MODUL PRAKTIKUM
DASAR PEMROGRAMAN

LABORATORIUM SISTEM PENGATURAN & KOMPUTER

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SUMATERA UTARA

Bab 1

Pengenalan Linux dan Aplikasi Dasar

1.1 Tujuan

- ✔ Mengenal perintah dasar Linux
- ✔ Mengenal shell sebagai salah satu interface
- ✔ Mengenal editor teks dan kompiler pada linux

1.2 Teori

Linux adalah suatu sistem operasi turunan UNIX, menggunakan Kernel Linux dan implementasi dari standar POSIX. Linux dapat ber-interoperated dengan sistem operasi lain seperti Microsoft Windows, Apple Mac, dan BSD Unix. Linux merupakan contoh yang terkenal dari pengembangan *Open Source Software*. Sekarang Linux banyak mendapatkan dukungan dari berbagai vendor seperti, Hewlett Packard, Novell, Red Hat, IBM, dan banyak lagi.

1.2.1 Sejarah Linux

Pengerjaan Kernel Linux dimulai pada tahun 1991 oleh seorang mahasiswa *University of Helsinki* bernama **Linus Torvalds**. Linus membuat kernel ini untuk menggantikan **Minix** yang tidak "*bebas"* buatan **Andrew S. Tanenbaum**. Atas kerjasamanya dengan **Richard M. Stallman**, pendiri GNU (Gnu Not Unix) Projects dan *The Free Software Foundation*, Linus diijinkan untuk menggunakan komponen yang dikembangkan GNU, seperti GCC (GNU C Compiler). Kernel Linux dikembangkan dengan bahasa C..

Gambar 1: Torvalds dan Stallman

1.2.2 Shell

Shell adalah sebuah program yang menyediakan interface pengguna dengan kernel. Shell terdiri atas dua jenis yaitu *Command Line Interface (CLI)* dan *Graphic User Interface (GUI)*. Terdapat beberapa CLI-Shell pada Unix, antara lain:

- ➔ Bourne Shell (bsh)
- ➔ Bourne Again Shell (bash)
- ➔ Korn Shell (ksh)
- ➔ C shell (csh)

CLI-Shell yang paling banyak digunakan sebagai *default* pada mayoritas distribusi Linux adalah *bash*. Dalam studi kasus kita juga menggunakan *bash* untuk CLI.

Beberapa perintah dasar yang bisa kita gunakan pada *bash*:

Perintah	Fungsi	Keterangan
`mkdir`	Untuk membuat sebuah direktori	Mirip dengan perintah `md` pada DOS Syntax: `$ mkdir /direktori`
`cp`	Untuk mengkopi file	Mirip dengan perintah `copy` pada DOS Syntax: `$ cp [option] file /dir/tujuan`
`ls`	Menampilkan isi suatu direktori beserta atribut file-nya	Mirip dengan perintah dir pada DOS Syntax: `$ ls [option] /direktori`
`cd`	Memindahkan direktori aktif	Mirip dengan perintah `cd` pada DOS Syntax: `$ cd /dir/tujuan`
`rm`	Menghapus file	Mirip dengan perintah `del` pada DOS Syntax: `$ rm file`
`rmdir`	Menghapus direktori	Mirip dengan perintah `rd` pada DOS Syntax: `$ rmdir /direktori`
`mv`	Memindahkan file atau direktori. Bisa juga untuk mengganti	Mirip dengan perintah `move` dan `ren` pada DOS

	nama file atau direktori	Syntax: `$ mv file/folder tujuan/nama baru`
`man`	Menampilan manual dari perintah yang ada pada linux	Mirip dengan perintah `help` pada DOS Syntax: `$ man nama_perintah`

Catatan

Perintah yang disajikan hanya beberapa dari sebagian banyak perintah. Cobalah untuk mengeksplorasi perintah dan opsi-opsi yang disediakan.

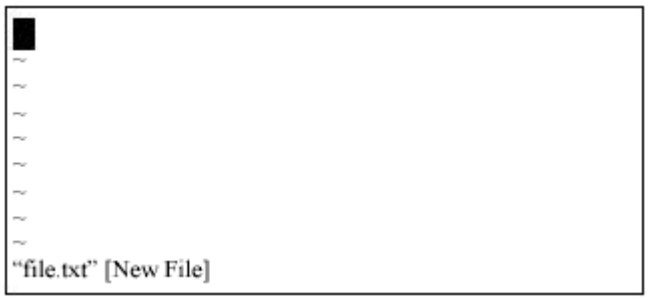
1.2.3 Text Editor (*Vim*)

Terdapat banyak text editor yang ada pada Linux, baik yang berbasis CLI ataupun GUI. Beberapa diantaranya adalah `Vi`, `joe`, `pico`, `ee`, `nano`, `mc` yang berbasis CLI dan `Kwrite`, `Kate`, `Gedit` yang berbasis GUI. Pada praktikum, kita akan menggunakan text editor berbasis CLI yaitu *Vim*. *Vim* adalah akronim dari "Vi improved" atau peningkatan dari *Vi*, sedangkan *Vi* adalah akronim dari "Visual".

Penggunaan *Vim* text editor

Untuk memulai *Vim,* ketikkan perintah:

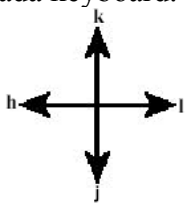
```
$ vim nama_file
```



Catatan

Jika nama file yang anda isikan sudah terdapat pada direktori aktif anda, maka editor akan menampilkan isi file tersebut dan anda bisa memulai untuk mengedit.

Untuk menggerakkan kursor pada Vim anda dapat menggunakan tombol **H, J, K** dan **L** atau anda juga dapat menggunakan tombol panah yang terdapat pada keyboard.



Untuk dapat mengetikkan teks anda harus berada pada modus insert. Tekan **i** atau **<Insert>** maka akan muncul indikator **--INSERT--** kemudian mulai ketikkan teks dan tekan **<Esc>** untuk kembali ke modus normal**.**

Beberapa perintah yang bisa digunakan antara lain:

Perintah	Keterangan
zz	Keluar dari editor dan simpan file yang di-edit
:q	Keluar dari editor
:w	Menyimpan file yang di-edit
:q!	Keluar dari editor tanpa menyimpan (override)
:wq	Keluar dari editor dan simpan file yang di-edit
:help atau **<F1>**	Menampilkan halaman bantuan
:help [subjek]	Menampilkan halaman bantuan dengan spesifik subjek
x	Menghapus karakter
X	Menghapus karakter disebelah kiri kursor
dd	Menghapus satu baris
dw	Menghapus 1 kata
d0	Menghapus dari awal teks sampai ke kursor
d$	Menghapus dari kursor sampai ke akhir teks
:set number	Memunculkan nomor baris
:set nonumber	Menghilangkan nomor baris

Catatan

Sebenarnya terdapat berbagai perintah pada editor *Vim*. Untuk mengeksplorasinya sendiri, silakan gunakan perintah `$ vimtutor` atau `www.vim.org`. **Jangan menghapal perintah yang ada pada *Vim*!!!**

1.2.4 Compiler (*gcc*)

GCC (Gnu C Compiler) pertama rilis dengan versi beta 0.9 pada 22 Maret 1987, kemudian rilis 1.0 pada 23 Mei 1987. GCC menjadi bagian penting dari perkembangan *opensource*. Keberadaan Linux juga sering dikaitkan dengan GCC. GCC bersifat “bebas”, karenanya anda dapat memperolehnya dengan mudah. GCC tersedia untuk berbagai platform.

Penggunaan *Gnu C Compiler*

Untuk meng-kompile program yang telah anda buat, gunakan perintah:

```
$ gcc [file.c] -o [file_eksekusi]
```

dan untuk run program yang telah anda buat:

```
$ ./file_eksekusi
```

Catatan

Tentang GCC dan kegunaannya lebih lanjut anda dapat menggunakan perintah `$ man gcc` atau http://gcc.gnu.org

1.3 Percobaan

- Buatlah sebuah direktori dan gunakan NIM anda sebagai namanya. Letak direktori anda pada `/home/praktikan`.
- Buatlah sebuah file `coba.c` dan simpan.
- Compile dan Run program yang telah anda buat tadi.

Bab 2

Bahasa C

2.1 Tujuan

- ✔ Mengenal dan memahami struktur bahasa pemrograman bahasa C.
- ✔ Mengetahui pemulisan program dalam bahasa C.

2.2 Teori

Hingga saat ini penggunaan bahasa C telah begitu luas di seluruh dunia. Berbagai perguruan tinggi di dunia menjadikan bahasa C sebagai salah satu mata kuliah wajib. Selain itu, banyak bahasa pemrograman populer seperti PHP dan Java menggunakan syntax dasar yang mirip dengan bahasa C. Sistem operasi seperti Microsoft Windows dan Linux juga dikembangkan dengan bahasa C.

2.2.1 Sejarah Bahasa C

Bahasa BCPL yang dikerjakan oleh Martin Richards pada tahun 1967 merupakan awal dari lahirnya bahasa C. Ken Thompson memulai pengembangan bahasa BCPL yaitu bahasa B pada tahun 1970. Perkembangan selanjutnya dari bahasa B dikembangkan menjadi bahasa C oleh Dennis Ritchie beberapa bulan berikutnya di Bell Telephone Laboratories Inc. (sekarang AT&T Bell Laboratories). Bahasa C pertama kali digunakan di komputer Digital Equipment Corporation PDP-11 yang menggunakan sistem operasi Unix.

Gambar 2: Thompson & Ritchie

2.2.2 Penulisan Program Bahasa C

Bentuk umum penulisan syntax bahasa C adalah:

```
1    main ()
2    {
3         //komentar
4         isi program
5    }
6
7    fungsi lain()
8    {
9         isi program    /*komentar*/
10   }
```

Pada baris pertama penulisan merupakan fungsi utama yang harus terdapat pada setiap pernulisan program. Baris kedua terdapat karakter "{" yang berarti "mulai". Baris berikutnya berisi syntax dari program yang hendak kita buat, termasuk Komentar, Deklarasi, Inisialisasi, dan Instruksi. Baris kelima terdapat karakter "}" yang berarti "akhir". Pada baris ke-tujuh terdapat fungsi lain yang dapat kita buat untuk membantu pengerjaan program yang akan kita buat. Biasanya programmer membuat fungsi lain diluar fungsi utama untuk menghemat penulisan program apabila dipergunakan berkali-kali.

2.3 Percobaan

- Tulislah program dibawah ini:

```
#include "stdio.h"

void main()
{
     //menampilkan Hello World ! Pada layar
     printf("Hello World!");
}
```

- Simpan dengan nama `hello.c.`
- Compile program tersebut.
- Run Program

Bab 3

Struktur Dasar Bahasa C

3.1 Tujuan

- ✔ Mengetahu tipe data yang ada pada bahasa C
- ✔ Mengetahui penggunaan konstanta dan variabel
- ✔ Mengetahui penulisan deklarasi dalam bahasa C
- ✔ Mengetahui jenis-jenis operator dan penggunaannya dalam bahasa C

3.2 Teori

Bahasa C menyediakan tipe-tipe data dasar yang cukup memadai. Walaupun dengan kebutuhan tertentu programmer dapat membuat tipe bentukan sendiri. Variabel-variabel dapat dibuat untuk tiap tipe data yang diperbolehkan. Dalam bahasa C, kita juga dapat membuat konstanta. Terdapat juga berbagai operator dalam bahasa C. Tipe data, Variabel, Konstanta, dan Operator akan kita bahas sebagai berikut.

3.2.1 Tipe Data

Tipe data merupakan bagian yang paling penting karena tipe data mempengaruhi seriap instruksi yang akan dilaksanakan oleh komputer. Misalnya saja 5 dibagi 2 bisa saja memberikan hasil yang berbeda tergantung pada tipe datanya. Jika 5 dan 2 bertipe *integer*, akan menghasilkan nilai 2. Namun jika kedianya bertipe *float* maka akan memberikan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien.

Dalam bahasa C terdapat lima tipe data dasar, yaitu:

No.	Tipe Data	Ukuran (byte)	Format	Keterangan
1	char	1	%c	Karakter / String
2	int	2	%i %d	Bilangan Bulat (*integer*)
3	float	4	%f	Bilangan pecahan (*float*)
4	double	8	%lf	Pecahan presisi ganda
5	void	0	-	Tidak bertipe

Beberapa tipe data dapat dimodifikasi dengan salah satu atau beberapa *modfier* berikut:

- ➔ signed
- ➔ unsigned
- ➔ short
- ➔ long

3.2.2 Variabel

Variabel adalah suatu pengenal (*identifier*) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variable dapat ditentukan sendiri oleh programmer dengan aturan sebagai berikut:

- ➔ Terdiri atas gabungan huruf, angka dan **"_"** (*underscore*).
- ➔ Bukan kata kunci yang terdapat pada bahasa C.
- ➔ Karakter pertama tidak dapat berupa angka.
- ➔ *Case-Sensitive* (huruf kapital dan huruf kecil berbeda)
- ➔ Tidak mengandung simbol dan spasi.
- ➔ Panjangnya bebas, namun hanya 32 karakter pertama yang terpakai.

Contoh Variabel yang BENAR
NIM _mhs Nama_siswa abc123

Contoh Variabel yang SALAH
%per penting! 123Abc $sine

3.2.3 Konstanta

Konstanta adalah suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta harus didefinisikan terlebih di awal program. Konstanta dapat bernilai *integer*, pecahan, karakter atau string. Selain itu, bahasa C juga menyediakan beberapa karakter khusus yang disebut *Escape Sequence*, antara lain:

Kode	Fungsi
\a	Untuk bunyi bell (alert)
\b	Mundur satu spasi (backspace)
\f	Ganti halaman (form feed)
\n	Ganti baris (new line)
\r	Ke kolom pertama, baris yang sama (carriage return)
\v	Tabulasi vertikal
\0	Nilai kosong (Null)
\'	Karakter **'**
\"	Karakter **"**
\?	Karakter **?**
\\	Karakter \
\N	Karakter Bilangan Oktal (dgn N = angka)
\xN	Karakter Bilangan Heksadesimal (dgn N = angka)

3.2.4 Operator

Di dalam bahasa C terdapat enam jenis operator yaitu:

➔ **Operator Penugasan**

Operator	Operasi
=	Inisialisasi

➔ **Operator Aritmatika**

Operator	Operasi
+	Penambahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulo (Sisa pembagian)

➔ **Operator Hubungan (Perbandingan)**

Operator	Operasi
<	Kurang dari
<=	Kurang dari sama dengan
>	Lebih dari
>=	Lebih dari sama dengan
==	Sama dengan
! =	Tidak sama dengan

➔ **Operator Logika**

Operator	Operasi
&&	Logika AND
||	Logika OR
!	Logika Not

➔ **Operator Bitwise**

Operator	Operasi
<<	Geser kiri
>>	Geser kanan
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
~	Bitwise NOT

➔ **Operator Unary**

Operator	Operasi	Letak (terhadap Operand)
++	Increment	Sebelum dan sesudah
--	Decrement	Sebelum dan sesudah
sizeof	Ukuran Operand	Sebelum
!	Unary NOT	Sebelum
~	Bitwise NOT	Sebelum
&	Alamat memori operand	Sebelum
*	Nilai dari pointer	Sebelum
-	Unary Minus	Sebelum

3.3 Percobaan

- Tulislah program berikut ini:

```
#include "stdio.h"
#define   A    5    //Konstanta Preposesor

void main()
{
     //Konstanta dgn fungsi const
     const double pi=3,1415926535897932384;

     //Deklarasi Variabel
     int x, nilai;
     float y;
     char z;
```

```
        double w;
        //Inisialisasi Variabel
        x=10;
        y=9.45;
        z='C';
        w=3.45E20;

        //Menampilkan variabel x
        printf ("Nilai dari x adalah  : %i\n, x);

        //Menampilkan variabel y
        printf ("Nilai dari y adalah  : %f\n, y);
        printf ("Nilai dari y adalah  : %g\n, y);
        printf ("Nilai dari y adalah  : %e\n, y);

        //Menampilkan variabel z
        printf ("Nilai dari z adalah  : %c\n, z);
        printf ("Nilai dari z adalah  : %u\n, z);

        //Penggunaan Operator
        printf ("Nilai dari a+x adalah : %1\n", a+x);
        printf ("Nilai dari a-x adalah : %i\n", a-x);
        printf ("Nilai dari a*x adalah : %i\n", a*x);
        printf ("Nilai dari y/a adalah : %f\n", y/a);
        printf ("Sisa 9 bagi 4 adalah  : %i\n", 9%4);

        /*Perbedaan ++ diletakkan sebelum dan sesudah*/
        nilai=++x;
        printf("x=%i\, nilai=%i\n", x, nilai);
        nilai=x++;
        printf("x=%i\, nilai=%i\n", x, nilai);
}
```

- Simpan dengan nama `tipe.c`.
- Compile program tersebut.
- Run program.

Bab 4

Input & Output

4.1 Tujuan

- ✔ Mengetahui fungsi – fungsi yang digunakan untuk proses memasukkan data.
- ✔ Mengetahui fungsi – fungsi yang digunakan untuk proses menampilkan data.

4.2 Teori

Input dan output pada program dimaksudkan agar program dan pengguna dapat saling berinteraksi. Sebagai contoh jika anda membuat sebuah program penghitung luas lingkaran, program akan menanyakan berapa jari-jari lingkaran. Setelah pengguna memasukkan nilai, kemudian komputer akan menampilkan luas lingkaran dengan jari-jari sesuai yang diisikan pengguna.

4.2.1 Input Data

Dalam bahasa C, proses untuk memasukkan sata bisa menggunakan beberapa fungsi pustaka yang telah tersedia. Beberapa fungsi pustaka yang bisa digunakan adalah:

➔ **`scanf()`**

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

➔ **`gets()`**

- ➲ Fungsi gets() digunakan untuk memasukkan data berupa tipe karakter dan tidak dapat digunakan untuk memasukkan data numerik
- ➲ Harus diakhiri dengan penekanan tombol user.
- ➲ Cursor secara otomatis akan berpindah baris.
- ➲ Tidak memerlukan penentu format.

Kode penentu format	Fungsi
%c	Membaca karakter
%s	Membaca string
%i %d	Membaca bilangan bulat
%f %e	Membaca bilangan pecahan
%o	Membaca bilangan oktal
%x	Membaca bilangan heksadesimal
%u	Membaca bilangan tak bertanda

➔ **`getchar()`**

- Fungsi `getchar()` digunakan untuk memasukkan data berupa tipe karakter dan tidak dapat digunakan untuk memasukkan data numerik.
- Harus diakhiri dengan penekanan tombol <Enter>.
- Karakter yang dimasukkan terlihat pada layar.
- Pergantian baris secara otomatis.

➔ **`getch() & getche()`**

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

4.2.2 Output Data

➔ **`printf()`**

- Fungsi printf() digunakan untuk menampilkan semua jenis data (numerik dan karakter).
- Tidak memberikan efek pergantian baris.

➔ **`puts()`**

- Fungsi `puts ()` digunakan untuk menampilkan data string.
- Memberikan efek pergantian baris secara otomatis.

➔ **`fprintf(), fputs() dan fputc()`**

- Fungsi `fprintf()` digunakan untuk mencetak semua jenis tipe data ke printer dan secara otomatis memberikan efek perpindahan baris.
- Fungsi `fputs()` digunakan untuk mencetak tipe data string ke printer.
- Fungsi `fputc()` digunakan untuk mencetak tipe data karakter ke printer.

4.3 Percobaan

- Tulislah program dibawah ini:

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

void main()
{
//Deklarasi Variabel
int umur;
char nama[20];
float nilai;
clrscr();

/*Memasukkan data*/
puts("Masukkan nama Anda : "); gets(nama);
puts("Masukkan umur Anda : "); scanf("%d", &umur);
puts("Masukkan nilai Anda : "); scanf("%f", &nilai);

/*Menampilkan data*/
printf("Nama anda : %s\n", nama);
printf("Umur anda : %s\n", nama);
printf("Anda memperoleh nilai %5.2f\n", nilai);

getch();
}
```

- Simpan dengan nama `io.c`.
- Compile program tersebut.
- Run program.

Bab 5

Pemilihan & Perulangan

5.1 Tujuan

- ✔ Mengetahui penggunaan struktur pemilihan.
- ✔ Mengetahui penggunaan struktur perulangan.

5.2 Teori

Struktur seleksi digunakan untuk memilih proses yang akan dijalankan bila suatu kondisi tertentu muncul. Seleksi memungkinkan program melakukan pengecekan terhadap kondisi yang terjadi, hingga hanya perintah – perintah yang sesuai dengan kondisi saja yang akan dieksekusi. Tentu akan sangat membuang waktu untuk menuliskan proses yang sama dalam sebuah program. Dengan menggunakan struktur pengulangan, kita dapat melakukan proses yang sama beberapa kali sesuai dengan keinginan kita.

5.2.1 Struktur Pemilihan

➔ **`if`**

Perintah `if` sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilasi benar, maka pernyataan yang berada dalam blok akan dieksekusi.

Bentuk Umum

```
if(kondisi)
   proses yang dilakukan bila kondisi terpenuhi
```

➔ **`if else`**

Dalam struktur ini minimal terdapat dua pernyataan. Jika kondisi yang diperiksa bernilai benar atau terpenuhi maka pernyataan pertama yang diproses dan jika kondisi yang diperiksa bernilai salah maka pernyataan yang kedua yang diproses.

Bentuk Umum

```
if(kondisi)
   proses yang dilakukan bila kondisi terpenuhi
else
   proses yang dilakukan bila kondisi tdk terpenuhi
```

atau

```
if(kondisi 1)
   proses yang dilakukan bila kondisi terpenuhi
else if(kondisi 2)
   proses yang dilakukan bila kondisi 1 tdk
   terpenuhi, tapi kondisi 2 terpenuhi
else if(kondisi 3)
   proses yang dilakukan bila kondisi 2 tdk
   terpenuhi, tapi kondisi 3 terpenuhi
else
   jika semua kondisi tidak terpenuhi
```

➔ **switch case**

Jika terdapat kemungkinan yang cukup banyak, dengan menggunakan struktur `if` kita akan menuliskan cukup banyak perintah `if`. Untuk itu, kita dapat menggunakan struktur `switch case`.

Bentuk Umum

```
switch(variabel atau ekspresi)
{
case nilai 1:
   proses yang dilakukan bila variabel atau ekspresi
   bernilai sama dgn nilai 1
   break;

case nilai 1:
   proses yang dilakukan bila variabel atau ekspresi
   bernilai sama dgn nilai 1
   break;

default:
   proses yang dilakukan bila variabel atau ekspresi
   tidak sama dgn nilai pd semua case
}
```

5.2.2 Struktur Perulangan

➔ **for**

Perintah `for` sangat cocok digunakan untuk perulangan karena jumlah pengulangan sudah diketahui. Begitu juga awal dan akhir pengulangan sudah diketahui. Bentuk pengulangan dengan perintah `for` relatif lebih mudah digunakan.

Bentuk Umum

```
for(inisialisasi;kondisi;ekspresi)
   proses yang akan diulang;
```

➔ **while**

Perintah `while` sama halnya dengan perintah `for`, yaitu melakukan perulangan selama kondisi "berhenti" dipenuhi. Perintah `while` akan mengecek apa kondisi sudah terpenuhi atau belum, jika belum maka proses akan dilaksanakan. Namun, jika sudah terpenuhi maka proses dihentikan.

Bentuk Umum

```
while(kondisi)
   proses yang akan diulang;
```

➔ **do while**

Pada perintah `for` dan `while`, pengecekan dilakukan sebelum melakukan perulangan. Namun, pada perintah `do while`, proses akan dijalankan terlebih dahulu kemudian dilakukan pengecekan terhadap kondisi. Jadi perintah ini paling tidak melakukan satu kali proses.

Bentuk Umum

```
do
   proses yang akan diulang;
while(kondisi);
```

5.3 Percobaan

- Tulislah program dibawah ini:

```
/*Program daftar makanan di Bioskop*/
#include "stdio.h"
int main()
{
char c;
int done;
float total=0;
printf("Silahkan buat pilihan anda:\n");
printf("1 - Minuman.\n");
printf("2 - Permen.\n");
printf("3 - Hot dog.\n");
printf("4 - Popcorn.\n");
printf("= - Selesai.\n");
printf("Pilihan anda:\n");
```

```
/* Memeriksa pilihan */
done=0;
while(!done)
{
     c=getchar();
     switch(c)
     {
     case '1':
          printf("Minuman\tRp.1500\n");
          total+=1500;
          break;
     case '2':
          printf("Permen\t\tRp.500\n");
          total+=500;
          break;
     case '3':
          printf("Hot dog\t\tRp.6000\n");
          total+=6000;
          break;
     case '4':
          printf("Popcorn\t\tRp.3000\n");
          total+=3000;
          break;
     case '=':
          printf("= Jumlah Rp.%.2f\n",total);
          printf("Silahkan bayar di kasir.\n");
          done=1;
          break;
     default:
          printf("Pilihan tidak benar.\n");
     } /* end switch */
} /* end while */
return(0);
}
```

- Simpan dengan nama `loop.c`.
- Compile program tersebut.
- Run program.

Bab 6

Larik (Array)[ ]

6.1 Tujuan

- ✔ Mengetahui jenis – jenis larik.
- ✔ Mengetahui penggunaan larik dalam bahasa C.

6.2 Teori

Larik (array) adalah kumpulan dari nilai – nilai yang bertipe sama dalam urutan tertentu dengan nama variabel yang sama. Letak atau posisi dari elemen larik ditunjukkan oleh indeks.

6.2.1 Larik Dimensi Satu

Bentuk Umum

```
Tipe_data nama_variabel [ukuran];
```

- ➔ Elemen data dapat diakses berdasarkan indeks.
- ➔ Indeks array secara *default* dimulai dari nol.

6.2.2 Larik Multi Dimensi

Bentuk Umum

```
Tipe_data nama_variabel [ukuran1][ukuran2][ukuranN};
```

- ➔ Elemen data dapat diakses berdasarkan indeks.
- ➔ Indeks array secara *default* dimulai dari nol.
- ➔ Untuk larik dua dimensi dapat dianggap sebagai sebuah matriks yang terdiri dari baris dan kolom.

6.3 Percobaan

- Tulislah program dibawah ini:

```
/*Program penjumlahan matriks dua dimensi*/
#include "stdio.h"
#include "conio.h"

void main()
{
```

```
//Deklarasi Variabel
int A[3][4], B[3][4], X[3][4], y[3][4], i, j;
clrscr();

/*Masukkan data matriks A*/
for(i=0;i<3;i++){
     for(j=0;j<4;j++){
     printf("input data matriks A[%i][%i]"i+1,j+1);
     fflush(stdin);
     scanf("%i", &A[i][j]);
     }
}

/*Masukkan data matriks B*/
for(i=0;i<3;i++){
     for(j=0;j<4;j++){
     printf("input data matriks B[%i][%i]"i+1,j+1);
     fflush(stdin);
     scanf("%i", &B[i][j]);
     }
}

/*Penjumlahan Matriks A dan B*/
for(i=0;i<3;i++){
     for(j=0;j<4;j++){
     x[i][j]=A[i][j]+B[i][j]
     }
}

/*Tampilkan matriks A*/
for(i=0;i<3;i++){
     for(j=0;j<4;j++){
     printf("%6i",A[i][j]);
     printf("\n");
     }
}

/*Tampilkan matriks B*/
for(i=0;i<3;i++){
     for(j=0;j<4;j++){
     printf("%6i",B[i][j]);
     printf("\n");
     }
}

/*Tampilkan hasil penjumlahan matriks A dan B*/
for(i=0;i<3;i++){
     for(j=0;j<4;j++){
     printf("%6i",x[i][j]);
```

```
            printf("\n");
            }
        }

        printf("\n\n");
        getch();
}
```

- Simpan dengan nama `array.c`.
- Compile program tersebut.
- Run program.

Bab 7

Fungsi( )

7.1 Tujuan

- ✔ Mengetahui fungsi – fungsi yang ada dalam bahasa C.
- ✔ Mengetahui penggunaan fungsi – fungsi dalam bahasa C.

7.2 Teori

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi main(). Fungsi banyak diterapkan dalam program-program C yang terstruktur. Keuntungan penggunaan fungsi dalam program yaitu program akan memiliki struktur yang jelas (mempunyai readability yang tinggi) dan juga akan menghindari penulisan bagian program yang sama

7.2.1 Beberapa Fungsi Pustaka Dalam Bahasa C

- ➔ Fungi Operasi String (dalam header file `string.h`)
  - ➲ `strcpy()`
    - ⇨ Berfungsi untuk menyalin suatu sreing asal ke variable tujuan.
    - ⇨ Bentuk umum : `strcpy(var_tujuan, string_asal);`
  - ➲ `strlen()`
    - ⇨ Berfungsi untuk memperoleh jumlah karakter dari suatu string.
    - ⇨ Bentuk umum : `strlen(string);`
  - ➲ `strcat()`
    - ⇨ Berfungsi menambahkan string sumber ke bagian akhir dari string tujuan.
    - ⇨ Bentuk umum : `strcat(tujuan, sumber);`
  - ➲ `strupr()`
    - ⇨ Berfungsi untuk mengubah setiap huruf dari suatu string menjadi huruf kapital.
    - ⇨ Bentuk umum : `strupr(string);`
  - ➲ `strlwr()`
    - ⇨ Berfungsi untuk mengubah setiap huruf dari suatu string menjadi hrurf kecil.
    - ⇨ Bentuk umum : `strlwr(string);`
  - ➲ `strcmp()`
    - ⇨ Berfungsi untuk membandingkan dua buah string.

- ⇨ Hasil dari fungsi ini bertipe *integer* dengan nilai :
  - Negatif → jika string pertama kurang dari string kedua.
  - Nol → jika string pertama sama dengan string kedua.
  - Positif → jika string pertama lebih dari string kedua.
- ⇨ Bentuk umum : `strcmp(string 1, string 2);`

➔ Fungsi Operasi Karakter (dalam header file `ctype.h`)

- ➲ `islower()`
  - ⇨ Menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kecil.
  - ⇨ Bentuk umum : `islower(char);`
- ➲ `isupper()`
  - ⇨ Menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf besar.
  - ⇨ Bentuk umum : `isupper(char);`
- ➲ `isdigit()`
  - ⇨ Menghasilkan nilai benar (bukan nol) jika karakter merupakan sebuah digit.
  - ⇨ Bentuk umum : `isdigit(char);`
- ➲ `tolower()`
  - ⇨ Mengubah huruf kapital menjadi huruf kecil.
  - ⇨ Bentuk umum : `tolower(char);`
- ➲ `toupper()`
  - ⇨ Mengubah huruf kecil menjadi huruf kapital.
  - ⇨ Bentuk umum : `toupper(char);`

➔ Fungsi Operasi Matematika (dalam header file `math.h` dan `stdlib.h`)

- ➲ `sqrt()`
  - ⇨ Menghitung akar dari sebuah bilangan.
  - ⇨ Bentuk umum : `sqrt(bilangan);`
- ➲ `pow()`
  - ⇨ Menghitung pemangkatan dari suatu bilangan.
  - ⇨ Bentuk umum : `pow(bilangan, pangkat);`
- ➲ `sin(), cos(), tan()`
  - ⇨ Menghitung nilai sinus, cosinus dan tangen dari suatu sudut;
  - ⇨ Bentuk umum : `sin(sudut); cos(sudut); tan(sudut);`
- ➲ `atof()`
  - ⇨ Mengkonversi nilai string menjadi bilangan bertipe *double.*
  - ⇨ Bentuk umum : `atof(char x);`
- ➲ `atoi()`
  - ⇨ Mengkonversi nilai string menjadi bilangan bertipe *integer.*
  - ⇨ Bentuk umum : `atoi(char x);`
- ➲ `div()`
  - ⇨ Menhitung hasil bagi dan sisa hasil bagi.
  - ⇨ Bentuk umum : `div_t div(int x, int y)`
  - ⇨ Strukturnya :

```
typedef struct{
     int quot; //hasil pembagian
     int rem; //sisa pembagian
     }div_t;
```

- `max()`
  - Menentukan nilai maksimal dari dua buah bilangan.
  - Bentuk umum : `max (bilangan 1, bilangan 2);`
- `min()`
  - Menentukan nilai minimal dari dua buah bilangan.
  - Bentuk umum : `min (bilangan 1, bilangan 2);`

7.2.2 Membuat Fungsi Sendiri

➔ **Deklarasi Fungsi**

Sebelum digunakan (dipanggil), suatu fungsi harus dideklarasikan dan didefinisikan terlebih dahulu. Bentuk umum pendeklarasian fungsi adalah :

```
tipe_fungsi nama_fungsi(parameter_fungsi);
```

Sedangkan bentuk umum pendefinisian fungsi adalah :

```
Tipe_fungsi nama_fungsi(parameter_fungsi)
{ statement
statement
.........
.........
}
```

Hal-hal yang perlu diperhatikan dalam penggunaan fungsi :

- Kalau tipe fungsi tidak disebutkan, maka akan dianggap sebagai fungis dengan nilai keluaran bertipe integer.
- Untuk fungsi yang memiliki keluaran bertipe bukan integer, maka diperlukan pendefinisian penentu tipe fungsi.Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void.
- Pernyataan yang diberikan untuk memberikan nilai akhir fungsi berupa pernyataan *return*.
- Suatu fungsi dapat menghasilkan nilai balik bagi fungsi pemanggilnya.

➔ **Parameter Formal dan Parameter Aktual**

- **Parameter Formal** adalah variabel yang ada pada daftar parameter dalam definisi fungsi.

Contoh parameter formal pada pendefinisian fungsi :

```
float tambah(float x, float y) //parameter formal
{ return (a+b);
}
```

- **Parameter Aktual** adalah variabel (parameter) yang dipakai dalam pemanggilan fungsi.

Contoh parameter aktual pada pemanggilan fungsi :

```
void main()
{ .......
.........
c = tambah(a, b); //parameter aktual
.........
}
```

➔ Cara Melewatkan Parameter

Cara melewatkan suatu parameter dalam Bahasa C ada dua cara yaitu:

- Pemanggilan Secara Nilai (*Call By Value*)
  - *Call by Value* menyalin nilai dari parameter aktual ke parameter formal.
  - Yang dikirmkan ke fungsi adalah nilai datanya, bukan alamat memori letak dari datanya.
  - Fungsi yang menerima nilai akan menyimpannya di alamat terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi.
  - Perubahan nilai pada parameter formal tidak akan merubah nilai parameter aktual.

- Pemanggilan Secara Referensi (*Call by Reference*)
  - *Call by Reference* adalah upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.
  - Yang dikirimkan ke fungsi adalah letak dari nilai datanya, bukan nilai datanya.
  - Fungsi yang menerima nilai akan menyimpannya di alamat yang sama dengan nilai aslinya.
  - Perubahan nilai pada parameter formal akan merubah nilai parameter aktual.

➔ Penggolongan Parameter Berdasarkan Kelas Penyimpanan

- Variabel Lokal
  - Variabel yang dideklarasikan di dalam fungsi.

Sifat-sifat variabel lokal :

- Secara otomatis akan diciptakan ketika fungsi dipanggil dan akan lenyap ketika proses eksekusi terhadap fungsi berakhir.
- Hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Tidak ada inisialisasi secara otomatis (saat variabel diciptakan nilainya random).
- Dideklarasikan dengan menambahkan kata **`auto`** (opsional).

➲ Variabel Global
↳ Variabel yang dideklarasikan di luar fungsi.

Sifat-sifat variabel global :
- Dikenal (dapat diakses) oleh semua fungsi.
- Jika tidak diberi nilai awal secar otomatis berisi nilai nol.
- Dideklarasikan dengan menambahkan kata **`extern`** (opsional).

➲ Variabel Statis
↳ Variabel yang nilainya tetap (dapat berupa variabel lokal dan global)

Sifat-sifat variabel statis :
- Nilai variabel statis tidak akan hilang walau eksekusi terhadap fungsi telah berakhir.
- Inisialisasi hanya diperlukan satu kalisaya, yaitu pada saat fungsi dipanggil pertama kali.
- Jika tidak diberi nilai awal secar otomatis berisi nilai nol.
- Dideklarasikan dengan menambahkan kata **`static`**

➲ Variabel Register
↳ Variabel yang disimpan pada register dan bukan pada RAM

Sifat-sifat variabel statis :
- Hanya dapat diterapkan pada variabel lokal yang bertipe `int` dan `char`
- Digunakan untuk proses perulangan.
- Proses perulangan akan lebih cepat karena variabel register memiliki kecepatan yang lebih tinggi daripada variabel biasa.
- Dideklarasikan dengan menambahkan kata **`register`**

7.2.3 Fungsi Rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri.

7.3 Percobaan

- Tulislah program dibawah ini:

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

void tukar (int *px, int *py); //Deklarasi Fungsi

void main(){

int a, b;
clrscr();
```

```
a=15;
b=10;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a=%i b=%1\n\n", a, b);

tukar(&a, &b);

printf("Nilai setelah pemanggilan fungsi\n");
printf("a=%i b=%1\n\n", a, b);

getch();
}

void tukar(int *px, int *py){
int z; //variabel sementara
z=*px;
*px=*py;
*py=z;
printf("Nilai di akhir fungsi tukar()\n);
printf("*px=%i *py=%i\n\n"), *px, *py);
}
```

- Simpan dengan nama `fungsi.c`.
- Compile program tersebut.
- Run program.

Bab 8

*Pointer

8.1 Tujuan

- ✔ Mengetahui penggunaan pointer dalam bahasa C.

8.2 Teori

8.2.1 Pengertian Pointer

Pointer (variabel penunjuk) adalah suatu variabel yang berisi alamat memori dari suatu variabel lain. Alamat ini merupakan lokasi dari obyek lain (biasanya variabel lain) di dalam memori. Contoh, jika sebuah variabel berisi alamat dari variabel lain, variabel pertama dikatakan menunjuk ke variabel kedua.

Operator pointer ada dua, yaitu:

- ➔ Operator &
  - ➲ Bersifat *unary*.
  - ➲ Menghasilkan alamat operandnya.

- ➔ Operator *
  - ➲ Bersifat *unary*.
  - ➲ Menghasilkan nilai operandnya.

8.2.2 Deklarasi Pointer

Seperti halnya variabel yang lain, variabel pointer juga harus dideklarasikan terlebih dahulu sebelum digunakan.

Bentuk Umum

```
tipe_data *nama_pointer;
```

Tipe data pointer mendefinisikan tipe dari objek yang ditunjuk oleh pointer. Secara teknis, tipe apapun dari pointer dapat menunjukkan lokasi(dimanapun)dalam memori. Bahkan operasi pointer dapat dilaksanakan relatif terhadap tipe dasar apapun yang ditunjuk. Contoh, ketika kita mendeklarasikan pointer bertipe int*, kompitler akan menganggap alamat yang ditunjuk menyimpan nilai integer, walaupun sebenarnya bukan (sebuah pointer int* selalu menganggap bahwa ia menunjuk ke sebuah obyek bertipe integer, tidak peduli isi sebenarnya). Karenanya sebelum mendeklarasikan sebuah pointer, pastikan tipenya sesuai dengan tipe objek yang akan ditunjuk.

Contoh:

```
int a, b, *px                int* a, b, px;
```

8.2.3 Operasi Pointer

- ➔ Operasi Penugasan.
- ➔ Operasi Aritmatika.
- ➔ Operasi Logika.

8.3 Percobaan

- Tulislah program dibawah ini:

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

void main()
{
//Deklarasi Variabel
int nilai[3], *p;
clrscr();

nilai[1]=125;
nilai[2]=345;
nilai[3]=750;
p=&nilai[0];

printf("Nilai %i ada di alamat memori %p\n", *(p+1),
p+1);

printf("Nilai %i ada di alamat memori %p\n", *(p+1),
p+1);

getch();
}
```

- Simpan dengan nama `point.c`.
- Compile program tersebut.
- Run program.

Bab 9

Operasi FILE

9.1 Tujuan

- ✔ Mengetahui jenis – jenis operasi file.
- ✔ Mengetahui penggunaan operasi file.

9.2 Teori

File adalah sebuah organisasi dari sejumlah record. Masing-masing record bisa terdiri dari satu atau beberapa field. Setiap field terdiri dari satu atau beberapa byte.

9.2.1 Membuka File

- ➔ Untuk membuka atau mengaktifkan file, fungsi yang digunakan adalah fungsi `fopen()`.
- ➔ File dapat berupa file biner atau file teks.
- ➔ File biner adalah file yang pola penyimpanan di dalam disk dalam bentuk biner, yaitu seperti bentuk pada memori (RAM) computer.
- ➔ File teks adalah file yang pola penyimpanan datanya dalam bentuk karakter.
- ➔ Penambahan yang perlu dilakukan untuk menentukan mode teks atau biner adalah `t` untuk file teks dan `b` untuk file biner.
- ➔ Prototype fungsi `fopen()` ada di header fungsi `stdio.h`

Bentuk Umum

```
file *fopen(char *namafile, char *mode);
```

Keterangan :

- ➔ namafile adalah nama dari file yang akan dibuka/diaktifkan.
- ➔ mode adalah jenis operasi file yang akan dilakukan terhadap file.

Jenis-jenis operasi file :

- ➔ `r` : menyarakan file hanya dapat dibaca (file harus sudah ada)
- ➔ `w` : menyatakan file baru akan dibuat/diciptakan (file yang sudah ada akan dihapus)
- ➔ `a` : untuk membuka file yang sudah ada dan akan dilakukan proses penambahan data (jika file belum ada, otomatis akan dibuat)
- ➔ `r+` : untuk membuka file yang sudah ada dan akan dilakukan proses pembacaan dan penulisan.
- ➔ `w+` : untuk membuka file dengan tujuan untuk pembacaan atau penulisan. Jika file sudah ada, isinya akan dihapus.
- ➔ `a+` : untuk membuka file, dengan operasi yang akan dilakukan berupa perekaman maupun pembacaan. Jika file sudah ada, isinya akan dihapus.

Contoh :

```
pf = fopen("COBA.TXT", "w");
```

9.2.2 Menutup File

- ➔ Untuk menutup file, fungsi yang digunakan adalah `fclose()`.
- ➔ Prototype fungsi fclose() ada di header file `stdio.h`

Bentuk Umum

```
int fclose(FILE *pf);
```

atau

```
int fcloseall(void);
```

9.2.3 Melaksanakan Proses File

- ➔ **Menulis Karakter**
  Untuk menulis sebuah karakter, bentuk yang digunakan adalah :

  ```
  putc(int ch, file *fp)
  ```

  `fp` adalah pointer file yang dihasilkan oleh `fopen()`
  `ch` adalah karakter yang akan ditulis.

- ➔ **Membaca Karakter**
  Untuk membaca karakter dari file, fungsi yang digunakan adalah :

  ```
  getc(file *fp);
  ```

  `fp` adalah pointer file yang dihasilkan oleh `fopen()`
  Fungsi `feof()`, digunakan untuk mendeteksi akhir file.
  Pada saat membaca data `foef(file *fp)`

- ➔ **Membaca dan Menulis String**
  Fungsi untuk membaca dan menulis string adalah : `fgets()` dan `fputs()`

  Bentuk Umum :

  ```
  fgets(char *str, int p, file *fp)
  fputs(char *str, file *fp)
  ```

- ➔ **Membaca dan Menulis Blok Data**
  Fungsi untuk membaca dan menulis blok data adalah : `fread()` dan `fwrite()`

Bentuk umum

```
fread(void *buffer, int b_byte, int c, file *fp);

fwrite(void *buffer, int b_byte, int c, file *fp);
```

Keterangan :

- `buffer` adalah pointer ke sebuah area di memori yang menampung data yang akan dibaca dari file.
- `b_byte` adalah banyaknya byte yang akan dibaca atau ditulis ke file
- `c` adalah banyaknya item dibaca/ditulis.

➔ **Membaca dan Menulis File yang Terformat**
Jika diinginkan, data bilangan dapat disimpan ke dalam file dalam keadaan terformat. Fungsi yang digunakan adalah :

```
fprintf(ptr_file, "string control", daftar
argument);

fscanf(pts_file, "string control", daftar argument);
```

9.2.4 File Sequensial

File sekuensial berisi rekord-rekord data yang tidak mengenal posisi baris atau nomor rekord pada saat aksesnya, dan setiap record dapat mempunyai lebar yang berbeda-beda. Akses terhadapnya selalu dimulai dari awal file dan berjalan satu persatu menuju akhir dari file. Dengan demikian, penambahan file hanya dapat dilakukan terhadap akhir file, dan akses terhadap baris tertentu harus dimulai dari awal file. Fungsi baku yang terkait dengan file sekuensial ini antara lain adalah **`fprintf`**, **`fscanf`**, dan **`rewind`** .

9.3 Percobaan

- Tulislah program dibawah ini:

```
#include "stdio.h"

FILE *in;
void BACA(int[]);
void CETAK(int[]);

void main()
{
int tabel[26]={0};
BACA(tabel);
CETAK(tabel);
```

```
}

void BACA(int huruf[]){
char c;
if((in = fopen("data.txt, "r"))) ==NULL)
     printf(File tidak bisa dibaca\n");
else
     while ((ch = fgetc(in)) != EOF){
          c=(((c>=97||(c<=122))?c-32:c);
          if((c>=65||(c<=90))
               huruf[c-65];
     }
fclose(in);
}

void CETAK (int huruf[]){
int c;
for(c=0c<=25;c++)
     printf("\n%x%5d", c+65, huruf[c]);
}
```

- Simpan dengan nama `file.c`.
- Compile program tersebut.
- Run program.