MODUL
PEMROGRAMAN DASAR

Fery Updi,S.Kom

PROGRAM STUDI KEAHLIAN TEKNIK KOMPUTER DAN INFORMATIKA
PAKET KEAHLIAN : RPL, TKJ, MM
KURIKULUM 2013

PETA KOMPETENSI	
Kompetensi Dasar	**Materi Pokok**
KD 3.1 - 4.1	Algoritma Pemrograman
KD 3.2 – 4.2	Algoritma Percabangan
KD 3.3 – 4.3	Algoritma Perulangan
KD 3.4 – 4.4	Bahasa Pemrograman
KD 3.5 – 4.5	Tipe Data, Variabel, Operator, dan Ekspresi
KD 3.6 – 4.6	Struktur Kontrol Percabangan
KD 3.7 – 4.7	Struktur Kontrol Perulangan
KD 3.8 – 4.8	Pengembangan Algoritma Aplikasi
KD 3.1, 3.2, 4.1, 4.2	Operasi Aritmatika dan Logika
KD 3.3, 3.4, 3.5, 4.3, 4.4, 4.5	Array
KD 3.6, 3.7, 4.6, 4.7	Operasi String dan Konversi Data
KD 3.8, 4.8,	Pointer
KD 3.9, 3.10, 3.11, 3.12, 4.9, 4.10, 4.11, 4.12	Fungsi
KD 3.13, 3.14, 4.13, 4.14	Pencarian dan Pengurutan Data
KD 3.15, 4.15	Pengembangan Aplikasi

BAB I KEGIATAN BELAJAR

Kegiatan Belajar 1 : Algoritma Pemrograman

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 1 ini siswa diharapkan dapat :

1) Memahami Konsep Algoritma
2) Memahami Struktur Algoritma
3) Memahami Algoritma menggunakan bahasa natural Algoritma
4) Memahami Pseudocode
5) Memahami Flowchart dan penggunaan Tool Flowcahrt
6) Pengenalan Variabel
7) Memahami Pengenalan tipe data
8) Memahami Pengenalan operator

B. Uraian Materi

1. Pengantar Algoritma Pemrograman

Belajar memprogram adalah belajar tentang strategi pemecahan masalah, metodologi dan sistematika pemecahan masalah tersebut kemudian menuangkannya dalam suatu notasi yang disepakati bersama.

"lebih bersifat pemahaman persoalan, analisis, sintesis"

Belajar bahasa pemrograman adalah belajar memakai suatu bahasa, aturan sintaks (tatabahasa), setiap instruksi yang ada dan tata cara pengoperasian kompilator atau interpreter bahasa yang bersangkutan pada mesin tertentu.

Jadi :

"BELAJAR MEMPROGRAM""
TIDAK SAMA DENGAN
"BELAJAR BAHASA PEMROGRAMAN"

1.1 Algortima dan pemrograman Dasar

Perangko dari Rusia pada Gambar di samping ini bergambar seorang pria dengan nama Muhammad bin Musa al-Khwarizmi. Bagi kalian yang sedang berkecimpung dalam dunia komputer maka seharusnya mengetahui siapa orang di samping ini. Dia adalah seorang ilmuwan Islam yang karya karyanya dalam bidang matematika, astronomi, astrologi dan geografi banyak menjadi dasar perkembangan ilmu modern. Dan dari namanya istilah yang akan kita pelajari dalam bab ini muncul. Dari Al-Khawarizmi kemudian berubah menjadi **algorithm** dalam Bahasa Inggris dan diterjemahkan menjadi **algoritma** dalam bahasa Indonesia.

1.2 Definisi Algortima

1. **Algoritma** adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis.

Algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki tingkat kerumitan yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan suatu masalah membutuhkan tingkat kerumitan yang tinggi.

1.3 Struktur Algortima

Perhatikan algoritma sederhana berikut :

Jika seseorang ingin mengirim surat kepada kenalannya di tempat lain, langkah yang harus dilakukan adalah:

1. Menyiapkan Peralatan Tulis
2. Menulis surat
3. Surat dimasukkan ke dalam amplop tertutup
4. Amplop ditempeli perangko secukupnya.
5. Pergi ke Kantor Pos terdekat untuk mengirimkannya

Algoritma menghitung luas persegi panjang:

1. Masukkan panjang (P)
2. Masukkan lebar (L)
3. Luas P * L
4. Tulis Luas

Pembuatan algoritma mempunyai banyak keuntungan di antaranya:

a) Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.
b) Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
c) Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

Beberapa hal yang perlu diperhatikan dalam membuat algoritma:

a) Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah.Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
b) Tidak ada notasi yang baku dalam penulisan teks algoritma seperti notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.
c) Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini dikarenakan teks algoritma tidak sama dengan teks program. Namun, supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.
d) Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalannya. *Pseudocode* adalah kode yang mirip dengan instruksi kode program sebenarnya.
e) Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa pemrograman.

f) Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasikan ke dalam notasi bahasa pemrograman

Perhatikan algoritma sederhana berikut :
Algoritma menghitung luas segitiga

1. Start
2. Baca data alas dan tinggi.
3. Luas adalah alas kali tinggi kali 0.5
4. Tampilkan Luas
5. Stop

Penjelasan :
Algoritma di atas adalah algoritma yang sangat sederhana, hanya ada lima langkah. Pada algoritma ini tidak dijumpai perulangan ataupun pemilihan. Semua langkah dilakukan hanya satu kali.
Sekilas algoritma di atas benar, namun apabila dicermati maka algoritma ini mengandung kesalahan yang mendasar, yaitu tidak ada pembatasan pada nilai data untuk alas dan tinggi.

Hasil perbaikan algoritma perhitungan luas segitiga

1. Start
2. Baca data alas dan tinggi.
3. Periksa data alas dan tinggi, jika nilai data alas dan tinggi lebih besar dari nol maka lanjutkan ke langkah ke 4 jika tidak maka stop
4. Luas adalah alas kali tinggi kali 0.5
5. Tampilkan Luas
6. Stop

Dari penjelasan di atas dapat diambil kesimpulan pokok tentang algoritma. Pertama, algoritma harus benar. Kedua algoritma harus berhenti, dan setelah berhenti, algoritma memberikan hasil yang benar.

Contoh : Algoritma Berangkat Sekolah
Mulai
    Bangun dari tempat tidur
    Mandi Pagi
    Sarapan Pagi
    Pergi Ke Sekolah
    Cari Ruang Kelas
    Masuk kelas untuk Belajar
Selesai

Beda Algoritma dan Program ?
Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakanbahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman.

Program = Algoritma + Bahasa (Struktur Data)

Penerjemah Bahasa Pemrograman
Untuk menterjemahkan bahasa pemrograman yang kita tulis maka diperlukan *Compiler* dan *interpreter*.

Compiler adalah suatu program yang menterjemahkan bahasa program (Source code) ke dalam bahasa obyek (object code) secara keseluruhan program.

Interpreter berbeda dengan ***Compiler, interpreter*** menganalisis dan mengeksekusi setiap baris dari program secara keseluruhan. Keuntungan dari interpreter adalah dalam eksekusi yang bisa dilakukan dengan segera. Tanpa melalui tahap kompilasi, untuk alas an ini interpreter digunakan pada saat pembuatan program berskala besar.

Perbedaan Compiler dan interpreter.

Compiler	***Interpreter***
Menterjemahkan secara keseluruhan	Menterjemahkan Instruksi per instruksi
Bila terjadi kesalahan kompilasi maka source program harus diperbaiki dan dikompilasi ulang	Bila terjadi kesalahan interprestasi dapat Diperbaiki
Dihasilkan Object program	Tidak dihasilkan obyek program
Dihasilkan Executable program	Tidak dihasilkan Executable program
Proses pekerjaan program lebih cepat	Proses pekerjaan program lebih lambat
Source program tidak dipergunakan hanya bila untuk perbaikan saja	Source program terus dipergunakan
Keamanan dari program lebih terjamin	Keamanan dari program kurang terjamin

1.4 Jenis-Jenis Bahasa Pemrograman
- Bahasa Pemrograman Tingkat rendah (Bahasa mesin, Biner)
- Bahasa Pemrograman Tingkat tinggi

Contoh-contoh Bahasa Pemrograman yang ada :
1. Prosedural : Algol, Pascal, Fortran, Basic, Cobol, C
2. Fungsional : LOGO, APL, LISP
3. Deklaratif : Prolog

Object oriented murni: Smalltalk, Eifel, Java, PHP

Cara penulisan algortima
Ada tiga cara penulisan algoritma, yaitu :

1. Structured English (SE)
SE merupakan alat yang cukup baik untuk menggambarkan suatu algoritma. Dasar dari SE adalah Bahasa Inggris, namun kita dapat memodifikasi dengan Bahasa Indonesia sehingga kita boleh menyebutnya sebagai Structured Indonesian (SI).

"SE atau SI lebih tepat untuk menggambarkan suatu algoritma yang akan dikomunikasikan kepada pemakai perangkat lunak"

2. Pseudocode
Pseudocode adalah kode yang mirip dengan instruksi kode program sebenarnya. Pseudocode didasarkan pada bahasa pemrograman yang sesungguhnya seperti BASIC,

FORTRAN atau PASCAL. Pseudocode yang berbasis bahasa PASCAL merupakan pseudocode yang sering digunakan.

"Pseudo berarti imitasi atau tiruan atau menyerupai, sedangkan code menunjuk pada kode program"

Contoh Pseudocode :

1. Start
2. READ alas, tinggi
3. Luas = 0.5 * alas * tinggi
4. PRINT Luas
5. Stop

Pada Contoh diatas tampak bahwa algoritma sudah sangat mirip dengan bahasa BASIC. Pernyataan seperti READ dan PRINT merupakan keyword yang ada pada bahasa BASIC yang masing-masing menggantikan kata "baca data" dan "tampilkan". Dengan menggunakan pseudocode seperti di atas maka proses penterjemahan dari algoritma ke kode program menjadi lebih mudah.

1.5 Membuat Alur Logika Pemograman

A. Penyajian atau Penulisan Algoritma

Penyajian algoritma secara garis besar bisa dalam 2 bentuk penyajian yaitu tulisan dan gambar. Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu (misalnya bahasa Indonesia atau bahasa Inggris) dan pseudocode.

Pseudocode adalah kode yang mirip dengan kode pemrograman yang sebenarnya seperti Pascal, atau C, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram. Sedangkan algoritma disajikan dengan gambar, yaitu dengan Flowchart

B. Flowchart (Diagram Alir)

Flowchart atau bagan alir adalah skema/bagan (chart) yang menunjukkan aliran (flow) di dalam suatu program secara logika.

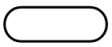
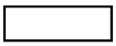
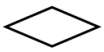
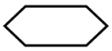
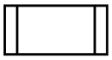
Flowchart merupakan alat yang banyak digunakan untuk menggambarkan algoritma dalam bentuk notasi-notasi tertentu. Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta pernyataannya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan antara proses digambarkan dengan garis penghubung. Dengan menggunakan flowchart akan memudahkan kita untuk melakukan pengecekan bagian-bagian yang terlupakan dalam analisis masalah. Di samping itu flowchart juga berguna sebagai fasilitas untuk berkomunikasi antara pemrogram yang bekerja dalam tim suatu proyek.

Walaupun tidak ada kaidah-kaidah yang baku dalam penyusunan flowchart, namun ada beberapa anjuran:

1) Hindari pengulangan proses yang tidak perlu dan logika yang berbelit sehingga jalannya proses menjadi singkat.
2) Jalannya proses digambarkan dari atas ke bawah dan diberikan tanda panah untuk memperjelas.

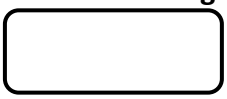
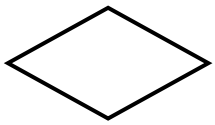
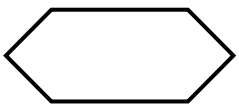
3) Sebuah flowchart diawali dari satu titik START dan diakhiri dengan END.

Berikut merupakan beberapa contoh simbol flowchart yang disepakati oleh dunia pemrograman:

Keterangan	Lambang
Mulai/selesai (terminator)	
Aliran data	
Input/Output	
Proses	
Percabangan (Decision)	
Pemberian nilai awal suatu variabel (Preparation)	
Memanggil prosedur/fungsi (Call)	
Connector (di halaman yg sama)	
Off page Connector (halaman lain)	

Penjelasan lebih lanjut :

Simbol-simbol bagan* alir program (*Flowchart)

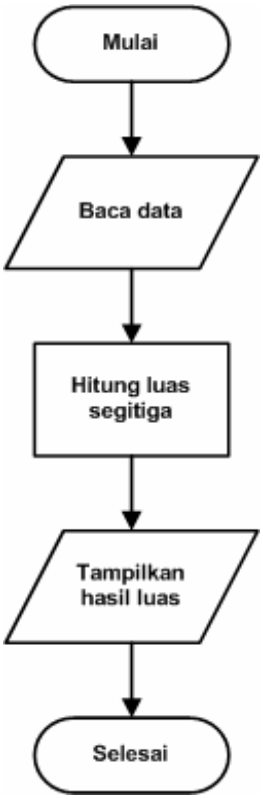
-  Notasi Membuat algoritma sederhana untuk menyelesaikan permasalahan menggunakan bahasa natural, flowchart dan pseudocode
-  Notasi ini disebut Data yang digunakan untuk mewakili data input atau output atau menyatakan operasi pemasukan data dan pencetakan hasil
-  Notasi ini disebut Process yang digunakan untuk mewakili suatu proses.
-  Notasi ini disebut Decision yang digunakan untuk suatu pemilihan, penyeleksian kondisi di dalam suatu program
-  Notasi ini disebut Preparation yang digunakan untuk memberi nilai awal, nilai akhir, penambahan/pengurangan bagi suatu variabel counter.

-  Notasi ini disebut Predefined Process yang digunakan untuk menunjukkan suatu operasi yang rinciannya ditunjukkan ditempat lain (prosedur, sub-prosedur, fungsi)

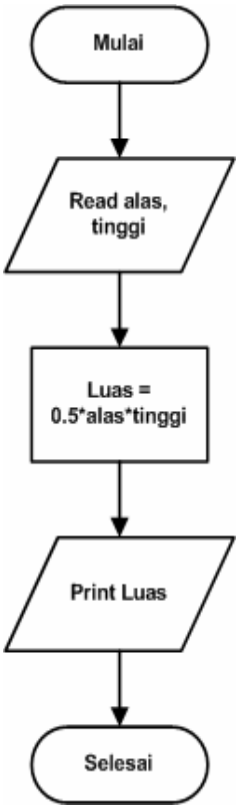
-  Notasi ini disebut Connector yang digunakan untuk menunjukkan sambungan dari flowchart yang terputus di halaman yang sama atau halaman berikutnya.

- Notasi ini disebut Arrow yang digunakan untuk menunjukkan arus data atau aliran data dari proses satu ke proses lainnya.

Contoh program Flowchart



Bagan alir logika program



Bagan alir program komputer terinci

C. Struktur Dasar Algoritma

Algoritma berisi langkah-langkah penyelesaian suatu masalah. Langkah-langkah tersebut dapat berupa runtunan aksi (sequence), pemilihan aksi (selection), pengulangan aksi (iteration) atau kombinasi dari ketiganya. Jadi struktur dasar pembangunan algoritma ada tiga, yaitu:

1. Struktur Runtunan / Beruntun : Digunakan untuk program yang pernyataannya sequential atau urutan.
2. Struktur Pemilihan / Percabangan : Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.
3. Struktur Perulangan : Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang.

1. Struktur Algoritma Runtunan / Berurutan :

Ada tiga struktur dasar yang digunakan dalam membuat algoritma yaitu struktur berurutan (sequencing), struktur pemilihan/keputusan/percabangan (branching) dan struktur pengulangan (looping). Sebuah algoritma biasanya akan menggabungkan ketiga buah struktur ini untuk menyelesaikan masalah.

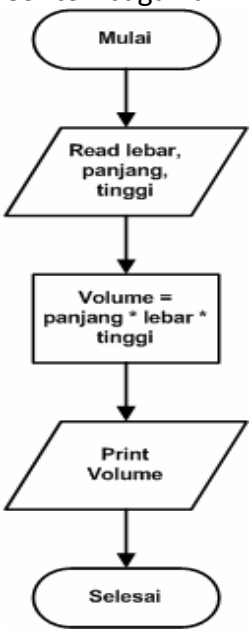
Struktur berurutan dapat kita samakan dengan mobil yang sedang berjalan pada jalur lurus yang tidak terdapat persimpangan seperti tampak pada Gambar disamping Mobil tersebut akan melewati kilometer demi kilometer jalan sampai tujuan tercapai. *Struktur berurutan terdiri satu atau lebih instruksi.*

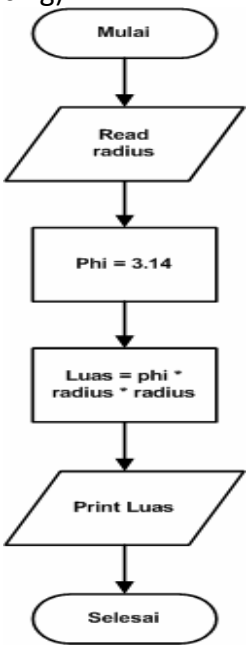
Tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya, yaitu sebuah instruksi dieksekusi setelah instruksi sebelumnya selesai dieksekusi. Urutan instruksi menentukan keadaan akhir dari algoritma. Bila urutannya diubah, maka hasil akhirnya mungkin juga berubah.

Menurut Goldshlager dan Lister (1988) struktur berurutan mengikuti ketentuan-ketentuan sebagai berikut:

- Tiap instruksi dikerjakan satu persatu
- Tiap instruksi dilaksanakan tepat sekali, tidak ada yang diulang
- Urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana yang tertulis di dalam algoritmanya
- Akhir dari instruksi terakhir merupakan akhir algoritma.

Contoh bagan alir logika program berurutan (sequencing)





1. Struktur Algoritma Percabangan

Sebuah program tidak selamanya akan berjalan dengan mengikuti struktur berurutan, kadang-kadang kita perlu merubah urutan pelaksanaan program dan menghendaki agar pelaksanaan program meloncat ke baris tertentu. Peristiwa ini kadang disebut sebagai percabangan/pemilihan atau keputusan. Hal ini seperti halnya ketika mobil/motor berada dalam persimpangan

Pada struktur percabangan, program akan berpindah urutan pelaksanaan jika suatu kondisi yang disyaratkan dipenuhi. Pada proses seperti ini simbol flowchart Decision harus digunakan. Simbol decision akan berisi pernyataan yang akan diuji kebenarannya. Nilai hasil pengujian akan menentukan cabang mana yang akan ditempuh.

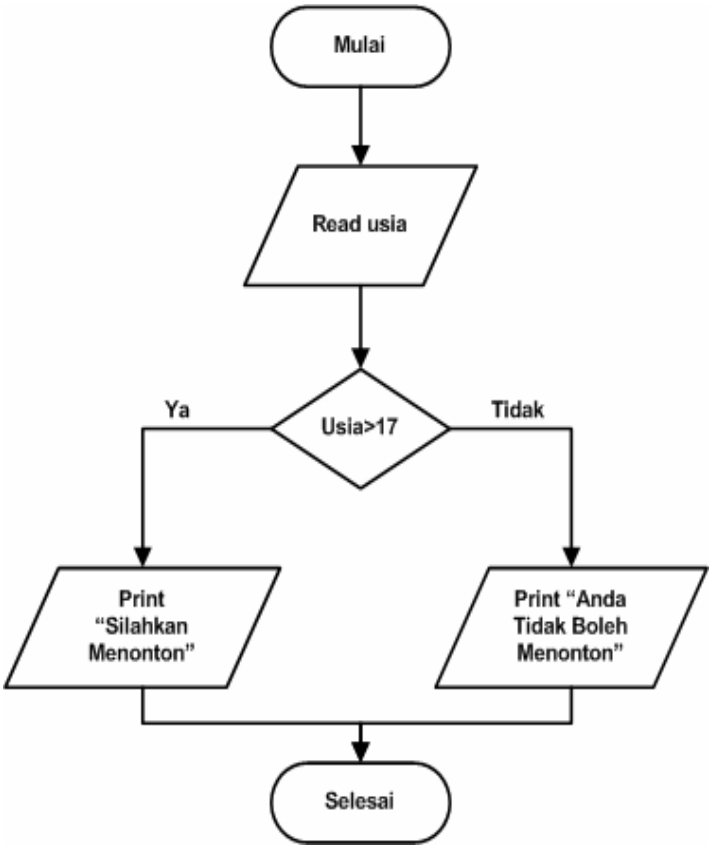
Contoh Struktur percabangan untuk masalah batasan umur.

Sebuah aturan untuk menonton sebuah film tertentu adalah sebagai berikut, jika usia penonton lebih dari 17 tahun maka penonton diperbolehkan dan apabila kurang dari 17 tahun maka penonton tidak diperbolehkan nonton. Buatlah flowchart untuk permasalahan tersebut.

Penyelesaian:

Permasalahan diatas merupakan ciri permasalahan yang menggunakan struktur percabangan. Hal ini ditandai dengan adanya pernyataan jika ..maka ...(atau If ... Then dalam Bahasa Inggris.

Bagan alir logika (Flowchart) penyelesaian masalah nonton film



2. Struktur Algoritma Perulangan / Pengulangan

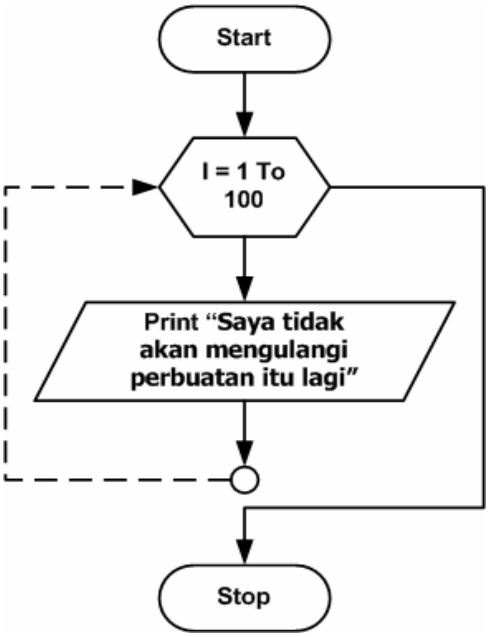
Dalam banyak kasus seringkali kita dihadapkan pada sejumlah pekerjaan yang harus diulang berkali.Salah satu contoh yang gampang kita jumpai adalah balapan mobil

Struktur pengulangan terdiri dari dua bagian :

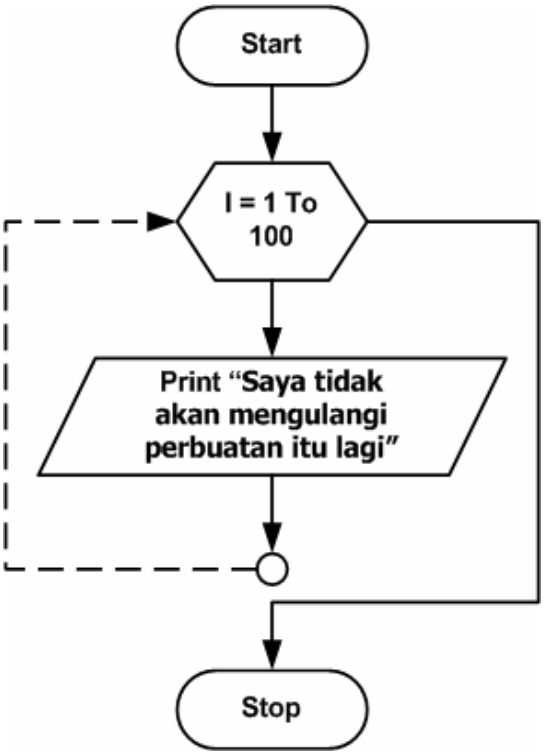
- Kondisi pengulangan, yaitu syarat yang harus dipenuhi untuk melaksanakan pengulangan. Syarat ini biasanya dinyatakan dalam ekspresi Boolean yang harus diuji apakah bernilai benar (true) atau salah (false)
- Badan pengulangan (loop body), yaitu satu atau lebih instruksi yang akan diulang

Pada struktur pengulangan, biasanya juga disertai bagian *inisialisasi* dan bagian *terminasi.* ***Inisialisasi*** adalah instruksi yang dilakukan sebelum pengulangan dilakukan pertama kali. Bagian insialisasi umumnya digunakan untuk memberi nilai awal sebuah variable. Sedangkan ***terminasi*** adalah instruksi yang dilakukan setelah pengulangan selesai dilaksanakan. Ada beberapa bentuk pengulangan yang dapat digunakan, masing-masing dengan syarat dan karakteristik tersendiri. Beberapa bentuk dapat dipakai untuk kasus yang sama, namun ada bentuk yang hanya cocok untuk kasus tertentu saja.
Pemilihan bentuk pengulangan untuk masalah tertentu dapat mempengaruhi kebenaran algoritma. Pemilihan bentuk pengulangan yang tepat bergantung pada masalah yang akan diprogram.

Bagan alir logika (flowchart) untuk mencetak pernyataan sebanyak 100 kali



Bagan alir logika (Flowchart) untuk mencetak anggota suatu himpunan.

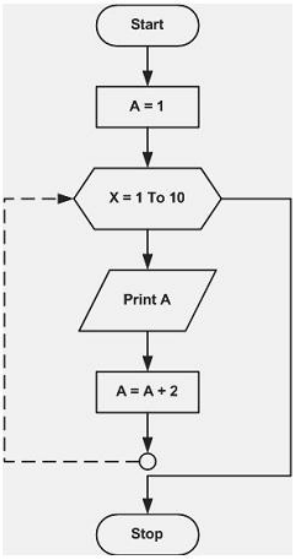


- **Struktur pengulangan dengan For**

Pengulangan dengan menggunakan For, merupakan salah teknik pengulangan yang paling tua dalam bahasa pemrograman. Hampir semua bahasa pemrograman menyediakan metode ini, meskipun sintaksnya mungkin berbeda. Pada struktur For kita harus tahu terlebih dahulu seberapa banyak badan loop akan diulang. Struktur ini menggunakan sebuah variable yang biasa disebut sebagai loop s counter, yang nilainya akan naik atau turun selama proses pengulangan.

Contoh :
Diketahui sebuah himpunan A yang beranggotakan bilangan 1, 3, 5, .., 19. Buatlah flowchart untuk mencetak anggota himpunan tersebut.

Penyelesaian:
Pada contoh ini, kita mencoba menentukan hasil dari sebuah flowchart . Bagaimana menurut kalian jawabannya? Marilah kita uraikan jalannya fowchart tersebut. Pada flowchart, setelah Start, kita meletakkan satu proses yang berisi pernyataan A = 1. Bagian inilah yang disebut inisialisasi . Kita memberi nilai awal untuk A = 1. Variabel counter-nya adalah X dengan nilai awal 1 dan nilai akhir 10, tanpa increment (atau secara default increment-nya adalah 1). Ketika masuk ke badan loop untuk pertama kali maka akan dicetak langsung nilai variabel A. Nilai variabel A masih sama dengan 1. Kemudian proses berikutnya adalah pernyataan A = A + 2. Pernyataan ini mungkin agak aneh, tapi ini adalah sesuatu yang pemrograman. Arti dari pernyataan ini adalah gantilah nilai A yang lama dengan hasil penjumlah nilai A lama ditambah 2. Sehingga A akan bernilai 3. Kemudian dilakukan pengulangan yang ke-dua. Pada kondisi ini nilai A adalah 3, sehingga yang tercetak oleh perintah print adalah 3. Baru kemudian nilai A kita ganti dengan penjumlahan A + 2. Nilai A baru adalah 5. Demikian seterusnya. Sehingga output dari flowchart ini adalah 1,3, 5, 7, .., 19.



- **Struktur pengulangan dengan While**

Pada pengulangan dengan For, banyaknya pengulangan diketahui dengan pasti karena nilai awal (start) dan nilai akhir (end) sudah ditentukan diawal pengulangan. Bagaimana jika kita tidak tahu pasti harus berapa kali mengulang? Pengulangan dengan While merupakan jawaban dari permasalahan ini. Seperti halnya For, struktur pengulangan dengan While juga merupakan struktur yang didukung oleh hampir semua bahasa pemrograman namun dengan sintaks yang berbeda.
Struktur While akan mengulang pernyataan pada badan loop sepanjang kodisi pada While bernilai benar. Dalam artian kita tidak perlu tahu pasti berapa kali diulang. Yang penting sepanjang kondisi pada While dipenuhi maka pernyataan pada badan loop akan diulang.
Penyelesaian: Perhatikan Gambar. bisakah kalian menentukan hasil dari flowchart tersebut? Perhatikan tahapan eksekusi flowchart berikut ini.

- Pada flowchart ini ada dua variabel yang kita gunakan yaitu A dan B. Kedua variabel tersebut kita inisialisasi nilai awalnya (A = 1 dan B = 0) sebelum proses loop terjadi. Variabel A adalah variabel counter.
- Pada simbol decision, nilai A akan diperiksa apakah memenuhi kondisi (<b10). Jika Ya maka perintah berikutnya dieksekusi, jika tidak maka program akan berhenti. Pada awal eksekusi ini kondisi akan terpenuhi karena nilai A= 1.
- Jalankan perintah Print B.
- Nilai variabel A kemudian diganti dengan nilai A lama (1) ditambah 2. Sehingga nilai variabel A baru adalah 3. Sedangkan nilai variabel B = 9 (hasil perkalian A = 3).
- Program akan berputar kembali untuk memeriksa apakah nilai variabel A masih lebih kecil dari 10. Pada kondisi ini nilai A = 3, sehingga kondisi masih terpenuhi. Kemudian langkah berulang ke langkah ke 3. Begitu seterusnya sampai nilai variabel A tidak lagi memenuhi syarat kurang dari 10.

LATIHAN

1. Buatkan Flowchart untuk mencari Luas persegi empat ?
2. Buatlah Algoritma dan Flowchart untuk menentukan kelulusan siswa ?
   Dengan ketentuan :
   - Jika Nilai >= 70 maka Lulus
   - Jika Nilai <= 70 maka Tidak Lulus

BAB II KEGIATAN BELAJAR

Kegiatan Belajar 2 : Bahasa Pemrograman

C. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 1 ini siswa diharapkan dapat :

1) Memahami Pengenalan bahasa pemrograman
2) Memahami Pengenalan tools/framework
3) Memahami instalasi tools bahasa pemrograman
4) Memahami struktur bahasa pemrograman
5) Memahami standar output dalam bahasa pemrograman
6) Memahami standar input dalam bahasa pemrograman
7) Memahami kompilasi dan eksekusi program
8) Memahami perbaikan kesalahan

D. Uraian Materi

1. Pengenalan bahasa pemrograman Bahasa C

Sejarah

Bahasa C merupakan perkembangan dari bahasa BCPL yang dikembangkan oleh Martin Richards pada tahun 1967. Selanjutnya bahasa ini memberikan ide kepada Ken Thompson yang kemudian mengembangkan bahasa yang disebut bahasa B pada tahun 1970. Perkembangan selanjutnya dari bahasa B adalah bahasa C oleh Dennis Ricthie sekitar tahun 1970-an di Bell Telephone Laboratories Inc. (sekarang adalah AT&T Bell Laboratories). Bahasa C pertama kali digunakan di computer Digital Equipment Corporation PDP-11 yang menggunakan system operasi UNIX. Hingga saat ini penggunaan bahasa C telah merata di seluruh dunia. Hampir semua perguruan tinggi di dunia menjadikan bahasa C sebagai salah satu mata kuliah wajib. Selain itu, banyak bahasa pemrograman populer seperti PHP dan Java menggunakan sintaks dasar yang mirip bahasa C. Oleh karena itu, kita juga sangat perlu mempelajarinya.

2. Kelebihan dan Kekurangan Bahasa C

Kelebihan Bahasa C

- Bahasa C tersedia hampir di semua jenis computer.
- Kode bahasa C sifatnya adalah portable dan fleksibel untuk semua jenis computer.
- Bahasa C hanya menyediakan sedikit kata-kata kunci, hanya terdapat 32 kata kunci.
- Proses executable program bahasa C lebih cepat
- Dukungan pustaka yang banyak.
- C adalah bahasa yang terstruktur
- Bahasa C termasuk bahasa tingkat menengah

Kekurangan Bahasa C

- Banyaknya Operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
- Bagi pemula pada umumnya akan kesulitan menggunakan pointer

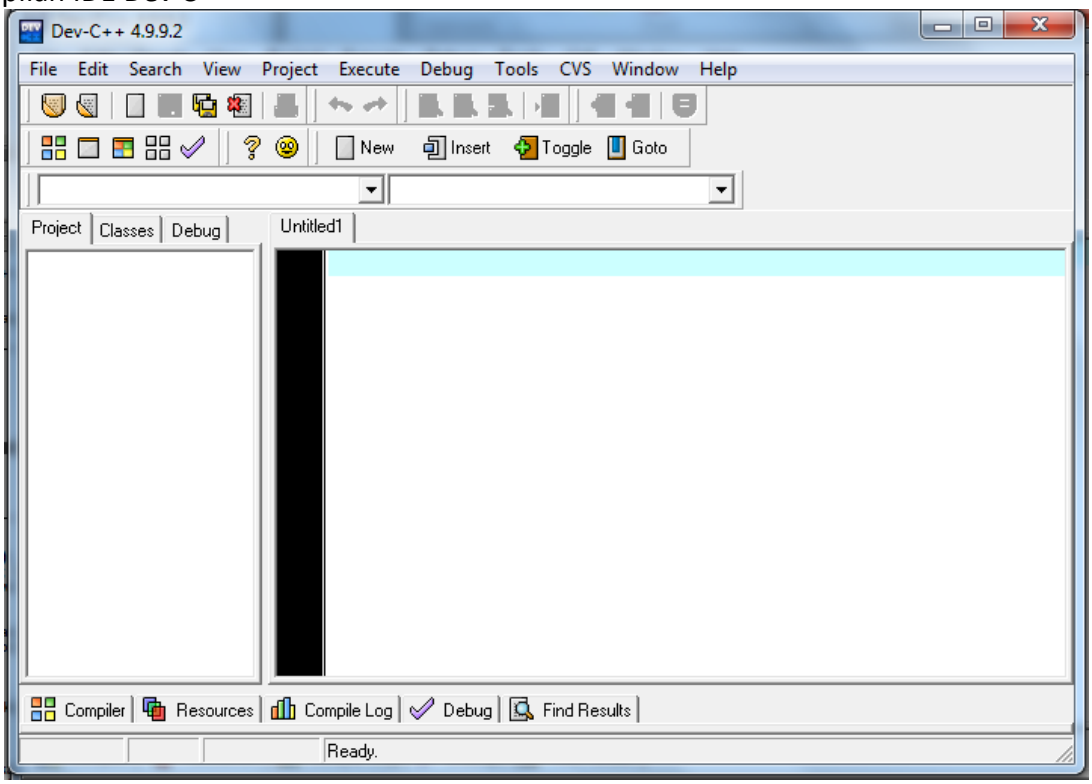
3. Pengenalan IDE Dev C++

Penjelasan

IDE merupakan singkatan dari Integrated Development Environment, merupakan Lembar kerja terpadu untuk pengembangan program. IDE dari Dev C++, dapat digunakan untuk :

- Menulis Naskah Program.
- Mengkompilasi Program (***Compile***)
- Melakukan Pengujian Program (***Debugging***)
- Mengaitkan Object dan Library ke Program (***Linking***)
- Menjalankan Program (***Running***)

Tampilan IDE Dev C++



Gambar 1.2. IDE Dev C++ 4.9.9.2

IDE pada Dev C++, terbagi menjadi 4 (empat) bagian, yaitu :

a. Menu Utama (Menubar)

Menu utama terdiri dari ; File, Edit, Search, View, Project,Execute,Debug,Tools,CVS,Windows dan Help

b. Jendela Text Edit

Tempat untuk mengetikan program dan membuat program. Jika pertama kali anda membuat program, nama file jendela editor adalah Untitled1

c. Jendela Message

Tempat untuk menampilkan pesan-pesan pada proses kompilasi dan link program.

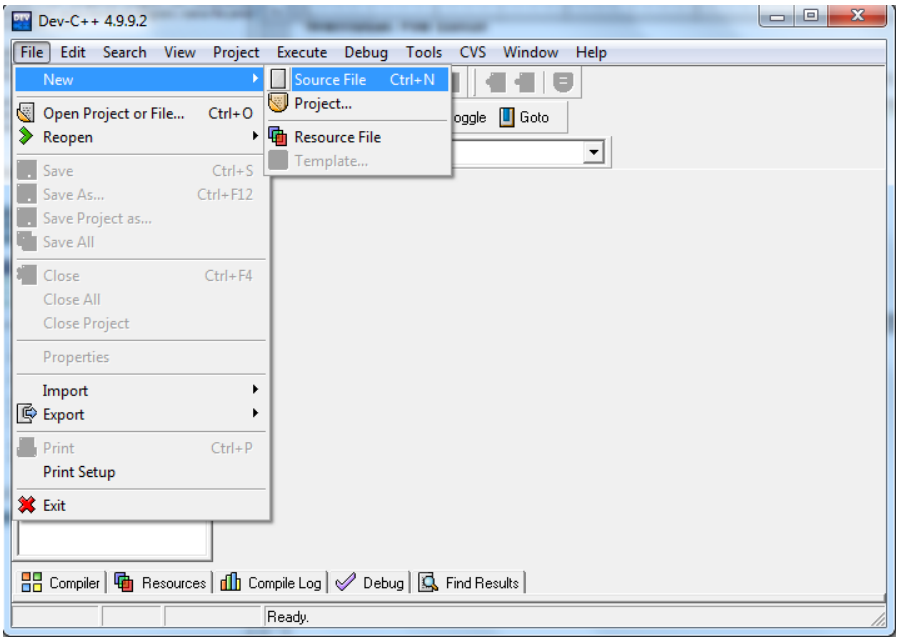
d. Baris Status

Baris dimana menampilkan keterangan-keterangan pada saat anda mengaktifkan menu bar dan sub menu.

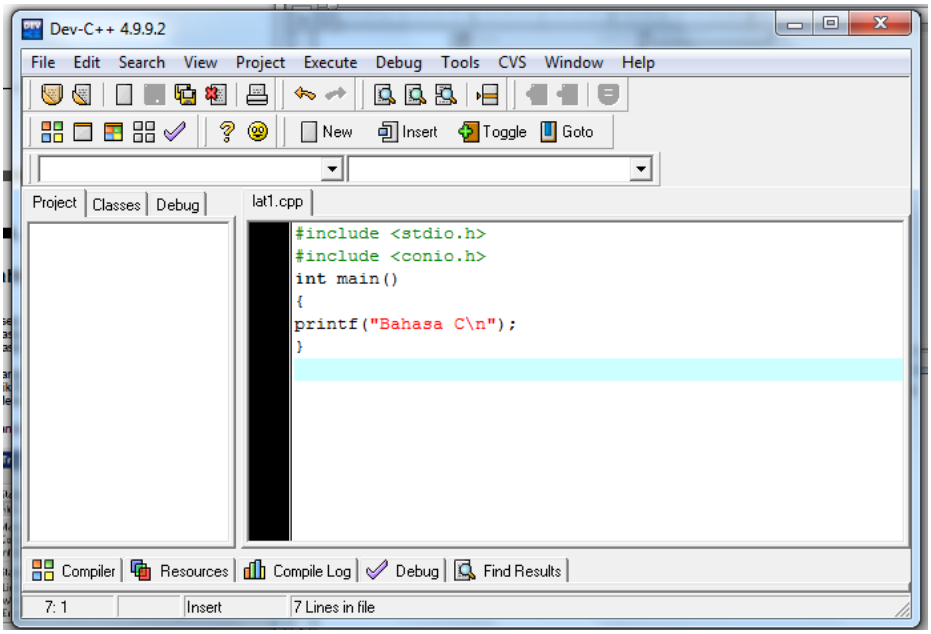
4. Membuat File Editor

Penjelasan

File Editor merupakan File Program yang dapat dikompile, dan dijalankan untuk menampilkan hasilnya serta mempunyai ekstensi **.CPP**. Cara mengaktifkannya : Klik Menu File Gambar 1.3 Klik New, Source File



Gambar 1.3 Membuat File baru



Gambar 1.4 Menulis kode program

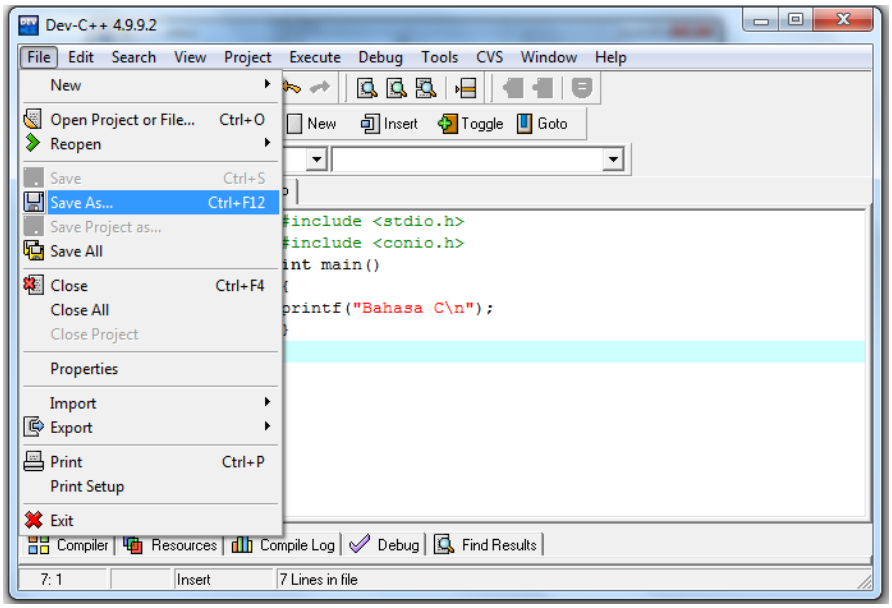
5. Menyimpan File Editor

Penjelasan

Setelah selesai mengetikan naskah program yang baru pada jendela Text Edit, maka selanjutnya disimpan dengan cara :

a. Kik Menu File Save
b. Menekan **Ctrl + S**.

Selanjutnya tampil jendela Save File As, seperti dibawah ini :



Gambar 1.5 menyimpan file

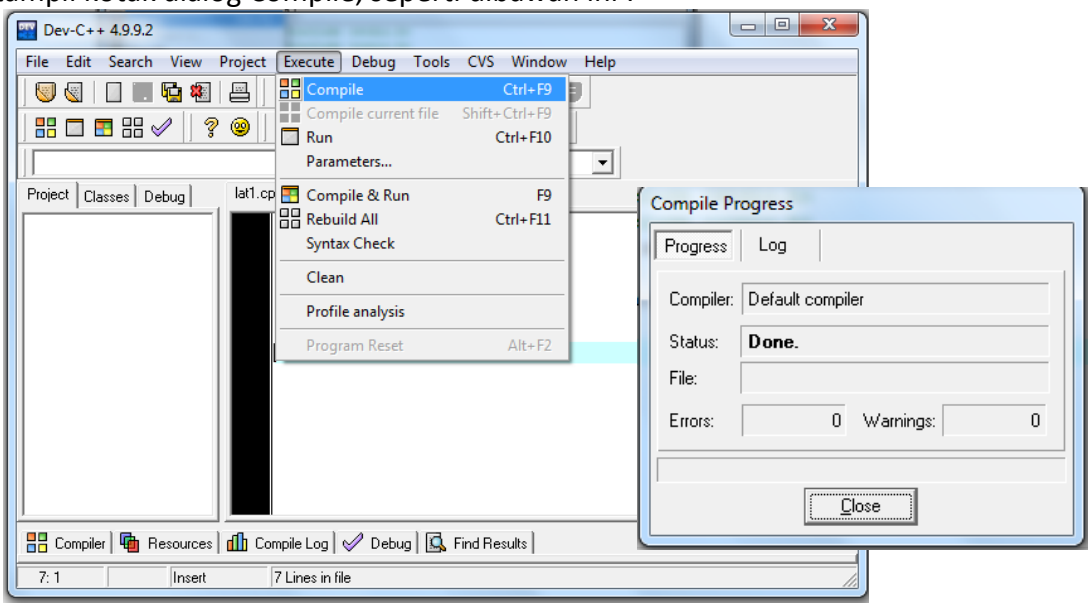
6. Menterjemahkan Program

Penjelasan

Proses Compile merupakan suatu proses menterjemahkan program dari bahasa manusia kedalam bahasa yang dimengerti oleh komputer yaitu bahasa mesin. Caranya adalah :

a. Kik Menu Project Compile
b. Menekan HotKey **Ctrl + F9**

Selanjutnya tampil kotak dialog Compile, seperti dibawah ini :



Gambar 1.6 Menterjemahkan/compile program

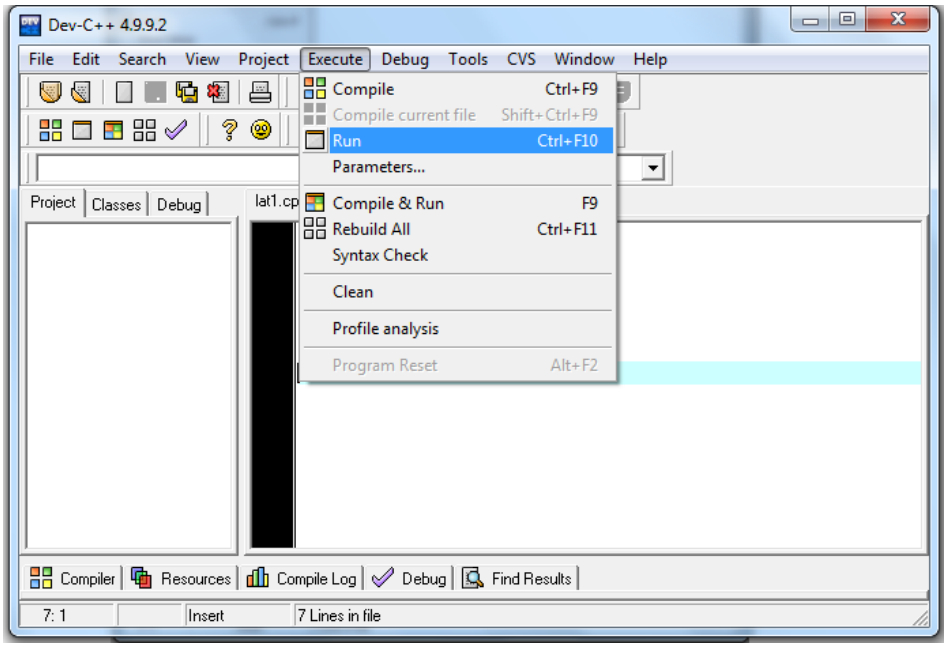
7. Menjalankan Program

Penjelasan

Proses Run merupakan suatu proses menterjemahkan program, melakukan proses linking, membuat file eksekusi (**.exe**) dan sekaligus menjalankan program. Caranya adalah :

a. Kik Menu Debug Run
b. Menekan HotKey **Ctrl + F10**

Selanjutnya tampil kotak dialog Run, seperti dibawah ini :



Gambar 1.7 Menjalankan program

Catatan

Jika program yang dijalankan tidak muncul, untuk melihat hasil compile dapat dijalankan di command prompt

8. Struktur Program C/C++

Penjelasan

Struktur program C++, sama seperti struktur program C yang terdahulu. Struktur program C++ terdiri sejumlah blok fungsi, setiap fungsi terdiri dari satu atau beberapa pernyataan yang melaksanakan tugas tertentu.

```
#include <file-include>
main()
{
    pernyataan;
}
```

Contoh-1

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

Output yang akan dihasilkan, dari program 1 diatas adalah :

```
Bahasa C
```

9. Model Memori

Penjelasan

C/C++, mempunyai enam model memori untuk program dan data. Model-model memori tersebut adalah :

- Model Tiny
- Model Small
- Model Medium
- Model Compact
- Model Large
- Model Huge

a. Model Tiny

Penjelasan

Model memori yang menyediakan jumlah memori untuk program dan data tidak lebih dari 64 Kb.

b. Model Small

Penjelasan

Model memori yang menyediakan jumlah memori untuk masing-masing program dan data tidak lebih dari 64 Kb.

c. Model Medium

Penjelasan

Model memori yang menyediakan jumlah memori untuk program tidak lebih dari 64 Kb dan data tidak lebih dari 64 K.

d. Model Compact

Penjelasan

Model memori yang menyediakan jumlah memori untuk program lebih dari 64 Kb dan data tidak lebih dari 64 K.

e. Model Large

Penjelasan

Model memori yang menyediakan jumlah memori untuk program dan data lebih dari 64 K.

f. Model Huge

Penjelasan

Model memori yang menyediakan jumlah memori untuk menyimpan satu jenis data.

BAB III KEGIATAN BELAJAR

Kegiatan Belajar 3 : Bahasa Pemrograman

E. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1) Memahami Tipe Data
2) Memahami Variabel
3) Memahami Operator
4) Memahami Ekspresi

F. Uraian Materi

Tipe Data, Variabel Konstanta, Operator, dan Ekspresi

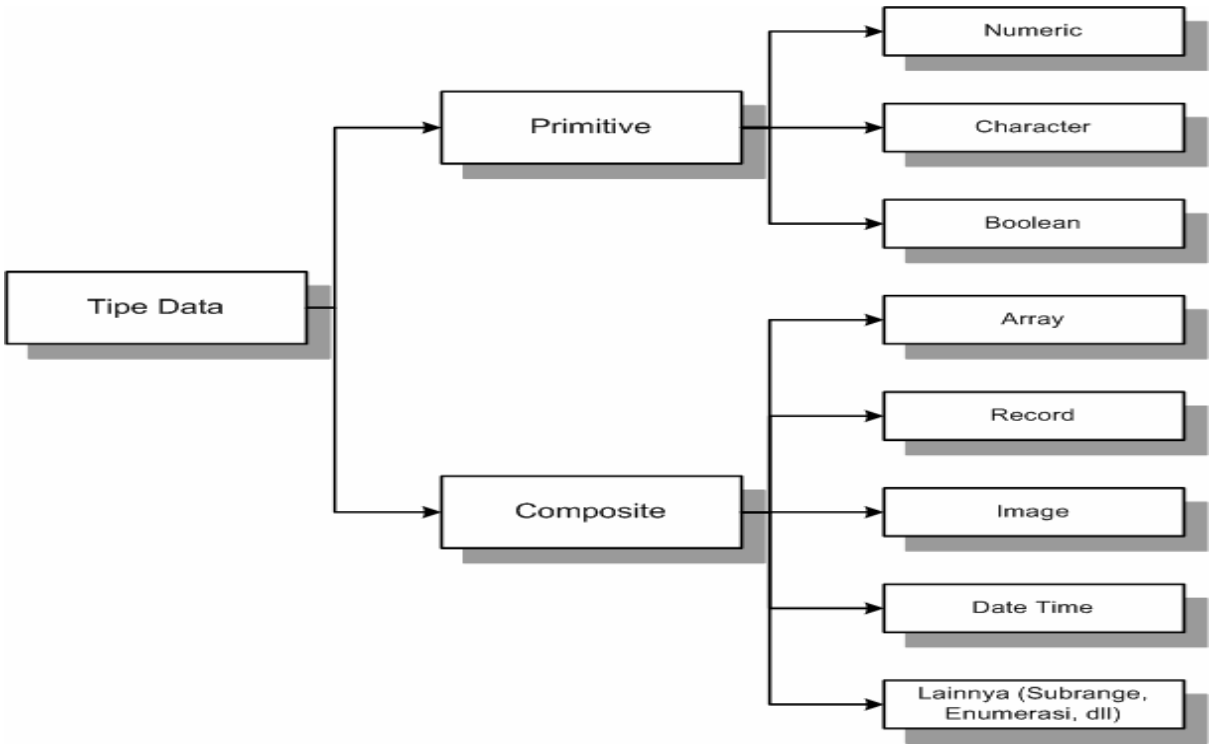
Variabel, konstanta dan tipe data merupakan tiga hal yang akan selalu kita jumpai ketika kita membuat program. Bahasa pemrograman apapun dari yang paling sederhana sampai yang paling kompleks, mengharuskan kita untuk mengerti ketiga hal tersebut.

1. Tipe Data

Tipe data adalah jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer.

Setiap variabel atau konstanta yang ada dalam kode program, sebaiknya kita tentukan dengan pasti tipe datanya. Ketepatan pemilihan tipe data pada variabel atau konstanta akan sangat menentukan pemakaian sumberdaya komputer (terutama memori komputer). Salah satu tugas penting seorang programmer adalah memilih tipe data yang sesuai untuk menghasilkan program yang efisien dan berkinerja tinggi.

Ada banyak tipe data yang tersedia tergantung jenis bahasa pemrograman yang dipakai. Namun secara umum dapat dikelompokkan seperti pada Gambar dibawah ini



Ada 2 jenis tipe data :

1. Tipe data primitive adalah tipe data dasar yang tersedia secara langsung pada suatu bahasa pemrograman.
2. Tipe data composite adalah tipe data bentukan yang terdiri dari dua atau lebih tipe data primitive.

- ***Tipe data numeric***

Tipe data numeric digunakan pada variabel atau konstanta untuk menyimpan nilai dalam bentuk bilangan atau angka. Semua bahasa pemrograman menyediakan tipe data numeric, hanya berbeda dalam jenis numeric yang diakomodasi.

Jenis yang termasuk dalam tipe data numeric antara lain :

1. integer (bilangan bulat)
2. float (bilangan pecahan).
3. tipe data Single adalah tipe data untuk bilangan pecahan dengan presisi yang terbatas
4. Tipe data Double adalah tipe data untuk bilangan pecahan dengan presisi yang lebih akurat.

Penentuan tipe data numeric untuk suatu variabel/konstanta harus sangat berhati-hati. Manual dan petunjuk pada masing-masing bahasa pemrograman pada bagian tipe data harus diperhatikan dengan seksama.

Tipe data	Ukuran memori	Jangkuan nilai	Jumlah Digit
Char	1 Byte	-128 s.d 127	
Int	2 Byte	-32768 s.d 32767	
Short	2 Byte	-2,147,435,648 s.d 2,147,435,647	
Long	4 Byte	-2,147,435,648 s.d 2,147,435,647	
Float	4 Byte	3.4 x 10-38 s.d 3.4 x 10+38	5–7
Double	8 Byte	1.7 x 10-308 s.d 1.7 x 10+308	15 – 16
Long Double	10 Byte	3.4 x 10-4932 s.d 1.1 x 10+4932	19

Tipe Data Tambahan, yang dimiliki oleh Bahasa C/C++, adalah :

Unsigned digunakan bila data yang digunakan hanya data yang positif saja

Tipe Data Tambahan

Tipe Data	Jumlah Memori	Jangkauan Nilai
Unsigned Integer	2 Byte	0 – 65535
Unsigned Character	1 Byte	0 – 255
Unsigned Long Integer	4 Byte	0 – 4,294,967,295

Contoh program bahasa C

```
//Contoh program tipe data
//Nama Programmer : .....
#include "stdio.h"
#include "conio.h"
int main()
{
int x;
float y;
char z;
```

```
double w;

x = 10;
y = 9.45;
z = 'C';
w = 3.45E+20;
printf("Nilai dari x adalah : %i\n", x);
printf("Nilai dari y adalah : %f\n", y);
printf("Nilai dari z adalah : %c\n", z);
printf("Nilai dari w adalah : %lf\n", w);
getch();
}
```

Dalam bahasa C terdapat lima tipe data dasar, yaitu :

1. char format penulisan **:** **%c**
2. int format penulisan **:** **%i, %d**
3. float format penulisan **:** **%f**
4. double format penulisan **:** **%lf**
5. void tidak bertipe

2. Variabel

Variabel adalah tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan. Setiap variabel akan mempunyai nama (identifier) dan nilai.

Contoh Nama variabel dan nilai.

username = “joni”
Nama = “Udin”
Harga = 2500
HargaTotal = 34000

Pada sebagian besar bahasa pemrograman, variabel harus dideklarasikan lebih dulu untuk mempermudah compiler bekerja. Apabila variabel tidak dideklarasikan maka setiap kali compiler bertemu dengan variabel baru Pemberian nama variabel harus mengikuti aturan yang ditetapkan oleh bahasa pemrograman yang kita gunakan. Namun secara umum ada aturan yang berlaku untuk hampir semua bahasa pemrograman. Aturan-aturan tersebut yaitu :

- Nama variabel harus diawali dengan huruf.
- Tidak boleh menggunakan spasi pada satu nama variabel. Spasi bisa diganti dengan karakter underscore (_).
- Nama variabel tidak boleh mengandung karakter-karakter khusus,seperti : .,+, -, *, /, <, >, &, (,) dan lain-lain.
- Nama variabel tidak boleh menggunakan kata-kata kunci d bahasa pemrograman

Contoh penamaan variabel

Penamaan yang benar	Penamaan yang salah
namasiswa XY12	nama siswa (salah karena menggunakan spasi) 12X (salah karena dimulai dengan angka)
harga_total	harga.total (salah karena menggunakan karakter)

JenisMotor	Jenis Motor (salah karena menggunakan spasi)
alamatRumah	for (salah karena menggunakan kata kunci bahasa pemrograman)

Jenis-jenis Variabel

1) Variabel Numerik

Variabel numerik ini dibagi menjadi menjadi 3 (tiga) macam :

- Bilangan Bulat
- Bilangan Desimal Berpresisi Tunggal atau Floating Point.
- Bilangan Desimal Berpresisi Ganda atau Double Precision.

2) Variabel Text

- Character (Karakter Tunggal)
- String (Untuk Rangkaian Karakter)

Deklarasi Variabel

Penjelasan

Adalah proses memperkenalkan variabel kepada bahasa C/C++ dan pendeklarasian tersebut bersifat mutlak karena jika tidak diperkenalkan terlebih dulu maka bahasa C/C++ tidak menerima variabel tersebut. Deklarasi Variabel ini meliputi tipe variabel, seperti : integer atau character dan nama variabel itu sendiri. Setiap kali pendeklarasian variabel harus diakhiri oleh tanda titik koma (**;**).

Bentuk penulisannya:

```
Tipe data nama variabel;
```

Contoh Deklarasi

```
char nama_siswa;
char grade;
float rata_rata ;
int nilai;
```

3. Konstanta

Konstanta adalah variabel yang nilai datanya bersifat tetap dan tidak bisa diubah. Jadi konstanta adalah juga variabel bedanya adalah pada nilai yang disimpannya. Jika nilai datanya sepanjang program berjalan tidak berubahubah, maka sebuah varibel lebih baik diperlakukan sebagai konstanta.

Sebagai contoh, jika kita membuat program perhitungan matematik yang menggunakan nilai pi (3.14159) yang mungkin akan muncul dibanyak tempat pada kode program, kita dapat membuat pi sebagai konstanta. Penggunaan konstanta pi akan lebih memudahkan penulisan kode program dibanding harus mengetikkan nilai 3.14159 berulang-ulang.

Penggunaan tipe data numeric.

Kode Program A

```
#include “stdio.h”

int main() {
        int x, z;
        float y;
```

Hasil eksekusi Program A

```
X =12
Y =2.15
Z =25
```

```
    x = 12;
    y = 2.15;
    z = x * y;
    cout << "X =" << x << endl;
    cout << "Y =" << y << endl;
    cout << "Z =" << z << endl;
    return 0; }
```

Kode Program B

```
#include <iostream>
using namespace std;
int main() {
    int x;
    float y, z;
    x = 12.8;
    y = 2.15;
    z = x * y;
    cout << "X =" << x << endl;
    cout << "Y =" << y << endl;
    cout << "Z =" << z << endl;
    return 0; }
```

Hasil eksekusi Program B

```
X =12
Y =2.15
Z =25.8
```

Kode Program C

```
#include <iostream>
using namespace std;
int main() {
    int x;
    float y, z;
    x = 12;
    y = 2.15;
    z = x * y;
    cout << "X =" << x << endl;
    cout << "Y =" << y << endl;
    cout << "Z =" << z << endl;
    return 0; }
```

Hasil eksekusi Program C

```
X =12
Y =2.15
Z =25.8
```

Selain itu, bahasa C juga menyediakan beberapa karakter khusus yang disebut karakter escape, antara lain :

- \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 (carriage return)
- \v : tabulasi vertical
- \0 : nilai kosong (null)
- \’ : karakter petik tunggal
- \” : karakter petik ganda

- \\ : karakter garis miring

- **Character**

Bersama dengan tipe data numeric, character merupakan tipe data yang paling banyak digunakan. Tipe data character kadang disebut sebagai char atau string. Tipe data string hanya dapat digunakan menyimpan teks atau apapun sepanjang berada dalam tanda petik dua ("…") atau petik tunggal ('…'). Perhatikan contoh berikut.

Contoh Deklarasi tipe data

```
# include <iostream.h>

void main ()
{
   cout << " HAi, selamat menggunakan C++ ";
}
```

- **Boolean**

Tipe data Boolean digunakan untuk menyimpan nilai True/False (Benar/Salah). Pada sebagian besar bahasa pemrograman nilai selain 0 menunjukkan True dan 0 melambangkan False. Tipe data ini banyak digunakan untuk pengambilan keputusan pada struktur percabangan dengan IF … THEN atau IF … THEN … ELSE.

Contoh :

Program Pascal

```
If Nilai >= 60 Then
        writeln('Lulus Ujian');
Else
        Writeln('Tidak lulus');
End if
```

- **Array**

Array atau sering disebut sebagai larik adalah tipe data yang sudah terstruktur dengan baik, meskipun masih sederhana. Array mampu menyimpan sejumlah data dengan tipe yang sama (homogen) dalam sebuah variabel. Setiap lokasi data array diberi nomor indeks yang berfungsi sebagai alamat dari data tersebut.

Contoh:

Penggunaan Array

```
Var
        X: array[1..100] of integer;
```

Cara mengisi data pada elemen larik dalam pemrograman adalah seperti contoh berikut :

```
X[1]:= 4;
X[2]:= 3;
X[3]:= 2;
X[4]:= 1;
```

- **Record atau Struct**

Seperti halnya Array, Record atau Struct adalah termasuk tipe data komposit. Record dikenal dalam bahasa Pascal/Delphi sedangkan Struct dikenal dalam bahasa C++. Berbeda dengan array, tipe data record mampu menampung banyak data dengan tipe data berbeda-beda (heterogen).

Sebagai ilustrasi array mampu menampung banyak data namun dengan satu tipe data yang sama, misalnya integer saja. Sedangkan dalam record, kita bisa menggunakan untuk menampung banyak data dengan tipe data yang berbeda, satu bagian integer, satu bagian lagi character, dan bagian lainnya Boolean. Biasanya record digunakan untuk menampung data suatu obyek. Misalnya, siswa memiliki nama, alamat, usia, tempat lahir, dan tanggal lahir. Nama akan akan menggunakan tipe data string, alamat bertipe data string, usia bertipe data single (numeric), tempat lahir bertipe data string dan tanggal lahir bertipe data date.

- **Image**

Image atau gambar atau citra merupakan tipe data grafik

- **Date Time**

Nilai data untuk tanggal (Date) dan waktu (Time) secara internal disimpan dalam format yang spesifik. Variabel atau konstanta yang dideklarasikan dengan tipe data Date dapat digunakan untuk menyimpan baik tanggal maupun jam. Tipe data ini masuk dalam kelompok tipe data composite karena merupakan bentukan dari beberapa tipe data.

Berikut ini contoh tipe data dalam Visual Basic.

```
Dim WaktuLahir As Date
WaktuLahir = “01/01/1997”
WaktuLahir = “13:03:05 AM”
WaktuLahir = “02/23/1998 13:13:40 AM”
WaktuLahir = #02/23/1998 13:13:40 AM#
```

- **Subrange**

Tipe data subrange merupakan tipe data bilangan yang mempunyai jangkauan nilai tertentu sesuai dengan yang ditetapkan programmer. Biasanya tipe data ini mempunyai nilai batas minimum dan nilai batas maksimum. Tipe data ini didukung dengan sangat baik dalam Delphi.

Berikut ini contoh deklarasi tipe data subrange dalam Delphi.

```
Type
    BatasIndeks = 1..20
    RentangTahun = 1950..2030
Var
    Indeks : BatasIndeks
    Tahun : RentangTahun
```

- **Enumerasi**

Tipe data ini merupakan tipe data yang mempunyai elemen-elemen yang harus disebut satu persatu dan bernilai konstanta integer sesuai dengan urutannya. Nilai konstanta integer

elemen ini diwakili oleh suatu nama variable yang ditulis di dalam kurung. Tipe data ini juga dijumpai pada Delphi dan bahasa pemrograman deklaratif seperti SQL.

Berikut ini contoh deklarasi tipe data enumerasi dalam Delphi.

```
Type
    Hari_dlm_Minggu = (Nol, Senin, Selasa, Rabu,Kamis, Jumat, Sabtu,Minggu)
    Nama_Bulan = (Nol, Januari, Pebruari, Maret,April, Mei, Juni, Juli,Agustus,
    September, Oktober, Nopember, Desember)
Var
    No_Hari : Hari_dlm_Minggu
    No_Bulan : Nama_Bulan
```

- **Object**

Tipe data object digunakan untuk menyimpan nilai yang berhubungan dengan obyek-obyek yang disediakan oleh Visual Basic, Delphi dan dan bahasa pemrograman lain yang berbasis GUI. Sebagai contoh, apabila kita mempunyai form yang memiliki control Command button yang kita beri nama Command1, kita dapat mendeklarasikan variabel sebagai berikut :

Contoh Penggunaan tipe data object.

```
    Dim A As CommandButton
    Set A = Command1
    A.Caption = "HEY!!!"
    A.FontBold = True
```

4. Operator

Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, seperti penjumlahan, pengurangan dan lain lain.

Operator mempunyai sifat sebagai berikut :

- **Unary**

Sifat Unary pada operator adalah hanya melibatkan sebuah operand pada suatu operasi aritmatik

Contoh : **-5**

- **Binary**

Sifat Binary pada operator adalah melibatkan dua buah operand pada suatu operasi aritmatik

Contoh : **4 + 8**

- **Ternary**

Sifat Tenary pada operator adalah melibatkan tiga buah operand pada suatu operasi aritmatik

Contoh : **(10 % 3) + 4 + 2**

Operator Aritmatika

Operator untuk operasi aritmatika yang tergolong sebagai operator binary adalah :

Tabel Operator Aritmatika

Operator	Keterangan	Contoh
*	**Perkalian**	**4 * 5**

/	**Pembagian**	**8 / 2**
%	**Sisa Pembagian**	**5 % 2**
+	**Penjumlahan**	**7 + 2**
-	**Pengurangan**	**6 - 2**

Tabel Operator Unary

Operator	Keterangan	Contoh
+	**Tanda Plus**	**-5**
/	**Tanda Minus**	**+6**

//Contoh penggunaan Operator Aritmatika

```
//Nama programmer :.....
#include "stdio.h"
#include "conio.h"
int main()
{
printf("Nilai dari 9 + 4 = %i\n", 9 + 4);
printf("Nilai dari 9 - 4 = %i\n", 9 - 4);
printf("Nilai dari 9 * 4 = %i\n", 9 * 4);
printf("Nilai dari 9 / 4 = %i\n", 9 / 4);
printf("Nilai dari 9 % 4 = %i\n", 9 % 4);
getch();
}
```

/* Penggunaan operator untuk
mencetak deret bilangan genap
antara 1 – 100 */

```
//Nama programmer :.....
#include "conio.h"
#include "stdio.h"
int main()
{
int bil;
for (bil=1; bil<100; bil++)
{
if(bil%2==0)
printf("%5.0i", bil);
}
getch();
}
```

Operator Penambah dan Pengurang

Masih berkaitan dengan operator pemberi nilai, Bahasa C menyediakan operator penambah dan pengurang. Dari contoh penulisan operator pemberi nilai sebagai penyederhanaannya dapat digunakan operator penambah dan pengurang.

Tabel Operator Penambah dan Pengurang

Operator	Keterangan
++	**Penambahan**
−−	**Pengurangan**

A = A + 1 atau **A = A - 1;** disederhanakan menjadi :
A += 1 atau **A -= 1;** masih dapat disederhanakan menjadi **A ++** atau **A--**
Notasi " ++ " atau " -- " dapat diletakan didepan atau di belakang variabel

Contoh : A ++ *atau* ++A / A-- *atau* –A
Kedua bentuk penulisan notasi ini mempunyai arti yang berbeda.

- **Jika diletakan didepan variabel**, maka proses penambahan atau pengurangan akan dilakukan sesaat sebelum atau langsung pada saat menjumpai ekspresi ini, sehingga nilai variabel tadi akan langsung berubah begitu ekspresi ini ditemukan, *sedangkan*
- **Jika diletakan dibelakang variabel**, maka proses penambahan atau pengurangan akan dilakukan setelah ekspresi ini dijumpai atau nilai variabel akan tetap pada saat ekspresi ini ditemukan.

/* Penggunaan Notasi Didepan Variabel*/

```
// nama programmer :.....
#include <stdio.h>
#include <conio.h>
int main()
{
int a = 10, b = 5;

printf("Nilai   A    = %d", a);
printf("\nNilai ++A  = %d", ++a);
printf("\nNilai A    = %d", a);
printf("\nNilai B    = %d", b);
printf("\nNilai --B  = %d", --b);
printf("\nNilai B    = %d", b);
getch();
}
```

Output :

```
Nilai A         = 10
Nilai ++A       = 11
Nilai A         = 11
Nilai B         = 5
Nilai –B        = 4
Nilai B         = 4
```

/* Perbedaan operator peningkatan ++
yang diletakkan di depan dan dibelakang
operand */

```
//Nama programmer :.....
#include <stdio.h>
#include <conio.h>
int main()
{
```

```
int x, nilai;
x = 5;
nilai = ++x;                          /* berarti x = x + 1; nilai = x; */
printf("nilai = %d, x = %d\n", nilai, x);
nilai = x++;                          /* berarti nilai = x; nilai = x + 1; */
printf("nilai = %d, x = %d\n", nilai, x);
getch();
}
```

Outputnya : nilai = 6, x = 6
nilai = 6, x = 7

```
/*Contoh ke-2 operator peningkatan
  unary */
//Programmer :……
#include "stdio.h"
#include "conio.h"
int main()
{
int b, nilai;
b = 15;
nilai = --b;                  /* berarti b = b - 1; nilai = b; */
printf("nilai = %d, b = %d\n", nilai, b);
nilai = b--;                   /* berarti nilai = b; b = b + 1; */
printf("nilai = %d, b = %d\n", nilai, b);
getch();
}
```

Ouputnya : nilai = 14, b= 14
nilai = 14, b=13

5. Komentar Program

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan komentar atau penjelasan dalam bahasa C digunakan pembatas /* dan */ atau menggunakan tanda // untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan).

LATIHAN

1. Sebutkan tipe data yang kamu ketahui ?
2. Buatlah algoritma menggunakan tipe data char/string ?
3. Buatkan program C untuk mencari bilangan ganjil ?

BAB IV KEGIATAN BELAJAR

Kegiatan Belajar 4 : Input dan Ouput

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1. Memahami Perintah memasukkan data
2. Memahami Perintah menampilkan data

B. Uraian Materi

1. MEMASUKKAN DATA

Dalam bahasa C proses memasukkan suatu data bisa menggunakan beberapa fungsi pustaka yang telah tersedia. Beberapa fungsi pustaka yang bisa digunakan adalah :

a. scanf()

- Fungsi pustaka scanf() digunakan untuk menginput data berupa data numerik, karakter dan string secara terformat.
- Hal-hal yang perlu diperhatikan dalam pemakaian fungsi scanf() :
  - Fungsi scanf() memakai penentu format
  - Fungsi scanf() memberi pergantian baris secara otomatis
  - Fungsi scanf() tidak memerlukan penentu lebar field
  - Variabelnya harus menggunakan operator alamat &

Kode penentu format :

- %c : Membaca sebuah karakter
- %s : Membaca sebuah string
- %i, %d : Membaca sebuah bilangan bulat (integer)
- %f, %e : Membaca sebuah bilangan pecahan (real)
- %o : membaca sebuah bilangan octal
- %x : Membaca sebuah bilangan heksadesimal
- %u : Membaca sebuah bilangan tak bertanda

Contoh Program :

```
/* Program memasukan inputan dengan beberapa tipe data */
//Nama programmer :....
#include <stdio.h>
#include <conio.h>
int main()
{
int jumlah;
char huruf, nis[10];
float nilai;

printf("Masukkan sebuah bilangan bulat :");
scanf("%d", &jumlah );
printf("Masukkan sebuah karakter : ");
scanf("%c", &huruf );
printf("Masukkan nis Anda : ");
scanf("%s", &nis );
```

```
printf("Masukkan sebuah bil pecahan : ");
scanf("%f", &nilai );
printf("\nNilai variable yang Anda masukkan adalah :\n");
printf("jumlah = %d \n", jumlah );
printf("huruf = %c \n", huruf );
printf("nis = %s \n", nis );
printf("nilai = %f \n", nilai );
getch();
}
```

b. gets()

- Fungsi gets() digunakan untuk memasukkan data bertipe karakter dan tidak dapat digunakan untuk memasukkan data numerik.
- Harus diakhiri dengan penekanan tombol enter
- Cursor secara otomatis akan pindah baris
- Tidak memerlukan penentu format

Contoh Program :

```
/* Program inputan tipe data karakter string dengan fungsi gets*/
//Nama programmer :....
#include "stdio.h"
#include "conio.h"
int main()
{
char nama[40];
char alamat[40];
printf("Masukkan nama Anda : "); gets(nama);
printf("Masukan alamat anda :"); gets(alamat);
printf("Nama Anda adalah %s \n", nama);
printf("Alamat Anda adalah %s \n", alamat);
getch();
}
```

c. getchar()

- Fungsi getchar() digunakan untuk membaca data yang bertipe karakter
- Harus diakhiri dengan penekanan tombol enter
- Karakter yang dimasukkan terlihat pada layar
- Pergantian baris secara otomatis

d. getch() dan getche()

- Fungsi getch() dan getche() digunakan untuk membaca data karakter.
- Karakter yang dimasukkan tidak perlu diakhiri dengan penekanan tombol enter.
- Tidak memberikan efek pergantian baris secara otomatis
- Jika menggunakan fungsi getch() karakter yang dimasukkan tidak akan ditampilkan pada layer sehingga sering digunakan untuk meminta inputan berupa password.
- Sedangkan pada getche() karakter yang dimasukkan akan ditampilkan pada layar.

Contoh Program :

```
//contoh penggunaan getch dan getche
```

```
//Nama programmer:....
#include "stdio.h"
#include "conio.h"
int main()
{
char a1, a2;
printf("Masukkan sebuah karakter : ");
a1 = getche();
printf("\nKarakter yang Anda masukkan adalah %c\n", a1);
printf("\nMasukkan sebuah karakter lagi: ");
a2 = getch();
printf("\nKarakter yang Anda masukkan  adalah : %c", a2);
getch();
}
```

CATATAN :

Jika terdapat beberapa proses input (memasukkan data) sekaligus, maka sebaiknya ditambahkan fungsi **fflush(stdin);** setelah fungsi scanf(). Fungsi fflush(stdin) berfungsi menghapus buffer di dalam alat I/O.

2. MENAMPILKAN DATA

a. Menampilkan data ke layar monitor

- Menggunakan fungsi **printf(), puts(),** dan **putchar().**
- Fungsi **printf()** digunakan untuk menampilkan semua jenis data (numeric dan karakter)
- Fungsi **puts()** digunakan untuk menampilkan data string dan secara otomatis akan diakhiri dengan perpindahan baris.
- Fungsi **putchar()** digunakan untuk menampilkan sebuah karakter.

b. Mengatur tampilan bilangan pecahan (float).

Bentuk umum :

printf(“%m.nf”, argument);

- m : menyatakan panjang range
- n : menyatakan jumlah digit di belakang koma.
- argument : nilai atau variable yang akan ditampilkan.

Contoh :

printf(“%5.2f”, nilai);

artinya variable **nilai** akan ditampilkan sebanyak 5 digit dengan 2 digit di belakang koma.

Contoh Program 1 :

```
/* Program untuk menampilkan data berupa bilangan pecahan */
//Nama programmer:.....
#include "stdio.h"
#include "conio.h"
int main()
{
float nilai;
```

```
puts("Masukkan nilai Anda :" ); scanf("%f", &nilai);
printf("\n Anda memperoleh nilai %5.2f", nilai);
printf("\n Apakah Anda telah puas mendapat nilai %6.4f ", nilai);
getch();
}
```

Contoh Program 2 :

```
/* Program untuk menampilkan data berupa bilangan integer dan string */
//Nama programmer:.....
#include "stdio.h"
#include "conio.h"
int main()
{
int umur;
char nama[30];
puts("Masukkan nama Anda : ");
gets(nama);
puts("Masukkan umur Anda : ");
scanf("%d", &umur);
printf("Nama Anda : %s \n", nama);
printf("Umur Anda : %d \n", umur);
getch();
}
```

LATIHAN

- Dari contoh program diatas tambahkan untuk memasukkan dan mencetak data masukan berupa :
  1. Alamat
  2. No telepon
  3. Hoby
  4. Kelas
  5. Jurusan

c. Menampilkan data ke printer

- Untuk menampilkan data ke printer dapat menggunakan fungsi **fprintf(), fputs()** dan **fputc().**
- Fungsi **fprintf()** digunakan untuk mencetak semua jenis tipe data ke printer dan secara otomatis memberikan efek perpindahan baris.
- Fungsi **fputs()** digunakan untuk mencetak tipe data string ke printer
- Fungsi **fputc()** digunakan untuk mencetak tipe data karakter ke printer

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
int main()
{
fprintf(stdprn, “Hallo, Saya akan tercetak di printer”);
fputs(stdprn, “Saya juga akan tercetak di printer”);
}
```

BAB V KEGIATAN BELAJAR

Kegiatan Belajar 5 : Struktur Kontrol Percabangan

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1. Memahami Percabangan 1 kondisi
2. Memahami Percabangan 2 kondisi
3. Memahami Percabangan lebih dari 2 kondisi
4. Memahami Percabangan bersarang

B. Uraian Materi

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. Fungsi penyeleksian kondisi penting artinya dalam penyusunan bahasa C, terutama untuk program yang kompleks.

1. Percabangan 1 kondisi (Tunggal) atau Struktur Kondisi IF….

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan.

Bentuk umum:

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

Contoh Program 1 :

```
/* Program struktur kondisi if tunggal untuk memeriksa suatu kondisi  */
//Nama programmer :.....
#include "stdio.h"
#include "conio.h"
int main()
{
   float nilai;
   printf("Masukan nilai yang didapat :");
   scanf("%f", &nilai);
   if(nilai > 65)
        printf("\n ANDA LULUS !!!!\n");
   getch();
}
```

INFO

Bila program tersebut dijalankan dan kita memasukan nilai 80, maka perintah mencetak perkataan LULUS !!!! akan dilaksanakan, namun sebaliknya bila kita memasukan sebuah nilai yang kurang dari 65 maka program akan ***berhenti dan tidak dihasilkan apa-apa***.

Contoh Program 2 :

```
/* Program contoh penerapan struktur kondisi if tunggal*/
//Nama programmer :.....
#include "stdio.h"
#include "conio.h"
int main()
{
int a,b,c,max;
printf("Masukan bil 1 : ");fflush(stdin); scanf("%i",&a);
printf("Masukan bil 2 : ");fflush(stdin); scanf("%i",&b);
printf("Masukan bil 3 : ");fflush(stdin); scanf("%i",&c);
if((a>b)&&(a>c))
   max=a;
if((b>a)&&(b>c))
   max=b;
if((c>a)&&(c>b))
   max=c;
   printf("Bil terbesar : %i \n",max);
if(max>0)
   printf("Bil tsb adalah bil positif \n");
if(max<0)
   printf("Bil tsb adalah bil negatif \n");
getch();
}
```

1. Percabangan Ganda

Percabangan ganda apabila terdapat 2 alternatif instruksi yang dijalankan. Logika ini memungkinkan kompiler menjalankan salah satu dari 2 alternatif instruksi yang ada, dan salah satu instruksi pasti dijalankan.
Notasi algoritmik yang digunkanan :

```
If (kondisi) {
 Instruksi pertama
}else{
Instruksi kedua
}
```

Contoh :
Program memeriksa inputan apakah bilangan ganjil atau genap. Apabila diperiksa bilangan genap maka tulis “Bilangan genap” dan kalau bukan maka tulis “Bilangan ganjil”.

```
#include <stdio.h>
void main(){
int angka;
scanf(“%d”,&angka);
if (angka % 2 == 0)
printf (“Bilangan Genap”);
else
printf(“Bilangan Ganjil”);
}
```

2. Percabangan Lebih dari 2

Pada dasarnya hanya terdapat 2 jenis seleksi dalam struktur algoritma pemrograman, namun bukan berarti hanya bisa dikembangkan pada 2 jenis tersebut saja. Struktur seleksi dapat dikembangkan menjadi bentuk yang tidak terbatas dan dapat dikombinasikan kedalam bentuk perulangan selama notasi penulisannya tidak terdapat kesalahan.
Kemampuan logika seseorang dalam merancang program dan mengamati dari permasalahan yang ada menjadi bagian yang paling penting dalam melakukan pengembangan dari bentuk seleksi ini.
Dibawah ini akan diberikan contoh seleksi menggunakan kondisi lebih dari 2.

Contoh :
Program untuk menentukan grade dari sebuah nilai ujian, dengan aturan grade A untuk rentang nilai 80 – 100, grade B untuk nilai 70 – 80 dan grade C untuk nilai 50 – 70 dan grade D untuk nilai dibawah itu.

```
#include <stdio.h>
void main(){
int nilai;
scanf("%d",&nilai);
if (nilai >= 80 && nilai <= 100)
 print("Grade A");
   else if (nilai >= 70 && nilai <= 80)
     printf("Grade B");
     else if (nilai >= 50 && nilai <= 70)
      printf("Grade C");
  else
printf("Grade D");
}
```

3. Percabangan Bersarang Struktur Case

Struktur case sebenarnya memiliki fungsi yang sama dengan struktur if yang telah kita pelajari diatas. Struktur case ini dapat meringkaskan alur logika yang terjadi apabila diaplikasikan pada pada alur seleksi yang memiliki lebih dari 2 kondisi. Berikut adalah notasi algoritmanya :

```
switch (kondisi){
    case kondisi_1 :
    break;
    case kondisi2 :
    break;
default : }
```

Struktur logika seleksi menggunakan struktur case ini jauh lebih ringkas apabila diaplikasikan pada struktur seleksi yang memiliki kondisi lebih dari 2. Kompiler program akan menjalankan instruksi dari struktur case dan memeriksa setiap kondisi yang ada, apabila belum ada kondisi yang bernilai benar maka kompiler akan terus menjalankan instruksi dibawahnya sampai ditemukan kondisi yang bernilai benar. Namun apabila hingga kondisi terakhir diperiksa dan tidak ditemukan kondisi yang bernilai benar maka kondisi default yang akan dijalankan.

Contoh :
Program untuk menentukan apakah karakter '%', spasi, '&' atau '$' yang ditekan oleh pengguna melalui keyboard.

```
#include <stdio.h>

void main(){
int tombol;
scanf("%d",&tombol);
```

```
swicth(tombol){
case '32':
printf("Anda menekan tombol spasi");
break;
case '36: printf("Anda menekan tombol $");
break;
case '37' : printf("Anda menekan tombol %");
break;
case '38': printf("Anda menekan tombol &");
break;
default : printf("Anda tidak mematahui aturan.");
}}
```

Perbedaan yang paling jelas antara stukrut if dengan struktur case adalah :

- Struktur if dapat menerima kondisi yang berupa operasi logika. Sedangkan stuktur case tidak.
- Struktur case lebih efektif apabila digunakan untuk logika seleksi lebih dari 2 kondisi.
- Struktur case dan sktuktur if dapat dikombinasikan kedalam satu bagian, dengan catatan tata cara penulisan notasi tidak terdapat kesalahan.
- Struktur case tidak dapat melakukan pengecekan terhadap tipe data string / kalimat.

Bab 6

Algoritma Perulangan

Kompetensi Dasar :

- Memahami struktur algoritma serta menganalisa data dalam suatu algoritma perulangan
- Memecahkan permasalahan dengan algoritma perulangan

Pokok Bahasan :

- Perulangan dengan kondisi diawal
- Perulangan dengan kondisi diakhir
- Perulangan dengan kondisi akhir diinputkan user
- Perulangan sebagai pencacah naik
- Perulangan sebagai pencacah turun.

Tujuan Belajar :

Setelah mempelajari ini, siswa diharapkan mampu :

- Menyimpulkan penerapan algoritma perulangan untuk menyelesaikan masalah
- Menganalisa algoritma perulangan dengan Pelbagai macam data

6.1 Perulangan

Salah satu bagian yang paling membedakan antara manusia dengan komputer adalah : Komputer mampu mengerjakan instruksi dalam hitungan ribuan bahkan jutaan kali tanpa mengenal lelah. Dalam mempelajari algoritma pemrograman, struktur perulangan menjadi bagian yang sangat penting untuk dipelajari.

Struktur Perulangan

Struktur perulangan terdiri dari 2 bagian, yaitu :

1. Kondisi perulangan, yaitu ekspresi yang dilakukan sebelum pengulangan dilakukan pertama kali.
2. Body atau tubuh pengulangan, yaitu satu atau lebih instruksi yang diulang.

Selain itu biasanya di perulangan juga terdapat 2 hal dibawah ini, antara lain :

- Inisialisasi : aksi yang dilakukan sebelum pengulangan dilakukan pertama kali.
- Terminasi : aksi yang dilakukan untuk membuat perulangan berakhir. Biasanya berupa sebuah kondisi.

Dalam setiap bahasa pemrograman pada umumnya biasanya terdapat 3 jenis perulangan, antara lain :

1. Struktur WHILE – DO
2. Struktur Do – WHILE / REPEAT – UNTIL
3. Struktur FOR.

Ketiga jenis diatas hanyalah sebuah metode dan pada implementasinya, notasi penulisannya (sintaks) sangat tergantung dari setiap bahasa pemrograman yang digunakan.

6.1.1 Struktur WHILE – DO

Ciri khas dari struktur ini adalah :

- Dilakukan pengencekan di awal pada kondisi sebelum menjalankan instruksi di tubuh perulangan.
- Ada kemungkinan tubuh perulangan tidak dijalankan sama sekali.
- Setiap kali hendak melakukan perulangan berikutnya, selalu memeriksa kondisi perulangan. Apabila kondisi perulangan telah
- memberikan nilai false / salah. Maka perulangan akan dihentikan.

Notasi algoritmiknya adalah :

```
while (KONDISI){
tubuh perulangan yang berisi instruksi untuk dijalankan.
}
```

Contoh :

Progam membuat tulisan di angka 1 .. 100.

```
# include <stdio.h>

void main(){
int nilai_awal = 1; // inisialisasi awal. Sangat penting.
While (nilai_awal <= 100){
Printf("/n%d",nilai_awal);
Nilai_awal ++; // memanipulasi variabel awal agar tercapai kondisi terminasi.
}}
```

Dalam struktur perulangan ini, ada 2 hal yang harus diperhatikan untuk menghindari terjadinya kesalahan logika pada program.

- Inisialisasi variabel awal.

Ini dimaksudkan agar ketika kompiler program melakukan pemeriksaan terhadap kondisi awal, ditemukan kondisi yang benar. Pada beberapa bahasa pemrograman tertentu, apabila sebuah variabel tidak diinisialisasikan maka nilainya bisa berupa random ataupun nol. (lihat contoh dibawah)

- Manipulasi variabel awal.

Banyak terjadi kesalahan pada programmer ketika mereka membuat program perulangan, memanipulasi variabel kondisi sangat penting untuk menjaga program tetap sesuai dengan yang diinginkan. Ketika kita lupa memanipulasi variabel awal, ada kemungkinan program mengulang terus menerus (looping forever) karena kondisi yang diinginkan tercapat terus tanpa ada perubahan. (lihat contoh dbawah)

Contoh 1 :
Program menulis angka dari 1 .. 100.

```
# include <stdio.h>

void main(){
int nilai_awal ; // tidak dilakukan inisialisasi awal.
While (nilai_awal <= 100){
printf("/n%d",nilai_awal);
Nilai_awal ++; // memanipulasi variabel awal agar tercapai kondisi terminasi.
 }}
```

Program diatas tidak melakukan inisialisasi awal terhadap variabel nilai_awal, sehingga ada kemungkinan nilai_awal berisni nilai random. Misalkan nilai_awal berisi -1200, dan secara logika -1200 memang kurang dari 100. Maka program bukan mencetak 1 – 100, melainkan mencetak -1200 – 100.

Contoh 2 :
Program menulis angka dari 1 .. 100.

```
# include <stdio.h>

void main(){
int nilai_awal  = 1; //  inisialisasi awal dilakukan
While (nilai_awal <= 100){
Printf("/n%d",nilai_awal);
}}
```

Program diatas tidak melakukan manipulasi terhadap nilai_awal , sehingga nilainya selalu 1. Program bukannya mencetak 1 – 100, melainkan mencetak angka 1 terus menerus dan tidak pernah berhenti karena kondisi perulangan selalu benar.

6.2.2 Struktur Do – WHILE / REPEAT – UNTIL

Struktur Do = WHILE / REPEAT – UNTIL hampir mirip dengan struktur WHILE – DO. Berikut adalah ciri khas dari struktur perulangan ini.

- Tidak dilakukan pengecekan kondisi perulangan di awal eksekusi program.
- Minimal perulangan yang terjadi di tubuh program sebanyak 1 kali (Kerena tidak ada pengecekan kondisi perulanga di awal).
- Setiap kali hendak melakukan perulangan berikutnya, selalu memeriksa kondisi perulangan. Apabila kondisi perulangan telah memberikan nilai false / salah. Maka perulangan akan dihentikan.

Perbedaan paling mendasar sebenarnya terletak pada pengecekan kondisi perulangan, struktur ini melakukan pengecekan kondisi perulangan di akhir tubuh perulangan (bukan di awal seperti struktur WHILE – DO) sehingga mengakibatkan instruksi dijalankan minimal 1 kali.

Notasi algoritmiknya adalah :

```
do {      repeat
Tubuh perulangan          atau
}while (KONDISI); until KONDISI
```

Pada impelemntasinya notasi penulisan struktur perulangan ini juga bergantung pada bahasa pemgoraman yang digunakan. Pembahasan ini menggunakan bahasa pemrograman Turbo C dan leih ditekankan kepada konsep – konsep perulangannya.

Contoh :

Program menulis angka dari 1 .. 100.

```
# include <stdio.h>
void main(){
int nilai_awal  = 1; //  inisialisasi awal dilakukan
do{
  Printf("/n%d",nilai_awal);
   Nilai_awal += 1;// tambahkan nilai_awal sebanyak 1.
 }while(nilai_awal <= 100);
}
```

Program untuk meminta inputan dari keyboard, apabila pengguna menekan tombol esc maka program akan berhenti.

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


void main(){
char tombol;

do{
tombol = getch(); // peminta penekanan tombol dari keyboard.
// selama tombol yang ditekan tidak memiliki ASCII = 27 (tombol esc) maka
//program akan tetap berjalan.
}while (tombol != 27);
// program berhenti, karena pengguna telah menekan tombol escape.
}
```

Kapan menggunakan WHILE – DO atau Do – WHILE ?

Pemilihan antara kedua struktur ini sangat tergantung pada permasalahan yang dihadapi. Apabila sebuah program memerlukan instruksi dijalankan dahulu dan baru diperiksa kondisinya maka strukltur DO – WHILE harus digunakan namun apabila sebuha program harus memeriksa kondisi perulangan terlebih dahulu dan baru menjalankan tubuh perulangan, maka kondisi WHILE – DO harus digunakan.

6.2.3 Struktur FOR

Struktur perulangan for ini digunakan untuk perulangan yang tidak perlu memeriksa kondisi apapun dan hanya melaksanakan perulangan sejumlah kali tertentu.

Struktur perulangan ini paling cocok untuk proses perulangan yang telah diketahui batas akhirnya, karena kompiler akan mengeksekusi lebih cepat daripada 2 jenis struktur perulangan diatas.

Notasi algoritmiknya :

```
For (variabel awal = nilai awal; kondisi ;  faktor penaik){
Tubuh perulangan
}
```

Contoh :
Program menulis bilangan genap dari 2 – 100.

```
#include <stdio.h>

void main(){
int nilai_genap = 2; // inisialisasi awal.
for (nilai_genap =2; nilai_genap <= 100; nilai_genap+=2)
printf("%d",&nilai_genap);}
```

Program diatas akan melakukan inisialisasi nilai_genap sebanyak 2, dan setelah itu akan dilakukan pemeriksaan apakan kondisi terpenuhi / memberikan nilai benar. Apabila kondisi terpenuhi maka tubuh perulangna akan dijalankan (mencetak nilai dari nilai_genap) dan kemudian menaikkan nilai_genap sebanyak 2. Setlah itu akan dilakukan pemeriksaan kondisi sekali lagi, dan apabila kondisi tersebut terpenuhi maka tubuh perulangan akan dijalankan lagi sedangkan apabila kondisi perulangan tidak terpenuhi maka struktur perulangan akan berakhir.

Inti dari struktur perulangan ini adalah :

- Lebih cocok untuk jenis perulangan yang memiliki batas akhir yang sudah jelas.
- Pemeriksaan kondisi awal akan dilakukan di awal. Apabila kondisi terpenuhi, maka tubuh perulangan akan dilakukan. Apabila tidak, maka tubuh perulangan tidak akan pernah dilakukan.
- Ada kemungkinan tubuh perulangan tidak dijalankan sama sekali.
- Memiliki proses yang lebih cepat dibandingkan bentuk DO – WHILE atau WHLE – DO dalam proses perhitungan matematika.

6.3 Algoritma pengulangan (While, While bersarang, Repeat)

STATEMEN/PERYATAAN WHILE

Pernyataan while digunakan untuk perulangan yang banyaknya perulangan tidak diketahui. Peryataan while mirip dengan peryataan if yang melakukan pemeriksaan ekspresi boolean sebelum sebuah atau serangkaian peryataan dilakukan.

Bentuk umum:

while kondisi do
Statemen

Kondisi adalah ekspresi boolean. Jika ekspresi bernilai true statemen dijalankan dan iperiksa kembali, dan keluar dari perulangan jika bernilai false.

Contoh_While1:

```
Program deretangka_1;
uses crt;
var i:integer;
Begin
        clrscr;
        i:=1;
        while i <= 10 do
        begin
        writeln(i);
        i:=i+1;
        end;
        readln;
End.
```

Hasil :

12345678910

Contoh_While2:

```
Program deretangka_2;
uses crt;
var i:integer;
Begin
        clrscr;
        i:=10;
        while i > 0 do
        begin
        writeln(i);
        i:=i-1;
        end;
        readln;
```

Hasil :

10987654321

```
End.
```

Contoh_While3:

```
Program jumlahinteger;
uses crt;
var i,batas,hasil:integer;
Begin
        clrscr;
        write('Masukkan integer positif :');readln(batas);
        hasil:=0;
        i:=0;
        while i < batas do
        begin
                i:=1+1;
                hasil:=hasil+1;
        end;
        write('Jumlah 1 sampai ',batas,'=');
        write(hasil);
        readln;
End.
```

While Bersarang

Contoh_While4:

```
Program bintang2;
uses crt;
var baris, kolom, jumbaris:integer;
Begin
clrscr;
        write('Jumlah baris : );readln(jumbaris);
        baris:=1;
        while baris <= jumbaris do
        begin
                write('*' :jumbaris+1-baris);
                kolom:=2;
                while kolom <= (2*baris-1) do
        begin
                write('*');
                kolom:=kolom+1;
        end;
        writeln;
        baris:=baris+1;
        end;
readln;
End.
```

LATIHAN

Soal-soal peryataan WHILE:

1. Buatlah program untuk menampilkan bilangan 1 sampai dengan 5 ?

2. Buatlah program untuk menampilkan bilangan 5 sampai dengan 1 ?

Bab 7

Pengenalan pemrograman C++

Kompetensi Dasar :

Pokok Bahasan :
- Pengenalan C++

Tujuan Belajar :
Setelah mempelajari ini, siswa diharapkan mampu :
- Membuat

7.1 Bahasa C++

Berbicara tentang C++ biasanya tidak lepas dari C, sebagai bahasa pendahulunya. Pencipta C adalah Brian W. Kerninghan dan Dennis M. Ritchie pada sekitar tahun 1972, dan sekitar satu dekade setelahnya diciptakanlah C++, oleh Bjarne Stroustrup dari Laboratorium Bell, AT&T,
Pada tahun 1983. C++ cukup kompatibel dengan bahasa pendahulunya C. Pada mulanya C++ disebut " a better C ". Nama C++ sendiri diberikan oleh Rick Mascitti pada tahun 1983, yang berasal dari operator increment pada bahasa C. Keistimewaan yang sangat berari dari C++ ini adalah karena bahasa ini mendukung pemrograman yang berorientasi objek (OOP / Object Oriented Programming).

7.2 Format Penulisan pada C++

Bentuk umum :
```
# prepocessor directive
void main()
{
// Batang Tubuh Program Utama
}
```

Contoh :
```
# include <iostream.h>
Void main()
{
  cout << " Hai, Selamat menggunakan C++ ";
```

```
}
```