MODUL PRAKTIKUM

DASAR PEMROGRAMAN

MENGGUNAKAN BAHASA C

Oleh : Fauzan Ismail, MEng.Sc

INSTITUT TEKNOLOGI PADANG

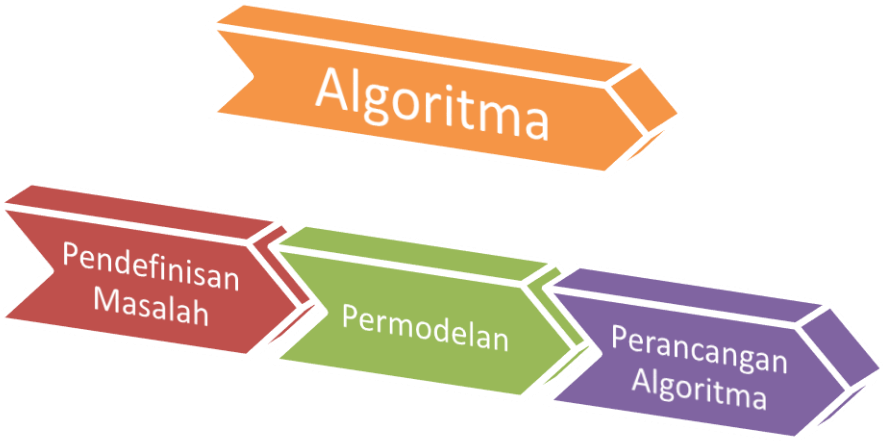
PADANG, 2016

KATA PENGANTAR

DAFTAR ISI

PENGANTAR BAHASA C

Bahasa Pemrograman secara umum digunakan untuk memberikan kemudahan dalam menyelesaikan suatu masalah yang dapat diproses menggunakan komputer. Permasalahan yang ada kemudian di rumuskan langkah-langkah penyelesaiannya secara terstruktur agar dapat di proses dengan bantuan komputer disebut dengan Algoritma. Ini menjadi hal penting dalam proses pemrograman. Algoritma menggambarkan alur pemikiran dalam menyelesaikan permasalahan tersebut.

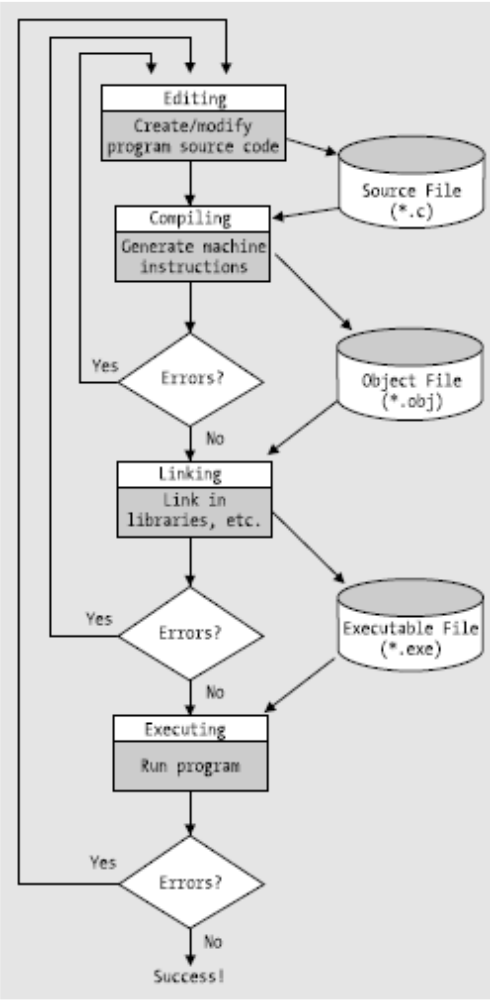


Gambar.1 Proses penyusunan Algoritma

Untuk perumusan suatu algoritma diperlukan tiga tahapan penting yaitu, pendefinisian masalah, permodelan, dan perancangan algoritma. Tiga tahapan tersebut terlihat pada gambar.1. *Tahapan pertama*, pendefinisian masalah mencakup kejelasan terhadap masalahnya, pemilihan metoda atau rumusan yang dapat dipakai untuk menyelesaikannya, dan identifikasi data atau parameter penting yang berperan dalam menyelesaikan masalah tersebut. *Tahapan kedua*, permodelan merupakan menjadikan metoda yang dipilih untuk dapat di jadikan model matematika atau bentuk permodelan lainnya. *Tahapan ketiga*, Perancangan Algoritma yaitu menyusun diagram alur proses penyelesaian masalah dengan sistematik dan terstruktur. Setelah ketiga proses tersebut dilakukan barulah kita bisa memulai untuk melanjutkannya pada proses emrograman menggunakan salah satu bahasa pemrograman.

Bahasa C merupakan bahasa pemrograman yang berada diantara bahasa tingkat rendah dan bahasa tingkat tinggi. Selain memiliki keunggulan dari bahasa pemrograman tingkat tinggi, yaitu menggunakan perintah yang mudah dipahami manusia, bahasa C juga mempunyai kecepatan eksekusi mendekati kecepatan eksekusi bahasa tingkat rendah dan juga kemampuan memanipulasi alamat data secara langsung. Keunggulan inilah yang kemudian menjadikan bahasa C banyak dipilih oleh programer komputer untuk menyusun aplikasi komputer.

Untuk membuat program menggunakan bahasa C diperlukan empat dasar tingkatan ataupun proses yaitu Editing, Compiling, Lingking, dan Executing. ***Pertama***, Editing merupakan proses pembuatan ataupun modifikasi listing program (source code). Ada beberapa C Compiler dengan editor khusus yang banyak memberikan bantuan dalam mengatur suatu program. Editor juga sering memberikan fitur yang lengkap untuk menulis, mengatur, mengembangkan, dan pengujian. Ini sering disebut Integrated Development Environment (IDE). ***Kedua***, Compiling mengubah source code menjadi bahasa mesin dan mendeteksi eror report pada proses compile. Input pada tingkatan ini adalah file yang dihasilkan selama proses pengeditan diwakilkan oleh source file. Dalam proses compile akan menghasilkan **object code**, dan distore dalam file disebut **object file** dengan extensi file **.obj**. ***Ketiga***, Linker meggabungkan file yang dihasilkan oleh compiler dari file source code, menambahkan file code yang diperlukan dari sumber program lirary sebagai bagian dari C. Program library mendukung pengembagan dari bahasa C dengan memberikan routine untuk mengambil operasi yang bukan bahagian dari bahasa pemrograman. Contohnya, library yang berisikan routine yang mendukung operasi seperti performa input dan output, perhitungan akar perpangkatan, dan perbandingan dua karakter string. Keempat, Executing adalah proses menjalankan program, dimana semua proses sudah sukses keseluruhannya. File execute ditandain dengan ekstensi **.exe**.



Gambar.2. Pembuatan dan eksekusi program

Bagaimanakah membuat program untuk pertamakali?

Pertama sekali aktifkanlah software editor / compiler C, pilihlah new project dan ketik listing program dibawah ini pada layar editor,

```
/* Program 1.1 Your Very First C Program - Displaying Hello World */
#include <stdio.h>

int main(void)
{
  printf("Hello world!");
  return 0;
}
```

Simpanlah file project pada suatu direktori dengan suatu nama yang berekstension **.c**, compile lah program untuk memastikan apakah ada error pada listring program. Jika program berhasil tanpa eror maka silahkan run program untuk melihat hasil seperti gambar dibawah ini,

```
Hello world!
```

Jika ingin mengubah, menambah atau memperbaiki listing program, maka bekerjalah pada layar editor;

```
#include<stdio.h>

int main(void)
{
  printf("If at first you don\'t succeed, try, try, try again!");
  return 0;
}
```

Bagaimanakah jika terjadi kesalahan pengetikan seperti listing dibawah ini, apa yang ditampilkan pada display output.

```
#include<stdio.h>

int main(void)
{
  printf("If at first you don\'t succeed, try, try, try again!")
  return 0;
}
```

Apabila di compile, maka akan ditampilkan pesan error pada layar output, seperti berikut;

```
Syntax error : missing ';' before '}'
HELLO.C - 1 error(s), 0 warning(s)
```

MODUL 1 : FUNGSI MASUKAN DAN FUNGSI KELUARAN

1.1. Tujuan Praktikum

- Memberikan pemahaman untuk menampilkan data keluaran program dan memberikan data masukan pada program.

1.2. Teori Dasar

Sebagai pengetahuan dasar pahamilah penggunaaan software compiler C sebaik mungkin dan instal sesuai keperluan. Listing program yang akan di tulis pada layer editor harus di mengerti untuk menghindari kesalahan dan error. Perhatikanlah listing dibawah ini;

```
#include<stdio.h>           /* preprosesor            */

int Umur;                   /* Global Variabel        */
int main()                  /* identifikasi Fungsi main   */
{                           /* tanda Mulai Fungsi main    */
  char Nama[10];                /* Local Variabel         */
  printf("Latihan Modul I: ");  /* perintah menampilkan   */
  scanf("%s",&Nama);            /* perintah membaca       */
  printf("Nama : %s \n",Nama);  /* perintah menampilkan   */
  scanf("%d",&Umur);            /* perintah membaca       */
  printf("Umur : %d \n",Umur);  /* perintah menampilkan   */
}
```

Perintah **`#include`** merupakan jenis preprosesor yang meminta komputer untuk membaca file **`stdio.h`** (yang dikategorikan sebagai file header) sebelum memulai perintah lainnya. File header (ekstensi **.h**) berisikan sekumpulan fungsi yang digunakan dalam listing program yang disusun. Misalkan, perintah **`printf`** yang digunakan ada pada file **`stdio.h`**, dimana **stdio** merupakan singkatan dari **standart input/output**.

Selanjutnya ada fungsi **`main()`**, merupakan fungsi utama yang harus ada dalam listing program C. Perhatikan dua bentuk kerangka fungsi utama yang sering digunakan;

1. Fungsi utama tanpa pengembalian nilai ke sistem operasi.

```
void main(void)
{
    // kode program yang akan di tulis
}
```

2. Fungsi utama dengan pengembalian nilai ke sistem operasi.

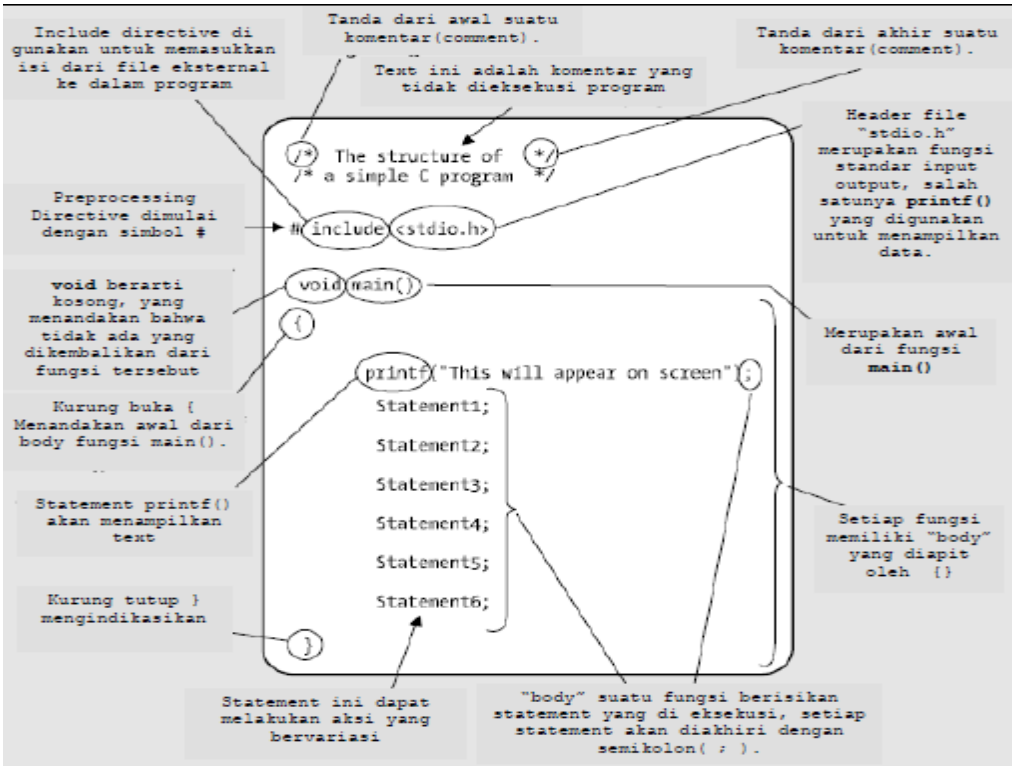
```
int main(void)
{
    // kode program yang akan di tulis
    Return 0;
}
```

Dari dua bentuk diatas di perlihatkan bahwa fungsi **`main`** ada yang mengembalikan nilai ke sistem operasi, dan ada yang tidak. Untuk bentuk kerangka yang menggunakan **`int`** (tipe fungsi) berarti fungsi ini akan mengembalikan nilainnya ke sistem operasi. **`Return 0`** berarti mengembalikan nilai nol (0) ke sistem, yang menyatakan bahwa tidak terjadi error pada program atau program berjalan dengan benar.

Kemudian perhatikanlah fungsi **`printf()`** dan fungsi **`scanf()`**. Fungsi **`printf()`** merupakan perintah untuk menampilkan string control atau argumen fungsi tersebut ke layar. Sedangkan fungsi **`scanf()`** merupakan perintah masukan. String yang akan ditampilkan oleh **`printf()`** dan format variabel yang dibaca oleh perintah **`scanf()`** diapit oleh tanda “. Untuk tanda **`\n`** yang terlihat pada listing program diatas bermaksud untuk meminta kursor berpindah satu baris ke bawah, di istilahkan dengan ***escape sequense***. Berikut daftar escape sequence di perlihatkan pada tbel dibawah;

Tabel 1. Escape Sequence

Escape Sequence	Keterangan
\n	Menyatakan karakter baris baru
\r	Menyatakan return atau kembali
\b	Menyatakan backspace
\f	Menyatakan form-feed character
\t	Menyatakan tab horizontal
\v	Menyatakan tab vertikal
\a	Memasukkan karakter bel peringatan
\?	menampilkan tandatanya
\”	Menampilkan karakter “
\’	Menampilkan karakter ‘
\\	Menampilkan backslash (\)



Gambar 3. Elemen dari program sederhana.

1.3. Lembar kerja

1. Buatlah Listing program yang dapat menampilkan keluaran seperti tulisan di bawah ini;

   PEMROGRAMAN MENGGUNAKAN BAHASA C
   Nama : Budi Raharjo // masukkan namamu
   NIM : .....

2. Ketiklah listing program ini pada layer editor,

```
#include<stdio.h>
void main(void)
{
      printf("Latihan Modul I: \n");
      Printf("Nama : %s \n",Nama);
      return 0;
}
```

   Silahkan Compile list program diatas, perhatikanlah hasil keluarannya dan berilah penjelasanmu.

1.4. Tugas

Buatlah program yang meminta masukan nama, umur, dan jenis kelamin. Kemudian tampilkanlah ke layar. Dan simpan file dengan nama elektroNIM.c , nim dimaksud adalah nim masing-masing mahasiswa. (Contoh: **elektro2015310121.c**).

MODUL 2 : VARIABEL DAN TIPE DATA

2.1.Tujuan Praktikum

- Memberikan pemahaman mengenai tipe variabel dan bagaimana cara mendeklarasi, mengoperasikan, dan format menampilkannya.

2.2.Teori Dasar

Dalam bahasa pemrograman kita mengenal istilah variabel. Variabel merupakan pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Variabel akan menggunakan memori untuk menyimpan data atau nilai dari variabel tersebut. Setiap data akan merujuk ke alamat pada memori komputer. Adapun tipe dari variabel tersebut dijelaskan pada Tabel.2 dibawah ini,

Tabel.2 Tipe Variabel

TIPE	DEFINISI	NILAI
int	Bilangan Integer	-32768 to 32767
float	Bilangan Real	+/- 3.4e +/- 38 (~7 digits)
double	Bilangan Real dengan rentang lebih luas	+/- 1.7e +/- 308 (~15 digits)
long double	Bilangan Real dengan rentang paling luas	+/- 3.4e +/- 308 (~15 digits)
char	Karakter : bisa alfabet, simbol, operator, karakter khusus	Misal : A, b, $, + dan lainnya
boolean	Kondisi benar atau salah	Hanya ada nilai **true** untuk benar dan **false** untuk salah

Untuk variabel dengan tipe integer terdapat beberapa tipe yang panjang datanya berbeda-beda, seperti ditampilkan Tabel.3 dan Tabel.4.

Tabel.3 Tipe Data sign integer

Nama	Jumlah Byte	Batasan Nilai
singned char	1	-128 s/d +127
short int	2	-32,768 s/d +32,768
int	4	-2,147,438,648 s/d +2,147,438,648
long int	4	-2,147,438,648 s/d +2,147,438,648
long long int	8	-9,223,372,036,854,775,808 s/d +9,223,372,036,854,775,808

Tabel.4 Tipe Data unsign integer

Nama	Jumlah Byte	Batasan Nilai
unsingned char	1	0 s/d 255
unsigned short	2	0 s/d 65,535
unsigned int	4	0 s/d 4,294,967,295
unsigned long	4	0 s/d 4,294,967,295
unsigned long long	8	0 s/d 18,446,744,073,709,551,615

Untuk tipe variabel yang “floating-point” yaitu data berkoma dengan lebar keakuratan yang berbeda. Ini diperlihatkan pada Tabel.5.

Tabel.5 Tipe Variabel Floating-Point

Nama	Jumlah Byte	Batasan Nilai
float	4	±3.4E38 (6 decimal digits precision)
double	8	±1.7E308 (15 decimal digits precision)
Long double	12	±1.19E4932 (18 decimal digits precision)

Contoh Program:

```
#include <stdio.h>
int main(void)
{
   float plank_length = 10.0f;
   float piece_count = 4.0f;
   float piece_length = 0.0f;
   piece_length = plank_length/ piece_count;
   printf("A plank %f feet long can be cut into %f pieces %f feet long.",plank_length,
   piece_count, piece_length);
   return 0;
}
```

Hasil eksekusi program:

```
A plank 10.000000 feet long can be cut into 4.000000 pieces 2.500000 feet long.
```

Bagaimanakah mengatur tampilan angka dibelakang koma? Untuk itu kita harus membatasi seperti contoh di bawah ini. Gantilah baris “printf” pada contoh diatas dengan ini,

```
printf("A plank %.2f feet long can be cut into %.0f pieces %.2f feet long.",plank_length,
piece_count, piece_length);
```

angka 2 pada “%.2f” diatas menandakan jumlah angka dibelakang koma yang akan ditampilkan.

Sehingga Hasil eksekusi:

```
A plank 10.00 feet long can be cut into 4 pieces 2.50 feet long.
```

Contoh:

```
#include <stdio.h>
int main(void)
{
   float radius = 0.0f;                          /* The radius of the table */
   float diameter = 0.0f;                        /* The diameter of the table */
   float circumference = 0.0f;                   /* The circumference of the table */
   float area = 0.0f;                            /* The area of a circle */
   float Pi = 3.14159265f;
   printf("Input the diameter of the table:");
   scanf("%f", &diameter);                       /* Read the diameter from the keyboard */
   radius = diameter/2.0f;                       /* Calculate the radius */
   circumference = 2.0f*Pi*radius;               /* Calculate the circumference */
   area = Pi*radius*radius;                      /* Calculate the area */
   printf("\nThe circumference is %.2f", circumference);
   printf("\nThe area is %.2f\n", area);
   return 0;
}
```

Hasil Eksekusi:

```
Input the diameter of the table: 6
The circumference is 18.85
The area is 28.27
```

Deklarasi Konstanta

a. Menggunakan keyword const
Contoh : const float PI = 3.14152965;

b. Menggunakan #define
Contoh : #define PI 3.14152965

Berbeda dengan variable, konstanta bernama tidak dapat diubah jika telah diinisialisasi

Nama dari suatu variable dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut :

1. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa C ++ bersifat ***case-sensitive*** artinya huruf besar dan kecil dianggap berbeda. Jadi antara **nim, NIM** dan **Nim** dianggap berbeda.
2. Tidak boleh mengandung spasi.
3. Tidak boleh mengandung symbol-simbol khusus, kecuali garis bawah (underscore). Yang termasuk symbol khusus yang tidak diperbolehkan antara lain : $, ?, %, #, !, &, *, (,), -, +, =dsb.
4. Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.

Contoh Program:

```
#include<stdio.h>
#define PI 3.14159f /* Definition of the symbol PI */
int main(void)
{
   float radius = 0.0f;
   float diameter = 0.0f;
   float circumference = 0.0f;
   float area = 0.0f;
   printf("Input the diameter of a table:");
   scanf("%f", &diameter);
   radius = diameter/2.0f;
   circumference = 2.0f*PI*radius;
   area = PI*radius*radius;
   printf("\nThe circumference is %.2f", circumference);
   printf("\nThe area is %.2f", area);
   return 0;
}
```

2.3.Lembar Kerja

1. Buatlah Listing Program untuk menghitung Volume silinder dengan satuan jari-jari alas 3cm, dan tinggi 5cm.
2. Hitunglah Luas segitiga sama kaki jika alas segitiga adalah 1,5cm dan tingginya 3,5cm. Perhatikanlah hasil keluaran dari eksekusi program.

2.4.Tugas

1. Buatlah listing program untuk menghitung setengah dari volume bola yang jari jarinya 5cm.

MODUL 3 : PENGAMBILAN KEPUTUSAN (1)

(SARAT DAN KONDISI)

3.1. Tujuan Praktikum

- Bagaimanakah membuat keputusan berdasarkan perbandingan aritmatika.
- memahami struktur control if statement dan if-else statement serta dapat mengaplikasikannya.

3.2. Teori Dasar

Pengambilan keputusan dalam suatu program berkaitan dengan pilihan untuk mengeksekusi suatu statement terhadap statement lain. Perhatikanlah contoh kalimat dibawah ini;

Johan adalah seorang pelajar, setiap hari johan harus mengikuti kelas tambahan disore hari. Sepulang sekolah johan terlelap karena kelelahan. Tepat pukul dua siang, johan terbangun, dan berfikir,
"Apakah ada kelas tambahan?"
"Jika Tidak, Johan boleh melanjutkan tidurnya. Jika Ya Johan harus bersiap-siap.
"Apakah Johan Masih Lelah?"
"Jika Ya, Beristirahatlah dan minta izin. Jika Tidak Johan berangkat kesekolah untu kelas tambahan".

Perbandingan Aritmatika

operator relasi sangat diperlukan dalam pengambilan keputusan, sehingga ada 3 fundamental operator relasi, yaitu;

<	Kecil dari	>=	Besar sama
==	Sama dengan	!=	Tidak sama dengan
>	Besar dari	<=	Kecil sama

if Statement

```
if(berat_kamu > berat_saya)
    printf("Kamu lebih berat dari saya.\n");
if(berat_kamu < berat_saya)
    printf("Saya lebih berat dari kamu.\n");
if(berat_kamu == berat_saya)
    printf("Kita memiliki berat yang sama.\n");
```

Sehingga secara umum syntax untuk if statement adalah;

```
if(expression)
     Statement1;
```

Jika di dalam " if " terdapat lebih dari satu statement maka sebaiknya di tulis seperti penjelasa dibawah ii;

```
if(expression)
{
      Statement1;
      Statement2
      …….
}
```

Untuk memastikan bahwa statement tersebut hanya dieksekusi dengan kondisi expresi yang sudah ditentukan.

if-else statement

if-else statement melengkapi statement **if** yang sebelumnya kita pelajari, statement **if** hanya akan mengeksekusi statement yang sesuai dengan kondisi expresi, sedang **if-else** statement memberikan kemungkinan untuk melakukan statement lain jika kondisi expresi tidak terpenuhi. Statement tersebut dapat kita lakukan di dalam **else**. Sehingga **if-else** memiliki arti, *jika kondisi expresi terpenuhi lakukan statement di dalam "**if**", dan jika tidak terpenuhi lakukan statement didalam "**else**".*

```
if(expression)
      statement1;
else
      statement2;
```

Contoh Program:

```
#include <stdio.h>
int main(void)
{
   double unit_price = 3.50;          /* Unit price in dollars */
   int quantity = 0;
   printf("Enter the number that you want to buy:");
   scanf(" %d", &quantity);           /* Read the input */

   /* Test for order quantity qualifying for a discount */
   if(quantity>10)                    /* 5% discount */
      printf("The price for %d is $%.2f\n", quantity, quantity*unit_price*0.95);
   else                               /* No discount */
      printf("The price for %d is $%.2f\n", quantity, quantity*unit_price);
   return 0;
}
```

if-else statement juga dapat membatasi kondisinya untuk beberapa statement yang terpenuhi, seperti;

```
if(expression)
{
    StatementA1;
    StatementA2;
    ...
}
else
{
    StatementB1;
    StatementB2;
    ...
}
```

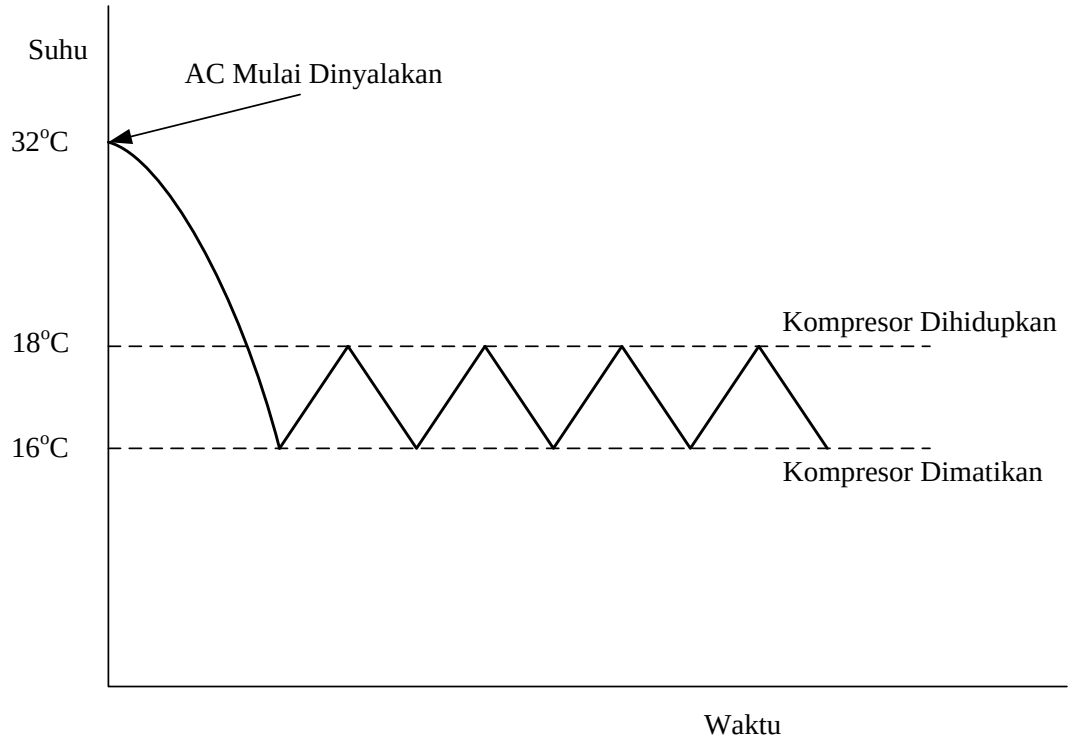
3.3. Lembar Kerja

1. Buatlah sebuah program yang dapat menentukan nilai tertinggi dari beberapa siswa di bawah ini;
   Andi = 75, Budi = 68, Arif = 78, Mira = 74, Ahmad = 65.
2. Carilah nilai rata-rata dari 5 orang siswa diatas dan tentukan berapa nilai dibawah rata
3. Berikanlah penjelasan mengenai program dibawah ini.

```
#include <stdio.h>
int main(void)
{
   double unit_price = 3.50;          /* Unit price in dollars */
   int quantity = 0;
   printf("Enter the number that you want to buy:");
   scanf(" %d", &quantity);            /* Read the input */

   /* Test for order quantity qualifying for a discount */
   if(quantity>10)                     /* 5% discount */
      printf("The price for %d is $%.2f\n", quantity,
   quantity*unit_price*0.95);
      printf("quantity besar dari 10”);
   else                               /* No discount */
      printf("The price for %d is $%.2f\n", quantity, quantity*unit_price);
      printf("quantity kecil dan sama dengan 10”);
   return 0;
}
```

3.4. Tugas

Pada grafik dibawah ini diperlihatkan suatu pengaturan suhu suatu ruangan yang menggunakan AC, diketahui suhu ruangan saat AC mati yaitu 32^{o}C. Kemudian AC dinyalakan untuk mencapai suhu 16^{o}C. Buatlah program pengendalian suhu ruangan berdasarkan grafik dibawah ini;



MODUL 4: PENGAMBILAN KEPUTUSAN (2)

(SARAT DAN KONDISI)

4.1. Tujuan Praktikum

- Apakah itu operator logika dan bagaimana menggunakannya.
- memahami struktur control if statement dan if-else statement serta dapat mengaplikasikannya.

4.2. Teori Dasar

Dalam pengambilan keputusan dirasa curang cukup apabila melakukan perbandingan satu variable saja, tetapi kadang diperlukan untuk membandingkan atau memutuskan dua atau lebih kondisi. Sehingga diperlukan operator logika.

Operators	Description	Associativity
( ) [] . ->	Parenthesized expression Array subscript Member selection by object Member selection by pointer	Left-to-right
+ - ++ -- ! ~ * & sizeof (type)	Unary + and - Prefix increment and prefix decrement Logical NOT and bitwise complement Dereference Address-of Size of expression or type Explicit cast to type such as (int) or (double) Type casts such as (int) or (double)	Right-to-left
* / %	Multiplication and division and modulus (remainder)	Left-to-right
+ -	Addition and subtraction	Left-to-right
<< >>	Bitwise shift left and bitwise shift right	Left-to-right
< <= > >=	Less than and less than or equal to Greater than and greater than or equal to	Left-to-right
== !=	Equal to and not equal to	Left-to-right
&	Bitwise AND	Left-to-right
^	Bitwise exclusive OR	Left-to-right
|	Bitwise OR	Left-to-right
&&	Logical AND	Left-to-right
||	Logical OR	Left-to-right
?:	Conditional operator	Right-to-left
= += -= /= *= %= <<= >>= &= |= ^=	Assignment Addition assignment and subtraction assignment Division assignment and multiplication assignment Modulus assignment Bitwise shift left assignment and bitwise shift right assignment Bitwise AND assignment and bitwise OR assignment Bitwise exclusive OR assignment	Right-to-left

Logika AND (&&)

Pada operator logika ini akan membanding kan dua nilai variable untuk mendapatkan suatu keputusan. Logika ini memiliki keputusan,

A	B	A && B
1	1	1
1	0	0
0	1	0
0	0	0

Contoh:

Jika A memiliki nilai decimal 69 maka nilai Binary dari A adalah 01000101, sedang kan B memiliki nilai 57 dengan Binary 00111001, maka A && B

A = 01000101

B = 00111001 &&

A &&B = 0

Jika ada yang tidak terpenuhi maka nilai hasil dari logika AND adalah nol, sehingga apabila,

```
if(A && B)
{
      // Jika A&&B bernilai 1 atau terpenuhi akan masuk looping
}else{
      // Jika A&&B bernilai 0 atau tidak terpenuhi akan masuk looping
}
```

`if(A && B)` juga dapat ditulis **`if((A && B)==0)`**

Bitwise AND (&)

Pada operator logika ini akan membanding bit dari nilai suatu variable untuk mendapatkan suatu keputusan ataupun hasil.

Contoh:

Jika A memiliki nilai decimal 69 maka nilai Binary dari A adalah 01000101, sedang kan B memiliki nilai 57 dengan Binary 00111001, maka A & B

A = 01000101

B = 00111001 &

A &&B = 00000001

Contoh lain penggunaanya;

```
1.    C  = A & B;
2.   If((A & B)>0)
      {
            // Terpenuhi
      }else{
            // Tidak Terpenuhi
      }
```

Logika OR (||)

Pada operator logika ini akan membanding kan dua nilai variable untuk mendapatkan suatu keputusan. Logika ini memiliki keputusan,

A	B	A || B
1	1	1
1	0	1
0	1	1
0	0	0

Contoh:

Jika benar itu bernilai 1 dan salah itu bernilai 0,dan Ada bernilai 1 dan Tidak_ada bernilai 0 maka;

Suatu ketika Amir Akan menghadapi Ujian kenaikan kelas, maka Amir akan memenuhi jadwal ujian tepat waktu walaupun hari hujan, asalkan Amir menggunakan mantel. Jika ternyata Hari hujan dan Amir tidak memiliki mantel amir akan mennunggu hujan reda.

```
if((Hujan==Benar)||(Mantel==Tidak_ada))
{
      printf("Amir akan menunggu hingga hujan reda");
}
if((Hujan==Benar)||(Mantel==ada))
{
      printf("Amir akan memenuhi jadwal ujian tepat waktu");
}
```

Logika NOT(!)

Pada operator logika ini akan memberikan keputusan yang berlawanan dari pernyataannya. Logika ini memiliki keputusan,

A	!A
1	0
0	1

Contoh:

```
      if(!(Hujan==Benar)||!(Mantel==Tidak_ada))
```

Maksudnya;

1. Jika tidak hujan samadengan benar atau tidak mantel sama dengan tidak ada
2. JIka Hujan tidak benar atau mantel sama dengan ada.

Statement Switch

Switch merupakan sebuah pengambilan keputusan dengan nilai yang tepat, maksudnya bukan nilai diantara, besar dari ataupun kecil dari.

```
switch(ekspresi)
{
      case 0:Pernyataan 1
             break;
      case 1:Pernyataan 2
             break;
      case 2:Pernyataan 3
             break;
      case 3:Pernyataan 4
             break;
}
```

Contoh:

```
#include <stdio.h>
int main(void)
{
      int angka;
      printf("Masukkanlah angka diantara 0 s/d 3:");
      scanf(" %d", &angka);            /* Read the input */
            switch(angka)
            {
                case 0: printf("Yang Anda inputkanAngka nol”);
                        break;
                case 1: printf("Yang Anda inputkanAngka satu”);
                        break;
                case 2: printf("Yang Anda inputkanAngka dua”);
                        break;
                case 3: printf("Yang Anda inputkanAngka tiga”)
                        break;
            }
      return 0;
}
```

4.3. Lembar Kerja

1. Ketiklah contoh program dibawah ini, dan carilah kesalahan pada listting program dibawah ini;

```
#include <stdio.h>
int main(void)
{
char angka;
printf(" ketiklah suatu hutuf kapital diantara huruf A s/d E :”);
scanf(" %c", &angka);            /* Read the input */
    switch(angka)
    {
         case A; printf("Huruf yang Anda ketik adalah A”);
                  break;
         case B: printf("Huruf yang Anda ketik adalah B”);
                  break;
         case C: printf("Huruf yang Anda ketik adalah C”);
                  break;
         case D: printf("Huruf yang Anda ketik adalah D”)
                  break;
         case E: printf("Huruf yang Anda ketik adalah E”)

    }
    return 0;
}
```

2. Perbaiki kesalahan program diatas sehingga dapat di eksekusi dengan benar.

4.4. Tugas

Carilah contoh aplikasi switch case yang sederhana.

MODUL 5: PERULANGAN (1)

(LOOP)

5.1. Tujuan Praktikum

- Memahami pengertian suatu perulangan dan bagaimana cara menggunakannya.
- Mengerti dan dapat membedakan penggunaann for dan while.
- Mampu mengaplikasikan for dan while dengan kondisi tertentu.

5.2. Teori Dasar

Perulangan ataupun Loop pada system pemrograman bermaksud untuk mengulangi statement tertentu dengan kondisi yang ditentukan. Perulangan ada yang dilakukan dengan batas suatu kondisi atau pun berkelanjutan (kontinu).

Operator increment dan decrement

Increment berarti menambahkan nilai variable dengan satu, sedangkan decrement berarti mengurangi nilai variable dengan satu. Operator ini nantinya banyak digunakan dalam teknik perulangan. Pra-increment ataupun pra-decrement memiliki arti bahwa suatu variable akan ditambahkan satu ataupun dikurangkan satu sebelum dipakai pada statement berikutnya. Sedangkan post-increment ataupun post decrement memiliki arti bahwa suatu variable akan dipakai terlebih dahulu baru ditambahkan satu ataupun dikurangkan satu.

Operator	Nama	Keterangan
`var++`	Post increment	Nilai yang ada di sebuah variabel digunakan dahulu, kemudian baru ditambah dengan satu
`++var`	Pre increment	Nilai yang ada di sebuah variabel ditambah satu, kemudian baru digunakan
`var--`	Post decrement	Nilai yang ada di sebuah variabel digunakan dahulu, baru dikurang satu
`--var`	Pre decrement	Nilai yang ada di sebuah variabel dikurang satu, baru kemudian digunakan

Perulangan dengan for

Format perulangan dengan perintah for adalah sebagai berikut;

```
for(ekspresi1; expresi2; ekspresi3)Pernyataan;
```

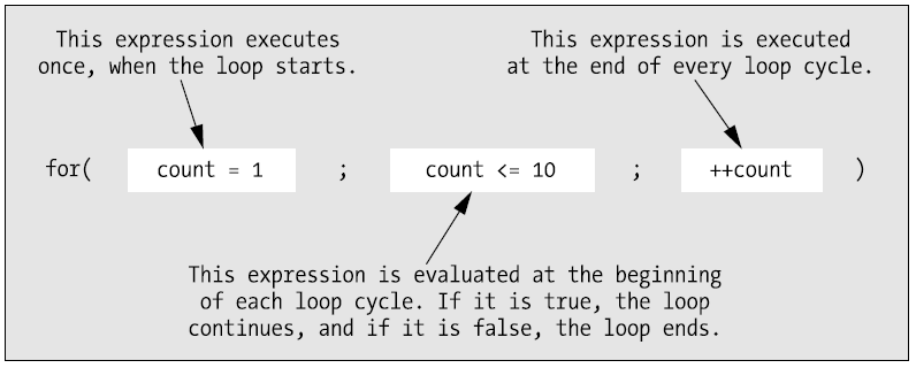
atau

```
for(ekspresi1; expresi2; ekspresi3)
{
   Pernyataan1;
   Pernyataan2;
   Pernyataan3;
}
```

Contoh: penggunaan increment pada perulangan.

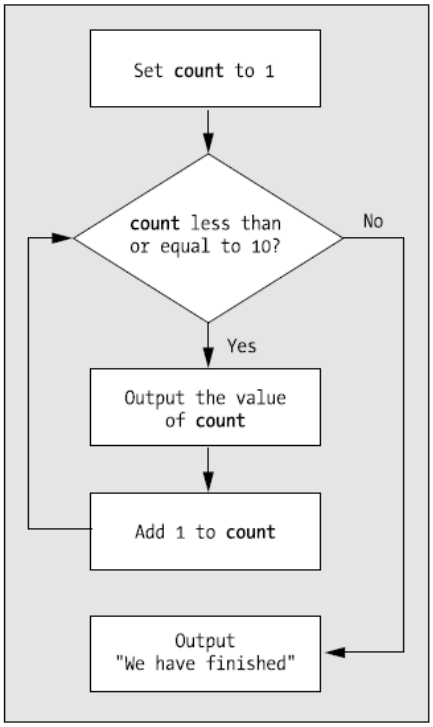
```
#include <stdio.h>
int main(void)
{
   int count;
   for(count = 1 ; count <= 10 ; ++count)
   {
       printf("\n%d", count);
   }
   printf("\n Selesai..\n");
   return 0;
}
```

Perhatikan penjelasan gambar dibawah ini;



Gambar .Expresi perulangan menggunakan for.

Adapun flow chart dari logika program diatas,



Gambar. Flow Chart

Contoh: penggunaan decrement pada perulangan.

```
#include <stdio.h>
int main(void)
{
   int count;
   for(count = 10; count >0 ; count--)
   {
       printf("\n%d", count);
   }
   printf("\n Selesai..\n");
   return 0;
}
```

Perulangan for tanpa parameter

Perulangan for tanpa parameter ataupun kondisi diperlihatkan,

```
for(  ;  ;  )
{

}
```

berarti perulangan for terjadi berkelanjutan atau kontinu.

Statement break didalam perulangan

Jika terdapat **break** pada perulangan maka perulangan di hentikan dan lanjut ke statement setelah perulangan,

Contoh:

```
#include <stdio.h>
int main(void)
{
```

```
    int count;
    for(count = 10; count >0 ; count--)
    {
        printf("\n%d", count);
        if(count==5) break;
    }
    printf("\n Selesai..\n");
    return 0;
 }
```

Perulangan while

Perulangan menggunakan while merupakan mekanisme perulangan yang membolehkan untuk melanjutkan perulangan tersebut apabila ekspresi logikanya bernilai benar.

```
while(expresi)        // jika ekspresi bernilai benar maka statement yang
{                     // berada di dalam perulangan while akan di eksekusi.
      Statement 1;    // jika terjadi perubahan nilai expresi menjadi salah
      Statement 2;    // maka perulangan tidak dilanjutkan.
}
```

Contoh:

```
 #include <stdio.h>
 int main(void)
 {
       int counter =0;
       int Angka=0;
       printf("Pilihlah satu angka diantara 0 s/d 30 \n”);
       printf(“Anda diberi 5 kali kesempatan \n”);

       while(counter<=5)
       {
            scanf(" %d", &Angka);            /* Read the input */
            while((angka%4)==1)
            {
                printf("Anda memasukkan nilai yang benar”);
               break;
            }
            if(counter<=5)
            {
             counter++;
             printf(“counter = %d”,counter);
            }
        }
       return 0;
}
```

Continuous while

Continuous while adalah perulangan yang berlangsung berkelanjutan.

```
#include <stdio.h>
int main(void)
{
    int counter =0;
    int Angka=0;
    printf("Ketiklah angka diantara 0 s/d 30 \n");
    printf("Anda diberi 5 kali kesempatan \n");

    while(1)
    {
        scanf(" %d", &Angka);            /* Read the input */
        if(counter<5)
        {
            counter++;
            printf("Anda sudah mengentrikan nilai sebanyak %d kali\n",counter);
        }else{
            break;
        }
     }

      return 0;

}
```

5.3. Lembar Kerja

Buatlah program untuk mengurutkan nilai, dari nilai terkecil ke nilai terbesar, sesuai dengan hasil berikut;

```
Nilai Mahasiswa:
76
73
80
68
70
Urutan Nilai dari yang terkecil:
68
70
73
76
80
```

5.4. Tugas

1. Buatlah program dengan memanfaatkan perulangan untuk menampilkan angka dengan kelipatan 3, untuk angka diantara 0 s/d 60;
2. Berikanlah penjelasan mengenai perbedaan perulangan for dengan perulangan while.

MODUL 6: PERULANGAN (2)

(LOOP)

6.1. Tujuan Praktikum

- Memahami pengertian suatu perulangan dan bagaimana cara menggunakannya.
- Mengerti dan dapat membedakan penggunaann perulangan for bersarang dan do-while.
- Mampu mengaplikasikan for bersarang dan do-while dengan kondisi tertentu.

6.2. Teori Dasar

Seperti yang sudah kita pelajari pada modul 5 bahwa perulangan for memiliki initial value atau nilai awal, ekspresi, dan loop cycle. Sehingga ada kemungkinan kita menempatkan perulangan di dalam suatu perulangan baik for maupun while. Perhatikan listing dibawah ini;

```
for(i=0; i<4; i++)
{
      for(j=0; j<3; j++)
      {
            Statement_1;
      }
      Statement_2;
}
```

Ataupun,

```
while(i<4)
{
      while(j<3)
      {
            Statement_1;
      }
      Statement_2;
}
```

do-while

Perulangan do-while dan while memiliki perbedaan dalam mengevaluasi ekspresi. While akan mengevaluasi ekspresi pada awal perulangan, sedangkan do-while mengevaluasi ekspresi pada akhir perulangan.

```
int number = 4;
do
{
   printf("\nNumber = %d", number);
   number++;
}
while(number < 4);
```

6.3. Lembar Kerja

1. Pahamilah perbedaan penggunaan while dan do-while dibawah ini, dan berilah pendapatmu.

`int number = 0;`
`do`
`{`
`  printf("\nNumber = %d", number);`
`  number++;`
`}`
`while(number < 4);`
	`int number = 0;`
	`while(number < 4)`
	`{`
	`  printf("\nNumber = %d", number);`
	`  number++;`
	`}`

2. Gantilah nilai number dengan angka 5, eksekusi program dan berikan pendapatmu.

6.4. Tugas

1. Carilah contoh program dengan mengaplikasikan while dan do-while.
2. Buktikanlah program itu sukses dan perlihatkan hasil nya.

MODUL 7: ARRAY

7.1. Tujuan Praktikum

- Memahami pengerian Array
- Mengerti penggunaan aray pada suatu aplikasi program
- Mengerti dan memahami array multidimensi

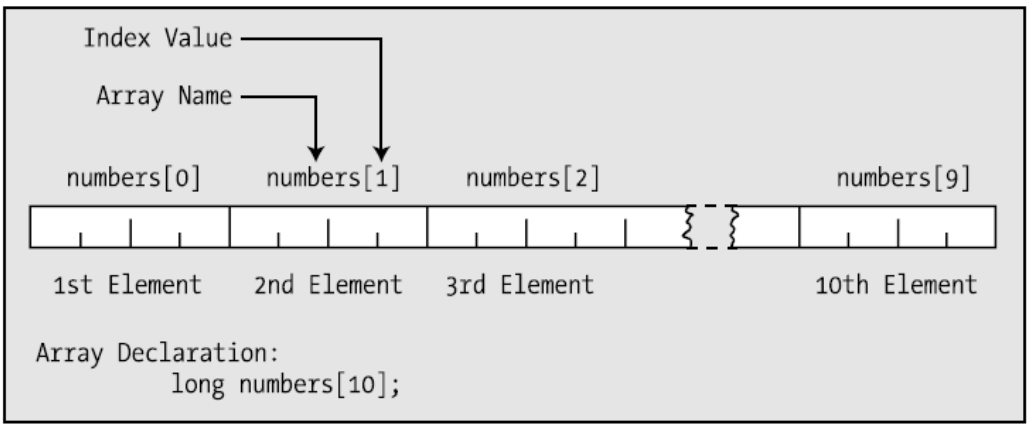
7.2. Teori Dasar

Array merupakan suatu variabel dengan tipe yang sama untuk seluruh elemennya. Misalnya;

```
int Number[10];
```

adalah sebuah variabel Number dengan tipe data integer yang memiliki 10 element. Jika variabel array memiliki data element yang tetap dapat di inisialisasikan sebagai berikut;

```
int Number[10]={1,3,2,4,5,7,6,9,8,0};
```



Gambar. Element pada variabel Array.

Perhatikanlah contoh dibawah ini;

```
#include <stdio.h>
int main(void)
{
    int numbers[10];
    int count = 10;
    long sum = 0L;
    float average = 0.0f;
    printf("\n Masukkanlah 10 Angka:\n");

    for(int i = 0; i < count; i ++)
    {
       printf("%2d> ",i+1);
       scanf("%d", &numbers[i]);             /* Read a number */

       sum += numbers[i];                    /* Jumlahkan setiap elemen */
    }
    average = (float)sum/count;              /* Hitung rata-rata */
    printf("\n Rata-rata dari sepuluh Angka yang dimasukkan: %f\n", average);
    return 0;
}
```

```
Hasil:
   Masukkanlah 10 Angka:
   1> 450
   2> 765
   3> 562
   4> 700
   5> 598
   6> 635
   7> 501
   8> 720
   9> 689
   10> 527
   Rata-rata dari sepuluh Angka yang dimasukkan: 614.700000
```

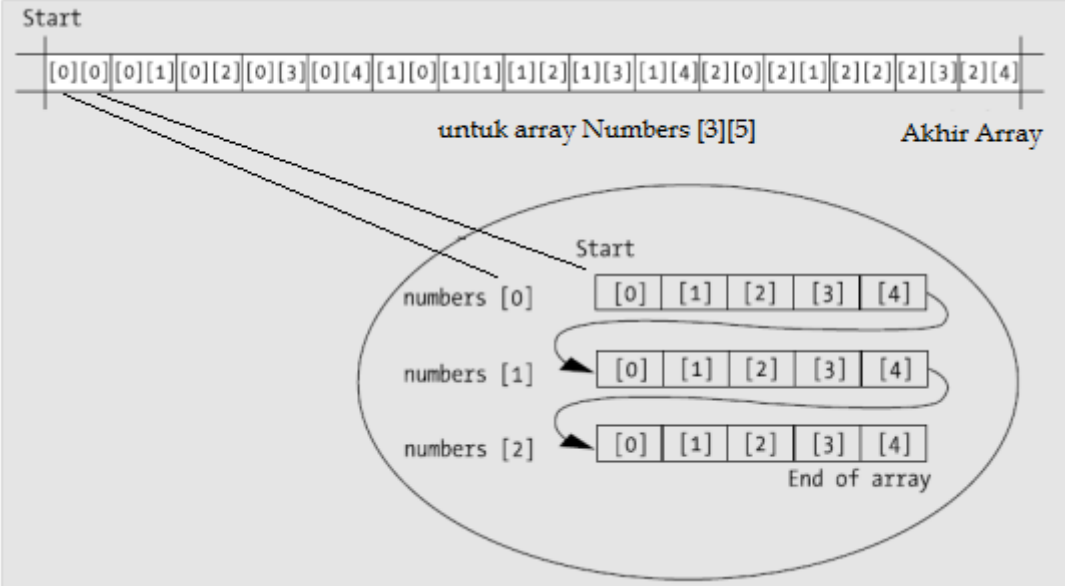
Kombinasi perulangan, kondisi dan array juga dapat mempermudah mencari nilai terbesar ataupun terkecil dari sebaran beberapa nilai. Perhatikanlah contoh dibawah ini:

```
#include <stdio.h>
int main(void)
{
    int numbers[10];
    int count = 10;
    int NilaiTerbesar;
    int NilaiTerkecil;
    printf("\n Masukkanlah 10 Angka:\n");

    for(int i = 0; i < count; i ++)
    {
       printf("%2d> ",i+1);
       scanf("%d", &numbers[i]);
    }

    /* mencari nilai terbesar */

    NilaiTerbesar = number[0];

    for(int i = 1; i < count; i ++)
    {
       if(NilaiTerbesar >= number[i]) NilaiTerbesar = NilaiTerbesar;
       else NilaiTerbesar = number[i];
    }

    NilaiTerkecil = number[0];

    for(int i = 1; i < count; i ++)
    {
       if(NilaiTerkecil <= number[i]) NilaiTerkecil = NilaiTerkecil;
       else NilaiTerkecil = number[i];
    }

    printf("\n Nilai terbesar adalah: %d\n", NilaiTerbesar);
    printf("\n Nilai terkecil adalah: %d\n", NilaiTerkecil);

    return 0;
}
```

Array Multidimensi

Sebelumnya kita mempelajari variabel bertipe aray dengan hanya menggunakan 1 tanda “[ ]”. Array multi dimensi akan menggunakan lebih dari satu “[ ]”. Untuk menyatakan elemen yang ada pada variabel tersebut.

Contoh: *float Numbers[3][5];*



Contoh inisialisasi array multidimensi seperti yang ditampilkan dibawah ini;

```
int numbers[3][4] = {
                      { 10, 20, 30, 40 },
                      { 15, 25, 35, 45 },
                      { 47, 48, 49, 50 }
                    };
```

```
int numbers[2][3][4] = {
                          {
                             { 10, 20, 30, 40 },
                             { 15, 25, 35, 45 },
                             { 47, 48, 49, 50 }
                          },
                          {
                             { 10, 20, 30, 40 },
                             { 15, 25, 35, 45 },
                             { 47, 48, 49, 50 }
                          }
                        };
```

Lihat bagaimana menyelesaikan penjumlahan dua matriks dibawah ini,

```
#include <stdio.h>

#define ROW  2
#define COL  3

int main(void)
{
    int Matriks_A[ROW][COL]={
                            {4,2,3},
                            {5,7,6}
                           };
    int Matriks_B[ROW][COL]={
                            {1,8,9},
                            {3,5,4}
                           };
    int Matriks_C[ROW][COL];

    printf("Matriks_C adalah : \n");

    for(int i = 0; i < ROW; i++)
    {
       for(int j = 0; j < COL; j++)
       {
          Matriks_C[i][j] = Matriks_A[i][j]+ Matriks_B[i][j];
       }
       printf(" %d  %d  %d", Matriks_C[i][0], Matriks_C[i][1], Matriks_C[i][2] \n);
    }
    return 0;
}
```

7.3. Lembar Kerja

1. Buatlah sebuah program yang dapat menentukan element keberapa dari angka terbesar yang ada pada nomor NIM kamu.
   Langkah 1: inisialisasi variabel, misal " int NIM[10] = {x,x,x,x,x,x,x,x,x,x};", angka sepuluh menyatakan jumlah elemen angka pada NIM kamu.
   Langkah 2: inisialisasi variabel penghitung dan tentukan nilainya sesuai dengan jumlah NIM kamu, misal " int Counter = 10; ".
   Langkah 3: Gunakanlah Perulangan dan kondisi untuk mendapatkan elemen ke berapa untuk angka terbesar pada NIM kamu.
2. Ketiklah listing Program dibawah ini, perhatikan mengapa tidak ada angka di pada "[ ]" sebagai jumlah elemen array? apakah spasi dihitung elemen array?

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

int main(void)
{
    char Nama[] = {"Thomas Alfa Edison"};
    int length;

    length = strlen(Nama);
    printf("length = %d \n"length);
    for(int i = 0; i<length; i++)
    {
       printf("%c",Nama[i]);
    }
    printf("\n");
    return 0;
}
```

7.4. Tugas

Buatlah program perkalian matriks dengan nilai-nilai yang ada pada MIM masing-masing. 5 angka pertama pada NIM merupakan baris pertama matriks, dan 5 angka kedua merupakan baris kedua matriks. Sehingga di dapat matriks 2 x 5. Kalikan lah matrik tersebut dengan matriks ;

$$\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

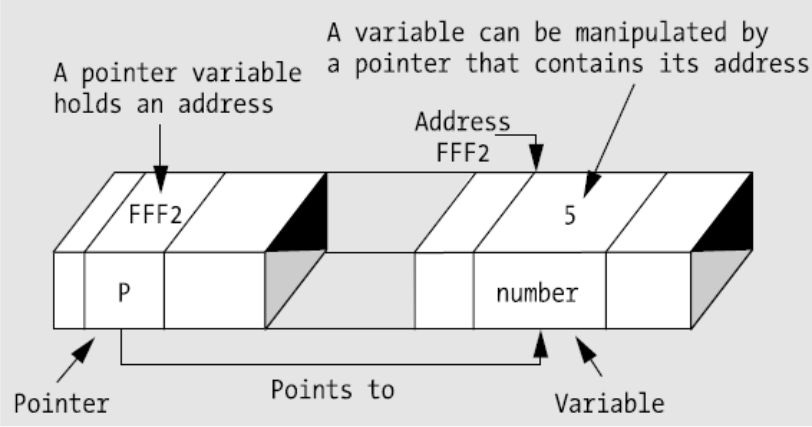
MODUL 8: POINTER

8.1. Tujuan Praktikum

- Mengerti apa yang dimaksud dengan pointer.
- Memahami penggunaan pointer.
- Memahami array suatu pointer.

8.2. Teori Dasar

Penggunaan pointer dalam bahasa pemrograman C sangatlah *powerful*, hanya saja dalam penggunaanya harus lah berhati-hati dan cermat. Karena besar kemungkinan kesalahan akan terjadi kalau tidak mengikuti aturan penggunaannya. Pointer sangat berkaitan dengan alokasi memori. Maksudnya, apabila pointer suatu variabel diinisialisasikan maka ini akan langsung merujuk ke alamat alokasi pada memori. Dapat disimpulkan bahwa Pointer adalah variabel yang menyimpan suatu alamat, dan alamat yang disimpan oleh pointer biasanya adalah suatu variabel lain.



Deklarasi Pointer

Deklarasi suatu pointer seperti dibawah ini;

int *pointer;

int *number;

untuk mengakses nilai suatu pointer haruslah berhati-hati, jangan ada yang terlupa, lihat contoh deklarasi dibawah ini,

```
int number = 15;
int *pointer = &number;
int result = 0;

result = *pointer + 5;
```

pada deklarasi diatas `*pointer` merujuk kepada alamat variabel `number` dimana alamat yang dituju itu bernilai 15. Sehingga `result` adalah 15 + 5 = 20.

Contoh 1:

```
#include <stdio.h>
int main(void)
{
    int value = 0;
    int *pvalue = NULL;
    pvalue = &value;                    /* Set pointer to refer to value */
    printf ("Input an integer: ");
    scanf(" %d", pvalue);               /* Read into value via the pointer */
    printf("\nYou entered %d\n", value);     /* Output the value entered */
    return 0;
}
```

Kemudian bagaimana jika mengaplikasikan array dan pointer, perhatikan contoh dibawah ini agar mudah untuk dipahami,

Contoh 2:

```
#include <stdio.h>
int main(void)
{
    char Nama[]={Thomas Alfa Edison};
    char *p = &Nama[0];
    printf("\n alamat dari element aray yang pertama: %p",p);

    p=&Nama[3];
    printf("\n karakter element ketiga dari Nama: %c",*p);

    p=p+2;
    printf("\n karakter element kelima dari Nama: %c",*p);

    return 0;
}
```

8.3. Lembar Kerja

1. Perhatikan contoh 2 diatas, carilah alamat (address) awal dan akhir untuk variabel "`Nama[ ]`", dan periksalah berapa Byte yang digunakan oleh variabel tersebut.
2. Cobalah simpulkan maksud statement:
   `p=&Nama[3];` dan, `p=p+2;`

8.4. Tugas

Buatlah suatu program yang menggunakan array dan pointer.

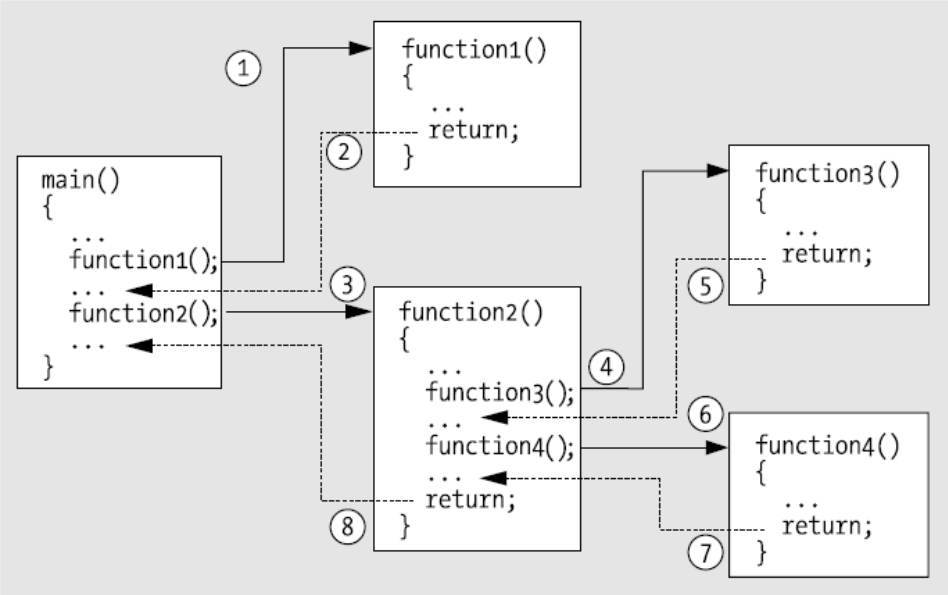
MODUL 9: FUNGSI

9.1. Tujuan Praktikum

- Mengenal dan Mengerti penggunaan Fungsi pada Suatu Program.
- Bagaimana Melewatkan data pada suatu Fungsi.
- Bagaimana Mengembalikan hasil dari suatu Fungsi.

9.2. Teori Dasar

Dalam bahasa pemrograman C penggunaan fungsi sudah pasti terpakai dan selalu digunakan. Fungsi `int main(void)` merupakan fungsi utama yang menjadi awal eksekusi program. Ketika kita menggunakan librari seperti `stdio.h` seperti `printf("...");` sebenarnya kita sudah memanggil suatu fungsi yang di deklarasikan pada librari tersebut. Perhatikan gambar dibawah ini,



Fungsi yang memiliki tipe fungsi selalu harus mengembalikan suatu nilai, sedangkan fungsi tanpa tipe fungsi atau kosong "void" tidak mengembalikan suatu nilai. Bagaimanakah menginisialisasikan suatu fungsi?

```
int Penjumlahan(void);
       // int merupakan tipe fungsi dengan tipe
       // pengembalian nilai. void berarti kosong.
int Penjumlahan(int A, int B);
       // variabel A dan B merupakan variabel internal fungsi.
```

dan bahgaimana fungsi itu kita deklarasikan,

```
int Penjumlahan(int A, int B)
{
    int hasil;
    hasil = A + B;
    Return hasil;
}
```

Sehingga pada fungsi utama akan di pangil fungsi yang sudah dideklarasikan diatas,

```
void main(void)
{
    int RataRata;
    int Total;
    int Matematika = 65;
    int Fisika = 74;
    Total = Penjumlahan(Matematika,Fisika); //pemanggilan fungsi
    RataRata = Total/2;
}
```

Contoh:

```
#include <stdio.h>

//inisialisasi fungsi
float average(float x, float y);

// Fungsi main
int main(void)
{
    float value1 = 0.0F;
    float value2 = 0.0F;
    float value3 = 0.0F;
    printf("Enter two floating-point values separated by blanks: ");
    scanf("%f %f", &value1, &value2);
    value3 = average(value1, value2);
    printf("\nThe average is: %f\n", value3);
    return 0;
}

/* Definisi fungsi untuk menghitung rata-rata */
float average(float x, float y)
{
    return (x + y)/2.0f;
}
```

Akan di dapat hasil:

```
Enter two floating-point value separate by blank: 2.34 4.567
The average is: 3.453500
```

9.3. Lembar Kerja

1. Periksa program dibawah ini, dan berilah komentarmu.

```
#include <stdio.h>

//inisialisasi fungsi
int average(int x, int y);

// Fungsi main
int main(void)
{
    int value1 = 0.0F;
    int value2 = 0.0F;
    int value3 = 0.0F;
    printf("Enter two floating-point values separated by blanks: ");
    scanf("%d %d", &value1, &value2);
    value3 = average(value1, value2);
    printf("\nThe average is: %d\n", value3);
    return 0;
}
```

```
/* Definisi fungsi untuk menghitung rata-rata */
int average(int x, int y)
{
   return (x + y)/2;
}
```

2. Buatlah suatu variabel array dengan 20 nilai element yang acak, kemudian buatlah sebuah program untuk mencari rata-rata, nilai tertinggi, dan nilai terrendah. Gunakan fungsi, array, perulangan, dan if kondisi untuk mempermudah atau menyederhanakan program.

9.4. Tugas

1. Hitunglah Rata-rata untuk bilangan dengan kelipatan 3 diantara 0-50 dengan menggunakan Fungsi dan Array.

MODUL 10: STRUKTUR DATA

10.1. Tujuan Praktikum

- Memahami pengertian Struktur Data dalam pemrograman C
- Mengerti cara mendeklarasi dan mendefinisikan Struktur Data
- Mengerti penggunaan struktur dan pointer berstruktur.
- Mengerti menggunakan pointer sebagai anggota struktur.

10.2. Teori Dasar

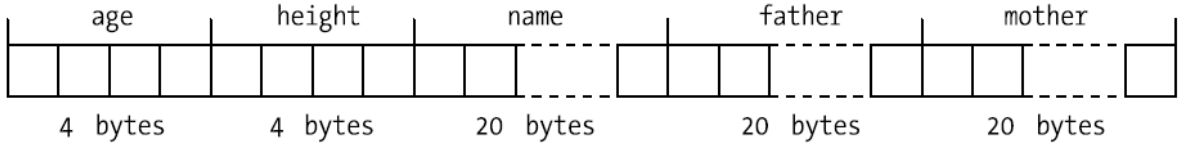
Struktur Data memberikan kemudahan untuk mendefinisikan beberapa variabel dengan tipe yang bervariasi. Struktur data dideklarasikan dengan menggunakan kata struct, seperti contoh dibawah ini;

```
Struct horse
{
        int age;
        int height;
}Silver;
```

Contoh diatas mendeklarasikan struktur dengan nama *horse*. Struktur ini memiliki anggota struktur yaitu *age*, dan *height*. Sedangkan *Silver* merupakan variabel dari tipe struktur. Sedangkan untuk contoh deklarasi yang lengkap adalah,

```
Struct horse
{
    int age;
    int height;
    char name[20];
    int father[20];
    int mother[20];
}Dobin = { 24, 17, "Dobbin","Trigger","Flossie" };
```

Jika dilihat dari peta memorinya maka Struktur dari Variabel Dobbin adalah,



Dapat disimpulkan bahwa struktur dapat mengelompokkan variabel dengan tipe yang berbeda

Tipe Struktur dan Variabel Struktur juga dapat didefinisikan secara terpisah. Pertama; definisikan Tipe Strukturnya dengan anggota struktur , dan kedua; definisikan variabel dari tipe tersebut. Perhatikan contoh dibawah,

```
struct horse
{
     int age;
     int height;
     char name[20];
     char father[20];
     char mother[20];
};

struct horse Dobbin = {
                     24, 17, “Dobbin”, “Trigger”, “Flossie”
                       };
```

Setelah Struktur didefinisikan barulah pemanggilan anggota struktur dapat dilakukan. Pemanggilan anggota struktur akan membawa variabel strukturnya, contohnya;

```
Dobbin.age = 12
```

Contoh 1:

```
#include <stdio.h>

int main(void)
{
  /* Structure declaration */
  struct horse
  {
    int age;
    int height;
    char name[20];
    char father[20];
    char mother[20];
  };

  struct horse My_first_horse;

  printf("Enter the name of the horse: " );
  scanf("%s", My_first_horse.name );
  printf("How old is %s? ", My_first_horse.name );
  scanf("%d", &My_first_horse.age );
  printf("How high is %s ( in hands )? ", My_first_horse.name );
  scanf("%d", &My_first_horse.height );
  printf("Who is %s's father? ", My_first_horse.name );
  scanf("%s", My_first_horse.father );
  printf("Who is %s's mother? ", My_first_horse.name );
  scanf("%s", My_first_horse.mother );
  printf("\n%s is %d years old, %d hands high,",
  My_first_horse.name, My_first_horse.age, My_first_horse.height);
  printf(" and has %s and %s as parents.\n", My_first_horse.father,
  My_first_horse.mother );
  return 0;
}
```

Sehingga hasil eksekusi terlihat sebagai berikut;

```
Enter the name of the horse: Neddy
How old is Neddy? 12
How high is Neddy (in hands)? 14
Who is Neddy’s father? Bertie
Who is Neddy’s mother? Nelie

Neddy is 12 years old, 14 hands high, and has Bertie and Nellie as
parents.
```

Struktur dengan Variabel Array

Struktur dengan variabel array dapat memberikan kemudahan bagi pengguna untuk pengolahan daata dengan bentuk yang serupa. Misalnya, ada 50 atau 100 kuda yang ingin didata. Sehingga dengan metoda ini akan lebih sederhana.

Contoh 2:

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{
    struct horse /* Structure declaration */
    {
      int age;
      int height;
      char name[20];
      char father[20];
      char mother[20];
    };
    struct horse My_horses[50]; /* Structure array declaration */
    int hcount = 0; /* Count of the number of horses */
    char test = '\0'; /* Test value for ending */
    for(hcount = 0; hcount<50 ; hcount++ )
    {
      printf("\nDo you want to enter details of a%s horse (Y or N)? ",
      hcount?"nother " : "" );
      scanf(" %c", &test );
      if(tolower(test) == 'n')
      break;
      printf("\nEnter the name of the horse: " );
      scanf("%s", My_horses[hcount].name ); /* Read the horse's name */
      printf("\nHow old is %s? ", My_horses[hcount].name );
      scanf("%d", &My_horses[hcount].age ); /* Read the horse's age */
      printf("\nHow high is %s ( in hands )? ", My_horses[hcount].name );
      /* Read the horse's height*/
      scanf("%d", &My_horses[hcount].height );
      printf("\nWho is %s's father? ", My_horses[hcount].name );
      /* Get the father's name */
      scanf("%s", My_horses[hcount].father );
      printf("\nWho is %s's mother? ", My_horses[hcount].name );
```

```
      /* Get the mother's name */
      scanf("%s", My_horses[hcount].mother );
   }
   /* Now tell them what we know. */
   for(int i = 0 ; i<hcount ; i++ )
   {
      printf("\n\n%s is %d years old, %d hands high,",
      My_horses[i].name, My_horses[i].age, My_horses[i].height);
      printf(" and has %s and %s as parents.", My_horses[i].father,
      My_horses[i].mother );
   }
   return 0;
}
```

10.3. Lembar Kerja

1. Ketiklah program seperti contoh 1 diatas, kemudian gantilah baris seperti ini

`struct horse`
`{`
`  int age;`
`  int height;`
`  char name[20];`
`  char father[20];`
`  char mother[20];`
`};`
 	
`struct horse My_first_horse;`
	`struct`
	`{`
	`  int age;`
	`  int height;`
	`  char name[20];`
	`  char father[20];`
	`  char mother[20];`
	`} My_first_horse;`

   Berikanlah penjelasan, apakah variabel struktur dapat di konfigurasi lebih dari satu?

2. Perhatikan! Lengkapilah listing program dibawah jika kita ingin menyimpan data 15 orang Mahasiswa dengan konfigurasi struktur debawah ini.

```
#include <stdio.h>

int main(void)
{
   /* Structure declaration */
   struct Student
   {
   int age;
   int height;
   };
   ...
   ...
   ...
   return 0;
}
```

10.4. Tugas

Buatlah program untuk mendata identitas mahasiswa kelas ini dengan menggunakan struktur data. Anggota struktur meliputi; nama, umur, nama ayah, nama ibu..

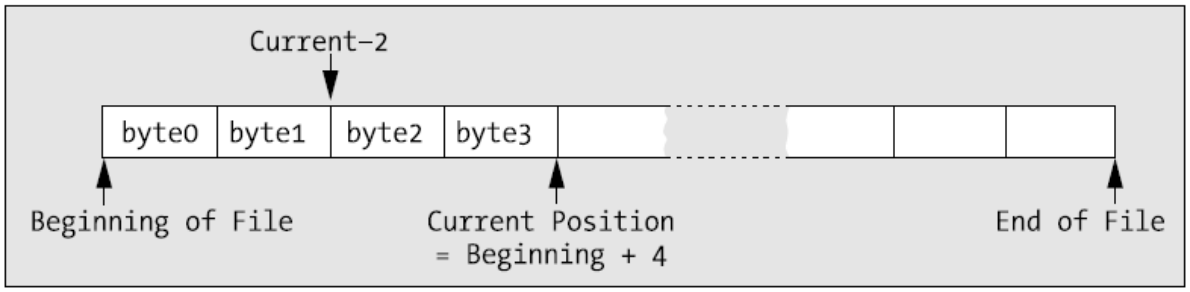
MODUL 11: MENGAKSES FILE

11.1. Tujuan Praktikum

- Memahami pengolahan file dalam pemrograman C
- Mengerti cara penulisan File pada suatu folder.
- Mengerti pembacaan file dari suatu Folder
- Mampu mengaplikasikan cara mengakses File

11.2. Teori Dasar

Mengakses File merupakan salah satu aplikasi pemrograman C yang erat kaitannya dengan memori alokasi. Disini kita akan menyimpan data dalam bentuk file dan ditempatkan pada suatu folder, ataupun kita juga akan membaca data yang ada dalam alamat folder tertentu agar ditampilkan pada layar komputer. Jika diperhatikan Gambar 11.1 maka terlihat alokasi memori untuk suatu file.



Gambar. 11.1 Struktur File pada memory

Dalam bahasa pemrograman C sudah tersedia library untuk mengakses data **stream**, dimana stream merupakan perwakilan abstrak dari sumber luar ataupun data tujuan. Keyboard, command line pada tampilan, file pada disk merupakan contoh dari stream.

Membuka File / Opening a File

Membuka file merupakan proses yang mengaitkan file eksternal tertentu dengan internal variabel pointer suatu file. Untuk membuka File kita perlu memanggil fungsi standart library `fopen()`, yang mengembalikan file pointer untuk eksternal file tertentu. Dimana `fopen()` di definisikan pada `<stdio.h>`, dan di deklarasikan sebagai berikut;

```
FILE *fopen(char *name, char *mode);
```

Argumen pertama pada fungsi adalah pointer sebuah string yang merupakan file eksternal untuk di proses. `nama` dapat ditulis sebagai argument, menggunakan array, atau variabel pointer to char yang berisikan alamat dari string karakter yang mendefinisikan file.

Argumen ke dua pada fungsi `fopen()` adalah character string yang disebut file mode, yang menyatakan proses apa yang akan dilakukan pada file tersebut. Apakah akan menulis pada file atau membaca suatu file.

Mode	Deskripsi
"w"	**Buka file text untuk menulis**
"r"	**Buka file text untuk membaca**

Sebagai contoh mendeklarasikannya;

```
/* buka file myfile.txt untuk ditulis */
FILE *pfile = fopen("myfile.txt", "w");

/* buka file myfile.txt untuk dibaca*/
pfile = fopen("myfile.txt", "r");
```

Menutup File

Untuk menyelesaikan suatu proses dalam mengakses file, maka file tersebut perlu di tutup. Sebagai comand dapat memanggil library `fclose()`,dengan deklarasi;

```
fclose(pfile);
```

Menulis pada File Text

Perintah menulis pada file text adalah fungsi `fputc()` yang menuliskan karakter tunggal pada file text,

```
int fputc(int c, FILE *pfile);
```

Fungsi `fputc()` merupakan perintah menuliskansuatu karakter yang erdapat pada argumen pertama, ke file yang sudah didefinisikan pada argument kedua. File pada argument kedua merupakam pointer. Pada fungsi fputc(), jika penulisan yang dilakukan sukses maka fungsi ini akan mengembaliksm (return)sebuah karakter yang menytakan kalausudah ditulis.

Membaca data dari File Text

Perintah membaca data dari file text adalah fungsi fgetc(), fungsi ini adalah kebalikan dari fputc dan membaca karakter dari File text.

```
mchar = fgetc(pfile);
```

Contoh:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
const int LENGTH = 80;         /* Maks. panjang karakter */
int main(void)
{
  char mystr[LENGTH];          /* Input string */
  int lstr = 0;                /* Panjang input string */
  int mychar = 0;              /* karakter output */
  FILE *pfile = NULL;          /* File pointer */
  char *filename = "C:\\myfile.txt";

  printf("\nMasukkan  string kurang dari 80 karakter:\n");
  fgets(mystr, LENGTH, stdin);  /* baca string */

  /* Buatlah file baru yang dapat kita tulis */
  if(!(pfile = fopen(filename, "w")))
  {
       printf("Error opening %s for writing.", filename);
       exit(1);
  }
  lstr = strlen(mystr);
  for(int i = lstr-1 ; i >= 0 ; i--)
  fputc(mystr[i], pfile); /* Tulis String ke file */
  fclose(pfile); /* Close the file */
  return 0;
}
```

11.3. Lembar Kerja

Mengacu pada contoh diatas buatlah suatu program untuk menuliskan Nama,Nim Dan Jurusan kamu kedalam file Text.

11.4. Tugas

Buatlah program untuk menulis file TXT yang berbentuk kartu hasil Studi