

MODUL LOGIKA DAN ALGORITMA

BAHASA C

“Bisa itu karena sudah terbiasa.” – “Repetition teaches even a donkey.”

Disusun oleh:

**Div. Education & Training PUB
(Periode 2019/2020)**

Nama :

Jurusan :

DIVISI PENDIDIKAN DAN PELATIHAN 2019/2020

A. PENGENALAN LOGIKA DAN ALGORITMA

Didalam kehidupan sehari-hari, kita pasti akan melakukan berbagai kegiatan ataupun menemui berbagai masalah. Nah, didalam melakukan kegiatan atau memecahkan masalah tersebut kita pasti memiliki langkah-langkah terurut dan terstruktur didalam prosesnya.

Oleh karena itu, tujuan diciptakannya komputer adalah untuk mempermudah pekerjaan manusia dalam memecahkan berbagai masalah yang mungkin akan sukar jika dilakukan oleh manusia itu sendiri, seperti perhitungan, mencari data, menghubungkan keluarga, memesan makanan, dll, dengan bantuan program-program yang ada didalam komputer itu sendiri.

Adapun penjelasan singkat mengenai logika dan algoritma:

1. Logika

Adalah suatu upaya berpikir secara cermat, tepat, dan logis. Maksudnya logis adalah dapat diterima oleh akal.

2. Algoritma

Adalah suatu urutan atau langkah yang tersusun secara terstruktur untuk menyelesaikan suatu masalah yang ditulis secara berurutan.

3. Logika dan Algoritma

Urutan/langkah-langkah yang terstruktur dan logis untuk memecahkan suatu masalah.



Logika



Algoritma

Gambar 1.1

Lalu, apa sajakah tujuan belajar Logika dan Algoritma? Secara garis besar, ada beberapa tujuan pembelajaran Logika dan Algoritma:

1. Memperkuat cara berpikir kita untuk menyelesaikan suatu masalah.
2. Mampu memecahkan masalah dengan menggunakan logika secara tepat dan efisien melalui langkah-langkah yang terstruktur.
3. Memperkuat analisis ketika pembuatan program.
4. Memperluas cara berpikir.
5. Merupakan **DASAR** dan **FONDASI** didalam belajar bahasa pemrograman. Jika kita sudah mendapat dasar dan fondasi ini, maka kita akan mudah mempelajari bahasa pemrograman manapun, karena logika bahasa pemrogramannya sama, tetapi yang berbeda hanyalah penulisan bahasanya saja.

Contoh Algoritma:

Algoritma membuat segelas kopi:

1. Mempersiapkan gelas, sendok, dan air panas.
2. Mempersiapkan kopi dan gula
3. Masukkan kopi, gula, lalu air panas
4. Aduk sampai rata
5. Kopi siap diminum

Apabila diubah urutannya menjadi:

1. Mempersiapkan gelas, sendok, dan air panas.
2. Mempersiapkan kopi dan gula
3. Aduk sampai rata
4. Kopi siap diminum
5. Masukkan kopi, gula, lalu air panas

Maka prosedur membuat kopi tersebut menjadi tidak logis, karena kita mengaduk gelas yang tidak berisi bahan-bahan membuat kopi alias gelas kosong.

Jadi, logika dan algoritma itu harus urut dan logis.

Coba perhatikan contoh kedua berikut ini:

Algoritma menghitung luas lingkaran:

1. Masukkan jari-jari (R)
2. Mengeset nilai $\Phi=3.14$
3. Hitung luas = $\Phi * R * R$
4. Mencetak luas

Jika diubah menjadi seperti ini:

1. Masukkan jari-jari (R)
2. Mengeset nilai $\Phi=3.14$
3. Mencetak luas
4. Hitung luas = $\Phi * R * R$

Jika kita mencetak luas terlebih dahulu lalu menghitung luas, maka apa yang akan tercetak? Tidak logis bukan? Ya, sekali lagi, logika algoritma adalah runtutan pemecahan suatu masalah yang logis.

Berikut beberapa kriteria algoritma yang baik

1. Memiliki Input
2. Memiliki Output minimal 1
3. Finite (terbatas)
4. Definite (pasti)
5. Efisien
6. Efektif
7. Terstruktur




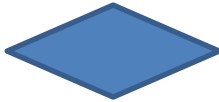




B. FLOWCHART/DIAGRAM ALUR

Algoritma dapat disajikan dalam dua bentuk, yaitu: tulisan dan bentuk gambar/symbol. Penyajian algoritma dalam bentuk tulisan biasanya menggunakan pseudocode dan penyajian dalam bentuk gambar biasanya menggunakan *flowchart*.

1. Pengertian Flowchart

Flowchart adalah bagan yang menggambarkan urutan instruksi/prosedur pemecahan masalah dengan menggunakan simbol-simbol standar yang mudah dimengerti dan mudah digunakan.

Adapun simbol-simbol standar dalam flowchart adalah sebagai berikut:

| | |
|---|---|
|  | Disebut dengan Terminator . Yaitu simbol yang digunakan untuk menyatakan awal dan akhir dari program. |
|  | Digunakan untuk menyatakan Input dan Output |
|  | Disebut dengan Process . Digunakan untuk menyatakan proses dalam program. |
|  | Disebut dengan Decision . Digunakan untuk menyatakan operasi perbandingan. |
|  | Arrow digunakan untuk menunjukkan arah proses. |
|  | Disebut dengan Connector on Page . Yaitu simbol yang digunakan untuk menggunakan flowchart di halaman yang sama. |
|  | Disebut dengan Connector off Page . Yaitu simbol yang digunakan untuk menggunakan flowchart di halaman yang berbeda. |
|  | Preparation digunakan untuk memberi nilai awal. Yaitu menggambarkan inisialisasi dalam alur logika. |

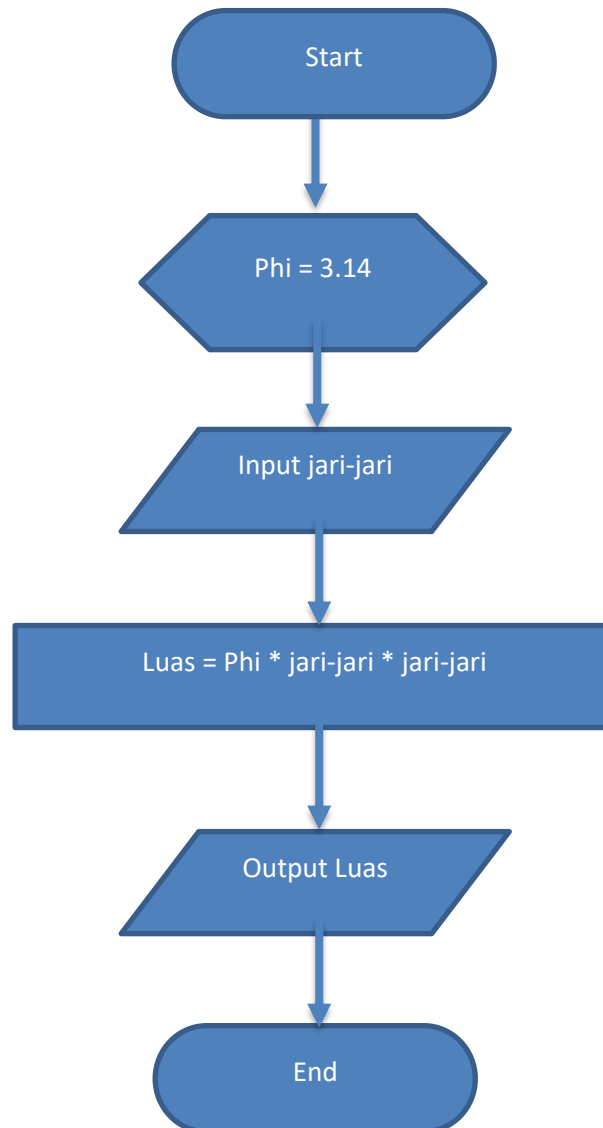
Gambar 1.2 | Tabel simbol-simbol flowchart

Contoh penggunaan flowchart:

Algoritma menghitung luas lingkaran:

- 1 Masukkan jari-jari (R)
- 2 Mengeset nilai $\Phi=3.14$
- 3 Mencetak luas
- 4 Hitung luas = $\Phi * R * R$

Penyajian dalam bentuk flowchart:



Gambar 1.3 | Contoh penyajian algoritma dengan flowchart

PSEUDOCODE

Selain dalam bentuk flowchart, algoritma juga bisa disajikan dalam bentuk tulisan atau biasa disebut dengan Pseudocode.

Pseudocode adalah cara penulisan algoritma yang hampir menyerupai Bahasa Pemrograman, namun Pseudocode ditulis lebih sederhana dengan menggunakan bahasa baku yang mudah dipahami oleh manusia. Tujuan menggunakan Pseudocode dalam mendeskripsikan suatu algoritma supaya programmer dapat memahami suatu kerangka awal (ide) suatu program dengan jelas, meskipun programmer tersebut belum bisa memahami bahasa pemrograman yang akan digunakan. Jadi Pseudocode digunakan untuk menggambarkan logika yang berupa urutan tahap pertama dari suatu ide program tanpa memandang Bahasa Pemrograman yang akan digunakan.

Berikut pseudocode menghitung luas persegi panjang:

| Urutan | Algoritma dengan kalimat deskriptif | Pseudocode |
|--------|--|-----------------------------------|
| 1. | Masukkan panjang | Input Panjang |
| 2. | Masukkan lebar | Input Lebar |
| 3. | Hitung luas dengan rumus Panjang * Lebar | Luas \leftarrow Panjang * Lebar |
| 4. | Tampilkan output | Print Output |

Contoh:

Menentukan nilai rata-rata dari mata pelajaran MTK, IPA, IPS:

Input Nilai MTK

Input Nilai IPA

Input Nilai IPS

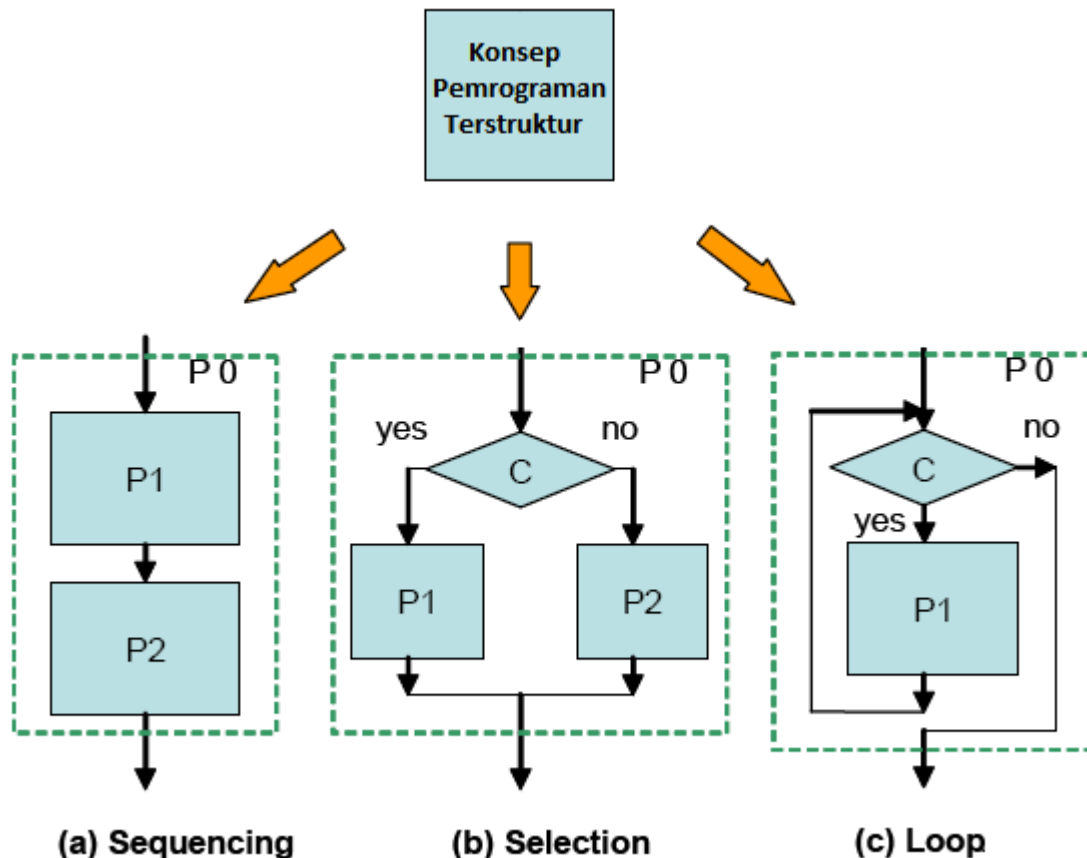
Rata-rata \leftarrow (Nilai MTK+ Nilai IPA+ Nilai IPS)/3

Print Rata-rata

Dari dua penyajian algoritma diatas, kita akan menerapkan dan mengimplementasikan kedua konsep dari 2 cara diatas dengan menggunakan salah satu bahasa pemrograman yang paling dasar yang harus mutlak dikuasai oleh setiap programmer yaitu menggunakan bahasa C. Apa itu bahasa C? Sebelum itu mari kita mengenal dulu apa itu bahasa pemrograman.

C. KONSEP-KONSEP PEMROGRAMAN

Didalam dunia pemrograman, terdapat 3 prinsip yang dikenal yang selalu digunakan oleh pemrogram didalam proses pembuatan suatu program aplikasi: ***runtutan, percabangan, dan perulangan***. Tiga hal mendasar inilah yang membentuk konsep segala jenis pemrograman. Jadi harus diingat 3 hal tadi jika berbicara tentang konsep pemrograman. Berikut gambarannya:



D. PEMROGRAMAN DAN BAHASA PEMROGRAMAN

Karena kita sedang belajar pemrograman dasar, maka kita harus tahu istilah-istilah yang dipakai didalam pembelajaran ini.

Secara garis besar, pemrograman berarti pembuatan progam, dalam hal ini yaitu program komputer yang dibuat dengan tujuan untuk mempermudah pekerjaan kita, seperti melakukan perhitungan, dll. Adapun pengertian yang lain untuk pemrograman: "*Pemrograman adalah proses menulis, menguji dan memperbaiki (debug), dan memelihara kode yang membangun suatu program komputer. Kode ini ditulis dalam berbagai bahasa pemrograman.*"

Adapun bahasa pemrograman yaitu bahasa yang dimengerti oleh komputer yang kita gunakan untuk dapat berinteraksi dan memberi perintah-perintah kepada komputer didalam program yang nanti kita definisikan sesuai kebutuhan. Berikut definisi yang lebih lengkap:

"Bahasa pemrograman, atau sering diistilahkan juga dengan bahasa komputer atau bahasa pemrograman komputer, adalah instruksi standar untuk memerintah komputer. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer. Bahasa ini memungkinkan seorang programmer dapat menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan/diteruskan, dan jenis langkah apa secara persis yang akan diambil dalam berbagai situasi."

Ada berbagai macam bahasa pemrograman. Berdasarkan tingkatan, ada 3 tingkatan bahasa pemrograman: Bahasa tingkat tinggi, bahasa tingkat menengah, dan bahasa tingkat rendah. Istilah untuk orang yang membuat program komputer ini disebut dengan *programmer*. Berikut beberapa contoh bahasa pemrograman:

- Python
- C
- C++
- C#
- Pascal
- PHP
- VB.NET
- Java
- COBOL
- Dan masih banyak lagi

Harus diperhatikan juga bahwa untuk memecahkan suatu masalah, ada 3 komponen utama yang komputer akan lakukan: INPUT -> PROSES -> OUTPUT

Input: Yaitu bagian masukan data yang nanti akan diolah menjadi informasi yang berarti

Proses: Yaitu bagian data akan diproses untuk menghasilkan output/hasil akhir. Disinilah rumus-rumus atau langkah-langkah didefinisikan untuk memecahkan masalah yang dimaksud.

Output: Yaitu hasil dari proses dan input data.

Contoh masalah:

Penjumlahan antara 5 dan 2

Penyelesaian:

Input: 5 dan 2

Proses: 5 ditambah 2

Output: 7

E. SEJARAH, PENGERTIAN DAN TUJUAN PEMBELAJARAN BAHASA C

Bahasa C adalah bahasa medium-level yaitu berada diantara bahasa pemrograman tingkat tinggi yang berorientasi pada manusia (misal: Java, Python) dan bahasa pemrograman tingkat rendah yang berorientasi pada mesin (misal: Assembler yang hanya dimengerti oleh mesin).

Pencipta bahasa C adalah Brian W. Kernighan dan Dennis M. Ritchi, sekitar tahun 1972. Penulisan program dalam bahasa C dilakukan dengan membagi dalam blok-blok, sehingga bahasa C merupakan bahasa yang terstruktur. Bahasa C dapat digunakan di berbagai mesin dengan mudah, mulai dari PC sampai dengan mainframe, dengan berbagai sistem operasi misalnya DOS, UNIX, VMS, dan lain-lain.

Tujuan pembelajaran bahasa C adalah untuk mengenalkan Logika dan Algoritma pemrograman melalui media bahasa C. Karena sebenarnya jika kamu sudah menguasai Logika dan Algoritma pemrograman, bahasa pemrograman manapun bisa dapat kamu kuasai karena logika pemrogramannya sama, yang membedakan adalah penulisan bahasanya saja yang berbeda (alih bahasa).

F. REVIEW BAB I

1. Apa pengertian Logika menurut kamu?
2. Apa pengertian Algoritma menurut kamu?
3. Apa pengertian Logika dan Algoritma menurut kamu?
4. Buat satu contoh Flowchart (bebas)!
5. Buat satu contoh Pseudocode (bebas)!
6. Apa pengertian pemrograman menurut kamu?
7. Apa pengertian bahasa pemrograman menurut kamu?
8. Kenapa pembelajaran Logika dan Algoritma itu penting?
9. Buat program untuk menghitung luas lingkaran!
10. Buat program untuk menghitung keliling segitiga!

BAB II SEQUENTIAL DAN DASAR-DASAR BAHASA C

A. Sequence/Runtutan

Sequence adalah konsep struktur algoritma paling dasar yang berisi rangkaian instruksi. Instruksi tersebut diproses secara runtut, satu persatu, mulai dari instruksi pertama sampai instruksi terakhir.

Setiap instruksi dipisahkan oleh semicolon (;). Algoritma merupakan runtutan (sequence) satu atau lebih instruksi. Ini artinya didalam algoritma :

- Tiap instruksi dikerjakan satu persatu.
- Tiap instruksi dilaksanakan tepat hanya satu kali, tidak ada instruksi yang di ulang.
- Urutan instruksi yang dilaksanakan program sama dengan urutan instruksi sebagaimana yang tertulis didalam teks algoritmanya.
- Akhir dari instruksi terakhir merupakan akhir algoritma.

Berikut adalah contoh sederhana Sekuensial dengan program penjumlahan sederhana bahasa C dengan penjelasannya:

```
1 #include <stdio.h>
  #include <conio.h>
2 main() {
  int a=5; 2
3 int b=3;
  printf("%d",a+b); 4
5 return 0;
}
```

Ini disebut dengan *library* yang biasa ada di program C. Nanti akan ada penjelasan lebih lanjut

Kurung buka kurawal adalah sintaks untuk menandakan awal dari suatu fungsi. Sedangkan kurung tutupnya menandakan akhir dari fungsi.

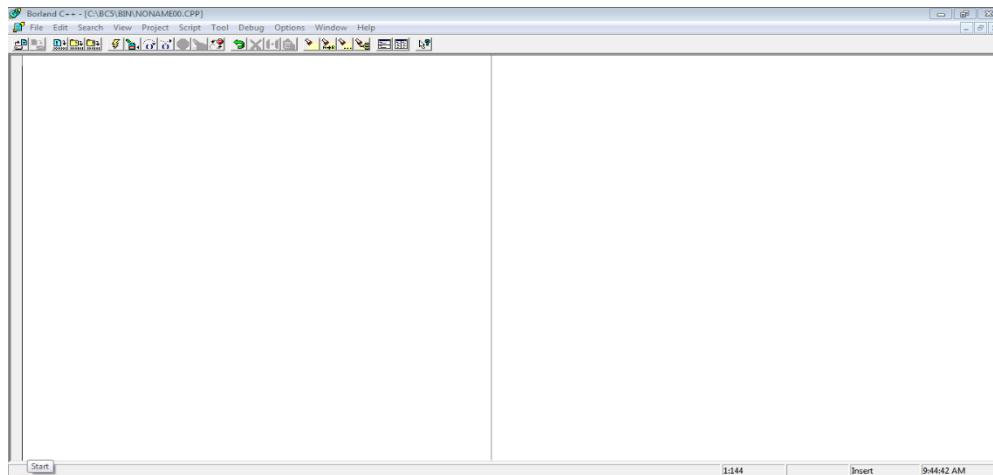
PENJELASAN:

1. Fungsi *main()* adalah fungsi yang pertama kali compiler jalankan.
2. Program akan mendeklarasi dan menginisialisasi variabel *a* yang bertipe integer.
3. Program akan mendeklarasi dan menginisialisasi variabel *b* yang bertipe integer.
4. Setelah program membuat variabel *a* dan *b*, kemudian program akan mencetak hasil penjumlahan kedua variabel tersebut.
5. Nilai 0 akan dikembalikan/*return* kepada komputer, karena *main()* adalah suatu fungsi. Penjelasan tentang Fungsi akan dijelaskan lebih lanjut di bab 5 tentang Prosedur dan Function.

B. Pengenalan Bahasa C

a. Cara Membuat dan Save Program

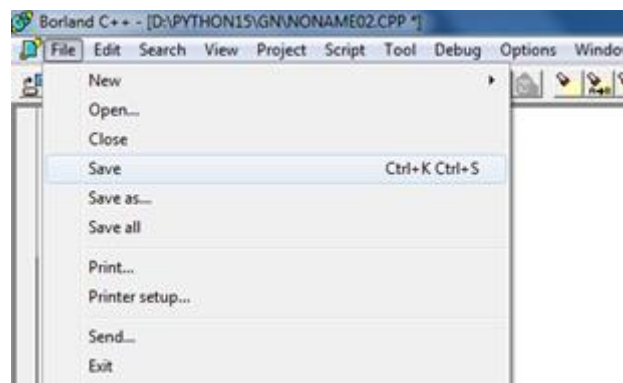
Sebelum kita memulai membuat program dengan bahasa C kita harus tau apa saja aplikasi bahasa C, yaitu Borland, Code Blocks, Dev C++. Namun disini kita akan mempelajarinya menggunakan Borland. Perhatikan gambar 2.1 Berikut :



Gambar 2.1 |Tampilan awal Borland

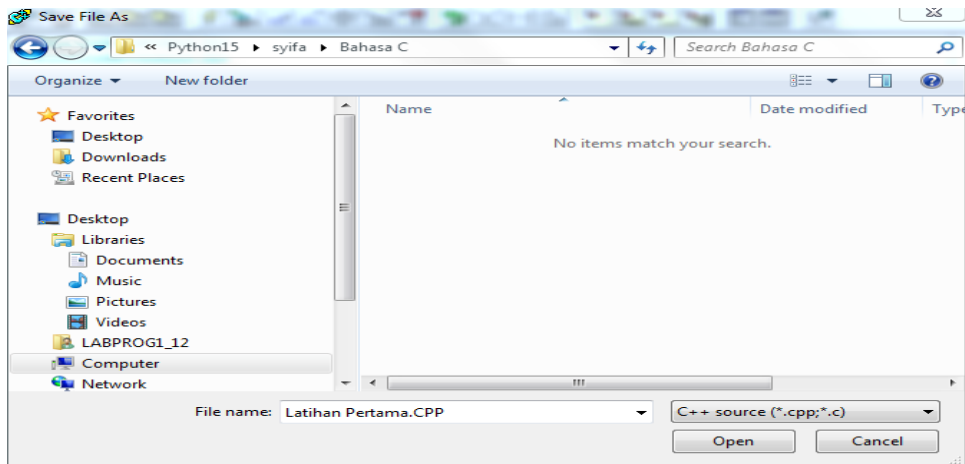
Untuk save program dalam Borland, dengan langkah sebagai berikut :

1. Klik file pada menu bar, lalu pilih save atau (Ctrl+K atau Ctrl+S)



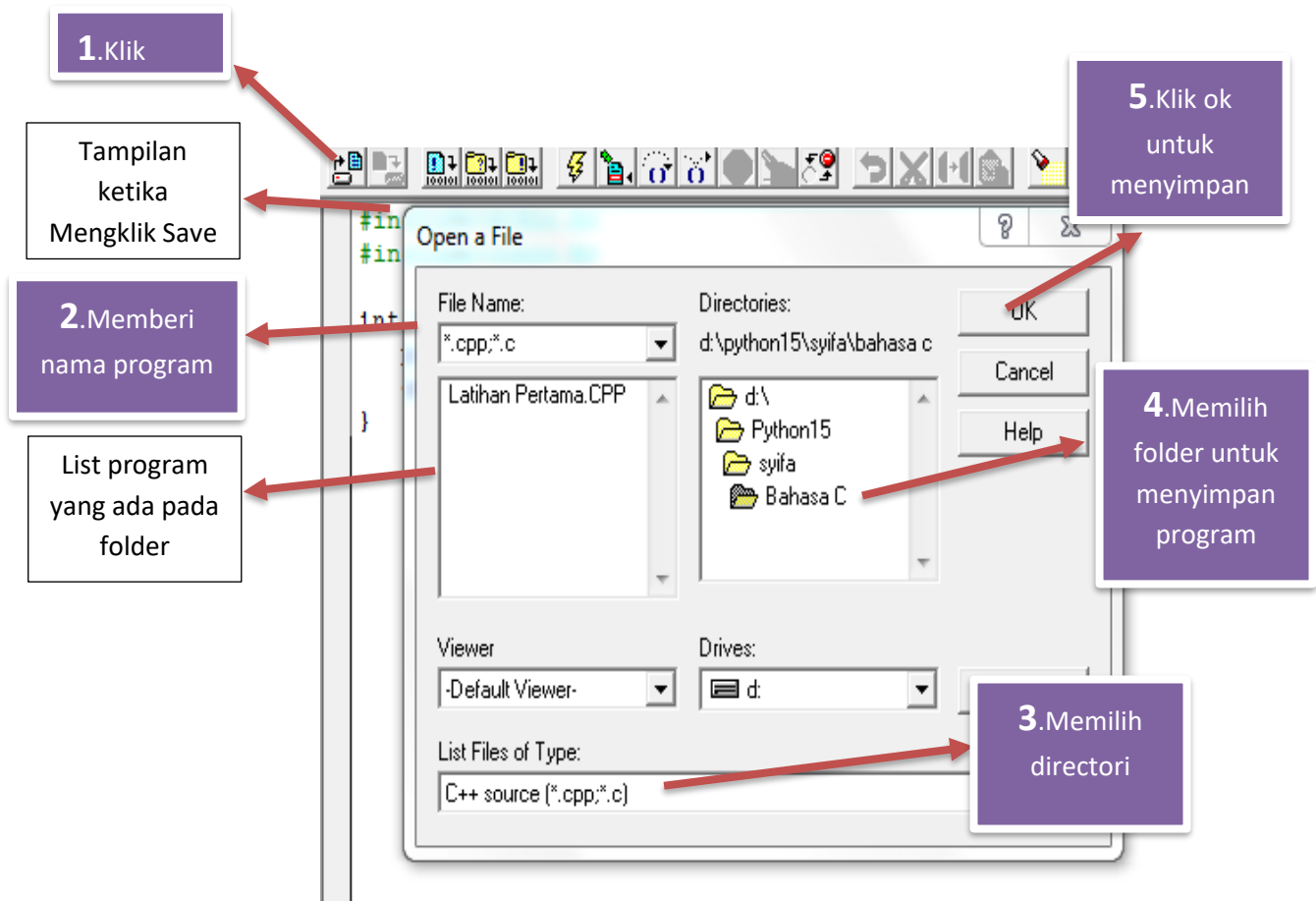
Gambar 2.2

2. Memilih direktori/folder untuk menyimpan program yang telah kita buat lalu klik save/open setelah memberi nama pada program yang kita buat.



Gambar 2.3

Selain seperti langkah diatas kita juga dapat menyimpan program dengan cara:



Gambar 2.5

b. Library Bahasa C

Library adalah kumpulan perintah yang akan memerintahkan compiler (program yang menerjemahkan bahasa pemrograman) untuk menyertakan file yang namanya ada diantara < > dalam proses kompilasi. Sebelum memasukan library terlebih dahulu kita mengetik #include. #include merupakan pengarah praprosesor (preprocessor directive) yang dipakai untuk membaca file yang berisi deklarasi fungsi dan definisi konstanta. Berikut adalah library yang sering digunakan dalam bahasa C :

1) Standard Input / Output <stdio.h>

| Fungsi | Keterangan | Cara penulisan |
|--------|---|-----------------------------|
| Printf | Mencetak (output program) | printf("Mencetak program"); |
| Scanf | Masukan tidak membaca spasi (input program) | scanf("%d",nama_variabel); |
| Gets | Masukan membaca spasi (input program) | gets(nama_variabel); |

2) Control Input / Output <conio.h>

| Fungsi | Keterangan | Cara Penulisan |
|--------|---|----------------|
| Getch | Inputan satu karakter/menahan output program | getch(); |
| Clrscr | Membersihkan layar output. | clrscr(); |
| Getche | Membaca karakter dengan sifat karakter yang tidak perlu diakhiri dengan ENTER | getche(); |

3) <string.h>

Berfungsi Untuk memanipulasi beberapa jenis String

| Fungsi | Keterangan | Cara penulisan |
|--------|---|--------------------------|
| Strcpy | Menyalin string | Strcpy(nama_variabel); |
| Strlen | Mengetahui panjang string | strlen(nama_variabel); |
| Strupr | Membuat string menjadi capital | strupr(nama_variabel); |
| Strlwr | Membuat string menjadi huruf kecil | strlwr(nama_variabel); |
| Strcmp | Membandingkan 2 buah string, apakah string pertama sama persis dengan string kedua jika sama akan mengembalikan nilai 0 dan garbage value(nilai sampah) jika kedua string berbeda | strcmp(string1,string2); |
| Strcat | Menggabungkan dua buah string | strcat(string1,string2); |

4) <math.h>

Berfungsi untuk menjalankan program yang berisi pustaka matematika seperti dibawah ini.

| Fungsi | Keterangan | Cara penulisan |
|-------------|--|--|
| Sqrt | Fungsi akar | <code>sqrt(nama_variabel);</code> |
| Sow | Fungsi pangkat | <code>pow(nama_variabel);</code> |
| sin,cos,tan | Masing-masing digunakan untuk menghitung nilai sinus, cosinus, tangen | <code>sin(nama_variabel);</code> <code>cos(nama_variabel);</code> <code>tan(nama_variabel);</code> |
| Max | Membandingkan dua buah bilangan dari dua bilangan untuk mencari nilai terbesar | <code>max(bil1,bil2);</code> |
| Min | Membandingkan dua buah bilangan dari dua bilangan untuk mencari nilai terkecil | <code>min(bil1,bil2);</code> |

5) Standard Library <stdlib.h>

| Fungsi | Keterangan | Cara Penulisan |
|--------|--|--|
| Atof | Mengkonversi nilai string menjadi bilangan bertipe double | <code>atof(nama_variable_string);</code> |
| Atoi | Mengkonfersi nilai string menjadi bilangan bertipe integer | <code>atoi(nama_variable_string);</code> |
| Pow | Digunakan untuk pemangkatan | <code>pow(bilangan,pangkat);</code> |

6) <windows.h>

Pada library ini biasanya digunakan untuk membuat design pada program seperti membuat warna menggunakan 'system', atau untuk meletakkan kursor (benda yang berkedip ketika kita mengetik) dengan koordinat x dan y menggunakan 'gotoxy'.

c. Kerangka Bahasa C

Setelah mengetahui library yang akan digunakan dalam bahasa C langsung saja kita mulai membuat program sederhana bahasa C. Perhatikan gambar 1.2.

```

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

int main(){
    printf("Hello World!");
    getch();
}

```

Gambar 2.6

Penjelasan :

- #include <stdio.h> dan #include <conio.h>: Bukan merupakan pernyataan sehingga tidak diakhiri semicolon(;). Baris ini akan memerintahkan compiler(program yang akan menerjemahkan bahasa pemrograman).

(?) Mengapa <stdio.h>? karena didalam fungsi main kita akan menggunakan fungsi printf.

(?) Mengapa <conio.h>?karena didalam fungsi main terdapat getch(); getch() merupakan anggota dari <conio.h>.

(?) bagaimana jika kita tidak menuliskan library? Tentu program akan error karena tidak ada perintah kepada compiler untuk menerjemahkannya.

(*)Jadi cantumkanlah library seperlunya saja , fungsi apa saja yang akan digunakan pada main. Dan **INGATLAH** fungsi tersebut merupakan anggota dari library mana.

- int main(){ } : fungsi utama yang akan dijalankan pertama kali saat program dieksekusi atau dijalankan. Program yang akan dijalankan yaitu yang ada didalam kurung kurawal {statement}.
- printf("Hello World"); : Bagian ini dinamakan statement yang akan dieksekusi oleh compiler.

DIINGAT !

Program yang merupakan pernyataan/statement, harus di akhiri dengan semicolon (;)

C. Tipe Data

1. Numerik dan Karakter

Tipe data adalah suatu pengenalan yang merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap intruksi yang akan dilaksanakan oleh komputer. Misalnya 5 dibagi 2 saja bisa menghasilkan hasil yang berbeda tergantung type datanya. Kita juga harus mengetahui format specifier, apa itu format specifier? Format specifier adalah sebuah kode yang digunakan untuk membaca variabel ketika akan dicetak. Format specifier biasanya diawali dengan menuliskan karakter %(persen) untuk kemudian diikuti huruf sesuai tipe datanya. Tipe data dasar dalam bahasa C :

DIINGAT !!!

| Type (penulisan) | Keterangan | Format specifier |
|---------------------------|--|------------------|
| Integer (int) | Bilangan bulat yang hanya memuat 8 angka | %d atau %i |
| Long (long) | Bilangan bulat yang memuat lebih dari 8 angka | %ld atau %li |
| Float (float) | Bilangan desimal yang memuat 8 angka di belakang koma | %f |
| Double (double) | Bilangan desimal yang memuat lebih dari 8 angka di belakang koma | %lf |
| Character(char) | Terdiri hanya 1 karakter | %c |
| String(char* atau char[]) | Kumpulan dari banyak karakter yang disebut string | %s |

2. Boolean

Boolean merupakan tipe data dasar dalam bahasa C hanya saja tidak ada penulisan atau pendeklaran secara jelas, hanya kondisi **True** atau **False** yang biasa digunakan dalam operator logika. Boolean juga bisa dikondisikan sebagai angka misalnya 1 sebagai kondisi True dan 0 sebagai kondisi False.

D. Variabel Dan Konstanta

a. Variabel

Variabel adalah wadah yang digunakan untuk menampung data/value. Bayangkan saja gelas dan susu. Analoginya gelas adalah tempat untuk menampung susu tersebut. Jadi, dalam membuat program, tentu kita sangat membutuhkan variabel. Untuk mengenali variabel tersebut maka harus ada tipe data. Jika kita memilih gelas maka gunakan gelas itu untuk menampung air, tidak mungkin kan gelas untuk menampung nasi. Oleh karena itu

dalam implementasi kedalam bahasa pemrograman juga harus menggunakan tipe data sebagaimana harusnya.

Berikut adalah aturan dalam penulisan variabel : **(DIINGAT!!!)**

- a) Boleh terdiri dari gabungan huruf dan angka dengan **karakter pertama harus huruf**.
- b) Bahasa C bersifat **case sensitive**, artinya huruf besar dan kecil dianggap berbeda. Jadi, antara NAMA, nama dan Nama itu dianggap berbeda.
- c) Tidak boleh mengandung spasi. Tidak Boleh mengandung simbol-simbol khusus, kecuali garis bawah (underscore).
- d) Yang termasuk simbol khusus yang tidak diperbolehkan antara lain : !, @, #, \$, %, ^, &, *, (,), -, +, =.

contoh penamaan variabel :

| Benar | Salah |
|----------------|-----------------|
| Nama_Mahasiswa | %nama-mahasiswa |
| Kelas2 | 2Kelas |
| _HaSiLaKhIr_ | Hasil Akhir |
| TotalBelanja | Total-belanja |

b. Deklarasi dan Inisialisasi

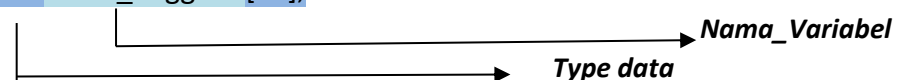
- *Deklarasi/Pendeklaran*

Deklarasi /Pendeklaran adalah penamaan atau pengenalan suatu variable Dalam suatu program. Dengan Struktur :

Tipe_Data nama_variabel;

Misalnya kita ingin membuat sebuah variable dengan nama *nama_anggota* dengan type data Char.

Char nama_anggota [25];



- *Inisialisasi*

Inisialisasi atau penginisialisasian adalah suatu pengisian variable dengan suatu nilai/value dalam program. Dengan Struktur :

Tipe_Data nama_variabel=**Value/nilai**;

Misalnya kita ingin mengisi variable *nama_anggota* dengan Elisa yang tipe data String.



Char nama_karyawan [25] = 'Elisa';

Contoh pendeklaran & inisial bertipe data integer:

int x; → **Pendeklaran Variabel**

x=8; → **Penginisialisasian Variabel**

Atau

int x=8;

c. Jenis Variabel

Jenis variable dibedakan berdasarkan letak pendeklaran variable, perhatikan code dibawah ini :

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

int angka1;

int main(){
    int angka2;
    angka1=8;
    angka2=9;
    printf("Angka1 : %d\nAngka2 : %d", angka1, angka2);
    getch();
}
```

Gambar 2.7

Jadi apa perbedaannya?

a) Variabel lokal

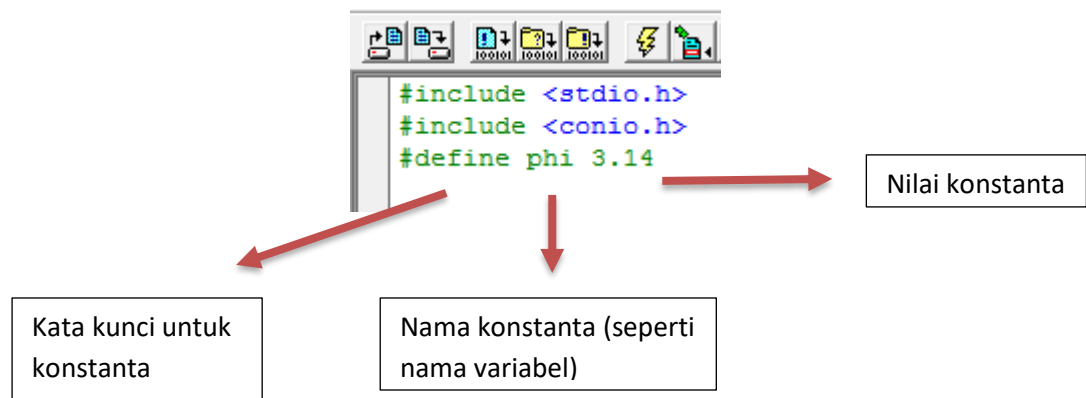
Variabel lokal adalah variabel yang deklarasinya berada **didalam fungsi**, misalnya **fungsi main**, sehingga variabel tersebut hanya bisa diakses oleh anggota main itu sendiri.

b) Variabel global

Variabel global adalah variabel yang deklarasinya berada diluar fungsi main atau fungsi manapun sehingga variabel itu bisa di akses oleh fungsi manapun dan dikenal oleh siapapun.

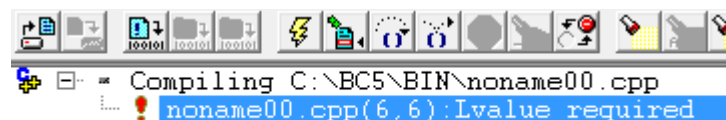
d. Konstanta

Konstanta adalah variabel yang isinya tetap/tidak dapat berubah-ubah. Konstanta tidak dapat dikenai assignment (datanya diisi lagi/diubah) dan saat pembuatan variabel konstanta harus langsung di inisialisasi/diisi data sesuai dengan tipe data. Perhatikan kodingan berikut :



Gambar 2.8

Penulisan konstanta berada bersama library dengan kata kunci `#define` maka nilai `phi` tersebut tidak bisa lagi diubah. Nah jika didalam main kita ubah nilai `phi` tersebut maka akan muncul eror seperti dibawah ini :



Gambar 2.9

E. Operator Dalam Bahasa C

a. Pengertian Operator

Operator secara umum adalah sesuatu yang melakukan operasi. Operator dalam bahasa C dikenal sebagai suatu karakter atau simbol khusus yang berguna untuk melakukan operasi seperti penjumlahan, perkalian, pembagian, pengurangan, perbandingan, dan penginisialisasian (pengisian data).

Operator yang paling kita kenal dalah operator perkalian, pembagian, penjumlahan, pengurangan dan modulus (sisa hasil bagi) yang dalam bahasa C dikenal sebagai operator **Aritmatika**. Namun dalam bahasa C masih banyak jenis operator yang tersedia dan harus kita pahami. Adapun jenis-jenis operator bahasa C adalah sebagai berikut :

1) Operator Penugasan (Assignment Operators)

Operator penugasan adalah operator yang digunakan untuk memberikan suatu nilai kedalam suatu variabel. Operator penugasan dalam bahasa C ditulis dengan symbol atau karakter '='. Perhatikan contoh dibawah ini !

```
int main(){
    int angka;
    angka=8;
    printf("Angka : %d",angka);
    getch();
}
```

Area penugasan !!!
Operator '=' memasukan angka
8 kedalam variabel angka

Gambar 2.10

Penjelasan :

- `int angka;` artinya kita membuat variabel bertipe data integer dan variabel bernama angka
- `Angka=8;` artinya kita memasukan angka 8 kedalam variabel angka yang bertipe data integer.

Operator penugasan yang lain:

- `+=` (Tambah Samadengan)
- `-=` (Kurang Samadengan)
- `/=` (Bagi Samadengan)
- `*=` (Kali Samadengan)
- `%=` (Modulus Samadengan)
- `++` (Plus plus. Ini sama seperti ditambah 1)
- `--` (Min min. Ini sama seperti dikurang 1)

Contoh:

```
int a=5;
a+=3;
a--;
printf("%d",a);
```

Maka yang akan dicetak adalah 7 karena setelah a ditambah nilai 3 lalu dikurangi 1.

2) Operator aritmatika (Arithmetic Operators)

| Simbol Operator | Nama Operator dan kegunaannya | Contoh penulisan |
|-----------------|-------------------------------|-------------------|
| * | Perkalian (Multiply) | <code>x*z;</code> |
| / | Pembagian(Divide) | <code>x/z;</code> |
| + | Penjumlahan(Add) | <code>x+z;</code> |
| - | Pengurangan(subtract) | <code>x-z;</code> |
| % | Modulus(sisa hasil bagi) | <code>X%z;</code> |

Untuk lebih jelas coba yuk program dibawah ini :

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

int main(){
    int x=8;
    int y=4;
    int Z;
    Z=x*y;
    printf("x * y= %d\n",Z);
    Z=x/y;
    printf("x / y= %d\n",Z);
    Z=x+y;
    printf("x + y= %d\n",Z);
    Z=x-y;
    printf("x - y= %d\n",Z);
    getch();
}
```

Gambar 2.11

3) Operator Perbandingan (Comparison Operators)

Operator perbandingan dalam bahasa C operator yang berfungsi untuk membandingkan suatu variabel dengan variabel lainnya atau suatu kondisi dengan kondisi lainnya. Operator perbandingan biasanya digunakan dalam kondisi pilihan (if else). Dalam bahasa C operator perbandingan ditulis dengan simbol-simbol sebagai berikut :

| Simbol Operator | Deskripsi |
|-----------------|-------------------------|
| == | Sama Dengan |
| != | Tidak Sama dengan |
| <> | Tidak Sama dengan |
| > | Lebih dari |
| < | Kurang dari |
| >= | Lebih dari sama dengan |
| <= | Kurang dari sama dengan |

Catatan:

Bedakan antara operator penugasan (=) dengan operator perbandingan (==)

4) Operator Logika (Logical Operators)

Berikut Adalah cara penulisan dan ketentuan operator dalam bahasa C.

| Simbol Operator | Nama Operator | Keterangan |
|-----------------|---------------|--|
| && | And(dan) | Membandingkan 2 atau lebih kondisi dimana kondisi akan True (benar) jika semua syarat terpenuhi |

| | | |
|---|------------------|--|
| | Or(atau) | Membandingkan 2 atau lebih kondisi dimana kondisi akan True jika salah satu syarat terpenuhi /benar maka akan menghasilkan kondisi True |
| ! | Not (Not/Negasi) | Menandakan negasi /awan dari suatu kondisi/ Pernyataan |

Berikut table perbandingan yang hanya akan membandingkan 2 kondisi

DIINGAT!!!:

1. Operator logika && (And)

| Kondisi 1 | Kondisi 2 | Hasil |
|-----------|-----------|-------|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

2. Operator logika || (Or)

| Kondisi 1 | Kondisi 2 | Hasil |
|-----------|-----------|-------|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

3. Operator logika !(Not)

| Kondisi | Hasil |
|---------|-------|
| True | False |
| False | True |

F. Sintaks Dasar Bahasa C

a. Komentar dalam Bahasa C

Komentar dalam bahasa C merupakan syntax agar tulisan yang berada dalam komentar tidak dieksekusi oleh compiler sehingga tidak menyebabkan error. Ada 2 macam komentar dalam bahasa C :

1. Komentar satu baris (single comment line), adalah komentar yang hanya berisi satu baris saja. Kodenya adalah `//` double slash pada awal kodingan yang akan dikomentari.

Contoh :

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

int main(){
    //ini program mencetak Hello World!
    printf("Hello World !");
    getch();
}
```

Sepanjang apapun tulisannya jika masih 1 baris tidak akan dieksekusi oleh compiler

Gambar 2.12

2. Komentar multiple lines (komentar dengan baris lebih dari satu), adalah komentar yang berisi lebih dari satu baris. Kodenya adalah `/*` slash bintang dan diakhir baris komentar ditutup lagi dengan `*/`. Contoh :

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

int main(){
    /*ini program mencetak Hello World!
    printf("Hello World !");
    getch();*/

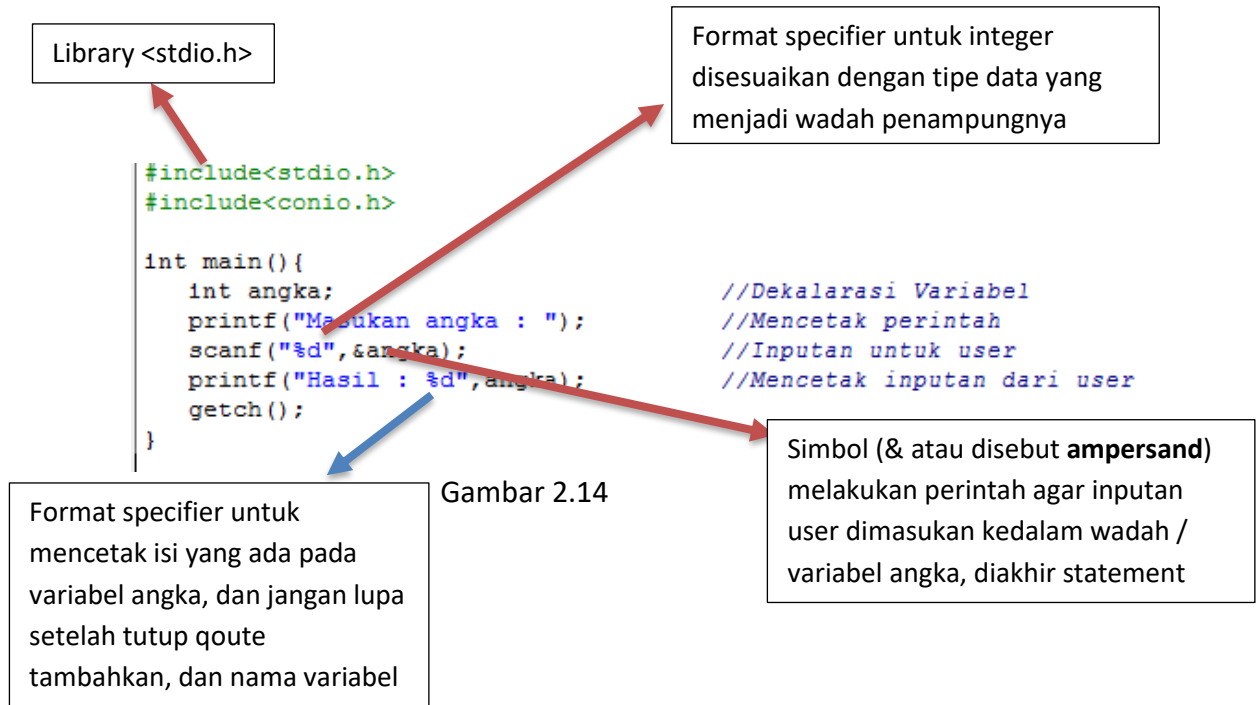
    printf("Halo Dunia");
    getch();
}
```

Tulisan yang diapit oleh `/*` dan `*/` tidak akan dieksekusi, oleh karena itu jangan lupa untuk menutupnya agar tidak menimbulkan error ya

Gambar 2.13

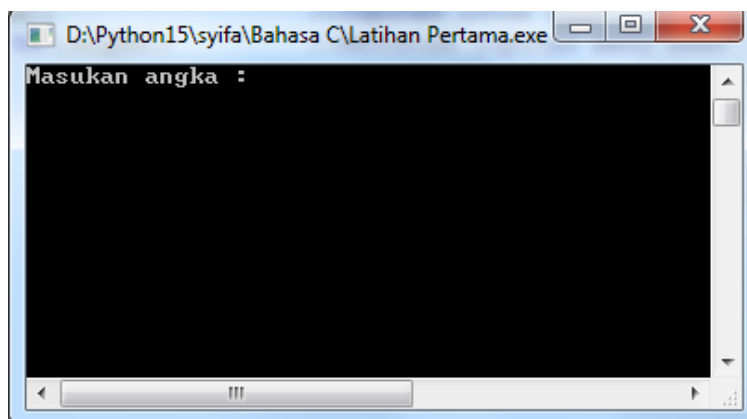
b. Perintah Input dan Output program

Input dan output adalah hal yang sangat dasar dalam program, dalam input dan output harus menggunakan format specifier. Perhatikan dalam kode berikut!



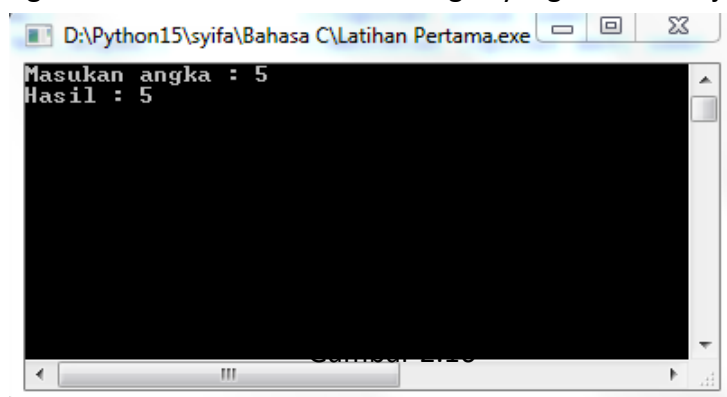
Hasil outputnya akan seperti ini :

- Program akan meminta inputan dari kita



Gambar 2.15

- Program akan mencetak kembali angka yang sudah menjadi inputan



Dan coba juga inputkan 1 2 (1 'spasi' 2) apakah hasilnya ??

Ya Betul, yang tercetak hanyalah angka 1.

(?) Kenapa yang tercetak hanya angka 1? Karena fungsi scanf tidak bisa membaca spasi jadi sepanjang apapun inputan yang dipisahkan oleh spasi tidak akan terbaca.

(?) Jika ingin menerima inputan yang dapat membaca spasi menggunakan apa ?
Menggunakan gets, oke langsung saja lihat syntax dibawah ini

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

int main(){
    char nama[15];
    printf("Masukan Nama :");gets(nama);
    printf("%s",nama);
    getch();
}
```

Gambar 2.17

Cobalah kode diatas apakah berhasil atau tidak ?

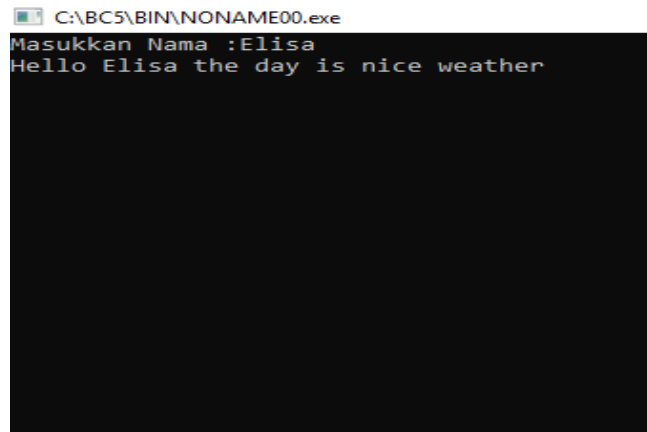
c. Escape Sequence

Karakter-karakter escape-sequence adalah suatu urutan karakter yang digunakan untuk mewakili ekspresi suatu karakter lain. Kumpulan karakter ini selalu dimulai oleh karakter backslash(\) atau kebalikan dari slash dan diikuti oleh suatu huruf atau suatu kombinasi angka. Berikut daftar karakter-karakter escape sequence umum:

| Simbol | Keterangan |
|--------|--|
| \n | New-line(enter) |
| \t | Horizontal-tab(tab) |
| \v | Vertical-tab |
| \b | Backspace |
| \r | Carriage-return |
| \f | Form-feed |
| \a | Bell |
| \' | Tanda petik |
| \" | Tanda kutip |
| \\ | Backslash |
| \ddd | Karakter ASCII dalam notasi octal |
| \xdd | Karakter ASCII dalam notasi heksadesimal |

G. Review

- Buatlah sebuah program yang mencetak Hello World!
- Buatlah sebuah program dengan output sebagai berikut :



Gambar 2.18

Note : nama yang tercantum harus sesuai dengan inputan !

- Berapakah hasil a? dan kenapa ?

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

int main(){
    int a=7;
    int b=8;
    int c=a;
    a=b-c;
    b=a;
    c=a+b-a;
    printf("a adalah %d ",a);
    getch();
}
```

BAB III DECISION ATAU PERCABANGAN

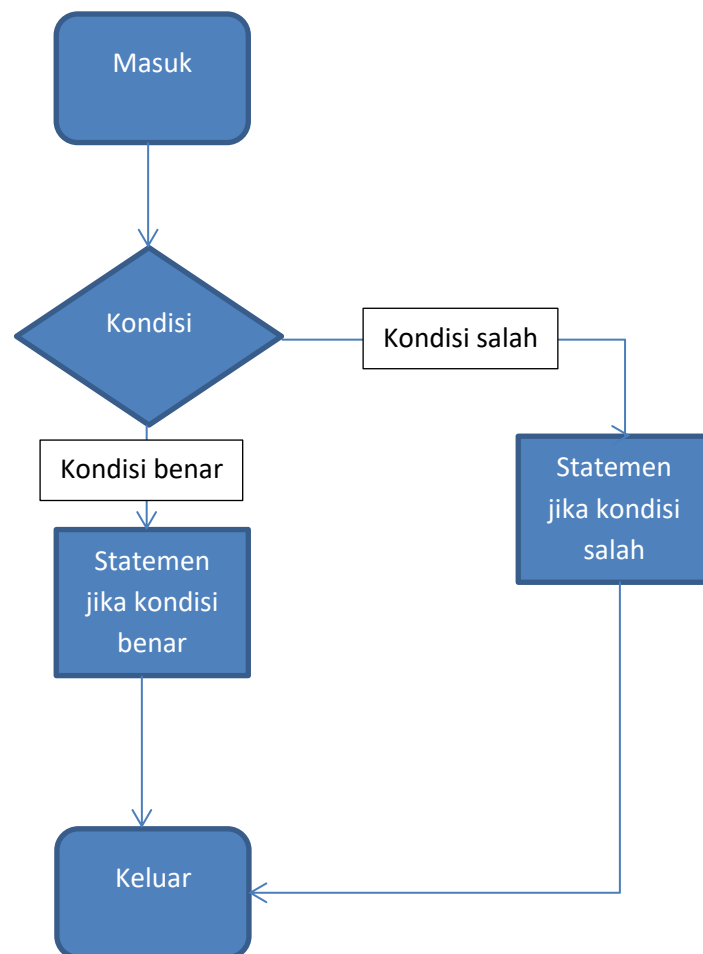
A. Pengertian Decision/Percabangan

Decision atau percabangan adalah suatu perintah dalam bahasa C untuk **memilih berdasarkan Kondisi yang dibuat**. Suatu perintah akan dijalankan oleh program **jika telah memenuhi kondisi**.

Berikut adalah contoh decision dalam kehidupan sehari-hari:

- Jika awan mendung, maka akan turun hujan.
Pada statemen diatas maka kita tahu bahwa hujan akan turun jika kondisi awannya mendung. Dan apabila awan tidak mendung maka hujan tidak adak turun.
- Jika Joni sehat, maka joni tidak akan meminum obat
Pada statemen diatas maka Joni akan meminum obat jika joni tidak sehat. Dan kita tahu dari statement di atas bahwa Joni sedang tidak sehat.

Contoh decision dalam flowchart:



B. Tujuan Decision

Adapun tujuan dari decision itu sendiri adalah untuk melakukan **penyaringan atau pemilihan** dalam suatu program yang akan kita buat.

C. Jenis Decision

Dalam bahasa C kita Decision atau percabangan ada 3 cara yaitu menggunakan `if`, `Switch...case`, dan tanda tanya(?) atau ternary operator.

Berikut adalah cara-cara penggunaan Decision :

a. Decision menggunakan If

1. If...else

Bentuk umum decision atau percabangan *if* kerangka dalam bahasa C umumnya sebagai berikut:

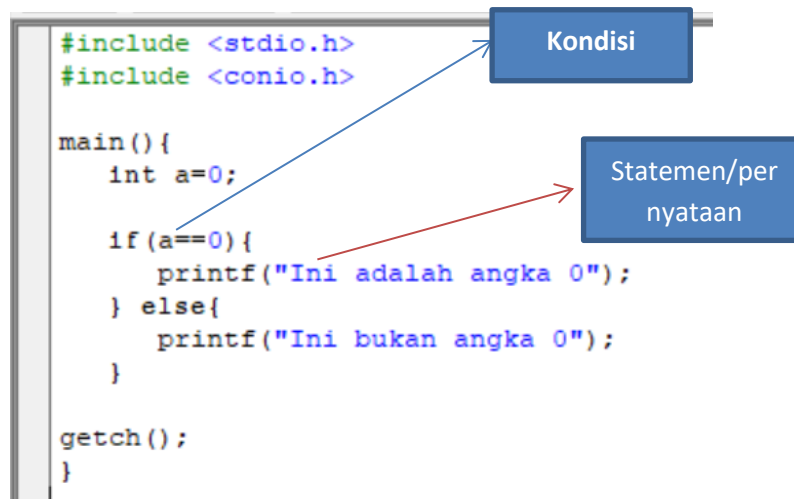
`if(kondisi)`

Statemen jika kondisi benar;

`else`

Statemen jika kondisi salah;

Contoh program:



Gambar 3.1

Gambar 3.1 adalah contoh program untuk mengecek variabel `a` yang bertipe integer bernilai 0 atau bukan jika variabel `a` bernilai 0 maka program akan mencetak "Ini adalah angka 0" jika salah maka program akan mencetak "Ini bukan angka 0".

II. If..else if ..else

Nahh, sekarang bagaimana bila kita ingin lebih dari 2 kondisi ? Caranya adalah kita menggunakan fungsi yang namanya *else if* . Berikut adalah contoh penggunaan *else if* :

```
if(kondisi)

    Statemen jika kondisi benar;

else if(kondisi)

    Statemen jika kondisi benar;

else

    Statemen jika kondisi salah;
```

Contoh program decision menggunakan else if:

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

main() {
    char a[12]="persegi";
    if (a=="persegi")
        printf("Ini adalah persegi");/*jika kondisi if benar*/
    else if (a=="segitia")
        printf("Ini adalah segitiga");/*jika kondisi else if benar*/
    else
        printf("Ini bukan persegi ataupun segitiga");/*jika kondisi if atau else if salah*/
    getch();
}
```

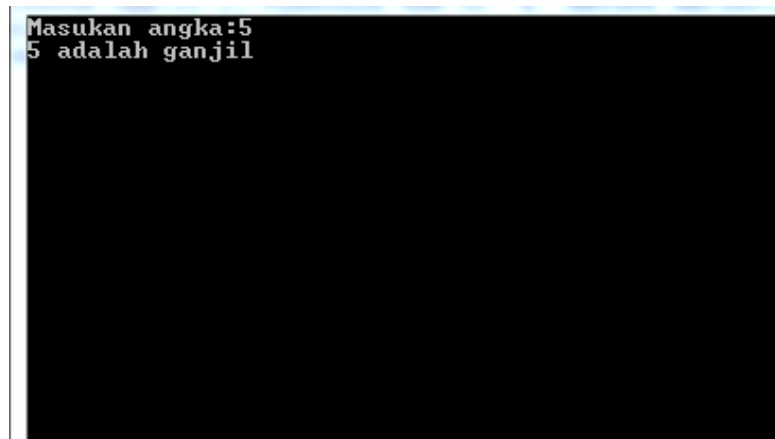
Gambar 3.2

Gambar 3.2 adalah contoh program untuk mengecek variabel a apakah berisikan kata “persegi” atau “segitiga” jika variabel a berisi persegi maka program akan mencetak “Ini adalah persegi” bila variabel a bukan berisi “persegi” melainkan “segitiga” maka program akan mencetak “Ini adalah segitiga” jika bukan kedua-duanya atau bukan “Persegi” ataupun “segitiga” maka program akan mencetak “Ini bukan persegi ataupun segitiga”.

Latihan!

1. Buatlah program yang mengecek sebuah inputan bilangan ganjil atau genap!

Contoh:



```
Masukan angka:5
5 adalah ganjil
```

Gambar 3.3

III. Nested if (If didalam If)

Apa sih yang dimaksud dengan nested *if*? nahh, jika belum tahu nested *if* dapat artikan menjadi *if* bersarang. *If* bersarang? Yah.. *if* bersarang maksudnya adalah fungsi *if* tetapi statemen didalam *if* adalah fungsi *if* lagi.

Agar lebih jelas lagi tentang nested *if* berikut adalah kerangka code nested if:

```
If(kondisi)
    If(kondisi)
        Statemen jika kondisi benar;
    Else
        Statemen jika kondisi salah;
```

Contoh program menggunakan nested if :

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

main() {
    int a=1,b=1;
    if(a==1)
        if(b==1)
            printf("variabel a dan b bernilai 1");
        else
            printf("hanya variabel a yang bernilai %d",a);
    else
        printf("variabel a bukan bernilai 1");
    getch();
}
```

Gambar 3.4

Gambar 3.4 adalah contoh program yang menggunakan nested if untuk mengecek apakah variabel a bernilai 1? Jika iya maka akan mengecek lagi apakah variabel b bernilai 1 juga? Jika iya maka program akan mencetak “variabel a dan b bernilai 1” jika hanya variabel a yang bernilai 1 maka program akan mencetak “hanya variabel a yang bernilai 1”. Jika variabel a bukan bernilai 1 maka program langsung akan mencetak “variabel a bukan bernilai 1”.

Just Info!

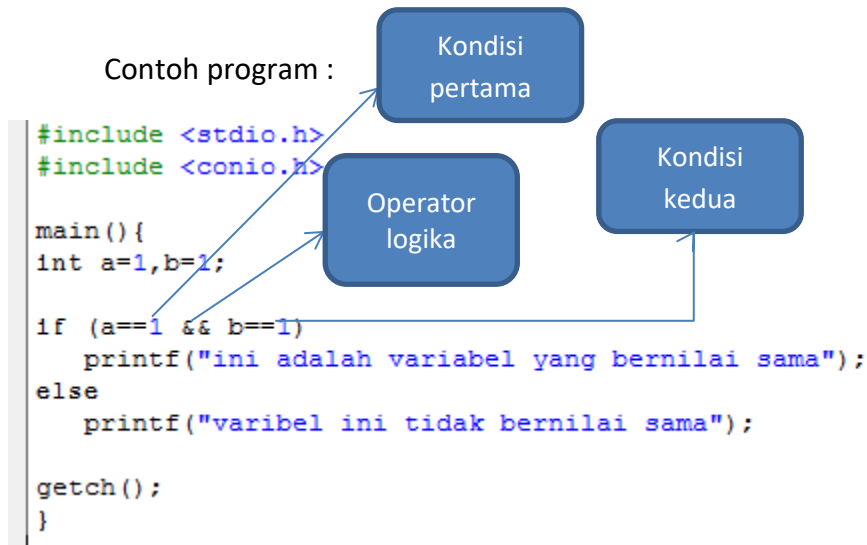
Tahukah kalian bila dalam decision kita dapat memasukan lebih dari satu kondisi? Nah jika kalian belum tahu, berikut adalah cara bagaimana menggunakan decision dengan dua kondisi:

if (kondisi_1 operator logika kondisi_2)

 Statemen jika kondisi benar;

else

 Statemen jika kondisi salah;



Gambar 3.5

Gambar 3.5 adalah contoh program yang mengecek apakah variabel a dan b sama isinnya jika sama maka program akan mencetak “ini adalah variabel yang bernilai sama”. Jika tidak sama maka program akan mencetak “variabel ini tidak sama”.

Nahh setelah kita belajar fungsi *if* pertanyaannya adalah bisa tidak jika kondisi *if* didalam mempunyai lebih dari satu statemen? Ya, jawabannya bisa. Tapi gimana caranya? Caranya adalah dengan menggunakan kurung kurawal ({statemen1; statemen2}) setelah kondisi *if*.

Berikut adalah contoh kerangka decision dengan 2 statemen:

```
If (kondisi){
    Statemen_1 jika kondisi benar;
    Statemen_2 jika kondis benar;
}else{
    Statemen_1 jika kondisi salah;
    Statemen_2 jika kondis salah;
}
```

Latihan !

Buat sebuah program bebas menggunakan lebih dari 1 statemen! Catatan tidak boleh sama satu sama lain!

a. Decision *switch case*

Decision menggunakan *switch case* adalah sebuah fungsi untuk memilih untuk kondisi yang memiliki nilai-nilai konstan (tetap/pasti). Oleh karena itu fungsi *switch case* biasa digunakan oleh variabel yang bertipe integer atau karakter.

Untuk lebih memahami tentang *switch case* berikut adalah kerangka umum *switch case*:

```
Switch(variabel){
    case nilai_konstan_1:
        statemen jika benar;
        break;
    case nilai_konstan_2:
        statemen jika benar;
        break;
    default:
        statemen jika tidak masuk semua kondisi;
}
```

Contoh program yang menggunakan *switch case*:



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

main() {
    int a;
    printf("Masukan nomor hari (1-7): "); scanf("%d", &a);
    switch(a) {
        case 1:
            printf("hari %d adalah hari senin", a);
            break;
        case 2:
            printf("hari %d adalah hari selasa", a);
            break;
        case 3:
            printf("hari %d adalah hari rabu", a);
            break;
        case 4:
            printf("hari %d adalah hari kamis", a);
            break;
        case 5:
            printf("hari %d adalah hari jum'at", a);
            break;
        case 6:
            printf("hari %d adalah hari sabtu", a);
            break;
        case 7:
            printf("hari %d adalah hari minggu", a);
            break;
        default:
            printf("error");
    }
    getch();
}
```

Gambar 3.6

Gambar 3.6 adalah salah satu program yang menggunakan fungsi *switch case* untuk mengecek hari menggunakan angka.

Just info!

Apakah *break* bagian dari fungsi *switch case*? Jawabannya bukan. Break adalah salah satu fungsi dari statemen peloncatan yang berfungsi untuk **memberhentikan** fungsi yang ada dalam suatu program dan melanjutkan ke fungsi selanjutnya. Terus kenapa kita harus menggunakan fungsi *break* di dalam fungsi *switch case*? seperti yang telah dijelaskan fungsi *break* adalah untuk memberhentikan jadi jika kita tidak menggunakan fungsi *break* maka program akan masuk kedalam kondisi yang benar dan akan menjalankan statemen yang ada didalam kondisi dan akan terus menjalankan statemen yang ada di dalam kondisi yang berada di bawah kondisi yang benar.

Let's Try!

Jika belum tahu juga maksud dari fungsi *break* yang ada di dalam fungsi *switch case* cobalah code berikut ini !

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

main() {
    int a,b;
    int c;
    printf("masukan angka pertama : ");scanf("%d",&a);
    printf("masukan angka kedua  ");scanf("%d",&b);
    printf("1.+ \n2.- \n3./ \n4.* \n5.% \n");
    printf("masuakan oprator aritmatika : ");scanf("%d",&c);
    switch(c){
        case 1:
            printf("%d + %d = %d\n",a,b,a+b);
        case 2:
            printf("%d - %d = %d\n",a,b,a-b);
        case 3:
            printf("%d / %d = %d\n",a,b,a/b);
        case 4:
            printf("%d * %d = %d\n",a,b,a*b);
        case 5:
            printf("%d ",a);printf("%");printf(" %d = %d\n",b,a%b);
    }
    getch();
}
```

Gambar 3.7

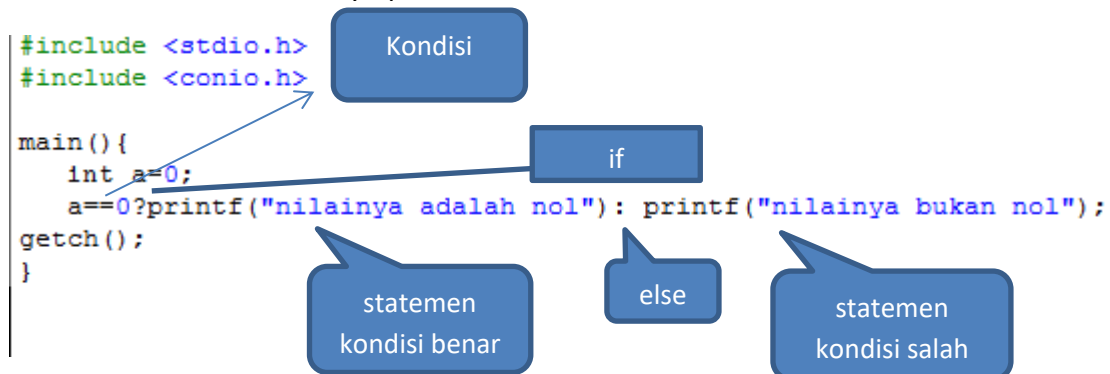
b. Ternary operator

Nahh setelah setelah kita belajar decision menggunakan *if* dan *switch case*. Di dalam bahasa C ada decision yang belum tentu ada di bahasa lain yaitu menggunakan tanda tanya (?). ama seperti fungsi decision lainnya ternary operator ini di gunakan untuk melakukan pemilihan atau pengecekan menggunakan tanda tanya (?) untuk menentukan kondisinya dan titik dua (:) untuk menentukan kondisi salah.

Berikut adalah contoh kerangka dalam code:

Kondisi ? statemen jika benar : statemen jika salah;

Contoh code ternary operator:



Gambar 3.8

Gambar 3.8 adalah contoh program yang mengecek variabel a bernilai 0 atau tidak.

Contoh code ternary operator lainnya:

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

main() {
    int a=1,b=2;
    a==b?printf("nilai a dan b sama"):b>a? printf("nilai b lebih besar dari a"):
    printf("nilai a lebih besar dari b");
    getch();
}
```

Gambar 3.9

Gambar 3.9 adalah salah satu contoh program yang mengecek apakah variabel a dan b bernilai sama jika, tidak program akan mengecek mana yang lebih besar antara variabel a dan b.

D. Review

1. Jelaskan pengertian decision menurut pendapat masing-masing!
2. Buat sebuah inputan yang menentukan grade suatu nilai!

Catatan:

- a. Jika nilai kurang dari 50 akan mencetak grade F
 - b. Jika nilai lebih dari sama dengan 50 akan mencetak grade D
 - c. Jika nilai lebih dari 59 akan mencetak grade C
 - d. Jika nilai lebih dari 69 akan mencetak grade B
 - e. Jika nilai lebih dari sama dengan 80 dan kurang dari sama dengan 100 akan mencetak grade A
 - f. Selainnya akan mencetak error.
3. Buatlah program kalkulator sederhana!

Contoh program:

```
Masukan angka pertama:3
Masukan angka kedua:7
1.+
2.-
3.*
4./
5.%
pilih operator : 3
3 * 7 = 21
```

Gambar 3.10

4. Buatlah sebuah program yang meminta inputan bertipe integer kemudian cek apakah bilangan ganjil atau genap

Catatan:

- Menggunakan switch case

BAB IV PERULANGAN/LOOPING DAN STATEMENT PELONCATAN

A. Perulangan/Looping

1. Pengertian perulangan/looping

Seperti yang kita ketahui perulangan asal kata dari ulang yang berarti mengulangi sesuatu hal yang sama. Di dalam bahasa pemrograman pun ada juga fungsi yang namanya looping atau sering disebut perulangan. Untuk apa kita belajar perulangan? Nahh perulangan atau looping sendiri bertujuan untuk menjalankan sesuatu yang sama sebanyak yang kita mau atau yang kita butuhkan. Contoh kita ingin mencetak "hello world" sebanyak 5 kali. Maka kita bisa menggunakan fungsi looping. Bukannya lebih mudah jika kita hanya menggunakan fungsi print sebanyak 5 kali? Iya memang bisa tetapi bagaimana jika kita ingin mencetak "hello world" menjadi 100 kali atau bahkan 1000 kali? Apakah masih memungkinkan kita menggunakan fungsi print sebanyak 1000 kali? Nahh makannya kita menggunakan fungsi looping tujuannya agar program kita lebih efisien atau efektif.

Sebagai gambaran program menggunakan looping dengan yang tidak menggunakan looping:

- Contoh program tidak menggunakan looping :

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

main() {
    /*contoh program mencetak hello world tanpa looping*/
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");
    printf("Hello world");

    getch();
}
```

Gambar 4.1

- Contoh program dengan menggunakan looping :

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

main() {
    /*contoh program mencetak hello world menggunakan looping*/
    for (int a=1;a<=10;a++){
        printf("Hello world");
    }
    getch();
}
```

Gambar 4.2

Program gambar 4.1 dan gambar 4.2 adalah program yang sama mencetak "hello world" sebanyak 10 kali. Tetapi terlihat jelaskan program yang lebih efisien atau efektif antara gambar 4.1 dengan 4.2.

Didalam bahasa C juga fungsi looping ada 3 yaitu :

a. For

Secara umum kerangka fungsi *for* adalah sebagai berikut:

```
for(kondisi1;kondisi2;kondisi3){  
    Statemen_yang_akan_di_ulang;  
}
```

Sedangkan jika untuk dua statemen atau lebih:

```
for(kondisi1;kondisi2;kondisi3){  
    Statemen_yang_akan_di_ulang_1;  
    Statemen_yang_akan_di_ulang_2;  
    .....  
}
```

Note:

- Kondisi1 untuk nilai awal dari perulangan atau sering disebut *counter/penghitung*
- Kondisi2 untuk kondisi atau batas perulangan
- Kondisi3 untuk menaikkan (*increment*) atau menurunkan(*decrement*) kondisi

sebagai contoh dalam program yang menggunakan looping atau perulangan:

```
#include <stdio.h>  
#include <conio.h>  
main() {  
    for (int a=1;a<=10;a++)  
        printf("%d ",a);  
  
    getch();  
}
```

Gambar 4.3

```

#include <stdio.h>
#include <conio.h>
main(){
for (int a=1;a<=10;a++){
    printf("%d",a);
    printf(" saya sedang belajar looping\n");
}

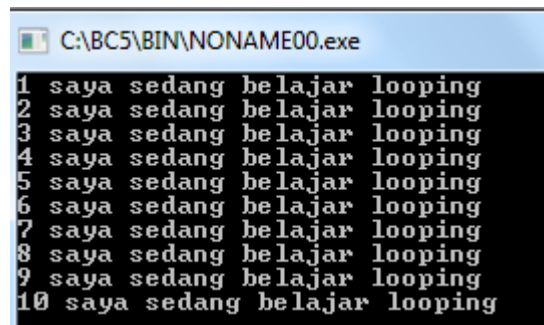
getch();
}

```

Gambar 4.4

Gambar 4.3 adalah salah satu contoh program menggunakan looping satu statement yang mencetak angka 1 sampai 10.

Sedangkan gambar 4.4 adalah salah satu contoh program menggunakan looping dengan lebih dari satu statement yang akan mencetak:



Gambar 4.5

Latihan!!!

1. Buat sebuah program yang akan mencetak:
 - a. Bilangan ganjil dari 1-19
 - b. Bilangan genap dari 2-20
2. Buat sebuah program yang akan mencetak bilangan prima saja yang inputan sebagai batas akhirnya!

Contoh :



Gambar 4.6

b. While

Setelah kita mempelajari looping atau perulangan dengan fungsi *for* sekarang kita akan mempelajari looping atau perulangan menggunakan fungsi *while*. Apasih perbedaan *while* dengan *for* ? sebenarnya sama saja mau *while* ataupun *for*, fungsi ini digunakan untuk menjalankan program dengan statemen yang sama sebanyak yang kita mau. Tetapi fungsi *while* biasanya digunakan untuk melakukan perulangan, tetapi perulangan tersebut tidak tahu akan berapa kali melakukan perulangan. nahh untuk lebih jelasnya berikut adalah kerangka fungsi *while* pada umumnya:

```
while(kondisi){
    Statement_yang_diulang_1;
    Statement_yang_diulang_2;
}
```

Nahh di dalam fungsi *while* kita tidak bisa mendeklar variable di dalam kondisi. Jika kita mendeklarnya di dalam kondisi maka program akan infinite loop atau perulangan tak terhenti. Berikut contoh penggunaan *while* yang benar:

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

main() {
    int a=1;
    while (a<=5){
        printf("saya sedang belajar fungsi while\n");
        a++;
    }

    getch();
}
```

Gambar 4.7

Gambar di atas adalah contoh program looping yang menggunakan *while* yang akan mencetak "saya sedang belajar fungsi *while*" dengan variable *a* sebagai counter-nya, 5 sebagai batas akhirnya dan *a++* sebagai increment-nya.

Jika memakai fungsi *for* maka akan menjadi:

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

main() {
    for(int a=1;a<=5;a++){
        printf("saya sedang belajar fungsi while\n");
    }
    getch();
}
```

Gambar 4.8

c. Do-while

Nahh setelah kita belajar looping menggunakan *while* dan *for* sekarang ada satu lagi dalam bahasa C looping menggunakan yang namanya *do-while*. Apasih bedanya *do-while* dengan fungsi

looping yang lainnya? Sebenarnya sama aja dengan fungsi looping lainnya hanya saja do-while ini memiliki keistimewaan di bandingkan fungsi looping lainnya yaitu **do-while pasti akan masuk kedalam statement yang ada didalamnya terlebih dahulu tanpa ada pengecekan terlebih dahulu**. Jadi kesimpulannya jika do-while pasti masuk ke dalam statementnya sementara while ataupun for belum tentu masuk kedalam statementnya apabila tidak masuk ke dalam kondisinya, kalau do while pasti akan menjalankan statement yang ada didalamnya meskipun kondisinya salah.

Untuk lebih jelasnya berikut kerangka do-while secara umum:

```
do{
    Statement_ke_1;
    Statement_ke_2;
    ....
} while (kondisi);
```

Dan berikut contoh programnya:

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

main() {
    int s=1;
    do{
        printf("Saya sedang belajar do-while\n");
        s++;
    }while (s<=5);

    getch();
}
```

Gambar 4.9

Gambar 4.9 adalah salah satu contoh program yang menggunakan do-while yang akan mencetak "saya belajar do-while" sebanyak 5 kali, dan jika kita memakai fungsi while maka :

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

main(){
    int s=1;
    while (s<=5){
        printf("saya sedang belajar do-while");
    }
    getch();
}
```

Gambar 4.10

Pada gambar 4.10 program akan mencetak "saya sedang belajar do-while" sebanyak 5 kali dan outputnya pun sama seperti program gambar 2.9 . Untuk mengetahui perbedaan antara while dengan do-while maka cobalah 2 codingan berikut ini :

- Menggunakan do-while:

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

main(){
    int s=10;
    do{
        printf("saya sedang belajar do-while\n");
    }while (s<=5);
    getch();
}
```

Gambar 4.11

- Menggunakan while:

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

main(){
    int s=10;
    while (s<=5){
        printf("saya sedang belajar do-while\n");
    }
    getch();
}
```

Gambar 4.12

Jadi, kesimpulan dari gambar 4.11 dan gambar 4.12 adalah seperti yang telah di jelaskan bahwa do-while pasti akan masuk dan melakukan statement yang ada didalamnya lalu melakukan pengecekan di akhir. Sementara while belum tentu masuk kedalam statement yang ada didalamnya karena harus melakukan pengecekan terlebih dahulu.

d. Nested loop

Apa itu nested loop? seperti yang telah di ajarkan pada BAB sebelumnya bahwa nested adalah sebuah fungsi yang statementnya adalah fungsi tersebut lagi. Jadi, nested loop adalah sebuah perulangan yang statement di dalam perulangan tersebut adalah perulangan lagi.

Untuk lebih jelasnya berikut adalah contoh kerangka nested loop pada umumnya :

```
for(kondisi1;kondisi2;kondisi3){
    for(kondisi4;kondisi5;kondisi6){
        Statement_1;
        ....
    }
}
```

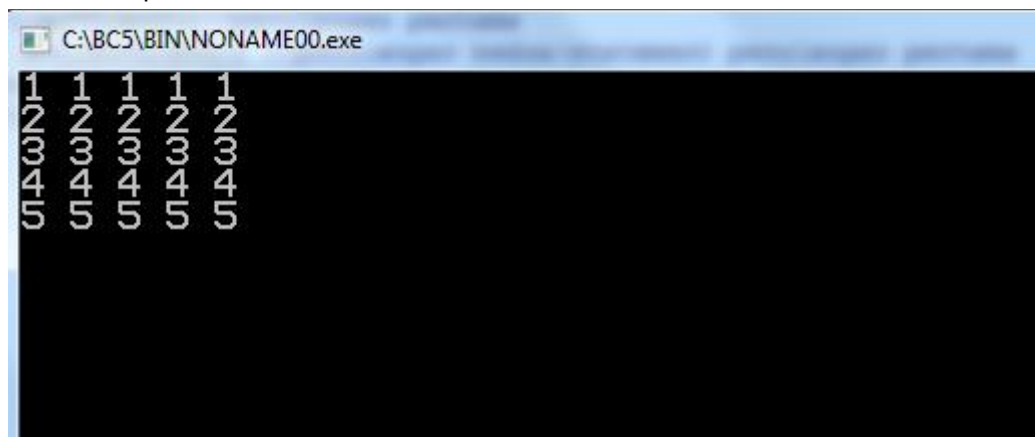
Jika masih kurang paham maka lihat codingan berikut ini dan coba kalian amatilah baik-baik outputnya:

```
#include <stdio.h>
#include <conio.h>
main() {
    for(int a=1;a<=5;a++){//perulangan pertama
        for(int f=1;f<=5;f++){ //perulangan kedua/statement perulangan pertama
            printf("%d ",a);//statement perulangan kedua
        }printf("\n"); //statement perulangan pertama
    }
    getch();
}
```

Gambar 4.13

Gambar 4.13 adalah contoh program yang menggunakan looping dengan batas awal 1 dan batas akhir 5. Dan statement dari looping adalah looping lagi maka program akan melakukan looping sebanyak looping yang di dalam looping dan akan diulang sebanyak looping yang di luar(pertama).

Output:



Gambar 4.14

B. Statement peloncatan

1. pengertian

Statement peloncatan pada umumnya digunakan pada saat kita sedang looping untuk memindahkan, melewati, ataupun memberhentikan looping tersebut. Di dalam bahasa C terdapat empat statement peloncatan, yaitu break, continue, goto, exit() .

2. Jenis-jenis statement perulangan

a. Break

Seperti yang telah di jelaskan dalam BAB sebelumnya bahwa break digunakan untuk memberhentikan sebuah fungsi.

Berikut adalah contoh pernggunaan break :

```
#include <stdio.h>
#include <conio.h>
main() {
    for(int a=1;a<=5;a++){//perulangan
        printf("Hello world\n");//statement di dalam looping
        if(a==3){//kondisi untuk memberhentikan
            break;//fungsi unntuk memberhentikan
        }
    }
    getch();
}
```

Gambar 4.15

Pada gambar 4.15 program seharusnya mencetak “hello world” sebanyak 5 kali, tetapi dikarenakan ada fungsi break pada saat a==3 maka program hanya akan mencetak 3 kali saja.

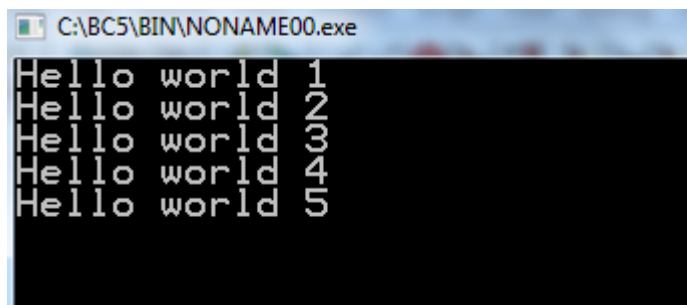
b. Continue

Statement pelompatan continue adalah statement untuk melewati statement yang ada dibawahnya dan akan kembali keatas untuk melanjutkan perulangan. contoh program yang menggunakan statement perloncatan continue :

```
#include <stdio.h>
#include <conio.h>
main() {
    for(int a=1;a<=5;a++){//perulangan
        if(a==3){//kondisi untuk memberhentikan
            continue;//fungsi unntuk melewati
        }
        printf("Hello world %d\n",a);
    }
    getch();
}
```

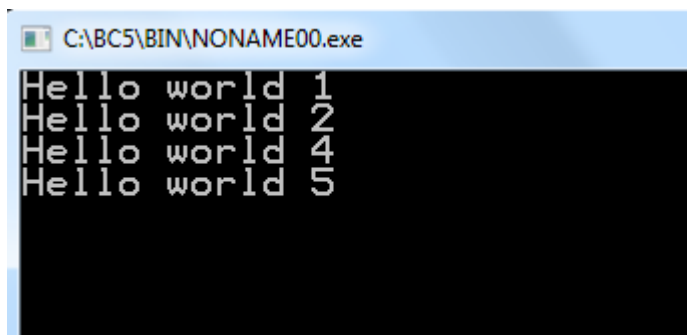
Gambar 4.16

Gambar 4.16 adalah sebuah program yang akan seharusnya akan mencetak :



Gambar 4.17

Tetapi diakarenakan di dalam looping terdapat kondisi a==3 dan statement konidisi tersebut adalah statement pelompatan continue maka outputnya menjadi :



Gambar 4.18

c. Goto

Statement pelompatan di dalam bahasa C yang terakhir adalah goto. Goto dapat diartikan menjadi go yang artinya “pergi” dan to yang artinya “ke”. Jadi, Statement goto ini dapat berfungsi untuk memindahkan alur program dari baris tertentu ke baris lain yang kita pilih.

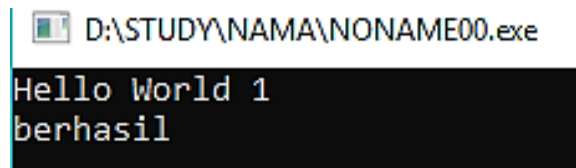
Untuk lebih jelasnya berikut adalah contoh program yang menggunakan goto:

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

main(){
    for(int a=1;a<=5;a++){ //perulangan
        if (a==2){ //kondisi
            goto pindah; // syntax statement goto untuk memindahkan alur program
        }
        printf("Hello World %d\n",a); //statement
    }
    pindah: //label untuk melanjutkan alur program
    printf("berhasil");
    getch();
}
```

Gambar 4.19

Output:



Gambar 4.20

Dari gambar 4.19 dan gambar 4.20 terlihat jelas fungsi goto bahwa ketika kondisi pada program true(benar) maka program langsung berpindah ke baris yang terdapat label “pindah” dan melanjutkan program seperti biasa kembali.

Just info!

Statement peloncatan goto kita bisa gunakan dimana saja yang kita mau. Jadi tidak harus di dalam fungsi perulangan saja.

d. Fungsi Exit()

Berbeda dengan statement break fungsi exit() digunakan untuk memberhentikan program dimanapun yang kita mau. Jika kita ingin menggunakan fungsi exit() ini maka kita harus menambahkan file header <stdlib.h>

Untuk mengetahui lebih lanjut dari fungsi exit() maka coba codingan berikut ini:

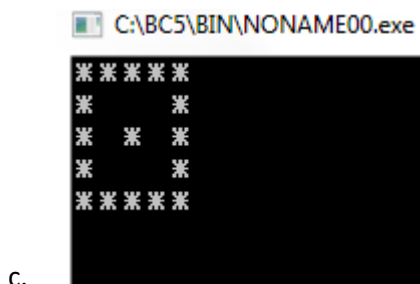
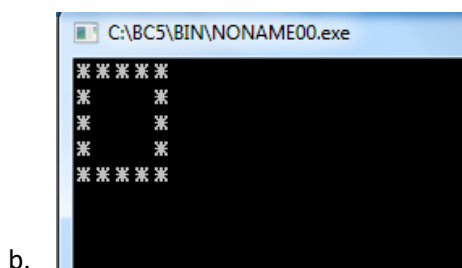
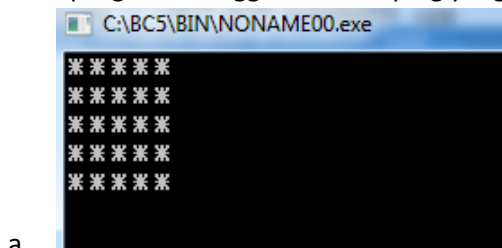
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
main() {
    int s=0;
    do{
        printf("inputkan angka 1 untuk keluar program\n");
        printf ("masukan angka : ");scanf("%d",&s);
        if (s==1)
            exit(0);
        else
            printf("anda mennginputkan angka %d\n",s);
    }while (s!=1);

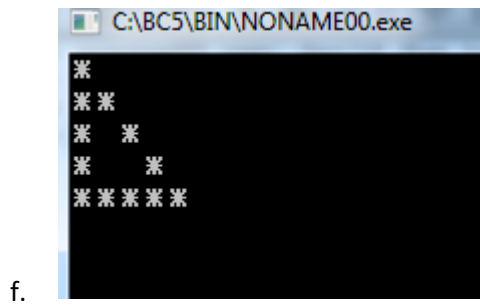
    printf("berhasil");
    getch();
}
```

Gambar 4.21

C. Review

1. Buat program menggunakan looping yang akan mencetak:





[Hint : menggunakan nested loop dan selection!](#)

2. Sebutkan menurut pendapat masing-masing perbedaan antara:
 - a. Break
 - b. Continue
 - c. Goto
 - d. Exit()

BAB V PROCEDURE & FUNCTION

Dalam membangun sebuah program yang besar tak dapat dihindari, ratusan bahkan ribuan kode program terlihat dilayar. Dan dari ribuan kode tersebut terdapat kode-kode yang ditulis dalam program, oleh karena itu pada bab ini akan dikenalkan prosedur dan function dimana akan membantu programmer meminimalisir kodingan dan juga kode tersebut bisa dipanggil berkali-kali. Selain efisien, hemat memori, juga baris kode dalam program tersebut menjadi efektif.

Lalu bagaimana cara untuk membuat suatu blok program dapat dipakai kembali di blok program ini? Dalam bahasa C dikenal dengan nama Fungsi dan Prosedur.

A. Pengertian Fungsi dan Prosedur

Fungsi atau Prosedur merupakan suatu blok program (sub program) yang digunakan untuk melakukan tugas-tugas tertentu yang **letaknya terpisah** dari program utamanya (main). Tugas tersebut bisa berupa perintah input, output maupun melakukan penyelesaian/perhitungan.

Perbedaan Fungsi dan Prosedur :

| Fungsi | Prosedur |
|---|--|
| Fungsi akan mengembalikan nilai hasil pengolahan kepada pemanggilnya | Prosedur tidak mengembalikan nilai kepada pemanggilnya |
| Return type-nya berupa tipe data seperti int, float, double, char, dll. Tergantung pada nilai apa yang akan kita kembalikan kepada si pemanggil | Return type-nya bertipe void atau kosong (tidak ada nilai kembalian) |
| Terdapat statement/tulisan 'return' (tanpa tanda petik) dalam blok programnya | Tidak terdapat statement/tulisan 'return' (tanpa tanda petik) dalam blok programnya |
| Dalam pemanggilan fungsi dibutuhkan penampung nilai hasil kembalian dari fungsi yang dipanggilnya | Dalam pemanggilan prosedur tidak dibutuhkan penampung karena tidak ada nilai yang dikembalikan |

B. Penulisan Prosedur

1. Prosedur Tanpa Parameter

Prosedur tanpa parameter adalah sebuah kode blok yang bisa mengeksekusi apa yang ada didalam prosedur tanpa bisa menerima lemparan dari prosedur lain. Jadi akan dieksekusi ketika dipanggil. Cara umum penulisan prosedur adalah sebagai berikut :

```
void NamaProsedur ( ) {  
  
    Statement (pernyataan);  
  
}
```


Penjelasan :

- void artinya kosong, maksudnya prosedur ini tidak mengembalikan nilai apa-apa (kosong).
- NamaProsedur adalah tempat untuk menamai prosedur yang kita buat.
- Tanda ' () ' buka dan tutup kurung ini untuk menandakan bahwa kode tersebut adalah sebuah prosedur yang bisa dipanggil.
- Tanda ' { } ' buka dan tutup kurawal ini adalah tempat untuk menandakan bagian/ statement tersebut adalah milik prosedur tersebut.

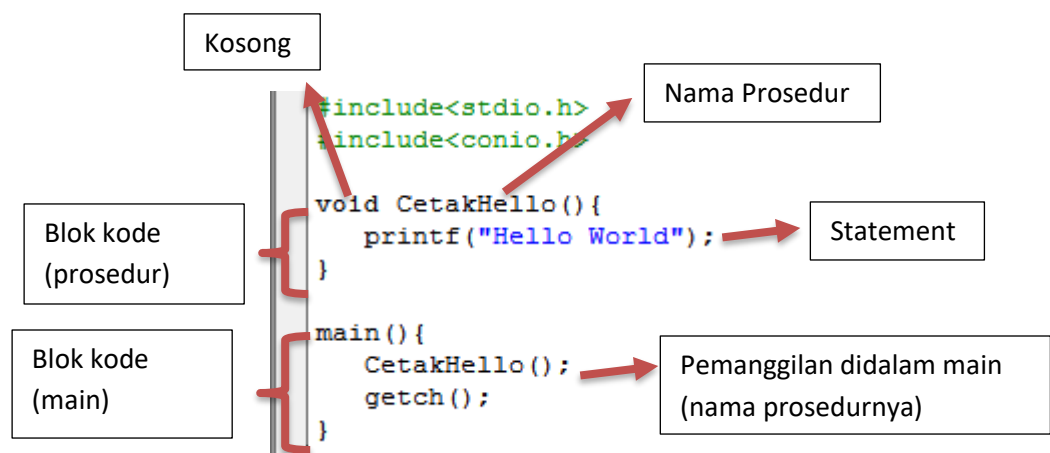
Cara pemanggilan :

NamaProsedur();

Penjelasan :

- Didalam main kita cukup menyebutkan NamaProsedur dan diakhiri () juga semicolon (;) karena didalam main,prosedur tersebut sudah merupakan statement yang dipanggil.

Agar lebih jelas perhatikan kode dibawah ini :



Gambar 5.1

DIINGAT ! Pemanggilan juga bisa didalam sebuah prosedur lain, tidak hanya dalam main namun prosedur yang memanggil prosedur lain pun harus dipanggil didalam main()

Jadi program tidak akan menjalankan prosedur jika tidak dipanggil oleh main, namun tetap dieksekusi dan menyebabkan error jika salah satu prosedur memiliki syntax eror.

2. Prosedur Dengan Parameter

Prosedur dengan parameter adalah blok kode yang menerima lemparan data dari main atau fungsi lain untuk dikelola lagi dalam prosedur tersebut. Tetapi sama seperti Prosedur sebelumnya prosedur ini juga hanya menerima lemparan parameter saja tidak bisa mengembalikan nilainya. Cara umum penulisan adalah sebagai berikut :

```
void NamaProsedur (parameter){  
  
    statement;  
  
}
```

Penjelasan :

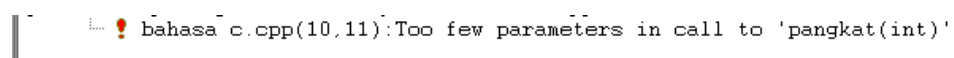
- Void artinya kosong, maksudnya prosedur ini tidak mengembalikan apa-apa (kosong).
- NamaProsedur adalah tempat untuk menamai prosedur yang kita buat.
- (parameter) adalah wadah untuk menerima lemparan dari kode yang memanggilnya.
- Tanda '{ }' buka dan tutup kurawal ini adalah tempat untuk menandakan bagian/ statement tersebut adalah milik prosedur tersebut.

Cara pemanggilan :

```
NamaProsedur(lemparan);
```

Penjelasan :

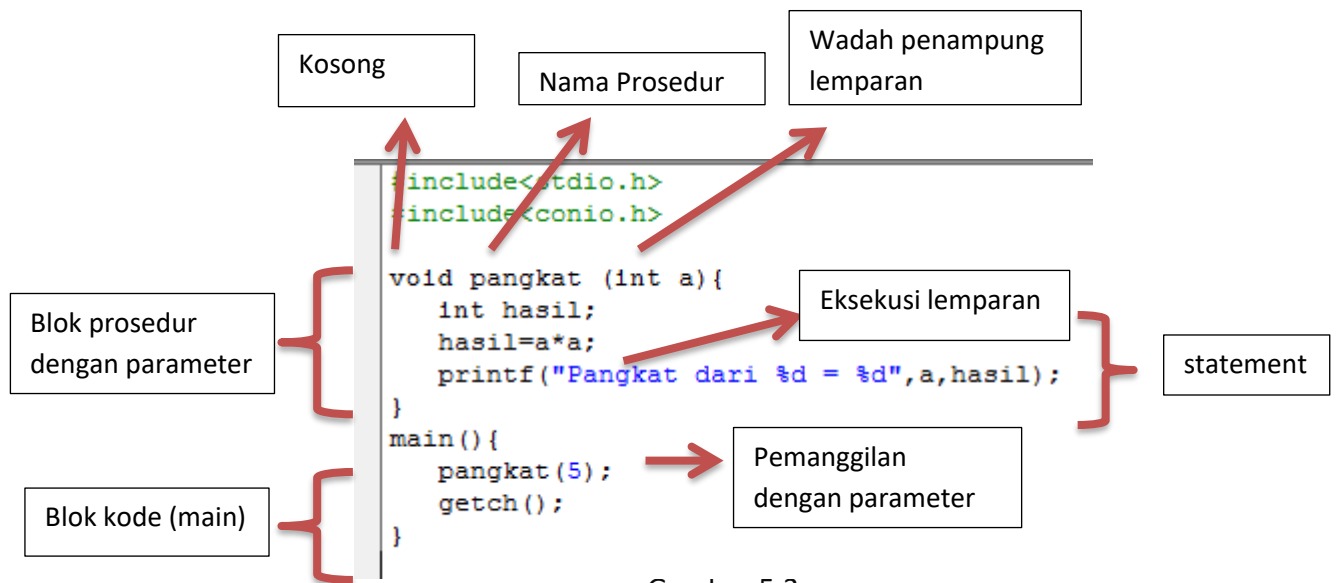
- Pemanggilan Prosedur dengan parameter sama seperti prosedur tanpa parameter hanya saja didalam tanda '()' kurung , kita harus menyertakan lemparan, jika kita memanggil prosedur yang memiliki parameter tapi dalam main kita memanggil tanpa menyertakan lemparan atau menyertakan lemparan yang tidak sesuai tipe datanya maka akan muncul laporan error seperti berikut :



```
... bahasa c.cpp(10,11): Too few parameters in call to 'pangkat(int)'
```

Gambar 5.2

Perhatikan kode berikut :



Gambar 5.3

CATATAN:

Perbedaan Parameter dengan Argumen:

- **Parameter** → nama variabel yang ada didalam prosedur/fungsi.

Misal: (void tambah(int a, int b)) .

- **Argumen** → nilai yang dilempar ke parameter prosedur/fungsi pada saat pemanggilan.

Misal: (tambah(4,6);).

C. Penulisan Function (Fungsi)

1. Function tanpa parameter

Function tanpa parameter adalah sebuah blok kode yang tidak bisa menerima lemparan karena tidak memiliki parameter namun function ini bisa mengembalikan nilai karena function memiliki return type. Return type adalah sebuah tipe data yang menampung nilai yang dikembalikan (di-return) oleh function, tentunya return type ini harus sesuai dengan apa yang ingin kita kembalikan. Bentuk umum penulisan function adalah :

```
ReturnType NamaFunction(){
    Statement;
    Return;
}
```

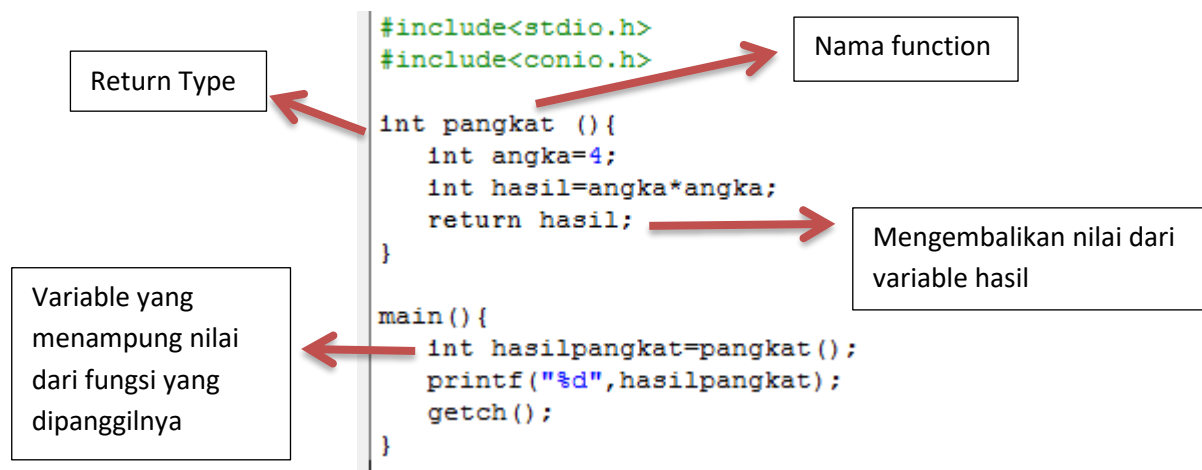
Penjelasan :

- Return Type adalah sebuah type data seperti (int,float,long,double dll) disesuaikan dengan kebutuhan pengembalian nilai.
- NamaFunction adalah tempat untuk memberi nama function tersebut
- Return kata kunci untuk mengembalikan nilai.

Cara pemanggilan function juga sama seperti prosedur, hanya jika kita ingin nilai yang dikembalikan oleh function tersebut maka kita harus membuat tampungan juga, contoh :

```
Main(){  
    NamaVariabel>NamaFunction();  
}
```

Untuk lebih jelasnya pahami kodingan dibawah ini !



Gambar 5.4

2. Function dengan parameter

Function dengan parameter adalah blok kode yang menerima lemparan (parameter) dari siapa yang memanggilnya juga siap mengembalikan nilai tersebut. Analoginya seorang bos adalah sebagai pemanggil function dan function adalah anak buahnya yang siap mengerjakan dengan bahan yang dikirimkan oleh si bos (lemparan) kemudian diterima oleh anak buahnya(parameter) dan data tersebut diolah oleh anak buah (function) ketika selesai data tersebut dikembalikan kepada si bos(pemanggil). Penulisan dalam kodingan bahasa C adalah :

```
ReturnType NamaFunction(parameter){  
    Statement;  
    return statement;  
}
```

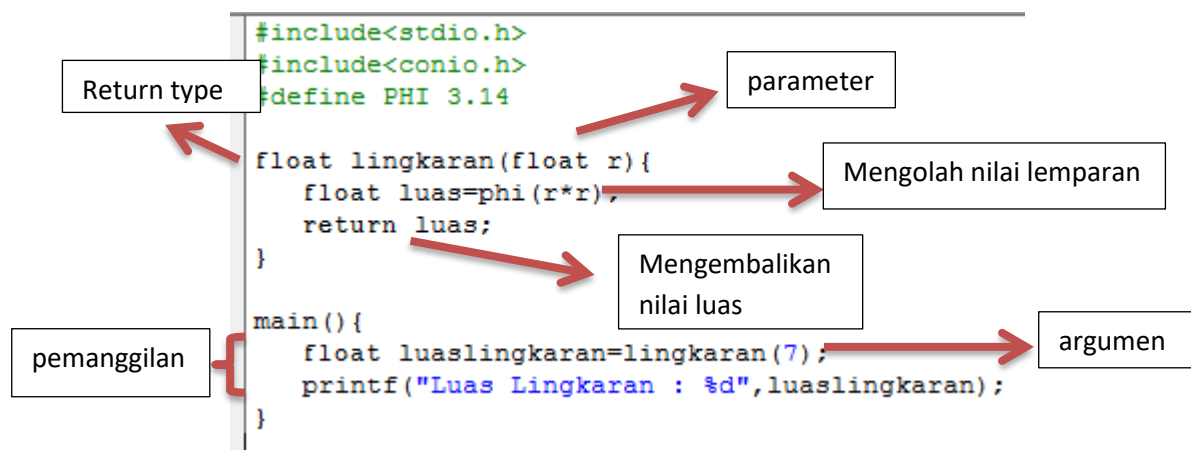
Penjelasan :

- Sama hal dengan prosedur parameter adalah wadah penampung lemparan.

Pemanggilan :

```
Main(){  
    NamaVariabel NamaFunction(argumen);  
}
```

Perhatikan kode dibawah ini :



Gambar 5.5

Note :

Dalam fungsi dan prosedur kita bebas menjabarkan apa saja, seperti menggunakan perulangan, pilihan if else disesuaikan dengan kebutuhan. Dan untuk lebih mengefisienkan program perhatikan kapan kita harus menggunakan fungsi dan kapan menggunakan prosedur.

D. Call by Value dan Call By References

Didalam pemanggilan parameter kedalam sebuah fungsi terdapat dua cara, yaitu cara panggilan berdasarkan nilainya (call by value) dan berdasarkan alamat variabelnya (call by reference).

1. Call By Value

Pada panggilan parameter berdasarkan nilai, terjadi proses penyalinan (copy) nilai dari argument ke parameter. Hal ini akan menyebabkan nilai variable yang dihasilkan oleh proses didalam fungsi tidak akan berpengaruh terhadap nilai variable yang terdapat di luar fungsi. Untuk lebih memahaminya perhatikan contoh program berikut.

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

int TambahSatu(int a){
    a+=1;
    printf("Nilai didalam fungsi : %d\n",a);
}

int main(){
    int bilangan;
    printf("Masukan bilangan bulat : ");
    scanf("%d",&bilangan);

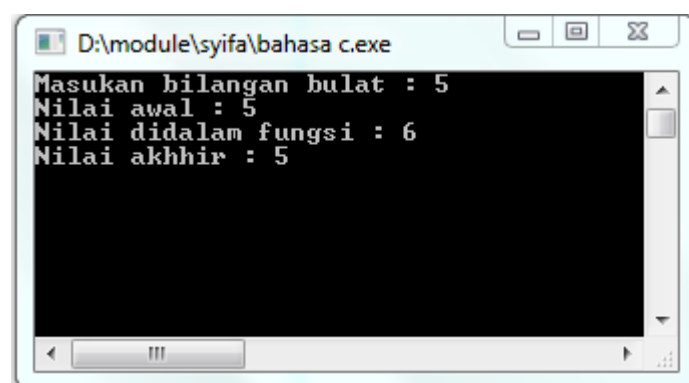
    printf("Nilai awal : %d\n",bilangan);

    TambahSatu(bilangan);

    printf("Nilai akhhir : %d\n",bilangan);
    getch();
}
```

Gambar 5.6

Kode tersebut akan menghasilkan output :



Gambar 5.7

Seperti yang kita lihat pada hasil program diatas bahwa nilai dari variable bilangan tetap bernilai 5 walaupun kita telah memanggil fungsi TambahSatu(). Hal ini disebabkan karena variable bilangan dan variable a merupakan dua variable yang tidak saling berhubungan

dan menempati alamat memori yang berbeda sehingga yang terjadi hanyalah proses penyalinan (peng-copy-an) nilai dari variable bilangan ke variable a. dengan demikian, perubahan nilai variable a tentu tidak akan mempengaruhi nilai variable bilangan.

2. Call By Reference

Parameter yang dipanggil kedalam fungsi bukanlah berupa nilai, melainkan suatu alamat memori. Pada saat kita memanggil parameter berdasarkan alamat, terjadi proses referensial antara variable yang terdapat pada parameter dengan variable yang terdapat pada argumen. Hal tersebut menyebabkan kedua variable tersebut akan berada pada satu alamat dimemori yang sama sehingga apabila terdapat perubahan nilai terhadap salah satu dari variable tersebut, maka nilai variable satunya juga akan ikut berubah. Perhatikan contoh program berikut !

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

int TambahSatu(int *a){
    *a+=1;
    printf("Nilai didalam fungsi : %d\n",*a);
}

int main(){
    int bilangan;
    printf("Masukan bilangan bulat : ");
    scanf("%d",&bilangan);

    printf("Nilai awal : %d\n",bilangan);

    TambahSatu(&bilangan);

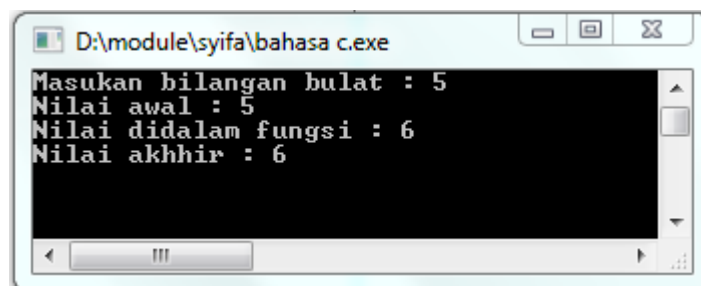
    printf("Nilai akhhir : %d\n",bilangan);
    getch();
}
```

Tanda '*' merupakan symbol bahwa yang diterima alamat bilangan

Tanda '&' merupakan symbol bahwa yang dikirim adalah alamat bilangan

Gambar 5.8

Output program :



Gambar 5.9

Seperti yang kita lihat diatas bahwa proses call by reference, perubahan nilai didalam fungsi akan mempengaruhi nilai diluar fungsi.

E. Rekursif

Rekursif artinya berulang. Rekursif adalah proses pemanggilan fungsi oleh dirinya sendiri secara berulang. Rekursif digunakan untuk penyederhanaan algoritma dari suatu proses sehingga program yang dihasilkan menjadi efisien.

a. Menentukan Nilai Faktorial

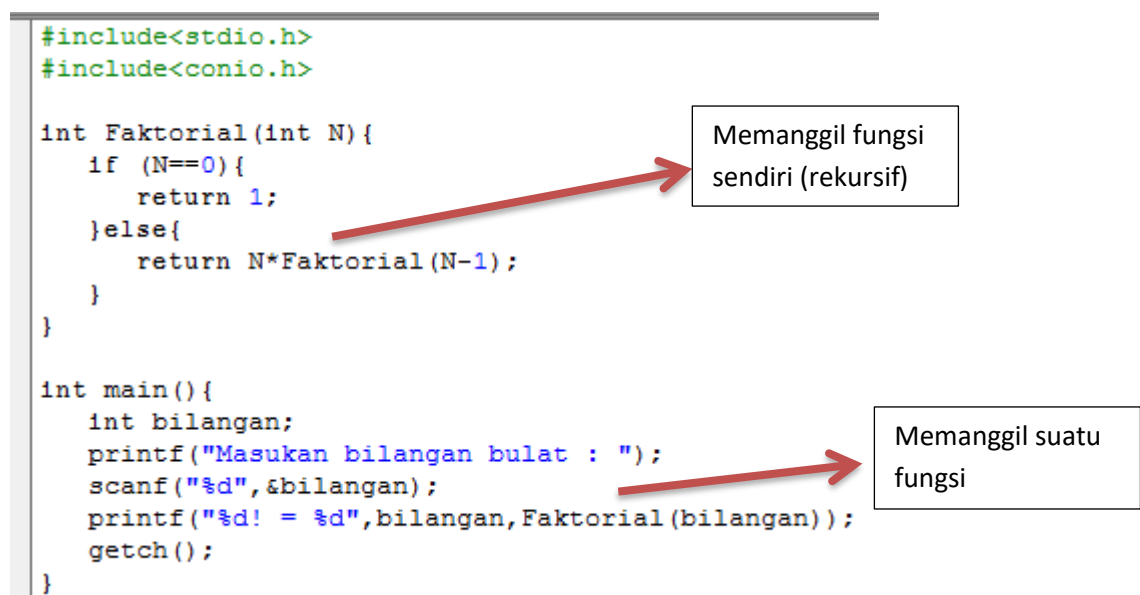
Pada bagian ini kita akan membuat sebuah fungsi rekursif untuk menentukan nilai faktorial dengan memasukan nilai yang akan dihitung sebagai parameter fungsi ini. Sebagai contoh apabila parameter yang kita masukkan adalah 5, maka hasilnya adalah

$$5!=5 \times 4 \times 3 \times 2 \times 1 = 120$$

Proses tersebut dapat kita sederhanakan melalui fungsi matematis sebagai berikut :

$$F!(N) = N * F!(N-1)$$

Namun yang harus kita perhatikan disini adalah $F!(0) = 1$, ini adalah suatu tetapan numerik yang tidak dapat diubah. Berikut ini contoh implementasi kasus tersebut ke dalam sebuah program.



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

int Faktorial(int N){
    if (N==0){
        return 1;
    }else{
        return N*Faktorial(N-1);
    }
}

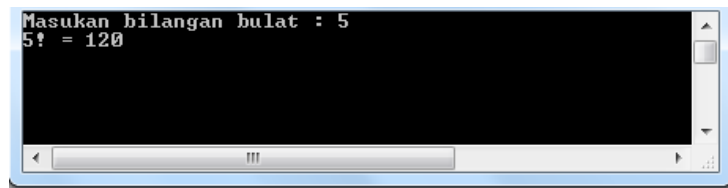
int main(){
    int bilangan;
    printf("Masukan bilangan bulat : ");
    scanf("%d",&bilangan);
    printf("%d! = %d",bilangan,Faktorial(bilangan));
    getch();
}
```

Memanggil fungsi sendiri (rekursif)

Memanggil suatu fungsi

Gambar 5.10

Output program :



Gambar 5.11

Konsep dari proses diatas sebenarnya sederhana, yaitu dengan melakukan pemanggilan fungsi Faktorial() secara berulang. Untuk kasus ini, proses yang dilakukan adalah sebagai berikut .

$$\text{Faktorial}(5)=5*\text{Faktorial}(4)$$

$$\text{Faktorial}(4)=4*\text{Faktorial}(3)$$

$$\text{Faktorial}(3)=3*\text{Faktorial}(2)$$

$$\text{Faktorial}(2)=2*\text{Faktorial}(1)$$

$$\text{Faktorial}(1)=1*\text{Faktorial}(0)$$

$$\text{Faktorial}(0)=1$$

$$\text{Faktorial}(1)=1*1$$

$$\text{Faktorial}(2)=2*1$$

$$\text{Faktorial}(3)=3*2$$

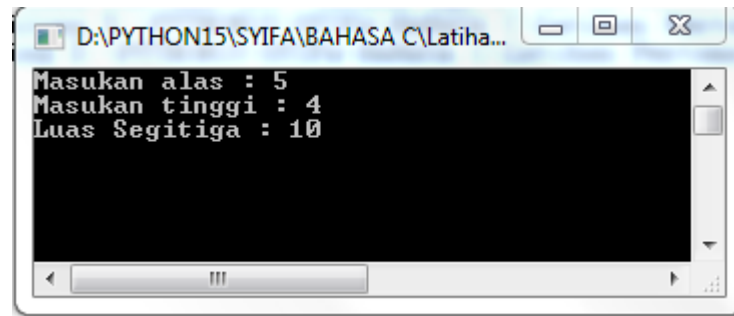
$$\text{Faktorial}(4)=4*6$$

$$\text{Faktorial}(5)=5*24$$

$$\mathbf{=120}$$

F. Review

- Buatlah sebuah function yang bernama LuasSegitiga yang memiliki 2 parameter alas dan tinggi, mencetak didalam main(), contoh output :



- Sebutkan perbedaan fungsi dan prosedur dan apa perbedaan call by value call by reference antara keduanya
- Tebak output : Berapakah hasil kesatu dan hasil kedua?

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

int TebakOutput(int x){
    x=15;
    x+=x;
    return x;
}

int main(){
    int x=10;
    printf("Hasil kesatu : ",TebakOutput(x));
    printf("Hasil kedua : ",x);
    getch();
}
```

- Apakah prosedur yang memiliki parameter ketika dipanggil tidak diisi parameternya?
- Buatlah sebuah rekursif tentang perpangkatan.

Hint : B^n dimana B adalah bilangan basis. Rumus fungsi tersebut $B^n = B * B^{n-1}$

BAB VI ARRAY

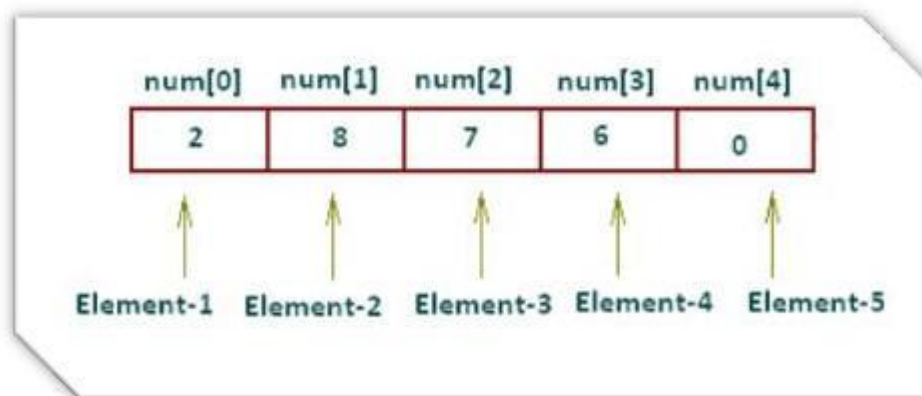
A. Pengertian Array

Secara bahasa, array berarti susunan atau jajaran. Bayangkanlah bahwa suatu array adalah satu kereta yang memiliki banyak gerbong. Setiap gerbong memiliki identitas masing-masing dan memiliki jumlah penumpang yang berbeda. Istilah untuk menunjuk suatu gerbong yaitu index (yang berarti penunjuk. Ingat bahwa index finger itu berarti jari telunjuk).

Harus diingat bahwa index array secara default dimulai dari 0 (nol) bukan 1. Array bisa dianalogikan dengan sebuah kereta api dengan gerbongnya adalah index-nya. Argo adalah nama array-nya, kepala kereta adalah index ke-0, gerbong ke-1 adalah index ke-1, dan seterusnya. Nilai-nilai data dari suatu array disebut dengan element-element array.

Secara teknis array merupakan sebuah variabel yang dapat menyimpan lebih dari 1 buah data yang memiliki tipe data yang sama. Jadi dapat dikatakan bahwa array merupakan kumpulan dari data-data tunggal yang dijadikan dalam 1 variabel array yang alamat memorinya berbeda yang selanjutnya disebut elemen-elemen array yang bisa kita akses berdasarkan index. Aturan penamaan array juga sama seperti penamaan variabel biasa, tidak boleh diawali angka, ataupun mengandung karakter khusus kecuali tanda garis bawah `_`, jika penamaan terdapat dua kata tidak boleh dipisah oleh spasi, boleh menggunakan garis bawah atau digabung secara **camel case**, yaitu penulisan yang naik turun seperti punuk unta Contoh: `iniNamaVariabel`.

Gambaran Array:



Gambar 6.1

Deklarasi array:

tipeDataArray namaArray [ukuran/panjang array];

Contoh:

int nilai[20];

Dari contoh diatas kita bisa lihat bahwa sebuah variabel array telah dideklarasikan dengan tipe data integer yang memiliki ukuran 20, yang berarti bisa menampung 20 data yang dimulai dari index ke-0 sampai index ke 19. Kenapa tidak sampai 20? Karena index array yang paling awal itu kan dimulai dari 0, dan kalau dihitung sampai 19, maka akan terdapat 20 data.

Cara pengisian/inisialisasi array:

- Cara pertama:

Statis:

```
nilai[0]=100;
```

```
nilai[1]=75;
```

```
....
```

```
Nilai[19]=90;
```

Dinamis:

```
scanf("%d", &nilai[0]);
```

```
scanf("%d", &nilai[1]);
```

```
....
```

```
scanf("%d", &nilai[19]);
```

- Cara kedua

Menggunakan perulangan (for):

```
for(int x=0;x<20;x++){
```

```
    scanf("%d", &nilai[x]);
```

```
}
```

Menggunakan perulangan (while):

```
int x=0;
```

```
while(x<20){
```

```
    scanf("%d", &nilai[x]);
```

```
    x++;
```

```
}
```

Cara mencetak array:

- Cara pertama:

```
printf("%d", nilai[0]);
```

```
printf("%d", nilai[1]);
```

```
....
```

```
printf("%d", nilai[19]);
```

- Cara kedua

Dengan menggunakan perulangan for:

```
for(int x=0;x<20;x++){
```

```
    printf("%d\n", nilai[x]);
```

```
}
```

Dengan menggunakan perulangan while:

```
int x=0;
while(x<20){
    printf("%d\n", nilai[x]);
    x++;
}
```

Tipe Data Array

Array tidak hanya dapat menampung tipe data integer/bilangan bulat saja, tetapi tipe data lain juga seperti float, char, ataupun string dapat dijadikan tipe data suatu array. Yang membedakan adalah cara inputannya dan mencetaknya, yaitu format specifier-nya yang berbeda. Misal, jika format specifier int adalah “%d” atau “%i” maka untuk float adalah “%f”, untuk char adalah “%c”, dan untuk string adalah “%s”. Berlaku juga untuk tipe data long (“%ld”/”%li”) dan double (“%lf”).

Contoh sederhana:

```
float nilaiUang[4];
nilaiUang[0]=1000.00;
printf(“Nilai uang: %f”, nilaiUang[0]);
```

```
char huruf[26];
huruf[0]='a'; (catatan: untuk inisialisasi karakter menggunakan kutip satu/’ ’);
printf(“Huruf alfabet pertama adalah: %c”, huruf[0]);
```

```
char* namaMahasiswa[3];
(catatan: Ini dapat menampung string. Tanda * menunjukkan bahwa ia adalah tipe data pointer,
nanti akan dipelajari di pelatihan dan perkuliahan Struktur Data)
namaMahasiswa[0] = "Zaki";
(catatan: untuk inisialisasi string menggunakan kutip dua/” ”);
printf(“Nama mahasiswa pertama adalah: %s”, namaMahasiswa[0]);
(catatan: mencetak menggunakan format specifier “%s”).
```

B. Tujuan Mempelajari Array

Array akan sangat berguna dimana suatu saat kita ingin menampung banyak data untuk suatu kepentingan yang nanti datanya akan kita manfaatkan. Seperti data nilai satu kelas, daftar nama, daftar grade kuliah, mencari total keseluruhan nilai, dll. Dan akan dipergunakan banyak di masa yang akan datang seperti perkuliahan dan pekerjaan.

Dengan menggunakan array, sejumlah variabel dapat memakai nama yang sama. Antara satu variabel dengan variabel lain di dalam array dibedakan berdasarkan subscript. Sebuah subscript berupa bilangan di dalam kurung siku [dan]. Misal ada sebuah array angka[5], variabel angka[0] dengan angka[3] itu berbeda dan isinya bisa berbeda, tetapi memiliki nama yang sama, kan?

C. Jenis-jenis Array

a. Array Satu Dimensi

Yaitu bentuk array yang paling umum digunakan. Contohnya sudah dijelaskan pada bagian A diatas. Pada bentuk ini setiap elemen array dapat diakses melalui index, index tersebut secara default dimulai dari 0 dan berikut deklarasi array satu dimensi dalam bentuk umum:

tipe_array nama_array[ukuran];

Tapi bagaimana jika kita ingin menyamakan nilai dari tiap element menjadi 0? Begini caranya:

int angka[5]={0};

Maka nilai dari index 0 sampai 4 adalah 0.

Tapi bagaimana jika begini?

int angka[5]={6};

Maka hasilnya yaitu index ke-0 berisi angka 6 sedangkan index 1,2,3,4 berisi 0. Kenapa? Karena begitu standar penulisan jika ingin menyetting nilai awal semua elemen array menjadi 0 di bahasa C.

Kita bisa juga mendeklarasikan suatu variabel array tanpa ukuran/panjang array, tetapi ada aturan penulisannya. Berikut contohnya:

int umur[]={};

atau

int umur[]={20,18,21,21};

Penjelasan:

Pada bentuk penulisan diatas kenapa tidak error? Karena array tersebut sudah sekaligus diinisialisasi setelah dideklarasikan, walaupun tidak memiliki ukuran. Untuk mengisinya bisa menggunakan cara manapun. Contoh: umur[0]=19;

Untuk contoh yang kedua, dia memiliki index dari 0 sampai 3, tapi apa jadinya kalau kita mencetak index ke 4 dan seterusnya? Maka yang akan tampil adalah angka acak yang merupakan lokasi memori di komputer yang biasa disebut dengan *garbage value* (nilai sampah).

Tetapi jika penulisannya begini maka akan error:

```
int umur[];
```

Karena dianggap tidak memiliki ukuran dan tidak punya nilai apapun.

CONTOH:

```
//Deklarasi array
```

```
int a[5];
```

```
//Inisialisasi elemen array
```

```
a[0]=6;
```

```
a[1]=3;
```

```
a[2]=6;
```

```
a[3]=0;
```

```
a[4]=1;
```

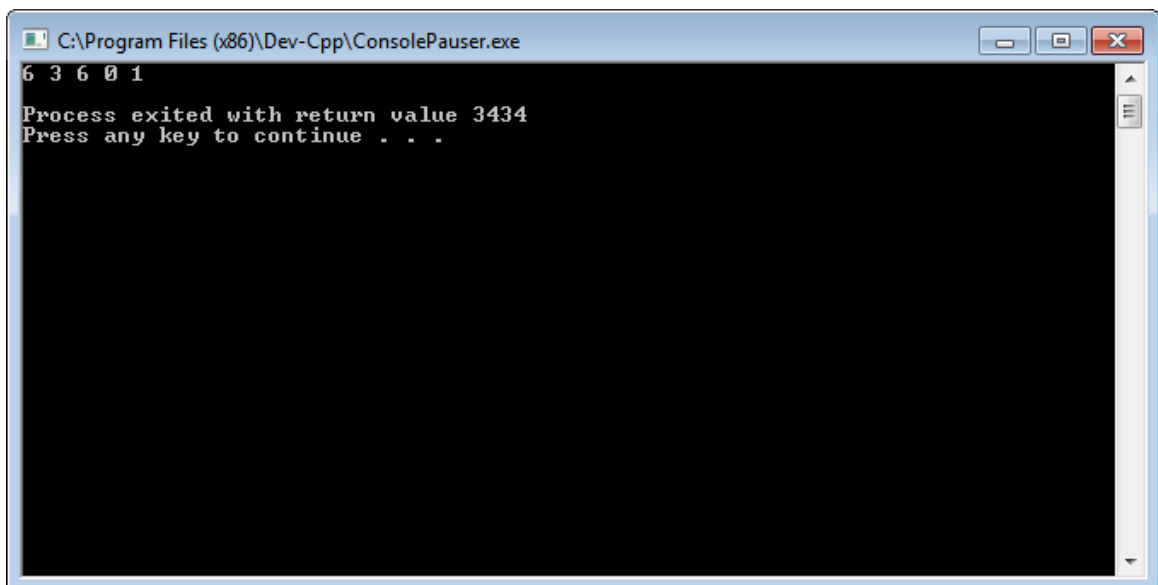
```
//Perulangan untuk mencetak array
```

```
for(int x=0;x<5;x++){
```

```
    printf("%d ",a[x]);
```

```
}
```

MAKA OUTPUTNYA:



```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
6 3 6 0 1
Process exited with return value 3434
Press any key to continue . . .
```

Gambar 6.2

b. Array Dua Dimensi

Array dua dimensi merupakan array yang terdiri dari 2 subskrip yaitu sebuah m baris dan sebuah n kolom. Bentuk array dua dimensi dapat berupa matriks atau tabel. Dan pengaksesannya melalui dua index.

Deklarasi array dua dimensi:

tipeDataArray namaArray [baris][kolom];

Contoh deklarasi:

int angka[4][3];

Contoh pengisian:

```
angka[0][0]=1;
angka[0][1]=2;
angka[0][2]=3;
angka[1][0]=4;
....
angka[3][1]=11;
angka[3][2]=12;
```

Gambaran:

| | | |
|----|----|----|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| 10 | 11 | 12 |

Perhatikan bahwa tabel diatas memiliki 4 baris dan 3 kolom, dan pengisian index sebelah kiri untuk baris sedangkan yang sebelah kanan untuk kolom.

Cara mencetak:

- Cara biasa:

```
printf("%d ",angka[0][0]);
```

- Menggunakan perulangan:

Kita bisa menggunakan nested-loop atau perulangan bersarang untuk mendapatkan hasil seperti diatas.

```
for(int a=0;a<4;a++){ ← Perulangan untuk baris
```

```
    for(int b=0;b<3;b++){ ← Perulangan untuk kolom
```


`printf("%d ",angka[a][b]);` ← Mencetak elemen array sesuai baris dan kolom

`}`

`printf("\n");` ← Membuat garis baru/newline ketika perulangan kolom selesai

`}` ← Akhir perulangan. Akan kembali ke perulangan pertama hingga kondisi tidak terpenuhi lagi

CONTOH:

`//Deklarasi array`

`int a[3][3];`

`//Inisialisasi elemen array`

`a[0][0]=9;`

`a[0][1]=8;`

`a[0][2]=7;`

`a[1][0]=6;`

`a[1][1]=5;`

`a[1][2]=4;`

`a[2][0]=3;`

`a[2][1]=2;`

`a[2][2]=1;`

`//Perulangan untuk mencetak array`

`for(int x=0;x<3;x++){`

`for(int y=0;y<3;y++){`

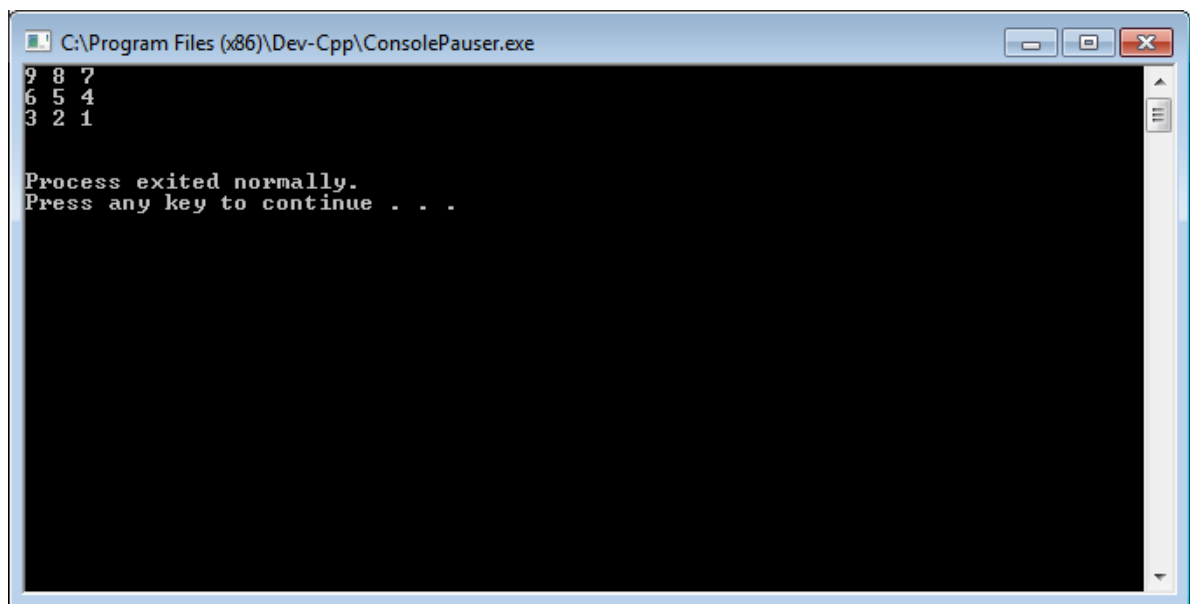
`printf("%d ",a[x][y]);`

`}`

`printf("\n");`

`}`

MAKA OUTPUTNYA:



Gambar 6.3

c. Array Multi Dimensi

Array multi dimensi adalah suatu array yang mempunyai subskrip lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua.

Deklarasi array multi dimensi:

tipeDataArray namaArray [ukuran 1][ukuran 2] ... [ukuran n];

Contoh deklarasi:

int angka[2][2][3];

Maka array akan mempunyai 2 baris dan 2 kolom, nah setiap cellnya mempunyai 3 data.

Contoh pengisian:

angka[0][0][0]=7;

angka[1][0][1]=9;

Cara mencetak:

- Cara biasa:

printf("%d ",angka[0][0][0]); ← Yang tercetak adalah angka 7

- Menggunakan perulangan:

for(int a=0;a<2;a++){ ← Perulangan ke-1

for(int b=0;b<2;b++){ ← Perulangan ke-2

```

        for(int c=0;c<3;c++){ ← Perulangan ke-3
            printf("%d ",angka[a][b][c]); ← Mencetak angka
        }
        printf("\n"); ← Mencetak baris baru jika perulangan ke-1 selesai
    }
    printf("\n"); ← Mencetak baris baru jika perulangan ke-2 selesai
} ← Akhir perulangan. Akan kembali ke perulangan pertama hingga kondisi tidak terpenuhi lagi

```

CONTOH:

```

//Deklarasi array
int angka[2][2][3];

//Inisialisasi elemen array (menggunakan perulangan For)
for(int a=0;a<2;a++){
    for(int b=0;b<2;b++){
        for(int c=0;c<3;c++){
            angka[a][b][c]=a+b+c;
        }
    }
}

//Perulangan untuk mencetak array
for(int a=0;a<2;a++){
    for(int b=0;b<2;b++){
        for(int c=0;c<3;c++){
            printf("%d ",angka[a][b][c]);
        }
        printf("\n");
    }
    printf("\n");
}

```

MAKA OUTPUTNYA:



```
C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
0 1 2
1 2 3
1 2 3
2 3 4

Process exited normally.
Press any key to continue . . . _
```

Gambar 6.4

D. Review

1. Apa pengertian Array menurut kamu?
2. Gambarkan konsep Array dengan menggunakan objek di dunia nyata (selain kereta)!
3. Buat program yang memiliki array 1 dimensi yang dapat menampung data nilai 20 siswa dalam satu kelas! Inputan harus dinamis dan dicetak menggunakan perulangan While!
4. Buat program yang memiliki array 2 dimensi yang dapat menampung data nilai dari 3 kelas dalam 1 sekolah dan tiap kelas memiliki 10 siswa! Inputan harus dinamis dan dicetak menggunakan perulangan For!
5. Buat program warung sederhana yang menampung 5 nama barang beserta harganya. Inputkan nama dan harga tersebut lalu cetak menggunakan perulangan!
6. Buat program untuk menjumlah semua nilai array di setiap index lalu dicetak! Isinya bebas!