MODUL PRAKTIKUM



ALGORITMA DAN PEMROGRAMAN

UNIVERSITAS TEKNOLOGI MATARAM

2018

Jl. Kampus Universitas Teknologi Mataram, Kekalik, Mataram

BAB 1
BAHASA DAN ALGORITMA PEMROGRAMAN

1.1. Bahasa pemrograman

Bahasa atau dalam bahasa inggris *language* adalah suatu sistim untuk berkomunikasi. Bahasa tertulis menggunakan simbol (yaitu huruf) untuk membentuk kata. Dalam ilmu komputer,bahasa manusia disebut bahasa alamiah (natural languages), dimana komputer tidak bisa memahaminya, sehingga diperlukan suatu bahasa komputer. Komputer mengerjakan transformasi data berdasarkan kumpulan perintah - program - yang telah dibuat oleh pemrogram. Kumpulan perintah ini harus dimengerti oleh komputer, berstruktur tertentu (syntax) dan bermakna. Bahasa pemrograman merupakan notasi untuk memberikan secara tepat program komputer. Berbeda dengan bahasa alamiah, mis. Bahasa Indonesia, Inggris dsb. yang merupakan Bahasa alamiah(natural language), sintaks dan semantik bahasa pemrograman (komputer) ditentukan secara kaku, sehingga bahasa pemrograman juga disebut sebagai bahasa formal (formal language). Jadi, dalam bahasa pemrograman yang digunakan sebagai alat komunikasi untuk memberikan perintah kepada komputer tidak berlaku kebebasan berekspresi seperti laiknya dalam bahasa alamiah. Pemrograman dalam pengertian luas meliputi seluruh kegiatan yang tercakup dalam pembuatan program, termasuk analisis kebutuhan (requirement's analysis) dan keseluruhan tahapan dalam perencanaan (planning) perancangan (design) dan pewujudannya (implementation). Dalam pengertian yang lebih sempit, pemrograman merupakan pengkodean (coding atau program writing = penulisan program) dan pengujiannya (testing) berdasarkan rancangan tertentu. Pemahaman yang lebih sempit ini sering digunakan dalam pembuatan program-program terapan komersial yang membedakan antara system analyst yang bertanggung jawab dalam menganalisa kebutuhan, perencanaan dan perancangan program dengan pemrogram (programmer) yang bertugas membuat kode program dan menguji kebenaran program.

Generasi bahasa pemrograman:

- Generasi I: machine language
- Generasi II: assembly language : Asssembler
- Generasi III: high-level programming language: C, PASCAL, dan sebagainya.
- Generasi IV: 4 GL (fourthgeneration language): SQL

1.1.1. Bahasa Tingkat Rendah

Merupakan bahasa assembly atau bahasa mesin. Bahasa ini lebih dekat ke mesin (*hardware*), dimana pada high-level programming languages, suatu bahasa yang sudah dekat pada bahasa manusia. Pada bahasa ini akan lebih cepat dipahami oleh programmer karena sifatnya yang dekat dengan bahasa manusia

1.1.2. Bahasa Mesin

Bahasa mesin merupakan representasi tertulis machine code (kode mesin), yaitu kode operasi suatu mesin tertentu. Bahasa ini bersifat khusus untuk mesin tertentu dan "dimengerti" langsung oleh mesin, sehingga pelaksanaan proses sangat cepat. Bahasa mesin kelompok komputer tertentu berlainan dengan bahasa mesin kelompok komputer yang lain. Abstraksi bahasa ini adalah kumpulan kombinasi kode biner "0" dan "1" yang sangat tidak alamiah bagi kebanyakan orang – kecuali insinyur pembuat mesin komputer. Karena tidak alamiah bagi kebanyakan orang, bahasa mesin juga disebut bahasa tingkat rendah.

1.1.3. Bahasa Assembly

Bahasa rakitan (assembly language) merupakan notasi untuk menyajikan bahasa mesin yang lebih mudah dibaca dan dipahami oleh manusia. Bahasa ini sudah menggunakan

simbol alpabet yang bermakna (mnemonic). Contoh “MOV AX 1111”, pindahkan ke register AX nilai 1111.Proses data oleh computer berdasarkan perintah bahasa rakitan adalah cepat. Meski demikian masih merepotkan-bahkan bagi kebanyakan pemrogram, karena masih harus mengingat-ingat tempat penyimpanan data. Bahasa rakitan juga bersifat khusus untuk mesin tertentu. Contoh: Assembler

2.1.4. Bahasa Tingkat Tinggi

Adalah bahasa pemrograman yang dekat dengan bahasa manusia, kelebihan utama dari bahasa ini adalah mudah untuk di baca, tulis, maupun diperbaharui, sebelum bisa dijalankan program harus terlebih dahulu di-compile. Contoh Ada, Algol, BASIC, COBOL, C, C++, FORTRAN, LISP, dan Pascal, dsb. Pada generasi bahasa pemrograman terakhir sekarang ini, kedua cara interpretasi dan kompilasi digabungkan dalam satu lingkungan pengembangan terpadu (IDE=integrated development environ-ment). Cara interpretasi memudahkan dalam pembuatan program secara interaktif dan cara kompilasi menjadikan eksekusi program lebih cepat. Pertama program dikembangkan interaktif, kemudian setelah tidak ada kesalahan keseluruhan program dikompilasi. Contoh bahasa program seperti ini adalah Visual BASIC yang berbasis BASIC dan Delphi yang berbasis PASCAL. Bahasa tingkat tinggi bersifat portable. Program yang dibuat menggunakan bahasa tingkat tinggi pada suatu mesin computer bersistem operasi tertentu, hamper 100% bisa digunakan pada berbagai mesin dengan aneka sistem operasi. Kalaupun ada perbaikan sifatnya kecil sekali.

2.1.5. Bahasa Generasi 3 & 4

Lebih dekat ke bahasa manusia dibandingkan dengan high-level programming languages. Biasanya dipakai untuk mengakses database. Contoh perintah pada bahasa SQL: FIND ALL RECORDS WHERE NAME IS "JOHN".

2.1.6. Bahasa Pemrograman untuk tujuan tertentu

Dibawah ini merupakan Tabel Bahasa Pemrograman untuk tujuan tertentu. (Mc. Connell, h 46)

Tabel 2.1. Bahasa Pemrograman

Jenis Program	Bahasa Terbaik	Bahasa terburuk
Data terstruktur	ADA, C /C++, PASCAL	Assembler, BASIC
Proyek cepat	BASIC	PASCAL, ADA, Assembler
Eksekusi cepat	Assembler, C	BASIC,Intrepreter Language
Kalkulasi matematika	FORTRAN	PASCAL
Menggunakan memori dinamis	PASCAL, C	BASIC
Lingkungan memori terbatas	BASIC, Assembler, C	FORTRAN
Program real-time	ADA, Assembler, C	BASIC, FORTRAN
Manipulasi string	BASIC, PASCAL	C
Program mudah dikelola	PASCAL, ADA	C, FORTRAN

Tabel Rasio pernyataan bahasa tingkat tinggi dengan kode bahasa rakitan yang setara. (Mc. Connell, h 46)

Tabel 2.2. Rasio bahasa tingkat tinggi dengan kode bahasa rakitan

BAHASA	RASIO
Assembler	1: 1
ADA	1 : 4.5
Quick / Turbo / Basic	1 : 5
C	1 : 2.5
FORTRAN	1 : 3
PASCAL	1 : 3.5

2.2. Compiler dan Intepreter

Compiler Adalah suatu program yang menterjemahkan Bahasa program (source code) ke dalam bahasa objek (object code). Compiler menggabungkan keseluruhan Bahasa program dikumpulkan kemudian disusun kembali.

2.2.1. Tahapan Kompilasi:

Dalam melakukan penulisan bahasa pemrograman komputer ada beberapa tahapan-tahapan yang harus dilakukan antara lain: Pertama source code (program yang ditulis) dibaca ke memory komputer.

- Source code tersebut diubah menjadi object code (Bahasa assembly)
- Object code dihubungkan dengan library yang dibutuhkan untuk membentuk file yang bisa di eksekusi.

Compiler memerlukan waktu untuk membuat suatu program yang dapat dieksekusi oleh komputer. Tetapi, program yang diproduksi oleh Compiler bisa berjalan lebih cepat dibandingkan dengan yang diproduksi oleh Interpreter, dan bersifat independen.

2.2.2.Interpreter

Berbeda dengan compiler, Interpreter menganalisis dan mengeksekusi setiap baris dari program tanpa melihat program secara keseluruhan. Keuntungan dari Interpreter adalah dalam eksekusi yang bisa dilakukan dengan segera. Tanpa melalui tahap kompilasi, untuk alasan ini interpreter digunakan pada saat pembuatan program berskala besar.

2.3. Tipe Pemrograman

Dalam Bahasa pemrograman komputer terdapat tipe pemrograman. Tipe pemrograman yang pertama adalah pemrograman terstruktur dan prosedural.

2.3.1. Pemrograman Terstruktur

Pemrograman terstruktur adalah cara pemrosesan data yang terstuktur. Terstruktur dalam: analisa, cara dan penulisan program. Prinsip pemrograman terstruktur adalah mempunyai ciri sebagai berikut:

- Menggunakan rancangan pendekatan dari atas ke bawah (top down design),
- Membagi program ke dalam modul-modul logika yang sejenis,
- Menggunakan sub-program untuk proses-proses sejenis yang sering digunakan,
- Menggunakan pengkodean terstruktur: IF ..... THEN, DO ..... WHILE dan lain-lainnya,
- Menghindari penggunaan perintah GO TO bila tidak diperlukan,
- Gunakan nama-nama bermakna (mnemonic names), dan
- Membuat dokumentasi yang akurat dan berarti.

Dalam perencanaan dan perancangan dari atas ke bawah, kategori dan penyelesaian masalah dimulai dari bagian yang utama kemudian dibagi menjadi bagian yang lebih kecil. Rancangan cara ini memudahkan penulisan, pengujian, koreksi dan dokumentasi program.

Tahapan rancangan atas ke bawah dalam pemrograman:

- Tentukan keluaran (output) yang diminta, masukan (input) yang diperlukan dan proses-proses utama yang diperlukan untuk transformasi data.
- Membagi proses utama ke dalam modul-modul fungsional.
- Buat algoritma msing-masing modul, dari modul utama ke subsub modul.

Setiap modul dalam proses rancangan atas ke bawah biasanya dibatasi dalam isi maupun batasanbatasan berikut:

- Setiap modul hanya mempunyai satu masukan dan keluaran
- Setiap modul hanya mewakili satu fungsi program.

Rancangan (design) terstruktur:

- Membagi program menjadi subprogram
- Menekankan fungsionalitas.
- Cocok untuk sistem yang banyak mempunyai fungsi independen.

Gaya penulisan program terstruktur:
Menggunakan indentasi sehingga jelas struktur dan kontrol program. Memudahkan pembacaan, pemahaman, penelusuran kesalahan dan pembuatan koreksi.

2.3.2. Pemrograman procedural dan terstruktur

Bahasa pemrograman prosedural adalah Bahasa pemrograman yang mendukung pembuatan program sebagai kumpulan prosedur. Prosedurprosedur ini dapat saling memanggil dan dipanggil dari manapun dalam program dan dapat menggunakan parameter yang berbeda-beda untuk setiap pemanggilan. Prosedur adalah bagian dari program untuk melakukan operasi-operasi yang sudah ditentukan dengan menggunakan parameter tertentu. Bahasa pemrograman terstruktur adalah pemrograman yang mendukung abstraksi data, pengkodean terstruktur dan control program terstruktur. Kontrol program terstruktur adalah sebagai berikut:

1. Runtun - urut (sequence)
2. Pilihan (selection)
3. Pengulangan (repetition - loop)
4. Batasan Masalah Merencanakan sistim dan spesifikasi program: Siapa yang akan menggunakan program dan untuk apa? Dengan cara:
   - Menentukan tujuan dan hasil yang akan dicapai
   - Menentukan hal-hal yang diperlukan oleh sistim
   - Pengumpulan data
5. Pengembangan Model Pembuatan model dari sistim yang akan kita bangun, model adalah suatu gambaran sederhana dari sistimyang kita buat. Dengan pembuatan model akan terlihat dengan jelas hubungan antara objek-objek dalam sistim yang akan kita bangun. Untuk penyelesaian aritmatik, biasanya model dibuat dalam bentuk rumus matematik. Contoh: untuk membuat program luas_lingkaran kita membuat model matematis c = axb
6. Rancangan algoritma Pembuatan urutan instruksi yang akan ditulis pada program (dijelaskan lebih lanjut)
7. Pemrograman Implementasi algoritma ke dalam program (algoritma sendiri dalam computer adalah merupakan program).
8. Uji dan Validasi Pengujian terhadap program : seperti kesalahan penulisan (syntax error), kesalahan saat eksekusi (runtime error) kesalahan logika program menghasilkan output yang salahfatal error).

Dokumentasi Pembuatan catatan pada program terutama pada modulmodul yang rumit.

2.4. Algoritma

Pemrograman komputer dan algoritma pemrograman adalah dua hal yang tidak dapat dipisahkan karena pembuatan program komputer akan lebih sulit dan lama tanpa mengetahui dengan pasti bagaimana algoritma penyelesaian masalahnya. Sebelum mengetahui lebih lanjut apa yang dimaksud dengan algoritma pemrograman, kita bahas dahulu apa yang dimaksud dengan pemrograman komputer atau program komputer. Definisi program computer adalah sederetan perintah-perintah (instruksi) yang harus dikerjakan oleh komputer untuk menyelesaikan masalah. Deretan perintah-perintah tersebut tidak bisa kita tulis secara sembarangan atau semau kita tetapi harus teratur agar komputer dapat memahami dan memprosesnya dengan baik sehingga permasalahan yang ada dapat diselesaikan dengan baik pula. Untuk itulah diperlukan algoritma karena definisi dari algoritma itu sendiri menurut Microsoft Press Computer and Internet Dictionary (1998) adalah urutan langkah logis tertentu untuk memecahkan suatu masalah. Yang ditekankan adalah urutan langkah logis, yang berarti algoritma harus mengikuti suatu urutan tertentu, tidak boleh melompat-lompat dan disusun secara sistematis. Sedangkan yang dimaksud dengan langkah-langkah logis adalah kita harus dapat mengetahui dengan pasti setiap langkah yang kita buat. Menurut Sjukani (2005), algoritma adalah alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Yang ditekankan pertama adalah alur pikiran, sehingga algoritma seseorang dapat berbeda dari algoritma orang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya dapat berupa kalimat, gambar, atau tabel tertentu. Jadi dapat disimpulkan bahwa algoritma lebih merupakan alur pemikiran untuk menyelesaikan suatu pekerjaan atau suatu masalah daripada pembuatan program komputer. Algoritma inilah yang kemudian dijadikan landasan (pedoman) untuk membuat program komputer. Meskipun algoritma tidak dapat dipisahkan dengan pemrograman komputer tetapi jika anda beranggapan bahwa algoritma identic dengan pemrograman komputer, anda salah besar. Hal ini dikarenakan dalam kehidupan sehari-haripun seringkali kita berhadapan dengan masalah-masalah yang kalau kita cermati mengikuti kaidah-kaidah penyelesaian secara algoritma. Misalkan saja cara-cara memasak mie instan, membuat kopi atau teh, memasak makanan yang dinyatakan dalam bentuk resep, dan masih banyak lagi yang semuanya itu dapat kita sebut sebagai algoritma. Pada mie instan misalnya, biasanya pada bungkusnya terdapat urutan langkahlangkah bagaimana cara memasak atau menyajikannya. Bila langkahlangkah tersebut tidak logis, maka dapat dipastikan bahwa kita akan memperoleh hasil yang tidak sesuai dengan yang diharapkan. Kita harus membaca satu demi satu langkahlangkah pembuatannya kemudian mengikutinya agar memperoleh hasil yang baik. Yang harus diingat disini adalah kita tidak harus mengikuti langkahlangkah yang sudah diberikan, tetapi kita dapat memodifikasinya atau bahkan membuat resep atau cara baru yang lebih baik tetapi menghasilkan hal yang sama (mempunyai tujuan yang sama), yaitu dapat menikmati hasil masakan. Demikian juga dengan pemrograman komputer, kita juga tidak harus mengikuti algoritma yang sudah ada, tetapi kita dapat menambah ataupun mengurangi bahkan membuat algoritma yang baru asalkan permasalahan yang ada dapat terpecahkan dengan baik.

2.5. Ciri Algoritma

Oleh karena algoritma digunakan untuk memecahkan suatu permasalahan maka algoritma tersebut harus menghasilkan suatu jawaban atas permasalahan tersebut. Dengan kata lain algoritma harus memiliki paling tidak satu keluaran. Sedangkan masukan dari algoritma dapat nol (tidak ada) atau banyak masukan (data). Yang dimakdud dengan nol masukan adalah jika algoritma itu hanya untuk menampilkan suatu informasi saja. Misalnya output "Hello World" yang sering kita temukan pada tutorial tutorial saat kita baru belajar membuat program dari suatu Bahasa pemrograman tertentu. Kedua hal diatas, memiliki paling sedikit satu keluaran dan dapat memiliki nol atau banyak masukan, merupakan dua dari beberapa ciri algoritma. Tugas algoritma dikatakan selesai kalau algoritma tersebut sudah menghasilkan satu atau lebih jawaban atas permasalahan yang ada. Dengan demikian setelah mengerjakan langkah-langkah penyelesaian masalah, maka algoritma tersebut harus berhenti tidak melakukan proses apapun. Berhenti di sini artinya adalah jika diterjemahkan ke dalam bentuk program dan program dijalankan, maka setelah menghasilkan suatu output, program dapat langsung berhenti atau menunggu instruksi lebih lanjut dari pengguna program seperti mengulang perhitungan lagi, keluar program (menghentikan program), dan lain sebagainya. Dengan demikian ciri ketiga dari algoritma adalah setelah selesai mengerjakan langkah-langkah penyelesaian masalah, algoritma harus berhenti. Ciri keempat dari algoritma adalah setiap langkah yang dibuat harus dibuat sesederhana mungkin tetapi efektif agar dapat dipahami oleh pemroses (manusia maupun komputer) sehingga dapat diselesaikan dalam waktu yang singkat serta masuk akal. Sedangkan ciri yang terakhir adalah setiap langkah dalam algoritma harus didefinisikan dengan tepat dan jelas sehingga tidak berarti-dua (ambiguitas) sehingga menimbulkan kesalahan dalam penafsiran oleh pemroses

2.6. Penerapan Algoritma

Berikut adalah contoh pemecahan masalah (algoritma) yang diambil dari permasalahan yang mungkin sering kita hadapi dalam kehidupan kita sehari-hari. Diketahui dua buah ember A dan B dimana ember A berisi air dan ember B berisi minyak tanah. Jika diinginkan isi kedua ember itu saling ditukar sehingga ember A berisi minyak tanah dan ember B berisi air, bagaimanakah caranya? Apakah cukup dengan cara (membuat algoritma), tuangkan isi ember A ke ember B dan kemudian tuangkan isi ember B ke ember A? Apakah permasalahan dapat diselesaikan dengan cara (algoritma) tersebut? Jawabannya adalah tidak, karena algoritma tersebut tidak logis dan hasilnyapun tidak sesuai dengan yang diinginkan karena algoritma tersebut akan menghasilkan ember A akan berisi campuran air dan minyak tanah sedangkan ember B akan kosong. Bagaimanakah cara (algoritma) yang benar dari permasalahan tersebut? Algoritma yang benar adalah pindahkan dahulu isi ember A ke ember lain (misal ember C), kemudian setelah ember A kosong pindah isi ember B ke ember A. Langkah terakhir adalah mengisi ember B dengan minyak tanah yang ada di ember C. Inilah algoritma yang paling logis dan menghasilkan jawaban yang benar atas permasalahan tersebut. Seperti yang telah dikemukakan sebelumnya, setiap orang mempunyai cara pemecahan sendiri sendiri sehingga setiap orang dimungkinkan mempunyai algoritma yang berbeda-beda untuk memecahkan suatu permasalahan yang sama. Contohnya adalah permasalahan di atas. Langkah pertama bisa saja yang dipindah bukan isi ember A ke ember C dulu, tetapi isi ember B yang dipindahkan ke dalam ember C terlebih dahulu. Setelah itu baru memindahkan isi ember A ke dalam ember B dan terakhir memindah isi ember C ke

ember A. Algoritma ini sedikit berbeda tetapi tetap menghasilkan jawaban yang sama atas persoalan yang ada.

2.7. Notasi Algoritma

Algoritma mempunyai aturan penulisan sendiri yang disebut dengan notasi algoritma. Notasi algoritma ini tidak tergantung dari spesifikasi bahasa pemrograman tertentu dan komputer yang mengeksekusinya. Hal ini dikarenakan notasi algoritma bukanlah notasi Bahasa pemrograman. Notasi algoritma merupakan bahasa universal yang dapat diterima oleh semua Bahasa pemrograman yang ada. Oleh sebab itu algoritma yang baik harus dapat diterjemahkan ke dalam bentuk source code dari semua Bahasa pemrograman yang ada. Untuk membuat algoritma dari suatu permasalahan, biasanya digunakan salah satu dari tiga buah notasi algoritma yang dikenal, yaitu uraian kalimat deskriptif, flow chart, atau pseudo code. Sebagai contoh permasalahan, jika diinginkan sebuah program komputer yang dapat mengetahui bilangan terbesar dari tiga buah bilangan yang dimasukkan. Bagaimanakah algoritmanya?

2.8. Deskriptif Algoritma

Algoritma dengan uraian kalimat deskriptif adalah notasi algoritma yang paling sederhana karena algoritma ini menggunakan Bahasa sehari-hari. Untuk permasalahan yang sederhana penggunaan notasi ini sangatlah mudah, akan tetapi untuk permasalahan yang lebih komplek dan rumit, penggunaan notasi ini akan lebih sulit dan sering kali terjadi ambigu dalam langlahlangkah penyelesaian masalah. Oleh karena itulah untuk kasus-kasus yang lebih komplek, penggunaan notasi ini jarang sekali bahkan tidak digunakan.

Permasalahan di atas, yaitu mencari bilangan terbesar dari tiga buah bilangan yang dimasukkan, tergolong permasalahan yang sederhana, jadi algoritmanya masih mudah dan dapat dijelaskan dengan uraian kalimat deskriptif sebagai berikut:

1. Masukkan sembarang bilangan sebanyak tiga buah.
2. Ambil bilangan pertama dan set maksimum - nya sama dengan bilangan pertama.
3. Ambil bilangan kedua dan bandingkan dengan maksimum.
4. Apabila bilangan kedua lebih besar dari maksimum maka ubah maksimum - nya menjadi sama dengan bilangan kedua.
5. Ambil bilangan ketiga dan bandingan dengan maksimum.
6. Apabila bilangan ketiga lebih besar dari maksimum maka ubah lagi maksimumnya menjadi sama dengan bilangan ketiga.
7. Variabel maksimum akan berisi bilangan yang terbesar dan tampilkan hasilnya.

Algoritma dengan uraian kalimat deskriptif seperti di atas sudah jarang sekali kita temukan karena kadang kala agak sulit untuk memahaminya. Yang paling banyak kita temukan adalah algoritma (dengan uraian kalimat deskriptif) yang ditulis secara lebih sistematis dan efisien sehingga lebih mudah untuk memahaminya. Algoritma tersebut adalah sebagai berikut:

1. Masukkan a, b, dan c.
2. mak ← a.
3. Jika b > mak, kerjakan langkah ke-4. Jika tidak, kerjakan langkah ke-5.
4. mak ← b.
5. Jika c > mak, kerjakan langkah ke-6. Jika tidak, kerjakan langkah ke-7.
6. mak ← c.
7. Tulis mak.

Dalam notasi algoritma, baik menggunakan notasi algoritma dengan uraian kalimat deskriptif, flowchart maupun pseudo code, kita tidak menggunakan tanda = (sama

dengan) tetapi menggunakan symbol anak panah ke arah kiri (□) seperti yang terlihat pada langkah ke-2, 4, dan 6. Sebagai contoh pada langkah ke-2, arti dari notasi tersebut adalah nilai variabel a (yang ada di sebelah kanan anak panah) diberikan kepada variabel mak (yang ada di sebelah kiri anak panah). Dengan demikian jika nilai variabel a adalah 10, maka nilai mak juga 10 atau dalam penulisan secara matematis mak = a. Penggunaan anak panah ini dikarenakan, seperti yang telah dikemukakan sebelumnya, algoritma tidak diperuntukkan untuk suatu bahasa pemrograman tertentu, tetapi dapat diaplikasikan atau diterjemahkan ke dalam bentuk source code dari semua Bahasa pemrograman yang ada. Dalam pascal misalnya, notasi yang digunakan untuk tanda = (sama dengan) adalah := (titik dua dilanjutkan dengan tanda sama dengan) sehingga langkah ke-2 akan diterjemahkan menjadi mak := a. Akan tetapi dalam bahasa C++ maupun Java, tanda = (sama dengan) masih tetap digunakan sehingga penerjemahannya adalah mak = a

2.9. Flow Chart

Notasi algoritma yang paling banyak digunakan adalah flow chart karena bentuknya yang sederhana dan mudah dipahami. Flow chart (diagram alir) adalah penggambaran secara grafik dari langkah-langkah pemecahan masalah yang harus diikuti oleh pemroses. Flow chart terdiri atas sekumpulan simbol, dimana masing-masing symbol menggambarkan suatu kegiatan tertentu. Flow chart diawali dengan penerimaan masukan (input), pemrosesan masukan, dan diakhiri dengan menampilkan hasilnya (output).

Adapun simbol-simbol yang sering digunakan untuk menyusun flow chart (dalam microsoft visio) adalah sebagai berikut :

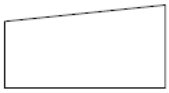
1. Masukan Masukan merupakan kegiatan penerimaan data yang disimbolkan dengan jajaran genjang. Kita dapat menuliskan masukan yang diperlukan pada suatu waktu secara satu per satu maupun secara keseluruhan, akan tetapi untuk alasan efisiensi ruang gambar biasanya masukan dituliskan bersamaan secara keseluruhan.



Gambar 2.1. Simbol masukan

2) Masukan manual

Untuk masukan secara manual yang dimasukkan melalui keyboard, atau perangkat input lainnya seperti barcode reader, kita dapat menggunakan symbol masukan secara manual. Sama dengan simbol masukan, pada simbol masukan manual ini untuk alasan efisiensi ruang gambar biasanya masukan juga dituliskan bersamaan secara keseluruhan.



Gambar 2.2. Simbol masukan manual

3) Proses

Data yang dimasukan kemudian diproses untuk menghasilkan jawaban atas persoalan yang ingin dipecahkan. Kegiatan memproses data ini disimbolkan dengan persegi panjang. Sama seperti simbol pada masukan, penulisan operasi-operasi pada data dapat dilakukan secara satu per satu maupun secara keseluruhan.



Gambar 2.3. Simbol proses

4) Keluaran

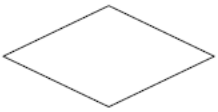
Keluaran adalah hasil dari pemrosesan data dan merupakan jawaban atas permasalahan yang ada. Keluaran ini harus ditampilkan pada layar monitor agar dapat dibaca oleh pengguna program. Sama seperti aturan pada simbol-simbol sebelumnya, penulisan hasil pemrosesan data dapat dilakukan secara satu persatu maupun secara keseluruhan.



Gambar 2.4. Simbol keluaran

5) Percabangan

Yang dimaksud dengan percabangan disini adalah suatu kegiatan untuk mengecek atau memeriksa suatu keadaan apakah memenuhi suatu kondisi tertentu atau tidak. Jadi dalam percabangan ini, kita harus menuliskan kondisi apa yang harus dipenuhi oleh suatu keadaan. Hasil dari pemeriksaan keadaan ini adalah YA atau TIDAK. Jika pemeriksaan keadaan menghasilkan kondisi yang benar, maka jalur yang dipilih adalah jalur yang berlabel YA, sedangkan jika pemeriksaan keadaan menghasilkan kondisi yang salah, maka jalur yang dipilih adalah jalur yang berlabel TIDAK. Berbeda dengan aturan pada simbol-simbol sebelumnya, penulisan kondisi harus dilakukan secara satu per satu (satu notasi percabangan untuk satu kondisi).



Gambar 2.5. Simbol percabangan

6) Sub rutin

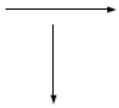
Sub rutin adalah suatu bagian dalam program yang dapat melakukan (atau diberi) tugas tertentu. Jadi sub rutin merupakan “program kecil” yang menjadi bagian dari suatu program yang besar. Sub rutin ada dua macam, yaitu prosedur (procedure) dan fungsi (function). Perbedaan antara keduanya adalah setelah dipanggil prosedur tidak mengembalikan suatu nilai sedangkan fungsi selalu mengembalikan suatu nilai. Dalam bahasa C++ kedua sub rutin tersebut dijadikan satu yaitu function, sedangkan untuk Java menggunakan class dimana keduanya bisa diatur untuk dapat mengembalikan nilai atau tidak dapat mengembalikan nilai. Sub rutin ini akan didiskusikan pada bab tersendiri. Sub rutin memiliki suatu flow chart yang berdiri sendiri diluar flow chart utama. Jadi dalam simbol sub rutin, kita cukup menuliskan nama sub rutinnya saja, sama seperti jika kita melakukan pemanggilan terhadap suatu sub rutin pada program utama (main program) yang akan anda pelajari pada bagian atau bab lain pada buku ini. Aturan penulisan simbol sub rutin sama dengan simbol percabangan, yaitu penulisan nama sub rutin dilakukan secara satu per satu.



Gambar 2.6. Simbol sub rutin

7) Arah aliran

Arah aliran merupakan jalur yang harus diikuti dan merupakan garis penghubung yang menghubungkan setiap langkah pemecahan masalah yang ada dalam flow chart. Arah aliran ini disimbolkan dengan anak panah



Gambar 2.7. Simbol arah aliran

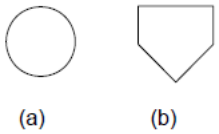
8) Terminator

Terminator berfungsi untuk menandai titik awal dan titik akhir dari suatu flow chart. Simbol terminator ini diberi label MULAI untuk menandai titik awal dari flow chart dan label SELESAI untuk menandai titik akhir dari flow chart. Jadi dalam sebuah flow chart harus ada dua symbol terminator, yaitu symbol terminator untuk MULAI dan SELESAI.

Gambar 2.8. Simbol terminator

9) Konektor

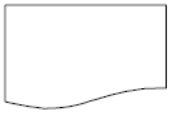
Konektor berfungsi untuk menghubungkan suatu langkah dengan langkah lain dalam sebuah flow chart dengan keadaan on page atau off page. Yang dimaksud dengan konektor on page adalah konektor yang digunakan untuk menghubungkan suatu langkah dengan langkah lain dalam satu halaman. Sedangkan konektor off page adalah konektor untuk menghubungkan suatu langkah dengan langkah lain dalam halaman yang berbeda. Konektor ini digunakan apabila ruang gambar yang kita gunakan untuk menggambar flow chart tidak cukup luas untuk memuat flow chart secara utuh, jadi perlu dipisahkan atau digambar di halaman yang berbeda. (a) (b)



Gambar 2.9. Simbol konektor on page (a) dan off page (b)

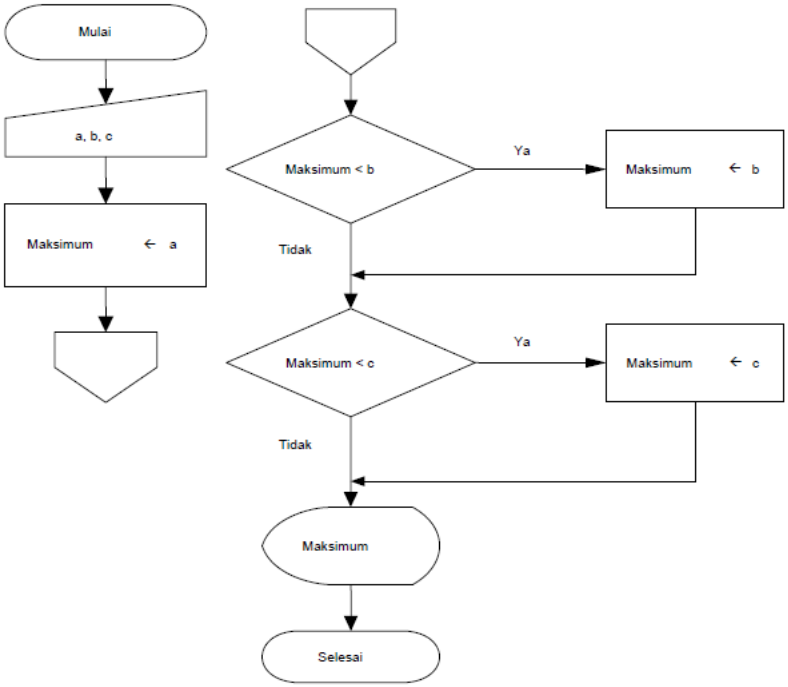
10) Dokumen

Dokumen merupakan tampilan data secara fisik yang dapat dibaca oleh manusia. Data ini biasanya merupakan hasil pemecahan masalah (informasi) yang telah dicetak (print out).



Gambar 2.10. Simbol dokumen

Berikut adalah flow chart untuk permasalahan yang diberikan (mencari bilangan terbesar dari tiga bilangan acak yang dimasukkan)



Gambar 2.11. Flowchart

2.10. Pseudo code

Pseudo code adalah algoritma yang bentuknya (strukturnya) sangat mirip dengan Bahasa pemrograman khususnya bahasa pemrograman terstruktur seperti pascal. Kemiripan ini merupakan keuntungan dari pseudo code karena implementasi atau penerjemahan algoritma ke dalam source code suatu Bahasa pemrograman sangatlah mudah meskipun penggunaannya tidak sepopuler flow chart.

Dalam penulisannya, pseudo code harus terdiri dari tiga bagian, yaitu :

1. Judul algoritma
   Bagian yang terdiri atas nama algoritma dan penjelasan Mulai

Maksimum □ a
a, b, c
Maksimum < b Maksimum □ b
Maksimum < c Maksimum □ c
Maksimum
Selesai
Ya
Ya
Tidak
Tidak

(spesifikasi) dari algoritma
tersebut. Nama sebaiknya
singkat dan menggambarkan
apa yang dapat dilakukan oleh

algoritma tersebut.
2. Deklarasi
Bagian untuk mendefinisikan semua nama yang digunakan di dalam program. Nama tersebut dapat berupa nama tetapan, peubah atau variabel, tipe, prosedur, dan fungsi.
3. Deskripsi
Bagian ini berisi uraian langkahlangkah penyelesaian masalah yang ditulis dengan menggunakan aturan-aturan yang akan dijelaskan selanjutnya. Algoritma untuk permasalahan di atas yaitu mencari bilangan terbesar dari tiga bilangan acak yang dimasukkan dengan menggunakan pseudo code adalah:
Algoritma bilangan_terbesar

```
{algoritma ini mencari bilangan terbesar dari tiga bilangan yang dimasukkan}
deklarasi
        a,b,c,mak : integer
deskripsi
        read(a,b,c)
        mak ← a
        if (mak<b)
              mak ← b
        else if(mak<c)
              mak ← c
        end if
        write(mak)
```

Dalam pseudo code, garis bawah harus digunakan untuk kata algoritma (yang diikuti oleh judul dari algoritma), kata deklarasi, kata deskripsi, tipe data, read, write, if, else, end if, for, end for, while, end while, do while, dan end do while

2.11. Penerjemahan ke kode sumber

Sebelum membahas mengenai penerjemahan algoritma ke dalam bentuk source code bahasa pemrograman (program komputer) C++ atau bahasa lainnya seperti Java maka sebaiknya kita mengetahui lebih dahulu langkah langkah yang biasa dilakukan untuk membuat suatu program komputer.
Langkah-langkah tersebut adalah :

1. Mendefinisikan permasalahan Ini merupakan langkah pertama yang sering dilupakan orang. Menurut hukum Murphy (Henry Ledgard): “Semakin cepat menulis program, akan semakin lama kita dapat menyelesaikannya”. Hal ini terutama dirasakan apabila kita membuat program untuk permasalahan yang komplek. Oleh karena itu sebelum kita menulis source code suatu program sebaiknya kita menentukan atau mengidentifikasi terlebih dahulu inti dari permasalahannya, kemudian apa saja yang dapat dan harus dipecahkan dengan bantuan komputer, dan yang terakhir adalah apa masukan untuk programnya dan bagaimana nanti keluarannya.
2. Menemukan solusi
   Setelah mendefinisikan masalah, maka langkah berikutnya adalah menentukan solusinya. Jika permasalahan terlalu komplek, maka ada baiknya masalah tersebut dipecah menjadi beberapa modul kecil dapat berupa prosedure, fungsi,

atau class sehingga akan lebih mudah untuk diselesaikan. Penggunaan modul ini akan membuat program utamanya menjadi lebih singkat, mudah untuk dilihat dan dianalisis untuk tujuan debuging serta untuk pengembangan dari program.

3. Memilih algoritma
   Pilihlah (atau buatlah) algoritma yang benar-benar sesuai dan efisien untuk permasalahan yang diberikan.
4. Memilih bahasa pemrograman dan menulis source code program
   Bahasa pemrograman yang digunakan hendaknya Bahasa yang memang sudah anda kuasai dengan baik. Atau jika masih dalam tahap belajar, pilihlah bahasa pemrograman yang mudah dipelajari dan digunakan serta memiliki tingkat kompatibilitas tinggi dengan perangkat keras dan platform (sistem operasi) lainnya.
5. Menguji program
   Setelah selesai menulis source code (program sudah jadi), ujilah program tersebut dengan segala macam kemungkinan yang ada, termasuk error-handling, sehingga program tersebut benar-benar handal dan layak untuk digunakan.
6. Menulis dokumentasi
   Dokumentasi sangatlah penting agar pada suatu saat jika kita ingin mengembangkan program, kita tidak mengalami kesulitan dalam membaca source code yang sudah kita tulis. Cara paling mudah dan sederhana dalam membuat suatu dokumentasi adalah dengan menuliskan komentar-komentar kecil pada suatu baris (atau suatu bagian dari source code) tentang apa maksud dari kode-kode tersebut dan apa kegunaannya, kemudian variabel apa saja yang digunakan dan untuk apa, serta parameter parameter yang ada pada suatu prosedur dan fungsi.
7. Merawat program
   Program yang sudah jadi sebaiknya juga dilakukan perawatan untuk mencegah munculnya bug yang sebelumnya tidak terdeteksi. Selain itu juga berguna untuk menambah fasilitas-fasilitas baru yang dahulu sewaktu dibuat belum ada.

Pada bab ini kita hanya fokus pada langkah pertama sampai dengan langkah keempat saja. Sedangkan untuk dapat menerjemahkan notasi algoritma yang telah dibuat ke dalam bentuk source code suatu Bahasa pemrograman, seharusnya kita mamahami terlebih dahulu bahasa pemrograman yang akan kita gunakan, seperti aturan tata bahasanya, intruksi-instruksi yang digunakan, tipe data, dan masih banyak lagi. Semua itu akan kita pelajari satu persatu dibagian lain pada buku ini. Oleh karena kita baru akan mempelajari bahasa pemrograman yaitu C++ atau program lainnya seperti java, maka untuk mempermudah pemahaman tentang algoritma akan diberikan contoh-contoh permasalahan sederhana yang sering dijumpai dalam pemrograman, bagaimana algoritmanya dalam bentuk tiga notasi algoritma yang telah diberikan, dan kemudian bagaimana mengimplementasikan atau menerjemakan algoritma tersebut ke dalam bentuk source code bahasa pemrograman C++ atau program lainnya seperti java.

Untuk membantu penerjemahan algoritma kita juga harus memperhatikan jenis-jenis proses yang biasanya kita temukan dalam algoritma. Dalam algoritma ada empat jenis proses yang dikenal, yaitu :

1. Sequence Process, merupakan instruksi yang dikerjakan secara sekuensial (berurutan).

2. Selection Process, adalah instruksi yang dikerjakan jika memenuhi suatu kriteria atau kondisi (keadaan) tertentu.
3. Iteration Process, yaitu instruksi yang dikerjakan selama memenuhi suatu kriteria atau kondisi (keadaan) tertentu.
4. Concurrent Process, beberapa instruksi dikerjakan secara bersama.

Berikut merupakan contohcontoh permasalahan, algoritma penyelesaian dalam tiga notasi algoritma serta penerjemahannya dalam bahasa pemrograman C++ atau program lainnya seperti java.

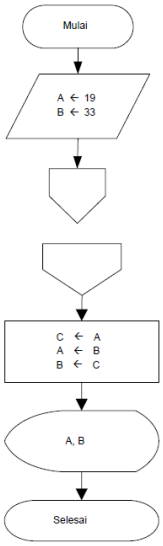
1. Masalah penukaran isi ember (contoh sequence process).
   Masalah penukaran isi ember dapat kita aplikasikan untuk masalah berikut ini. Mula-mula A bernilai 19 dan B bernilai 33. Jika nilai kedua variabel A dan B tersebut ditukar, A menjadi 33 dan B menjadi 19, maka algoritmanya adalah :
   Menggunakan uraian kalimat deskriptif

   1) Set nilai variabel A ← 19 dan B ← 33
   2) Set nilai dari variabel C menjadi sama dengan variabel A.
   3) Set nilai variabel A menjadi sama dengan B.
   4) Set nilai variabel B menjadi sama dengan C.
   5) Tampilkan nilai variabel A dan B.

   Algoritma diatas dapat juga ditulis seperti dibawah ini:

   1) A ← 19 dan B ← 33
   2) C ← A
   3) A ← B
   4) B ← C
   5) Tulis A dan B

Jika algoritma tersebut diatas digambar menggunakan flow chart adalah sebagai berikut:



Gambar 2.12. Gambar flowchart

a) Menggunakan pseudo code
Algoritma tukar_data

```
{algoritma ini digunakan untuk menukarkan dua buah data }
deklarasi
a,b,c : integer
deskripsi
a • 19
b • 33
c • a
a • b
b • c
write(a,b)
```

Dari ketiga notasi algoritma di atas dapat disimpulkan bahwa langkah-langkah pemecahan masalah yang telah dibuat dapat dibagi dalam tiga kelompok, yaitu memberi nilai variabel A dan B, menukarkan nilainya, dan terakhir adalah menampilkannya. Oleh karena itu penerjemahan notasi algoritma ke dalam bentuk source code bahasa pemrograman juga harus mengikuti langkah-langkah yang diberikan dalam ketiga algoritma. Berikut adalah penerjemahan algoritma ke dalam bahasa C++.

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4. int a = 19, b = 33;
5. int c = a;
6. a = b;
7. b = c;
8. cout << “a = “ << a << endl;
9. cout << “b = “ << b;
10. return 0;
11.}
```

Langkah selanjutnya, langkah ke-5 sampai dengan langkah ke-7, merupakan proses penukaran data. Dan langkah terakhir, yaitu menampilkan data yang telah ditukar, ada dalam langkah ke-8 dan 9. Langkah-langkah yang lainnya merupakan aturan tata bahasa dalam bahasa C++ yang akan dijelaskan dalam bab-bab selanjutnya.

Sedangkan penerjemahan algoritma algoritma tersebut dalam Bahasa Java adalah sebagai berikut:

```
1. class tukarData {
2. public static void main(String[ ] args) {
3. int a=19, b=33;
4. int c = a;
5. a = b;
6. b = c;
7. System.out.println("a = " + a);
8. System.out.println("b = " + b);
9. }
10. }
```

Langkah ke-3 merupakan pemberian nilai variabel A dan B, langkah ke-4 sampai dengan langkah ke-6 merupakan proses penukaran data, dan langkah ke- 7 dan 8 merupakan kode untuk menampilkan data yang telah ditukar. Sedangkan langkah yang lainnya merupakan aturan tata bahasa Java yang akan dijelaskan dalam bab-bab selanjutnya.

2. Masalah mencari bilangan terbesar dari tiga bilangan yang dimasukkan (contoh selection process).

   Algoritma pada permasalahan ini sudah diberikan, silakan lihat kembali sub bab notasi algoritma. Dari ketiga notasi algoritma tersebut dapat kita ketahui bahwa ada dua proses pemeriksaan keadaan (data). Pada algoritma dengan uraian kalimat deskriptif, proses pemeriksaan ini dapat kita identifikasi dengan adanya kata kunci JIKA. Dengan demikian pada saat menerjemahkan langkah ke-3 dan langkah ke-5 dari algoritma tersebut ke dalam bentuk source code bahasa C++ atau program lainnya seperti java, kita harus menggunakan klausa if().

   Sedangkan pada algoritma dengan flow chart, proses pemeriksaan keadaan disimbolkan dengan symbol percabangan dimana kalua jalurnya kita telusuri, baik jalur YA maupun TIDAK, akan terhubung dengan langkah penyelesaian masalah selanjutnya, tidak kembali ke langkah sebelumnya ataupun kembali ke symbol percabangan semula karena jika ini terjadi kita tidak bisa menggunakan klausa if() dalam penerjemahan ke bentuk source code bahasa C++ atau program lainnya seperti java (lihat contoh permasalahan berikutnya, yaitu contoh ke-3 dan ke-4).

   Untuk algoritma dengan pseudo code, penerjemahan akan jauh lebih mudah karena seperti yang telah dikemukakan sebelumnya bahwa pseudo code ini strukturnya sangat mirip dengan penulisan source code bahasa pemrograman. Dengan demikian penerjemahan algoritma ke dalam bahasa C++ adalah (perhatikan source code baris ke-12 dan 13):

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4. int a,b,c,mak;
5. cout << "Bilangan pertama = ";
6. cin >> a;
7. cout << "Bilangan kedua = ";
8. cin >> b;
9. cout << "Bilangan ketiga = ";
10. cin >> c;
11. mak = a;
12. if (mak < b) mak = b;
13. else if (mak < c) mak = c;
14. cout << "Bilangan terbesarnya = " << mak;
15. return 0;
16. }
```

Sedangkan penerjemahan dalam bahasa lain misalnya Bahasa Java adalah sebagai berikut (perhatikan source code baris ke-14 dan 15)

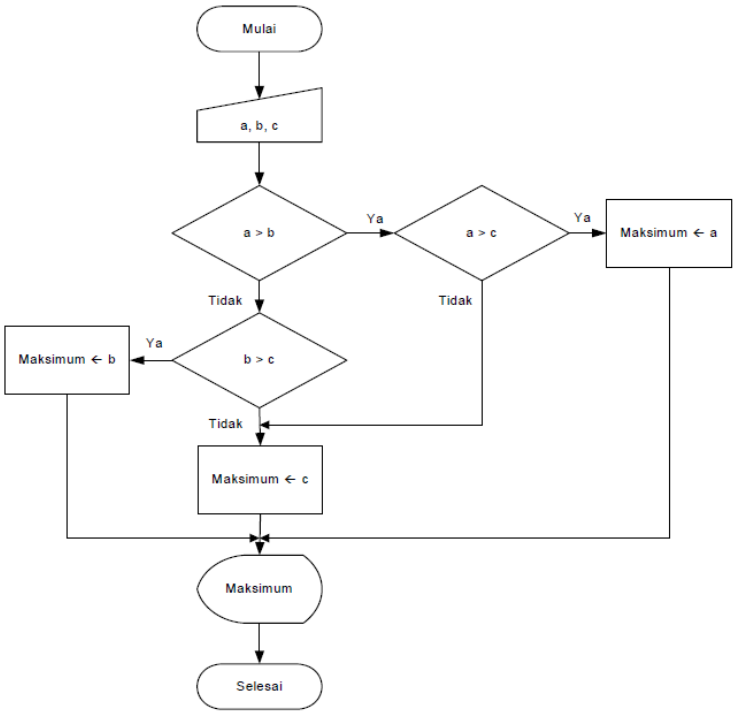
```
1. import java.util.Scanner;
2. import java.io.*;
3. class bilTerbesar {
4. public static void main(String[ ] args) {
5. int a,b,c;
6. Scanner input = new Scanner(System.in);
7. System.out.print("Bilangan pertama = ");
8. a = input.nextInt();
9. System.out.print("Bilangan kedua = ");
10. b = input.nextInt();
11. System.out.print("Bilangan ketiga = ");
12. c = input.nextInt();
13. int mak = a;
14. if (mak < b) mak = b;
```

Pada permasalahan ini, kita juga dapat menggunakan algoritma lain, yaitu :

a) Menggunakan uraian kalimat deskriptif

1) Masukkan a, b, dan c.
2) Jika a > b, maka kerjakan langkah ke - 3. Jika tidak, kerjakan langka ke - 5.
3) Jika a > c, maka kerjakan langkah ke - 4. Jika tidak, kerjakan langka ke - 7.
4) mak □ a.
5) Jika b > c, kerjakan langkah ke - 6. Jika tidak, kerjakan langkah ke - 7.
6) mak □ b.
7) mak □ c.
8) Tulis mak.

b) Menggunakan flow chart



Gambar 2.13. Flowcart

a) Menggunakan pseudo code
Algoritma bilangan_terbesar

```
{algoritma ini mencari bilangan terbesar dari tiga bilangan yang dimasukkan secara acak}
deklarasi
a,b,c,mak : integer
deskripsi
read(a,b,c)
if (a > b)
if (a > c)
mak ← a
else mak ← c
end if
else
if(b > c)
mak ← b
else mak ←  c
end if
end if
write(mak
```

Pada algoritma dengan uraian kalimat deskriptif kita dapat mengidentifikasi bahwa terdapat tiga buah pemeriksaan keadaan, yaitu langkah ke-2, 3, dan 5. Pada langkah ke-

2, jika kondisi terpenuhi, langkah selanjutnya, yaitu langkah ke-3, adalah merupakan pemeriksaan keadaan lagi. Demikian pula jika kondisi yang telah ditentukan tidak terpenuhi, langkah berikutnya, yaitu langkah ke-5, adalah merupakan pemeriksaan keadaan kembali. Dari sini dapat disimpulkan bahwa nantinya penggunaan klausa if() yang kedua (langkah ke-3) akan berada di dalam klausa if() yang pertama (langkah ke-2). Sedangkan penggunaan klausa if() yang ketiga (langkah ke-5) juga ada dalam klausa if() yang pertama pada bagian else. Hal ini dinamakan percabangan bersarang, yaitu percabangan yang ada dalam percabangan lainnya.

Keberadaan percabangan bersarang ini juga dapat diidentifikasi dari simbol percabangan yang terhubung dengan symbol percabangan lainnya. Identifikasi ini lebih jelas terlihat pada algoritma dengan pseudo code. Dengan demikian penerjemahan algoritma ke dalam bahasa C++ adalah:

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4. int a,b,c,mak;
5. cout << "Bilangan pertama = ";
6. cin >> a;
7. cout << "Bilangan kedua = ";
8. cin >> b;
9. cout << "Bilangan ketiga = ";
10. cin >> c;
11. if (a > b) {
12. if (a > c)
13. mak = a;
14. else mak = c;
15. }
16. else {
17. if (b > c)
18. mak = b;
19. else mak = c;
20. }
21. cout << "Bilangan terbesarnya = " << mak;
22. return 0;
23. }
```

Klausa if() pada source code baris ke-12 sampai dengan baris ke- 14 ada di dalam klausa if() baris ke- 11. Sedangkan klausa if() baris ke-17 sampai dengan baris ke-19 juga ada di dalam klausa if() (baris ke-11) bagian else (baris ke-16). Sedangkan penerjemahan algoritma ke dalam bahasa Java adalah (perhatikan baris ke-13 sampai dengan baris ke 22) :

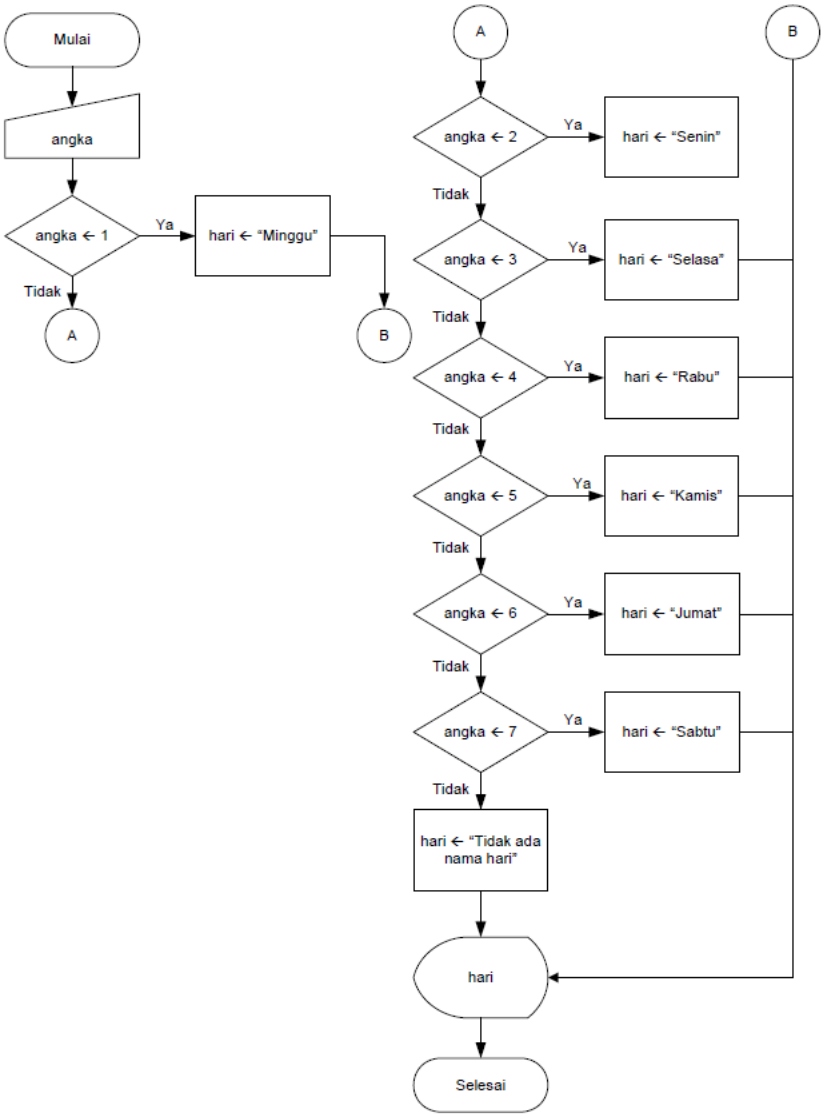
```
1. import java.util.Scanner;
2. import java.io.*;
3. class bilBesar {
4. public static void main(String[ ] args) {
5. int a,b,c,mak;
6. Scanner input = new Scanner(System.in);
7. System.out.print("Bilangan pertama = ");
8. a = input.nextInt();
9. System.out.print("Bilangan kedua = ");
10. b = input.nextInt();
11. System.out.print("Bilangan ketiga = ");
12. c = input.nextInt();
13. if (a > b) {
14. if (a > c)
15. mak = a;
16. else mak = c;
17. }
18. else {
19. if (b > c)
20. mak = b;
21. else mak = c;
22. }
23. System.out.print("Bilangan terbesarnya = " + mak);
24. }
25. }
```

3. Diminta suatu program dimana jika dimasukkan suatu bilangan (1 sampai dengan 7) maka akan menampilkan nama hari, yaitu :
   - Bilangan 1 untuk hari Minggu.
   - Bilangan 2 untuk hari Senin.
   - Bilangan 3 untuk hari Selasa.
   - Bilangan 4 untuk hari Rabu.
   - Bilangan 5 untuk hari Kamis.
   - Bilangan 6 untuk hari Jumat.
   - Bilangan 7 untuk hari Sabtu.

   Algoritmanya adalah : Menggunakan uraian kalimat deskriptif

```
1) Masukan bilangan sebuah bilangan bulat (angka).
2) Jika angka ← 1, maka kerjakan langkah ke - 3. Jika tidak, kerjakan langkah ke - 4.
3) hari ← “Minggu”.
4) Jika angka ← 2, maka kerjakan langkah ke - 5. Jika tidak, kerjakan langkah ke - 6.
5) hari ← “Senin”.
6) Jika angka ← 3, maka kerjakan langkah ke - 7. Jika tidak, kerjakan langkah ke - 8.
7) hari ← “Selasa”.
8) Jika angka ← 4, maka kerjakan langkah ke - 9. Jika tidak, kerjakan langkah ke - 10.
9) hari ← “Rabu”.
10) Jika angka ← 5, maka kerjakan langkah ke - 11. Jika tidak, kerjakan langkah ke - 12.
11) hari ← “Kamis”.
12) Jika angka ← 6, maka kerjakan langkah ke - 13. Jika tidak, kerjakan langkah ke - 14.
13) hari ← “Jumat”.
14) Jika angka ← 7, maka kerjakan langkah ke - 15. Jika tidak, kerjakan langkah ke - 16.
15) hari ← “Sabtu”.
16) hari ← “Tidak ada nama hari untuk angka tersebut”
17) Tampilkan hari.
```

a) Menggunakan flow chart



Gambar 2.14. Flowchart

b) Menggunakan pseudo code
Algoritma nama hari

```
{algoritma ini digunakan untuk mengetahui nama hari dari bilangan bulat
yang dimasukkan}
deklarasi
        angka : integer
        hari : char
deskripsi
read(angka)
if(angka ← 1)
        hari ← “Minggu”
else if(angka ← 2)
        hari ← “Senin”
else if(angka ← 3)
        hari ← “Selasa”
else if(angka ←4)
        hari ← “Rabu”
else if(angka ←5)
        hari ← “Kamis”
else if(angka ←6)
        hari ← “Jumat”
else if(angka ←7)
        hari ← “Sabtu”
else
        hari ← “Tidak ada nama hari”
end if
write(hari)
```

Algoritma diatas dapat juga ditulis seperti dibawah ini: Algoritma nama hari

```
{algoritma ini digunakan untuk mengetahui nama hari dari bilangan bulat
yang dimasukkan}
deklarasi
        angka : integer
        hari : char
deskripsi
        read(angka)
        switch(angka)
                case 1 : hari ←“Minggu”
                case 2 : hari ← “Senin”
                case 3 : hari ← “Selasa”
                case 4 : hari ← “Rabu”
                case 5 : hari ← “Kamis”
                case 6 : hari ← “Jumat”
                case 7 : hari ← “Sabtu”
                default : hari ← “Tidak ada nama hari”
        end switch
        write(hari)
```

Pada algoritma dengan uraian kalimat deskriptif kita dapat mengidentifikasi bahwa terdapat tujuh buah pemeriksaan keadaan, yaitu langkah ke-2, 4, 6, 8, 10, 12, dan 14. Pemeriksaan yang sebanyak ini sangat tidak efektif kalau kita menggunakan klausa if(). Agar program kita lebih efisien, maka kita menggunakan switch() untuk pemeriksaan keadaan sebanyak ini. Dengan demikian penerjemahan algoritma ke dalam bahasa C++ adalah :

```
1. #include <iostream>
2. #include <cstring>
3. using namespace std;
4. int main() {
5. int angka;
6. string hari;
7. cout << "Bilangan = ";
8. cin >> angka;
9. if (angka==1) hari="Minggu";
10. else if (angka==2) hari="Senin";
11. else if (angka==3) hari="Selasa";
12. else if (angka==4) hari="Rabu";
13. else if (angka==5) hari="Kamis";
14. else if (angka==6) hari="Jumat";
15. else if (angka==7) hari="Sabtu";
16. else hari="tidak ada";
17. cout << "Nama hari ke - " << angka << " adalah " << hari;
18. return 0;
19. }
```

Program diatas dapat juga ditulis seperti dibawah ini:

```
1. #include <iostream>
2. #include <cstring>
3. using namespace std;
4.          int main() {
5.          int angka;
6.          string hari;
7.          cout << "Bilangan = ";
8.          cin >> angka;
9.          switch(angka) {
10.                    case 1 : hari = "Minggu";
11.                              break;
12.                    case 2 : hari = "Senin";
13.                              break;
14.                    case 3 : hari = "Selasa";
15.                              break;
16.                    case 4 : hari = "Rabu";
17.                              break;
18.                    case 5 : hari = "Kamis";
19.                              break;
20.                    case 6 : hari = "Jumat";
21.                              break;
22.                    case 7 : hari = "Sabtu";
23.                              break;
24.                    default : hari = "tidak ada";
25.                    }
26.         cout << "Nama hari ke - " << angka << " adalah = " << hari;
27.         return 0;
28.          }
```

Sedangkan dalam contoh dalam bahasa Java adalah :

```
1. import java.util.Scanner;
2. import java.io.*;
3. class bab2_03 {
4.          public static void main(String[ ] args) {
5.          int angka;
6.          String hari;
7.          Scanner input = new Scanner(System.in);
8.          System.out.print("Bilangan = ");
9.          angka = input.nextInt();
10.         if (angka==1) hari="Minggu";
11.         else if (angka==2) hari="Senin";
12.         else if (angka==3) hari="Selasa";
13.         else if (angka==4) hari="Rabu";
14.         else if (angka==5) hari="Kamis";
15.         else if (angka==6) hari="Jumat";
16.         else if (angka==7) hari="Sabtu";
17.         else hari="tidak ada";
18.         System.out.print("Nama hari ke - " + angka + " adalah = " + hari);
19.         }
20.         }
```

Program diatas dapat juga ditulis seperti dibawah ini:

```
1. import java.util.Scanner;
2. import java.io.*;
3. class bab2_03_2 {
4.         public static void main(String[ ] args) {
5.         int angka;
6.         String hari;
7.         Scanner input = new Scanner(System.in);
8.         System.out.print("Bilangan = ");
9.         angka = input.nextInt();
10.        switch(angka) {
11.                case 1 : hari = "Minggu";
12.                        break;
13.                case 2 : hari = "Senin";
14.                        break;
15.                case 3 : hari = "Selasa";
16.                        break;
17.                case 4 : hari = "Rabu";
18.                        break;
19.                case 5 : hari = "Kamis";
20.                        break;
21.                case 6 : hari = "Jumat";
22.                        break;
23.                case 7 : hari = "Sabtu";
24.                        break;
25.                default : hari = "tidak ada";
26.                }
27.                System.out.print("Nama hari ke - " + angka + " adalah = " + hari);
28.        }
29. }
```

4. Masalah menghitung rata-rata dari sejumlah data yang dimasukkan secara manual melalui keyboard (contoh iteration process). Algoritmanya adalah: Menggunakan uraian kalimat deskriptif

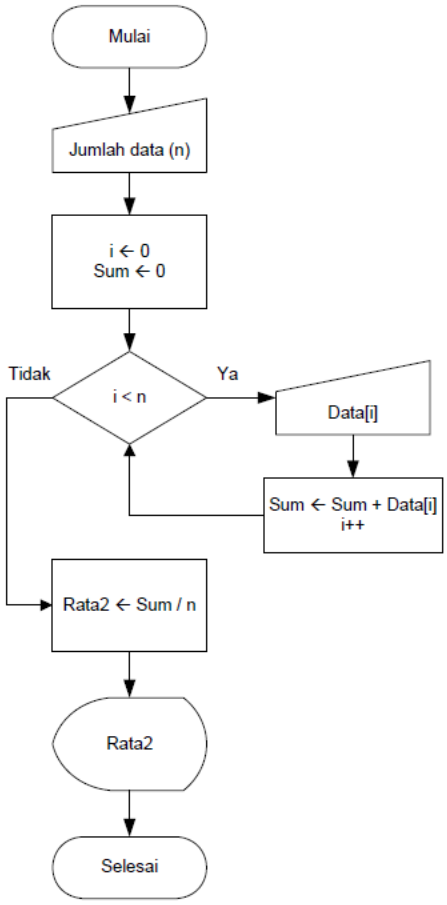
```
1) Masukan jumlah (n) data yang diinginkan.
2) Selama jumlah data yang dimasukkan kurang dari n, kerjakan langkah ke -
3 dan ke - 4. Jika tidak kerjakan langkah ke - 5.
3) Masukan data.
4) Tambahkan data yang dimasukkan dengan data sebelumnya.
5) Hitung rata - rata data.
6) Tampilkan besar rata - ratanya.
```

Uraian diatas dapat juga ditulis seperti dibawah ini:

```
1) Masukkan n
2) i □ 0
3) Sum □ 0
4) Selama i < n, kerjakan langkah ke - 5, 6, dan 7.
5) Masukkan data[i]
6) Sum □ Sum + data[i]
7) i++
8) Rata2 □ Sum / n
9) Tulis Rata2
```

Langkah ke-2 dan ke-3 merupakan proses inisialisasi atau pemberian nilai awal yang diperlukan dalam penulisan source code. Sebaiknya variabel Rata2 juga diinisialisasi agar keluaran dari program lebih valid. Variabel I digunakan sebagai counter (penghitung) untuk proses perulangan yang digunakan untuk memasukkan data satu persatu dan sekaligus menjumlahkan data yang dimasukkan dengan penjumlahan data sebelumnya dimana penjumalahan data sebelumnya disimpan dalam variabel Sum

b) Menggunakan flow chart



Gambar 2.15. Flowchart

Proses perulangan diperlukan karena banyak data yang dimasukkan belum diketahui dengan pasti (lihat langkah pertama yang meminta pengguna untuk memasukkan banyak data). Langkah ke-7 merupakan increment variabel i, yaitu menaikkan nilai variabel I sebesar satu tingkat. Data pada permasalahan ini disimpan dalam tipe data array (data[i]). Tipe data array ini akan dibahas pada bab tersendiri.

c) Menggunakan pseudo code

Algoritma hitung_rata2

```
{algoritma ini digunakan untuk menghitung rata - rata dari sejumlah data yang
dimasukkan melalui keyboard}
deklarasi
        i, n, sum, data[20] : integer
        rata2 : float
deskripsi
        read(n)
        i □ 0
        sum □ 0
        while(i < n)
                read(data[i])
                sum += data[i]
                i++
        end while
        rata2 ← sum / n
        write(rata2)
```

Algoritma diatas dapat juga ditulis seperti dibawah ini:
Algoritma hitung rata2

```
{algoritma ini digunakan untuk menghitung rata - rata dari sejumlah data yang
dimasukkan melalui keyboard}
deklarasi
        i, n, sum, data[20] : integer
        rata2 : float
deskripsi
        read(n)
        sum □ 0
        for(i=0; i<n; i++)
                read(data[i])
                sum += data[i]
        end for
        rata2 ← sum / n
        write(rata2)
```

Langkah ke-2 pada algoritma dengan uraian kalimat deskriptif yang pertama dan langkah ke-4 pada algoritma dengan uraian kalimat deskriptif yang kedua, terdapat kata kunci selama. Kata kunci ini mengidentifikasikan bahwa dalam penerjemahan ke dalam bentuk source code bahasa C++ atau program lainnya seperti java nanti kita akan menggunakan klausa untuk looping (perulangan) yaitu while() dan atau for() karena langkahlangkah tersebut memerintahkan kita untuk melakukan langkah berikutnya, bukan kembali ke langkah sebelumnya, jika kondisi yang telah ditentukan terpenuhi. While() dan for() dapat digunakan secara bergantian jika perulangan yang terjadi dapat diketahui banyaknya (jumlah perulangan yang terjdi), dalam hal ini banyak perulangannya adalah n, yaitu banyak data yang akan dimasukkan. Tetapi jika banyak perulangannya tidak dapat diketahui dengan pasti, maka kita harus menggunakan while(), tidak bisa dengan for(). Pada contoh berikutnya akan diberikan perulangan yang tidak diketahui berapa banyak perulangan yang terjadi. Langkah-langkah yang masuk dalam proses perulangan ini

adalah langkah ke-3 dan 4 pada algoritma dengan uraian kalimat deskriptif yang pertama dan langkah ke-5, 6, dan 7 pada algoritma dengan uraian kalimat deskriptif yang kedua. Pada bab selanjutnya anda akan mengetahui bahwa klausa perulangan ini, dalam bahasa C++ atau program lainnya seperti java, ada tiga macam bentuk yang dapat anda pilih, yaitu for(), while(), atau do while(). Untuk algoritma dengan flow chart, proses perulangan yang menggunakan klausa while() atau for() dapat diidentifikasi dari simbol percabangan dimana kalau jalurnya, dalam hal ini jalur yang berlabel YA, kita telusuri akan terhubung ke symbol berikutnya dan kemudian kembali lagi simbol percabangan semula. Sedangkan untuk algoritma dengan pseudo code, penerjemahan tidak akan mengalami kesulitan karena, sekali lagi, struktur dari pseudo code sangat mirip dengan penulisan source code bahasa pemrograman. Dengan demikian penerjemahan algoritma ke dalam bahasa C++ adalah (perhatikan baris ke-8 sampai dengan baris ke-14 dimana baris ke-9 dan 10 merupakan baris tambahan agar program lebih user friendly).

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4.         int n, i=0, sum=0, data[20];
5.         float rata2;
6.         cout << "Jumlah data (maksimum 20 data) = ";
7.         cin >> n;
8.         while(i < n) {
9.                   int j = i + 1;
10.                  cout << "Masukkan data ke - " << j << " : ";
11.                  cin >> data[i];
12.                  sum += data[i];
13.                  i++;
14.        }
15.        rata2 = sum / n;
16.        cout << "Rata - ratanya = " << rata2;
17.        return 0;
18. }
```

Program diatas dapat juga ditulis seperti dibawah ini:

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4. int n, i, sum=0, data[20];
5. float rata2;
6. cout << "Jumlah data (maksimum 20) = ";
7. cin >> n;
8. for(i=0; i < n; i++) {
9. int j = i + 1;
10. cout << "Masukkan data ke - " << j << " : ";
11. cin >> data[i];
12. sum += data[i];
13. }
14. rata2 = sum / n;
15. cout << "Rata - ratanya = " << rata2;
16. return 0;
17. }
```

Sedangkan jika ditulis dalam bahasa Java adalah sebagai berikut:

```
1. import java.util.Scanner;
2. import java.io.*;
3. class bab2_03 {
4. public static void main(String[ ] args) {
5. int n, i=0, sum=0;
6. int[ ] data = new int[20];
7. float rata2;
8. Scanner input = new Scanner(System.in);
9. System.out.print("Jumlah data (mak. 20) = ");
10. n = input.nextInt();
11. while(i < n) {
12. int j = i + 1;
13. System.out.print("Masukkan data ke - " + j + " : ");
14. data[i] = input.nextInt();
15. sum += data[i];
16. i++;
17. }
18. rata2 = sum / n;
19. System.out.print("Rata - ratanya = " + rata2);
20. }
21. }
```

Program diatas dapat juga ditulis seperti dibawah ini:

```
1. import java.util.Scanner;
2. import java.io.*;
3. class bab2_03_2 {
4. public static void main(String[ ] args) {
5. int n, i, sum=0;
6. int[ ] data = new int[20];
7. float rata2;
8. Scanner input = new Scanner(System.in);
9. System.out.print("Jumlah data (mak. 20) = ");
10. n = input.nextInt();
11. for(i=0;i < n;i++) {
12. int j = i + 1;
13. System.out.print("Data ke - " + j + " : ");
14. data[i] = input.nextInt();
15. sum += data[i];
16. }
17. rata2 = sum / n;
18. System.out.print("Rata - ratanya = " + rata2);
19. }
20. }
```

5. Masalah menentukan Faktor Persekutuan Besar (FPB).

Untuk memecahkan persoalan FPB di atas, kita dapat menggunakan algoritma Euclid yang langkah-langkahnya adalah
sebagai berikut :

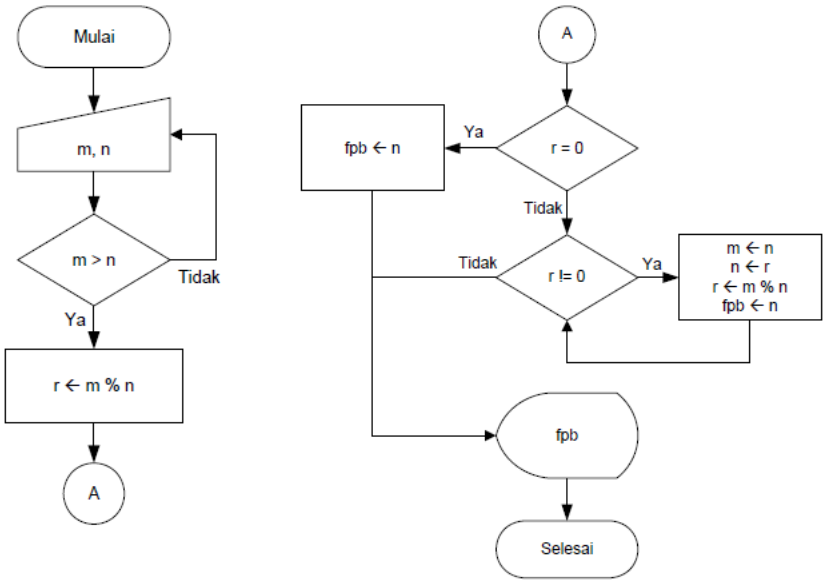
1) Diberikan dua buah bilangan bulat positif, misalnya m dan n dimana syaratnya adalah m > n.
2) Bagilah m dengan n, misalnya hasilnya adalah r.

3) Apabila r = 0, maka hasilnya adalah n itu sendiri (merupakan FPB dari m dan n).
4) Jika r ≠ 0 maka proses pencarian dilanjutkan yaitu ganti m dengan n dan n dengan r dan ulangi langkah ke-2, demikian selanjutnya.

Misalnya akan dicari FPB dari 36 dan 20, maka penyelesaiannyamenurut algoritma Euclid adalah :

1. $\frac{m}{n} = \frac{36}{20}$ → mempunyai sisa
   r = 16
2. r = 16 → r ≠ 0
3. m = 20 dan n = 16
4. $\frac{m}{n} = \frac{20}{16}$ → mempunyai sisa
   r = 4
5. r = 4 ← r ≠0
6. m =16 dan n = 4
7. $\frac{m}{n} = \frac{16}{4}$ → mempunyai sisa
   r = 0
   Jadi FPB(36, 20) = 4

b) Menggunakan flow chart



Gambar 2.16. Flowchart

Algoritma Euclid di atas merupakan algoritma dengan uraian kalimat deskriptif. Algoritma tersebut dapat ditulis kembali sebagai berikut:

a) Menggunakan uraian kalimat deskriptif
1) Masukkan m dan n.
2) Selama m < n ulangi kembali langkah pertama.
3) r ← m % n
4) Jika r = 0 maka fpb ← n.Jika tidak kerjakan langkah ke - 5.
5) Selama r != 0 kerjakan langkah ke - 6 sampai ke - 8.
6) m ← n dan n ← r
7) r ← m % n
8) fpb ← n
9) Tulis fpb.

c) Menggunakan pseudo code

```
{algoritma ini digunakan untuk mencari faktor persekutuan besar dari dua
buah bilangan yang dimasukkan}
deklarasi
        m, n, r, fpb : integer
deskripsi
        do read(m,n)
        while(m<n)
        end do while
        r ← m % n
        if (r == 0) fpb ← n
        else
        while(r != 0)
                m ← n
                n ← r
                r ← m % n
        fpb ← n
        end while
        end if
        write(fpb)
```

Langkah ke-2 pada algoritma dengan uraian kalimat deskriptif mempunyai kata kunci SELAMA yang mengidentifikasikan bahwa dalam penerjemahannya ke dalam source code Bahasa pemrograman nanti menggunakan klausa untuk perulangan. Akan tetapi kita tidak dapat menggunakan klausa while() seperti pada contoh sebelumnya karena pada kasus ini langkah tersebut memerintahkan untuk kembali ke langkah sebelumnya jika kondisi yang diberikan terpenuhi.

Oleh karena itu klausa yang paling tepat digunakan adalah dowhile(). Sedangkan langkah ke-5 kita gunakan klausa while() karena langkah ke-5 memerintahkan kita untuk melakukan langkah-langkah berikutnya, bukan kembali ke langkah sebelumnya. Kita tidak bisa menggunakan klausa for() karena, sesuai dengan penjelasan pada contoh sebelumnya, kita tidak bisa mengetahui dengan pasti berapa banyak perulangan yang terjadi.

Dengan demikian dapat kita simpulkan bahwa jika setelah proses pemeriksaan keadaan memerintahkan kita untuk melakukan langkah-langkah berikutnya maka kita menggunakan klausa while(). Sebaliknya jika setelah proses pemeriksaan keadaan memerintahkan kita untuk kembali lagi melakukan langkahlangkah sebelumnya maka kita menggunakan klausa do while().

Untuk algoritma dengan flow chart, proses perulangan yang menggunakan klausa do while() dapat diidentifikasi dari symbol percabangan dimana kalua jalurnya, dalam hal

ini jalur yang berlabel TIDAK, kita telusuri akan terhubung ke simbol sebelumnya dan kemudian kembali lagi symbol percabangan semula.

Sedangkan untuk algoritma dengan pseudo code, penerjemahan tidak akan mengalami kesulitan karena, sekali lagi, struktur dari pseudo code sangat mirip dengan penulisan source code Bahasa pemrograman. Dengan demikian penerjemahan algoritma ke dalam bahasa C++ adalah (perhatikan baris ke-5 sampai dengan baris ke-10 dan baris ke- 14 sampai dengan baris ke-19)

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4.          int m,n,r,fpb;
5.          do {
6.                    cout << "Masukkan bilangan pertama = ";
7.                    cin >> m;
8.                    cout << "Masukkan bilangan kedua = ";
9.                    cin >> n;
10.         } while (m < n);
11.         r = m % n;
12.         if (r==0) fpb = n;
13.         else {
14.                   while(r!=0) {
15.                             m = n;
16.                             n = r;
17.                             r = m % n;
18.                             fpb = n;
19.                   }
20.         }
21.         cout << "FPB - nya = " << fpb;
22.         return 0;
23. }
```

Sedangkan jika ditulis dalam bahasa Java adalah sebagai berikut:

```
1. import java.util.Scanner;
2. import java.io.*;
3. class bab2_04 {
4.          public static void main(String[ ] args) {
5.                    int m,n,r,fpb;
6.                    Scanner input = new Scanner(System.in);
7.                    do {
8.                              System.out.print("Masukkan bilangan pertama = ");
9.                              m = input.nextInt();
10.                             System.out.print("Masukkan bilangan kedua = ");
11.                              n = input.nextInt();
12.                   } while(m < n);
13.                             r = m % n;
14.                             fpb = 0;
15.                             if (r == 0) fpb = n;
16.                   else {
17.                   while(r != 0) {
18.                              m = n;
19.                             n = r;
20.                             r = m % n;
21.                             fpb = n;
22.                   }
23.         }
24.         System.out.print("Bilangan terbesarnya = " + fpb);
25.         }
26. }
```

BAB 3
TIPE DATA DAN OPERATOR

3.1. Pengertian Data

Data merupakan bentuk jamak dari bahasa Latin dengan kata *datum*, yangberarti fakta atau sesuatu yang diberikan. Data adalah kelompok simbol-simbol yang teratur dan mewakili kuantitas, tindakan, benda dan sebagainya. Dalam istilah umum data mewakili angka, karakter dan simbol-simbol lain yang berfungsi sebagai masukan untuk proses komputer. Data bisa berujut suatu keadaan, gambar, suara, huruf, angka, matematika, bahasa ataupun simbol-simbol lainnya yang bisa kita gunakan sebagai bahan untuk melihat lingkungan, obyek, kejadian ataupun suatu konsep.

Data belum mempunyai arti apabila tidak diolah. Data yang telah diolah menjadi sebuah bentuk yang berarti disebut informasi . Data yang mewakili simbol-simbol bukan merupakan informasi kecuali dalam

pengertian tertentu. Pada komputer data disimpan dalam memori sebelum dan sesudah pemrosesan oleh mikroprosesor.

Jenis data dalam setiap bahasa pemrograman belum tentu sama, namun sebagian besar biasanya terbagi menjadi tiga, yaitu :

1. Data Numerik atau bilangan, yaitu jenis data yang digunakan dalam proses aritmatika atau proses matematis lainnya.
2. Data String, yaitu jenis data yang dapat terdiri dari berbagai macam karakter. Digunakan untuk proses yang non matematis.
3. Data Logika, yaitu data yang hanya terdiri dari dua satuan, yaitu benar (true) dan salah (false). Digunakan dalam suatu proses logika yang terdiri dari persamaan Boolean

3.2. Indentifier

Identifier adalah nama yang didefinisikan oleh programmer dan digunakan untuk mewakili sebuah elemen pada program. Nama variabel merupakan salah satu contoh dari identifier. Programmer dapat memilih sendiri nama sebuah variabel pada C++, selama tidak menggunakan salah satu dari kata kunci (*keyword)* yang dimiliki oleh C++. *Keywords* atau kata kunci merupakan “inti” pada bahasa dan memiliki tujuan tertentu. Tabel dibawah ini menunjukkan daftar lengkap kata kunci C++. Dan yang perlu diperhatikan pada bahasa C adalah bahwa semua harus huruf kecil.

asm	auto	break	bool	case
catch	char	class	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
false	float	for	friend	goto
if	inline	int	long	mutable
namespace	new	operator	private	protected
public	register	reinterpret_cast	return	short
signed	sizeof	static	static_cast	struct
switch	template	this	throw	true
try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t	while			

Indentifier atau pengenal adalah nama yang diberikan untuk nama variable, nama konstanta, nama fungsi, nama objek, nama method nama class, dan obyek yang lain yang didefinisikan oleh pemrogram.

Dalam menulis program harus selalu memilih nama variabel yang memberikan indikasi mengenai hal yang berhubungan dengan yang digunakan variable tersebut. Jika dimungkinkan mendeklarasikan sebuah variabel dengan nama seperti berikut ini:

```
int x:
```

maka x tidak termasuk sesuatu jenis tertentu,sehingga hal tersebut tidak memberikan petunjuk pada tujuan variabel. Perhatikan contoh berikut dibawah yang lebih baik.

```
int daftarUrutan:c
```

Nama daftarUrutan memberikan sesbuat pada pembaca program sehingga akan ide dari variabel yang digunakan. Cara coding ini akan membantu dalam menghasilkan dokumentasi program sendiri, yang berarti anda akan mendapatkan pemahaman tentang apa yang dilakukan program hanya dengan membaca kode yang digunakannya.

Karena dunia program biasanya ada ribuan baris, hal ini penting agar bisa diingat oleh programmer itu sendiri dan sebagai dokumentasi yang baik. Selain itu yang perlu disadari adalah adanya campuran huruf besar dan huruf dalam variable nama daftarUrutan. Walaupun semua C++, kata kunci harus ditulis dalam huruf kecil, tetapi dalam penulisan tersebut dapat menggunakan huruf besar sebagai variabel nama.

Dalam bahasa pemrograman C maupun pada bahasa java identifier sifatnya case sensitive, artinya huruf besar dan huruf kecil dianggap berbeda artinya, walaupun dipernolehkan. Identifier terdiri dari :

- Karakter alphabet, yaitu : huruf ‘A’ sampai ‘Z’ dan huruf ‘a’ sampai ‘z’
- Underscore (_) dan tanda dollar ($)
- Digit decimal, yaitu: bilangan antara ‘0’ sampai ‘9’

Dalam menulis program ada halhal yang harus diperhatikan sebagai ketentuan dalam pemberian nama identifier. Ketentuan-ketentuan dalam penulisan identifier antara lain:

- Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf atau underscore atau tanda dollar ($).
- Tidak boleh mengandung spasi dan symbol-simbol khusus, kecuali garis bawah (underscore) atau tanda dollar ($) khusus untuk java. Yang termasuk symbol khusus yang tidak diperbolehkan antara lain: $, ?, %, #, !, &, *, (,), -, +, =, dan sebagainya.
- Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai, tetapi sebaiknya sependek mungkin minimal satu karakter.
- Tidak boleh sama dengan kata kunci (keyword) dan kata tercadang (reserved word**)** yang ada dalam bahasa pemrograman.

Untuk lebih jelasnya mengenai identifier dapat dilihat pada contoh identifier yang benar dan yang salah

Tabel 3.1. Penulisan Identifier

PENULISAN	**KETERANGAN IDENTIFIER**
Nomor Nama Nilai_siswadata1	Penulisan Benar

Rekam_1 _temp $harga_jual	
1data Nomor - siswa,nama siswa if 2_jalan 2$ main	Penulisan Salah

3.3. Konstanta

Suatu data yang sifatnya tetap, dan digunakan dalam pemrograman diistilahkan dengan konstanta. Konstanta adalah sebuah nama tempat penyimpanan sementara di dalam memori yang nilainya tetap atau tidak dapat diubah. Konstanta harus didefinisikan terlebih dahulu pada awal program. Konstanta dapat bernilai integer, pecahan, karakter atau string. Perhatikan contoh sebuah konstanta berikut ini:

```
50
3.14
'A'
"Bahasa Pemrograman"
```

Dalam melakukan pemrograman ada dua tipe konstanta yaitu konstanta bilangan dan konstanta teks atau string.

3.3.1. Konstanta Bilangan

Konstanta bilangan atau sering juga disebut dengan konstanta numeric yaitu suatu konstanta yang nilai tetapnya berupa bilangan. Konstanta bilangan atau numerik dibagi dalam dua kelompok, yaitu :

a. Konstanta bilangan bulat (integer)

Konstanta bilangan bulat merupakan sebuah konstanta yang nilai tetapnya berupa bilangan bulat. Konstanta bilangan bulat dapat disajikan dalam bentuk decimal, octal ataupun heksadesimal. Dalam sistem decimal, bilangan yang digunakan berkisar antara 0 sampai 9. Penulisan konstanta langsung angkanya, misalnya: 10, 9, 5 dan sebagainya.

Dalam sistem octal, bilangan yang digunakan berkisar antara 0 sampai dengan 7. Penulisan konstanta diawali dengan 0 misalnya: 010, 046, 055 dan sebagainya. Dalam sistem heksadesimal, bilangan yang digunakan berkisar antara 0 sampai dengan F. Bilangan yang dapat dipakai berupa salah satu diantara 16 simbol adalah berikut:

```
0 1 2 3 4 5 6 7 8 9 A B C D E F
```

Bisa juga ditulis sebagai berikut:

```
0 1 2 3 4 5 6 7 8 9 a b c d e f.
```

Cara penulisan konstanta bilangan ini diawali dengan 0x (nol dan x), misalnya: 0xAF, 0x7F, dan sebagainya.

b. konstanta bilangan real atau pecahan

Konstanta dengan nilai tetapnya berupa bilangan pecahan. Konstanta bilangan real atau pecahan dibagi dua jenis, yaitu: Konstanta data bilangan desimal berpresisi tunggal (floating point), dimana bilangan ini memiliki derajat ketelitian sampai 7 digit dan dapat dinyatakan dalam dua bentuk tampilan, yaitu :

- Bentuk desimal, contoh : 21.333
- Bentuk eksponensial, dituliskan dengan notasi *scientific*. Dengan bentuk umum: bulat pecahan{E|D} {[+] | [-]} pangkat misalnya: 0.21333E+2 dimana hal penulisan ini mempunyai arti: 0.2133 x 102

  Sedangkan konstanta data bilangan desimal berpresisi ganda (double precision) serupa dengan floating point, hanya derajat ketelitian dan range jangkauan yang dimiliki lebih tinggi. Derajat ketelitian untuk floating point 7 digit, sedangkan double precision 16 digit.

3.3.2. Konstanta Teks/String.

Konstanta teks atau string adalah suatu konstanta yang nilai tetapnya berupa teks. Konstanta teks dibedakan dalam dua jenis, yaitu:

a. Konstanta data karakter

Konstanta data karakter terdiri dari sebuah karakter saja dan ditandai dengan dua tanda kutip tunggal (‘.’) sebagai pembatasnya. misalnya: ‘A’, ‘b’, ‘&’, ‘!’. Selain itu juga ada beberapa diantara konstanta data karakter yang ditulis dengan diawali tanda \ (penempatan tanda \ setelah tanda petik tunggal pembuka).

Karakter ini dinamakan rangkaian escape (escape sequence), sehingga disebut dengan konstanta karakter rangkaian escape. Beberapa karakter rangkaian escape, antara lain :

Tabel 3.2. karakter rangkaian ESCAPE

KARAKTER	KETERANGAN
\a	Untuk bunyi bell (alert)
\b	Mundur satu spasi (backspace)
\f	Ganti halaman (form feed)
\n	Ganti baris baru (new line)
\r	Ke kolom pertama, baris yang sama
\v	Tabulasi vertical
\0	Nilai kosong (null)
\’	Karakter petik tunggal
\”	Karakter petik ganda
\\	Karakter garis miring

Untuk menyatakan sembarang karakter ASCII, notasi yang dapat dipergunakan, antara lain: \DDD dengan DDD = 1 sampai dengan 3 digit octal atau \xHH dengan HH = 1 sampai dengan 2 digit heksadesimal, misalnya: ‘\”’ atau ‘\42’ atau ‘\x22’. Selain karakter biasa, juga terdapat karakter khusus yang didefinisikan dengan cara mengawalinya menggunakan tanda \ seperti pada dijelaskan pada tabel dibawah ini:

Tabel 3.3. karakter khusus

KODE	NAMA	NILAI UNICODE
\t	Tab	\u0009
\b	Backspace	\u0008

\n	Linefeed	\u000a
\r	Carriage return	\u000d
\’	Single quote	\u0027
*	Double quote	\u0022
\\	Backslash	\u005c

b. Konstanta data string.

Konstanta data string merupakan rangkaian dari beberapa karakter dan ditandai dengan dua tanda kutip ganda (“ ”) sebagai pembatasnya. Perhatikan contoh penulisan dibawah ini:

```
“Helo”, “Lagi Belajar”
“Bahasa Pemrograman”, “Di Sekolahmu ya?”
```

Dalam penulisan program, sebuah konstanta perlu dideklarasikan, dimana deklarasi konstanta merupakan proses untuk menyebutkan karakteristik konstanta seperti nama, tipe data, jangkauan, masa hidup, dan nilai awal. Dalam bahasa C pendeklarasikan konstanta ada 2 cara, yaitu: dengan menggunakan pengarah compiler atau preprocessor #define, dan dengan keyword const. penulisan sintaks dari deklarasi konstanta dalam bahasa C adalah sebagai berikut:

```
#define <namakonstanta> <nilai>
```

Atau dapat juga ditulis sebagai berikut:

```
const <tipedata> <namakonstanta> =
<nilai>;
```

Untuk lebih jelasnya pehatikan contoh program dibawah ini:

```
#define nomor 100
#define phi 3.14
#define huruf ‘K’
#define nama “Hello”
```

Atau bisa juga ditulis seperti program dibawah ini:

```
const int nomor = 100;
const float phi = 3.14;
const char huruf = ‘K’;
const String nama=“Hello”;
```

Program 3.1. Penggunaan konstanta dalam program

```
#include <conio.h>
#include <iostream>
#define nomor 100
#define phi 3.14
#define huruf 'K'
#define nama "Hello"
using namespace std;
main()
{
        cout << "Nilai konstanta nomor: "<< nomor;
        cout << "\nNilai konstanta phi : "<< phi;
        cout << "\nNilai konstanta huruf: "<< huruf;
        cout << "\nNilai konstanta nama : "<< nama;
        getch();
        return 0 ;
}
```

Keluaran programnya adalah sebagai berikut:

```
Nilai konstanta nomor: 100
Nilai konstanta phi : 3.14
Nilai konstanta huruf: K
Nilai konstanta nama : Hello
```

Pada program dibawah ini hasil setelah decompile akan sama dengan program diatas. Program dibawah merupakan salah satu contoh penulisan yang berbeda tetapi akan menghasilkan hasil yang sama dengan program sebelumnya.

Perhatikan pada penggunaankonstanta dalam aplikasi program dibawah ini:

Program 3.2

```
#include <conio.h>
#include <iostream>
const int nomor = 100;
const float phi = 3.14;
const char huruf = 'K';
const char nama[] = "Hello";
using namespace std;
main()
{
        cout << "Nilai konstanta nomor: "<< nomor;
        cout << "\nNilai konstanta phi : "<< phi;
        cout << "\nNilai konstanta huruf: "<< huruf;
        cout << "\nNilai konstanta nama : "<< nama;
        getch();
        return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Nilai konstanta nomor: 100
Nilai konstanta phi : 3.14
Nilai konstanta huruf: K
Nilai konstanta nama : Hello
```

3.4. Variabel

Variabel atau pengubah adalah suatu lambang dari sebuah lokasi yang berada dimemori utama komputer yang dapat berisi suatu nilai. Nilai yang berada di lokasi memori tersebut dapat berubah selama program dieksekusi. Guna variable adalah sebagai tempat sementara untuk menyimpan data yang sedang diolah. Pemberian nilai kedalam suatu variabel mempunyai bentuk penulisan yang berbeda-beda pada setiap bahasa pemrograman. Meskipun mempunyai arti yang sama dalam pemrogramannya.

Variabel dibedakan atas variable numerik dan variabel string. Variabel numerik adalah variabel yang mengandung nilai numerik atau angka sedangkan variabel string adalah variabel yang berisi nilai huruf /alpha-numerik. Suatu variabel dapat mewakili:

- Nilai Konstanta

```
double pecahan = 13.45;
int nilai = 85:
```

- Nilai dari pengubah lain

```
char abjad = "T" ;
char huruf = abiad:
```

- Nilai yang diperoleh dari kombinasi beberapa pengubah atau nilai konstanta dengan melalui operator.

```
double pi = 3.141593;
double r;
double l  lingkaran = pi * r * r;
```

3.4.1. Tipe Variabel

Berdasarkan pada jenis data yang disimpan pada variabel yaitu: Variabel numeric dan variable teks. Pada variable numeric merupakan suatu variabel yang dipakai untuk menampung data berupa angka.

Variabel numerik digolongkan atas: bilangan bulat atau integer dan bilangan real atau pecahan, dimana bilangan real terdiri dari dua yaitu:

- bilangan decimal berpresisi tunggal (floating point)
- bilangan decimal berpresisi ganda (double precision).

Sedangkan pada variable teks merupakan suatu variabel yang dipergunakan untuk menampung data berupa huruf atau teks. Variabel teks dibedakan menjadi dua yaitu:

- Karakter (untuk karakter tunggal)
- String (untuk rangkaian karakter)

3.4.2. Jenis Variabel

Berdasarkan pada lokasi atau wilayah dan waktu pengaksesannya, variabel dapat dibedakan atas 2 jenis, yaitu: Variabel Lokal dan Variabel Global. Variabel local merupakan Variabel yang dideklarasikan pada fungsi termasuk fungsi utama (main), variable dialokasikan selama fungsi pendeklarasi dijalankan dan Variabel didealokasikan jika fungsi pendeklarasi tidak dijalankan.

Sedangkan Variabel Global adalah Variabel yang dideklarasikan sebelum definisi fungsi termasuk funsi utama, Variabel dialokasikan selama program dijalankan dan Variabel didealokasikan setelah program selesai dijalankan

3.4.3. Deklarasi Variabel

Deklarasi variable adalah proses untuk menyebutkan karakteristik variable seperti nama, tipe data, jangkauan, masa hidup, dan nilai awal. Penulisan Sintaks untuk mendeklarasikan sebuah variable adalah sebagai berikut:

```
<tipedata> <namavariabel> [= nilaiawal];
<tipedata> <namavariabel 1> [= nilaiawal], …, <namavariabel n> [= nilaiawal];
```

Dalam penulisan sintak tersebut yang perlu diperhatikan adalah bahwa semua yang ada dalam tanda <…> harus ada dan semua yang ada dalam tanda […] bersifat tambahan boleh ada atau boleh tidak tergantung keperluan. Perhatikan contoh penulisan dibawah ini:

```
int bilangan;
char karakter;
float bildesimal:
```

Setelah mendeklarasikan sebuah variabel dengan tipe data, selanjutnya memberikan nilai variable tersebut dengan tanda = (sama dengan)

```
bilangan = 20;
karakter = 'k';
bildesimal = 22.2;
```

ata

```
int bilangan = 20;
char karakter = 'k';
char huruf, huruf1 = 'C', huruf2 = 'O';
float bildesimal = 22.2;
```

3.4.4. Kelas Variabel

Selain digolongkan berdasarkan tipe datanya, sebuah variabel juga diklasifikasikan berdasarkan kelas penyimpanan, yaitu: Variable local atau otomatis, variable eksternal, variable statis dan variable register

Variable local adalah merupakan variabel yang dideklarasikan dalam sebuah fungsi dan mempunyai sifatsifat sebagai berikut:

- Secara otomatis diciptakan ketika fungsi dipanggil dan akan lenyap ketika eksekusi terhadap fungsi berakhir
- Hanya dikenal oleh fungsi tempat variable dideklarasikan
- Tidak ada inisialisasi secara otomatis (saat variable diciptakan nilainya tak menentu)
- Deklarasinya dengan kata kunci auto.

Perhatikan penulisan variable local berikut ini:

```
auto int x;
```

Untuk lebih jelasnya perhatikan program dengan menggunakan variable local berikut ini:

Program 3.3

```
#include <conio.h>
#include <iostream>
using namespace std;
void tampil(void);
main()
{
auto int i = 11;
tampil();
cout <<"\ni dalam main() = "<< i;
getch();
return 0 ;
}
void tampil(void)
{
auto int i = 10;
cout <<"\ni dalam tampil() = "<< i;
}
```

Keluaran program diatas adalah sebagai berikut:

```
i dalam tampil() = 10
i dalam main() = 11
```

selain variable local seperti dijelaskan diatas, juga terdapat kelas variable eksternal dimana variable ini mendeklarasikan diluar sebuah fungsi, dan mempunyai sifat berikut:

- Dapat diakses oleh semua fungsi
- Kalau tidak diberi nilai, secara otomatis diinisialisasi dengan nilai sama dengan nol
- Deklarasinya dengan kata kunci extern didalam fungsi yang menggunakan variable jenis ini.
  Perhatikan penulisan variable eksternal berikut ini:

```
extern int x;
```

untuk lebih jelasnya perhatikan program dengan menggunakan variable eksternal berikut ini:
Program 3.4

```
#include <conio.h>
#include <iostream>
using namespace std;
/* variabel j disini merupakan variabel eksternal bagi
fungsi kali dan fungsi main, sehingga variabel j dapat
diakses oleh semua fungsi yang ada yaitu kali dan main */
int j = 1;
void kali(void);
main()
{
        cout <<"Nilai awal j = "<< j<<"\n";
        j = j * 5;
        cout <<"Nilai j kemudian = "<< j<<"\n";
        kali();
        cout <<"Nilai j kini = "<< j<<"\n";
        kali();
        cout <<"Nilai j sekarang = "<< j<<"\n";
        getch();
        return 0 ;
}
void kali(void)
{
j*=10;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Nilai awal j = 1
Nilai j kemudian = 5
Nilai j kini = 50
Nilai j sekarang = 500
```

Variable statis merupakan sebuah variable internal yang didefinisikan didalam fungsi maupun variable eksternal, dan variable ini mempunyai sifat-sifat sebagai berikut:

- Kalau bersifat internal, maka hanya dikenal oleh fungsi tempat variable dideklarasikan
- Kalau bersifat eksternal, maka dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama tempat variable dideklarasikan
- Tidak hilang sekeluarnya dari fungsi
- Inisialisasi hanya dilakukan sekali pada saat fungsi dipanggil pertama kali, jika tidak ada inisialisasi oleh pemrogram secara otomatis akan diberi nilai awal nol.
- Deklarasinya dengan kata kunci static.
  Perhatikan penulisan variable statis berikut ini:

```
static int x;
```

Untuk lebih jelasnya perhatikan program dengan menggunakan variable statis berikut ini:

Program 3.5

```
#include <conio.h>
#include <iostream>
using namespace std;
void tambah(void);
main()
{
        int k = 100;
        tambah();
        tambah();
        cout<<"Nilai k dalam main() = "<< k<<"\n";
        getch();
        return 0 ;
}
void tambah(void)
{
        static int k; //variabel statis
        k++;
        cout <<"Nilai k dalam tambah() = "<< k<<"\n";
}
```

Keluaran program diatas adalah sebagai berikut:

```
Nilai k dalam tambah () = 1
Nilai k dalam tambah () = 2
Nilai k dalam main () = 100
```

Variable register adalah variable yang nilainya disimpan dalam register mikroprosesor bukan didalam memori RAM, variable ini mempunyai sifatsifat sebagai berikut:

- Mempunyai kecepatan akses lebih tinggi
- Hanya dapat diterapkan pada variable local atau parameter formal yang bertipe char atau int
- Biasa diterapkan untuk pengendalian loop.
- Deklarasinya dengan kata kunci register.

Perhatikan penulisan variable register berikut ini:

```
register int x;
```

untuk lebih jelasnya perhatikan program dengan menggunakan variable register berikut ini:

Program 3.6

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
        register int n, jml;//variabel register
        int m = 242;
        for(n=1; n<=m; n++)
        jml += n;
        cout <<" 1 + 2 + 3 + ... + = " << m<< jml;
        getch();
        return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
1 + 2 + 3 + … + 242 =32621
```

3.5. Tipe Data

Tipe data merupakan tempat untuk menentukan pemberian nilai terhadap suatu variabel yang diberikan oleh user. Selain itu tipe data juga dapat diartikan sebagai batasan terhadap fungsi tanda pengenal terhadap semua nilai yang diterima. Sebagai gambaran dari pengertian tersebut adalah ketika kita menempatkan tanda pengenal harga hanya mengenal angka, maka ketika kita memberikan nilai berupa string maka secara otomatis data tersebut akan ditolak karena nilai tersebut tidak dikenali oleh tipe data yang diberikan. Tipe data dalam variable menentukan tipe data yang bisa disimpan didalamnya, format data yang disimpan, dan berapa banyak memori yang dialokasikan untuk menyimpan data.

Tipe data tersebut diklasifikasikan berdasarkan bagaimana keadaan data disimpan dalam memori, dan jenis operasi yang dapat dilakukan.

Tipe data merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh komputer. Misalnya saja 3 dibagi 2 bisa saja menghasilkan hasil yang berbeda tergantung tipe datanya. Jika 3 dan 2 bertipe integer, maka program akan menghasilkan nilai 1, namun jika keduanya bertipe float maka akan menghasilkan nilai 1.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif. Fungsi Tipe data antara lain: Merepresentasikan nilai dari suatu variabel maupun konstanta, Penyimpanan data di memori, dan Menentukan nilai yang dapat diisikan ke dalam sebuah variable.

Sedangkan jenis tipe data ada dua macam: Tipe dasar atau primitive dan Tipe bentukan atau referensi. Dalam bahasa C terdapat lima tipe data dasar, yaitu: int, char, float, double, dan void. Sedangkan dalam bahasa pemrograman lain seperti java misalnya memiliki tipe data yang dapat dikategorikan menjadi dua kelompok, yaitu tipe data primitif dan referensi. Ada delapan macam tipe data primitif dalam pemrograman

Java, yaitu : char, byte, short, int, long, float, double, dan Boolean. Tipe data referensi digunakan untuk mereferensikan objek atau class tertentu, seperti String.
Tabel 3.4. Tipe Data (16 bit) Dalam C

TIPE	UKURAN	RANGE
unsigned char	8 bit	0 - 255
char	8 bit	- 128 - 127
enum	16 bit	- 32768 - 32767
unsigned int	16 bit	0 - 65535
short int	16 bit	- 32768 - 32767
int	16 bit	- 32768 - 32767
unsigned long	32	bit 0 - 4294967295
long	32 bit	- 2147483648 - 2147483647
float	32 bit	3.4E - 38 - 3.4E+38
double	64 bit	1.7E - 308 - 1.7E+308
long double	80 bit	3.4E - 4932 - 1.1E+4932
void	0 bit	
near (pointer)	16 bit	
far (pointer)	32 bit	

Tabel 3.5. Tipe Data (32 bit) Dalam C

Tipe	Ukuran	Range
unsigned char	8 bit	0 - 255
char	8 bit	- 128 - 127
short int	16 bit	- 32768 - 32767
unsigned int	32 bit	0 - 4294967295
int	32 bit	- 2147483648 - 2147483647
unsigned long	32 bit	0 - 4294967295
enum	32 bit	- 2147483648 - 2147483647
Long	32 bit	- 2147483648 - 2147483647
float	32 bit	3.4E - 38 - 3.4E+38
double	64 bit	1.7E - 308 - 1.7E+308
long double	80 bit	3.4E - 4932 - 1.1E+4932
void	0 bit	
near (pointer)	16 bit	
far (pointer)	32 bit	

Tabel 3.6. Format Tipe Data Dalam C

TIPE	FORMAT	KETERANGAN
char	%c	Karakter/string
int	%i, %d	Integer/bilangan bulat
float	%f	Float/bilangan pecahan presisi tunggal
double	%lf	Bilangan pecahan presisi ganda

3.5.1. Tipe data Integer

Program Komputer merupakan kumpulan potongan data dan memanipulasi data tersebut dalam berbagai cara. Ada berbagai jenis data dalam bidang informasi, misalnya: bilangan, terdapat bilangan bulat dan bilangan pecahan, ada bilangan positif dan negative, dan ada pula bilangan besar serta bilangan kecil dan lain-lain, atau bahkan bilangan yang tidak memiliki nama.

Kemudian ada juga informasi tekstual, misalnya Nama dan alamat, maka data seperti ini akan disimpan sebagai karakter. Bila programmer menulis sebuah program, maka harus menentukan jenis informasi yang digunakannya. Jika menulis sebuah program untuk menentukan jauhnya berapa mil dari bumi ke bintang, maka diperlukan variabel yang dapat menampung jumlah yang sangat besar. Jika Anda sedang merancang perangkat lunak untuk merekam sebuah dimensi mikroskopis, maka hanya perlu untuk menyimpan data yang sangat kecil dan angka sesuai. Sebagai tambahan, jika Anda menulis sebuah program yang harus melakukan ribuan perhitungan intensif, maka akan menginginkan variabel yang dapat diproses dengan cepat. TipeData variabel inilah yang akan menentukan semua factor tersebut.

Bahasa C++ menyediakan berbagai tipe data, yang secara garis besar hanya dibagi menjadi dua yaitu: data numerik dan karakter. Numerik merupakan tipe data yang dibagi menjadi dua kategori yaitu: integer dan floating-point. Integers adalah bilangan yang meliputi seluruh nomor seperti 12, 157, -34, 2 daln lain sebagainya. Floating-point merupakan angka yang ada angka decimal di belakang koma, misalnya 23,7, 189,0231, 0,987 dan lain sebagainya.

Sebelum kita membicarakan tipe data karakter, mari kita hati-hati memeriksa variasi dari data numerik. Pertimbangan utama untuk memilih tipe data numerik adalah:

- Angka Terbesar dan terkecil yang dapat disimpan dalam variable.
- Berapa banyak memori yang digunakan oleh variabel.
- Apakah variabel menangani penyimpanan (baik positif maupun negatif) atau bilangan unsigned (hanya positif).
- Bilangan decimal merupakan variabel yang presisi

Ukuran variabel adalah jumlah byte memori yang digunakan. Biasanya, semakin besar range variable, maka jumlah yang digunakan akan lebih besar pula.

3.5.2. Tipe Data Karakter

Karakter adalah sembarang huruf, angka, atau tanda baca tunggal. Tipe data karakter merupakan kumpulan bermacam macam karakter yang terdiri dari alfabet. Diman karakter antara lain: Alfabet bilangan decimal: 0, 1, 2, …, 9, Alfabet huruf latin besar : A, B, C, …, Z, Alfabet huruf latin kecil : a, b, c, ... , z dan Tanda baca tunggal : !, @, ~, ?, ;, ‘, &, dan sebagainya.

Tipe data karakter ini hanya terdiri dari 1 karakter dan Bentuk tipe data dari karakter yaitu: char. Perhatikan penulisan contoh deklarasi char, dimana char huruf yang ditampilkan adalah = ‘A’ ;

Program 3.7

```
include <iostream>
using namespace std;
main()
{
        char huruf_1 = 'C', huruf_2 = '+';
        cout <<"Tipe Data Char pada " << huruf_1 << huruf_2<< huruf_2;
        getch();
        return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Tipe Data Char pada C++
```

3.5.3. Tipe Data String

String adalah deretan karakter yang diakhiri dengan sebuah karakter kosong. Konstanta bertipe ditulis diantara tanda petik dua (“ ….”). Dalam bahasa C string merupakan larik atau array dari tipe data char, sedangkan dalam bahasa java string merupakan tipe data referensi atau sebuah objek. Contoh dibawah merupakan deklarasi string dalam C:

```
char tek [ ] = " C++ " ;
char kata[ ] = {'C', '+', '+'};
```

untuk Contoh penulisan deklarasi string dalam C adalah sebagai berikut:
Program 3.8

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
        char huruf[] = "Tipe Data String pada C++";
        cout << huruf;
        getch();
        return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Tipe Data String pada C++
```

3.5.4. Tipe Data Bilangan Bulat

Tipe data ini digunakan untuk data-data angka yang tidak mengandung angka di belakang koma (int) atau digunakan untuk menyatakan bilangan bulat. Perubahan tanda bilangan pada bilangan bulat dapat diset dalam dua tipe, yaitu: bilangan bulat bertanda (signed integer), yang merupakan bilangan bulat yang memiliki range dari bilangan negatip sampai positip dan bilangan bulat tak bertanda (unsigned integer), yang merupakan bilangan bulat yang hanya memiliki range nilai positip saja.

Tipe data yang termasuk ke dalam bilangan bulat adalah: yang pertama, char atau signed char dan unsigned char atau byte dalam java dan pascal. Rentang nilai signed char mulai -128 sampai 127. Kedua, rentang nilai unsigned char mulai 0 sampai 255, short int atau signed short int dan unsigned short int. Rentang nilai signed short int mulai - 32.768 sampai 32.767. Rentang nilai unsigned short int mulai 0 sampai 65.535. Ketiga adalah int atau signed int dan unsigned int. Rentang nilai signed int mulai -32.768 sampai 32.767.

Rentang nilai unsigned int mulai 0 sampai 65.535, dan keempat adalah long int atau signed long int dan unsigned long int. Rentang nilai signed long int mulai -21474836478 sampai 2147483647. Rentang nilai unsigned long int mulai 0 sampai 4294967295. Rentang di atas untuk tipe data bilangan bulat dalam 16 bit.
Contoh pendeklarasian bilangan int adalah sebagai berikut:

```
int nilai;
int total ;
int harga = 30000;
```

```
int nilai, total ;
int harga;
harga = 30000;
```

3.5.5. Tipe Data Bilangan Real atau Pecahan

Tipe ini merepresentasikan datadata bilangan yang mengandung angka di belakang koma atau menyatakan bilangan pecahan, maupun eksponensial. Tipe data yang termasuk ke dalam kategori ini adalah: float dan double. Contoh deklarasi float dan double adalah sebagai berikut:

```
float nilai;
double beta;
```

Semua bilangan pecahan atau desimal dalam Java tanpa diakhiri huruf **f** akan dianggap sebagai double. Sedangkan bilangan yang ingin dikategorikan sebagai float harus diakhiri dengan huruf **F**. Misalnya : 4.22F atau 2.314f. Sedangkan untuk bilangan double, bisa menambah dengan huruf D, karena secara default bilangan dengan koma atau pecahan atau desimal akan dianggap sebagai double. Pehatikan contoh program berikut ini:
Program 3.9

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
Tipe data dan Operator Bahasa Pemrorgraman
2008 | Direktorat Pembinaan SMK 107
short int ssintmin =  - 32768, ssintmak = 32767;
unsigned short int usintmak = 65535;
int intmin =  - 32768, intmak = 32767;
unsigned int uintmak = 65535;
long int slintmin =  - 2147483648, slintmak = 2147483647;
unsigned long int ulintmak = 4294967295;
cout <<"\nRange signed short int :"<< ssintmin<< ssintmak;
cout <<"\nRange unsigned short int :"<< usintmak;
cout <<"\nRange signed int : "<< intmin<< intmak;
cout <<"\nRange unsigned int :"<< uintmak;
cout <<"\nRange signed long int :"<< slintmin<< slintmak;
cout <<"\nRange unsigned long int :"<< ulintmak;
getch();
```

Keluaran program diatas adalah sebagai berikut:

```
Range signed short int          :  - 32768 s/d 32767
Range unsigned short int        : 0 s/d 65535
Range signed int                :  - 32768 s/d 32767
Range unsigned int              : 0 s/d 65535
Range signed long int           :  - 2147483648 s/d 2147483647
Range signed long int           : 0 s/d 4294967295
```

Program 3.10 Perhatikan juga contoh program dibawah ini:

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
float floatmin = 3.4E - 38, floatmak = 3.4E+38;
double doublemin = 1.7E - 308, doublemak = 1.7E+308;
long double ldoublemin = 3.4E - 4932 , ldoublemak = 1.1E+4932;
cout <<"Range float : \n"<< floatmin<< floatmak;
cout <<"Range double : \n"<< doublemin<< doublemak;
cout <<"Range long double : \n"<< ldoublemin<< ldoublemak;
getch();
return 0 ;
}
```

Keluaran program adalah sebagai berikut:

```
Range float : 0.000000 s/d 339999995214436425000000000000.0
Range double : 0.000000E+00 s/d 1.700000E=300
Range long doble : 0 s/d 1.1E+4932
```

3.5.6. Tipe Data Logika

Tipe data logika digunakan untuk merepresentasikan data-data yang mengandung dua buah nilai logika boolean, yaitu: nilai 1 dan 0 atau sering juga disebut sebagai nilai true dan false (benar dan salah). Dalam bahasa C++ tidak ada data tipe boolean. Perhatikan contoh deklarasi Boolean berikut ini:

```
Boolean kondisi;
Boolean kondisi_awal = true;
```

Pehatikan contoh program berikut ini:
Program 3.11

```
include <conio.h>
#include <iostream>
using namespace std;
main()
{
enum boolean {FALSE, TRUE};
enum boolean kondisi;
kondisi = TRUE;
if (kondisi)
cout <<"Kondisi : "<< kondisi;
else
{
kondisi = FALSE;
cout <<"Kondisi : "<< kondisi;
}
getch();
return 0 ;
}
```

Keluaran program adalah sebagai berikut:

```
Kondisi : 1
```

3.5.7. Enumerasi / ENUM

Enumerasi adalah serangkaian symbol berurutan yang menspesifikasikan konstanta bertipe integer. Dalam C++ tidak terdapat tipe Bolean, sehingga untuk merepresentasikan TRUE dengan nilai integer bukan nol (1, 2, dst), sedangkan FALSE dengan nilai nol (0). Perhatikan contoh deklarasi
enum:

```
enum boolean { false, true } ; atau
enum boolean { salah = 0, benar = 1 } ;
```

3.5.8. Tipe Data Void

Void menyatakan tipe kosong dan digunakan untuk: pertaman untuk mendeklarasikan fungsi yang tidak mengembalikan nilai apapun. Serta fungsi kedua adalah untuk mendeklarasikan fungsi yang tidak menerima parameter apapun. Pada tipe data ini jika diawali dengan operator *, menyatakan penunjuk terhadap sembarang tipe data.

Perhatikan contoh deklarasi void seperti berikut ini:

```
void cctrputs (char*,int );
```

atau ditulis:

```
main (void) ;
```

atau dapat juga ditulis

```
void* action ;
int ivalue = 100 ;
action = &ivalue ;
```

3.5.9. Tipe Data Pointer

Pointer adalah variable yang berisi nilai alamat suatu lokasi memori tertentu. Deklarasi penunjuk dilakukan dengan menspesifikasikan *, sebelum nama varibel/konstanta.

Contoh deklarasi pointer adalah sebagai berikut:

```
char *p;
```

untuk lebih jelasnya mengenai tipe data pointer akan dijelaskan dalam bab pointer

3.5.10. Tipe Data Larik / array

Array adalah sekelompok data bertipe sama yang menduduki lokasi memori yang berurutan. Jumlah elemen array dinyatakan dengan cara mengapit jumlah yang di maksud dengan tanda ‘ […] ‘. Cara penulisan sintaks tipe data larik adalah sebagai berikut :

```
<tipedata> <namavariabelArray> <[
jumlahelemen ]> ;
```

Misalnya penulisan tipe char dengan array adalah sebagai berikut:

```
char kata[5];
```

Untuk menyatakan array berdimensi lebih dari 1 (satu), maka ditambahkan tanda ‘[…]’ sebanyak dimensi yang diinginkan. Misalnya jika ingin mendeklarasikan array 2 dimensi adalah sebagai berikut:

```
int matrix [2][3] ;
```

Untuk lebih jelasnya mengenai tipe data array akan dijelaskan dalam bab selanjutnya yang membahas array.

3.5.11. Tipe Data Struct, Union

Tipe data ini digunakan untuk mendeklarasikan sekelompok data yang memiliki tipe yang berlainan. **struct**: elemennya berada dilokasi memori yang berbeda, dan **union**: elemennya ada dilokasi memori yang sama. Perhatikan potongan program dibawah ini:

```
union namaunion
{
Tipeanggota1 namaAnggota1 ;
Tipeanggota2 namaAnggota2 ;
…….
};
```

Atau dapat juga ditulis seperti dibawah ini:

```
union
{
        Tipeanggota1 namaAnggota1 ;
        Tipeanggota2 namaAnggota2 ;
        …….
}
namaunion;
struct namaStruktur
{
        Tipeanggota1 namaAnggota1 ;
        Tipeanggota2 namaAnggota2 ;
        …….
};
```

Atau dapat juga ditulis

```
struct
{
Tipeanggota1 namaAnggota1 ;
Tipeanggota2 namaAnggota2 ;
…….
}
namaStruktur ;
```

3.5.12. Tipe Data typedef

Tipe data typedef digunakan untuk menamakan suatu tipe data dengan pengenal yang lebih berarti atau mudah diingat. Sintaks penulisan program tipe data ini adalah sebagai berikut:

```
typedef tipedata namabaru;
```

Perhatikan contoh penulisan program dibawah ini :

```
typedef unsigned char byte;
```

3.6. Operator Bahasa C++

Bahasa C++ menyediakan beberapa operator untuk memanipulasi data. Secara umum, terdapat tiga jenis operator: unary, binary dan ternary. Istilah tersebut mencerminkan jumlah operands operator yang dibutuhkan.

Operator unary hanya memerlukan satu operand. Misalnya, mempertimbangkan ekspresi berikut: -5. Dalam contoh diatas perlu dipahami bahwa mewakili angka lima bernilai negatif. Konstanta 5 adalah diawali dengan tanda minus. Tanda minus, bila digunakan dengan cara seperti ini, yang disebut penyangkalan operator. Karena hanya memerlukan satu operand, hal tersebu merupakan operator unary. Operator binary bekerja dengan dua operand.

Tugas operator ini biasanya pada operasi aritmetik yang hal tersebut sudah sangat umum dalam bahasa pemrograman. Tabel dibawah menunjukkan operator arithmetic pada bahasa C++.

Tabel 3.6. Operator pada bahasa C++

OPERATOR	ARTI	TIPE	CONTOH
*	perkalian	Binary	tax = cost * rate;
/	Pembagian	Binary	salePrice = original / 2;

%	Penjumlahan	Binary	remainder = value % 3;
+	Penjumlahan	Binary	total = cost + tax
-	Pengurangan	binary	cost = total - tax;

Kebanyakan sebagai seorang programmer tidak akan asing dan selalu bekerja dengan operator. Operator Penambahan digunakan untuk menjumlahkan dua operand. Pada pernyataan berikut ini merupakan fungsi variabel jumlah yang diberi dengan nilai 12 adalah:

```
jumlah = 4 + 8;
```

operator pengurangan merupakan operasi dimana nilai penguranga berasal dari operand sebelah kiri dikurangi atau disubtracted operand sebelah kanan. Pernyataan dibawah ini akan memperoleh nilai 98 untuk suhu:

```
suhu = 112-14;
```

Pada sebuah operator perkalian akan mengalikan dua buah operand. Perhatian pernyataan berikut ini,dimana markup akan diisi nilai 3:

```
markup= 12 * 0,25;
```

pada operator pembagian hasil diperoleh dengan cara memlakukn bagi operand disebelah kiri dibagi dengan operand sebelah kanan. Dalam pernyataan berikutnya, variable poin akan memperoleh nilai 5:

```
pain=  100 / 20;
```

Dalam operasi pembagian integer ternyata tidak selalu memasukan hasil dari operasi pembagian tetapi yang diambil adalah sisa dri hasil bagi tersebut. Untuk lebih jelasnya perhatikan pernyataan berikut ini, dimana hasilnya adalah 2 atau nilai sisa sama dengan 2:

```
sisa = 17% 3;
```

Program 3.12 Perhatikan contoh program dibawah ini:

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
        double regHours = 40.0,
        otHours = 10,
        regPayRate = 18.25,
        otPayRate = 27.78,
        regWages,
        otWages,
        totalWages;
        regWages = regPayRate * regHours;
        otWages = otPayRate * otHours;
        totalWages = regWages + otWages;
        cout << "Upah untuk minggu ini adalah Rp. " << totalWages << endl;
        getch();
        return 0 ;
}
```

Keluaran Program diatas adalah

```
Upah untuk minggu ini adalah Rp. 1007.8
```

3.7. Operator Unary

Operator Unary

Operator Unary merupakan operator yang hanya memiliki atau melibatkan sebuah operand saja. Terdapat beberapa operator unary, seperti terlihat pada tabel berikut ini:

Tabel 3.7. Operator Unary

OPERATOR	ARTI	LETAK	CONTOH	EQUIVALEN
-	Unary minus	Sebelum operator	- A + - B * C	A + (- B) * C
++	Peningkatan dengan Penambahan nilai 1	Sebelum dan sesudah	A++	A = A + 1
--	Penurunan dengan Pengurang-an nilai 1	Sebelum dan Sesudah	A--	A = A – 1
Sizeof	Ukuran dari operand dalam byte	Sebelum	sizeof(I)	-
!	Unary NOT	Sebelum	!A	-
~	Bitwise NOT	Sebelum	~A	-
&	Menghasilkan alamat memori operand	Sebelum	&A	-
*	Menghasil-kan nilai dari pointer	Sebelum	*A	-

Dalam operator unary ada yang perlu diperhatikan ketika menulis program yaitu: bahwa operator peningkatan ++ dan penurunan -- jika diletakkan sebelum atau sesudah operand
terdapat perbedaan.

3.7.1. Operator Increment dan Decrement

Suatu variable yang nilainya selalu bertambah satu, seperti pada sebuah variable pencacah naik (up counter) dengan rumus: hitung = hitung + 1 dapat ditulis dengan hitung += 1 dan secara singkat ditulis hitung++ atau ++hitung. Notasi ++ disebut operator penambah (increment) dan pada sebuah variable pencacah turun (down counter) dengan rumus: hitung = hitung - 1 dapat ditulis dengan hitung -= 1 dan secara singkat ditulis hitung- - atau --hitung. Notasi – disebut operator pengurang (decrement).

Program dibawah merupakan operasi hitung perulangan sebanyak loop kali yang

dimulai dari 0 naik ke satu, dua dan seterusnya. Operasi dilakukan dengan melakukan
penambahan satu keatas
(increment). Perhatikan program
dibawah ini:
Program 3.13

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
int hitung = 0, loop;
loop = ++hitung;
cout <<"Loop = %d, Hitung = "<< loop<< hitung;
loop = hitung++;
cout <<"Loop = %d, Hitung = "<< loop<< hitung;
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Loop = 1, Hitung = 1
Loop = 1, Hitung = 2
```

Program dibawah merupakan operasi hitung perulangan sebanyak loop kali yang dimulai dari 10 turun ke sembilan, delapan dan seterusnya. Setiap operasi looping dilakukan dengan melakukan pengurangan satu kebawah. Perhatikan contoh program dibawah ini:

Program 3.14

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
int hitung = 10, loop;
loop =  -  - hitung;
cout <<"Loop = , Hitung = "<< loop<< hitung;
loop = hitung -  - ;
cout <<"Loop = , Hitung = "<< loop<< hitung;
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:
Loop = 9, Hitung = 9
Loop = 9, Hitung = 8

3.7.2. Operator sizeof

Operator akan menghasilkan
ukuran dari suatu variable atau tipe
pada saat dikompilasi. Ukuran ini
digunakan untuk mengetahui tipe
data apakah dan berapa ukuran data
yang ada didalam sebuah variable
tersebut. Perhatikan potongan
program dibawah ini:

```
sizeof(tipe data);
sizeof(char);
sizeof(int);
```

program dibawah ini akan melakukan
pengukuran terhadap variable tipe
data dan kemudian hasilnya akan
dimunculkan setelah dilakukan
kompilasi. Perhatikan program
dibawah ini:

Program 3.15

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
cout <<"Size of char = "<< sizeof(char);
```



```
cout <<"Size of short int = \n"<< sizeof(short int);
cout <<"Size of int = \n"<< sizeof(int);
cout <<"Size of long int = \n"<< sizeof(long int);
cout <<"Size of float = \n"<< sizeof(float);
cout <<"Size of double = \n"<< sizeof(double);
cout <<"Size of long double = \n"<< sizeof(long double);
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Size of char = 1 byte
Size of sort int = 2 byte
Size of int = 2 byte
Size of long int = 4 byte
Size of float = 4 byte
Size of double = 8 byte
Size of long double = 10 byte
```

Contoh program yang menggunakan
operator unary * (pointer) dan &
(alamat memori). Penjelasan dari

program tersebut dapat diuraikan
dalam bentuk program secara
langsung sehingga lebih mudah
dipahami.
Program 3.16

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
int x = 67, y; /* var pointer menunjuk ke data yg bertipe int */
int *px; // px diisi dengan alamat dari var x
px = &x; // y diisi dengan nilai yang ditunjuk oleh px
y = *px;
cout <<"\nAlamat x = "<< &x;
cout <<"\nIsi px = "<< px;
cout <<"\nIsi x = "<< x;
cout <<"\nNilai yang ditunjuk oleh px = "<< *px;
cout <<"\nNilai y = "<< y;
getch();
return 0 ;
}
```


Keluaran program diatas adalah sebagai berikut:

```
Alamat x = 2A6F : 223A
Isi px = 2A6F : 223A
Isi x = 67
Nilai yang ditunjuk oleh px = 67
Nilai y = 67
```

Adapun penjelasan mengenai pointer program diatas adalah sebagai berikut :

```
Alamat Memori Isi memori Variabel
…… …… …….
2A6F:2232 2A6F:223A px
…… …… …….
2A6F:2238 2A6F:223A y
…… …… …….
2A6F:223A 67 x
…… …… …….
```

Perhatikan program Operator
Unary & dan * serta gambar diatas.
Variabel x dan y merupakan suatu
lambang dari sebuah daerah di
memori utama komputer. Artinya x
dan y masing-masing sebenarnya
adalah suatu alamat memori. Ketika

variabel tersebut dideklarasikan sebagaimana dibawah ini:

int x = 67, y;

maka variabel x sama dengan alamat memori misalnya 2A6F:223A dialokasikan dan diisi data 67, variabel y sama dengan alamat memori misalnya 2A6F:2238 dialokasikan. Jadi x berisi 67 atau alamat memori 2A6F:223A berisi data 67.

Variabel tersebut bertipe integer maka akan menempati lokasi memori sebesar 16 bit. Alamat-alamat memori tersebut dipergunakan selama fungsi main dijalankan.

Deklarasi:

int *px;

berarti variabel px sama dengan alamat memori misalnya 2A6F:2232 dialokasikan dengan ukuran memori sebesar 16 bit (ukuran tipe integer).

Perintah penugasan:

px = &x;

berarti variabel px (alamat memori 2A6F:2232) diisi data 2A6F:223A (alamat dari variabel x).

Jadi isi px = 2A6F:223A, Isi px ini merupakan alamat dari variabel x, perhatikan gambar diatas agar lebih jelas. Perintah penugasan selanjutnya adalah:

y = *px;

berarti variabel y (alamat memori 2A6F:2238) diisi data alamat memori variable px (2A6F:2232). Alamat tersebut berisi data 67. Suatu alamat memori yang berisi data alamat memori disebut pointer.

3.8. Operator Binary

Operator binary adalah operator yang melibatkan atau dikenakan pada dua buah operand. Dibawah ini merupakan tergolong dalam operator

binary adalah adalah sebagaimana dijelaskan dibawah ini:

3.8.1. Operator Aritmatika

Digunakan untuk mengoperasikan data-data numerik, seperti perkalian, pembagian, sisa hasil bagi, penjumlahan, dan pengurangan. Dalam proses aritmatika tersebut, pengerjaan operasi tergantung dari tingkat valensi operator-operator yang terlibat. Perkalian memiliki valensi tertinggi, kemudian dilanjutkan dengan sisa pembagian, pembagian, sedangkan penjumlahan dan pengurangan mempunyai valensi yang terendah. jenis operator aritmatika, yaitu :

Tabel 3.8. Operator aritmetika

OPERATOR ARTI CONTOH

* Kali a * b a dikalikan dengan b
3 * 2 = 6

% Modulo atau sisa pembagian bulat
a%b sisa hasil pembagian bulat a dibagi dengan b
3 % 2 = 1

/ Bagi a / b a dibagi dengan b
3 / 2 = 1 untuk tipe data integer
3 / 2 = 1.5 ntuk tipe data float

+ Plus atau tambah a + b a ditambah dengan b
3 + 2 = 5

- Minus atau kurang a – b a dikurangi b
3 – 2 = 1

Program dibawah ini merupakan operasi-operasi aritmetika dan bertujuan untuk melakukan operasi pembagian, modulo, serta operasi kombinasi aritmatika pada bilangan bulat. Hasil operasi tersebut kemudian ditampilkan dilayar monitor. Untuk lebih jelasnya perhatikan program dibawah ini:

Program 3.17

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

```
using namespace std;
main()
{
/*operasi aritmatika
dengan bilangan bulat */
int v, w, x, y, z;
v = 100;
w = 3;
x = v / w;
y = v % w;
z = v * w  -  w + v % v / w;
cout <<"Operasi Aritmatika pada Bilangan Bulat\n";
cout <<"X = \n"<< v<< w;
cout <<" = \n"<< x;
cout <<"Y = "<< v<< w;
cout <<" = "<< y;
cout <<"Z = \n"<< v<< w<< w<< v<< v<< w;
cout <<" = \n"<< z;
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Operasi Aritmetika pada bilangan Bulat
X = 100/3
= 33
Y = 100 MOD 3
= 1
Z = 100 * 3 – 3 + 100 mod 100/3
= 297
```

Operator % (modulo) hanya berlaku pada tipe data integer, Perhatikan program operasi aritmatika dengan bilangan real dibawah ini:

Program 3.18

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
float a, b, c, d, e;
```



```
a = 100.0;
b = 3.0;
c = a / b;
d = 100 % 3;
e = a * b  -  b + 100 % 100 / b;
```

```
cout <<"\nOperasi Aritmatika pada Bilangan Real\n\n";
cout <<"C = \n"<< a<< b;
cout <<" = \n"<< c;
cout <<"D = \n"<< a<< b;
cout <<" = \n"<< d;
cout <<"E = \n"<< a<< b<< b<< a<< a<< b;
cout <<" = \n"<< e;
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Operasi Aritmetika pada bilangan Real
C = 100.000000 / 3.000000
= 33.333332
D = 100.000000 mod 3.000000
= 1.000000
E = 100.00000 * 3.00000 – 3.00000 + 100.000000 / 3.000000
= 297.000000
```

Dari contoh program diatas dapat dilihat dengan jelas bahwa Operator / (pembagian) dapat berfungsi sebagai pembagian bulat (div) atau pembagian real. Hal ini tergantung pada tipe data yang dipergunakan. Tingkat pengerjaan operasi dari operator aritmatika adalah: * (perkalian), % (modulo), / (pembagian), + (penjumlahan), dan – (pengurangan).

Perhatikan contoh mengenai hal tersebut diatas dapat dilihat pada program dibawah ini:

```
z = 100 * 3  -  3 + 100 % 100 / 3
```

Proses penyelesaian dalam program adalah sama halnya dengan ketika menyelesaikan persamaan aritmatika secara ditulis sebagai berikut ini:

```
z = ((100 * 3) – (3 + ((100 % 100) / 3)))
= ( 300 – (3 + ((100 % 100) / 3)))
= ( 300 – (3 + ( 0 / 3 )))
= ( 300 – (3 + 0 ))
= ( 300 – 3 )
= 297
```

3.8.2. Operator Relasional

Operator relasi digunakan untuk membandingkan hubungan antara dua buah operand (sebuah nilai atau variable) atau digunakan untuk mewakili sebuah nilai logika (nilai boolean), dari suatu persamaan atau nilai. Jenis-jenis operator relasi,sepeti terlihat pada tabel dibawah ini:

Tabel 3.9. Operator relasi

OPERATOR ARTI CONTOH

< Kurang dari x < y Apakah x kurang dari y

<= Kurang dari sama dengan x <= y Apakah x kurang dari sama dengan y

> Lebih dari x > y Apakah x lebih dari y

>= Lebih dari sama dengan x >= y Apakah x lebih dari sama dengan y

== Sama dengan x == y Apakah x sama dengan

!= Tidah sama dengan x != y Apakah x tidak sama dengan y

3.8.3. Operator logika

Operator logika digunakan untuk membandingkan logika hasil dari operator-operator relasi atau digunakan untuk mengoperasikan operand (konstanta, variabel, atau suatu ekspresi) secara logis.

Operator logika ada tiga macam yaitu operator AND, OR dan operator NOT. Untuk lebih jelasnya perhatikan dalam tabel berikut:

Tabel 3.10. Operator logika

OPERATOR ARTI CONTOH

&& AND

a b a && b

0 0 0

0 1 0

1 0 0

1 1 1

(2<1) && (3<1) = 0

0 && 0 = 0

(2<4) && (3<1) = 0

1 && 0 = 0

|| OR

a b a || b

0 0 0

0 1 1

1 0 1

1 1 1

(2<1) || (4<1) = 0
0 || 0 = 0
(2<4) || (3<1) = 1
1 || 0 = 0
! NOT
a !a !(2>3) = 1
0 1 !0 = 1
1 0
Program Operasi logika pada Operator Binary Logika

Program 3.19

```
include <conio.h>
#include <iostream>
using namespace std;
main()
{
int x, x1, x2, y, y1, y2, z, z1, z2, a, b, c;
a = 125;
b = 100;
c = 25;
x1 = a < b;
x2 = a > b;
y1 = c <= b;
y2 = c >= b;
z1 = a == c;
z2 = a != c;
x = x1 && x2;
y = y1 || y2;
z = !z1;
cout <<"\nA = "<< a<< b<< c ;
cout <<"\nX1 = " ;
cout <<"\n = "<< a<< b ;
cout <<"\n = "<< x1 ;
cout <<"\nX2 = " ;
cout <<"\n = "<< a<< b ;
cout <<"\n = "<< x2 ;
cout <<"\nY1 = " ;
cout <<"\n = "<< c<< b ;
cout <<"\n = "<< y1 ;
cout <<"\nY2 = " ;
cout <<"\n = "<< c<< b ;
cout <<"\n = "<< y2 ;
cout <<"\nZ1 = " ;
cout <<"\n = "<< a<< c ;
cout <<"\n = "<< z1 ;
```

```
cout <<"\nZ2 = " ;
cout <<"\n = "<< a<< b ;
cout <<"\n = "<< z2 ;
cout <<"\nX = " ;
cout <<" \n = "<< x1<< x2 ;
```

```
cout <<" \n = "<< x ;
cout <<"Y\n = " ;
cout <<"\n = "<< y1<< y2 ;
cout <<"\n = "<< y ;
cout <<"Z\n = " ;
cout <<"\n = "<< z1 ;
cout <<"\n = "<< z ;
getch();
return 0 ;
}
```
Keluaran program diatas adalah sebagai berikut:

```
A = 125
B = 100
C = 25
X1 = a < b
= 125 < 100
= 0
X2 = a > b
= 125 > 100
= 1
V1 = C <= B
= 25 <= 100
= 1
V2 = C >= B
= 25 <= 100
= 0
Z1 = A != C
= 125 != 25
= 0
Z2 = A == B
= 125 == 100
= 1
X = X1 && X2
= 0 && 1
= 0
Y = Y1 || Y2
= 1 || 0
= 1
Z = !Z1
```

= !0
= 1

3.8.4. Operator Bitwise atau manipulasi bit

Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori. Semua operator bitwise hanya bisa dikenakan pada operand bertipe integer atau karakter. Operator bitwise sebagaimana terlihat dalam tabel dibawah ini:

Tabel 3.11. Operator bitwise

OPERATOR	ARTI	CONTOH
<<	Pergeseran bit ke kiri	9 << 2
>>	Pergeseran bit ke kanan	9 >> 2
&	Bitwise AND	9 & 2
^	Bitwise XOR (Exclusive OR)	9 ^ 2
|	Bitwise OR	9 | 2
~	Bitwise NOT	~ 9

Perhatikan contoh operasi bitwise XOR antara data 9D dengan 2D maka akan menghasilkan data 11D

9D = 00001001B di XOR dengan
2D = 00000010B
11D = 00001011B <= hasil XOR

Atau contoh lain misalnya antara data 9D dengan 7D maka akan dihasilkan data 14D seperti dibawah ini:

9D = 00001001B di XOR dengan
7D = 00000111B
14D = 00001110B <= hasil XOR

Tabel 3.12. Kebenaran XOR

A	B	A XOR B
0	0	0
1	0	1
0	1	1
1	1	0

Pada tabel XOR tersebut diatas adalah: data keluaran akan bernilai 0 jika data masukannya bernilai sama baik 0 semua atau 1 semua. Dalam operasi aritmetika sebuah data 9D misalnya akan digeser kekanan atau

kekiri untukan operasi baik OR, AND,

OR maupun NOT. Perhatikan contoh
pergeseran data 9D yang digeser
kekiri maka nilai akan berubah.
Perubahan tersebut akan
mewakili sebuah operasi perkalian,
pembagian atau operasi yang lain.
Perhatikan contoh pergeseran data
9D dibawah ini:
9D = 00001001B
digeser kekiri:
1x => 00010010B = 18D
2x => 00100100B = 36D
Kalau digeser kekanan:
1x => 00000100B = 4D
2x => 00000010B = 2D
9D = 00001001B di AND dengan
2D = 00000010B
0D = 00000000B <= hasil AND
9D = 00001001B di OR dengan
2D = 00000010B
11D = 00001011B <= hasil OR
9D = 00001001B di NOT menjadi:
~9D = 11110110B = - 10D model komplemen 2
Selain hasil tersebut diatas,
dapat dibuktikan dengan cara
melakukan komplemen kedua.
Komplemen satu yaitu dengan
melakukan konversi data 0 menjadi 1
atau sebaliknya, sedangkan
komplemen kedua yaitu dengan
melakukan penambahan data 01
pada hasil komplemen pertama.
Untuk lebih jelasnya perhatikan
contoh dibawah ini:
10D = 00001010B
- 10D = 11110101B model komplemen 1
= 11110110B model komplemen 2
Untuk menyatakan bilangan negatip
dalam bilangan biner dipakai model
komplemen. Model komplemen 2
diperoleh dengan cara model
komplemen 1 ditambah 1.
Program 3.20. Operasi bit pada Operator Bitwise

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

```
using namespace std;
main()
{
int u, v, w, x, y, z;
u = 9 << 2;
v = 9 >> 2;
w = 9 & 2;
x = 9 ^ 2;
y = 9 | 2;
z = ~9;
cout <<"U = 9 << 2\n";
cout <<" = %d\n", u;
cout <<"V = 9 >> 2\n";
cout <<" = %d\n", v;
cout <<"W = 9 & 2\n";
cout <<" = %d\n", w;
cout <<"X = 9 ^ 2\n";
cout <<" = %d\n", x;
cout <<"Y = 9 | 2\n";
cout <<" = %d\n", y;
cout <<"Z = ~9\n";
cout <<" = %d\n", z;
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:

```
U = 9<< 2
= 36
V = 9>>2
= 2
W = 9 & 2
= 0
X = 9 ^ 2
= 11
Y = 9 | 2
= 11
Z = ~9
=  - 10
```

Contoh program diatas menunjukkan bahwa hasil operasi dari operator relasi atau operator logika hanya mempunyai satu nilai dari dua nilai kebenaran, yaitu :

- 0 = false = salah atau
- 1 = true = benar

3.8.5. Operator Penugasan dan Operator Kombinasi

Operator penugasan (*Assignment operator*) atau pemberi nilai berupa tanda sama dengan (“=”). Perhatikan contoh dibawah ini:

nilai = 80; Artinya : variable “nilai” diisi dengan 80

A = x * y; Artinya : variable “A” diisi dengan hasil perkalian antara x dan y.

Operator kombinasi merupakan gabungan antara dua operator tertentu, yaitu antara operator aritmatika atau operator bitwise dengan operator penugasan yang digunakan untuk memendekkan penulisan operasi penugasan, seperti berikut:

Tabel 3.13. Operator penugasan

OPERATOR ARTI CONTOH ARTINYA

*= Memberi nilai setelah nilai semula dikalikan x *= 5; x = x * 5;

/= Memberi nilai sisa bagi setelah nilai semula dibagi

x /= 5; x = x / 5;

%= Memberi nilai sisa bagi dari pembagian nilai semula

x %= 5; x = x % 5;

+= Memberi nilai setelah nilai semula ditambahkan x += 5; x = x + 5;

- = Memberi nilai setelah nilai semula dikurangkan x - = 5; x = x - 5;

<<= Memberi nilai dari pergeseran bit ke kiri x <<= 5; x = x << 5;

>>= Memberi nilai dari pergeseran bit ke kanan x >>= 5; x = x >> 5;

&= Memberi nilai hasil bitwise AND x &= 5; x = x & 5;

|= Memberi nilai hasil bitwise OR x |= 5; x = x | 5;

^= Memberi nilai hasil bitwise XOR x ^= 5; x = x ^ 5;

3.9. Operator Ternary

Operator ternary adalah operator yang melibatkan tiga buah operand.

Yang tergolong operator ini adalah:

Operator kondisi dengan symbol ?:

Bentuk ungkapan operator kondisi adalah :

<kondisi> ? <ungkapan_1> : <ungkapan_2>

Maksud dari ungkapan kondisi adalah: Jika kondisi bernilai benar, maka nilai ungkapan kondisi berupa nilai ungkapan_1 dan jika kondisi bernilai salah, maka nilai ungkapan kondisi berupa nilai ungkapan_2.

Perhatikan contoh dibawah ini:

maksim = nilai_awal > nilai_akhir ?
nilai_awal : nilai_akhir;
maksud dari contoh diatas dapat di
ungkapkan dalam bentuk kalimat
seperti ini:
jika nilai_awal = 80 dan nilai_akhir = 75, maka maksim = 80 (sama dengan nilai_awal)
jika nilai_awal = 75 dan nilai_akhir = 80, maka maksim = 80 (sama dengan nilai_akhir)
jika nilai_awal = 75 dan nilai_akhir = 75, maka maksim = 75 (sama dengan nilai_akhir)
Program 3.21. Operasi kondisi pada Operator Kondisi

```
#include <conio.h>
#include <iostream>
using namespace std;
main()
{
int a, b, c, d;
a = 80;
b = 75;
c = a > b ? a : b;
cout <<"Mencari nilai yang lebih tinggi\n\n";
cout <<"a = \n"<< a<< b;
cout <<"c = \n";
cout <<" = \n"<< a<< b<< a<< b;
cout <<" = \n\n"<< c;
a = 75;
b = 80;
c = a > b ? a : b;
cout <<"Mencari nilai yang lebih tinggi\n\n";
cout <<"a = \n"<< a<< b;
cout <<"c = \n";
cout <<" = \n"<< a<< b<< a<< b;
cout <<" = \n"<< c;
getch();
return 0 ;
}
```

Keluaran program diatas adalah sebagai berikut:
Mencari Nilai yang lebih tinggi
a = 80
b = 75

c = a > b ? a : b
= 80 > 75 ? 80 : 75
= 80
Mencari Nilai yang lebih tinggi
a = 75

b = 80
c = a > b ? : b
= 75 > 80 ? 75 : 80
= 80

3.10. Ungkapan (Ekspresi)

Ungkapan dapat berupa konstanta (untai/numerik), variabel dan nilai tunggal yang diperoleh dengan mengkombinasikan operand dan operator, seperti 5+4. Ungkapanungkapan dibagi menjadi empat kategori :

a. Ungkapan numerik
2 + 5 3 * 4 2 + 7 / 5
b. Ungkapan string
“ABCD” + “EFGH”
nim + nama
Satu-satunya operator yang berlaku pada ungkapan string hanyalah tanda +, yang berfungsi untuk menggabungkan dua untai.
c. Ungkapan relasi/hubungan
Tipe untai dapat juga menggunakan operator relasi seperti halnya dengan tipe numerik. Misalnya diketahui bahwa:
‘A’ lebih kecil dari ‘B’
d. Ungkapan logika
!A
((A>5) && (B=4))
((Nama = ”Agus”) || (nama = ”Doni”))
e. Program sederhana menggunakan Tipe data

Program 3.22. contoh penggunaan tipe data

```
#include <conio.h>
#include <iostream>
using namespace std;
typedef unsigned char byte;
main()
{
long int data1 = 546767226531;
int data2 = 2235641;
```

```
short int data3 = 714;
byte data4 = 34;
float ata6 = 1.733; // tipe data pecahan
double ata5 = 4.967; // tipe data pecahan
char data7 = 'C';
enum boolean {false, true};
enum boolean kondisi;
kondisi = true;
char data8[6];
data8[] = kondisi = = 1 ? "true":"false";
printf("Nilai Long : %ld\n"<< data1;
cout <<"Nilai Int : %d\n "<< data2;
cout <<"Nilai Short : %hd\n"<< data3;
cout <<"Nilai Byte : %d\n"<< data4;
cout <<"Nilai Double : %lf\n"<< data5;
cout <<"Nilai Float : %f\n"<< data6;
cout <<"Nilai Char : %c\n"<< data7;
cout <<"Nilai Boolean : %s\n"<< data8;
getch(;
return 0 ;
}
```

3.11. Soal Latihan

Jawablah soal latihan dibawah ini dengan baik dan benar.

1. Apa yang dimaksud dengan data
2. Sebutkan jenis-jenis data yang digunakan pada bahasa pemrograman
3. Apa yang dimaksud dengan konstanta
4. Buatlah sebuah program sederhana menggunakan variabel konstanta
5. Apa yang dimaksud dengan operator unary, binary dan ternary
6. Apa yang dimaksud dengan variabel
7. Sebutkan tipe variabel yang digunakan dalm bahasa c++
8. Buatlah program sederhana yang menggunakan operator unary, binary dan ternary

BAB 4
STRUKTUR PERULANGAN

4.1. Perulangan

Perulangan atau iterasi atau yang biasa disebut dengan “looping” adalah proses melakukan tindakan yang sama secara berulang-ulang

atau berkali-kali sampai batas yang telah ditentukan. Perulangan digunakan untuk menjalankan satu atau beberapa pernyataan sebanyak beberapa kali. Dengan kata lain, perulangan dipakai untuk menjalankan beberapa pernyataan dengan hanya menuliskan pernyataan tersebut satu kali. Hal ini banyak sekali dijumpai dalam pemrograman.

Perulangan proses dalam bahasa pemrograman ditangani dengan suatu mekanisme yang disebut loop. Dengan memakai loop, suatu proses yang berulang misalnya menampilkan angka 1 sampai 1000 atau tulisan yang sama sebanyak sepuluh kali di layar dapat diimplementasikan dengan kode program yang pendek.

Pada pemrograman proses perulangan dibagi menjadi 2 jenis, yaitu: Perulangan yang telah diketahui jumlah perulangannya sebelum perulangan tersebut di lakukan. Jenis perulangan ini dilakukan dengan penyataan for. Dan kedua adalah perulangan yang belum di ketahui jumlah perulangannya sebelum perulangan tersebut dilakukan. Perulangan jenis ini terdiri

4.1. Perulangan
4.2. Operator Increment dan Decrement
4.3. Ekspresi Matematika ++ dan --
4.4. Penghitung
4.5. Pernyataan FOR
4.6. Pernyataan NESTED - FOR
4.7. Pernyataan WHILE
4.8. Pernyataan NESTED-WHILE
4.9. Perulangan Do-WHILE
4.10. Pernyataan NESTED DO-WHILE
4.11. Perulangan Tidak Berhingga
4.12. Pernyataan Break
4.13. Pernyataan Continue
4.14. Pernyataan Goto

dari dua kategori, yaitu: kondisi perulangan diperiksa diawal perulangan. Jenis perulangan ini dilakukan dengan penyataan while. Kondisi perulangan diperiksa diakhir perulangan. Jenis perulangan ini dilakukan dengan penyataan dowhile. Struktur perulangan secara umum terdiri dari dua bagian:

- Kondisi perulangan, yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan perulangan.
- Badan (body) perulangan, yaitu bagian algoritma yang diulang.

4.2. Operator Increment dan Decrement

Sebelum jauh membahas perulangan, akan dipelajari dahulu mengenai operasi increment. Operator Increment digunakan untuk menaikan atau bisa juga untuk meningkatkan nilai dengan satu, sedangkan decrement digunakan untuk mengurangi nilai turun dengan satu. Kedua pernyataan berikut merupakan untuk menaikan variabel num dengan satu:

```
num = num + 1;
num += 1;
```

num di kurangi dengan satu dapat dilihat dari pernyataan berikut ini:

```
num = num  -  1;
num  - = 1;
```

dalam bahasa C++ menyediakan satu set operator unary sederhana yang dirancang hanya untuk menambah dan mengurangi sebuah variabel dengan 1. Operator increment adalah mengunakan kode ++. Sedangkan operator decrement adalah --. Pernyataan yang menggunakan operator ++ untuk menaikan variabel num adalah

sebagai berikut:
num++;
sedangkan pernyataan decrement
yang digunakan untuk mengurangi
variabel num adalah sebagai berikut:
num - - ;
dalam bahasa pemrograman baik
sebelum maupun sesudah bab ini
dibahas sering menggunakan operasi
kenaikan dan pengurangan yang
menggunakan dalam mode postfix,
mode postfik artinya operator
diletakkan setelah variabel. Operator
juga bekerja dalam mode prefik,
dimana operator ditempatkan
sebelum nama variabel:
+ + num;
- - num;
Kedua operator mode postfix dan
prefix diatas akan menambahkan 1
atau mengurangi dengan 1 pada
setiap operand. Program dibawah
menunjukkan operator increment dan
decrement.

Program 4.1
```
#include <iostream>
using namespace std;
int main()
{
int bigVal = 10, smallVal = 1;
cout << "vabesar adalah " << bigVal << " dan valkecil adalah " << smallVal << endl;
smallVal++;
bigVal - - ;
cout << "vabesar adalah " << bigVal<< " dan valkecil adalah " << smallVal << endl;
++smallVal;
- - bigVal;
cout << "vabesar adalah " << bigVal<< " dan valkecil adalah " << smallVal << endl;
return 0;
}
```
Keluaran programnya adalah :
vabesar adalah 10 dan valkecil adalah 1
vabesar adalah 9 dan valkecil adalah 2
vabesar adalah 8 dan valkecil adalah 3
4.3. Ekspresi Matematika ++ dan - -

Pada operator kenaikan dan pengurangan dapat juga digunakan pada variabel dalam ekspresi matematika. Perhatikan potongan program berikut ini:

```
a = 2;
b = 5;
c = a * b++;
cout << a << " " << b << " " << c;
```

dalam pernyataan c = a * b++, c adalah hasil dari perkalian a dan b, dimana 10. Merupakan sebuah variabel b yang ditambahkan dengan satu. Kemudian dengan menggunakan pernyataan cout maka hasil ditampilkan sebagai berikut:

2 6 10

Jika pernyataannya c tersebut diatas berubah maka dapat dibaca:

```
c = a * ++b;
```

variabel b akan ditambahkan dengan satu sebelum keduanya dikalikan dengan a. dalam kasus ini c merupakan hasil dari nilai 2 dikalikan 6, sehingga pernyataan cout akan menampilkan:

2 6 12

Dengan satu saja dapat membawa beberapa aksi dalam satu pernyataan menggunakan operator kenaikan dan pengurangan, hal tersebut juga tidak terlalu rumit untuk digunakan. Perhatikan potongan program dibawah ini:


```
a = 2;
b = 5;
c = ++(a * b); // Error!
```

pernyataan tersebut diatas merupakan pernyataan sederhana yang tidak bekerja karena adanya operator kenaikan dan pengurangan. Pada operator kenaikan dan pengurangan biasanya ada variabel operand, tetapi umumnya, sesuatu

yang ada di sebelah kiri dari operator yang disetujui.
Seperti sudah kita ketahui bahwa dalam bab ini, operator ++ dan -- digunakan sebagai kalimat penghubung. Sama seperti dalam ekspresi matematika, perbedaan antara mode postfix dan prefix sangat dekat. Perhatikan potongan program berikut ini:

```
x = 10 ;
if (x + +> 10)
cout << "x lebih besar daripada 10. \n";
```

Dua operasi yang terjadi dalam pernyataan IF tersebut diata adalah: (1) nilai yang diuji pada x untuk menentukan apakah nilainya lebih besar daripada 10, dan (2) x adalah diincremented. Karena kenaikan operator yang digunakan dalam mode postfix, Bandingan hal yang terjadi terlebih dahulu. Sejak 10 tidak lebih dari 10, pernyataan cout tidak akan dijalankan. Jika mode kenaikan operator berubah, sedangkan jika pernyataan akan membandingkan 11-10 dan pernyataan cout akan dijalankan.

4.4. Penghitung

Kadang-kadang penting bagi sebuah program untuk melacak jumlah Iterasi yang dilakukan dalam satu loop. Misalnya, Program dibawah menampilkan sebuah tabel yang terdiri dari angka 1 sampai dengan 10, jadi loop harus iterasi 10 kali. Program tersebut dibawah akan menampilkan angka 1 sampai dengan 10 dan kemudian dikuadratkan:

Program 4.2.

```
include <iostream>
using namespace std;
int main()
{
int num = 1; // inisialisasi penghitung
```

```
cout << " Angka Angka kuadrat\n";
cout << " - - - - - - - - - - - - - - - - - - - - - - - - - \n";
while (num <= 10)
{
cout << num << "\t\t" << (num * num) << endl;
num++; // penghitung Increment
```



```
}
return 0;
}
```

Keluaran programnya adalah sebagai berikut:

```
Angka Angka kuadrat
 - - - - - - - - - - - - - - - - - - - - - - - - -
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

Dalam Program diatas, variabel num, dimulai dari angka 1, kemudian diincrement pada setiap kali putaran loop. Ketika num mencapai 11 lingkaran berhenti. num digunakan sebagai counter variabel, yang berarti variabel tersebut secara teratur di increment pada setiap perulangan. Pada dasarnya, num terus menghitung jumlah lingkaran Iterasi yang telah dilakukan. Ketika variabel num digunakan sebagai pengendalian pada loop dan kapan akan keluar dari loop, disebut lingkaran kontrol variabel.

Dalam contoh program diatas, penambahan variabel num adalah pada lingkaran. Pendekatan lain adalah dengan menggabungkan laba operasi yang berhubungan dengan pengujian, seperti yang ditunjukkan dalam pada program dibawah ini.

Program 4.3

```
#include <iostream>
using namespace std;
int main()
{
int num = 0;
cout << "Angka Angka kuadrat\n";
cout << " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - \n";
while (num++ < 10)
cout << num << "\t\t" << (num * num) << endl;
return 0;
}
```


Keluaran programnya adalah sebagai berikut:

```
Angka Angka kuadrat
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

Perhatikan bahwa sekarang variabel num diinisialisasi ke 0, bukan 1, dan relatif menggunakan ekspresi <operator bukan <=. Hal ini karena cara operator untuk menaikan bekerja jika digabungkan dengan ekspresi penghubung.

Kenaikan operator yang digunakan dalam mode postfix, berarti pada variabel num menambah satu setelah berhubungan ujian. Ketika lingkaran pertama melaksanakan, num diatur kenilai 0, kemudian 0 dibandingkan dengan 10. Operator ++ kemudian akan menambahkan variabel num segera setelah dilakukan perbandingan. Bila pernyataan cout dieksekusi, num memiliki nilai 1.

num dibandingkan dengan 10, kemudian di - increment. Bila pernyataan cout dilaksanakan, num 1 bernilai lebih besar dari itu pengujian tersebut.

while (**num**++ < 10)
cout << num << "\t\t" << (num * num) << endl;

Di dalam looping, num selalu memiliki nilai lebih dari 1 dibanding sebelumnya nilai 10. Itulah mengapa penghubung operator menggunakan < bukan <=. Ketika variabel num bernilai 9 maka diuji, sehingga akan menjadi bernilai 10 pada pernyataan cout.

4.5. Pernyataan FOR

Jenis loop dalam bahasa C++ adalah loop FOR. FOR sangat ideal untuk situasi yang memerlukan penghitung karena ekpresinya sudah *built-in* (menjadi satu) dengan memperbarui variabel. Berikut ini adalah format perulangan FOR

for loop.

```
for (initialization; test; update)
{
pernyataan;
pernyataan;
// tempat banyaknya pernyataan
```

Bisa juga sebuah perulangan menggunakan pernyataan for ditulis dalam bentuk format potongan program sebagai berikut:

for (inisialisasi; syarat perulangan; pengubah nilai pencacah)
pernyataan;

Bisa juga ditulis seperti dibawah ini:

for (ungkapan1; ungkapan2; ungkapan3)
pernyataan;

Penjelasan mengenai format pernyataan FOR diatas adalah seperti dibawah ini:

- Ungkapan1 atau inisialisasi: digunakan untuk memberikan inisialisasi terhadap variabel pengendali loop.
- Ungkapan2 atau sebagai syarat

perulangan: dipakai sebagai pemegang kontrol terhadap pengulangan, karena bagian ini yang akan menentukan suatu kondisi perulangan untuk diteruskan atau dihentikan dari loop.
 Ungkapan3 atau pengubah nilai pencacah: dipakai sebagai pengatur perubahan nilai variabel pengendali loop. Perubahan nilai bisa kenaikan atau penurunan nilai pencacah

Ketiga ungkapan dalam FOR tersebut harus dipisahkan dengan titik koma (;). Pernyataan dalam for dapat berupa pernyataan tunggal maupun jamak (lebih dari satu). Jika pernyataannya berbentuk jamak, maka pernyataan-pernyataan tersebut harus diletakan didalam satu blok dengan memakai tanda kurung kurawal seperti berikut:

```
for (inisialisasi; syarat pengulangan; pengubah nilai pencacah)
{
pernyataan_1;
pernyataan_2;
……
Pernyataan_n;
}
```


Berikut ini adalah sebuah diagram alir pernyataan FOR adalah sebagai berikut:

Gambar 4.1. Diagram Alir Pernyataan FOR

Seperti perulangan dua lainnya, jika hanya ada satu pernyataan dalam tubuh perulangan, kotak pernyataan (brace) akan dapat diabaikan. Perulangan FOR mempunyai tiga ekspresi didalam tanda kurung, dipisahkan oleh semicolon. (Perhatikan tidak ada koma setelah ketiga ekspresi.)

Pertama adalah ekspresi initialization expression. Ekspresi ini biasanya digunakan untuk sebuah

initialize penghitung atau variabel yang harus memiliki nilai awal. Ini adalah tindakan yang pertama dilakukan oleh perulangan dan hanya dilakukan sekali.
Ekspresi yang kedua adalah test expression. Seperti ekspresi test dalam perulangan while dan dowhile, test expression mengendalikan pelaksanaan perulangan. Selama ini ungkapan itu benar, mka tubuh perulangan FOR akan mengulang lagi.
Ekpresi ketiga adalah ekspresi update expression. Ekspresi ini akan melaksanakan diakhir setiap perulangan. Biasanya, ia akan menambahkan sebuah penghitung atau variabel yang harus diubah pada setiap perulangan. Program dibawah adalah Program yang menggunakan FOR sebagai pengganti dari while loop.

pengubah nilai pencacah
loop++ atau loop = loop + 1
Pernyataan
Cetak bilangan
cout << loop << " ";
Inisialisasi
Loop = 1
Syarat
perulangan
Loop <= 10
Benar
Salah

Program dibawah menggunakan pernyataan FOR untuk menampilkan angka 1 sampai dengan 10 dan kemudian mengkuadratkannya. Untuk lebih jelasnya perhatikan program dibawah ini:
Program 4.4

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

```
using namespace std;
int main()
{
int num;
cout << "Angka Angka kuadrat \n";
cout    <<    " - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
_______________\n";
for (num = 1; num <= 10; num++)
cout << num << "\t\t" << (num * num) << endl;
getch ();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Angka Angka kuadrat
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```


Gambar dibawah menjelaskan mengenai mekanisme looping dalam perulangan untuk menjelaskan proses langkah-langkah program diatas.

Step 1. Melakukan inisialisasi ekspresi.
Step 2. Mengevaluasi tes ekspresi. Jika memang, lanjut ke Step 3.
Jika tidak, menghentikan loop
Step 4. Melakukan update ekspresi.
Kembali ke step 2.

```
for (num = 1; num <= 10; num++)
cout << num << "\t\t" << (num * num) << endl;
```

Step 3. Jalankan tubuh perulangan.

Gambar 4.2. Mekanisme Perulangan

Meskipun pada umumnya gaya pemrograman diatas dianggap jelek, tetapi satu atau beberapa kalimat perulangan dapat diabaikan.

Initialization ekspresi yang mungkin

dapat dihilangkan dari dalam kurung loop, jika sudah dilakukan atau jika tidak ada initialization diperlukan. Berikut dibawah ini adalah contoh program perulangan dalam melakukan pengaksesan sebelum perulangan:

```
int num = 1;
for ( ; num <= 10; num++)
cout << num << "\t\t" << (num * num) << endl;
```

Kita juga dapat mengabaikan pembaruan ekspresi, jika sedang dilakukan di tempat lain didalam perulangan atau jika tidak ada yang diperlukan. perulangan berikut ini untuk bekerja seperti loop while.

```
int num = 1;
for ( ; num <= 10; )
{
cout << num << "\t\t" << (num * num) << endl;
num++;
}
```


Kita bahkan dapat tidak menggunakan atau mengabaikan semua tiga ekspresi dari tanda kurung loop FOR. Jika hal tersebut diabaikan maka akan terjadi peringatan bahwa kita membiarkan test expression. Perhatikan contoh dibawah ini:

```
for ( ; ; )
cout << "Hello World\n";
```

Karena perulangan ini tidak memiliki cara untuk berhenti, maka program diatas akan menampilkan kata "Hello World \ n" selamanya atau sampai terjadi interupsi program.

Bentuk lain yang memperbarui Ekspresi dilakukan supaya kita tidak dibatasi dalam menggunakan pernyataan increment pada ekspresi update. Berikut ini adalah sebuah perulangan yang menampilkan

semua atau bahkan angka 2 hingga
100 dengan menambahkan 2 pada
setiap penghitung. Perhatikan
program dibawah ini:

```
for (number = 2; number <= 100; number += 2)
cout << number << endl;
```

program dibawah ini merupakan
program perulangan yang melakukan
hitungan turun mulai dari 10 turun
sampai nilai 0.

```
for (number = 10; number >= 0; number -  - )
cout << number << endl;
```

perulangan dibawah ini tidak memiliki
badan resmi. Gabungan kenaikan
operasi dan pernyataan cout
pembaruan ekspresi akan melakukan
semua pekerjaan pada setiap iterasi:

```
for (number = 1; number <= 10; cout << number++);
```

Jika perulangan memerlukan
melakukan lebih dari satu pernyataan
sebagai bagian dari inisialisasi,
pernyataan dipisahkan dengan tanda
koma. Program dibawah adalah versi
Program sebelumnya yang diubah
untuk memberitahu pengguna
mengenai angka penjualan selama
satu minggu.
Program dibawah ini akan
berlangsung setiap hari selama satu
minggu dan kemudian program akan
menghitung totalnya. Perhatikan
program berikut ini:

Program 4.5

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
const int NUM_DAYS = 7;
int count;
double total;
for (count = 1, total = 0.0; count <= NUM_DAYS; count++)
{
double sales;
```

```
cout << "Masukkan penjualan untuk hari " << count << ": ";
cin >> sales;
total += sales;
}
cout << fixed << showpoint << setprecision(2);
cout << "Total penjualan adalah $" << total << endl;
return 0;
}
```

Keluaran Program setelah memberikan masukan adalah sebagai berikut:

```
Masukkan penjualan untuk hari 1: 489.32[Enter]
Masukkan penjualan untuk hari 2: 421.65[Enter]
Masukkan penjualan untuk hari 3: 497.89[Enter]
Masukkan penjualan untuk hari 4: 532.37[Enter]
Masukkan penjualan untuk hari 5: 506.92[Enter]
Masukkan penjualan untuk hari 6: 489.01[Enter]
Masukkan penjualan untuk hari 7: 476.55[Enter]
Total penjualan adalah $ 3413.71
```

Dalam perulangan menggunakan FOR, penghitung diinisialisasi ke 1, maka total inisialisasi adalah 0,0. Kita dapat menempatkan lebih dari satu pernyataan untuk melakukan pembaharuan sebuah ekspresi. Perhatikan program dibawah ini:

```
double sales;
for (count = 1, total = 0.0; count <= NUM_DAYS; count++, total += sales)
{
cout << "Masukkan penjualan untuk hari " << count << ": ";
cin >> sales;
```



```
}
```

Dalam melakukan pembaruan ekspresi sebuah perulangan, penghitung di increment, maka nilai penjualan tersebut akan ditambahkan ke total pada akhir setiap perulangan. Perhatikan contoh mengenai program looping menggunakan FOR. Program dibawah ini digunakan untuk mencetak bilangan dari 1 hingga 10 secara naik.

Program 4.6

```
#include<conio.h>
#include <iostream.h>
using namespace std;
```

```
int main()
{
int loop;
for (loop = 1; loop <= 10; loop++)
cout << loop << " ";
getch();
return 0;
}
```

Keluaran program adalah sebagai berikut:

1 2 3 4 5 6 7 8 9 10

Pada program diatas terdapat

int loop;

tipe data variabel loop adalah integer

for (loop = 1; loop <= 10; loop++)

cout << loop << " ";

Perintah di atas akan menampilkan angka dari 1 dan sebuah spasi kosong sampai angka 10 dan spasi kosong secara horizontal atau berjajar. Proses tersebut dilakukan secara iterasi sepuluh kali yang diawali dengan nilai loop = 1, kenaikan nilai loop satu (loop++), dan berakhir sampai syarat terpenuhi loop <= 10, , yaitu: loop = 10. Perhatikan tanda loop++ yang berarti loop = loop + 1. Proses perulangan potongan program diatas adalah sebagai berikut:

Pada loop = 1 ditampilkan : 1 disambung dengan loop ke 2

Pada loop = 2 ditampilkan : 2 disambung dengan loop ke 3

……

……

……

Pada loop = 10 ditampilkan : 10

Pada loop = 11 proses iterasi selesai atau berhenti karena nilai ini sudah tidak memenuhi syarat yang ditentukan, yaitu: 11 <= 10.

Selanjutnya adalah:

getch();

meminta masukan sembarang tombol, perintah ini dimaksudkan untuk menahan hasil tampilan dilayar supaya tetap nampak dan akan menutup sampai adanya sembarang

tombol ditekan.
return 0;
untuk memberi nilai kembalian pada fungsi main. Pada program di atas, kenaikan terhadap variabel pengendali loop sebesar 1 (positif), yang dinyatakan dengan ungkapan:
loop ++
artinya ungkapan tersebut sama dengan:
loop = loop + 1
Kenaikan terhadap variabel pengendali loop bisa diatur jaraknya dengan mengatur stepnya. Misalnya untuk jarak kenaikkannya 2, maka dinyatakan dengan:
Loop += 2
Hal diatas sama artinya sama dengan program dibawah ini:
Loop = loop + 2
Pada contoh yang melibatkan pernyataan for diatas, kenaikan variabel pengendali loop berupa nilai positif. Sebenarnya kenaikan terhadap variabel pengendali loop dapat pula diatur supaya bernilai negatif, seperti dicontohkan pada program dibawah ini. Program dibawah akan mencetak bilangan dari 10 hingga 1 secara menurun.
Program 4.7

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop;
for (loop = 10; loop >= 1; loop -  - )
cout << loop << " ";
getch();
return 0;
}
```


Hasil keluaran program adalah sebagai berikut:
10 9 8 7 6 5 4 3 2 1

Program lainnya yang mennggunakan pernyataan FOR untuk menampilkan bilangan ganjil antara 1 hingga 10 adalah sebagai berikut:

Program 4.8

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop;
for (loop = 1; loop <= 10; loop+=2)
cout << loop << " ";
getch();
return 0;
}
```

Hasil keluaran program diatas adalah sebagai berikut:

1 3 5 7 9

Selain berupa angka, pencacah perulangan juga dapat berupa karakter. Perhatikan contoh program dibawah ini:

Program 4.9

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
char huruf;
for (huruf = 'A'; huruf <= 'Z'; huruf++)
cout << "Huruf abjad = " << huruf << "\n";
getch();
for (huruf = 'A'; huruf <= 'Z'; huruf+=13)
cout << "Huruf abjad = " << huruf << "\n";
getch();
for (huruf = 'z'; huruf >= 'a'; huruf -  - )
cout << "Huruf abjad = " << huruf << "\n";
getch();
```



```
for (huruf = 'z'; huruf >= 'a'; huruf - =8)
cout << "Huruf abjad = " << huruf << "\n";
getch();
return 0;
}
```

Hal yang perlu diperhatikan pada program diatas adalah potongan program seperti dibawah ini:

```
for (huruf = 'A'; huruf <= 'Z'; huruf++)
cout << "Huruf abjad = " << huruf <<
```

"\n";
Perintah di atas akan menampilkan
teks Huruf abjad = mulai dari A
sampai dengan Z. Perhatikan
perintah pada **huruf++.**
Huruf abjad = A
Huruf abjad = B
Huruf abjad = C
Huruf abjad = D
Huruf abjad = E
Huruf abjad = F
Huruf abjad = G
Huruf abjad = H
Huruf abjad = I
Huruf abjad = J
Huruf abjad = K
Huruf abjad = L
Huruf abjad = M
Huruf abjad = N
Huruf abjad = O
Huruf abjad = P
Huruf abjad = Q
Huruf abjad = R
Huruf abjad = S
Huruf abjad = T
Huruf abjad = U
Huruf abjad = V
Huruf abjad =W
Huruf abjad = X
Huruf abjad = Y
Huruf abjad = Z
Perhatikan potongan program
dibawah ini:
for (huruf = 'A'; huruf <= 'Z'; huruf+=13)
cout << "Huruf abjad = " <<
huruf << "\n";
Perintah diatas akan menampilkan
teks Huruf abjad = mulai dari A
sampai dengan Z dengan step 13,
maka yang muncul adalah:
Huruf abjad = A
Huruf abjad = N
Perhatikan pada instruksi **huruf+=13**
for (huruf = 'z'; huruf >= 'a'; huruf - -)
cout << "Huruf abjad = " << huruf <<
"\n";

Perintah tersebut akan menampilkan teks Huruf abjad = mulai dari z sampai dengan a. Perhatikan pada perintah **huruf--**

```
Huruf abjad = z
Huruf abjad = y
Huruf abjad = x
Huruf abjad = w
Huruf abjad = v
Huruf abjad = u
Huruf abjad = t
Huruf abjad = s
Huruf abjad = r
Huruf abjad = q
```



```
Huruf abjad = p
Huruf abjad = o
Huruf abjad = n
Huruf abjad = m
Huruf abjad = l
Huruf abjad = k
Huruf abjad = j
Huruf abjad = i
Huruf abjad = h
Huruf abjad = g
Huruf abjad = f
Huruf abjad = e
Huruf abjad = d
Huruf abjad = c
Huruf abjad = b
Huruf abjad = a
```

Perhatikan potongan program dibawah ini:

```
for (huruf = 'z'; huruf >= 'a'; huruf - =8)
cout << "Huruf abjad = " << huruf <<
"\n";
```

Perintah itu akan menampilkan teks Huruf abjad = mulai dari z sampai dengan a dengan step -8, maka yang muncul adalah:

```
Huruf abjad = z
Huruf abjad = r
Huruf abjad = j
Huruf abjad = b
```

Perhatikan instruksi huruf -=8.

Kadang-kadang dijumpai adanya pernyataan FOR yang tidak mengandung bagian ungkapan yang lengkap (beberapa ungkapan dikosongkan). Dengan cara ini, maka pernyataan adalah sebagai berikut:

```
for (loop = 1; loop <= 10; loop+=2)
cout << loop << " ";
```

Dari potongan program diatas dapat ditulis menjadi :

```
loop = 1;
for ( ; loop <= 10; )
{
cout << loop << " ";
loop += 2;
}
```

Pengosongan juga dilakukan pada ungkapan untuk menaikkan nilai variabel pengendali loop. Sebagai gantinya, di dalam tubuh loop diberikan pernyataan untuk menaikkan nilai variabel pengendali loop, yaitu berupa:

```
loop += 2;
```

Ungkapan yang tidak dihilangkan berupa loop <=10. Ungkapan ini tetap disertakan karena dipakai sebagai kondisi untuk keluar dari loop. Sebenarnya ungkapan yang dipakai sebagai kondisi keluar dari loop juga bisa dihilangkan, sehingga bentuknya menjadi:

```
for (;;)
pernyataan;
```

Lalu bagaimana cara keluar dari loop tersebut. Caranya adalah dengan menggunakan pernyataan yang dirancang khusus untuk keluar dari loop. Pernyataan itu adalah break yang akan dijelaskan dalam sub bahasan tersendiri. Pernyataan break digunakan untuk keluar dari satu blok loop for. Perhatikan program FOR untuk menampilkan warna seperti program berikut:

Program 4.10

```
#include<stdio.h>
#include<conio.h>
using namespace std;
int main(void)
{
int i = 0;
for (;;)
{
i++;
textattr(i + ((i+1) << 4));
cprintf("Warna atribut\r\n");
if (i==20) break;
}
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:

Pada contoh berikut merupakan penggunaan loop tanpa henti. Karena program memang dirancang untuk tidak berhenti maka untuk menghentikannya dilakukan dengan cara menekan tombol CTRL–PAUSE atau CTRL – BREAK. Perhatikan program FOR untuk menampilkan warna seperti berikut:

Program 4.11

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



```
using namespace std;
main()
{
int n = 0;
for(;;)
{
++n;
gotoxy(n, n);
textcolor(n);
cprintf("warna ke %d\n",n);
}
getch();
return 0;
```

}
Keluaran program adalah sebagai berikut:

4.6. Pernyataan NESTED - FOR

Pernyataaan nested for adalah suatu perulangan for didalam perulangan for dalam bentuk lain. Dalam mempelajari perulangan ini dituntut hrus teliti. Perulangan didalam perulangan sering kali masuk digunakan oleh program

Bentuk umum pernyataan Nested for adalah sebagai berikut:

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )
{
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah)
{
pernyataan;
}
}
```


Selain pernyataan diatas, nested For dapat juga ditulis seperti dibawah ini:

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )
{
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah)
{
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah)
{
……..
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah)
{
pernyataan;
}
…….
}
```

Didalam penggunaan NESTEDFOR, perulangan yang berada didalam terlebih dahulu harus dihitung sampai selesai, kemudian perulangan yang diluar diselesaikan terus sampai perulangan yang paling luar. Perhatikan contoh program dengan NESTED FOR pada operasi bilangan naik berikut ini:

Program 4.12

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

```
#include <iostream.h>
using namespace std;
int main()
{
int a, b;
for(a = 1; a <= 5; a++)
{
cout << "\n ";
for(b = a; b <= 5; b++)
cout << a << " ";
}
getch();
return 0;
}
```

Keluaran program diatas adalah sebai berikut:

1 1 1 1 1
2 2 2 2

3 3 3
4 4
5

4.7. Pernyataan WHILE

Bab sebelumnya telah memperkenalkan konsep statement kontrol, yang berupa aliran program langsung. Sebuah loop adalah struktur kontrol yang menyebabkan pernyataan atau kelompok untuk mengulang pernyataan. Bahasa C++ mempunyai tiga looping struktur kontrol: while loop, do-while loop, dan loop FOR.

Perbedaan antar masing-masing looping adalah bagaimana mereka melakukan kontrol pengulangannya. Selama melakukan loop ada dua bagian penting: (1) sebuah ekspresi yang diuji dengan nilai benar atau salah, dan (2) pernyataan atau blok yang berulang-ulang selama ekspresinya benar.

Perulangan dengan pernyataan **while** merupakan perulangan yang mirip dengan perulangan **for**. Perulangan for dipakai pada

perulangan yang sudah diketahui berapa kali akan dijalankan. Sedangkan yang belum diketahui berapakali akan diulangi maka digunakan while. Pernyataan while digunakan ketika kondisi perulangan diperiksa terlebih dahulu sebelum menjalankan pernyataan. Pada pernyataan while, disini pemeriksaan terhadap loop dilakukan di bagian awal (sebelum tubuh loop). Pernyataan while akan mengulang proses secara terus menerus sampai kondisi bernilai benar atau akan diulangi selama kondisi bernilai benar, jika kondisi bernilai salah maka perulangan (loop) selesai. Lebih jelasnya, bentuk pernyataan while adalah sebagai berikut:
Bentuk perulangan **while** dikendalikan oleh syarat/kondisi tertentu, yaitu perulangan akan terus dilaksanakan selama syarat/kondisi tersebut terpenuhi. Pernyataan dalam while akan dilaksanakan berulangkali selama syarat/kondisi bernilai benar. Jika syarat/kondisi bernilai salah badan perulangan tidak akan dilaksanakan, yang berarti perulangan selesai. Yang harus diperhatikan adalah perulangan harus berhenti. Perulangan yang tidak pernah berhenti menandakan bahwa logika dari algoritma tersebut salah.
Bentuk umum perulangan **while**, sebagai berikut :
while (syarat/kondisi)
pernyataan;
penjelasan mengenai hal tersebut diatas adalah bahwa syarat/kondisi merupakan ungkapan logika yang hanya bernilai benar atau salah, sehingga operator yang dipakai disini adalah operator relasi dan operator logika atau gabungan dari keduanya.

Untuk lebih jelasnya gambar diagram alir dari pernyataan while adalah sebagai berikut:

Gambar 4.3. Diagram pernyataan While

Pernyataan dalam **while** dapat berupa pernyataan tunggal maupun jamak (lebih dari satu). Jika pernyataannya berbentuk jamak, maka pernyataan-pernyataan tersebut harus diletakan didalam satu blok dengan memakai tanda kurung kurawal.

Bentuk umum perulangan **while**, dengan satu dengan lebih dari satu pernyataan, adalah seperti berikut :

```
while ( syarat )
{
Pernyataan;
Pernyataan;
}
```

Untuk lebih jelasnya pengenai pernyataan while, dibawah ini merupakan contoh program untuk menaik bilangan. Program lengkapnya adalah sebagai berikut:

Program 4.13

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop = 1;
while(loop <= 10)
cout << loop++ << " ";
getch();
return 0;
}
```

pernyataan
Kondisi
Benar
Salah

Keluaran program adalah sebagai berikut:

1 2 3 4 5 6 7 8 9 10
Program dibawah ini merupakan sebuah penerapan pernyataan while yang digunakan untuk melakukan perhitungan turun. Program dibawah ini akan menghitung bilangan turun mulai dari angka 10, menjadi 9, 8, 7 sampai angka 1. Perhatikan program dibawah ini:
Program 3.14

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop = 10;
while(loop >= 1)
cout << loop -  -  << " ";
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:
10 9 8 7 6 5 4 3 2 1
Sebuah program yang menggunakan pernyataan while juga bisa digunakan untuk menentukan bilangan ganjil. Program dibawah ini merupakan program yang keluaranya bilangan 1, 3, 5 dan seterusnya sampai batasan looping yang dimasukan. Batasan program dibawah ini adalah 10, untuk lebih jelasnya perhatikan program dibawah ini:
Program 4.15

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop = 1;
while(loop <= 10)
{
cout << loop << " ";
```

```
loop+=2;
}
getch();
return 0;
}
```
Keluaran program diatas adalah:
1 3 5 7 9
Program dibawah ini penggunaan while untuk aplikasi manampilkan huruf. Huruf yang ditampilkan adalah huruf abjad. Untuk lebih jelasnya perhatikan program dibawah ini:
Program 4.16
```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
char huruf = 'A';
while(huruf <= 'Z')
cout << "Huruf abjad = " << huruf++ << "\n";
getch();
huruf = 'A';
while(huruf <= 'Z')
{
cout << "Huruf abjad = " << huruf << "\n";
huruf+=13;
}
getch();
huruf = 'z';
while(huruf >= 'a')
cout << "Huruf abjad = " << huruf -  -  << "\n";
getch();
huruf = 'z';
while(huruf >= 'a')
{
cout << "Huruf abjad = " << huruf << "\n";
huruf - =8;
}
getch();
return 0;
```

```
}
```
Keluaran program adalah sebagai berikut:
Huruf abjad = A

Huruf abjad = B
Huruf abjad = C
Huruf abjad = D
Huruf abjad = E
Huruf abjad = F
Huruf abjad = G
Huruf abjad = H
Huruf abjad = I
Huruf abjad = J
Huruf abjad = K
Huruf abjad = L
Huruf abjad = M
Huruf abjad = N
Huruf abjad = O
Huruf abjad = P
Huruf abjad = Q
Huruf abjad = R
Huruf abjad = S
Huruf abjad = T
Huruf abjad = U
Huruf abjad = V
Huruf abjad =W
Huruf abjad = X
Huruf abjad = Y
Huruf abjad = Z

4.8. Pernyataan NESTED-WHILE

Pernyataaan nested while adalah suatu perulangan while didalam perulangan while yang lainnya.
Bentuk umum pernyataan Nested while sebagai berikut :

```
while (syarat)
{
while (syarat)
{
pernyataan;
}
}
```


Selain cara penulisan pernyataan ditulis diatas, dapat juga ditulis sebagai berikut:

```
while (syarat)
{
while (syarat)
{
```

……..
while (syarat)
{
pernyataan;
}
…….
}
}
Didalam penggunaan NESTEDWHILE, perulangan yang berada didalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang diluar diselesaikan terus sampai perulangan yang paling luar.
Perhatikan Contoh program dengan nested while dimana program tersebut merupakan program Segitiga Pascal dengan NESTED WHILE seperi berikut ini.
Program 4.17

```
#include<conio.h>
#include <iostream.h>
#include <math.h>
using namespace std;
int main(void)
{
int row = 0;
while(row<=3)
{
int col = 1;
while(col <= 3 - row)
{
cout << " ";
col++;
}
col=0;
while(col<=row)
```



```
{
cout << " " << (int)pow(2, col);
col++;
}
col= row  -  1;
while (col>= 0)
{
```

```
cout << " " << (int)pow(2, col);
col -  - ;
}
cout << "\n";
row++;
}
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:

```
1
1 2 1
1 2 4 2 1
1 2 4 8 4 2 1
```

Contoh program dibawah ini digunakan untuk menjumlahkan sejumlah data angka. Angka yang akan dijumlahkan dimasukan satupersatu. Proses pemasukan data angka akan berhenti ketika dimasukkan angka –1. Setelah itu tampil hasil penjumlahannya.

Program 4.18

```
#include <iostream.h>
#include <conio.h>
using namespace std;
void main()
{
int cacah = 0, data = 0, jumlah = 0;
while (data !=  - 1)
{
cout << "Masukkan data angka : ";
cin >> data;
jumlah += data;
cacah++;
}
cout << "Jumlah data adalah : " << jumlah" << endl;
```



```
cout << "Rata - rata : " << jumlah/cacah;
}
```

4.9. Perulangan DO-WHILE

Selain operasi loop menggunakan WHILE, bahasa C++ juga mempunyai DO-WHILE dan untuk perulangan. Setiap perulangan

dapat tepat untuk mengatasi berbagai masalah pemrograman. Yang dilakukan DO-WHILE terlihat mirip dengan perulangan yang berpaling terbalik.

Perulangan dengan pernyataan do-while merupakan perulangan yang mirip dengan perulangan while ataupun for. Perulangan for dipakai pada perulangan yang sudah diketahui berapa kali akan dijalankan. Sedangkan yang belum diketahui berapakali akan diulangi maka digunakan while atau do-while. Pernyataan do-while digunakan untuk menjalankan pernyataan terlebih dahulu baru kemudian memeriksa kondisi perulangan.

Pada pernyataan do-while, disini pemeriksaan terhadap loop dilakukan di bagian akhir (setelah tubuh loop). Pernyataan do-while akan mengulang proses secara terus menerus selama kondisi bernilai benar dan perulangan (loop) selesai jika kondisi bernilai salah.

Bentuk perulangan do-while dikendalikan oleh syarat/kondisi tertentu, yaitu perulangan akan terus dilaksanakan selama syarat/kondisi tersebut terpenuhi. Pernyataan dalam do-while akan dilaksanakan berulangkali selama syarat/kondisi bernilai benar. Jika syarat/kondisi bernilai salah badan perulangan tidak akan dilaksanakan, yang berarti perulangan selesai. Yang harus diperhatikan adalah perulangan harus berhenti. Perulangan yang tidak pernah berhenti menandakan bahwa logika dari algoritma tersebut salah.

Perbedaan dengan while sebelumnya yaitu bahwa pada dowhile statement perulangannya dilakukan terlebih dahulu baru

kemudian di cek kondisinya.
Sedangkan while kondisi dicek dulu
baru kemudia statement
perulangannya dijalankan. Akibat dari
hal ini adalah dalam do-while minimal
terdapat sekali perulangan.
Sedangkan while dimungkinkan
perulangan tidak pernah terjadi yaitu
ketika kondisinya langsung bernilai
salah. Bentuk umum perulangan dowhile,
sebagai berikut :

do
pernyataan;
while (syarat/kondisi)

Syarat/kondisi: merupakan ungkapan
logika yang hanya bernilai benar atau
salah, sehingga operator yang
dipakai disini adalah operator relasi
dan operator logika atau gabungan
dari keduanya. Gambar diagram alir
dari pernyataan do-while adalah
sebagai berikut:

Gambar 4.4. Diagram alir pernyataan do-while

Pernyataan dalam do-**while**
dapat berupa pernyataan tunggal
maupun jamak (lebih dari satu). Jika
pernyataannya berbentuk jamak,
maka pernyataan-pernyataan
tersebut harus diletakan didalam satu
blok dengan memakai tanda kurung
kurawal. Bentuk umum perulangan
do-**while**, dengan lebih dari satu
pernyataan, seperti berikut dibawah
ini:

```
do
{
Pernyataan;
Pernyataan;
}
while (syarat)
```

program dibawah ini merupakan
aplikasi dengan menggunakan dowhile
pada operasi bilangan naik.
Program dengan do-while dibawah ini

merupakn program do-while bilangan naik. Untuk lebih jelasnya perhatikan contoh program berikut:

Program 4.19

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

pernyataan

Kondisi

Benar

Salah


```
using namespace std;
int main()
{
int loop = 1;
do
cout << loop++ << " ";
while(loop <= 10);
getch();
return 0;
}
```

Hasil keluaran program adalah:

1 2 3 4 5 6 7 8 9 10

Seperti halnya program diatas, pernyataan do-while juga bisa digunakan untuk operasi bilangan turun. Program dibawah ini digunakan untuk menurunkan bilangan mulai dari 10 kemudian turun menjadi 9, 8 dan seterusnya. Untuk lebih jelasnya perhatikan program berikut ini:

Program 4.20

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop = 10;
do
cout << loop -  -  << " ";
while (loop >= 1);
getch();
return 0;
}
```

Hasil keluaran program adalah:
10 9 8 7 6 5 4 3 2 1
Selain digunakan untuk
menaikan dan menurunkan bilangan,
dibawah ini juga diberikan contoh
sebuah program yang menggunakan
pernyataan perulangan do-while yang
menampilkan bilangan ganjil.

Bilangan yang ditampilkan adalah
bilangan ganjil dengan urutan mulai
dari satu sampai batas yang
ditentukan yaitu 10. Sehingga
bilangan yang keluar dari program
adalah 1, 3, 5, 7, dan 9. Untuk lebih
jelasnya perhatikan program dibawah
ini:
Program 4.21

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int loop = 1;
do
{
cout << loop << " ";
loop+=2;
}
while (loop <= 10);
getch();
return 0;
}
```

Hasil keluaran program diatas adala sebagai berikut:
1 3 5 7 9 _

4.10. Pernyataan NESTED DO-WHILE

Pernyataaan nested do-while
adalah suatu perulangan do-while
didalam perulangan do-while lainnya.
Bentuk umum pernyataan Nested dowhile
sebagai berikut :

```
do
{
do
{
```

```
pernyataan;
}
while (syarat);
}
while (syarat);
```
pernyataan diatas dapat juga ditulis seperti potongan pernyataan dibawah ini:

```
do
{
do
{
……..
do
{
pernyataan;
}
while (syarat);
. …….
}
while (syarat);
}
while (syarat);
```
Didalam penggunaan nested dowhile, perulangan yang didalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang diluar diselesaikan terus sampai perulangan yang paling luar.
Program perulangan nested dowhile sebenarnya lebih komplek dibandingkan program do while biasa. Dalam mengerjakan program nested, baik while, do-while tentunya harus lebih teliti, karena jika tidak seringkali terjadi kesalahan ketika decompile.
Program dibawah ini merupakan contoh aplikasi nested do-while yang digunakan untuk mencari Segitiga Pascal. Untuk lebih jelasnya perhatikan contoh program dibawah ini:
Program 4.22
```
#include<conio.h>
#include <iostream.h>
#include <math.h>
```

```
#include<stdlib.h>
using namespace std;
int main(void)
{
int row = 0;
do
{
int col = 0;
do
{
cout << " ";
col++;
```



```
}
while(col <= 3 - row);
col=0;
do
{
cout << " " << (int)pow(2, col);
col++;
}
while(col<=row);
col = row  -  1;
do
{
cout << " " << (int)pow(2, col);
col -  - ;
}
while (col>= 0);
cout << "\n";
row++;
}
while(row<=3);
getch();
return 0;
}
```

Hasil keluaran program adalah:

```
1 0
1 2 1
1 2 4 2 1
1 2 4 8 4 2 1
```

4.11. Perulangan Tidak Berhingga

Perulangan tidak berhingga merupakan perulangan (loop) yang tak pernah berhenti atau mengulang

terus, hal ini sering terjadi
disebabkan adanya kesalahan
penanganan kondisi yang dipakai
untuk keluar dari loop. Bahkan suatu
ketika program perulangan ini
memang dirancang untuk tidak
pernah berhenti, walaupun hal ini
sangat jarang sekali. Perulangan
tidak terhingga biasanya karena
adanya kesalahan penulisan
program, sehingga ketika program
dijalankan akan berjalan terusmenerus,.
Sebagai contoh saja pada
program yang digunakan untuk
penulisan perintah nilai pencacah
salah seperti pada program berikut:

Program 4.23

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main()
{
int b;
for(b = 6; b >= 1; b++)
cout << b;
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:
Pada program tersebut diatas
tidak akan berhenti sampai dilakukan
penghentian dengan paksa. Pada
pernyataan for diatas tidak akan
berhenti untuk menampilkan bilangan
menaik, kesalahan terjadi pada
pengubah nilai pencacah,
seharusnya penulisan yang benar
adalah:
b –
Akan tetapi yang ditulis dalam
program adalah :
b ++
Oleh karena kondisi b >= 1 selalu
bernilai benar (karena b bernilai 6),

maka pernyataan cout << b; akan terus dijalankan. Jika terjadi hal semacam ini, untuk menghentikan proses yang terus menerus semacam ini dilakukan dengan menekan

tombol CTRL+ PAUSE atau CTRL + BREAK.

4.12. Pernyataan Break

Kadang-kadang perlu untuk menghentikan satu looping sebelum meninggalkan semua Iterasi. Pernyataan break, dapat digunakan untuk beralih pada didalam satu perulangan. Bila menemui break , perulangan berhenti dan program melompat ke pernyataan setelah perulangan.

Perulangan dengan while pada segmen program untuk menjalankan 10 kali, namun pernyataan break dapat menyebabkannya berhenti setelah kelima perulangan atau perulangan tertentu. Untuk lebih jelasnya perhatikan potongan program dibawah ini:

```
int count = 0;
while (count++ < 10)
{
cout << count << endl;
if (count == 5)
break;
}
```

Pernyataan break berfungsi untuk keluar dari perulangan baik for, while dan do-while serta struktur switch. Jika pernyataan break dikerjakan, maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir dari badan perulangan (loop).

Perhatikan contoh program pemakaian break dalam pernyataan for dibawah. Program digunakan untuk menghentikan bilangan deret menggunakan break

Program 2.24

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
int jumlah= 0;
int bilangan;
for (bilangan=0;bilangan < 20;bilangan++)
{
jumlah += bilangan;
if (jumlah >= 100) break;
}
cout << "Deret Bilangan : 1 + 2 + ... + " << bilangan << endl;
cout << "Jumlah Deret Bilangan = " <<jumlah;
```



```
getch();
return 0;
}
```

Hasil keluaran program diatas adalah sebagai berikut:

Deret Bilangan : 1 + 2 + …… + 14
Jumlah Deret Bilangan = 105

Program dibawah ini menggunakan
break dengan pernyataan while.
Program deret bilangan yang
menggunaan break adalah sebagai
berikut:

Program 4.25

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
int jumlah= 0;
int bilangan = 0;
while (bilangan < 20)
{
bilangan++;
jumlah += bilangan;
if (jumlah >= 100) break;
}
cout << "Deret Bilangan : 1 + 2 + ... + " << bilangan << endl;
cout << "Jumlah Deret Bilangan = " <<jumlah;
getch();
return 0;
```

```
}
```
Hasil keluaran program:
Deret Bilangan : 1 + 2 + …… + 14
Jumlah Deret Bilangan = 105
Contoh program dibawah ini menggunakan break dengan pernyataan do-while. Perhatikan program deret bilangan dengan menggunakan break berikut ini:

Program 4.26
```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
int jumlah= 0;
int bilangan = 0;
do
{
bilangan++;
jumlah += bilangan;
if (jumlah >= 100) break;
}
while (bilangan < 20);
cout << "Deret Bilangan : 1 + 2 + ... + " << bilangan << endl;
cout << "Jumlah Deret Bilangan = " <<jumlah;
getch();
return 0;
}
```
Hasil keluaran program diatas adalah sebagai berikut:
Deret Bilangan : 1 + 2 + …… + 14
Jumlah Deret Bilangan = 105

4.13. Pernyataan Continue

Pernyataan continue digunakan untuk mengarahkan eksekusi ke iterasi (proses loop) berikutnya yang berada pada loop yang sama, atau dengan kata lain mengembalikan proses yang sedang dilaksanakan keawal loop lagi, tanpa menjalankan sisa perintah dalam loop tersebut. Perhatikan contoh program yang menggunakan continue dalam pernyataan for berikut ini:

Program 4.27

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
```



```
int jumlah= 0;
int bilangan;
for(bilangan = 1; bilangan <= 20;bilangan++)
{
if (bilangan == 10 || bilangan == 11) continue;
jumlah += bilangan;
}
cout << "Deret Bilangan : 1 + 2 + ... + 9 + 12 + 13 + ... + " << bilangan - 1 << endl;
cout << "Jumlah Deret Bilangan = " <<jumlah;
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Deret Bilangan : 1 + 2 + …… + 9 + 12 + 13 + … + 20
Jumlah Deret Bilangan = 189
```

Perhatikan contoh program yang menggunakan continue dengan pernyataan while sebagai berikut ini:

Program 4.28

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
int jumlah= 0;
int bilangan = 0;
while (bilangan < 20)
{
bilangan++;
if (bilangan == 10 || bilangan == 11) continue;
jumlah += bilangan;
}
cout << "Deret Bilangan : 1 + 2 + ... + 9 + 12 + 13 + ... + " << bilangan << endl;
cout << "Jumlah Deret Bilangan = " <<jumlah;
getch();
return 0;
}
```

Hasil keluaran program:

```
Deret Bilangan : 1 + 2 + …… + 9 + 12 + 13 + … + 20
```

Jumlah Deret Bilangan = 189
Berikut ini merupakan contoh
program menggunakan continue
dalam sebuah pernyataan do-while.
Perhatikan program bilangan deret
dengan mengguakan continue seperti
dibawah:
Program 4.29

```
#include<conio.h>
#include <iostream.h>
using namespace std;
int main(void)
{
int jumlah= 0;
int bilangan = 0;
do
{
bilangan++;
if (bilangan == 10 || bilangan == 11) continue;
jumlah += bilangan;
}
while (bilangan < 20);
cout << "Deret Bilangan : 1 + 2 + ... + 9 + 12 + 13 + ... + " << bilangan << endl;
cout << "Jumlah Deret Bilangan = " <<jumlah;
getch();
return 0;
}
```

Hasil keluaran program:
Deret Bilangan : 1 + 2 + …… + 9 + 12 + 13 + … + 20
Jumlah Deret Bilangan = 189

4.14. Pernyataan Goto

Pernyataan goto merupakan
instruksi untuk mengarahkan
eksekusi program menuju pernyataan
yang diawali dengan suatu label.
Label merupakan suatu pengenal
(identifier) yang diikuti dengan tanda
titik dua (:).
Bentuk pemakaian goto adalah
sebagai berikut:
goto label;
Contoh penggunaan instruksi goto
dapat dilihat pada contoh program
berikut ini:

Program 4.30

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
using namespace std;
main()
{
int a, b;
char lagi;
atas:
cout << "Masukkan Bilangan = ";
cin >> a;
b = a % 2;
cout <<"Nilai " << a << " % 2 adalah = " << b;
cout << "\n\nIngin Hitung Lagi [Y/T] : ";
lagi = getche() ;
if (lagi == 'Y' || lagi == 'y')goto atas;
getch();
return 0;
}
```

Keluaran program tersebut diatas adalah:

```
Masukan Bilangan =
Nilai 5 % 2 adalah = 1
Ingin Hitung lagi (Y/T) : t
```

4.15. Soal Latihan

Jawablah soal latihan dibawah ini dengan baik dan benar.

1. Struktur perulangan secara umum terdiri dari dua bagian, sebutkan
2. Sebutkan fungsi operator increment dan decrement
3. Jelaskan bentuk pernyataan for dalam bahasa pemrograman
4. Jelaskan apa yang dimaksud dengan potongan program dibawah ini:

```
int num = 1;
for ( ; num <= 10; )
{
cout << num << "\t\t" << (num * num) << endl;
num++;
}
```

5. Sebutkan fungsi break
6. Sebutkan fungsi goto
7. Apa yang dimaksud dengan NESTED-WHILE

BAB 5
STATEMENT KENDALI
5.1. Pengertian Statement

Penyeleksian kondisi digunakan untuk mengarahkan perjalanan suatu proses. Penyeleksian kondisi dapat diibaratkan sebagai katup atau kran yang mengatur jalannya air. Bila katup terbuka maka air akan mengalir dan sebaliknya bila katup tertutup air tidak akan mengalir atau akan mengalir melalui tempat lain. Seleksi kondisi adalah proses penentuan langkah berikutnya berdasarkan proses yang terjadi sebelumnya. Seleksi kondisi ini sangat penting dalam pemrograman sebab dengan adanya seleksi kondisi, program dapat menentukan proses apa yang harus dilakukan selanjutnya berdasarkan keadaan sebelumnya. Sehingga nampak seolah olah program dapat berpikir dan mengambil keputusan. Disinilah letak kekurangan komputer yaitu tidak mampu berpikir sendiri, semua hal yang dilakukan adalah berdasarkan perintah.

Dalam memprogram seringkali digunakan suatu percabangan untuk pengambilan keputusan dari sejumlah pilihan yang mungkin. Bahasa pemrograman menyediakan pernyataan IF…THEN dan kata kunci yang lain seperti SWITCH...CASE untuk melakukan suatu percabangan. Dalam percabangan, keputusan diambil berdasarkan ekspresi kondisi. Ekpresi berkondisi adalah sebagian dari pernyataan program yang menanyakan pertanyaan True atau False (Benar atau Salah) mengenai

5.1. Pengertian Statement
5.2. Operator Relasi
5.3. Statement IF
5.4. Pernyataan IF/ELSE
5.5. Pernyataan IF/ELSE IF
5.6. Pernyataan IF/ELSE Majemuk
5.7. Pernyataan NESTED IF

5.8. Operator Logika
5.9. Operator Kondisional
5.10. Statement SWITCH
5.11. Pernyataan Switch … Case
5.12. IF...THEN, IF...THEN...ELSE dan Nested IF
5.13. Aplikasi Pernyataan IF pada Menu
5.14. Soal Latihan

properti, variabel, atau data lain pada kode program.

Kode-kode perintah yang diberikan dari suatu bahasa pemrograman untuk melakukan suatu proses. Blok merupakan satu atau lebih statemen yang harus dikerjakan setelah suatu kondisi dipenuhi.

Pernyataan Percabangan digunak-an untuk memecahkan persoalan untuk mengambil suatu keputusan diantara sekian pernyataan yang ada. Untuk keperluan pengambilan keputusan, Borland C++ menyediakan beberapa perintah antara lain.

5.2. Operator Relasi

Dalam melakukan pemrograman, kebanyakan programmer tentunya sudah banyak tahu bahwa programprogram komputer dibuat supaya mengikuti alur. Tentunya kita juga berfikir bagaimana jika komputer diprogram ternyata tidak mengikuti alur atau urutan yang telah dibuat? Maka komputer tersebut akan tidak bisa berjalan sesuai dengan yang kita inginkan. Alur atau urutan pada komputer ini antara lain:

- Adanya masukan dari pengguna.
- Dapat melakukan Satu atau lebih perhitungan atau proses.
- Menampilkan hasilnya pada layar.

Sebuah komputer yang baik, selain dapat melakukan perhitungan, tetapi juga sangat ahli dalam

membandingkan sebuah nilai untuk menentukan apakah lebih besar, kurang dari, atau sama dengan. Jenis operasi tersebut sangat berharga untuk memeriksa tugas-tugas seperti angka-angka penjualan, penentuan laba rugi, memeriksa untuk memastikan angka tersebut dalam rentang yang dapat diterima, dan memvalidasi input yang diberikan oleh pengguna.

Data Numerik pada bahasa C++ akan dibandingkan dengan menggunakan sebuah operator. Sebuah karakter juga dapat dibandingkan dengan menggunakan operator, karena dianggap sebagai karakter yang mempunyai nilai numerik pada C++. Penghubung pada setiap operator akan menentukan apakah ada hubungan antara dua nilai. Sebagai contoh, operator lebih (>) menentukan jika nilai lebih besar daripada yang lain. Operator kesetaraan (==) menentukan apakah dua nilai sama. Tabel dibawah merupakan semua operator penghubung pada bahasa C++.

Table 5.1. Operator Relasional

RELASI OPERATOR
> Lebih besar daripada
< Lebih kecil daripada
>= Lebih besar dari sama dengan
<= Kurang dari samadengan
== Sama dengan
!= Tidak sama dengan

Semua operator relasional tersebut diatas, bisa juga disebut dengan operator biner. Hal ini karena mereka menggunakan dua operand. Berikut adalah contoh yang menggunakan operator ekspresi yang lebih besar daripada:

x > y

Ekspresi tersebut merupakan ekspresi penghubung atau *relational expression*. Digunakan untuk menentukan apakah x lebih besar daripada y. Berikut ekspresi untuk menentukan jika x kurang dari y:

x <y

Selanjutnya bagaimana menentukan nilai sebuah operator penghubung tersebut dan bagaimana relational expression digunakan dalam program? Untuk menjawah hal tersebut ada yang perlu diingat yaitu: semua memiliki nilai. Relational ekspresi adalah ekspresi Boolean, yang berarti mereka hanya terdapat nilai benar atau salah. Jika x lebih besar dari y, maka ekspresi x> y akan benar, sedangkan kalimatnya dapat ditulis sebgai berikut:

Y == X

Pernyataan diatas akan akan salah jika, operator == digunakan untuk menentukan apakah operand sebelah kiri adalah sama dengan operand di kanan. Jika kedua operand memiliki nilai yang sama, ungkapan itu benar. Dengan asumsi bahwa a adalah 4 merupakan ungkapan yang benar:

a == 4

Namun berikut ini adalah salah:

a == 2

pasangan sesuatu hal yang berhubungan merupakan sebuah operator yang digunakan untuk menguji dua hubungan. Pada operator >= digunakan untuk menentukan jika operand disebelah kiri lebih besar dari atau sama dengan operand disebelah kanan. Dengan asumsi bahwa jika a adalah 4, dan b adalah 6, serta c adalah 4, maka ekspresi yang paling benar adalah sebagai berikut:

b> = a
a> = c
Namun ekpresi berikut dibawah ini adalah salah:
a> = 5
operator <= digunakan untuk menentukan jika operand disebelah kiri kurang dari atau sama dengan operand disebelah kanan. Perlu ditekankan sekali lagi, bahwa dengan asumsi a adalah 4, b adalah 6, dan c adalah 4, maka ekspresi yang benar adalah sebagai berikut:
a <= c
b <= 10
Namun ekpresi berikut adalah salah:
b <= a
Operator penghubung terakhir adalah !=, yang artinya adalah operator tidak

sama. Operator ini untuk menentukan jika operand disebelah kiri tidak sama dengan operand disebelah kanan, sedangkan kebalikan dari operator tersebut adalah operator ==. Seperti contoh sebelumnya, dengan asumsi a adalah 4, b adalah 6, dan c adalah 4, maka ekspresi yang benar adalah sebagai berikut:
a != b
b != c
Kalimat tersebut adalah benar karena a tidak sama dengan b dan b tidak sama dengan c. Namun berikut ini adalah ekspresi salah karena sama dengan c:
a != c
Tabel dibawah ini menunjukkan operator dimana nilai-nilai benar atau salah.
Table 5.2. Operator nilai benar atau salah.

EKPRESI	NILAI
x < y	salah, karena x lebih kecil daripada y
x > y	Betul, karena x lebih besar daripada y
x >= y	Betul, karena x lebih besar atau sama dengan y

x <= y Salah, karena x lebih kecil atau sama dengan y
y != x Betul, karena y tidak sama dengan x
Setelah diamati, sebenarnya terdapat sebuah pertanyaan yang sering kita tidak tahu yaitu: "apa yang dimaksud dengan benar?".
Pertanyaan tersebut merupakan suatu hal yang sangat perlu untuk dipertimbangkan. Jika ekspresi relatif dapat dilakukan secara benar atau salah, bagaimana sesuatu yang mewakili nilai-nilai internal dalam sebuah program? Pada sebuah komputer misalnya, bagaimana komputer menyimpan benar?. Bagaimana pula cara menyimpan data yang salah?. Hal ini membutuhkan suatu tanda atau kode yang disepakati sesuai dengan kesepakatan umum pada sebuah komputer.
Dalam bahasa C++, nilai benar akan diwakili dengan angka 1 dan salah dengan angka 0. Untuk menggambarkan lebih lengkap, perhatikan program dibawah ini:
Program 5.1.

```
#include<conio.h>
#include <iostream>
using namespace std;
int main()
{
bool nilaiBetul, nilaiSalah;
```



```
int x = 5, y = 10;
nilaiBetul = x < y;
nilaiSalah = y == x;
cout << "Benar adalah " << nilaiBetul << endl;
cout << "Salah adalah " << nilaiSalah << endl;
getch();
return 0;
}
```

Keluaran Programnya adalah sebagai berikut:

```
Benar adalah 1
Salah adalah 0
```

Selanjutnya coba kita memeriksa pernyataan-pernyataan yang berisi ekspresi relasi sedikit lebih dekat:
nilaiBetul= x < y;
nilaiSalah = y == x;
pernyataan-pernyataan tersebut tampaknya aneh karena menempatkan nilai perbandingan ke dalam sebuah variabel. Pada pernyataan pertama, variabel **trueValue** dengn diberi hasil x <y. Ketika x kurang dari y, ungkapan itu benar, dan variabel trueValue memberikan nilai 1. Sedangkan pada pernyataan yang kedua ungkapan y == x adalah salah, sehingga variabel falseValue akan keluar nilai 0.
Perhatikan bahwa dalam kedua kasus operasi relasional dilaksanakan sebelum tugas operasi dilakukan.
Selain itu tanda kurung dapat digunakan untuk mengubah urutan operasi, hal ini selalu terjadi karena adanya operator relasional tingkatannya relatif lebih tinggi dan harus didahulukan dalam operasinya daripada operator yang tingkatnya lebih rendah. Demikian juga, ada operator aritmatika yang harus lebih diutamakan daripada operator relasional. Seperti dalam pernyataan berikut ini:
NilaiSalah = x < y – 8;
Dari peryataan diatas misalnya, yang pertama Y - 8 akan dievaluasi terlebih dahulu untuk menghasilkan nilai 2. Kemudian X yang bernilai 5 akan dibandingkan dengan 2. Jika nilai 5 kurang dari 2, nilai nol akan masukan ke falseValue. Tabel dibawah menunjukkan contoh lainnya termasuk pernyataan yang relatif biasa.

Tabel 5.3. Pernyataan ekpresi relasional

PERNYATAAN KELUARAN

z = x < y z sama dengan 0 karena x tidak kurang daripada y

cout << (x > y); Tampilkan 1 karena x lebih besar daripada y.

z = x >= y; z samadengan 1 karena x lebih besar samadengan y.

cout << (x <= y); Tampilkan 0 karena x tidak kurang samadengan y

z = y != x; z samadengan 1 karena y tidak samadengan x

cout << (x == y + 3); Tampilkan 1 karena x sama dengan y + 3

Operator relasional juga memiliki urutan yang mempunyai prioritas diantara mereka sendiri. Kedua operator dalam tes kesamaan atau tidak sama (== dan ! =) mempunyai tingkat kesamaan yang sama antara satu sama lain.

Empat penghubung operator lainnya, ada yang mempunyai prioritas tinggi atau rendah antara satu sama lain. Keempat operator relasional seperti dalam. Tabel dibawah menunjukkan diutamakan yang berhubungan operator.

Tabel 5.4. Prioritas operator relasional (dari tertinggi ke rendah)

== !=

> >= < <=

Berikut ini adalah sebuah contoh bagaimana operator diterapkan. Jika a = 9, b = 24, dan c = 0, pernyataan berikut ini akan menyebabkan a bernilai 1 akan ditampilkan.

cout <<(c == a> b);

Karena adanya nilai yang relatif diutamakan dari operator dalam berekspresi ini, a > b akan dievaluasi terlebih dahulu. Ketika 9 tidak lebih besar dari 24, maka hal tersebut akan mengevaluasi salah atau 0. Kemudian ketika C == 0 maka akan dihasilkan. Ketika c tidak sama dengan 0, hal tersebut akan menghasilkan nilai betul, atau 1. Sehingga nilai 1 akan dimasukkan ke dalam output stream dan ditampilkan.

5.3. Statement IF

Anda mungkin berpikir mengenai pernyataan dalam program

procedural seperti langkah-langkah yang diambil sebagai individu. Untuk mencapai tujuan tersebut , maka harus dimulai dari awal dan mengikuti setiap langkah, satu setelah yang lain hingga dapat mencapai tujuan. Program-program yang ditulis ini adalah seperti halnya sebuah "jalan" dimana pelaksanaannya harus diikuti. Perhatikan langkah program dibawah ini:

Gambar 5.1. Langkah-langkah program

Seperti ditunjukkan dalam Gambar diatas, pelaksanaan program mengalir secara berurutan dari satu pernyataan ke pernyataan berikutnya. Jenis program ini sering disebut-garis lurus karena program yang dijalankan dalam pernyataan yang lurus "baris," tanpa simpangan kearah yang lain. Apakah hal tersebut tidak akan berguna, jika sebuah program dapat memiliki lebih dari satu "jalur atau arah" pelaksanaan? Bagaimana jika sebuah program dapat menjalankan beberapa pernyataan hanya dalam kondisi tertentu?.

Hal yang dapat dicapai dengan pernyataan IF, seperti digambarkan oleh Program dibawah ini. Pengguna memasukan tiga nilai ujian dan program menghitung rata-rata nilai tersebut. Jika rata-rata lebih besar dari 95, program memberikan selamat pada pengguna mendapatkan nilai tinggi tersebut. Untuk lebih jelasnya perhatikan program untuk mencari rata-rata pada tiga nilai masukan dibawah:

Program 5.2

```
#include <iostream>
#include <iomanip>
#include <iostream>
```

```
using namespace std;
int main()
{
double panjang, lebar, luas;
cout << "masukan panjang segiempat: ";
cin >> panjang;
cout << "Masukan lebar segiempat: ";
cin >> lebar;
luas = panjang * lebar;
cout << "Luasnya: " << luas << endl;
return 0;
}
```



```
using namespace std;
int main()
{
int score1, score2, score3;
double average;
cout << "Masukan 3 nilai Ujian dan saya akan mencari rata - ratanya: ";
cin >> score1 >> score2 >> score3;
average = (score1 + score2 + score3) / 3.0;
cout << fixed << showpoint << setprecision(1);
cout << "Rata - rata Nilai Anda Adalah" << average << endl;
if (average == 100)
{
cout << "Selamat….! ";
cout << "Nilai Anda Sempurna!\n";
}
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:
Masukan 3 nilai Ujian dan saya akan mencari rata - ratanya: **80 90 70[Enter]**
Rata - rata Nilai Anda Adalah **80.0**
Keluaran program diatas adalah sebagai berikut:
Masukan 3 nilai Ujian dan saya akan mencari rata - ratanya: 100 100 100[Enter]
Rata - rata Nilai Anda Adalah 100.0
Selamat….! Nilai Anda Sempurna!
Jika dalam program tersebut pada potongan program seperti dibawah ini:

```
if (average == 100)
{
cout << "Selamat….! ";
cout << "Nilai Anda Sempurna!\n";
}
```

Maka potongan program tersebut
diatas akan menampilkan data tulisan

"selamat….! "….., jika data yang dimasukan sama dengan seratus. Jika tidak maka hanya akan menampilkan rata-rata nilainya saja tanpa ada ucapan selamat atau " Selamat….! ". Gambar dibawah ini menunjukkan bentuk pernyataan yang digunakan pada pernyataan IF dan flowchart visual dapat dijelaskan cara kerjanya sebagai berikut:

Gambar 5.2. Diagram alir pernyataan IF

Dari gambar tersebut diatas pernyataan IF dapat ditulis sebagai berikut:

```
if (ekspresi)
{
Pernyataan 1;
Pernyataan 2;
.
.
Pernyataan n;
}
```

Pernyataan IF adalah sangat sederhana cara kerjanya. Jika ekspresi didalam tanda kurung yang benar, maka pernyataan didalam braces (kotak) yang akan dijalankan, jika tidak, maka mereka diabaikan. Blok pernyataan ini adalah merupakan *conditionally executed* karena pernyataan hanya dijalankan didalam kondisi yang diekspresikan dalam tanda kurung yang benar.

Program diatas menggambarkan sebuah pernyataan IF. Dalam pernyataan cout hanya dilaksanakan dibawah kondisi yang sama dengan rata-rata 100.

Misalnya kalau diuraikan dalam sebuah pernyataan dalm keseharian adalah sebagai berikut: Jika bensin mobil sudah hampir habis, maka berhenti dipompa bensin untuk mengisi bensin.

□□ Jika diluar hujan, pergi kedalam rumah.

□□ Jika Anda lapar, carilah sesuatu
untuk dimakan.
Jika blok pernyataan
conditionally executed hanya satu
Betul
salah
Kondisi
Pernyataan

pernyataan, maka braces dapat
diabaikan. Misalnya, dalam Program
diatas, jika dua pernyataan cout yang
digabungkan menjadi satu
pernyataan, mereka dapat menulis
seperti yang ditunjukkan di sini.
if (average == 100)
cout << "selamat ! Nilai anda Sempurna!\n";
Perintah IF yang menyatakan
pernyataan kondisional (bersyarat)
dapat ditulis sintaks sederhana IF
adalah
if (kondisi) statement;
Statement pada sintaks di atas
akan dilakukan jika kondisinya
bernilai TRUE (tidak sama dengan
nol). Apabila statement yang akan
dilakukan lebih dari satu, maka
sintaksnya menjadi seperti dibawah
ini:
if (kondisi)
{
statement1;
statement2;
..
}
Contoh sederhana penggunaan
IF adalah untuk menentukan boleh
tidaknya seseorang melihat film
bioskop. Seseorang diperbolehkan
menonton jika usianya 17 tahun ke
atas. Perhatikan contoh program C++
berikut ini:
Program 5.3
```
#include <iostream.h>
#include <conio.h>
```

```
using namespace std;
int main()
{
int usia;
cout << "Berapa usia Anda : ";
cin >> usia;
if (usia < 17)
cout << "Anda tidak boleh menonton bioskop";
getch();
return 0;
}
```


Keluaran program diatas adalah:

```
Berapa usia Anda : 9
Anda tidak boleh menonton bioskop
```

Statement IF juga dapat ditambahkan ELSE sebagai konsekuensi alternatif jika kondisi tidak dipenuhi (FALSE). Untuk lebih jelasnya perhatikan sintaks program dibawah ini:

```
if (kondisi)
{
statement1;
statement2;
..}
else {
statement1;
statement2;
}
```

Anda dapat pula memodifikasi program C++ untuk menentukan boleh tidaknya seseorang menonton bioskop seperti program dibawah ini:

Program 5.4

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main()
{
int usia;
cout << "Berapa usia Anda : ";
cin >> usia;
if (usia < 17)
cout << "Anda tidak boleh menonton bioskop";
```

```
else
cout << "Anda boleh menonton bioskop";
getch();
return 0;
}
```
Keluaran programnya adalah sebagai berikut:
```
Berapa usia Anda : 16
Anda tidak boleh menonton bioskop
```

Untuk menyatakan kondisi atau syarat, akan dicek pada IF, Anda dapat menggunakan operator logika dan operator relasional seperti yang telah dijelaskan sebelumnya.
Perhatikan contoh penulisan program dibawah ini:
```
if ((a >= 2) && (b == 3))
{
….
}
```
Jangan dituliskan seperti ini:
```
if (a >= 2) && (b == 3)
{
….
}
```
Dan juga tidak boleh seperti ini:
```
if ((a >= 2) && (b = 3))
{
….
}
```
Perintah b = 3 merupakan assignment bukan relasional. C++ selalu memperlakukan nilai tidak sama dengan nol sebagai TRUE atau benar dan nilai nol sama dengan FALSE atau nilai salah. Oleh karena itu, dua perintah dibawah ini adalah identik. Perhatikan potongan program dibawah ini:
```
if (bil % 2 != 0)
cout << “Bilangan ganjil”;
if (bil % 2)
cout << “Bilangan ganjil”
```
Selain itu, IF juga dapat berbentuk seperti program dibawah ini:
```
if (kondisi1)
```

```
statement1;
else if (kondisi2)
statement2;
else if (kondisi3)
statement3;
.
.
else statement;
```

IF dapat juga menggunakan operator pembanding (*comparison operators*) untuk mengeksekusi suatu pilihan dari 2 pilihan yang ada, tergantung hasil pembandingannya.
Penulisan Instruksi atau sintaknya adalah sebagai berikut:

```
if (Kondisi)
{
```



```
Perintah yang akan dieksekusi jika kondisi bernilai true
}
else
{
Perintah yang akan dieksekusi jika kondisi bernilai false
}
```

Kondisi menunjukkan berbagai *tipe statement* atau fungsi yang menghasilkan nilai benar atau salah.
Berikut ini diberikan contoh kondisi yang berupa perbandingan antara variabel dengan nilai, variabel lain atau fungsi. Perhatikan potongan program dibawah ini:

```
if (x > 5)
{
y = 10;
}
if (x > y)
{
y = x;
}
if (x > val(angka))
{
y = 20;
}
```

Suatu ketika akan ditemui pula

penggunaan instruksi *IF .. THEN* statements, dimana isntruksi ini digunakan untuk mengevaluasi lebih dari satu kondisi. Untuk itu digunakan tambahan *blok IF..THEN..ELSE IF* statements. Perhatikan contoh program dibawah ini:

```
if (x < 5)
cout<<“Nilai x kurang dari 5”;
else
if (x < 10)
cout<<“Nilai X antara 5 dan 9”;
else
cout<<“Nilai x lebih dari 9”;
```

Jika suatu kondisi tergantung pada kondisi lain yang sudah bernilai true (seperti if hari = senin dan if jam = 6.30) maka harus digunakan percabangan bersarang (nested If statements). Perhatikan contoh berikut ini:


```
if (hari = senin)
{
if (jam = 6.30)
{
…
}
}
```

Untuk lebih jelasnya mengenai pernyataan IF, perhatikan sebuah kasus yang digunakan untuk menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dimana kasus tersebut mempunyai kriteria sebagai berikut:

- Tidak ada potongan jika total pembelian kurang Rp. 50.000,-
- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.

Perhatikan Program yang menggunakan pernyataan IF.

Program 5.5

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
using namespace std;
main()
{
double tot_beli, potongan=0, jum_bayar=0;
cout<<"Total Pembelian Rp. ";
cin>>tot_beli;
if (tot_beli >= 50000)
potongan = 0.2 * tot_beli;
cout<<"Besarnya Potongan Rp. ";
jum_bayar = tot_beli  -  potongan;
cout<<"Jumlah yang harus dibayarkan Rp. ",jum_bayar;
getch();
return 0;
}
```

5.4. Pernyataan IF/ELSE

Pernyataan IF/ELSE merupakan sebuah pengembangan pernyataan IF. Secara umum bentuk pernyataan ini juka digambarkan dalam bentuk flowchart sebagai berikut:

Gambar 5.3. Diagram alir IF-ELSE

Jika diuraikan dalam bentuk kalimat, gambar diatas dapat ditulis menjadi sebuah pernyataan seperti dibawah ini:

```
if (kondisi)
{
Pernyataan 1
}
else
{
Pernyataan 2
}
```

Seperti pada pernyataan IF diatas, sebuah kondisi akan dilihat terlebih dahulu. Jika kondisi adalah benar, maka sebuah blok yang berisi satu atau lebih pernyataan akan dijalankan. Jika ekspresi adalah salah, maka pernyataan grup

berbeda yang akan dijalankan.
Program dibawah merupakan contoh
penggunaan pernyataan bersama
operator modulus untuk menentukan
apakah angka yang dihasilkan ganjil
atau genap.
Program 5.6

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
```

Kondisi
Pernyataan 1 Pernyataan 2
Betul salah


```
{
int number;
cout << "masukan bilangan bulat dan saya akan memilahnya dengan IF\n";
cout << "adalah ganjil atau genap. ";
cin >> number;
if (number % 2 == 0)
cout << number << " adalah genap.\n";
else
cout << number << " adalah ganjil.\n";
getch();
return 0;
}
```

Keluaran program adalah sebagai berikut:

```
masukan bilangan bulat dan saya akan memilahnya dengan IF
adalah ganjil atau genap. 17[Enter]
17 adalah ganjil
```

Bagian lain diakhir pernyataan IF
akan menentukan pernyataan yang
akan dijalankan ketika kondisi yang
ada tidak sesuai atau salah. Bila
angka 2% tidak sama 0, pesan yang
dicetak menunjukkan angka ganjil.
Perlu diketahui bahwa program ini
hanya akan mengambil salah satu
dari dua jalur pernyataan IF/ELSE..
Jika Anda berpikir tentang
pernyataan dalam program komputer,
langkah-langkah yang diambil
berdasarkan pertimbangkan
pernyataan IF/ELSE sebagai penentu

jalan. Bahkan kadang terjadi jalanya program memutar, seperti halnya pada sebuah pernyataan IF, sedangkan pada pernyataan IF/ELSE menyebabkan pelaksanaan program mengikuti salah satu dari dua jalur yang tersedia.
Perhatikan gaya pemrograman yang digunakan untuk membangun pernyataan IF/ELSE. Atau dengan kata lain adalah pada tingkat yang sama seperti halnya perbedaan. Pernyataan eksekusi yang dikendalikan oleh orang lain kadang berbeda satu tingkat.
Seperti halnya pada pernyataan IF, jika tidak menggunakan kotak sebagai bagian kontrol sebuah pernyataan. Program diatas menggambarkan hal tersebut ini. Ia juga memperlihatkan salah satu cara untuk menangani sebuah masalah pemrograman yaitu pembagian oleh nol.
Pembagian dengan nol sangat mustahil untuk dilakukan secara matematis dan biasanya menyebabkan program crash. Hal Ini berarti program akan terhenti sebelum waktunya, kadang-kadang akan muncul dengan pesan kesalahan. Program diatas menunjukkan salah satu cara untuk menguji nilai dari pembagi sebelum dilakukan pembagian. Nilai num2 akan diuji sebelum pembagian dilakukan. Jika pengguna memasukkan angka 0, maka baris dikontrol oleh bagian IF ketika program dijalankan, menampilkan

pesan yang menunjukkan program tidak dapat melakukan pembagian nol. Jika tidak, bagian lain mengambil kontrol, dan kemudian membagi

num1 oleh num2 serta menampilkan hasilnya.

Salah satu aplikasi penggunaan IF/ELSE misalnya digunakan pada suatu permasalahan untuk menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria sebagai berikut:

- jika total pembelian kurang dari Rp. 50.000,- potongan yang diterima sebesar 5% dari total pembelian.
- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.

Untuk lebih jelasnya perhatikan contoh program IF/ELSE dibawah ini:

Program 5.7

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
using namespace std;
main()
{
double tot_beli, potongan=0, jum_bayar=0;
cout<<"Total Pembelian Rp. ";
cin>>tot_beli;
if (tot_beli >= 50000)
potongan = 0.2 * tot_beli;
else
potongan = 0.05 * tot_beli;
cout<<"Besarnya Potongan Rp. "<<potongan<<endl;
jum_bayar = tot_beli  -  potongan;
cout<<"Jumlah yang harus dibayarkan Rp. ",jum_bayar;
getch();
return 0;
}
```

Program dibawah ini digunakan untuk menentukan sebuah operasi pembagian yang menggunakan IF/ELSE , jika suatu bilangan dibagi dengan nol maka program akan memberitahukan bahwa program tidak bisa melakukan operasi.

Program 5.8

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
double num1, num2, quotient;
cout << "Masukan Angka: ";
cin >> num1;
cout << "Masukan Angka Lain: ";
cin >> num2;
if (num2 == 0)
{
cout << "Pembagian oleh 0 tidak mungkin dilakukan.\n";
cout << "silakan masukan angka lagi, ";
cout << "Angka lain yang lebih besar daripada 0.\n";
}
else
{
quotient = num1 / num2;
cout << "The quotient of " << num1 << " divided by ";
cout << num2 << " is " << quotient << ".\n";
}
getch();
return 0;
}
```

Keluaran program setelh diberi masukan adalah sebagai berikut:
Masukan Angka: **10[Enter]**
Masukan Angka Lain: **0[Enter]**
Pembagian oleh 0 tidak mungkin dilakukan.
silakan masukan angka lagi, Angka lain yang lebih besar daripada 0

5.5. Pernyataan IF/ELSE IF

Pernyataan IF/ELSE IF
merupakan sebuah pernyataan
dimana programmer dapat membuat
beberapa keputusan dengan
menggunakan aturan yang berbeda
tetapi ada kaitannya dengan
pernyataan sebelumnya. Misalnya,
programmer akan menentukan jenis
mantel atau jaket yang dipakai dalam

konsultasi sehingga aturanya adalah sebagai berikut:

if sangat dingin, pakailah mantel yang tebal,
else, jika dingin, memakai jaket yang tipis,
else, jika berangin, memakai baju hem,
else, jika panas, tidak perlu memakai jaket.
Tujuan aturan tersebut adalah untuk menentukan jenis pakaian luar yang akan dipakai. Jika sangat dingin, pertama menharuskan menggunakan aturan yang memakai mantel tebal. Semua peraturan lainnya kemudian dapat diabaikan. Jika aturan pertama tidak berlaku, dan jika tidak dingin sekali, maka aturan kedua digunakan. Jika aturan tidak berlaku, aturan yang ketiga digunakan, dan seterusnya.
Suatu cara yang digunakan untuk menhubungkan peraturan tersebut sangat penting. Jika mereka hanya berdiri sendiri, dimungkinkan akan keluar rumah dengan memakai jaket yang salah, atau mungkin, lebih dari satu jaket. Misalnya, jika berangin, maka aturannya harus berpakaian baju hem. Bagaimana jika ada keduanya yaitu sangat dingin dan berangin? Apakah akan memakai sebuah baju hem?. Karena aturan harus dipatuhi, pertama akan menentukan aturan memakai mantel tebal dan memakai baju hem karena berangin.
Jenis pengambilan keputusan juga sangat umum dalam pemrograman. Dalam bahasa C++ hal tersebut dicapai melalui pernyataan IF/ELSE IF. Gambar dibawah menunjukan format cara kerjanya.

Gambar 5.4. Pernyataan IF/ELSE IF
Dari gambar diatas dapat diuraikan dalam pernyataan umu IF/ELSE IF adalah sebagai berikut:

```
if (kondisi 1)
{
```

Pernyataan 1;
}
else if (kondisi *2*)
{
Pernyataan 2;
}
.
.
else if (kondisi *n*)
{
Pernyataan n;
}
Betul
salah
Kondisi 1 Pernyataan 1
Betul
salah
Kondisi 2 Pernyataan 2
Betul
salah
Kondisi n Pernyataan n

Susunan ternyataan tersebut seperti halnya sebuah pernyataan berbentuk rantai IF/ELSE. Pada bagian ELSE sebuah pernyataan akan terkait dengan IF dari bagian yang lain. Ketika digabungkan cara ini, maka IF/ELSE dari rantai akan menjadi sebuah pernyataan. Program dibawah menunjukkan contoh, dimana Pengguna diminta untuk memasukkan angka skor tes dan program menampilkan huruf yang besar.

Program dibawah ini menggunakan pernyataan IF/ELSE yang digunakan untuk operasi file huruf kapital.

Program 5.9

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
```

```
int testScore;
char grade;
cout << "Masukan Nilai Angka maka saya akan mengujinya\n";
cout << "Nilai Konversi huruf adalah: ";
cin >> testScore;
if (testScore < 60)
grade = 'F';
else if (testScore < 70)
grade = 'D';
else if (testScore < 80)
grade = 'C';
else if (testScore < 90)
grade = 'B';
else if (testScore <= 100)
grade = 'A';
cout << " Grade Anda Adalah " << grade << ".\n";
getch();
return 0;
}
```

Keluaran program setelah memasukan data adalah sebagai berikut:

```
Masukan Nilai Angka maka saya akan mengujinya
Nilai Konversi huruf adalah: 90
```



```
Grade Anda Adalah A.
```

Pernyataan IF/ELSE mempunyai sejumlag karakterisitik khusus. Perhatikan dan analisis kerja dari ekprsesi relasional pengujian sebuah testscore < 60 dibawah ini:

```
if (testScore < 60)
nilai = 'F';
```

Jika testScore kurang dari 60, huruf 'F' dimasukan ke nilai dan sisanya dari pernyataan yang terkait dengan IF diabaikan. Jika TestScore tidak kurang dari 60, maka tugsa akan diambil alih oleh bagian lain pernyataan IF berikutnya yang akan dijalankan.

```
else if (testScore < 70)
nilai = 'D';
```

Pertama jika seluruh pernyataan disaring untuk memilih nilai kurang dari 60, sehingga saat ini berikutnya pernyataan IF memilih, variabel

testScore akan memiliki nilai 60 atau lebih besar. Jika "testScore" kurang dari 70, huruf 'D' dimasukan ke nilai dan sisanya dari Pernyataan IF/ELSE IF ini diabaikan. lingkaran peristiwa ini terus berlangsung sampai salah satu ekspresi kondisional ditemukan benar atau akhir pernyataan yang dihadapi. Dalam kedua kasus, program ini dilanjutkan kembali pada pernyataan yang mengikutinya. Pernyataan IF/ELSE IF, merupakan pernyataan untuk mencetak kondisi yang berhasil. Gambar dibawah menunjukkan jalan yang dapat diambil melalui pernyataan IF/ELSE IF.

Gambar 5.5. Pernyataan IF/ELSE IF

pernyataan IF pada setiap struktur bergantung pada semua pernyataan IF sebelumnya yang salah. Pernyataan ELSE IF berikutnya dijalankan apabila ekspresi kondisional mengikuti ELSE IF itu benar dan semua ekspresi kondisional sebelumnya adalah salah. Untuk mendemonstrasikan bagaimana bekerjanya, perhatikan program dibawah, yang menggunakan pernyataan IF yang independen bukan sebuah pernyataan IF/ELSE. Program

SkorTest
< 60 ?
Nilai= ’F ’
SkorTest
< 70 ?
Tidak Ya
Nilai= ’D ’
SkorTest
< 80 ?
Tidak Ya
Nilai= ’C ’
SkorTest

< 90 ?
Tidak Ya
Nilai= ’B ’
SkorTest
<=100 ?
Tidak Ya
Nilai= ’A ’
Tidak Ya

tersebut digunakan untuk
mengilustrasikan kesalahan yang
terjadi ketika pernyataan IF/ELSE
yang digunakan untuk menentukan
nilai huruf ke nilai angka. Perhatikan
program dibawah ini:
Program 5.10

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
int skortest;
char nilai;
cout << "Masukan Nilai Angka maka saya akan mengujinya\n";
cout << "Nilai Konversi huruf adalah: ";
cin >> skortest;
if (skortest < 60)
nilai = 'F';
if (skortest < 70)
nilai = 'D';
if (skortest < 80)
nilai = 'C';
if (skortest < 90)
nilai = 'B';
if (skortest <= 100)
nilai = 'A';
cout << " Nilai Anda Adalah " << nilai << ".\n";
getch();
return 0;
}
```

Keluaran program setelah dilakukan pengisian data pada masukan adalah:
Masukan Nilai Angka maka saya akan mengujinya
Nilai Konversi huruf adalah: 40**[Enter]**
Nilai Anda Adalah A.
Pada program dibawah, semua

pernyataan IF dijalankan karena hal tersebut merupakan pernyataan yang berdiri sendiri. Pada contoh keluaran, testScore diberikan nilai 40, namun ada siswa yang menerima A. Berikutnya apa yang terjadi ?. Karena skor siswa kurang dari 60,

yang pertama pernyataan IF akan menyebabkan nilai 'F' memasukannya ke variabel grade. Namun, karena pernyataan berikutnya jika tidak tersambung ke pernyataan pertama, maka hal tersebut akan dieksekusi dengan baik. ketika testScore kurang dari 70, dan menyebabkan nilai 'D' ditugaskan ke grade, dan mengganti 'F' yang sebelumnya disimpan di sana. Ini terus sampai semua jika ada pernyataan dijalankan. Yang terakhir akan menyebabkan 'A' akan ditugaskan untuk "grade".

Pada program dibawah menggunakan pernyataan if/else if untuk melaksanakan konversi nilai hurif (A, B, C, D, or F) menjadi nilai angka. Perhatikan program dibawah ini:

Program 5.11

```
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
int testScore;
char grade;
bool goodScore = true;
cout << "Masukan Nilai Angka maka saya akan mengujinya\n";
cout << "Nilai Konversi huruf adalah: ";
cin >> testScore;
if (testScore < 60)
grade = 'F';
else if (testScore < 70)
grade = 'D';
```

```
else if (testScore < 80)
grade = 'C';
else if (testScore < 90)
grade = 'B';
else if (testScore <= 100)
grade = 'A';
else
goodScore = false;
if (goodScore)
cout << " Grade Anda Adalah " << grade << ".\n";
else
{
```

```
cout << testScore << " Adalah skor anda yang tidak sah.\n";
cout << "Silakan masukan skor dibawah 100. \n";
}
getch();
return 0;
}
```
Keluaran program ddiatas setelah memasukan data adalah sebagai berikut:
Masukan Nilai Angka maka saya akan mengujinya
Nilai Konversi huruf adalah: 110
110 Adalah skor anda yang tidak sah.
Silakan masukan skor dibawah 100.

5.6. Pernyataan IF/ELSE Majemuk

Bentuk dari IF-ELSE bertingkat sebenarnya serupa dengan NESTED IF, keuntungan penggunaan IF-ELSE bertingkat dibanding dengan NESTED IF adalah cara atau bentuk penulisan yang lebih sederhana. Untuk lebih jelasnya perhatikan bentuk penulisan program dibawah ini:

```
if (syarat)
{
… perintah;
… perintah;
}
else if (syarat)
{
… perintah;
… perintah;
}
else
```

{
… perintah;
… perintah;
}
Untuk lebih jelasnya, perhatikan sebuah kasus di suatu perusahaan memberikan komisi kepada para selesman dengan ketentuan sebagai berikut:
- Bila salesman dapat menjual barang hingga Rp. 200.000 ,- , akan diberikan uang jasa sebesar Rp. 10.000 ditambah dengan uang komisi Rp. 10% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 200.000 ,- , akan diberikan uang jasa sebesar

Rp. 20.000 ditambah dengan uang komisi Rp. 15% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 500.000 ,- , akan diberikan uang jasa sebesar Rp. 30.000 ditambah dengan uang komisi Rp. 20% dari pendapatan yang diperoleh hari itu.

Kasus yang ada di perusahaan tersebut dapat diatasi dengan menggunakan program IF-Else-IF, dimana programnya adalah sebagai berikut:
Program 5.12

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;
int main()
{
float pendptan, jasa=0, komisi=0, total=0;
cout<<"Pendapatan Hari ini Rp. ";
cin>>pendptan;
if (pendptan >= 0 && pendptan <= 200000)
{
```

```
jasa=10000;
komisi=0.1*pendptan;
}
else if(pendptan<=500000)
{
jasa=20000;
komisi=0.15*pendptan;
}
else
{
jasa=30000;
komisi=0.2*pendptan;
}
/* menghitung total */
total = komisi+jasa;
cout<<"Uang Jasa Rp. "<<jasa<<endl;
cout<<"Uang Komisi Rp. "<<komisi<<endl;
cout<<"============================"<<endl;
```



```
cout<<"Hasil Total Rp. "<<total<<endl;
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:

```
Pendapatan Hari ini Rp. 10000
Uang Jasa Rp. 10000
Uang Komisi Rp. 1000
============================
Hasil Total Rp. 11000
```

5.7. Pernyataan NESTED IF

Ketika sebuah pernyataan IF muncul didalam pernyataan IF lain, maka hal ini dikategorikan sebagai nested. Kenyataannya dalam struktur IF/ELSE adalah pernyataan nested if. Jika masing-masing setelah if pertama adalah nested di bagian lain IF sebelumnya. Karena nested if merupakan pernyataan if yang berada didalam pernyataan if yang lainnya, sehingga utuk lebih jelasnya bentuk penulisan pernyataan Nested if dapat ditulis seperti berikut dibawah ini:

```
if(syarat)
```

```
if(syarat)
… perintah;
else
… perintah;
else
if(syarat)
… perintah;
else
… perintah;
```

Sebagai contoh saja misalnya program dibawah digunakan untuk menentukan apakah pelanggan bank memenuhi syarat khusus untuk mendapatkan bunga pinjaman, dimana syarat khusus ini ditujukan untuk pelanggan orang-orang yang baru lulus dari sekolah dan bekerja. Program berikut ini mendemontasikan pernyataan nested if.

Program 5.13

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



```
#include <iostream>
using namespace std;
int main()
{
char pekerja, barululus;
cout << "Jawablah pertanyaan dibawah ini\n";
cout << "dengan jawaban Y untuk Ya atau ";
cout << "T untuk Tidak\n";
cout << "Apakah Anda Bekerja? ";
cin >> pekerja;
cout << "telahkan Anda selesai sekolah ";
cout << "dalam dua tahun ini? ";
cin >> barululus;
if (pekerja == 'Y')
{
if (barululus == 'Y')
{
cout << "Kwalitas anda khusus ";
cout << "tertarik ?\n";
}
}
getch();
```

```
return 0;
}
```
Keluaran program diatas setelah diberi masukan adalah:
Jawablah pertanyaan dibawah ini
dengan jawaban Y untuk Ya atau T untuk Tidak
Apakah Anda Bekerja? Y
Karena pernyataan IF pertama
kondisinya melaksanakan masalah
kedua, baik variabel pekerja dan
barululus harus diatur ke 'Y' supaya
pesan mengenai pengguna dengan
kualifikasi khusus dicetak sesuai
dengan tingkat suku bunga. Jenis
pernyataan nested if yang baik untuk
pilihan yang kecil dengan
mengkategorisasikan data. Satusatunya
cara untuk menjalankan
program pernyataan IF yang kedua
adalah dengan ekspresi kondisional
yang pertama harus benar. Namun
kadang ada fitur yang tidak
dikehendaki atau sering dikenal
sebagai kesalahan dalam sebuah
program. Jika seorang pengguna

memasukkan 'N' (atau karakter selain
'Y') sebagai pekerja atau barululus,
program tidak mencetak pesan yang
memberitahukan bahwa mereka tidak
memenuhi syarat.
Pernyataan lain yang dapat
digunakan untuk mengatasi masalah
ini digambarkan dalam program
dibawah. Program dibawah ini
merupakan sebuah pernyataan
NESTED IF.
Program 5.14
```
#include<conio.h>
#include <iostream>
using namespace std;
int main()
{
char pekerja, barululus;
cout << "Jawablah pertanyaan dibawah ini\n";
cout << "dengan jawaban Y untuk Ya atau ";
```

```
cout << "T untuk Tidak\n";
cout << "Apakah Anda Bekerja? ";
cin >> pekerja;
cout << "telahkan Anda selesai sekolah ";
cout << "dalam dua tahun ini? ";
cin >> barululus;
if (pekerja == 'Y')
{ // Nested if
if (barululus == 'Y')
{
cout << "Kwalitas anda khusus ";
cout << "Anda tertarik\n";
}
else
{
cout << "Anda harus Lulus dari ";
cout << "sekolah paling lama 2 tahun\n";
cout << "dan berkwalitas.\n";
}
}
else
{
cout << "Anda harus menjadi pekerja yang berkwalitas.\n";
}
getch();
```



```
return 0;
}
```

Keluaran program diatas setelah diberi masukan adalah:

```
Jawablah pertanyaan dibawah ini
dengan jawaban Y untuk Ya atau T untuk Tidak
Apakah Anda Bekerja? Y
telahkan Anda selesai sekolah dalam dua tahun ini? Y
Kwalitas anda khusus Anda tertarik
contoh lainnya misalnya pada
suatu perusahaan yang memberikan
komisi kepada para selesman
dengan ketentuan sebagai berikut:
□□ Bila salesman dapat menjual
barang hingga Rp. 20.000 ,- , akan
diberikan uang jasa sebesar Rp.
10.000 ditambah dengan uang
komisi Rp. 10% dari pendapatan
yang diperoleh hari itu.
□□ Bila salesman dapat menjual
```

barang diatas Rp. 20.000 ,- , akan diberikan uang jasa sebesar Rp. 20.000 ditambah dengan uang komisi Rp. 15% dari pendapatan yang diperoleh hari itu.

□□ Bila salesman dapat menjual barang diatas Rp. 50.000 ,- , akan diberikan uang jasa sebesar Rp. 30.000 ditambah dengan uang komisi Rp. 20% dari pendapatan yang diperoleh hari itu.

Perhatikan contoh program dengan menggunakan pernyataan Nested-If dibawah:

Program 5.15

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;
int main()
{
float pendptan, jasa=0, komisi=0, total=0;
cout<<"Pendapatan Hari ini Rp. ";
cin>>pendptan;
if (pendptan >= 0 && pendptan <= 200000)
{
jasa=10000;
komisi=0.1*pendptan;
}
else
```



```
{
if(pendptan<=500000)
{
jasa=20000;
komisi=0.15*pendptan;
}
else
{
jasa=30000;
komisi=0.2*pendptan;
}
}
/* menghitung total */
total = komisi+jasa;
```

```
cout<<"Uang Jasa Rp. "<<jasa<<endl;
cout<<"Uang Komisi Rp. "<<komisi<<endl;
cout<<"============================"<<endl;
cout<<"Hasil Total Rp. "<<total<<endl;
getch();
return 0;
}
```

Keluaran program diatas adalah:

```
Pendapatan Hari ini Rp. 5600
Uang Jasa Rp. 10000
Uang Komisi Rp. 560
============================
Hasil Total Rp. 10560
```

5.8. Operator Logika

Selain operator relasional, bahasa pemrograman umumnya mendukung tambahan operator yaitu AND, OR, dan NOT. Operator And, Or, dan Not dikenal dengan nama operator logika. Dengan menggunakan operator ini dua atau lebih test perbandingan dapat dilakukan. Pada bagian sebelumnya telah ditulis sebuah program tes dengan dua kondisi pada pernyataan IF. Dalam bagian ini ditulis program yang menggunakan operator logis untuk menggabungkan dua atau lebih menjadi satu kalimat relasional. Tabel berikut merupakan daftar operator logika pada bahasa C++.

Tabel 5.5. Operator Logika

OPERATOR	ARTI	PENJELASAN
&&	AND	Menghubungkan dua ekpresi menjadi satu. Kedua ekpresi tersebut harus benar supaya hasilnya benar
||	OR	Menghubungkan dua ekpresi menjadi satu. Hasil akan menjadi benar jika salah satu atau keduanya ada yang benar.
!	NOT	Operator yang melakukan kebalikan dari ekpresi. Jika ekpresi benar maka hasil salah atau sebaliknya.

5.8.1. Operator &&

Operator && yang dikenal sebagai operator logika. Dua kalimat diperlukan sebagai operands dan

membuat kalimat benar, jika kedua sub-kalimatnya benar. Berikut dibawah ini adalah contoh program pernyataan IF yang menggunakan operator &&:

```
if (suhu < 20 && waktu > 12)
cout << "suhu telah mencapai level yang berbahaya";
```

Perhatikan bahwa kedua kalimat yang diANDkan bersama adalah kalimat yang lengkap untuk mengevaluasi benar atau salah. Pertama suhu <20 dievaluasi untuk menghasilkan hasil benar atau salah. Kemudian menit> 12 dievaluasi untuk menghasilkan hasil benar atau salah. Kemudian kedua hasil tersebut diANDkan sehingga akan mendapatkan hasil akhir untuk seluruh ekspresi. Pernyataan cout hanya akan dijalankan jika suhu kurang dari 20 dan waktunya lebih besar daripada 12. Jika salah satu penghubung teresebut salah, seluruh ekspresi yang dihasilkan juga salah dan pernyataan cout tidak dijalankan. Tabel dibawah meruapkan tabel kebenaran untuk operator &&. Kebenaran tabel berisi semua kemungkinan kombinasi nilai yang mungkin dimiliki oleh dua kalimat, Sebagaimana dalam tabel menunjukkan, kedua sub-ekspresi harus benar untuk operator && sehingga akan menghasilkan nilai yang benar pula.

Table 5.6. Logika AND

EKPRESI	NILAI KEBENARAN	LOGIKA
Salah && salah	Salah	0
Salah && benar	Salah	0
Benar && salah	Salah	0
Benar && benar	benar	1

Operator && dapat digunakan untuk mempermudah programprogram lain yang akan

menggunakan pernyataan NESTED
IF. Program dibawah ini adalah versi
Program sebelumnya yang kemudian
ditulis ulang dengan operator logika.
Perhatikan program dibawah ini:

Program 5.16

```
#include<conio.h>
#include <iostream>
using namespace std;
int main()
{
char pekerja, barululus;
cout << "Jawablah pertanyaan dibawah ini\n";
cout << "dengan jawaban Y untuk Ya atau ";
cout << "T untuk Tidak\n";
cout << "Apakah Anda Bekerja? ";
cin >> pekerja;
cout << "telahkan Anda sekolah ";
cout << "dalam dua tahun ini? ";
cin >> barululus;
if (pekerja == 'Y' && barululus == 'Y') // menggunakan &&
{ // logical operator
cout << "Kwalitas anda spesial ";
cout << "dan anda tertarik.\n";
}
else
{
cout << "anda harus bekerja dan mempunyai\n";
cout << "lulusan dari sekolah SMK dalam\n";
cout << "waktu paling lama 3 tahun.\n";
}
getch();
```



```
return 0;
}
```

Keluaran program diatas setelah diberi masukan adalah

```
Jawablah pertanyaan dibawah ini
dengan jawaban Y untuk Ya atau T untuk Tidak
Apakah Anda Bekerja? T
telahkan Anda sekolah dalam dua tahun ini? Y
anda harus bekerja dan mempunyai
lulusan dari sekolah SMK dalam
waktu paling lama 3 tahun.
```

5.8.2. Operator ||

Operator || yang dikenal sebagai

operator logika OR. Dua kalimat yang diperlukan sebagai operand akan membuat kalimat benar bila salah satu dari sub-kalimat yang benar. Berikut dibawah ini adalah contoh sebuah pernyataan IF yang menggunakan operator ||

```
if (suhu < 20 || suhu > 100)
cout << "suhu dalam level yang berbahaya.";
```

Pernyataan cout akan dijalankan jika suhu kurang dari 20 ATAU suhu lebih besar dari 100. Jika salah satunya benar, maka seluruh ekspresi akan menghasilkan keluaran benar dan pernyataan cout akan dijalankan. Setelah cout dijalankan maka akan menghasilkan keluaran "suhu dalam level yang berbahaya". Table dibawah ini merupakan table kebenaran operator OR

Table 5.7. Logika OR

EKPRESI	NILAI KEBENARAN LOGIKA
Salah && salah	Salah 0
Salah && benar	Benar 1
Benar && salah	Benar 1
Benar && benar	benar 1

Semua ekspresi pada operator OR akan menjadi salah jika semua dari sub-kalimat salah, tetapi akan benar jika salah satu sub-kalimatnya benar. Tidak peduli apakah salah satu sub-ekspresinya salah atau benar. Program dibawah ini akan melakukan tes untuk mengetahui orang yang memenuhi syarat untuk mendapatkan pinjaman. Program ini akan menentukan apakah pelanggan mempunyai gaji setidaknya $35,000

per tahun atau telah bekerja selama lebih dari lima tahun. Perhatikan program dibawah ini:

Program 5.17

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

```
#include <iostream>
using namespace std;
int main()
{
double income;
int years;
cout << "Apakah ini masukan tahunan Anda? ";
cin >> income;
cout << "berapa banyak anda bekerja setiap tahunnya "
<< "Pekerjaan anda sekarang? ";
cin >> years;
if (income >= 35000 || years > 5) // Uses the || logical operator
cout << "Kwalifikasi Anda.\n";
else
{
cout << "Anda akan mendapatkan gaji paling tidak $35,000 atau lebih\n";
cout << "setelah bekerja lebih dari lima tahun.\n";
}
getch();
return 0;
}
```

Keluaran program diatas setelah diberi masukan adalah
Apakah ini masukan tahunan Anda? 30000
berapa banyak anda bekerja setiap tahunnya Pekerjaan anda sekarang? 5
Anda akan mendapatkan gaji paling tidak $35,000 atau lebih
setelah bekerja lebih dari lima tahun.

5.8.3. Operator !

Operator ! akan melakukan
operasi logika NOT. Dibutuhkan
sebuah operand atau sebeliknya
seperti benar atau salah. Dengan
kata lain, jika ungkapan itu benar,
maka operator ! akan membalik
menjadi salah, dan jika ekspresi
salah, maka akan dibalik menjadi
benar. Dibawah ini adalah suatu

pernyataan IF yang meggunakan
operator !:

```
if (!(suhu > 100))
cout << "anda dibawah suhu
maksimum.\n";
```

pada potongan program diatas,
ekspresi (suhu> 100) yang akan diuji
benar atau salah. Maka operator !

akan diterapkan pada nilai. Jika ekspresi (suhu> 100) itu benar, maka operator ! akan membalik menjadi salah. Jika hal tersebut salah, maka operator ! akan kembali benar. Tabel dibawah merupakan table kebenaran operator !

.

Tabel 5.8. operator !

EKPRESI	HASIL EKPRESI	LOGIKA
!	salah	benar 1
!	benar	Salah 0

Program dibawah sama dengan program sebelumnya, yaitu akan melakukan operasi dengan operator !. Pernyataan IF tersebut menggunakan operator ! untuk menentukan apakah pengguna tidak membuat rekening di bank yang setidaknya ada $ 35.000 atau belum mempunyai pekerjaan yang lebih dari lima tahun. Perhatikan program dibawah ini:

Program 5.18

```
#include <iostream>
using namespace std;
int main()
{
double income;
int years;
cout << "Apakah ini masukan tahunan Anda? ";
cin >> income;
cout << "berapa banyak anda bekerja setiap tahunnya "<< "Pekerjaan anda?
";
cin >> years;
if (!(income >= 35000 || years > 5)) // menggunakan operator !
```



```
{
cout << "Anda akan mendapatkan gaji paling tidak $35,000 atau
lebih\n";
cout << "setelah bekerja lebih dari lima tahun.\n";
}
else
cout << "Kwalifikasi Anda.\n";
return 0;
```

}
Keluaran program diatas sama dengan program sebelumnya

5.8.3. Variabel Boolean dan Operator !

Fitur yang menarik dari variabel Boolean adalah nilainya dapat diuji. Misalnya moreData adalah sebuah variabel Boolean. Kemudian pengujiannya adalah:

if (moreData == true)

can be written simply as

if (moreData)

and the test

if(moreData == false)

can be written simply as

if(!moreData)

Program diatas biasanya diselesaikan dengan menggunakan operator !.

5.9. Operator Kondisional

Operator kondisional merupakan operator kuat dan unik. Operator tersebut menyediakan sebuah metode sedernana yang mengekpresikan pernyataan IF/ELSE. Operator ini terdiri dari tanda tanya (?) dan titik dua (:). Untuk lebih jelasnya mengenai operator ini perhatikan ekpresi dibawah inI:

ekspresi ? ekspresi : ekspresi;

disini diberikan sebuah contoh pernyataan yang menggunakan operator kondisional

x < 0 ? y = 10 : z = 20;

pernyataan diatas sering disebut dengan *conditional expression dan terdiri dari tiga* sub-expressions ang dipisahkan dengan menggunakan tanda Tanya dan symbol titik dua (:). Sebuah ekspresi diatas < 0, y = 10, dan z = 20 dapat diuraikan sebagai berikut:

Gambar 5.6 conditional expression

Sebuah Ekspresi kondisional yang diatas melakukan operasi yang sama seperti halnya pernyataan IF/ELSE.

Perhatikan pernyataan dibawah ini:
if (x < 0)
y = 10;
else
z = 20;
Bagian dari ekspresi kondisional sebelum tanda tanya adalah ekspresi yang akan diuji. Hal ini merupakan kalimat dalam kurung seperti pada sebuah pernyataan IF. Jika ekspresi benar, bagian dari pernyataan antara tanda tanya (?) dan titik dua (:) akan dijalankan. Jika tidak, bagian setelah : yang dijalankan. Gambar dibawah ini menggambarkan aturan yang diakukan oleh tiga sub-ekspresi adalah sebagai berikut:
Ekspresi pertama yang akan diuji
Ekspresi kedua yang akan dieksekusi jika ekspresi pertama benar
Ekspresi ketiga yang akan dieksekusi jika ekspresi pertama salah
X< 0 ? Y=10 : Z = 20
Gambar 5.7. aturan yang diakukan oleh tiga sub-ekspresi
Anda dapat meletakkan tanda kurung di sekitar sub-ekspresi, seperti contoh berikut ini:
(x < 0) ? (y = 10) : (z = 20);
Dalam menggunakan nilai ekspresi dalam bahasa C++ yang perlu diingat, bahwa pada bahasa C++ semua ekspresi memiliki nilai, dan hal ini termasuk ekspresi kondisional. Jika sub-ekspresi pertama benar, maka nilai dari ekspresi kondisional adalah nilai kedua sub-ekspresi tersebut. Sebaliknya Jika hal tersebut bukan berarti nilai sub-ekspresi yang ketiga. Berikut dibawah ini adalah contoh sebuah pernyataan yang menggunakan nilai ekspresi kondisional
a = x > 100 ? 0 : 1

Nilai akan ditetapkan baik 0 atau 1, tergantung apakah x lebih besar dari 100. Pernyataan tersebut dapat dinyatakan dalam pernyataan IF/ELSE sebagai berikut:

X< 0 ? Y=10 : Z = 20


```
if (x > 100)
a = 0;
else
a = 1;
```

Program dibawah dapat digunakan untuk membantu seorang konsultan untuk menghitung biaya. Konsultasn tersebut meminta bayaran $ 50,00 per jam, tetapi minimum selama lima jam, jika kurang dari 5 jam maka tidak akan dibayar. Operator kondisional digunakan dalam pernyataan tersebut untuk memastikan bahwa jumlahnya tidak kurang atau dibawah lima. Untuk lebih jelasnya perhatikan program dibawah ini

Program 5.19

```
#include<conio.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
const double PAY_RATE = 50.0;
double hours, charges;
cout << "Berapa jam anda bekerja? ";
cin >> hours;
hours = hours < 5 ? 5 : hours; // Conditional operator
charges = PAY_RATE * hours;
cout << fixed << showpoint << setprecision(2);
cout << "Besar gaji anda adalah $" << charges << endl;
getch();
return 0;
}
```

Program Output with Example Input Shown in Bold

Berapa jam anda bekerja? 5

Besar gaji anda adalah $250.00

5.10. Statement SWITCH

Statement SWITCH juga berfungsi sama dengan IF. Perintah SWITCH sama dengan perintah CASE OF dalam PASCAL. Dimana sintaksnya menggunakan:

```
switch (variabel)
{
case value1 : statement1;
break;
case value2 : statement2;
break;
..
default : statement; /* optional */
break;
}
```

Perhatikan contoh program menggunakan statement switch adalah sebagai berikut:

Program 5.20

```
#include <conio.h>
#include <iostream>
using namespace std;
void main()
{
int bil;
cout << "Masukkan bilangan : ";
cin >> bil
switch (bil)
{
case 1 : cout << "Anda memasukkan bil. satu";
break;
case 2 : cout << "Anda memasukkan bil. dua";
break;
case 3 : cout << "Anda memasukkan bil. tiga";
break;
default: cout << "Anda memasukkan bil selain 1, 2, dan 3";
break;
}
}
```


Selanjutnya coba kalian hapus semua break program di atas dan kalian jalankan. Apa yang terjadi? Keanehan akan muncul. Mengapa?

5.11. Pernyataan SWITCH … CASE

Cara lain untuk menangani pengambilan keputusan dalam sebuah program adalah dengan menggunakan Statement switch…. case, yang mampu menangani sejumlah kondisi dari satu variabel. switch .... case serupa dengan If ... Then ..... ElseIf, tetapi lebih efisien apabila percabangan bergantung kepada satu kondisi saja. Dengan menggunakan switch .... case sebagai pengganti dari If ... Then ... Else If, akan membuat program menjadi lebih sederhana. Format penggunaan switch .... case:

```
switch nama_variabel
{
case nilai_1:{
Perintah yang akan dieksekusi jika memenuhi nilai_1;
break;}
case nilai_2:{
Perintah yang akan dieksekusi jika memenuhi nilai_2;
break;}
default:{
Perintah yang akan dieksekusi jika tidak memenuhi semua;
exit(0);}
}
```

Struktur Select Case dimulai dengan kata Select Case dan diakhiri dengan kata End Select. nama_variabel dapat diganti dengan variabel, properti, atau ekpresi lain yang akan dijadikan sebagai kondisi. Sedangkan nilai_1, nilai 2 dapat diganti dengan angka, string atau nilai lain yang berkaitandengan nama_variabel. Jika salah satu nilai sesuai dengan variabel, maka pernytaan di bawah kata case akan dijalankan dan Visual Basic akan melanjutkan mengeksekusi program setelah End. Anda dapat menggunakan kata case sebanyak mungkin dalam struktur select... case, dan anda juga dapat menyertakan lebih dari satu nilai

pada kata case.
Jika anda menyertakan banyak
nilai setelah case, pisahkan dengan
tanda koma.

```
Select Case intAge
Case 5 : lblTitle.Caption = "Kindergarten"
Case 6 : lblTitle.Caption = "1st Grade"
```



```
Case 7 : lblTitle.Caption = "2nd Grade"
Case 8 : lblTitle.Caption = "3rd Grade"
Case 9 : lblTitle.Caption = "4th Grade"
Case 10 : lblTitle.Caption = "5th Grade"
Case 11 : lblTitle.Caption = "6th Grade"
Case Else: lblTitle.Caption = "Advanced"
End Select
```

Bentuk dari SWITCH - CASE
merupakan pernyataan yang
dirancangan khusus untuk
menangani pengambilan keputusan
yang melibatkan sejumlah atau
banyak alternatif penyelesaian.
Pernyataan SWITCH - CASE ini
memiliki kegunaan sama seperti IF –
ELSE bertingkat, tetapi
penggunaannya untuk memeriksa
data yang bertipe karakter atau
integer. Bentuk penulisan perintah ini
sebagai berikut :

```
switch (ekspresi integer atau karakter )
{
case konstanta - 1 :
… perintah;
… perintah;
break;
case konstanta - 2 :
… perintah;
… perintah;
break;
……
……
default :
… perintah;
… perintah;
}
```

Setiap cabang akan dijalankan

jika syarat nilai konstanta tersebut dipenuhi dan default akan dijalankan jika semua cabang diatasnya tidak terpenuhi. Pernyataan *break* menunjukan bahwa perintah siap keluar dari *switch*. Jika pernyataan ini tidak ada, maka program akan diteruskan kecabang – cabang yang lainnya. Perhatikan contoh program dibawah ini menggunakan Switch-Case, untuk menentukan alat yang dikehendaki, seperti alat olahraga, alat elektronik atau alat masak.

Program 5.21


```
include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;
main()
{
char kode;
cout<<"Masukkan Kode Barang [A..C] : ";
cin>>kode;
switch(kode)
{
case 'A' :
cout<<"Alat Olah Raga";
break;
case 'B' :
cout<<"Alat Elelktronik";
break;
case 'C' :
cout<<"Alat Masak";
break;
default:
cout<<"Anda Salah Memasukan kode";
break;
}
getch();
}
```

Keluaran program diatas adalah:

```
Masukkan Kode Barang [A..C] : B
Alat Elelktronik
```

Program diatas juga bisa ditulis

dengan program dibawah ini, dimana kelebihan program dibawah bisa membaca baik kode huruf kecil maupun huruf besar. Pada program diatas hanya bisa membaca kode huruf besar saja. Kedua-duanya sama-sama menggunakan switchcase. Jika dalam memasukan kodenya salam maka akan muncul pesan “Anda Salah Memasukan kode”. Untuk lebih jelasnya perhatikan program dibawah ini:

Program 5.22


```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
using namespace std;
main()
{
char kode;
cout<<"Masukkan Kode Barang [A..C] : ";
cin>>kode;
switch(kode)
{
case 'A' :
case 'a' :
cout<<"Alat Olah Raga";
break;
case 'B' :
case 'b' :
Cout<<"Alat Elelktronik";
break;
case 'C' :
case 'c' :
cout<<"Alat Masak";
break;
default:
cout<<"Anda Salah Memasukan kode";
break;
}
getch();
return 0;
}
```

Struktur kondisi switch....case....

default digunakan untuk penyeleksian kondisi dengan cukup banyak kemungkinan yang terjadi. Struktur ini akan melaksanakan salah satu dari beberapa pernyataan ‘case’ tergantung nilai kondisi yang ada di dalam switch. Selanjutnya proses diteruskan hingga ditemukan pernyataan ‘break’. Jika tidak ada nilai pada case yang sesuai dengan nilai kondisi, maka proses akan diteruskan kepada pernyataan yang ada di bawah ‘default’. Bentuk umum dari struktur kondisi ini adalah:


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

Pada program dibawah merupakan program dengan menggunakan switch, dimana program tersebut digunakan untuk menentukan jumlah hari. Perhatikan program dibawah ini:

Program 5.23

```
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
int AngkaBulan,Tahun, JumlahHari;
cout<<"Angka Bulan : ";
cin >> AngkaBulan;
cout <<"Tahun : ";
cin >> Tahun;
switch (AngkaBulan) {
case 1:
```

```
case 3:
case 5:
case 7:
case 8:
case 10:
case 12: JumlahHari = 31;break;
case 4:
case 6:
case 9:
```

```
case 11 : JumlahHari = 30;break;
case 2 : if (Tahun % 4 == 0) JumlahHari = 29;
else JumlahHari = 28;break;
}
cout<<"Jumlah hari "<< JumlahHari;
getch();
return 0;
}
```
Keluaran program diatas adalah:
Angka Bulan : 4
Tahun : 1990
Jumlah hari 30
Program dibawah merupakan
program yang digunakan untuk
menentukan bilangan Genap atau
Ganjil dengan case:
Program 5.24
```
#include <conio.h>
#include <iostream>
using namespace std;
main() {
int Op1, Op2, Pilih;
float Hasil;
cout << "Operand 1: ";
cin >> Op1;
cout << "Operand 2: ";
cin >> Op2;
cout << "1. Operator +\n";
cout << "2. Operator  - \n";
cout << "3. Operator *\n";
cout << "4. Operator /\n";
cout << "Pilih Operator : ";
cin >> Pilih;
switch (Pilih) {
case 1 : {Hasil = Op1 + Op2; break;}
```

```
case 2 : {Hasil = Op1  -  Op2; break;}
case 3 : {Hasil = Op1 * Op2; break;}
case 4 : {if (Op2 != 0)
cout <<Op1/Op2;
```

```
else
cout <<"error";
break;}
}
printf("Hasil dari perhitungan tersebut adalah %f",Hasil);
getch();
return 0;
}
```
Keluaran program diatas adalah sebagai berikut:
Operand 1: 1
Operand 2: 3
1. Operator +
2. Operator -
3. Operator *
4. Operator /
Pilih Operator : 3
Hasil dari perhitungan tersebut adalah 3.000000
Dari beberapa contoh program
diatas, maka dapat diketahui bentuk
bahasa umum mengenai pernyataan
If–Else adalah sebagai berikut:
If Kondisi Then
Perintah yang akan dieksekusi jika kondisi bernilai true
ELSE
Perintah yang akan dieksekusi jika kondisi bernilai false
End If

5.12. IF...THEN, IF...THEN...ELSE dan Nested IF

Struktur IF, kita dapat membuat
algoritma-algoritma yang melakukan
perintah berdasarkan kondisi tertentu.
Perintah akan dilakukan jika hanya
jika ekspresi boolean bernilai true
(jika ekspresi boolean bernilai false
maka perintah tidak akan dikerjakan).
IF <ekspresi boolean> then
Perintah_1
Else
Perintah_2
Endif
IF <ekspresi boolean 1> then

Perintah 1
Statement Kendali Bahasa Pemrorgraman
***2008 | Direktorat Pembinaan SMK* 219**
Else if <ekspresi boolean 2> then
Perintah 2
Else if <ekspresi boolean n> then
Perintah n
Else Perintah n+1
Endif
Program statement kondisi IFELSE
untuk mencari bilangan Genap
dan Ganjil. Perhatikan contoh
program dibawah ini:
Program 5.25

```
#include <conio.h>
#include <iostream>
using namespace std;
main() {
int Bil;
cout << "masukan bilangan : ";
cin >> Bil;
if (Bil % 2 == 0)
{
cout << "bilangan genap";
}
else
{
cout << "bilangan ganjil";
}
getch();
return 0;
}
```

Keluaran program diatas adalah
masukan bilangan : 5
bilangan ganjil
Program dibawah ini digunakan untuk menghitung atau mencari bilangan
terbesar.
Program 5.26

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

Bahasa Pemrograman Statement Kendali
220 Direktorat Pembinaan SMK | 2008

```
using namespace std;
main() {
int A, B, C,maks;
cout << "A : ";
```

```
cin >> A;
cout << "B : ";
cin >> B;
cout << "C : ";
cin >> C;
if ((A>B) && (A>C))
maks = A;
else if ((B>A) && (B>C))
maks = B;
else
maks = C;
cout <<"bilangan terbesar adalah " << maks;
getch();
return 0;
}
```

Keluaran program diatas adalah sebagai berikut:

```
A : 40
B : 56
C : 59
bilangan terbesar adalah 59
```

5.13. Aplikasi Pernyataan IF pada Menu

Menu adalah layar yang menampilkan beberapa pilihan sehingga pengguna dapat memilih. Sebagai contoh pada sebuah program yang dapat memberikan menu pilihan berikut ini:

1. Menambahkan nama ke dalam daftar.
2. Hapus nama dari daftar.
3. Mengubah nama dalam daftar.
4. Mencetak daftar.
5. Keluar dari program ini.

Pengguna dapat memilih salah satu operasi dengan memasukkan nomor. Misalnya memasukan nomer 4, sehingga akan menyebabkan daftar dicetak, dan memasukkan 5 akan menyebabkan keluar program. struktur IF/ELSE IF dapat digunakan untuk membuat menu seperti diatas. Setelah pengguna memasukkan nomor, maka nomor akan digunakan untuk membandingkan pilihan yang tersedia dan melaksanakan sebuah pernyataan yang melakukan operasi.

program dibawah digunakan untuk
menghitung biaya keanggotaan
disebuah klub kesehatan. Klub
memiliki tiga paket keanggotaan
pilihan: keanggotaan standar

dewasa, anggota anak, dan anggota
orang senior.
Program ini menyajikan menu
yang memungkinkan pengguna untuk
memilih paket yang diinginkan,
kemudian menghitung biaya
keanggotaannya. Untuk lebih
jelasnya perhatikan program dibawah
ini:
Program 5.27

```
#include <conio.h>
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
int choice, months;
double charges;
// Display menu pilihan
cout << "\t\tMenu Anggota Klub Kesehatan\n\n";
cout << "1. Keanggotaa Standard Dewasa\n";
cout << "2. Keanggotaan Anak\n";
cout << "3. Keanggotaan Masyarakat Senior\n";
cout << "4. Keluar Program\n\n";
cout << "Masukan Pilihan Anda: ";
cin >> choice;
cout << fixed << showpoint << setprecision(2);
if (choice == 1)
{
cout << "Untuk berapa Bulan? ";
cin >> months;
charges = months * 40.00;
cout << "Total pembayaranya adalah: $" << charges << endl;
}
else if (choice == 2)
{
cout << "Untuk berapa Bulan? ";
cin >> months;
charges = months * 20.00;
```

```
cout << "Total pembayaranya adalah: $" << charges << endl;
}
else if (choice == 3)
```

```
{
cout << "Untuk berapa Bulan? ";
cin >> months;
charges = months * 30.00;
cout << "Total pembayaranya adalah: $" << charges << endl;
}
else if (choice != 4)
{
cout << "Pilihan yang valid adalah antara 1 sampai 4\n";
cout << "Program akan memilih lagi dari salah satu menu diatas\n";
}
getch();
return 0;
}
```
Keluaran program diatas seperti dibawah ini:

```
Menu Anggota Klub Kesehatan
1. Keanggotaa Standard Dewasa
2. Keanggotaan Anak
3. Keanggotaan Masyarakat Senior
4. Keluar Program
Masukan Pilihan Anda: 3
Untuk berapa Bulan? 3
Total pembayaranya adalah: $90.00
```

Dalam program tersebut memungkinkan pengguna yang tidak valid dapat membuat pilihan. Jika angka lain seperti 1, 2, 3, atau 4 dimasukkan, maka akan ada pesan kesalahan yang dicetak. Kesalahan ini dikenal sebagai masukan validasi (*input validation).*

5.14. Soal Latihan

Jawablah soal latihan dibawah ini dengan baik dan benar.

1. Apa yang dimaksud dengan statement
2. Sebutkan beberapa operator relasionaldalam bahasa c++
3. Jelaskan cara kerja pernyataan if pada bahasa c++
4. Apa perbedaan antara pernyataan if dangan if-else
5. Apa yang dimaksud dengan if-else majemuk
6. Apa yang dimaksud dengan nested if
7. Sebutkan beberapa operator logika
8. Tuliskan perintah yang digunakan oleh pernyataan switch

Prosedur dan Subroutin Bahasa Pemrorgraman