3. include "conio.h"

Menjalankan perintah-perintah sebagai berikut :

a. getch merupakan fungsi untuk input sebuah karakter, karakter tidak ditampilkan dilayar dan tidak perlu menekan tombol enter untuk melanjukan ke perintah berikutnya.
b. clrscr merupakan fungsi untuk membersihkan layar.
c. getche merupakan fungsi untuk input sebuah karakter, karakter yang dimasukan ditampilkan di layar dan tidak perlu diakhiri dengan menekan tombol enter.
d. putch merupakan fungsi untuk menampilkan karakter ASCII dari nilai x ke layar monitor tanpa memindahkan letak kursor ke baris berikutnya.

e. `clreol` merupakan fungsi untuk membersihkan layar mulai dari posisi kursor hingga kolom terakhir, posisi kursor tidak berubah.
f. `gotoxy` merupakan fungsi untuk memindahkan kursor ke kolom x, baris y.
g. `wherex` merupakan fungsi untuk mengembalikan posisi kolom kursor.
h. `wherey` merupakan fungsi untuk mengembalikan posisi baris kursor.
i. `window` merupakan fungsi untuk mendefinisikan sebuah window berdasarkan koordinat kiri atas dan kanan bawah.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{
     char nama[30];
     printf("\nMasukkan nama : ");
     gets(nama);
     clrscr();
     printf("Nama anda %s",nama);
     getch();
}
```

Ada 2 macam cara untuk mencantumkan file header kedalam program:

1. #include<file>

Bentuk ini digunakan untuk *system header files*. Yang akan mencari file kedalam daftar file pada kompailer.

2. #include"file"

Bentuk ini untuk header file yang dibuat oleh programmer. File akan dicari pada driver yang aktif saat itu.

Comment

Komentar program adalah pernyataan penjelasan yang dapat disertakan dalam kode C ++. Komentar ini membantu memberi penjelasan kode program. Semua bahasa pemrograman menyediakan fasilitas komentar. C ++ mempunyai 2 macam bentuk komentar, yaitu komentar *single-line* dan *multi-line*. Semua karakter yang berada pada komentar dalam bentuk apapun diabaikan oleh kompiler C ++.

Komentar dimulai dengan tanda /* dan diakhiri dengan */. Contoh:

```
/* Komentar */
/* Komentar pada C++ dapat berupa
   Multi-line
*/
```

Selain itu komentar dapat juga menggunakan tanda //. Dengan tanda // komentar hanya bisa single-line. Contoh:

```
#include <iostream>
using namespace std;
main()
{
cout << "Hello World"; // cetak Hello World
return 0;
}
```

Ketika program dikompail, maka `// cetak Hello World` akan diabaikan dan akhir ekseskusi akan menampilkan tulisan :

Hello World

Tipe Data

Setiap nama dan ekspresi mempunyai tipe yang menentukan operasi yang dapat dilakukan. Contoh, deklarasi : `int meter;` menunjukkan bahwa `meter` bertipe `int`, artinya `meter` merupakan variabel yang menyimpan data integer. Deklarasi adalah suatu perintah untuk mengenalkan nama objek beserta tipe data yang tersimpan pada nama tersebut kedalam program. Variabel merupakan nama untuk objek yang membawa nilai dengan tipe data tertentu.

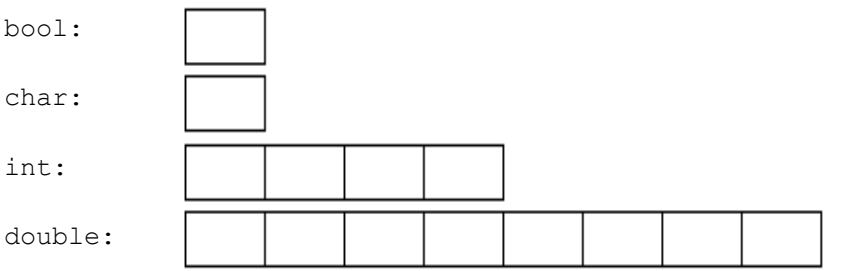
Tipe data mendefinisikan jenis nilai dan operasi yang dapat dilakukan terhadap nilai tersebut. Tipe data menentukan besarnya memori yang harus di alokasikan untuk tipe tersebut. Tipe data berbeda dapat mempunyai jumlah bit yang berbeda.

Ada dua macam tipe data dalam C++, konstan dan variabel. Perbedaan antara tipe data tersebut adalah, untuk konstan, kompilator bisa menentukan tipenya dan untuk variabel, pengguna harus mendeklarasikan tipenya.

C++ mempunyai beberapa tipe data dasar yang umum digunakan untuk menyimpan data, yaitu :

Boolean (bool)
    Bernilai true atau falase
Karakter (char dan wchar_t)
    Contoh : 'a', 'z', '9'
Integer (int dan long long)
    Contoh : 1, 42, dan 1066
Floating-point (double dan long double)
    Contoh : 3.14 dan 299793.0

Masing-masing tipe data berhubungan secara langsung dengan hardware dan akan menentukan ukuran tertentu setiap tipenya. Berikut contoh ukuran alokasi tipe data dasar alam memori:



Setiap tipe data mempunyai objek data masing-masing. Objek data setiap tipe data mempunyai jangkauan nilai tertentu yang akan menentukan berapa besar memori yang harus diaolkasikan untuk tipe data tersebut. Berikut adalah daftar banyaknya bit dan jangkauan nilai dari setiap tipe data:

Tipe	Jumlah bit	Jangkauan
char	1byte	-127 s/d 127 atau 0 s/d 255
unsigned char	1byte	0 s/d 255
signed char	1byte	-127 s/d 127
int	4bytes	-2147483648 s/d 2147483647
unsigned int	4bytes	0 s/d 4294967295
signed int	4bytes -	2147483648 s/d 2147483647
short int	2bytes	-32768 s/d 32767
unsigned short int	Range	0 s/d 65,535
signed short int	Range	-32768 s/d 32767
long int	4bytes	-2,147,483,647 s/d 2,147,483,647
signed long int	4bytes	sama dengan long int
unsigned long int	4bytes	0 s/d 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Ukuran tipe data bisa berbeda dari daftar diatas, tergantung dari kompailer dan computer yang digunakan.

Boolean

Boolean (bool) mempunyai nilai true atau false. Boolean digunakan untuk mengekspresikan hasil operasi logika.

Contoh :

```
void f(int a, int b)
{
      bool b1 {a==b};
      // ...
}
```

Jika a dan b mempunyai nilai sama, maka b1 bernilai true. Jika tidak sama, b1 bernilai false. Umumnya bool digunakan sebagai hasil dari fungsi yang melakukan tes pada beberapa kondisi.
Contoh:

```
bool is_open(File*);
bool greater(int a, int b) { return a>b; }
```

Nilai Boolean dapat dikonversikan kedalam nilai integer. Untuk nilai true memiliki nilai 1 ketika dikonversi ke integer dan false memiliki nilai 0. Sebaliknya, integer dapat secara implisit dikonversikan ke nilai bool: integer selain nol dikonversikan menjadi true dan 0 dikonversikan ke false.
Contoh:

```
bool b1 = 7;       // 7!=0, b→true
bool b2 {7};       //  error,  jumlah  bit  int
                   // lebih besar dari bool
int i1 = true;     // i1 → 1
int i2 {true};     // i2 →  1
```

Dalam operasi aritmetik dan logika pada suatu variabel bertipe boolean, nilai boolean akan dikonversikan menjadi nilai integer. Jika hasil operasi dikonversikan kembali ke nilai boolean, maka nilai selain nol menjadi true dan nilai nol menjadi false.
Contoh:

```
bool a = true;
bool b = true;
bool x = a+b;  // a+b adalah 2, x → true
bool y = a||b; //a||b adalah 1,y→true(||->or)
bool z = a-b;  // a-b adalah 0, z → false
```

Tipe Int, Long, Short

Tipe int, long dan short adalah bilangan bulat. Tipe ini merupakan integer bertanda : yakni bisa bilangan bulat negatif, nol atau positif. Jika user tidak ingin detail bisa digunakan tipe int. Satu bit dalam integer bertanda digunakan untuk indikasi tanda, sedangkan pada integer tak bertanda (unsigned integer); hanya diperbolehkan bilangan nol dan positif. Jangkauan dari integer bertanda - -2147483648 s/d 2147483647; sedangkan integer tak bertanda 0 s/d 4294967295.

Sebuah variabel yang mempunyai tipe integer sebelum digunakan harus dideklarasikan terlebih dahulu. Berikut adalah contoh mendeklarasikan variabel dengan tipe integer:

```
int panjang;
short suhu; atau   short int suhu;
long gaji ; atau   long int gaji ;
int lebar, tinggi, diameter;
```

Agar sebuah variabel integer dapat digunakan dalam sebuah operasi dapat diberi nilai terlebih dahulu atau dikenal sebagai inisialisasi. Inisialisasi variabel dengan tipe integer seperti dalam contoh berikut:

```
panjang     = 100;
suhu        =  -3;
gaji        = 1234560;
```

Selain itu, inisialisasi dapat dilakukan ketika deklarasi seperti dalam contoh berikut:

```
int lebar = 23;
int diameter = 32, jari2 = 14;
int tinggi, luas = 92;
```

Tipe Float dan Double

Tipe float dan double merupakan bilangan real. Tipe ini biasanya menggunakan 32 bit untuk menyimpannya, 8 bit untuk nilai dan tanda eksponen, 24 bit untuk bagian non eksponen. Tipe float menghasilkan presisi 6 atau 7 digit desimal dan jangkauan (10-37 s/d

10+38) sedang tipe double digunakan utk memperoleh presisi ganda, biasanya digunakan 2x32 bit untuk penyimpanan.

Sama seperti dalam variabel integer, sebelum digunakan variabel float juga harus dideklarasikan. Bentuk deklarasi variabel float seperti contoh berikut:

```
float       rata2,luas;
double      bonus;
float       hasil = 6.63e-34;
```

Tipe Char

Tipe char merupakan karakter dimana dalam prosesnya menghasilkan integer tak bertanda dalam range 0 – 255. Tipe ini disimpan dalam 1 byte dan sebagian besar menggunakan kode ASCII. Contoh : huruf A dalam ASCII konversinya adalah 65 (desimal).

Sebuah nilai karakter dituliskan dalam diapit oleh single quote (' '). Misalnya variabel `jenis` mempunyai tipe char, maka bisa dituliskan :

```
jenis = 'T';
```

tidak bisa dituliskan

```
jenis = T;
```

Variabel atau konstan char hanya boleh mempunyai nilai sebuah karakter. Contoh :

```
char status;
bovine = 'ox'; /*salah*/
```

Ada beberapa karakter tidak dapat dicetak, contoh : `'\007'` menyebabkan terminal berbunyi. Berikut adalah beberapa karakter yang tidak tercetak seperti aslinya:

1. Menggunakan kode ASCII nya didahului backslash.

contoh : `beep = '\007';`

2. Mengunakan "escape sequence";

\n newline

\t tab (default 8 karakter)

\b	backspace
\r	carriage return
\f	formfeed
\\	backslash (\)
\'	single quote (')
\"	double quote (")
\xoa	kode ASCII dalam hexadesimal
\aaa	kode ASCII dalam oktal

disebut escape sequence karena notasi '\' dianggap sebagai karakter "escape" (menghindar) dalam arti bahwa karakter yang terdapat setelah tanda '\' dianggap bukan merupakan teks biasa.
Contoh :

```
char nerf;
nerf = '\n';
```

Contoh program yang menggunakan escape sequence:

```
main()
{
      clrscr();
      printf("\'DPR\'singkatan dari\"Dewan
      Perwakilan Rakyat\"");
}
```

Output:

'DPR' singkatan dari Dewan Perwakilan Rakyat

Berikut adalah contoh dari deklarasi variabel karakter :

```
char tombol;
char huruf, vokal;
char tekan = 'Y';
```

Variabel

Suatu variabel digunakan dalam program untuk menyimpan suatu nilai dan nilai yang ada padanya dapat berubah selama proses program berlangsung.

Dalam membuat nama variabel, C++ mempunyai peraturan sebagai berikut:

1. Karakter pertama berupa huruf (A-Z) atau (a-z) atau underscore (_)
   contoh : `nama1, kode2` (benar)
   `_gaji_pokok` (benar)
   `1nama, 2kode` (salah)

2. Tidak boleh mengandung simbol khusus kecuali underscore (_)
   contoh : `nama_mahasiswa` (benar)
   `kode-wilayah` (salah)

3. Nama variabel tidak boleh sama dengan kata kunci C++
   contoh : `auto, break, main, printf` (salah)

4. Nama variabel boleh terdiri dari kata kunci (reserved word) yang digabung dengan kata lain .
   contoh : `char_pertama, data_float`

5. Huruf besar dan kecil dari nama variabel akan dibedakan oleh bahasa C
   contoh : `nama,Nama,NAMA` adalah 3 variabel yang berbeda

6. Nama variabel tidak boleh menggunakan operator aritmetika (+ - / * %)
   contoh : `jumlah+anak, potongan%` (salah)

7. Nama variabel tidak boleh mengandung spasi
   contoh : `gaji pokok` (salah)

Mendeklarasikan Variabel

Variabel yang akan digunakan dalam program harus dideklarasikan terlebih dahulu, tanpa adanya deklarasi variabel kompailer tidak akan menerima variabel tersebut. Tipe variabel harus ditentukan sebagai informasi pada kompailer jenis entity tersebut.

Pengertian deklarasi di sini berarti memesan memori dan menentukan jenis data yang bisa disimpan di dalamnya. Deklarasi variabel meliputi tipe variabel seperti integer(int), floating point (float), double precision (double) dan character (char) dan nama variabel itu sendiri.

Bentuk deklarasi variabel :

```
tipe-variabel  nama-variabel;
```

Pada deklarasi variabel, nama-variabel dapat berupa sebuah variabel atau beberapa variabel yang sama yang dipisahkan dengan koma.

Contoh :

```
int  a;
int i, j;
long k; atau long int k;
short c; atau short int c;
float m;
double n; sama dengan long float n;
```

Variabel dengan nama yang sama TIDAK BOLEH dideklarasikan ulang.

Contoh :

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

Program ini akan memberikan pesan kesalahan redeclaration, yang berarti adanya deklarasi ulang atas variabel yang sama

Inisialisasi Variabel

Untuk memberikan nilai ke variabel yang telah dideklarasikan, bentuk pernyataannya sebagai berikut :

```
variabel = nilai;
```

Contoh :

```
main(){
      float harga,jumlah,hrg_tot;   /*deklarasi*/
      clrscr();
      jumlah     = 10;              /*inisialisai*/
      harga       = 15.50;          /*inisialisai*/
      hrg_tot    =  jumlah * harga;
      printf ("harga total : %f",hrg_tot);
}
```

Variabel `harga` dan `jumlah` yang bertipe float lebih dulu akan diberi nilai sebesar 15.50 dan 10 sebelum digunakan untuk menghitung deklarasi `hrg_tot` yang juga bertipe `float`. Fungsi `clrscr()` digunakan untuk menghapus/membersihkan layar. fungsi ini harus ditulis setelah deklarasi variabel.

Deklarasi nilai ke variabel dilakukan pada :

1.Pada saat deklarasi variabel

```
int     a = 8;
float a =  8.0;
char a = 'A';
```

2. Diluar deklarasi variabel

```
int a;
a = 8;
```

Operator Aritmetik, Relasional dan Logika

Operator adalah simbol untuk memberi perintah pada kompailer untuk melakukan manipulasi matematika atau logika tertentu. C++ mempunyai banyak jenis operator, yaitu operator aritmetik, relasional, logika, bitwise, assignment dan Misc.

Operator Aritmetik

Beberapa operator dalam bahasa pemrograman C++ antara lain:

1. Operator untuk operasi aritmatika yang tergolong sebagai operator binary adalah :

   - * simbol operasi perkalian, untuk perkalian antara 2 operand.

     Contoh : `A = B * C;`

   - / simbol operasi pembagian, untuk membagi operand pertama dengan operand kedua

     Contoh : `A = B / C;`

   - % simbol operasi sisa pembagian, menghitung sisa pembagian operand pertama dibagi operand kedua.

     Contoh : A = 10 % 3;

     Maka A akan mendapatkan hasil 1 karena 10 dibagi 4 hasilnya 3 dan sisanya 1

   - + simbol operasi penjumlahan antara operand pertama dengan operand kedua.

     Contoh : `A = B + C;`

   - - simbol operasi pengurangan dimana operand pertama dikurangi dengan operand kedua

     Contoh: `A = B – C;`

   - ++ merupakan operator kenaikan. Menambahkan variabel dengan 1.

`x = x+1;` Dapat ditulis menjadi : `++x;` atau `x++;`

- `--` operator penurunan merupakan operator untuk mengurangi nilai variabel dengan 1

  `y = y-1;` Dapat ditulis menjadi `--y;` atau y--;

2. Operator yang tergolong sebagai operator unary : (-) tanda minus dan (+) tanda plus

   Contoh : `A = -10 + 12;`

Untuk operator – dan ++ , jika digunakan pada operasi yang melibatkan variabel itu sendiri, penulisan ++ dan – didepan atau dibelakang variabel tidak akan mempengaruhi hasil. Tetapi jika melibatkan variabel lain maka akan terlihat perbedaannya. Contoh:

```
void main(){
   int c,y;
      c = 10;
   y = ++c; printf("y = %d, c= %d\n", y,c);
   y = c++ ;  printf("y = %d, c= %d\n", y,c);
}
```

Hasilnya : y=11 c =11
y =11 c=12

Pada perintah `y = ++c,` nilai c dinaikkan 1 dulu. Kemudian nilainya disimpan pada variabel y.
Sedang pada perintah `y = c++,` nilai c disimpan dulu oleh y. Setelah itu baru nilai c dinaikan 1.

Penulisan operator aritmetik yang binary dapat dituliskan dengan cara yang lebih singkat seperti berikut:

`x += 2;`	kependekan dari	`x = x + 2;`
`x -= 2;`	kependekan dari	`x = x - 2;`
`x *= 2;`	kependekan dari	`x = x * 2;`
`x /= 2;`	kependekan dari	`x = x / 2;`
`x %= 2;`	kependekan dari	`x = x % 2;`

Operator Relasional

Ekspresi yang digunakan untuk membandingkan operand disebut ekspresi relasional. Ekspresi relasional sederhana terdiri dari operator relasional yang menghubungkan dua variabel atau operand konstan. Operator relasional dapat digunakan dengan data integer, Boolean, ganda, atau karakter. Berikut adalah operator relasional yang tersedia di C ++:

- < merupakan simbol lebih kecil
  Contoh : umur < 30

- > simbol lebih besar dari
  Contoh : tinggi > 6.2

- <= simbol lebih kecil atau sama dengan
  Contoh : harga <= 20000

- >= simbol lebih besar atau sasma dengan
  Contoh : temp >= 98.6

- == simbol sama dengan
  Simbol sama dengan (==) pada operator relasional berbeda dengan simbol assignment (=) pada operator aritmetik.
  Contoh :
  1. Operasi relasional : urutan == 100
     akan menghasilkan nilai 1 jika urutan bernilai 100 dan 0 jika urutan bukan 100.
  2. Sedangkan operasi assignment : urutan = 100
     Berarti variabel urutan diberi nilai 100.

- != simbol tidak sama dengan
  Contoh : bilangan != 250

Seperti semua ekspresi C ++, ekspresi relasional dievaluasi untuk menghasilkan hasil numerik. Suatu kondisi relasi akan ditafsirkan true jika bernilai 1, dan kondisi false jika bernilai 0.

Contoh seperti pada perintah berikut :

```
1. cout << "Nilai dari 3 < 4 adalah  " << (3 < 4);
```

Hasil dari 3 < 4 adalah 1 karena pernyataan 3 lebih kecil dari 4 adalah benar(*true*). Maka pernyataan tersebut akan mencetak :

Nilai dari 3 < 4 adalah 1

```
2. cout << "Nilai dari 2 > 3 adalah  " << (2 > 3);
```

Hasil dari 2 > 3 adalah 0 karena pernyataan 2 lebih besar dari 3 adalah salah(*false*). Maka pernyataan tersebut akan mencetak :

Nilai dari 2 > 3 adalah 0

Selain operand numerik, data karakter dapat dibandingkan dengan menggunakan operator relasional. Misalnya, dalam kode ASCII, huruf 'A' disimpan dengan menggunakan kode dengan nilai numerik lebih rendah daripada huruf 'B', kode untuk 'B' memiliki nilai lebih rendah daripada kode untuk 'C', dan begitu seterusnya. Berikut contoh operasi relasional dengan data karakter, perbandingan dilakukan berdasar kode ASCII masing-masing karakter:

'A' > 'C'	bernilai 0 karena pernyataan false
'D' <= 'Z'	bernilai 1 karena pernyataan true
'E' == 'F'	bernilai 0 karena pernyataan false
'g' >= 'm'	bernilai 0 karena pernyataan false
'b' != 'c'	bernilai 1 karena pernyataan true
'a' == 'A'	bernilai 0 karena pernyataan false

Operator Logika

Selain menggunakan ekspresi relasional sederhana sebagai kondisi, kondisi yang lebih kompleks dapat dibuat dengan menggunakan operator logika AND, OR, dan NOT. Dalam C++ operator-operator ini diwakili oleh simbol &&, ||, dan!.

Untuk operator AND yang menggunakan simbol &&, digunakan dengan dua ekspresi sederhana. Kondisi *true* hanya jika kedua ekspresi bernilai *true*. Contoh pada kondisi logika berikut:

```
(usia > 40) && (term < 10)
```

Kondisi akan true hanya jika `usia` lebih besar dari dan `term` lebih kecil dari 10. Karena operator relasional memiliki prioritas lebih tinggi daripada operator logika, tanda kurung dalam ekspresi logika ini bisa dihilangkan.

Operator OR yang menggunakan symbol ||, juga digunakan dengan dua ekspresi. Saat menggunakan operator OR, kondisi terpenuhi jika satu atau kedua ekspresi itu benar. Oleh karena itu, untuk contoh kondisi

```
(usia > 40)  ||  (term <10)
```

akan memberikan hasil *true* jika `usia` lebih dari 40, atau `term` kurang dari 10, atau kedua kondisi itu benar. Tanda kurung pada ekspresi relasional hanya untuk membuat pernyataan lebih mudah dibaca. Karena operator relasional memiliki prioritas lebih tinggi daripada operator logika, tanda kurung dapat dihilangkan.

Contoh potongan program dengan beberapa ekspresi yang menggunakan operator relasional dan operator logika:

```
int i, j;
double a, b, complete;

a = 12.0;
b = 2.0;
i = 15;
j = 30;
completeƒ=ƒ0.0;

a > b;
// menghasilkan nilai 1 atau true
(i == j) || (a < b) || complete;
// menghasilkan nilai 0 atau false
(a/b > 5) && (i <= 20);
// menghasilkan nilai 1 atau true
```

Operator NOT disimbolkan dengan !. Digunakan untuk mengubah ekspresi ke kondisi sebaliknya; yaitu, jika ekspresi memiliki nilai bukan nol (*true*), ekspresi dengan operator ! akan menghasilkan nilai nol (*false*). Sebaliknya, jika ekspresi *false* (memiliki nilai nol), maka ekspresi dengan operator ! akan menghasilkan *true* dan bernilai 1. Misalkan pada contoh berikut :

- angka 26 disimpan dalam variabel `usia`,
- ekspresi `usia > 40` akan memiliki nilai 0 (*false*),
- dan ekspresi `!(usia > 40)` akan memiliki nilai sebaliknya, yaitu nilai 1 (*true*).

Karena operator NOT digunakan dengan hanya satu ekspresi, maka termasuk dalam operator unary.

Operator relasional dan operator logika memiliki hierarki yang mirip dengan operator aritmatika. Berikut adalah urutan prioritas operator-operator relasional, logika dan aritmetik :

PRIORITAS	**OPERATOR**	**URUTAN**
Tertinggi	( )	dari kiri ke kanan
	! ++ -- + -	dari kanan ke kiri
	* / %	dari kiri ke kanan
	+ -	dari kiri ke kanan
	< <= > >=	dari kiri ke kanan
	== !=	dari kiri ke kanan
	&&	dari kiri ke kanan
	||	dari kiri ke kanan
Terendah	= += -= *= /= %=	dari kanan ke kiri

Rangkuman

Struktur program C terdiri dari satu atau lebih fungsi, salah satu diantaranya adalah `main()`. Fungsi terdiri dari sebuah header dan sebuah body. Header berisi praprosesor seperti #include dan nama fungsi. Body ditandai dengan {} dan berisi sekumpulan pernyataan yang masing-masing diakhiri dengan tanda (;).

Untuk menyimpan suatu nilai dalam program digunakan variabel dan nilai yang ada didalamnya dapat berubah selama program berjalan. Pemberian nama variabel harus mengikuti aturan yang ditetapkan oleh C++. Sebelum variabel digunakan, harus dideklarasikan terlebih dahulu. Tipe data pada variabel yang dituliskan saat deklarasi akan menentukan besarnya alokasi memori untuk variabel tersebut.

Untuk melakukan manipulasi matematika atau logika tertentu digunakan operator. C++ mempunyai beberapa jenis operator, yaitu operator aritmetik, relasional, logika, bitwise, assignment dan Misc.

Latihan

1. Dibawah ini adalah beberapa nama variabel. Nama variabel yang manakah yang dianggap valid oleh C++?

```
tank_capacity
namespace
something
length2
left foot
account
Success!
i
2nd_account
```

2. Jika T adalah *true* dan F adalah *false*, apakah hasil dari ekspresi berikut?

```
(((T&&F) || (!(T||F))) && ((!F) && ((!F)|| T)))
```

3. jika x = 5, y = 6, z = 4, dan w = 3.5, berapakah hasil dari persamaan berikut?
   a. (x + z) % y b. (x + y) % w c. (y + w) % x d. (x + y) *w
   e. (x % y) % z f. (y % z) % x g. (x *z) % y h. ((x *y) *w) *z

4. Deklarasikan variabel untuk masing-masing variabel berikut :
   a. Sebuah variabel untuk menyimpan nama seorang siswa.
   b. Sebuah variabel untuk menyimpan harga diskon.
   c. Sebuah variabel untuk menyimpan jumlah botol.
   d. Sebuah variabel untuk menyimpan jarak.
   e. Sebuah variabel untuk menyimpan skor tertinggi.

5. Koreksilah kesalahan pada program berikut!

   a.
```
#includef<iostream>
using namespace std;
int main()
{
      Width = 15
      Area = length * width;
      Cout << "Luas adalah "<<area
}
```

   b.
```
#include<iostream>
using namespace std;
int main()
{
    int length, width, area;
    area = length * width;
    length = 20;
    width = 15;
    cout << "Luas adalah " << area;
    return 0;
```

   c.
```
#include<iostream>
int main()
{
   int length = 20; width = 15, area;
   Length * width = area;
   cout << "luas adalah ",area;
   return 0;
}
```

Latihan Program

1. Carilah kesalahan dalam program di bawah ini :

```
main ()
{
   INT jumlah;

   /* PERHITUNGAN HASIL
   jumlah = 25 + 37 - 19;

   /* TAMPILKAN HASIL
   printf("Berapa hasil perhitungan 25 + 37 - 19
?\n);
   printf("Jawabannya adalah %d\n" jumlah);  }
```

2. Program di bawah ini seharusnya menampilkan output satu baris sbb :

   c * c = 25,000000

   Namun, belum berhasil karena masih ada beberapa kesalahan. Temukan minimal 3 kesalahan dalam program tersebut.

```
#include <Studio.h>
main ()
{
   float a, b, c;

   a = 3;
   b = 4.0;

   c = a * a + b * b
   printf("c * c = %d", c);
}
```

3. Berapakah nilai jawaban yang ditampilkan oleh program di bawah ini :

```
 #include <stdio.h>
 main()
 {
   int jawab, hasil;
```

```
    jawab = 100;
    hasil = jawab - 10;

    printf("Jawabannya adalah %d\n",hasil+ 6);
}
```

4. Tulislah program berikut, kemudian jalankan :

```
#include "stdio.h"
#include <Conio.h>
void main()
{
printf("Bahasa C\n");
getch();
}
```

Apakah hasilnya?

5. Tulislah program berikut, kemudian jalankan:

```
#include "stdio.h"
#include <Conio.h>
void main()
{
printf("Saya belajar);
printf("Bahasa C++\n")
getch();
}
```

Apakah ada pesan kesalahan? jika ada, perbaiki program sesuai pesan.

3

Input Output

Dalam bab ini akan dibahas mengenai beberapa jenis perintah-perintah input output dan perbedaan di setiap perintah tersebut. Penjelasan juga beserta contoh setiap perintah untuk memudahkan pemahaman.

Output

Fungsi printf()

Fungsi yang paling umum digunakan dalam menampilkan data.Berbagai jenis data dapat ditampilkan ke layar dengan memakai fungsi ini.

Bentuk umum :

```
printf( "string control",argumen1, argumen2,...)
```

String control adalah keterangan dalam tanda " " yang akan ditampilkan pada layar beserta identifier (spt %d, %f). String control terdiri dari 2 bentuk informasi:

1. Karakter-karakter yang akan di cetak secara literal.
2. Data identifier = conversion specification.

Argumen1, argumen2 dll adl sesuatu yg akan mensubstitusi identifier bisa berupa variabel, konstanta, atau ekspresi / ungkapan yang dievaluasi dahulu sebelum nilainya dicetak.
Contoh :

```
main() {
    int a = 10;

    printf("saya belajar C++");
    //argumen string
    printf('a');
    //argumen konstanta karakter
    printf("%d",70);
    //argumen konstanta integer
    printf("%d",a);
    //argumen variabel
    printf("%d",a+70);
    //argumen ekspresi
}
```

Berikut adalah contoh program dengan instruksi printf dimana digunakan argumen didalamnya :

```
#include <stdio.h>

main(){
    unsigned int segmen = 0xb880;
    printf("nilai segmen grafik (oktal):%o\n",
    segmen);
    printf("nilai segmen grafik (desimal):%u\n",
    segmen);
    printf("nilai segmen grafik (heksadesimal):
    %x\n",segmen);
}
```

OUTPUT :

nilai segmen grafik (oktal) : 134200
nilai segmen grafik (desimal) : 47232
nilai segmen grafik (heksadesimal) : b880

Tabel 3.1. Daftar string control

IDENTIFIER	OUTPUT
%d	integer bertanda dalam bentuk desimal
%c	karakter tunggal
%s	string
%e	bil.floating, notasi dengan e (eksponensial)
%f	bil.floating,notasi desimal
%g	bilangan floating / real, gunakan %f atau %e
%u	integer desimal, unsigned
%o	integer oktal unsigned
%x	integer heksadesimal unsigned

Desimal dan panjang field yang diberikan sebenarnya dapat diatur. Cara mengatur lebar field dan jumlah desimal yang ingin dicetak cukup dengan memberikan format tambahan pada %f .

Contoh program:

```
main()
{
    float bil=2.5 , nomor = 30.756;
    clrscr();
    printf ('bilangan = %f \n",bil);
    printf ("nomor    = %f",nomor);
}
```

Output:

bilangan = 2.500000
nomor = 30.756001

Dari contoh diatas jika lebar fieldnya diatur dapat menjadi seperti contoh berikut:

```
main()
{
    float bil = 2.5, nomor = 30.756;
    clrscr();
    printf ("bilangan = %10.2f \n",bil);
    printf ("nomor     = %10.2f, nomor);
}
```

Output :

bilangan = ------ 2.50
nomor = ----- 30.76

Bila jumlah desimal yang ada lebih panjang dari yang akan dicetak, maka desimal tsb akan dibulatkan ke angka terdekat dapat dibulatkan ke atas atau ke bawah. Pengaturan lebar field dapat juga diabaikan dengan pengertian hanya jumlah angka di belakang komanya saja yang diperhatikan.

Contoh :

```
main()
{
      float bil = 2.5, nomor = 30.756;
      clrscr();
      printf ("bilangan  = %.2f \n",bil);
      printf ("nomor     = %.2f, nomor);
}
```

Output :

bilangan = 2.50----
nomor = 30.76---

Data dapat juga dicetak dalam format rata kiri, dengan menyisipkan tanda - (minus) pada format tambahan tadi.

Contoh :

```
main()
{
      float bil = 2.5, nomor = 30.756;
      clrscr();
      printf ("bilangan = %-10.2f \n",bil);
      printf ("nomor    = %-10.2f, nomor);
}
```

Output :

bilangan = 2.50----
nomor = 30.76---

Fungsi output cout

Digunakan untuk menampilkan kelayar tanpa menggunakan string kontrol, baik untuk string maupun identifier.
Bentuk umum:

```
cout << argumen1 << argumen2 <<...;
```

Argumen dapat berupa string, variabel atau suatu konstanta. Untuk menggunakan fungsi cout diperlukan include iostream.h.
Contoh:

```
1. cout<<"belajar c++";
```

Output:

belajar c++

```
2. A=10;
   cout<<"nilai A= "<<A;
```

Output:

nilai A= 10

```
3. A=1;B=2;
   cout<<"nilai A="<<A<<endl<<"nilai B="<<B;
```

Output:

nilai A= 1
nilai B= 2

Berikut contoh penggunaan fungsi cout dengan berbagai jenis argumen:

```
void main(){
   int a=10; float b=78.9; char c='a';
   cout<<a<<b<<c<<endl;
   cout<<"a = "<<a<<" b = "<<b<<" c = "<<c<<"\n";
   cout<<20; cout<<endl;
   cout<<40.4; cout<<endl;
   cout<<'D';
}
```

Output:

10 78.9 a

a = 10 b = 78.9 c = a

20

40.4

D

Saat mencetak hasil komputasi terkadang diinginkan untuk mencetak sesuai dengan format tertentu. Misal mencetak hasil bagi yang memberikan hasil bilangan desimal. Seringkali memberikan nilai dibelakang koma yang cukup panjang. Dan diinginkan untuk tercetak hanya 2 angka dibelakang koma, seperti contoh berikut :

Hasi bagi : 0.04 untuk menggantikan nilai : 0.040972

Perintah berikut pada fungsi `cout` akan mencetak bilangan desimal sampai 2 digit dibelakang koma:

```
cout << fixed << setprecision(2);
```

Perintah diatas tidak memberikan suatu output, hanya memanipulasi cout agar format ouputnya berubah. Nilai `fixed` dan `setprecision` dinamakan manipulator. .Untuk menggunakan manipulator harus memasukkan file header <iomanip> dalam program:

```
#include <iomanip>
```

Manipulator dan nilai dapat dikombinasikan dalam 1 perintah:

```
cout<<fixed<<setprecision(2<<"Hasil:"<<Bagi<< endl;
```

Manipulator `setw` untuk mengatur lebar field output berikutnya. Lebar adalah total jumlah karakter yang digunakan untuk menampilkan nilai, termasuk digit, titik desimal, dan spasi. Mengatur lebar penting saat ingin membuat tampilan dalam bentuk kolom. Contoh, misalnya bilangan akan dicetak dalam sebuah kolom yang mempunyai 8 karakter lebarnya, maka digunakan perintah :

```
cout << setw(8) << Bagi;
```

Pada contoh diatas, akan tercetak nilai Bagi sebesar 8 digit. Jika nilai bagi kurang dari 8 digit maka akan digantikan dengan spasi diawal sebuah item.

Table 3.2 Format output

output statement	output	Penjelasan
cout << 12.345678;	12.3457	Default, angka dicetak 6 digit
cout << fixed << setprecision(2) << 12.3;	12.30	Manipulator fixed dan setprecision untuk mengatur jumlah digit dibelakang koma.
cout<< ":" << setw(6) << 12;	: 12	4 spasi dicetak sebelum angka, total 6 digit termasuk spasi.
cout<< ":" << setw(2) << 123;	: 123	Jika lebar tidak cukup, diabaikan.
cout<< setw(6) << ":" << 12;	:12.3	Lebar hanya dengan item berikutnya, : didahuli dengan 5 spasi.

Fungsi puts()

Digunakan khusus untuk menampilkan data string ke layar. Sifat fungsi ini adalah string yang ditampilkan secara otomatis akan diakhiri dengan \n (pindah baris)

Contoh :

```
#include <stdio.h>
main()
{
    puts("Jakarta ibu kota Indonesia");
}
```

Output :

```
Jakarta ibu kota Indonesia
```

Fungsi putchar()

Fungsi putchar() digunakan khusus untuk menampilkan sebuah karakter ke layar. Penampilan karakter tidak diakhiri dengan perpindahan baris.

Contoh :

```
putchar('A');
```

menghasilkan output yang sama dengan

```
printf("%c", 'A');
```

Input

Fungsi scanf()

Fungsi scanf() merupakan fungsi yang dapat digunakan untuk membaca data dari keyboard dan memasukkan ke dalam program. Fungsi scanf() hampir sama dengan fungsi printf(). Identifier yang digunakan dalam scanf() umumnya sama dengan yang digunakan dalam printf(), sedikit perbedaan yaitu scanf() tidak mengenal penggunaan %g. Fungsi scanf() dan printf() memungkinkan terjadinya komunikasi 2 arah dengan komputer, programnya disebut Interaktif. Daftar argumen dapat berupa satu atau beberapa argumen dan haruslah berupa alamat untuk menyatakan suatu alamat dari variabel di depan dapat ditambahkan tanda & (tanda & dinamakan sebagai operator alamat). & dikenal sebagai operator alamat (address operator). Contoh : `scanf`

("%f",&radius). Berarti "baca sebuah bilangan real (%f) dan tempatkan ke alamat dari radius (&radius).

Satu hal penting yaitu scanf() tidak dapat menggunakan pengaturan lebar field dan jumlah desimal.

Contoh :

```
printf("masukkan bil. pertama : ");
scanf() ("%10.2f",&bil);
```

scanf() menggunakan pointer ke variabel

1. jika variabelnya bertipe data dasar, maka gunakan &

2. jika variabelnya adalah string, tidak menggunakan &

Fungsi scanf() mengijinkan memasukkan beberapa data sekaligus dalam satu baris selama jumlah serta tipe data yang dimasukkan sesuai dengan identifier yang diberikan dalam scanf(). Data yang akan dimasukkan dapat dipisahkan dengan spasi, tab atau dengan tanda pemisah lain seperti koma, garis hubung, titik dua. Pemisah data dalam input yang diketikkan harus sama dengan pemisah data yang digunakan dalam scanf()

Contoh :

```
printf( "masukkan 3 bilangan bulat :");

scanf("%d  %d  %d ",&bil1,&bil2,&bil3);
```

Output :

masukkan 3 bilangan bulat : 12 30 8

Beberapa contoh program dengan scanf:

1.
```
#include <stdio.h>
main()
{
    float dolar,rupiah;
    char  bel;

    bel = '\007';
```

```
    /*character untuk bunyi bel */

    printf ("berapa dollar uang anda ?\n");
    scanf ("%f", &dolar);

    rupiah = dolar * 2000.00;

    printf ("%c uang anda setara dengan ",bel);
    printf ("%2.2f rupiah %c \n",rupiah,bel);
}
```

Output :

berapa dollar uang anda ?
1.2
uang anda setara dengan 2400.00 rupiah.

2.
```
#include<stdio.h>
#define PI 3.141593
main(){
    float radius, keliling, luas;

    printf("masukkan jari-jari lingkaran: ");
    scanf(%f", &radius);
    keliling = 2 * PI * radius;
    luas   = PI * radius * radius;
    printf ("Data lingkaran :\n);
    printf ("jari - jari = %f \n",radius);
    printf ("keliling    = %f \n",keliling);
    printf ("luas        = %f \n",luas);
}
```

Output :

masukkan data jari - jari : 5
data lingkaran = 5
keliling = 31.415930
luas = 78.539825

Fungsi Input cin

Fungsi `cin` digunakan untuk memasukkan data dalam program saat dijalankan. Data yang dimasukkan dengan fungsi `cin` melalui keyboard. Nilai yang dimasukkan disimpan dalam sebuah variabel.

Ketika program dijalankan dan ditemui pernyataan seperti `cin >> num1;` maka, komputer menghentikan eksekusi program dan menerima data dari keyboard. Ketika item data diketik, objek cin menyimpan item dalam variabel yang terdapat setelah operator >>. Program kemudian melanjutkan eksekusi dengan pernyataan berikutnya setelah panggilan ke `cin`. Untuk melihat bagaimana objek ini bekerja, dapat dilihat pada contoh berikut:

```
#include <iostream>

using namespace std;
int main()
{
double num1, num2,hasil;
cout << "masukkan bilangan 1:";
cin >> num1;
cout << "masukkan bilangan 2:";
cin >> num2;
hasil = num1 * num2;
cout<<num1<<" X "<<num2<<" = "<<hasil<<endl;
return 0;
}
```

Dalam hal ini, prompt memberi tahu pengguna untuk mengetik bilangan. Komputer kemudian menjalankan pernyataan berikutnya, yang mengaktifkan `cin`. Objek `cin` menempatkan komputer ke dalam jeda sementara (atau menunggu) sementara pengguna mengetik nilai, dan kemudian pengguna memberi sinyal objek `cin` bahwa entri data selesai dengan menekan tombol Enter. Nilai yang dimasukkan disimpan dalam variabel di sebelah kanan operator >> (num1), dan komputer dikeluarkan dari keadaan dijeda. Eksekusi program berlanjut dengan pernyataan berikutnya, yaitu aktivasi `cout` lain yang menampilkan pesan yang meminta pengguna untuk memasukkan nomor lain. Pernyataan `cin` kedua kembali

menempatkan komputer ke dalam kondisi tunggu, sementara pengguna mengetikkan nilai kedua. Angka kedua ini disimpan dalam variabel num2.

Berikut hasil eksekusi program diatas :

masukkan bilangan 1:30
masukkan bilangan 2:0.05
30 X 0.05 = 1.5

Dalam program diatas, setiap kali `cin` dipanggil, digunakan untuk menyimpan satu nilai dalam variabel. Objek `cin`, dapat digunakan untuk memasukkan dan menyimpan beberapa nilai. Misalnya pada pernyataan `cin >> num1 >> num2;`. Dua buah nilai harus dimasukkan dari keyboard dan disimpan dalam 2 variabel `num1` dan `num2`. Jika data dimasukkan

0.052 245.79

Maka variabel num1 akan berisi 0.052 dan num2 berisi 245.79. Catatan: harus terdapat spasi diantara dua nilai ketika diinputkan.

Untuk memasukkan beberapa data karakter juga dilakukan dengan tanda spasi untuk memisahkan tiap data. Misalnya, dalam pernyataan :

```
char ch1, ch2, ch3; //Deklatarasi 3 variabel
cin >> ch1 >> ch2 >> ch3; //input 3 karakter
```

Jika data dimasukkan

a b c

maka huruf a disimpan pada variabel ch1, huruf b disimpan pada variabel ch2, dan huruf c disimpan pada variabel ch3. Karena sebuah variabel hanya dapat digunakan untuk menyimpan satu buah karakter.

Bentuk umum :

```
cin>>argumen1>>argumen2>>...;
```

Argumen berupa variable. Untuk menggunakan fungsi cin diperlukan include iostream.h. Ada beberapa cara memasukkan data kedalam fungsi input cin yang mempunyai beberapa argument:

1. Setiap argument dipisahkan dengan spasi
2. Setiap argument dipisahkan dengan enter
3. Pemisahan argument berdasar masukan sesuai dengan tipe data argument masing-masing. Jika argumennya mempunyai tipe data sama, cara ini tidak dapat digunakan.

Contoh:

```
#include <iostream>
using namespace std;
int main(){
    int a = 10, b = 20,c = 30, d = 40;

    cout << "Enter four integers: ";
    cin >> a >> b >> c >> d;

    cout << "The numbers you entered are:"<< endl;
    cout<<"a = "<<a<<", b = "<<b<<", c = " << c <<",
    d = " << d;

    return 0;
}
```

Fungsi gets()

Fungsi gets() adalah fungsi input yang khusus digunakan untuk memasukkan string kedalam program.
Bentuk umum:

```
gets (variable);
```

variable merupakan variable yang akan menerima hasil masukan dalam bentuk string, sehingga variable harus bertipe string.

Contoh :

```
#include <stdio.h>
main()
{
    char nama[20];
    clrscr();

    printf("Masukkan nama : "); gets(nama);
    printf("\n Nama anda %s",nama);
}
```

Fungsi getch(), getche(), getchar()

Fungsi getch, getche dan getchar adalah input khusus untuk karakter.

getch() = input 1 karakter dan tidak ditampilkan

getche() = input 1 karakter, ditampilkan & tidak perlu Enter

getchar() = input 1 karakter, ditampilkan dan perlu Enter

Contoh penggunaan :

```
#include <stdio.h>
main()
{
    printf("\n tekan sembarang tombol ”);
    getch();
}
```

Rangkuman

Terdapat beberapa instruksi yang dapat digunakan untuk menginputkan dan menampilkan hasil program. Instruksi untuk menampilkan hasil program dalam C yaitu, `printf`, `cout`, `puts` dan `putchar`. Sedang instruksi untuk input yaitu `scanf`, `cin`, `gets`, `getche`, `getch` dan `getchar`. Untuk membuat format tampilan tertentu dengan `cout`, diperlukan manipulator tertentu.

Latihan

1. Misal x dan y adalah variabel integer dan ch adalah variabel char. Jika pada beberapa pernyataan dibawah diberi input : 5 28 36 , Apakah yang akan tersimpan dalam x, y, dan ch dari setiap pernyatan berikut?
   a. `cin >> x >> y >> ch;`
   b. `cin >> ch >> x >> y;`
   c. `cin >> x >> ch >> y;`

2. Apakah yang akan tercetak dari potongan program dibawah, jika diberi input : 46 A 49 ?

```
int x = 10, y = 18;
char z = '*';
cin >> x >> y >> z;
cout << x << " " << y << " " << z << endl;
```

3. Jika x dan y adalah variabel int, z adalah variabel double, dan ch adalah variabel char. Dengan pernyataan input berikut:

   ```
   cin >> x >> y >> ch >> z;
   ```

   Apakah nilai yang tersimpan pada x, y, z dan ch jika diberi input :

   a. 35 62.78

   b. 86 32A 92.6

   c. 12 .45A 32

4. Apakah yang akan dicetak program berikut?

```
#include <iostream>
using namespace std;
int main(){
cout << "Hello" << endl << "World" << endl;
return 0;
}
```

5. Misalkan `nama` adalah variabel string, buatlah sebuah pernyataan input untuk melakukan input pada variabel `nama` yang dapat menyimpan dan membaca masukan “Brenda Clinton”.

Latihan Program

1. Sebuah toko mengambil keuntungan 30% dari sebuah barang yang akan dijualnya. Buatlah program, untuk menentukan berapa keuntungan yang didapat toko tersebut ketika seorang pembeli membeli sejumlah tertentu barang tersebut. Sebelum membuat program , analisa masalah tersebut dengan menentukan input, proses dan outputnya serta buatlah flowchartnya.

2. Tulislah sebuah program untuk memasukkan dua buah bilangan integer dan mengeluarkan hasil penjumlahan kedua bilangan tersebut.

3. Tulislah sebuah program yang membaca nilai panjang dan lebar dari sebuah segi empat dan menampilkan luas dan keliling segi empat tersebut.

4. Tulislah sebuah program untuk menghitung keliling lingkaran. Masukkan nilai diameter dan hasil perhitungan keliling ditampilkan. Anda menggunakan nilai Phi = 3,14159.

5. Tulis program yang mempunyai fitur input seperti berikut :

   ```
   Masukkan radius :
   ```

   Sesudah menerima nilai radius, program harus menghitung dan menampilkan luas lingkaran (luas = 3.1416 × radius2.)

4

KONDISIONAL

Dalam bab ini dibahas mengenai aliran data dengan perintah kondisional. Terdapat beberapa bentuk kondisional, dengan if, if-else, switch dan perintah kondisional yang digunakan dalam bentuk bersarang. Semua bentuk kondisional akan disertakan contoh-contoh untuk memudahkan penjelasan perintah kondisional

Alur program dieksekusi mengacu pada urutan pernyataan pada program. Secara normal, alur program adalah sekuensial yaitu pernyataan dieksekusi secara berurutan, satu demi satu, dalam urutan di mana mereka ditempatkan dalam program. Tetapi terkadang terdapat alur yang memerlukan percabangan karena alasan tertentu. Untuk melakukan percabangan ini terdapat perintah seleksi yang memerlukan kondisional sebagai syarat seleksi.

Dalam perintah seleksi, untuk pengambilan keputusan mengharuskan programmer menentukan satu atau lebih kondisi yang akan dievaluasi atau diuji oleh program, bersama dengan pernyataan yang akan dieksekusi jika kondisinya benar, dan secara opsional, pernyataan lain yang akan dieksekusi jika kondisi ditentukan sebagai salah. Meskipun hanya ada dua nilai logika, *true* dan *false* tetapi sangat berguna untuk memasukkan pengambilan keputusan yang mengubah aliran proses..

Struktur Kondisi if

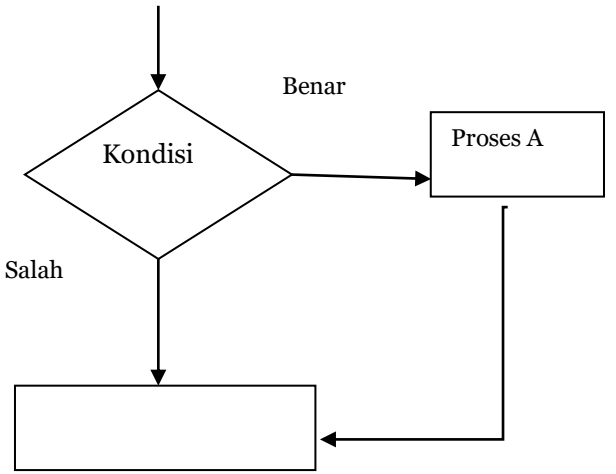
Sebuah bank ingin mengirim pemberitahuan kepada pelanggan jika saldo rekening gironya turun di bawah saldo minimum yang disyaratkan. Artinya, jika saldo akun di bawah saldo minimum yang disyaratkan, harus dikirim pemberitahuan kepada pelanggan. Jika tidak, ia tidak akan melakukan apa pun. Demikian pula, pada sebuah perusahaan asuransi, jika pemegang polis asuransi bukan perokok, perusahaan ingin menerapkan diskon 10% untuk premi polis. Kedua contoh ini merupakan contoh adanya seleksi satu arah. Dimana ketika terdapat kondisi tertentu, maka ada perlakuan tertentu yang dilakukan, tetapi jika tidak terdapat kondisi tertentu maka tidak ada yang dilakukan.

Dalam C++, seleksi satu arah diimplementasikan menggunakan pernyataan `if`. Kondisi `if` memeriksa apakah suatu kondisi dinyatakan benar (*true*). Kondisi merupakan suatu operasi logika. Jika kondisi bernilai *true* maka program akan melakukan suatu pernyataan yang telah ditentukan untuk kondisi tersebut.

Bentuk umum dari pernyataan `if` adalah:

```
if (kondisi)
    pernyataan
```

Bentuk umum pernyataan `if` dimulai dengan perintah `if`, diikuti oleh kondisi di dalam tanda kurung, diikuti oleh pernyataan. Tanda kurung setelah `if` adalah bagian dari sintaks. Kondisi menentukan keputusan seleksi apakah akan pernyataan akan dijalankan. Kondisi biasanya merupakan ekspresi yang logis. Jika nilai kondisi itu benar, pernyataan dijalankan. Jika nilainya salah, pernyataan tidak dijalankan dan komputer melanjutkan ke pernyataan berikutnya dalam program. Pernyataan yang dikerjakan hanya ketika kondisi benar disebut dengan *action statement*. Gambar 4.1 menunjukkan aliran eksekusi pernyataan `if` (*one-way selection*).



Gambar 4.1. Flowchart *one-way selection*.

Ekspresi pada kondisi dapat melibatkan operator relasional dan logika. Contoh pernyataan if :

```
if (score >= 60)
grade = 'P';
```

Dalam potongan program ini ekspresi kondisional (score >= 60) bernilai true, maka grade = 'P'; akan dijalankan. Jika ekspresi false, maka pernyataan setelah struktur `if` akan dijalankan. Contoh, misal score bernilai 65 maka variabel `grade` adalah 'P'.

Jika pernyataan yang dikerjakan saat nilai kondisional *true* lebih dari satu pernyataan, maka harus diberikan tanda { }. Contoh pada potongan program berikut :

```
if (score > 80)
{ nilai = score / 10;
  Grade = 'A';
}
```

Berikut adalah kesalahan yang sering terjadi dalam penulisan pernyataan kondisional :

1.
```
if score >= 60 //syntax error
grade = 'P';
```

   Pernyataan ini menggambarkan pernyataan `if` yang salah karena tanda kurung pada ekspresi kondisional tidak ada.

2. Menempatkan tanda titik koma setelah tanda kurung dalam pernyataan `if` adalah kesalahan semantik. Contoh seperti berikut:

```
if (score >= 60); //baris 1
grade = 'P'; //baris 2
```

   Karena ada titik koma di akhir ekspresi komdisional baris 1), pernyataan if di baris 1 berakhir. Tidak ada pernyataan khusus yang dikerjakan apapun nilai kondisionalnya. Pernyataan di baris 2 bukan bagian dari pernyataan if di baris 1. Oleh karena itu, pernyataan di baris 2 dijalankan terlepas dari bagaimana kondisional `if` dievaluasi.

Untuk menyatakan kondisi dalam struktur kondisional, maka bentuk kondisi merupakan pernyataan yang akan memberikan hasil benar atau salah. Operator yang digunakan dalam menyatakan tes kondisi adalah operator relasiona dan logika

Operator relasi

Tabel 4.1. Operator Relasi

	Arti
==	sama dengan
>=	lebih dari sama dengan
>	lebih dari
<=	kurang dari sama dengan
<	kurang dari
=	tidak sama dengan

Contoh :

```
if(harga>100000)
      discount = harga * 0.1;
```

Contoh program dengan kondisi `if`:

```
void main(){
      int Num1, Num2;

      printf("masukkan nilai 1:");
      scanf("%d", &Num1);

      printf("masukkan nilai 2:");
      scanf("%d", &Num2);
      if(Num1>Num2)
            printf("\n%d+%d = %d\n\n", Num1, Num2,
      Num1+Num2);
}
```

Dalam contoh program diatas, pemakai diminta memasukkan 2 buah nilai integer, kemudian dua buah nilai tersebut dibandingkan. Jika kondisi nilai pertama lebih besar dari nilai kedua maka akan ditampilkan dilayar kedua nilai tersebut dan hasil penjumlahan dari kedua nilai tersebut. Jika kondisi tersebut tidak terpenuhi tidak ada yang dilakukan program.

Berikut adalah contoh program untuk membandingkan 2 nilai:

```
void main()
{
      int Num1, Num2;

      printf("masukkan nilai 1:");
      scanf("%d", &Num1);
      printf("masukkan nilai 2:");
      scanf("%d", &Num2);

      if(Num1>Num2)
      printf("%d lebih kecil dari %d",Num1, Num2);

      if(Num1==Num2)
      printf("%d sama dengan  %d", Num1, Num2);

      if(Num1>Num2)
      printf("%d lebih besar dari %d",Num1, Num2);
}
```

Dalam contoh di atas, dilakukan perbandingan antara variabel `Num1` dan `Num2`, jika kondisi `Num1 < Num2` bernilai benar, maka akan tercetak "Num1 lebih kecil dari Num2". Jika kondisi `Num1 = Num2` bernilai benar, maka akan tercetak "Num1 sama dengan Num2" dan jika kondisi `Num1 > Num2` bernilai benar, maka akan tercetak "Num1 lebih besar dari Num2".

Operator logika

Tabel 4.2. Operator Logika

Simbol	**Arti**
&&	AND
||	OR
!	NOT

Contoh:

```
if((x>0) && (x<20) || !ada())
{
      //pernyataan
}
```

Tes kondisi `((x>0) && (x<20) || !ada())` digunakan untuk mengetahui apakah x lebih besar dari 0 dan lebih kecil dari 20 atau fungsi `ada()` tidak memberikan nilai *true* (mengembalikan nilai 0). Secara umum nilai selain 0 dianggap *true*, sedang 0 dianggap *false*.

Selain perbandingan dilakukan dengan angka, dapat juga dilakukan perbandingan variabel dengan suatu konstanta. Berikut adalah contoh program perbandingan yang dilakukan antara variabel dengan kostanta:

```
void main()
{
      char c;
      printf("ketikkan sebuah huruf:");
      scanf("%c", &c);
      if(c=='a'||c='e'||c=='i' ||c='o'||c=='u')
            printf("anda mengetik huruf vokal");
}
```

Dalam contoh diatas, program akan mencetak hasil dilayar hanya jika masukan berupa karakter vokal. Masukan dibandingkan dengan karakter vokal 'a', 'i', 'u', 'e', 'o'. Karena perbandingan dilakukan terhadap banyak nilai dan akan bernilai benar jika masukan merupakan salah satu vokal, maka digunakan operator or (||).

Struktur Kondisi if-else

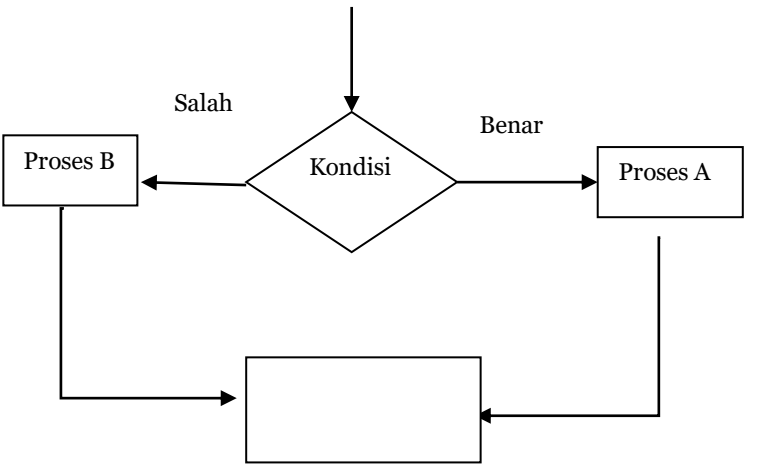
Perolehan gaji karyawan pada sebuh perusahaan menggunakan cara sebagai berikut : jika karyawan paruh waktu bekerja lembur, gajinya adalah dihitung menggunakan rumus pembayaran lembur; jika tidak, gaji dihitung menggunakan rumus biasa. Ini adalah contoh *two-way selection*. Memilih di antara dua alternatif. Dalam C++, proses ini dapat dilakukan dengan perintah `if-else`.

Kondisi `if` melakukan operasi hanya berdasar satu kondisi, selain kondisi tersebut akan diabaikan. Pada pernyataan kondisi `if-else`, jika kondisi yang diinginkan tidak terpenuhi maka akan dilakukan suatu operasi lain. Bentuk umum:

```
if (kondisi) pernyataan1 else pernyataan 2
```

Pada sintaks ini dimulai dengan `if` diikuti oleh ekspresi kondisional yang terkandung di dalam tanda kurung, diikuti oleh pernyataan, diikuti oleh kata `else`, diikuti oleh pernyataan kedua.

Ketika dieksekusi, pertama yang dilakukan adalah memeriksa kondisi. Jika kondisi bernilai *true*, sistem mengeksekusi pernyataan 1. Jika kondisi bernilai *false* sistem akan mengeksekusi pernyataan 2 (sesudah `else`) dan kemudan melanjutkan dengan sisa program. Gambar 4.2 menunjukkan aliran eksekusi pernyataan `if-else`.



Gambar 4.2. Flowchart *two-way selection*.

Contoh program dengan struktur kondisi if-else:

```
void main()
{
    char c;

    printf("ketikkan sebuah huruf:");
    scanf("%c",&c);

    if(c=='a'||c='e'||c=='i' ||c='o'||c=='u')
      printf("anda mengetik huruf vokal\n\n");
    else
      printf("bukan huruf vokal\n\n");
}
```

Dalam program diatas, program akan mencetak hasil dilayar, apakah yang dimasukkan merupakan karakter vokal atau bukan. Jika inputnya berupa karakter vokal yaitu 'a', 'i', 'u', 'e', 'o' maka akan tercetak "anda mengetik huruf vokal" selain karakter tersebut akan tercetak "bukan huruf vokal". Karena perbandingan dilakukan terhadap banyak nilai dan akan bernilai benar jika masukan merupakan salah satu vokal, maka digunakan operator or(||).

Jika pernyataan yang dikerjakan pada operasi if atau else lebih dari satu pernyataan, maka harus diberikan tanda { }.
Contoh pada potongan program berikut :

```
if (score > 80)
{
  nilai = score / 10;
  Grade = 'A';
}
else
{
  Nilai = score / 10
  Grade = 'B';
}
```

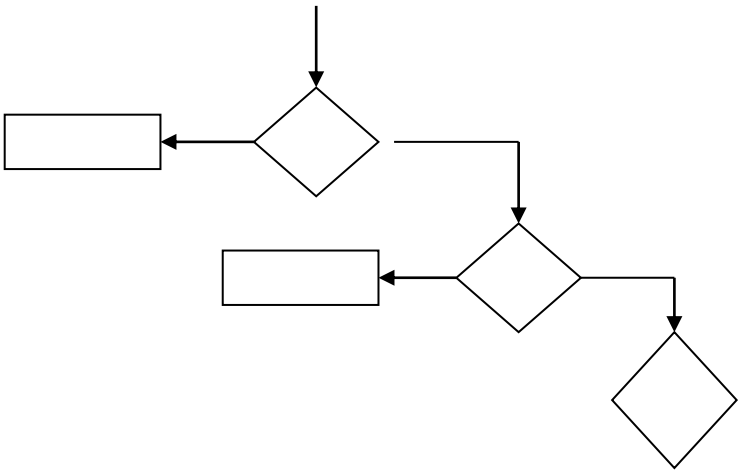
Struktur Kondisional Bersarang

Sebuah universitas membuat aturan penilaian dengan mengkoversikan nilai angka kedalam nilai huruf. Jika nilai 90 sampai 100 dikonversikan menjadi A, nilai 80 sampai 89 dikonversikan menjadi B, nilai 70 sampai 79 dikonversikan ke C, nilai 60 sampai 69 menjadi D, sedang 0 sampai 59 menjadi E.

Di bagian sebelumnya, telah dipelajari cara menerapkan *one-way selection* dan *two-way selection* dalam suatu program. Beberapa masalah memerlukan implementasi lebih dari dua alternatif. Seperti pada contoh konversi nilai diatas.

Contoh diatas terdapat lima alternatif — yaitu seleksi berganda. Seleksi berganda ini dapat diimplementasikan menggunakan if atau if-else. Ketika sebuah pernyataan kondisional berada pada pernyataan kondisional lain maka dikatakan sebagai nested-if.

Salah satu bentuk flowchart dari nested *if* seperti pada gambar 4.3 berikut



Gambar 4.3. Flowchart nested `if`.

Contoh dibawah merupakan ilustrasi dari salah satu bentuk nested if.

```
if(tes1)
      { if(tes2)
            {if (tes3)
                  {…}
            }
            else
            {……}
      }
      else
            {……}
```

Pada sebuah struktur nested, bagaimana cara untuk mengetahui manakah pasangan `if` atau `else`? Dalam C++, tidak ada pernyataan `else` yang berdiri sendiri. Setiap `else` harus didahului dengan `if`. Sedangkan `if` tidak harus berpasangan dengan `else`. Pasangan `else` adalah `if` yang terdekat.

Berikut contoh-contoh program menggunakan nested if:

1. Program konversi nilai angka kedalam nilai huruf:

```
if (nilai >= 90)
   cout << "Nilai huruf adalah A.";
else
   if (nilai >= 80)
      cout << " Nilai huruf adalah B.";
   else
      if (nilai >= 70)
         cout << " Nilai huruf adalah C.";
      else
         if (nilai >= 60)
            cout << " Nilai huruf adalah D.";
         else
            cout << " Nilai huruf adalah F.";
```

2. Program berikut untuk menebak tanggal lahir yang sudah ditentukan. Kondisi digunakan untuk membatasi agar tebakan sesuai dengan tanggal 1 sampai 31. Selain itu kondisi juga digunakan untuk memberi masukan apakah tebakan terlalu tinggi, rendah atau tepat.

```
void main()
{
 int Ultah = 13;
 int tebak;

 cout<<"tebak tanggal lahir dari 1 sampai 31\n";
 cout<<"masukkan tebakan anda : ";
 cin>>"%d",&tebak;
 if (tebak<0)
   cout<< "tanggal tidak negatif";
 else
   if(tebak>31)
      cout<< "tidak ada tanggal diatas 31\n";
   else
   {  if(tebak == ultah)
         cout<< "anda benar";
      else
         if(tebak<ultah)
            cout<< "tebakan terlalu rendah";
         else
            cout<<"tebakan terlalu tinggi";
   }
}
```

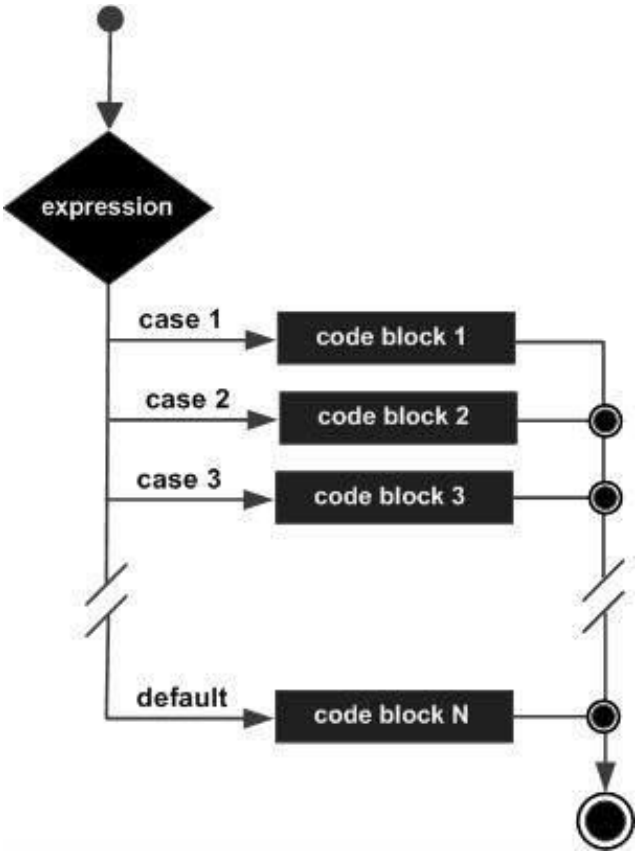
Struktur Kondisi Switch

Struktur kondisi `switch` digunakan seperti struktur kondisi `if-else` yang berurutan. Jika struktur kondisi `if-else` memeriksa masing-masing kondisi yang berturutan, struktur kondisi `switch` akan memeriksa nilai yang sesuai untuk kondisi tersebut.

Aliran jalannya program untuk pernyataan switch seperti pada gambar 4.4. Nilai pada ekspresi akan dicocokkan dengan nilai yang terdapat pada masing-masing case. Pernyataan pada case yang mempunyai nilai yang sesuai akan dijalankan.

Bentuk umum dari pernyataan switch dalam C++ seperti berikut:

```
switch (ekspresi)
{
        case........: pernyatan;
                break;
        case........: pernyatan;
             break;
        .
        .
       default: pernyataan;  //pilihan
}
```



Gambar 4.4. Flowchart pernyataan switch

Ekspresi yang digunakan pada switch harus bernilai integer atau nilai yang dapat dikonversikan kedalam bentuk integer, seperti karakter. Ekspresi bisa berupa variabel atau persamaan yang memberikan hasil integer.

Dalam sebuah pernyataan switch bisa terdapat beberapa pernyataan case. Pernyataan case diikuti dengan nilai dan titik dua. Jadi pernyataan case terdiri dari :

- Kata case
- Diikuti dengan konstanta yang dapat dikonversikan ke tipe integer.
- Tanda :
- Satu atau lebih instruksi

Contoh:

```
case 1: sum = sum+x;
```

Nilai untuk sebuah case harus mempunyai tipe data yang sama dengan variabel atau nilai ekspresi pada switch, dan harus berupa konstanta atau literal.

Nilai ekspresi pada switch akan dibandingkan dengan nilai yang mengikuti setiap case. Ketika ditemukan nilai yang sesuai, sistem akan mengerjakan instruksi dalam pernyataan case tersebut. Pada contoh case diatas, berarti `sum = sum + x` akan diproses jika ekspresi memberikan nilai 1. Atau sama dengan pernyataan

```
if (ekpresi ==1) sum = sum+x;
```

Dalam pernyataan switch, jika sudah ditemukan case yang sesuai maka instruksi dalam case tersebut akan dilakukan dan terus berlanjut ke instruksi-instruksi selanjutnya sampai ditemukan perintah break.. Setelah itu akan dilanjutkan pada instruksi setelah pernyataan switch.

Perintah break digunakan untuk membatasi instruksi-instruksi yang dapat dikerjakan ketika nilai pada case tertentu sesuai dengan nilai ekspresi pada switch. Tetapi tidak semua case harus mempunyai perintah break. Jika tidak ada break pada suatu case,

maka eksekusi program akan berlanjut ke instruksi-instruksi pada case berikutnya sampai ditemukan break

Pernyataan switch dapat memiliki case opsional default, yang berada di akhir switch. case default dapat digunakan untuk melakukan tugas saat tidak ada pernyataan case yang benar. Dalam case default tidak diperlukan break, karena case default ditulis di akhir pernyataan switch.

Masing-masing pernyataan switch dapat memasukkan satu case *default*, yang dikodekan sebagai berikut:

```
Default: pernyataan;
```

Jika nilai dari ekspresi yang tidak cocok dengan case yang ada, akan dilakukan instruksi dalam case default. Berikut contoh struktur kondisi switch dengan case default:

```
char char1;
char1=getchoice();
cwitch(char1)
{     case'a':
      case'A': ProsesChoiceA();
               Break;
      case'b':
      case'B': ProsesChoiceB();
               Break;
      case'q':
      case'Q': ProsesChoiceQ();
               Break;
      default:
            printf("Pilihan salah");
}
```

Dalam contoh diatas, jika ekspresi yang merupakan karakter bernilai ‘a’ maka akan dilakukan pemanggilan fungsi `ProsesChoiceA()` pada case ‘A’. Hal ini karena pada case ‘a’ tidak ada operator break, sehingga akan melanjutkan kepernyataan berikutnya. Hal sama berlaku juga untuk case’b’ dan case ‘q’, karena tidak ada operator break maka akan dijalankan pernyataan pada case berikutnya dan akan berhenti jika menemukan operator break. Selanjutnya jika char1 bernilai selain ‘a’, ‘A’, ‘b’, ‘B’, ‘q’, ‘Q’ maka program akan menjalankan pernyataan dalam case default.

Berikut adalah contoh program yang memberikan nilai ekspresi integer dengan `case default`:

```
void main()
{
   int angka;

   printf("tulis angka antara 1 sampai 5:");
   scanf("%d", &angka);

   switch(angka){
   case 1: cout<<"anda menuliskan angka"<<angka;
           break;
   case 2: cout<<"anda menuliskan angka"<<angka;
           break;
   case 3: cout<<"anda menuliskan angka"<<angka;
           break;
   case 4: cout<<"anda menuliskan angka"<<angka;
           break;
   case 5: cout<<"anda menuliskan angka"<<angka;
           break;
   default: cout<<angka<<"tidak antara 1 sampai
   5",angka);
   }
}
```

Pernyataan switch dapat dikonversikan dalam bentuk if-else. Keduanya mempunyai tugas yang sama, yaitu membandingkan beberapa pernyataan untuk dicari yang sesuai dengan yang diinginkan.

Berikut merupakan contoh perintah switch yang dikonversikan ke dalam perintah if else:

Perintah switch :

```
switch(int(score)/10){
      case 10:
      case 9: cout<<"Grade = A";
                  break;
      case 8: cout<<"Grade = B";
                  break;
      case 7: cout<<"Grade = C";
                  break;
```

```
case 6: cout<<"Grade = D";
            break;
default:cout<<"Grade = F";
}
```

Konversi dalam bentuk if else menjadi :

```
if (score >= 90)
  cout << "Grade = A";
else if (score >= 80)
  cout << "Grade = B";
else if (score >= 70)
     cout << "Grade = C";
else if (score >= 60)
     cout << "Grade = D";
else  // score < 59
     cout << "Grade = F";
```

Rangkuman

Dalam bab ini dibahas pernyataan kondisional dalam C++ dimana didalamnya terdapat pengambilan keputusan akan kondisi yang ada apakah bernilai benar atau salah. Struktur kondisi dalam C++ yang dibahas adalah struktur kondisi if, if-else, ?: dan switch. Masing-masing struktur mempunyai bentuk umum yang berbeda dan penggunaan yang berbeda juga.

Dalam struktur kondisi if, statemen yang ada hanya dijalankan jika kondisi dengan nilai benar. Sedang dalam struktur kondisi if-else ada pilihan pernyataan lain jika ternyata kondisi ternyata bernilai salah. Struktur kondisi switch sama dengan beberapa struktur kondisi if-else yang berturutan yang memeriksa beberapa kemungkinan kondisi untuk sebuah ekspresi.

Latihan

1. Perbaikilah program berikut sehingga memberikan hasil w = 21

```
const int SECRET = 5
main ()
{
    int x, y, w, z;
    z = 9;
```

```
    if z > 10
    x = 12; y = 5, w = x + y + SECRET;
    else
    x = 12; y = 4, w = x + y + SECRET;
    cout << "w = " << w << endl;
}
```

2. Apakah kesalahan dari program berikut? Perbaikilah sehingga program dapat dijalankan. Apakah hasilnya?

```
main (){
    int a, b;
    bool found;

    cout << "masukkan dua bilangan integer:" ;
    cin >> a >> b;

    if a > a *b && 10 < b
       found = 2 * a > b;
    else
    {
       found = 2 * a < b;
       if found
           a = 3;
       c = 15;
       if b
       {
           b = 0;
           a = 1;
       }
    }
}
```

3.
```
int main()
{
int nilai;

cout << "Masukkan nilai: ";

cin >> nilai;
cout << endl;
```

```
switch (nilai / 10)
{
   case 0:
   case 1:
   case 2:
   case 3:
   case 4:
   case 5: cout << " Nilai huruf F." << endl;
   case 6: cout << " Nilai huruf D." << endl;
   case 7: cout << " Nilai huruf C." << endl;
   case 8: cout << " Nilai huruf B." << endl;
   case 9:
   case 10:cout << " Nilai huruf A." << endl;
   default:cout << "Input salah." << endl;
}
return 0;
}
```

Pada program diatas untuk input diluar 1 sampai 100 memberikan hasil yang benar. Tetapi untuk input 1 sampai 100 hasilnya salah. Perbaikilah sehingga mendapatkan hasil yang benar.

4. Apakah output dari potongan program dibawah ?

```
x = 100;
y = 200;
if (x > 100 && y <= 200)
cout << x << " " << y << " " << x + y << endl;
else
cout << x << " " << y << " " << 2 * x - y <<
endl;
```

5. Dalam suatu ujian, jika nilai yang dicapai 60 keatas, maka siswa dinyatakan lulus. Jika kurang maka dinyatakan gagal. Perbaikilah pernyataan kondisional berikut sehingga memberi hasil yang benar.

```
if (Nilai >= 60)
cout << "Anda lulus." << endl;
else;
cout << "Anda gagal" << endl;
```

Latihan Program

1. Sebuah supermarket membuat prrogram promo, dimana jika seorang pembeli mempunyai total pembelanjaan Rp. 100.000,- keatas maka akan mendapat potongan sebesar 10 %. Tulislah sebuah program untuk menentukan harga yang harus dibayar oleh seorang pembeli dimana total pembelanjaan diinputkan dari keyboard.

2. Tulislah sebuah program yang membandingkan dua bilangan integer yang dimasukan oleh pemakai dan mengeluarkan pesan bilangan mana yang lebih kecil.

3. Tulislah sebuah program untuk membagi dua bilangan integer yang dimasukkan oleh pemakai. Tampilkan pesan tidak dapat dibagi nol jika bilangan yang kedua sama dengan nol.

4. Buatlah program untuk menghitung diskriminan dan mencari akar-akar dari persamaan kuadrat : ax2 + bx + c = 0, dengan ketentuan sbb :

   D = b2 - 4ac

   - Jika D = 0 , maka terdapat 2 akar real yang kembar, yaitu : x1 = x2 = -b / 2a
   - Jika D > 0 , maka terdapat 2 akar real yang berlainan, yaitu :
     x1 = (-b + sqrt(D)) / 2a
     x2 = (-b - sqrt(D)) / 2a
   - Jika D < 0 , maka terdapat 2 akar imaginair yang berlainan, yaitu :
     x1 = -b / 2a + (sqrt(-D) / 2a) i
     x2 = -b / 2a - (sqrt(-D) / 2a) i

   Input : a, b, c (float)

   Output : Nilai Diskriminan serta nilai akar-akar persamaan tsb (x1& x2).

5. Buatlah program menggunakan switch-case untuk menampilkan menu dan melakukan proses sbb :

Menu :

1. Menghitung volume kubus
2. Menghitung luas lingkaran
3. Menghitung volume silinder.

Jika pilihan selain 1, 2 & 3 (default) : Tampilkan pesan kesalahan..

Jika pilihan = 1, maka :
Input : panjang sisi kubus
Output : Volume kubus (vol = $sisi^3$)
Jika pilihan = 2, maka :
Input : panjang jari-jari lingkaran
Output : Luas lingkaran (luas = 3.14 * r^2)
Jika pilihan = 3, maka :
Input : panjang jari-jari lingkaran & tinggi silinder
Output : Volume silinder (vol = 3.14 * r^2 * t)
Jika pilihan selain 1, 2 & 3 (default) : Tampilkan pesan kesalahan.

5

Perulangan

Dalam bab ini akan dibahas tentang perintah-perintah perulangan dan bagaimana menyusun instruksi perulangan. Beberapa perintah perulangan yang akan dijelaskan adalah `while`, do `while` dan `for`. Selain itu juga akan dibahas tentang perulangan bersarang serta contoh-contoh penggunaan perulangan

Dalam membuat program, terkadang ada permasalahan seperti mencari rata-rata dari lima bilangan yang diinputkan. Untuk menyelesaikan masalah tersebut dapat dilakukan dengan membuat program seperti dibawah :

```
cin >> num1 >> num2 >> num3 >> num4 >> num5;
 //read five numbers
sum = num1 + num2 + num3 + num4 + num5;
//add the numbers
average = sum / 5;
//find the average
```

Tapi ketika bilangan yang akan ditambahkan dan dicari rata-rata berjumlah 100, 1000, atau lebih banyak lagi maka harus mendeklarasikan banyak variabel dan menuliskan dalam pernyataan cin serta pernyataan cout sebanyak jumlah bilangan tersebut. Tentu saja akan membutuhkan jumlah ruang dan waktu yang terlalu tinggi. Dan ketika ingin menjalankan program lagi dengan banyaa bilangan yang berbeda atau dengan jumlah nilai yang berbeda, harus menulis ulang program.

Misalkan lima data yang akan dijumlahkan adalah 5 3 7 9 4. Maka proses penjumlahan dapat dilakukan melalui beberapa pernyataan berikut:

1. `sum = 0;`
2. `cin >> num;`
3. `sum = sum + num;`

Pernyataan pertama memberi nilai awal pada `sum` = 0. Selanjutnya menjalankan pernyataan 2 dan 3. Pernyataan 2 menyimpan 5 dalam num, pernyataan 3 merubah nilai `sum` dengan menambahkan `num` ke dalamnya. Setelah pernyataan 3, nilai `sum` adalah 5.

Selanjutnya ulangi pernyataan 2 dan 3. Setelah pernyataan 2 (setelah kode membaca angka berikutnya):

`num` = 3,

dan setelah pernyataan 3:

`sum = sum + num` = 5 + 3 = 8

Sampai disini, **sum** berisi jumlah dari dua angka pertama.

Ulangi pernyataan 2 dan 3 (ketiga kalinya). Setelah pernyataan 2 (setelah kode membaca nomor berikutnya):

```
num = 7
```

dan setelah pernyataan 3:

```
sum = sum + num
```

= 8 + 7 = 15

Sekarang, **sum** berisi jumlah dari tiga angka pertama. Jika diulangi pernyataan 2 dan 3 dua kali lagi, **sum** akan berisi jumlah dari semua lima angka.

Jika ingin menambahkan 10 angka, maka harus mengulangi pernyataan 2 dan 3 sepuluh kali. Dan jika ingin menambahkan 100 angka, harus mengulangi pernyataan 2 dan 3 sebanyak seratus kali. Ada banyak situasi lain di mana perlu mengulangi serangkaian pernyataan. Misalnya, pengolahan nilai untuk setiap siswa di kelas. Dimana cara pengolahan nilai setiap siswa sama.

C ++ memiliki tiga struktur perulangan, yang memungkinkan untuk mengulangi pernyataan sama berulang sampai kondisi tertentu terpenuhi tanpa harus menulis berulangkali pernyataan yang sama. 3 struktur perulangan dalam C++ yaitu : `while`, do `while`, dan `for`.

Struktur Perulangan while

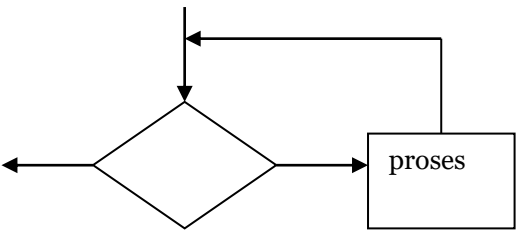
Seperti pada penjelasan sebelumnya, terkadang perlu mengulangi serangkaian pernyataan beberapa kali. Salah satu cara untuk mengulangi serangkaian pernyataan adalah dengan mengetikkan serangkaian pernyataan dalam program berulang-ulang. Namun, solusi ini mengulangi serangkaian pernyataan tidak praktis. C ++ menyediakan cara yang lebih baik untuk mengulangi serangkaian pernyataan. Seperti disebutkan sebelumnya, C++ memiliki tiga struktur perulangan. Struktur yang memungkinkan untuk mengulangi serangkaian pernyataan sampai kondisi tertentu terpenuhi. Bagian ini membahas struktur perulangan pertama, yang disebut perulangan `while`.

Perulangan `while` digunakan untuk mengeksekusi beberapa pernyataan i jika kondisinya bernilai `true`. Jika kondisi bernilai `false` dari awal eksekusi maka pernyataan-pernyataan yang bersangkutan tidak akan pernah dieksekusi.

Bentuk umum:

```
while (kondisi)
pernyataan
```

Sewaktu dieksekusi instruksi `while` pertama memeriksa kondisi. Jika kondisi memberikan nilai `true` , sistem akan mengeksekusi pernyataan. Kemudian mengulangi tes kondisi dan mengerjakan pernyata jika hasilnya masih `true`. Hal ini diulangi sampai kondisi mengembalikan nilai `false`. Alur proses perulangan `while`, seperti pada gambar 5.1.



Gambar 5.1. Flowchart Perulangan `while`

Sebuah perulangan dapat terus berlanjut mengeksekusi pernyataan tanpa henti, yang disebut *infinite loop*. Untuk menghindari perulangan tak terbatas, pastikan bahwa dalam perulangan itu berisi pernyataan yang menjamin bahwa nilai ekspresi kondisi pada akhirnya akan salah.

Berikut contoh sederhana penggunaan struktur perulangan `while` :

```
i = 0;                    // Baris 1
while (i <= 20)           // Baris 2
{
      cout << i << " "; // Baris 3
      i = i + 5;          // Baris 4
}
cout << endl;
```

Output :

```
0 5 10 15 20
```

Pada baris 1, variabel `i` diberi nilai 0. Kondisi dalam pernyataan `while` (di baris 2), `i <= 20`, dievaluasi. Karena kondisi `i <= 20` bernilai *true*, pernyataan-pernyataan dalam `while` akan dieksekusi. Pernyataan di baris 3 menampilkan nilai `i`, yaitu 0. Pernyataan di Baris 4 mengubah nilai `i` menjadi 5. Setelah menjalankan pernyataan di baris 3 dan 4, kondisi di perulangan `while` (baris 2) dievaluasi lagi. Karena `i` bernilai 5, kondisi `i <= 20` bernilai *true* dan isi dari perulangan `while` dieksekusi lagi. Proses mengevaluasi kondisi dan mengeksekusi pernyataan dalam perulangan `while` berlanjut sampai kondisi, `i <= 20` (pada Baris 2), tidak lagi bernilai *true*.

Variabel `i` dalam kondisi pada contoh diatas disebut dengan variabel kontrol. Beberapa hal yang perlu diperhatikan pada perulangan `while` berdasar contoh diatas:

a. Dalam perulangan, `i` menjadi 25 tetapi tidak dicetak karena kondisi bernilai salah.

b. Jika pernyataan `i = i + 5;` dihilangkan maka perulangan akan menjadi infinite loop. Perintah cetak akan dilakukan berulangkali tanpa henti karena kondisi selalu bernilai *true*.

c. Sebagai variabel kontrol, `i` perlu diberi nilai awal sebelum dievaluasi dalam kondisi. Jika pernyataan `i = 0;` dihilangkan, perulangan tidak akan pernah dikerjakan.

d. Jika pada contoh sebelumnya, 2 pernyataan dalam perulangan dirubah, maka akan memberi hasil yang sangat berbeda. Contoh seperti berikut:

```
i = 0;
while (i <= 20)
{
   i = i + 5;
   cout << i << " ";
}
   cout << endl;
```

output :

5 10 15 20 25

Jika diinginkan program akan mencetak bilangan yang kurang atau sama dengan 20, maka susunan pernyataan dalam perulangan akan memberi hasil yang salah. Karena saat `i = 20`, nilai kondisi akan bernilai *true* dan perulangan tetap akan dijalankan. Dengan penjumlahan `i` yang dieksekusi cetak, maka hasil akhir akan lebih besar dari 20.

e. Jika tanda titik koma diletakkan di akhir perulangan `while`, (setelah kondisi), maka tidak ada pernyataan yang diulang. Pernyataan `while` hanya berhenti sampai kondisi saja.

```
i = 0;
   while (i <= 20);
   {
         i = i + 5;
         cout << i << " ";
   }
   cout << endl;
```

Seperti dalam Contoh sebelumnya, pernyataan dalam `while` dieksekusi hanya ketika kondisi bernilai *true*. Biasanya, kondisi diperiksa apakah variabel (*variable loop kontrol* (LCV)) memenuhi kondisi tertentu. Dalam Contoh kondisi pernyataan `while` memeriksa apakah `i <= 20`. LCV harus diinisialisasi dengan benar sebelum perulangan, dan harus mempunyai kondisi yang berakhir dengan nilai *false*. Untuk itu perlu adanya perubahan nilai LCV sehingga kondisi akan menuju nilai *false*. Bentuk dari perulangan `while` secara lengkap seperti berikut:

```
//inisialisasi loop control variable
  while (kondisi) //evaluasi LCV
 {
     .
.//perubahan loop control variable
     .
  }
```

Contoh penggunaaan perulangan dengan `while` dalam program untuk menjumlahkan 10 angka yang diinputkan:

```
#include <iostream.h>

void main()

{
         int number;
         int sum = 0;
         int count = 0;
        while (count < 10)
         {        printf("Masukkan angka");
                  scanf("%d",&number);
                  sum += number;
                  count++;
         }
         printf("jumlah: %d", sum);
}
```

Dalam contoh program diatas, perulangan dilakukan 10 kali dengan instruksi `while (count < 10)`, kode setelah kata `while` dan dalam ( ) adalah tes kondisi.

Dalam hal ini evaluasi adalah:“ apakah lokasi memori yang disimbolkan dengan count berisi lebih kecil dari 10 ? “. Jika jawabannya “ya” maka instruksi didalam perulangan ‘`while` dieksekusi. Jika “tidak” komputer meloncat ke instruksi pertama setelah perulangan `while`, dimana dalam program ini adalah instruksi `printf("jumlah: %d", sum);`

Dua baris pertama dalam perulangan `while`, user diminta memasukkan suatu angka dan komputer menunggu sampai ditekan enter, kemudian komputer menyimpan nilai kedalam lokasi memori yang disimbolkan dengan variable `number`. Baris berikutnya: `sum += number;` adalah untuk menambahkan nilai yang disimpan dalam `number` ke nilai yang disimpan dalam `sum`. Kemudian perintah: `count++;` perintah ini digunakan untuk perubahan kondisi yang menunjukkan banyaknya input yang sudah dimasukkan, sehingga ketika `count` sudah mencapai nilai 10 maka perulangan berhenti.

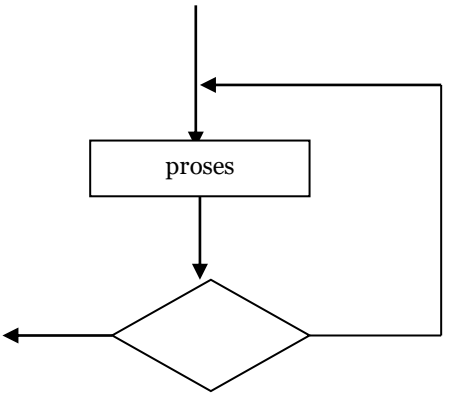
Struktur perulangan do `while`

Perulangan `do-while` juga mengeksekusi beberapa pernyataan berulang jika kondisi yang telah ditentukan dipenuhi. Bentuk umum perulangan `do-while`:

```
do {
    pernyataan
} while (kondisi)
```

Perbedaan antara perulangan `do-while` dengan perulangan '`while`' adalah jika didalam perulangan '`while`' tes kondisi dilakukan pada awal perulangan sebelum masuk ke pernyataan dalam perulangan, sedang dalam perulangan `do-while` tes kondisi dilakukan diakhir perulangan. Jadi setelah pernyataan dalam perulangan dikerjakan, tes kondisi baru dilakukan. Jika nilai kondisi *true*, maka pernyataan dalam perulangan akan dikerjakan kembali.

Sebagai catatan, jika tes kondisi bernilai *false* ketika awal pernyataan perulangan '`while`' maka pernyataan dalam perulangan tidak akan pernah dikerjakan. Sedang dalam perulangan `do-while` tes kondisi dilakukan pada akhir pernyataan sehingga apapun hasil dari tes kondisi, pernyataan dalam perulangan pasti akan dikerjakan meskipun hanya satu kali. Alur ekseskusi perulangan `do-while` seperti pada gambar 5.2.



Gambar 5.2 Flowchart perulangan `do-while`

Untuk menghindari perulangan tanpa batas. Perlu ada perubahan nilai didalam pernyataan perulangan yang menyebabkan nilai kondis berakhir dengan *false*.

Berikut contoh perulangan `do-while`,

```
i = 0;
do
{
      cout << i << " ";
      i = i + 5;
}
while (i <= 20);
```

output:

0 5 10 15 20

Setelah dicetak 20 pernyatan i = i + 5; merubah nilai i menjadi 25, sehingga kondis i <= 20 menjadi *false*, dan perulangan berhenti.

Berikut ini adalah contoh program untuk menghitung nilai count dimulai dengan 0, dengan penambahan 2 dan akan berhenti pada 15:

```
#include <iostream.h>

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

main()
{
      int Count = 0;

      do {
            printf("%d\n", Count * 2);
            Count++;
      } while( Count <= 15 );
}
```

Pada contoh program diatas, pertama kali akan mencetak `Count * 2`, kemudian `Count` ditambahkan satu sebagai perubahan tes kondisi

untuk menghitung berapa kali perulangan sudah dilakukan perulangan. Tes kondisi dilakukan setelah instruksi dalam perulangan dilakukan dengan instruksi sebagai berikut `(Count <= 15 )`, jika hasil tes kondisi bernilai *true* maka perulangan dikerjakan jika berbernilai *false* maka perulangan berhenti.

Perulangan do-while sering digunakan dalam sebuah perulangan yang tidak dapat diperkirakan berapa kali akan dilakukan perulangan. Contoh :

```
char jawab;
do{
      ……..
      cout << "lanjut(y/t)? ";
      cin >> jawab;
} while(jawab!='n');
```

Pada potongan program diatas, perulangan hanya akan berhenti saat `jawab` diisi dengan 'n'. Disini, berapa jumlah perulangan tergantung dari input pengguna dan tidak dapat diketahui sebelumnya.

Contoh perbedaan antara perulangan `while` dan `do-while` seperti berikut :

1. Perulangan `while`

```
i = 11;
while (i <= 10)
{
cout << i << " ";
i = i + 5;
}
cout << endl;
```

2. Perulangan `do-while`

```
 i = 11;
do
{
   cout << i << " ";
```

```
    i = i + 5;
}
while (i <= 10);
cout << endl;
```

Dalam perulangan `while` tidak akan mengeluarkan output apaun karena `i = 11` menyebabkan kondisi awal bernilai *false*. Dalam perulangan `do-while`, akan keluar output 11, dan pernyataan dalam perulangan hanya dikerjakan satu kali, karena setelah mengerjakan pernyataan, nilai I berubah menjadi 16 yang menyebabkan kondisi bernilai *false*.

Struktur Perulangan for

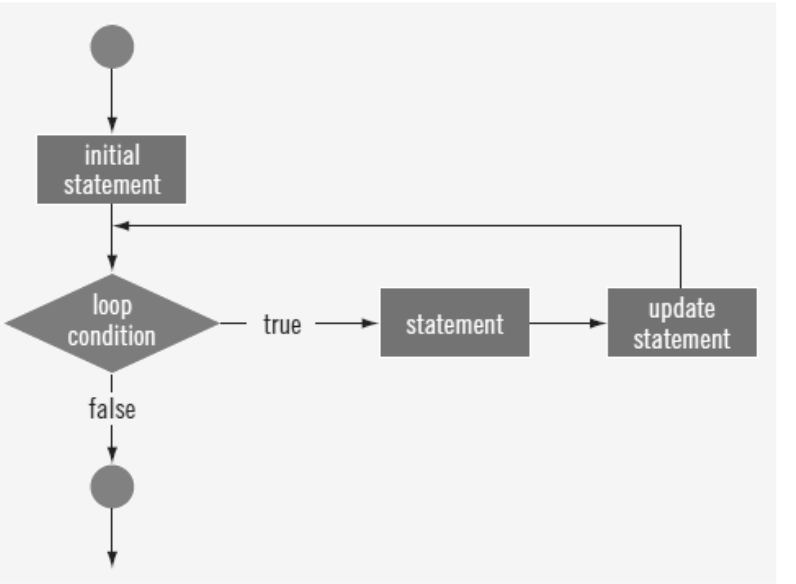
Struktur perulangan ketiga adalah perulangan '`for`'. Perulangan '`for`' dapat mengeksekusi pernyataan berulang sejumlah yang sudah ditentukan. Bentuk umumnya adalah sebagai berikut:

```
for (inisialisasi ; tes kondisi ; perubahan nilai)
{
        Pernyataan-pernyataan
}
```

Inisialisasi merupakan pemberian nilai awal pada variabel yang akan dilakukan tes kondisi. **Tes kondisi** untuk menentukan apakah perulangan bisa dikerjakan. Dan yang terakhir **perubahan nilai** yaitu perubahan nilai untuk tes kondisi sehingga perulangan bisa mencapai batas untuk berhenti. Alur proses perulangan `for` seperti gambar 5.3.

Ekseskusi perulangan `for` sebagai berikut:

1. Eksekusi inisialisasi.
2. Dilakukan tes kondisi, jika memberi nilai *true*:
    i. Eksekusi pernyataan dalam perulangan.
    ii. Ekseskusi perubahan nilai.
3. Ulangi langkah 2 sampai hasil tes kondisi bernila *false*.



Gambar 5.3. Flowchart perulangan `for`

Cara sederhana untuk mengerti perulangan '`for`' dapat dengan mempelajari beberapa contoh. Berikut adalah contoh perulangan '`for`' untuk menghitung dari 1 sampai 10:

```
for (int count = 1; count <= 10; count++)
{
    printf("%d\n", count);
}
```

Tes kondisi mungkin tidak berhubungan dengan variabel yang diinisialisi dan diperbaharui. Disini perulangan menghitung sampai user memberikan respon untuk menghentikan perulangan. Contoh seperti berikut:

```
for (count = 1; response != 'T'; count++)
{
    printf("%d\n", count);
    printf("Lanjut (Y/T): ");
    scanf("%c",&response);
}
```

Dapat juga tes kondisi lebih rumit, misalnya user dari contoh diatas tidak pernah memasukkan 'T' tetapi perulangan harus berhenti ketika dicapai angka 100, maka instruksi menjadi seperti berikut:

```
for (count = 1; (response != 'N') && (count <=
100); count++)
{
    printf("%d\n", count);
    printf("Lanjut(Y/T): ");
    scanf("%c",&response);

}
```

Selain itu bisa terdapat multiple inisialisasi dan multiple perubahan nilai. Perulangan dimulai dengan dua variabel ,yang pertama dengan nilai 0 dan yang lain dengan nilai 100 dan dicari nilai pertengahan diantaranya, seperti dalam contoh berikut:

```
for (i = 0, j = 100; j != i; i++, j--)
{
      printf("%d    %d\n",i,j);
}
 printf("%d    %d\n",i,j);
```

Inisialisasi adalah pilihan, misalkan diperlukan untuk menghitung dari angka yang dimasukkan user sampai 100. Meskipun tidak ada inisialisasi tetap diperlukan ; pertama. Contoh seperti pada potongan program berikut:

```
printf("Masukkan angka sebagai awal hitungan:");
scanf('%d"& count);

for ( ; count < 100 ; count++)
{
    printf("%d\n",count);
}
```

Perubahan nilai juga merupakan pilihan. Berikut adalah contoh sederhana perulangan menginputkan nilai sampai user menghentikan perulangan.

```
for (number = 5; response != 'Y';) {
    printf("%d\n" , number);
   printf("Cukup (Y/T)\n ");
   scanf("%c", response);
}
```

Berikut beberapa contoh perulangan `for` yang perlu mendapat perhatian :

1.
```
for (int i = 10; i <= 9; i++)
   cout << i << " ";
cout << endl;
```

   Pada perulangan ini , inisialisasi i diberi nilai 10. Tes kondisi `(i <= 9)` akan benilai *false*, maka pernyataan dalam perulangan tidak akan dikerjakan.

2.
```
for (int i = 9; i >= 10; i--)
cout << i << " ";
cout << endl;
```

   Pada perulangan ini inisialisasi i diberi nilai 9. Karena Tes kondisi (i >= 10) bernilai *false,* maka pernyataan dalam perulangan tidak akan dikerjakan.

3.
```
for (int i = 10; i <= 10; i++) //baris 1
cout << i << " "; //baris 2
cout << endl; //baris 3
cout << endl;
```

   Pada perulangan ini inisialisasi i diberi nilai 10. Tes kondisi `(i <= 10)` akan benilai *true*, maka pernyataan pada baris 2 akan di eksekusi sehingga akan dihasilkan output 10. Selanjutnya pada perubahan nilai, nilai i menjadi 11. Tes kondisi memberikan nilai *false* dan perulangan berakhir. Jadi pernyataan pada perulangan hanya dikerjakan satu kali

4.
```
for (int i = 1; i <= 10; i++); //baris 1
cout << i << " "; //baris 2
cout << endl; //baris 3
```

Perulangan ini tidak ada hubungannya dengan baris ke 2, karena tanda titik koma di akhir pernyataan for mengakhiri perulangan `for`.

5.
```
for (int i = 1; ; i++)
cout << i << " ";
cout << endl;
```

Dalam perulangan ini, tes kondisi loop dihilangkan dari pernyataan for, kondisi perulangan selalu benar, sehingga menjadi perulangan tanpa batas.

Berikut adalah perbandingan format dan contoh program antara perulangan for dan perulangan while :

Perulangan for	Perulangan while

for (inisialisasi;
tes kondisi;
perubahan nilai)
 	
Pernyataan
	inisialisai
	while (kondisi)
	{
	Pernyataan
	perubahan nilai
	}

`for (int i=0; i<10; i++)`
`cout << i << " ";`
`cout << endl;`
	`int i = 0;`
	`while (i < 10)`
	`{`
	`cout << i << " ";`
	`i++;`
	`}`
	`cout << endl;`

Meskipun dalam penggunaannya perulangan for dan while sama, tetapi ketika jumlah perulangan sudah diketahui sebelumnya, programmer cenderung menggunakan perulangan for.

Perulangan Bersarang (Nested Loop)

Pada sebuah perulangan didalamnya dapat terjadi perulangan juga, yang dikenal dengan nested loop. Perulangan yang berada didalam (inner loop) harus dieksekusi sejumlah perulangan yang berada diluarnya(outer loop). Misalnya pada table perkalian berikut:

Contoh pada permasalahan daftar perkalian. Misalkan akan dibuat table daftar perkaliam 6 x 6. Maka akan dibuat dalam bentuk baris dan kolom, barisnya akan dikalikan dengan setaip kolom. Pada baris 0 perkalian akan berulang dari kolom 0 sampai 5. Proses perkalian akan diulang kembali untuk baris 1 dan seterusnya. Untuk permasalahan tersebut maka diperlukan dua perulangan, perulangan yang berada didalam untuk mengulang kolom dan perulangan diluar untuk mengulang baris. Berikut contoh programnya:

```
void main()
{ int row;         // Outer loop counter
int col;    // Inner loop counter

for(row=1; row<=10; row++){
  {   for(col=1; col<=6; col++)
            cout << row*col << " ";
      cout << endl;
   }
}
```

Output :

1 2 3 4 5 6

2 4 6 8 10 12

3 6 9 12 15 18

4 8 12 16 20 24

5 10 15 20 25 30

6 12 18 24 30 36

Untuk melakukan perulangan bersarang, bentuk perulangan bisa dicampur, tidak harus sama. Contoh pada program berikut, dimana perulangan gabungan dari `while-do` dan `for`.

```
int counter, prakt=8;
double avg, score, tscore;
char ulang;
do{
      tscore = 0;
      for(counter =1; counter <=prakt; counter ++){
         cout<<"Masukkan nilai untuk praktikum ke "
            << counter << ": ";
          cin >> score;
          tscore += score;
      }
      avg = tscore/double(prakt);
      cout << "Nilai rata-rata " << avg << endl;
      cout << "Input mahasiswa lain (y/n)? ";
      cin >>ulang;
}while(ulang=='y' || ulang=='Y');
```

Semua bentuk perulangan dapat disusun menjadi nested loop.

Rangkuman

Dalam bab ini dibahas mengenai perulangan dalam C++. Perulangan dalam C++ yang dibahas, yaitu perulangan dengan ‘while’, ‘do while’ dan ‘for’. Dimana masing-masing mempunyai bentuk umum yang berbeda dan penggunaan yang berbeda juga.

Dalam perulangan ‘while’, perulangan terhadap pengerjaan instruksi dalam perulangan hanya dilakukan jika tes kondisi yang dilakukan diawal instruksi memberikan nilai true. Untuk perulangan ‘do while’, instruksi dalam perulangan akan dikerjakan dulu. Setelah itu baru dilakukan tes kondisi, jika menghasilkan nilai true maka instruksi dalam perulangan akan dikerjakan lagi. Catatan untuk perulangan ‘while’ dan ‘do while’, harus dilakukan perubahan tes kondisi sehingga perulangan bisa berhenti.

Perulangan yang ketiga yaitu ‘for’, biasanya digunakan ketika banyaknya perulangan sudah diketahui sebelumnya. Beberapa elemen dalam perulangan ‘for yaitu inisialisasi yaitu pemberian nilai awal pada variabel akan dilakukan tes kondisi, kemudian tes kondisi untuk menentukan apakah perulangan bisa dikerjakan dan yang terakhir perubahan nilai yaitu perubahan nilai untuk tes kondisi sehingga perulangan bisa mencapai batasnya.

Latihan

1. Program berikut mempunyai kesalahan dalam perulangan. Temukan kesalahan tersebut dan perbaiki.

```
void main()(
      int i = 0, value = 0;
      while (i <= 20)
      {
            if (i % 2 == 0 && i <= 10)
            value = value + i * i;
            else if (i % 2 == 0 && i > 10)
            value = value + i;
            else
            value = value - i;
      }
      cout << "value = " << value << endl;
   }
```

2. Rubahlah program no 1 menjadi bentuk `for` loop!

3. Rubahlah program berikut menjadi bentuk do `while` loop

```
int main()
{
    int counter,sum,N;
    cout << "masukkan bilangan positif :”;
    cin >> N;
    sum = 0;
    for(counter=1; counter<= N; counter++)
        sum = sum + counter;
```

```
    cout << "Jumlah  : " << N
<< " positive integers is " << sum<< endl;
    return 0;
}
```

4. Apakah output yang dihasilkan, jika pada potongan program dibawah diberi input 38 35 71 14 -1.?

```
sum = 0;
cin >> num;
while (num != -1)
{
      sum = sum + num;
      cin >> num;
}
cout << "Sum = " << sum << endl;
```

5. Koreksilah potongan program berikut agar melakukan proses penjumlaham 20 bilangan dengan benar.

```
sum = 0;
while (count < 20)
cin >> num;
sum = sum + num;
count++;
```

Latihan Program

1. Buatlah program untuk menampilkan semua bilangan genap yang terletak antara 20 sampai dengan 120, dengan menggunakan perulangan `while`.

2. Buat program untuk menampilkan seperti contoh berikut dengan perulangan `for`, dimana n diinputkan

   jika n = 5
   hasil: 55555
   4444
   333
   22
   1

3. Buat program untuk membuat deret 11 sampai nn dimana n diinputkan. Contoh jika n =3, maka output : 1 4 27. Gunakan fungsi looping do `while`.

4. Buatlah program untuk membuat daftar perkalian dari 1 sampai 10 dengan perulangan `for`.

5. Buat program untuk membuat deret 1^1 sampai n^n dimana n diinputkan. Contoh jika n =3, maka output : 1 4 27. Gunakan fungsi looping `while` / do `while`.

6

Array & String

Berisi penjelasan tentang variabel yang berbentuk array baik satu dimensi maupun multi dimensi beserta contoh penggunaannya. Selain itu juga dijelaskan operasi-operasi yang berhubungan dengan variabel array. Salah satu bentuk array yang mempunyai perilaku khusus adalah string.

Array adalah sekelompok data yang mendeskripsikan sesuatu yang sejenis dan disimpan ke dalam variabel dengan nama yang sama. Ada dua macam array yaitu array satu dimensi dan array multi-dimensi.

Array satu dimensi hanya mempunyai satu baris penggambaran atau satu kategori data, misalnya data dosen elektro, data peserta ujian dasar pemrograman, data buku perpustakaan.

Sedang array multi-dimensi mempunyai karakteristik penggambaran lebih dari satu baris atau kategori. Array dimensi dibuat dengan kolom dan baris. Masing-masing kolom menggambarkan suatu kategori dimana datanya mempunyai kategori lain yang digambarkan pada baris. Contoh array multidimensi misalnya negara dalam peta bisa dikategorikan berdasar bendera bisa dengan populasinya , bisa pula dengan luasnya dan sebagainya. Contoh lain array mahasiswa elektro bisa dikategorikan dalam angkatan dan program studinya.

Array Satu Dimensi

Seperti variabel lain, array harus dideklarasikan sebelum digunakan. Perbedaannya, dalam array kompailer harus diberitahu jenis array apa yang didefinisikan apakah array buku? Array mahasiswa? Array pakaian? Selain itu harus tahu juga berapa tempat yang diperlukan untuk array yang akan dipakai komputer. Ketika mendeklarasikan data array kompailer mengambil masing-masing data dalam lokasi yang sudah ditentukan.

Deklarasi Arrray Satu Dimensi

Sebelum digunakan dalam program, sebuah variabel harus dideklarasikan. Bentuk umum pendeklarasian array adalah sebagai berikut:

```
identifier nama[ukuran];
```

Identifer adalah tipe dari array apakah karakter, integer, real dan sebagainya. Kemudian diikuti dengan nama berdasar aturan C. Nama diikuti dengan tanda '[ ]', dimana didalamnya terdapat ukuran tempat yang dipesan untuk array tersebut.

Contoh deklarasi array satu dimensi seperti berikut:

```
char Nama[12];

float Nilai[100];

double sudut[360];
```

Inisialisasi Array Satu Dimensi

Seperti variabel yang lain juga, array dapat diinisialisasi. Bentuk umum inisialisasi array satu dimensi adalah sebagai berikut:

identifier name[dimension] = { element1, element2, ..., elementn};

Elemen1 menggambarkan elemen pertama dari array. Elemen dihitung dari 0 kemudian 1 dan sebagainya. Elemen2 menggambarkan elemen kedua dan seterusnya.

Contoh inisialisasi array satu dimensi seperti berikut:

```
int angka[6] = { 0, 1, 2, 3, 4, 5 };

double Akar[7]={0,1,1.414, 1.732, 2, 2.236, 2.449 };

char Warna[5] = { 'H', 'I', 'T', 'A', 'M' };
```

Berikut adalah beberapa contoh program dimana terdapat inisialisasi array satu dimensi:

1. Sebagai bilangan integer:

```
#include <iostream.h>
void main(){
    int Bilangan[6] = { 0, 1, 2, 3, 4, 5 };
```

```
    printf("Bilangan 1 = %d", Bilangan[0]);
    printf("\n Bilangan 2 =%d " , Bilangan[1]);
    printf("\n Bilangan 3 =%d " , Bilangan [2]);
    printf("\n Bilangan 4 = %d " , Bilangan [3]);
    printf("\n Bilangan 5 =%d ", Bilangan[4]);
    printf("\nBilangan6= \n\n%d", Bilangan [5]);
}
```

Hasil ekeskusi program sebagai berikut:

Bilangan 1 = 0
Bilangan 2 = 1
Bilangan 3 = 2
Bilangan 4 = 3
Bilangan 5 = 4
Bilangan 6 = 5

2. Sebagai karakter

```
#include <iostream.h>

void main()
{
char Country[10] = { 'S', 'w', 'a', 'z', 'i',
'l', 'a', 'n', 'd' };

printf( "%s di Africa!\n\n", Country);
}
```

Hasil ekeskusi program sebagai berikut:
Swaziland di Africa!

Dalam contoh diatas berarti bahwa Country [0] = 'S', Country [1] = 'w', Country [2] = 'a' dan seterusnya.

3. Sebagai bilangan real

```
#include <iostream.h>
void main(){
float Akar[7] = { 0, 1, 1.414, 1.732, 2, 2.236,
2.449 };
```

```
printf("\tBilangan\t\Akar");
printf( "\n  ------------------------------");
printf("\n\t1\t\t %f" , Akar[0]);
printf("\n\t2\t\t%f " , Akar1]);
printf("\n\t3\t\t%f " , Akar[2]);
printf( "\n\t4\t\t%f " , Akar[3]);
printf("\n\t5\t\t%f " , Akar[4]);
printf( "\n\t6\t\t%f " , Akar[5]);

}
```

Hasil eksekusi program sebagai berikut:

```
    Bilangan   Akar
 ------------------------------
    1          0
    2          1
    3          1.414
    4          1.732
    5          2
    6          2.236
```

Dalam contoh diatas berarti bahwa `Akar` [0] = 0, `Akar` [1] = 1, `Akar` [2] = 1.414 dan seterusnya.

Inisialisasi dapat dilakukan bersama jika variabel-variabel tersebut mempunyai tipe sama. Seperti pada contoh berikut:

```
int Start[0] = 10, Start[1] = 10, Start[2] = 10,
End[0] = 14, End[1] = 18, End[2] = 18;
```

Array yang telah dijelaskan diatas adalah array yang mempunyai ukuran tertentu. Hal ini digunakan jika sudah diketahui berapa anggota yang terdapat dalam array. Terkadang keadaannya tidak seperti itu. Karena itu jika tidak diketahui berapa anggota array , maka ukuran array dapat dikosongkan, tapi diberikan elemen yang berada didalamnya. Array yang didefinisikan tanpa ukuran tertentu dideklarasikan sebagai berikut:

indentifier Nama Array[] = Anggota;

Contoh inisialisasi array dengan ukuran array yang tidak ditentukan:

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

void main()
{
    char State[] = "Indonesia";
    char Answer;

    printf("Anda tinggal di %s(y = Ya, t = Tidak)?
    ", State);
    scanf("%c", Answer);

    Answer = tolower(Answer);

    if( Answer == 'y' )
      printf("\nAnda tinggal di  %s", State);
    else
      printf("\nAnda tinggal dinegara lain!!!");
}
```

Hasil eksekusi program sebagai berikut:

Anda tinggal di Indonesia (y = Yes, n = Tidak)? t
Anda tinggal di negara lain!!!

Array Dua Dimensi

Array berdimensi dua adalah array dari array berdimensi satu. Dengan kata lain array dua dimensi adalah array dimana masing-masing anggotanya juga array. Contoh seperti tabel berikut:

Negara\Data	Luas(km2)	Jumlah Penduduk
Amerika Serikat	9.629.091	272.639.608
Kamerun	475.440	15.456.092
Guatemala	108.890	12.335.580

Itali	301.23056.	735.130
Oman	212.4602	446.645

Deklarasi Array Dua Dimensi

Array dua dimensi terdiri dari baris dan kolom. Masing-masing baris menggambarkan suatu kategori dimana datanya mempunyai kategori lain yang digambarkan pada kolom. Contoh array dua dimensi misalnya negara dalam peta bisa dikategorikan berdasar bendera bisa dengan populasinya , bisa pula dengan luasnya dan sebagainya. Contoh lain array mahasiswa elektro bisa dikategorikan dalam angkatan dan program studinya.

Karena array dua dimensi terdiri dari baris dan kolom maka, digunakan dua tanda '[ ]'. Bentuk umum array dua dimensi:

```
identifier nama[ukuran1][ukuran2]
```

Identifer adalah tipe dari array apakah karakter, integer, real dan sebagainya. Kemudian diikuti dengan nama berdasar aturan C. Nama diikuti dengan dua tanda '[ ]', dimana didalamnya terdapat dua ukuran tempat yang dipesan untuk array tersebut. Ukuran1 menunjukkan ukuran baris yang dipesan, sedang ukuran2 merupakan ukuran kolom dari array tersebut.

Contoh deklarasi array dua dimensi seperti berikut:

```
int JumlahPelajarPerKelas[12][50];
```

Deklarasi diatas membuat kelompok pertama 12 elemen, sehingga memerlukan array 12 kelas. Sedang masing-masing array berisi 50 elemen, jadi masing-masing dari 12 anggota kelompok adalah array dari 50 item. Secara sederhana dikatakan, deklarasi ini membuat 12 kelas dan masing-masing kelas berisi 50 pelajar.

Inisialisai Array Dua Dimensi

Sebelum anggota dari array digunakan harus mempunyai nilai terlebih dahulu. Seperti dalam array satu dimensi, ada dua cara untuk memecahkan masalah ini, pertama dengan membuat inisialisasi array sedang yang kedua dengan memberikan nilai saat ekesekusi.

Inisialisasi dilakukan sama seperti pada array satu dimensi. Karena array dua dimesni digambarkan sebagai baris dan kolom maka jumlah elemen yang dimiliki array dua dimensi dapat ditentukan dari hasil perkalian jumlah baris * jumlah kolom. Sehingga banyaknya nilai yang akan diinisialisasi pada array harus lebih kecil atau sama dengan jumlah elemen array tersebut. Bentuk umum inisialisasi array satu dimensi adalah sebagai berikut:

identifier nama[ukuran1][ukuran2] = { elemen1, elemen2, ..., };

Contoh inisialisasi array berdimensi dua seperti berikut:

```
float bil[2] [3] = { 1,2,3, 4,5,6}
```

Dari contoh inisialisasi tersebut maka berarti bahwa elemen bil [0] [0] = 1, elemen bil [0] [1] = 2, elemen bil [0] [2] = 3, elemen bil [1] [0] = 4, elemen bil [1] [1] = 5, elemen bil [1] [2] = 6.

Berikut adalah contoh program dimana terdapat inisialisasi array dua dimensi:

```
#include <iostream>
int main() {
    float  Jarak[2][4]  =  {44.14,  720.52,  96.08,
    468.78, 6.28, 68.04, 364.55, 6234.12};

    cout << "Anggota array";
    cout << "\njarak[0][0]: " << Jarak[0][0]);
    cout << "\njarak[0][1]: " << jarak[0][1]);
    cout << "\njarak[0][2]: " << jarak[0][2]);
    cout << "\njarak[0][3]: " << jarak[0][3]);
    cout << "\njarak[1][0]: " << jarak[1][0]);
```

```
    cout << "\njarak[1][1]: " << jarak[1][1]);
    cout << "\njarak[1][2]: " << jarak[1][2]);
    cout << "\njarak[1][3]: " << jarak[1][3]);

    return 0;
}
```

Hasil dari program diatas :

Anggota array
Jarak[0][0] : 44.14
Jarak [0][1]: 720.52
Jarak [0][2]: 96.08
Jarak [0][3]: 468.78
Jarak [1][0]: 6.28
Jarak [1][1]: 68.04
Jarak [1][2]: 364.55
Jarak [1][3]: 6234.12

Agar elemen array pada contoh diatas mudah dibaca ketika inisialisasi, maka nilai dapat ditulis perbaris, seperti contoh berikut:

```
float Jarak[2][4] ={44.14,720.52,96.08,468.78,
      //baris 1
                    6.28,68.04,64.55,6234.12 };
      //baris2
```

Atau untuk lebih spesifik maka dapat diberi tanda '{ }' untuk membedakan setiap barisnya, seperti berikut

```
float Jarak[2][4]={{44.14, 720.52,  96.08,  468.78,}
      //baris 1

                {6.28,  68.04, 364.55, 6234.12 }};
      //baris2
```

Pemrosesan Array dua Dimensi

Untuk menelusuri array, harus diketahui berapa kolom dan baris yang dimiliki array. Untuk memproses array dua dimensi dapat digunakan dua perulangan for. Seperti dalam contoh berikut :

```
#include <iostream>

int main()
{
   float Distance[][4] = { { 44.14, 720.52,  96.08,
   468.78 },{  6.28,  68.04, 364.55, 6234.12 }
                };

   printf("Anggota array");
   for(int i = 0; i < 2; ++i)
      for(int j = 0; j < 4; ++j)
            printf("\nDistance  [%d][%d]:  %f"  ,  i,
   j, Distance[i][j]);

   return 0;
   }
```

Array Multidimensi

Array multi-dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua.Bentuk umumnya yaitu:

```
tipe_array nama_array[ukuran1][ukuran2]...[ukuranN];
```

Contoh deklarasi array tiga dimensi:

```
float X[2][4][3];
```

Contoh program dengan array multi-dimensi:

```
#include "stdio.h"
#include "conio.h"

void main()
{  int i, j, k;

   static int data_huruf[2][8][8] =
   { { { 1, 1, 1, 1, 1, 1, 1, 0 },
   { 1, 1, 0, 0, 0, 0, 1, 0 },
   { 1, 1, 0, 0, 0, 0, 0, 0 },
   { 1, 1, 1, 1, 1, 1, 1, 0 },
   { 0, 0, 0, 0, 1, 1, 1, 0 },
   { 1, 0, 0, 0, 1, 1, 1, 0 },
   { 1, 1, 1, 1, 1, 1, 1, 0 },
   { 0, 0, 0, 0, 0, 0, 0, 0 }
      },
   { { 1, 1, 0, 0, 0, 1, 1, 0 },
   { 1, 1, 0, 0, 0, 1, 1, 0 },
   { 1, 1, 0, 0, 0, 1, 1, 0 },
   { 1, 1, 1, 1, 1, 1, 1, 0 },
   { 1, 1, 1, 0, 0, 1, 1, 0 },
   { 1, 1, 1, 0, 0, 1, 1, 0 },
   { 1, 1, 1, 0, 0, 1, 1, 0 },
   { 0, 0, 0, 0, 0, 0, 0, 0 }
   }  };

   for(i=0; i<2; i++)
   { for(j=0; j<8; j++)
      { for(k=0;k<8; k++)
            if (data_huruf[i][j][k])
                  putchar('\xDB');
            else
                  putchar(" ");
            puts("");
      }
      puts("");
   }
   getch(); }
```

String

Untuk membuat suatu string diperlukan array karena C++ hanya mempunyai tipe char.

Contoh deklarasi string:

```
char Nama[20];
char Judul[40];
char Hari[4];
```

Untuk inisialisasi variable string sama seperti inisialisasi data array. Contoh inisialisasi string:

```
char Nama[6] = { 'F', 'a', 'r', 'i', 's' };
```

atau dapat dilakukan seperti berikut:

```
char Nama[12] = "Faris";
```

Seperti array yang lain, masing-masing anggota array karakter dapat diakses berdasar indeksnya.

Akses String

Pengeloaan string mempunyai perbedaan dengan array tipe lainnya. Input sebuah string dapat dilakukan tanpa menggunakan indeks. Contoh seperti berikut :

```
void main()
{     char Hari[7];

      cout << "Masukkan nama hari: ";
      cin >> Hari;
}
```

Begitu juga untuk mencetak sebuah string, bisa langsung dituliskan nama variabel string tanpa menggunakan indeks.

Contoh:

```
void main()
{  char Hari[12];
   char EndMe[] = "\nPress any key to continue...";

   cout << "Masukkan nama hari: ";
   cin >> Hari;
   cout << "Sekarang hari " << Hari;
   cout << EndMe;
}
```

Jika sebuah string dicetak dengan indeksnya, maka yang akan tercetak adalah salah satu karakter dari string tersebut.

Contoh:

```
void main() {
   char Hari[12]= "Selasa";

   cout << "Sekarang hari " << Hari << endl;
   cout <<"Karakter Hari indeks ke 3 : " << Hari[3];
}
```

Ouput:

Sekarang hari Selasa

Karakter Hari indeks ke 3 : a

Array String

Jika diperlukan daftar string lebih dari satu, maka digunakan array 2 dimensi, dimana dimensi pertama merupakan jumlah string pada variabel tersebut dan dimensi kedua menunjukkan banyaknya karakter untuk masing-masing string.

Contoh:

```
char NamaMhs[4][10]={"Adi","Budi","Charles","Doni"};
```

NamaMhs adalah array dengan 4 string, dan tiap string mempunyai maksimal 9 karakter (+1 untuk karakter null-terminated).

Untuk membaca setiap string, dituliskan nama variabel dan indeks dari baris. Contoh:

```
void main() {
char NamaMhs[4][10]={"Adi","Budi","Charles","Doni"};
cout << "Nama: ";
cout << "\nMhs 1: " <<NamaMhs[0];
cout << "\nMhs 2: " << NamaMhs[1];
cout << "\nMhs 3: " <<NamaMhs[2];
cout << "\nMhs 4: " << NamaMhs[3];
}
```

Deklarasi `char NamaMhs[4][10],` berarti string `NamaMhs` terdiri dari 4 string dan masing-masing string mempunyai panjang maksimal 10 karakter.

Fungsi-fungsi untuk string

1. Fungsi strlen().
   Untuk mengetahui banyaknya karakter pada suatu string.
   Sintaks: int strlen(const char* Value);
   Contoh:

```
void main()
{ char School[] = "UNESA Surabaya";
   int Length = strlen(School);
   cout << "Panjang " << School << " adalah "
        << Length << " karakter\n\n";
}
```

   Output:

   Panjang UNESA Surabaya adalah 14 karakter

2. Fungsi strcat()
   Untuk menambahkan dua string
   Sintaks : char *strcat(char *Destination, const char *Source);
   - Destination adalah string pertama yang akan ditambahkan dan untuk menyimpan hasil penambahan string.
   - Source adalah string kedua yang ingin ditambahkan pada string pertama.

Contoh:

```
void main()
{  char Kata1[] = "UNESA ";
   char *Kata2 = "Surabaya";
   cout << "Kata1 = " << Kata1;
   strcat(Kata1, Kata2);
   cout << "\nSesudah penggabungan<<endl;
   cout << "Kata1 = " << Kata1 << endl;
}
```

Output:

Kata1 = UNESA
Sesudah penggabungan
Kata1 = UNESA Surabaya

3. Fungsi strncat()

Untuk menambahkan sejumlah karakter pada satu string kesuatu string yang lain.

Sintaks: char* strncat(char* Destination, const char* Source, int Number);

- number adalah banyaknya karakter pada source yang akan dijumlahkan.

Contoh:

```
void main()
{ char Kata1[] ="UNESA ";
  char *Kata2 = "Surabaya";
  cout << "Kata1 = " << Kata1;
  strncat(Kata1, Kata2, 3);
  cout << "\nSesudang penggabungan<<endl;
  cout << "Kata1 = " << Kata1 << endl;
}
```

Output:

Kata1 = UNESA
Sesudah penggabungan
Kata1 = UNESA Sur

4. Fungsi strcpy()

Untuk menyalin satu string ke string yang lain.

Sintaks : char* strcpy(char* Destination, const char* Source);

- Destination adalah string yang akan diganti,
- Source adalah string yang akan disalin.

– Dua kegunaan fungsi strcpy() :

Fungsi strcpy() dapat digunakan untuk :

- inisialisasi string, contoh :

```
int main(){
  char Kata1[10];
  strcpy(Kata1,"UNESA Surabaya");
  cout << "Kata1 = " << Kata1 << endl;
}
```

Output:

Kata1 = UNESA Surabaya

- menyalin string, contoh :

```
void main()
{ char Kata1[] = "UNESA ";
  char *Kata2 = "Surabaya";

   strcpy(Kata1, Kata2);
   cout << "Kata1 = " << Kata1 << endl;
   cout << "Kata1 = " << Kata2 << endl;
}
```

Output :

Kata1 = Surabaya
Kata2 = Surabaya

5. Fungsi strncpy()

Untuk menyalin sejumlah karakter dari string yang lain.

Sintaks: char* strncpy(char* Destination, const char* Source, int Number);

– Number menunjukkan banyaknya karakter dari source yang akan disalin.

Contoh:

```
void main()
{ char Kata1[]= "UNESA ";
  char *Kata2 = "Surabaya";

   cout << "Kata1 = " << Kata1 << endl;
   cout << "Kata2 = " << Kata2 << endl;
   strncpy(Kata1,Kata2,3);
```

```
    cout << "Setelah menyalin 3 huruf dari
              Kata2"<<endl;
    cout << "Kata1 = " << Kata1 << endl;
    cout << "Kata2 = " << Kata2 << endl;
}
```

Ouput:

Kata1 = UNESA
Kata2 = Surabaya
Setelah menyalin 3 huruf dari Kata2
Kata1 = SurSA
Kata2 = Surabaya

6. Fungsi strcmp

Digunakan untuk membandingkan 2 string .
Sintaks: int strcmp(const char* S1, const char* S2);
Hasil perbandingan:

- negative jika S1 < S2
- 0 jika S1 dan S2 sama
- positive jika S1 > S2

Contoh

```
void main()
{  char *Kata1 = "UNESA ";
   char *Kata2 = "Surabaya";
   char *Kata3 = "Unesa";
   char *Kata4 = "Surabaya" ;

    int banding1 = strcmp(Kata1, Kata3);
    int banding2 = strcmp(Kata3, Kata1);
    int banding3 = strcmp(Kata2, Kata4);
    cout<<"Hasil  perbandingan  1  =  "<<banding1<<
    endl;
    cout<<"Hasil perbandingan 2 = "<< banding2 <<
    endl;
    cout<<"Hasil perbandingan 3 = "<<banding3;
}
```

Output:

Hasil perbandingan 1 = -1
Hasil perbandingan 2 = 1
Hasil perbandingan 3 = 0

7. Fungsi stricmp()
   Digunakan untuk melakukan perbandingan dua string tanpa memandang huruf besar atau huruf kecil.
   Sintaks: int stricmp(const char* S1, const char* S2);
   Hasil perbandingan:
   - negative jika S1 < S2
   - 0 jika S1 dan S2 sama
   - positive jika S1 > S2

   Contoh:

```
int main()
{
   char *Kata1 = "UNESA";
   char *Kata2 = "Surabaya";
   char *Kata3 = "Unesa";
   char *Kata4 = "Surabaya" ;

  int banding1 = stricmp(Kata1, Kata3);
  int banding2 = stricmp(Kata3, Kata1);
  int banding3 = stricmp(Kata2, Kata4);
  cout << "Hasil perbandingan 1 = " << banding1
  << endl;
  cout << "Hasil perbandingan 2 = " << banding2
  << endl;
  cout << "Hasil perbandingan 3 = " << banding3;
}
```

   output:

   Hasil perbandingan 1 = 0
   Hasil perbandingan 2 = 0
   Hasil perbandingan 3 = 0

8. Fungsi strlwr()
   Digunakan untuk mengkonversikan string ke huruf kecil.
   Sintaks : char *strlwr(const char *S);
   Contoh:

```
void main() {
  char Kata1[] = "UNESA Surabaya";*Kata2 ;
  cout << "Kata1 = " << Kata1 << endl;
  Kata2 = strlwr(Kata1);
  cout << "Kata2 = " << Kata2 << endl;
}
```

Ouput:
Kata1 = UNESA Surabaya
Kata2 = unesa surabaya

9. Fungsi strupr()
Untuk mengkonversikan string ke huruf besar.
Sintaks : char *strupr(const char *S);
Contoh:

```
void main()
{ char Kata1[] = "UNESA Surabaya";
  char *Kata2 ;

   cout << "Kata1 = " << Kata1 << endl;
   Kata2 = strupr(Kata1);
   cout << "Kata2 = " << Kata2 << endl;
}
```

Output :
Kata1 = UNESA Surabaya
Kata2 = UNESA SURABAYA

10. Fungsi strchr()
Memeriksa apakah huruf pertama pada string sesuai dengan karakter tertentu.
Sintaks : char* strchr(const char* S, char c);
- S → string yang akan diperiksa
- C → karakter yang dicari
- Hasil: jika c ada pada huruf pertama S fungsi akan memberikan hasil string S, jika tidak ada akan memberikan nilai NULL.

Contoh:

```
int main()
{
   char*Kata1 = "UNESA Surabaya";
   char *Kata2, *Kata3, *Kata4 ;

   Kata2 = strchr(Kata1, 'a');
   Kata3 = strchr(Kata1, 'S');
   Kata4 = strchr(Kata1, 'k');
```

```
    cout << "Kata2 = " << Kata2 << endl;
    cout << "Kata3 = " << Kata3 << endl;
    cout << "Kata4 = " << Kata4 << endl;
}
```

Output:

Kata2 = abaya
Kata3 = SA Surabaya
Kata4 =

11. Fungsi strrchr()

Memeriksa apakah huruf terakhir pada string sesuai dengan karakter tertentu

Sintaks : char* strrchr(const char* S, char c);

Hasil : jika c ada pada huruf terakhir S fungsi akan memberikan hasil karakter yang sesuai, jika tidak ada akan memberikan nilai NULL.

Contoh :

```
int main()
{
    char*Kata1 = "UNESA Surabaya";
    char *Kata2, *Kata3, *Kata4 ;

    Kata2 = strrchr(Kata1, 'a');
    Kata3 = strrchr(Kata1, 'S');
    Kata4 = strrchr(Kata1, 'k');
    cout << "Kata2 = " << Kata2 << endl;
    cout << "Kata3 = " << Kata3 << endl;
    cout << "Kata4 = " << Kata4 << endl;

}
```

Output :

Kata2 = a
Kata3 = Surabaya
Kata4 =

Rangkuman

Array adalah sekelompok data yang mendeskripsikan sesuatu yang sejenis dan disimpan ke dalam variabel dengan nama yang sama. Ada dua macam array yaitu array satu dimensi dan array multi-dimensi. Dalam pembahasan ini array multi-dimensi dibahas dalam dua bagian yaitu array dua dimensi dan array yang mempunyai dimensi lebih dari dua.

Seperti variabel lain sebelum digunakan array harus dideklarasikan terlebih dulu. Bedanya dalam mendeklarasikan array harus dicantumkan juga berapa ukuran dari array tersebut. Pada array satu dimensi hanya terdapat satu ukuran array Sedang pada array dua dimensi, array mempunyai dua ukuran yang biasanya direpresentasikan sebagai banyaknya baris dan banyaknya kolom.

Seperti juga pada variabel biasa, sebelum digunakan untuk pemrosesan array juga perlu diberi nilai yang salah satunya melalui proses inisialisasi. Inisialisasi array dilakukan dengan pemberian nilai pada masing-masing elemen yang banyaknya lebih kecil atau sama dengan ukuran array.

Latihan

1. Apakah output dari potongan program berikut?

```
int temp[5];
for (int i = 0; i < 5; i++)
temp[i] = 2 * i - 3;
for (int i = 0; i < 5; i++)
cout << temp[i] << " ";
cout << endl;
temp[0] = temp[4];
temp[4] = temp[1];
temp[2] = temp[3] + temp[0];
for (int i = 0; i < 5; i++)
cout << temp[i] << " ";
cout << endl;
```

2. Misalkan `list` adalah sebuah array yang mempunyai ukuran 5 dan bertipe int, Apakah yang akan tersimpan pada `list` setelah program berikut di eksekusi

```
for (int i = 0; i < 5; i++)
{
    list[i] = 2 * i + 5;
    if (i % 2 == 0)
    list[i] = list[i] - 3;
}
```

3. Perbaikilah program berikut agar memberikan output yang benar.

```
myList;
int myList[10];

for (int i = 1; i <= 10; i++)
cin >> myList;
for (int i = 1; i <= 10; i++)
cout << myList[i] << " ";
cout << endl;
```

4. Dari beberapa pernyataan berikut, manakah deklarasi yang valid?

a. `int a[5] = {0, 4, 3, 2, 7};`
b. `int b[10] = {0, 7, 3, 12};`
c. `int c[7] = {12, 13, , 14, 16, , 8};`
d. `double lengths[] = {12.7, 13.9, 18.75, 20.78};`
e. `char name[8] = "Samantha";`

5. Apakah output dari program berikut?

```
int list[] ={6, 8, 2, 14, 13};

for (int i = 0; i < 4; i++)
list[i] = list[i] - list[i + 1];

for (int i = 0; i < 5; i++)
cout << i << " " << list[i] << endl;
```

Latihan Program

1. Buatlah program yang akan memberikan hasil berupa transpose dari suatu matriks. Elemen dalam matrik diinputkan.

2. Buat program untuk membalik string masukan. (Petunjuk : gunakan fungsi strlen() untuk mendapatkan panjang kalimat)

   Contoh :
   Kalimat yang mau dibalik : Saya sedang belajar C
   Hasil pembalikan kalimat : C rajaleb gnades ayaS

3. Buat program untuk menampilkan deret Fibonacci sbb :
   Input : jumlah deret
   Output : deret[i] = deret[i-1] + deret[i-2]

4. Buat program untuk menginputkan n bilangan Kemudian cetaklah semua bilangan dimana bilangan yang sama cukup dicetak satu kali. Contoh, jika diinputkan : 1 2 3 1 3 4, maka output : 1 2 3 4.

5. Buat program untuk memasukkan data nama mahasiswa dalam satu kelas. Kemudian cetaklah daftar nama mahasiswa yang mempunyai huruf pertama sesuai masukan

7

Fungsi

Dalam bab ini, akan dijelaskan bagaimana caranya membuat fungsi, bagimana cara meneruskan data kepada fungsi, memproses data yang dikirimkan, dan mengembalikan hasilnya. Selain itu juga dijelaskan bagaimana cara pemanggilan fungsi oleh fungsi yang lain, karena fungsi hanya akan bekerja saat dipanggil atau diperintahkan untuk melaksanakan tugasnya.

Program profesional dirancang, dikodekan, dan diuji seperti halnya perangkat keras: sebagai satu set modul yang terintegrasi untuk melakukan keseluruhan secara lengkap. Analogi seperti sebuah mobil; satu modul utama adalah mesin, yang lain adalah transmisi, sepertiga sistem pengereman, keempat bodi, dan sebagainya. Semua modul ini dihubungkan bersama dan ditempatkan di bawah kendali pengemudi, yang dapat dibandingkan dengan program utama () modul. Seluruhnya sekarang beroperasi sebagai unit yang lengkap, mampu melakukan pekerjaan yang bermanfaat, seperti mengemudi ke toko. Selama proses perakitan, setiap modul dibangun, diuji, dan ditemukan bebas dari cacat (bug) sebelum dipasang di produk akhir.

Dalam analogi ini, setiap komponen mobil utama dapat dibandingkan dengan suatu fungsi. Misalnya, pengemudi memanggil mesin saat pedal gas ditekan. Mesin menerima input bahan bakar, udara, dan listrik untuk merubah permintaan pengemudi menjadi produk yang bermanfaat — tenaga — dan kemudian mengirim output ini ke transmisi untuk diproses lebih lanjut. Transmisi menerima output mesin dan mengubahnya menjadi roda yang bisa berjalam. Input tambahan untuk transmisi adalah pemilihan roda gigi pengemudi (maju, mundur, netral, dan seterusnya).

Mesin, transmisi, dan modul lainnya “hanya tahu” yang berhubungan dengan input dan output. Pengemudi tidak perlu mengetahui operasi internal dari modul yang dikontrol. Yang diperlukan hanyalah mengetahui apa yang dilakukan setiap modul dan bagaimana "memanggil" ketika output modul diperlukan.

Komunikasi antar modul dibatasi hanya dengan meneruskan input ke setiap modul saat dipanggil untuk melakukan tugasnya, dan setiap modul beroperasi dengan cara yang cukup independen. Programer menggunakan pendekatan modular yang sama untuk membuat dan memelihara program C++ yang handal dengan menggunakan fungsi.

Setiap program C ++ harus berisi fungsi utama (). Selain diperlukan fungsi ini, program C ++ juga dapat berisi sejumlah fungsi lainnya. Dalam bab ini akan dibahas tentang caranya untuk menulis fungsi-fungsi ini, meneruskan data kepada fungsi, memproses data yang dikirimkan, dan mengembalikan hasilnya.

Deklarasi Fungsi

Dalam membuat fungsi C ++, harus diperhatikan bagaimana perilaku fungsi tersebut dan bagaimana interaksinya dengan fungsi lain, seperti main (). Interaksi dengan fungsi lain termasuk bagaimana meneruskan data ke fungsi dengan benar saat dipanggil (input) dan bagaimana mengembalikan nilai dari suatu fungsi (keluaran). Bagian ini menjelaskan bagian pertama dari antarmuka, meneruskan data ke suatu fungsi dan bagimana fungsi menerima, menyimpan, dan memproses data yang dikirimkan dengan benar.

Seperti fungsi matematika, fungsi dipanggil, atau digunakan, dengan nama fungsinya dan mengirimkan data yang diperlukan fungsi, sebagai argumen. Fungsi yang dipanggil harus dapat menerima data dikirimkan oleh fungsi yang melakukan panggilan. Setelah fungsi yang dipanggil berhasil menerima data, data dimanipulasi untuk memberi hasil yang diperlukan.

Sebelum suatu fungsi dapat dipanggil, harus dideklarasikan ke fungsi yang akan melakukan pemanggilan. pernyataan deklarasi untuk suatu fungsi disebut sebagai prototipe fungsi. Prototipe fungsi memberitahu fungsi yang melakukan panggilan tipe nilai yang akan dikembalikan, jika ada, dan tipe data dan urutan nilai-nilai yang harus dikirim fungsi pemanggil ke fungsi yang dipanggil.

Bentuk umum deklarasi fungsi atau prototipe adalah sebagai berikut:

```
ReturnValue NamaFungsi ( Argumen );
```

Sebuah penugasan atau permintaan, seperti fungsi terdiri dari tiga bagian yaitu, tujuan(NamaFungsi), kebutuhan(Argumen) dan harapan(*ReturnValue*).

Tujuan dari fungsi adalah mengidentifikasi apakah yang harus dikerjakan fungsi. Kapan fungsi diminta, maka fungsi akan memberikan hasilnya. Contoh, jika fungsi diminta untuk menghitung luas segitiga, maka hasilnya adalah luas segitiga. Hasil dari fungsi dinamakan return value.

Ada dua bentuk harapan dari sebuah fungsi yaitu nilai tertentu dan penugasan/permintaan sederhana. Jika diinginkan melakukan suatu permintaan tanpa menginginkan suatu hasil, maka fungsi dikualifikasikan sebagai 'void' dan tidak mempunyai return value. Deklarasinya seperti berikut:

```
void NamaFungsi(Argumen);
```

Tetapi jika suatu fungsi diharapkan akan memberikan nilai tertentu maka return value berupa tipe data tertentu, seperti int, float, double, atau char. Berikut adalah contoh deklarasi fungsi yang mempunyai return value :

```
double NamaFungsi(Argumen);

char NamaFungsi(Argumen);

float NamaFungsi(Argumen);
```

Nama sebuah fungsi mengikuti aturan penamaan variable dalam C++. Nama fungsi seharusnya sesuai dengan apa yang ingin diharapkan untuk dikerjakan sebuah fungsi. Contoh:

```
double HitungLuas(Argumen);

int Nilai(Argumen);
```

Ketika melakukan tugasnya fungsi mungkin memerlukan sesuatu. Contoh, ketika fungsi digunakan untuk menghitung luas segiempat maka fungsi memerlukan sisi-sisi segiempat. Selain itu fungsi yang digunakan untuk mendapatkan nama depan seorang pelajar mungkin tidak membutuhkan sesuatu untuk menyelesaikan tugasnya.

Jadi fungsi ada yang membutuhkan sesuatu ada yang tidak. Kebutuhan fungsi ini ditempatkan dalam tanda '()' setelah nama fungsi. Fungsi yang tidak membutuhkan sesuatu maka dalam '()' kosong. Fungsi dapat memerlukan lebih dari satu kebutuhan. Kebutuhan fungsi ini dinamakan argumen. Argumen didefinisikan dengan tipe data dan nama argumen.

1. Suatu fungsi digunakan untuk menghitung luas bujursangkar, diharapkan akan memberikan nilai luas bujursangkar dan membutuhkan panjang sisi bujursangkar, maka akan dideklarasikan sebagai berikut:

```
Double hitung_luas(double sisi);
```

2. Suatu fungsi digunakan untuk memasukkan sebuah nilai ujian, diharapkan fungsi tersebut akan memberikan nilai ujian yang sudah dimasukkan dan fungsi tersebut tidak membutuhkan sesuatu untuk mengerjakan tugasnya, maka deklarasi menjadi seperti berikut:

```
int input_nilai();
```

Definisi Fungsi

Suatu fungsi didefinisikan ketika ditulis. Setiap fungsi didefinisikan satu kali (yaitu, ditulis sekali) disuatu program dan kemudian dapat digunakan oleh fungsi lain dalam program yang sesuai.

Seperti fungsi main (), setiap fungsi C ++ terdiri dari dua bagian, header fungsi dan sebuah badan fungsi. Tujuan header fungsi adalah untuk mengidentifikasi tipe data dari nilai yang dikembalikan fungsi, memberi nama fungsi, dan menentukan jumlah, urutan dan tipe argumen yang diharapkan fungsi. Tujuan badan fungsi adalah untuk mengoperasikan data yang dikirimkan dan mengembalikan nilai ke fungsi yang memanggil.

Header fungsi selalu berada baris pertama dari suatu fungsi dan berisi tipe nilai yang dikembalikan, nama fungsi dan tipe data argumennya. Contoh header fungsi :

`void findMax (intx, inty)` ← tidak ada titik koma

findMax () tidak mengembalikan nilai apa pun dan menerima dua nilai integer.

Nama argumen dalam tanda kurung di header disebut parameter formal fungsi (atau parameter, singkatnya). Oleh karena itu, parameter x digunakan untuk menyimpan nilai pertama dilewatkan ke findMax () dan parameter y digunakan untuk

menyimpan nilai kedua yang dilewatkan pada saat panggilan fungsi. Fungsi tidak tahu dari mana nilai-nilai itu berasal saat panggilan dibuat dari main (). Bagian pertama dari prosedur panggilan yang dijalankan oleh komputer melibatkan proses pergi ke variabel `firstnum` dan `secnum` dan mengambil nilai yang disimpan. Nilai-nilai ini kemudian diteruskan ke `findMax ()` dan disimpan dalam parameter x dan y .

Untuk menggunakan fungsi maka kompiler harus tahu apa yang akan dikerjakan suatu fungsi. Kadang-kadang tidak perlu mendeklarasikan fungsi sebelumnya, tetapi harus memberitahu kompiler kelakuan apa yang diharapkan.

Agar kompiler mengetahui apa yang dikerjakan fungsi, maka fungsi harus didefinisikan yang berarti juga menggambarkan perlakuan fungsi. Bentuk umum pendefinisian fungsi :

```
ReturnValue NamaFungsi ( parameter )
{
        Badan fungsi
}
```

Fungsi didefinisikan menggunakan aturan seperti aturan pada fungsi main(). Pendefinisian dimulai dengan return valuenya (jika tidak ada digunakan void, kemudian diikuti dengan nama fungsi, argumen(jika ada) dan badan fungsi. Ketika fungsi didefinisikan, maka fungsi lain dapat menggunakannya.

Badan fungsi menggambarkan apa yang diharapkan untuk dikerjakan fungsi. Badan dimulai dengan tanda '{'dan diakhiri dengan tanda ´}´. Contoh sebuah fungsi sederhana, dimana fungsi tidak mempunyai pengembalian nila dan parameter:

```
void Message()
{
      printf(¨contoh Fungsi dalam C++¨);
}
```

Contoh definisi fungsi dengan parameter dan pengembalian nilai:

```
double volume_kubus(double panjang_sisi)
{
      double volume = panjang_sisi * panjang_sisi *

      panjang_sisi;
      return volume;
 }
```

Pemanggilan Fungsi

Salah satu alasan peggunaan fungsi dalam sebuah program adalah untuk membagi-bagi tugas sehingga dapat dengan mudah diketahui ketika terdapat masalah. Fungsi bersifat individual, sebuah fungsi tidak perlu mengetahui apa yang dikerjakan fungsi lain dan apa yang dibutuhkan fungsi lain.

Ketika fungsi telah didefinisikan, maka fungsi lain dapat menggunakan hasil dari pekerjaannya. Misalnya terdapat dua fungsi A dan B. Jika fungsi A memerlukan hasil dari fungsi B, maka fungsi A harus memanggil nama fungsi B.

Memanggil fungsi cukup mudah. Hanya menggunakan nama fungsi dan melampirkan data yang harus diteruskan ke fungsi. Data harus sesuai dengan parameter fungsi yang terletak dalam dalam tanda kurung yang mengikuti nama fungsi. Data jua harus menggunakan urutan dan tipe yang sama dengan yang dideklarasikan dalam prototipe fungsi. Data yang dikirimkan kepada fungsi disebut argumen dari fungsi yang dipanggil.

Untuk memanggil satu fungsi dari fungsi lain, dituliskan nama dari fungsi tersebut serta dicantumkan argumen jika diperlukan. Bentuk umumnya :

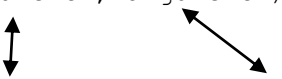
```
NamaFungsi ( Argumen ) ;
```

Hubungan antara argument dalam pemanggilan_fungsi dan parameter dalam definisi fungsi adalah *one-to-one*. Contoh seperti program berikut:

```
int argument1;
double argument2;

//Pemanggilan fungsi
hasil = namafungsi(argument1, argument2);

//Definisi fungsi
int namafungsi(int parameter1,double parameter2)
{
 //parameter1 = argument1 parameter2 = argument2
}
```



Dari contoh diatas bisa dilihat hubungan antara argument dengan parameter, tipe parameter1 sama dengan argument1, yaitu int dan tipe parameter2 sama dengan argument2 yaitu double. Jumlah argument yang memanggil sama dengan jumlah parameter dari fungsi yang dipanggil.

Jika sebuah variabel merupakan salah satu argumen dalam pemanggilan fungsi, fungsi yang dipanggil menerima salinan nilai yang disimpan dalam variabel. Misalnya, pernyataan `namafungsi(argument1,argument2);` memanggil fungsi `namafungsi()` dan menyebabkan nilai disimpan dalam variabel `argument1` dan `argument2` untuk diteruskan ke `namafungsi()`. Nama variabel dalam tanda kurung adalah argumen yang memberikan nilai ke fungsi yang dipanggil. Setelah nilai dilewatkan, kontrol ditransfer ke fungsi yang dipanggil.

Fungsi `namafungsi()` tidak menerima variabel bernama `argument1` dan `argument2` dan tidak memiliki mengetahui nama-nama variabel ini. Fungsi hanya menerima nilai dalam variabel ini dan kemudian harus menentukan di mana menyimpan nilai-nilai ini sebelum melakukan hal lain. Hal itu merupakan tindakan pengamanan untuk memastikan bahwa fungsi yang dipanggil tidak secara tidak sengaja mengubah data yang disimpan dalam suatu variabel. Fungsi hanya mendapatkan salinan data yang akan

digunakan. Salinan data dapat diubah sesuai kebutuhan. Begitu juga mengubah variabel-variabel yang dideklarasikan dalam fungsi.

Berikut contoh pemanggilan fungsi tanpa argument/void. Dalam contoh tersebut, fungsi `main()` memanggil fungsi yang bernama `message()` yang tidak memerlukan argument tertentu.

```
void main() {
      message();//pemanggilan fungsi message()
}
```

Contoh pemanggilan fungsi dengan pengembalian nilai dan argument:

```
void main() {
      double vol;

      vol = volume_kubus(12.5)
}
```

Penempatan Fungsi Dalam Program

Ada dua cara untuk menempatkan definisi fungsi dalam program:

1. Definisi fungsi diletakkan setelah fungsi yang memanggil.
   Untuk kondisi ini fungsi harus dideklarasikan terlebih dahulu sebelum dilakukan pemanggilan.

   Bentuknya sebagai berikut:

```
ReturnValue FungsiC();

main()
{
  // Memanggil fungsi FungsiC()
     FungsiC();
 }

ReturnValue FungsiC()
{
  Badan FungsiC
}
```

Contoh:

```
void LuasBujurSangkar();
void main()
{
   printf("Hitung Luas Bujur Sangkar");
   LuasBujurSangkar();
}

void LuasBujurSangkar ()
{
   int Sisi;
   printf("\n\nBerapa sisi Bujursangkar? ");
   scanf("%d", &Sisi);

   printf("\nLuas Bujursangkar adalah %d", Sisi
* Sisi");
}
```

2. Deklarasi dan definisi fungsi dilakukan sebelum dilakukan pemanggilan. Pemanggilan dapat dilakukan dimanapun setelah itu.

```
ReturnValue FungsiC()
{
        Badan FungsiC
}

main()
{
 / / Memanggil fungsi FungsiC()
        FungsiC();
}
```

Contoh:

```
void LuasBujurSangkar ()
{
      int Sisi;

      printf("\n\nBerapa sisi Bujursangkar? ");
      scanf("%d", &Sisi);
      printf("\nLuas Bujursangkar adalah %d", Sisi
* Sisi");
}

void main()
{
      printf("Hitung Luas Bujur Sangkar");
      LuasBujurSangkar();
}
```

Teknik Pengembalian Nilai

Karena C ++ tidak menyediakan setiap fungsi yang di perlukan dan perancang tidak mungkin mengetahui kebutuhan spesifik pengguna, maka harus dibuat sendiri fungsi sesuai dengan kebutuhan..

Fungsi yang ditentukan pengguna dalam C ++ diklasifikasikan ke dalam dua kategori:

1. Fungsi dengan return value
   Fungsi yang memiliki tipe pengembalian. Fungsi mengembalikan nilai yang mempunyai tipe data tertentu menggunakan pernyataan pengembalian.

2. Fungsi void
   Fungsi yang tidak memiliki tipe pengembalian. Fungsi ini tidak menggunakan pernyataan pengembalian untuk mengembalikan nilai.

void

Fungsi yang tidak mempunyai return value dideklarasikan dan diddefinisikan sebagai 'void'. Berikut contoh fungsi dengan 'void':

```
void Pengantar()
{
      Cout<<"Program ini digunakan untuk
      menghitung Luas"
}
```

Sebuah fungsi dapat bertipe void jika tidak diharapkan akan memberikan nilai tertentu. Tetapi fungsi void dapat diminta untuk mengerjakan tugas tertentu seperti contoh fungsi berikut ini yang mempunyai tugas menghitung luas bujursangkar dan menampilkan hasilnya.

```
void Luas Bujursangkar()
{
      int Sisi;

      printf("\nMasukkan sisi: ");
      scanf("%d", &Sisi);
      printf("\nLuas Bujursangkar adalah %d",
      Sisi * Sisi");
}
```

Ketika sebuah fungsi diberi tipe void maka tidak dapat memberikan hasilnya pada suatu variabel. Karena itu fungsi void hanya bisa dipanggil.

Return Value

Jika sebuah fungsi dideklarasikan untuk mengembalikan suatu nilai tertentu, maka kompiler harus mengetahui nilai apa yang akan dihasilkan fungsi. Nilai yang dihasilkan harus sesuai dengan tipe yang dideklarasikan.

Bentuk umum dari fungsi dengan pengembalian nilai tertentu adalah sebagai berikut:

```
ReturnValue FungsiC()
{
        Pernyataan-pernyataan;
        return Nilai;
 }

void main()
{
  // Memanggil fungsi FungsiC()
  variabel = FungsiC();
  }
```

Pada bentuk umum di atas, tipe dari Nilai, ReturnValue dan variabel yang menerima nilai fungsi harus sama. Jika fungsi dideklarasikan sebagai char, maka fungsi harus menghasilkan nilai bertipe karakter, dan variabel fungsi pemanggil yang menerima nilai fungsi juga harus bertipe karakter.

Nilai yang merupakan pengembalian nilai dalam suatu fungsi dapat berupa :

1. Konstanta
   Contoh:
```
int Luas()
{
  return 100;
}
```

2. Variabel
   Contoh:
```
double Luas Bujursangkar(double Sisi){
      double Luas;
      Luas = Sisi * Sisi;
      Return Luas;
}
```

3. Ekspresi
Contoh:

```
double Luas Bujursangkar(double Sisi)
{
      return Sisi * Sisi;
}
```

Teknik Pengiriman Argumen

Untuk melakukan pekerjaannya, terdapat fungsi yang memerlukan argumen. Fungsi yang ingin menggunakan hasil dari fungsi lain harus menyediakan argumen yang diperlukan oleh fungsi yang dipanggil. Ketika mendeklarasikan fungsi yang memerlukan argumen, masing-masing argumen dispesifikasi dengan tipe data dan nama argumen.

Pengiriman Argumen berdasarkan Nilai

Untuk menggunakan fungsi didalam fungsi lain maka harus dilakukan pemanggilan fungsi dengan memanggil nama fungsi dan memberikan argumen yang diperlukan. Hanya nama dari argumen yang diperlukan tanpa tipe datanya. Sehingga jika mendeklarasikan fungsi seperti berikut:

```
int jamkerja(char nama[20]);
```

maka pemanggilan fungsi tersebut dengan perintah sebagai berikut:

```
jamkerja(nama);
```

Contoh program sebagai berikut:

```
float jumlah(float, float);
main()
{
      float a, b,c;

      printf("Masukkan nilai a : ");
      scanf("%f", &a);

      printf("Masukkan nilai b : ");
      scanf("%f", &b);
```

```
	c = jumlah(a, b);

	printf("\nHasil penjumlahan a + b = %g\n",
c);
}

float jumlah(float x, float y)/* definisi fungsi */
{
	return(x + y);
}
```

Ketika mendeklarasikan fungsi, kompiler tidak memerlukan nama dari masing-masing argumen, hanya memerlukan tipe dari argumen tersebut dan berapa argumen yang diperlukan oleh suatu fungsi.

Pengiriman Argumen dengan Referensi

Ketika mendeklarasikan variabel, kompiler menyediakan sejumlah tempat untuk variabel tersebut. Jika perlu menggunakan variabel di program, maka variabel tersebut cukup dipanggil dan digunakan nilainya. Ada dua masalah utama berhubungan dengan variabel, nilai dan lokasinya dalam memori. Lokasi dalam memori disebut alamat.

Jika argumen yang diperlukan hanya dicantumkan namanya, kompiler hanya menyalin nilai argumen dan memberikan ke fungsi yang dipanggil. Meskipun fungsi yang dipanggil menerima nilai argumen dan dapat menggunakan dalam berbagai cara, hal ini tidak dapat mengubah nilai argumen dari fungsi pemanggil. C mengijinkan fungsi yang dipanggil untuk memodifikasi nilai argumen yang dikirimkan jika diperlukan. Jika diperlukan fungsi yang dipanggil untuk merubah nilai dari argumennn yang dikirimkan dan memberikan hasil nilai yang sudah dirubah maka argumen harus dikirimkan menggunakan referensinya.

Untuk mengirimkan argumen sebagai referensi, ketika mendeklarasikan fungsi, nama argumen diawali dengan ´&´. Argumen yang dikirimkan sebagai referensi biasa 0, satu, atau lebih atau bisa juga semua argumen.

Berikut adalah contoh dari pengiriman argumen berdasarkan referensi:

1. argumen dikirim berdasar referensi
   ```
   void luas(double &sisi);
   ```
2. satu argumen yang dikirim berdasarkan referensi
   ```
   bool Pilihan(char &jawab, int umur);
   ```
3. Semua argumen berupa referensi
   ```
   float Penjualan(float &hargadiscount, float &discount, char &komisi);
   ```

Berikut adalah contoh program untuk menukar nilai dengan argumen berdasar referensi::

```
void tukar (&int, &int);
main(){
      int a,b;
      a = 88;
      b = 77;
      printf("Nilai sebelum pemanggilan fungsi\n");
      printf("a = %d    b = %d\n", a, b);
      tukar(a,b);
      printf("\nNilai      setelah      pemanggilan
fungsi\n");
      printf("a = %d    b = %d\n", a, b);
}

void tukar(int &x, int &y)
{
      int z;
      z = x;
      x = y;
      y = z;
      printf("\nNilai di akhir fungsi tukar()\n");
      printf("x = %d    y = %d\n", x, y);
}
```

Rangkuman

Fungsi merupakan sebuah subprogram yang dapat memanipulasi data atau parameter dan mengembalikan nilai. Setiap program C mempunyai sedikitnya satu fungsi, main(). Ketika suatu program mulai bekerja, main() secara otomatis dipanggil. Fungsi main() dapat memanggil fungsi lain. Setiap fungsi mempunyai nama yang berbeda dengan fungsi lainnya

Latihan

1. Cobalah program untuk memanggil fungsi berulang-ulang berikut:

```
void cetak_pesan(void);

main(){
   int i;

   for(i=1; i<=5; i++)
         cetak_pesan();
   printf("\n");
}

void cetak_pesan()
{  printf("Cetak pesan ini\n");}
```

   Rubahlah program diatas dengan banyaknya perulangan berdasar input.

2. Perbaikilah program berikut untuk mengkonversi besaran sudut dari derajat ke radian.

```
#define PI 3.14159f
float radian(float);

main()
{
   float derajat;
```

```
    cout<<"Masukkan besar sudut dlm derajat";
    cin >> derajat;
    cout<<derajat<<"derajat"<<'='
          <<radian(derajat);
}

float radian(float x){
    float hasil;

    hasil = x / 180.0f * PI;
}
```

3. Berdasarkan fungsi berikut, apakah yang dihasilkan dari perintah pemanggilan fungsi dibawah.

```
int secret(int one){
          int i;
          int prod = 1;
          for (i = 1; i <= 3; i++)
          prod = prod * one;
    return prod;
}

i. cout << secret(5) << endl;
ii. cout << 2 * secret(6) << endl;
```

4. Tulis definisi fungsi yang mempunyai parameter tiga bilangan dan mengembalikan jumlah dari dua angka pertama dikalikan dengan angka ketiga. (Asumsikan bahwa ketiga angka tersebut bertipe double.)

5. Pada sebuah main program terdapat deklarasi variabel seperti berikut :

```
    double num1, num2, num3;
    int int1, int2, int3;
    int value;
    num1 = 5.0; num2 = 6.0; num3 = 3.0;
    int1 = 4; int2 = 7; int3 = 8;
```

Pada program terdapat prototype fungsi seperti berikut :

```
double cube(double a, double b, double c);
```

Dari beberapa pernyataan dibawah ini, manakah pernyataan yang valid ?

```
a. value = cube (num1, 15.0, num3);
b. cout << cube(num1, num3, num2) << endl;
c. cout << cube(6.0, 8.0, 10.5) << endl;
d. cout << cube(num1, num3) << endl;
e. value = cube(num1, int2, num3);
f. value = cube(7, 8, 9);
g. value = cube(int1, int2, int3);
```

Latihan Program

1. Buatlah fungsi untuk menghitung jumlah dan rata-rata dari tiga bilangan riil.

2. Tulislah sebuah program untuk menghitung jumlah luas dua bujursangkar. Rumus bujursangkar terletak didalam fungsi.

3. Buatlah suatu fungsi ganjil() yang mengembalikan nilai 1 jika argumen yang diberikan adalah bilangan ganjil dan mengembalikan nilai 0 jika argumen tsb bukan bilangan ganjil.

4. Buatlah sebuah fungsi untuk memeriksa apakah sebuah bilangan integer yang dimasukkan merupakan bilangan prima atau tidak.

5. Tulislah sebuah program yang menggunakan fungsi dimana di dalam fungsi terdapat operasi modulus. Hasil pembagian modulus dikembalikan ke fungsi utama.

Daftar Pustaka

Bronson, Gary. 2012. *A First Book of C++, fourth edition*. USA: Cengage Learning.

D.S. Malik. 2013. C++ *Programming: From Problem Analysis To Program Design, Fifth Edition*. USA: Cengage Learning.

J Malini Devi. 2014. *C++ Programming Language for Beginners with Easy tips*.

Kadir, Abdul. 2013. *Pengenalan Algoritma*. Yogyakarta: Penerbit Andi.

Keyser, John.2019. *Introduction to C++ Programming Concepts and Applications*. United States of America: The Great Courses.

Ladd, Scot Robert. 1990. *Turbo C++ Techniques and Applicatio*. M&T Publishing,.

Paul Barry and David Griffiths . 2009. *Head First Programming*. United States of America: O'Reilly Media.

Schildt, Herbert. 1995. *C Complete Refernce, third edition*. Mc Graw Hill.

Stroustrup, Bjarne. 2013. *The C++ programming language, Fourth edition*. United States of America: Pearson Education, Inc.