BAB 1

DASAR-DASAR BAHASA C

I. TUJUAN INSTRUKSIONAL KHUSUS

- ✓ Mengetahui dan memahami tentang Pemroraman Terstruktur
- ✓ Mengetahui dan memahami tentang perintah dasar pemrograman bahasa c.
- ✓ Membuat program yang melibatkan permasalahan-permasalahan yang sederhana

II. DASAR TEORI

Program bahasa C adalah suatu program yang terdiri dari satu atau lebih fungsi-fungsi.

1. Pengertian Pemrograman Terstruktur

Pemrograman Terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program. Selain pengertian diatas Pemrograman Terstruktur adalah suatu aktifitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis , dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami.

Prinsip dari pemrograman terstruktur adalah Jika suatu proses telah sampai pada suatu titik / langkah tertentu , maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya / kembali lagi ke baris sebelumnya, kecuali pada langkah – langkah untuk proses berulang (Loop).

Bahasa pemrograman yang mendukung pemrograman terstruktur:
C
Pascal
Delphi

2. Program C Dasar

Berikut ini adalah contoh potongan program sederhana yang ditulis dalam bahasa C untuk menampilkan kalimat hallo world pada layar monitor

```
#include <stdio.h>
main()
{
printf("hallo world");
}
```

Dalam hal ini **main()** adalah sebuah fungsi yang **harus selalu ada** pada setiap program C. sedangkan program utama diletakan didalam tanda brace {}. Pada dasarnya program utama ini berupa pemanggilan fungsi-fungsi standar yang tersedia pada library (misal dalam hal ini printf adalah fungsi library yang digunakan untuk menampilkan hasil pada layar) atau fungsi-fungsi yang dibuat oleh programmer.

Komponen-Komponen Program bahasa C

Setiap program C terdiri dari beberapa komponen yang dikombinasikan dengan susunan tertentu. Untuk mendapatkan pemahaman yang menyeluruh berkaitan dengan struktur program C dan komponen-komponen programnya, dibawah ini akan diberikan program yang relative lengkap walaupun ukurannya kecil

```
/* Program untuk menghitung perkalian dua bilangan. */
#include <stdio.h>
int a,b,c;
int kali(int x, int y);
main()
{
/* masukan bilangan pertama */
printf("Masukan bilangan antara 1 sampai 100: ");
scanf("%d", &a);
/* masukan bilangan kedua */
printf("Masukan bilangan lain antara 1 sampai 100: ");
scanf("%d", &b);
/* hitung dan tampilkan hasil */
c = kali(a, b);
printf ("%d kali %d = %d\n", a, b, c);
return 0;
}
/* fungsi untuk menghitung perkalian dua argumen*/
int kali(int x, int y)
{
return (x * y);
}
```

```
Masukan bilangan antara1 sampai 100: 35
Masukan bilangan lain antara 1 sampai 100: 23
35 kali 23 = 805
```

(keterangan: pada program sebenaranya nomer bilangan tidak ditulis, penulisan disini semata-matas digunakan hanya untuk memudahkan penjelasan)

- **Fungsi main() (baris 8 sampai 28)**

  Seperti yang telah disinggung diatas secara sekilas, komponen yang harus pada setiap program C adalah fungsi main(). Eksekusi program dimulai pada statement pertama pada main() dan berakhir di statemen terakhir dalam main().

- **Pengarah (directive) #include (baris 2)**

  Pengarah #include mengintruksikan pada compiler C untuk menambahkan isi file include (file header) pada program yang dibuat saat

compilasi. Pengarah #include pada contoh program diatas berarti “tambahkan isi file stdio.h”. Umumnya program C selalu membutuhkan satu atau lebih file header.

- **Definisi Variabel (baris 4)**

  Varibel adalah sebuah nama yang digunakan sebagai petunjuk lokasi tempat simpan data. Dalam C, sebuah varibel harus didefinisikan terlebih dahulu sebelum digunakan. Sebuah definisi varibabel menginformasikan pada compiler nama varibel dan tipe data yang bisa disimpan. Dalam program contoh, definisi varibel baris 4 int a,b,c mendefinisikan tiga variable dengan nama a, b dan c yang masing-masing akan menyimpan tipe data integer.

- **Prototipe Fungsi (baris 6)**

  Prototipe Fungsi berguna untuk menginformasikan pada compiler C nama dan argument-argumen fungsi yang ada pada program. Prototype ini harus ditulis sebelum fungsi tersebut digunakan, harus ditekankan disini bahwa prototype fungsi ini berbeda dengan definisi fungsi (function definition).

- **Statemen-Statemen Program (baris 11,12,15,16,19,20,22, dan 28)**

  Perintah – perintah operasi yang dijalankan suatu program C pada dasarnya merupakan statemen-statemen program: Statemen-stamen ini dapat berupa operasi matematika, pemanggilan fungsi, membaca input keyboard, menampilkan informasi pada layar, dan lain-lain. Perlu diingat disini bahwa akhir statemen pada C harus selalu diakhiri oleh tanda baca semicolon.

- **printf()**

  Statemen printf() (baris 11, 15, dan 20) adalah fungsi library untuk menampilkan informasi pada layar monitor. Statemen ini dapat menampilkan text sederhana (seperti pada baris 11 dan 15), atau pesan dan nilai satu atau lebih variable program (seperti pada baris 20)

- **scanf()**

  statemen scanf() (baris 12 dan 16) adalah fungsi library juga yang berguna untuk membaca data dari keybord dan memberikan data tersebut pada satu atau lebih variable program.

  Statemen program pada baris 19 adalah pemanggilan fungsi kali(). Dengan kata lain statemen ini mengeksekusi fungsi kali, dan mengirimkan argument a dan b pada fungsi, setelah fungsi selesai dipanggil, kali() mengembalikan return sebuah nila pada program yang selanjutnya disimpan pada variable c.

- **return**

  Pada baris 22 dan 28 terdapat statemen return, statemen return pada baris 28 merupakan bagian dari fungsi kali(). Yaitu untuk menghitung perkalian x dan y dan mengembalikan hasil pada statemen program yang memanggil kali(). Statemen return pada baris 22 mengembalikan nilai 0 pada system operasi setelah program selesai dilaksanakan.

- **Definisi Fungsi**

  Fungsi adalah bagian program yang bersifat independent yang ditulis atau dibuat untuk melakukan tugas-tugas tertentu. Setiap fungsi harus memiliki nama yang unik yang mencerminkan tugas yang akan dilaksanakan oleh fungsi tersebut.

  Fungsi dengan nama kali(), pada baris 26 sampai 29, adalah fungsi yang didefinisikan programmer (user-defined function).

  Bahasa C juga telah menyediakan fungsi-fungsi library yang sering digunakan oleh programmer, misal untuk operasi-operasi input/output, operasi matematika, dll. Yang perlu dilakukan oleh programmer untuk menggunakan fungsi-fungsi library ini adalah cukup meng include-kan file header yang berisi informasi fungsi-fungsi yang tersedia beserta argumennya, misal untuk memanggil fungsi trigonometri seperti sin, cos, dan lain-lain pada awal program harus kita tulis : #include <math.h>

- **Komentar Program**

  Komentar program bertujuan untuk dokumentasi listing sumber. Komentar pada C harus dimulai dengan /* dan diakhiri dengan */. Setiap bagian program yang berada diantara tanda tersebut tidak akan dicompile oleh compiler (tidak mempengaruhi ukuran file exe).

- **Braces (Baris 9, 23, 27, dan 29)**

  Setiap fungsi yang dibuat pada C harus berada didalam tanda kurung kurawal (brace), hal ini juga termasuk fungsi main(). Satu atau lebih statemen yang berada dalam tanda kurung kurawal dinamakan block.

3. TIPE DATA

Tipe data digunakan untuk mendeklarasikan suatu pengenal/variabel atau konstanta. Bahasa C menyediakan lima macam tipe data dasar, yaitu :

No	Tipe Data	Deklarasi	Keterangan
1	Integer	**int**	Nilai numerik bulat
2	Floating-point	**float**	Nilai numerik ketepatan pecahan tunggal
3	Double-point	**double**	Nilai numerik ketepatan pecahan ganda
4	Karakter	**char**	Karakter
5	Kosong	**void**	Tidak ada nilai / kosong

Selanjutnya int, float, double dan char dapat dikombinasikan dengan pengubah (modifier) signed, unsigned, long dan short. Hasil kombinasi tipe data tersebut secara keseluruhan adalah sebagai berikut :

Tipe	Lebar Dalam Bit	Jangkauan Nilai	
		Dari	Sampai Dengan
Int	16	-32768	32767
Signed int			
Short int			
Signed short int			
Unsigned int	16	0	65535
Unsigned short int			
Long int	32	-2147483648	2147483649
Signed long int			
Unsigned long int	32	0	4294967296
Float	32	3.4E-38	3.4E+38
Double	64	1.7E-308	1.7E+308
Long double	80	3.4E-4932	3.4E+4932
Char	8	-128	127
Signed char			
Unsigned char	8	0	255

Pengubah signed berarti nilai variabel dapat mempunyai tanda, sehingga dapat bernilai positif atau negatif.

4. VARIABEL

Variabel adalah suatu pengenal yang digunakan untuk menyimpan suatu nilai dan nilai dari variabel tersebut dapat berubah-ubah selama proses program.

a. Mendeklarasikan Variabel

Variabel belum dapat digunakan didalam program sebelum dideklarasikan terlebih dahulu. Dengan kata lain, deklarasi variabel harus dilakukan terlebih dahulu sebelum variabel tersebut digunakan.

Deklarasi variabel berarti memberitahukan kepada C tentang tipe data dari suatu variabel, sehingga C dapat mempersiapkan terlebih dahulu tempat dari variabel tersebut di memori.

- **Bentuk umum deklarasi**

```
typedata namavariabel;
```

- *Contoh* :

```
int jumlah;
float rata_rata;
char nama;
```

b. Menentukan tipe data variabel

Untuk menentukan tipe dari pada variabel tergantung dari jenis data yang akan di tampung pada variabel.

c. Penugasan variabel

Memberikan nilai awal kepada suatu variabel

- Bentuk umum

```
Namavariabel = nilai;
```

- *Contoh* :

```
jumlah = 5;
rata_rata = 2.5;
```

d. Pendeklarasian dan penugasan variabel

Nilai awal suatu variabel dapat langsung diberikan pada saat mendeklarasikannya.

- Bentuk umum

```
typedata namavariabel = nilai;
```

- *Contoh* :

```
int jumlah = 5;
float rata_rata = 2.5;
```

e. Tipe data string

Bahasa C tidak menyediakan deklarasi variabel untuk tipe string. Nilai string adalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk larik dimensi satu. Dengan demikian, untuk nilai string dapat dideklarasikan sebagai larik bertipe char.

- Bentuk umum deklarasi

```
char nama[elemen_karakter]
```

- Contoh deklarasi dan penugasan

```
char nama[17] = 'Jhon Travolta'
```

5. KONSTANTA

a Konstanta Bernama

Konstanta bernama adalah suatu pengenal yang berhubungan dengan nilai tetap (variabel konstan). C++ memungkinkan pendefinisian suatu konstanta bernama. Hal ini dilakukan dengan menggunakan kata kunci const.

- Bentuk umum :

```
const tipe_data nama_konstanta = nilai;
```

- *Contoh :*

const int bulan = 12;

const float phi = 3.14285;

- Keterangan :
  1. variabel bulan adalah konstanta bernama bertipe integer dengan nilai 12.
  2. variabel phi adalah konstanta bernama bertipe float dengan nilai 3.141592.

Konstanta bernama berbeda dengan variabel, suatu konstanta bernama nilai yang ada padanya tidak dapat diubah setelah didefinisikan.

Keuntungan pemakaian konstanta bernama adalah untuk menghindari salah tulis terhadap nilai yang terkandung didalam variabel.

Contoh :

```
#include <stdio.h>
main( )
{
const float phi = 3.14285;
float luas, jari_jari;
printf (" Masukkan Nilai Jari-jari? = ");
scanf("%f",&jari_jari);
luas = 2 * phi * jari_jari;
printf(" Luas lingkaran = %f \n",luas);
}
```

b Konstanta Karakter dan Konstanta String

Konstanta karakter merupakan nilai sebuah karakter yang ditulis diantara tanda petik tunggal (‘ ‘), Contoh : ‘a’, ‘7’, dll. Konstanta string (literal string) merupakan nilai sebuah atau lebih karakter yang ditulis diantara tanda petik ganda (“ “), Contoh : “a”, “bahasa c”, dll.

c Konstanta Karakter Escape

Konstanta karakter escape banyak digunakan di statemen-statemen untuk menampilkan hasil, misal menggeser kursor ke baris berikutnya (ganti baris

baru) , membuat kursor kembali ke kolom pertama (carriage return) atau membunyikan bel dan lain sebagainya. Suatu konstanta karater escape diawali dengan karakter atau tanda ‘ \’.

III. LANGKAH KERJA

a. Program pertama

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
main()
{
   clrscr();
   printf("Program pertama C oleh Raras\n");
   printf("Kelas IK1B\n");
   printf("NIM:3.34.11.1.21");
   printf("Politeknik Negeri Semarang");
getch();
}
```

5. Alt+F, pilih save ketik nama lat1a.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Menampilkan tipe data string, integer, floating, dan karakter

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
main()
{
  clrscr();
  printf("\tNama Mahasiswa:%s\n", "Raras");
  printf("\tNo Absen:%d\n", "21");
  print("\tKelas:%s\n", "IK1B");
  printf("\tNilai:%f Predikat:%c\n", 95,5 ,'B');
getche();
}
```

5. Alt+F, pilih save ketik nama lat1b.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ

8. Alt+C, compile to EXE
9. Alt+R

c. Komentar program

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/*------------------------------*
 * Program Prl_3.c        *
 * Dibuat oleh : Raras         *
 * Tanggal : 17 desember 2011  *
 *------------------------------*/
 #include<stdio.h>

 main()
 {
  clrscr();
  printf("Pemberian Komentar\n");
  printf("Tidak dieksekusi\n");
 getch();
 }
```

5. Alt+F, pilih save ketik nama lat1c.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Apa yang dimaksud dengan interpreter dan compiler.

2. Sebutkan arti dari fungsi main(), printf(), dan #include.

3. Apa kegunaan dari perintah \n,\t,\f,\d, dan \c.

4. Buatlah program dan simpan dengan nama prl_4c :
   Selamat
   Belajar
   "Turbo C"
   Bahasa Pemograman Arras Menengah/Medium Level Language
   *Perintah * *\ berarti komentar*
   Tidak dieksekusi

BAB 2

STRUKTUR PERCABANGAN SEDERHANA

I. TUJUAN INSTRUKSIONAL KHUSUS

1. Mengetahui dan memahami tentang peritah percabangan bersyarat dalam bahasa C.
2. Membuat program yang melibatkan permasalahan-permasalahan percabangan dimana komputer harus melakukan pemilihan untuk menuju ke salah satu cabang berdasarkan kondisi tertentu.

II. DASAR TEORI

Salah satu program adalah proses seleksi atau kondisional. Dalam bahasa C ada beberapa antara lain :

1) if

Bentuk IF Sederhana Pernyataan IF dipakai untuk mengambil keputusan berdasarkan suatu syarat/kondisi.

Kondisi yang digunakan berupa kondisi ungkapan yang melibatkan operator relasi atau operator logika.

Bentuk Penulisan:

- ♦ if (kondisi)
- ♦ Pernyataan

Keterangan :

- Kondisi digunakan untuk menentukan penganbilan keputusan.
- Pernyataan dapat berupa sebuah pernyataan atau statement. Bagian pernyataan akan dijalankan/ diproses hanya jika kondisi bernilai benar.

2) if else

Kondisi adalah ungkapan bernilai Boolean maka untuk menyeleksi kondisi tersebut perlu tanda-tanda operasi (pembanding).

Macam operator dalam Bahasa C :

Operator relasi

Tanda	Arti
=	Sama dengan
!= atau <>	Tidak sama dengan
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan
>	Lebih besar dari
<	Lebih kecil dari

Operator Logika	
Tanda	Arti
&&	DAN (logical AND)
||	ATAU (logical OR)
!	BUKAN (logical NOT)

Struktur if dan if else adalah sebagai berikut :

Struktur if	**Struktur if else tunggal**	**Struktur if else jamak**
If(syarat) *Perintah 1;* *Atau* *If(syarat)* *{* *Perintah 1;* *Perintah 2;* *Perintah 3;* *... ... ...* *Perintah1;* *}*	*if(syarat)* *perintah 1;* *else* *perintah 2;*	*If(syarat)* *{* *Perintah 1a;* *Perintah 1b;* *... ... ...* *Perintah 1n;* *}* *Else* *{* *Perintah 2a;* *Perintah 2b;* *... ... ...* *Perintah 2a;* *}*

Keterangan:

- syarat yang dapat berupa ekspresi relasional atau logika
- kata tunggal berarti hanya ada satu perintah yang dikerjakan bila suatu syarat terpenuhi,
- kata jamak berarti ada dua atau lebih eprintah yang dikerjakan bila suatu syarat terpenuhi, perintah jamak diawali dengan atanda kurung kurawa buka dan di kurung kurawal tutuup "{}".
- Utnuk percabangan if else tidak sealu harus semua, akan tetapi bisa kombinasi antara keduanya tergantung dari permasalahannya.

3) nested if

Nested if atau struktur if bersyarang atau struktur if dalam if digunakan untuk menyelesaikan masalah-masalah yang memiliki lebih dari dua cabang. Seperti halnya struktur if atau if else, alternative-alternatif dalam nested if dapat berupa alternative hanya terdiri dari satu perintah (tunggal) atau alternative yang terdiri dari sejumlah perintah (jamak) ataupun kombinasi antar keduanya.

Bentuk umum nested if :

```
if (kondisi 1)                  else/*JIkatidak*/
 {                          {
   proses 1b;                   proses 1a;
 if(kondisi 3)                    if(kondisi 2)
  proses 3b;                       proses 2b;
 else                         else
  proses 3a;                       proses 2a;
 proses 3c;                   proses 2c;
  }                          }
```

Perhatian :

Bila perintah if atau else perintah tidak diapit tanda {} maka hanya satu perintah setelah if atau else yang diuji berdasarkan syarat atau kondisi dalam percabangan tersebut.

III. LANGKAH KERJA

a. Menentukan suatu keputusan mendapatkan korting

1. Pilih menu→program→accesoris→command prompt
2. Jalankan program turbo C
3. Alt+F, pilih new
4. Ketik :

```
/*----------------------------------------*
 * Program Lat4a.c                        *
 * Untuk menentukan besaran korting yang* *
 * diterima seorang pembeli               *
 *----------------------------------------*/
 #include <stdio.h>
 #include <conio.h>
 main()

 {

double   total_pembelian,   korting,   jml_byr;   /*
deklarasi variable bertipe bilangan double */

  clrscr();
```

```
  printf("Total pembelian anda : Rp ");
  scanf("%lf", &total_pembelian);
  korting = 0";

  /* proses seleksi kondisi */
  if (total_pembelian >= 50000)
  korting = 0.05 * total_pembelian;

  jml_byr = total_pembelian – korting;
/* menghitung jumlah bayar */

  clrscr();
  printf("Total Pembelian          =              Rp
%.2lf\n",total_pembelian);
printf("Besarnya Korting       = Rp %.2lf\n",korting);
printf("Jumlah yang dibayarkan     =              Rp
%.2lf\n",jml_byr);
  getch();
  }
```

5. Alt+F, pilih save ketik nama Lat2a.c
6. Alt+F, pilih directories, ganti semua dengan C:\TC,lalu di esc
7. Alt+F, pilih Compile to OBJ
8. Alt+F, pilih Compile to EXE
9. Alt+X, pada tampilan Command Prompt ketik lat2a.exe lalu tekan enter

b. <u>Menentukan suatu keputusan seorang siswa lulus atau gagal dalam ujian</u>

1. Pilih menu→program→accesoris→command prompt
2. Jalankan program turbo C
3. Alt+F, pilih new
4. Ketik :

```
/*-----------------------------------------*
 * Program Lat4b.c                         *
 * Untuk menguji suatu nilai yang diinputkan *
 * dari keyboard                           *
 *-----------------------------------------*/
 #include <stdio.h>
 #include <conio.h>
```

```
main()
{
  int nilaiujian;
  clrscr();
  printf("Masukkan nilai ujian anda : ");
  scanf("%d", &nilaiujian);
  /* proses seleksi kondisi */

  if  (nilaiujian > 60)
puts("kamu berhasil lulus ujian");
else
puts("kamu gagal");
getch();
}
```

5. Alt+F, pilih save ketik nama Lat2b.c
6. Alt+F, pilih directories, ganti semua dengan C:\TC,lalu di esc

7. Alt+F, pilih Compile to OBJ
8. Alt+F, pilih Compile to EXE
9. Alt+X, pada tampilan Command Prompt ketik lat2b.exe lalu tekan enter

c. <u>Menunjukkan suatu perintah jamak dalam suatu percabangan</u>

1. Pilih menu→program→accesoris→command prompt
2. Jalankan program turbo C
3. Alt+F, pilih new
4. Ketik :

```
/*-----------------------------------------------*
 * Program Lat4c.c                               *
 * Untuk menguji suatu nilai yang diinputkan     *
 *            dari keyboard                      *
 *-----------------------------------------------*/
 #include <stdio.h>
 #include <conio.h>
 main()
     {
  int bilangan;
  clrscr;
  printf("Masukkan sebuah bilangan bulat : ");
  scanf("%d", &bilangan);
  /* proses seleksi kondisi */
  if (bilangan % 2)
          {
       printf("Bilangan %d tidak habis dibagi 2 \n",
bilangan);
       puts("Karena  itu  termasuk  sebagai  bilangan
Ganjil");
           }
            else
```

```
            {
        printf("Bilangan   %d   habis   dibagi   2\n",
bilangan);
        puts("Karena   itu   termasuk   sebagai   bilangan
Genap");
             }
   }
```

5. Alt+F, pilih save ketik nama Lat2c.c
6. Alt+F, pilih directories, ganti semua dengan C:\TC,lalu di esc
7. Alt+F, pilih Compile to OBJ
8. Alt+F, pilih Compile to EXE
9. Alt+X, pada tampilan Command Prompt ketik lat2c.exe lalu tekan enter

d. <u>Kasus nested if</u>

1. Pilih menu→program→accesoris→command prompt
2. Jalankan program turbo C
3. Alt+F, pilih new
4. Ketik :

```
/*-----------------------------------------*
* Program Lat4d.c                          *
* Untuk seleksi calon menantu              *
* Hanya guyoo..n waton lhoooo       *
*-----------------------------------------*/
#include <stdio.h>
#include <conio.h>
main()
 {
float gaji;
int tinggi, usia;
char kerja, agama, nama[20];

clrscr;
puts("\t\tData pribadi saya");
puts("================================");
printf("Masukkan nama anda     :");
scanf("%s",&nama);

printf("Apakah anda taat beragama (T/B) : "); /* T
bila taat dab B bila biasa saja */
agama =getche();

printf("\nApakah anda sudah bekerja(S/B)     : ");/* S
bila sudah kerja dan B bila belum kerja */
kerja=getche();

printf("\nMasukkan usia dan tinggi anda :");
scanf("%d %d", &usia, &tinggi);
```

```
clrscr();
gaji = 0;
/* proses seleksi kondisi */
if (((agama ==’T’) :: (agama == ‘t’)) && ((kerja
==’S’):: (kerja == ‘s’)))
   if ((usia >= 25) & (tinggi >= 160))
   {
    puts(“Syarat menantu pertama sudah
     terpenuhi”);
     puts(“Eeeee tapi jangan bangga dulu,
     masih ada syarat lain lhoooo”);
     printf(“\n\nSebutin berapa gaji mas %s dulu
    : “,nama);
     scanf(“%f”, &gaji); clrscr();
    }


if(gaji >=  1000000)
      {
          clrscr();
          printf(“\nSelamat  yaaa  mas?  Semua  syarat
          sudah terpenuhi”);
          printf(“\nMas   %s bisa jadi mantu guee
          sekarang”,nama);
          printf(“\a”); printf(“\a”);
       }
       else
           printf(“Mas   %s maaf yaa ? tolong cari
        calon yang lain ajaaa deh ?\n”, nama);
 printf(“\n\npress any key......”);getch();
  }
```

5. Alt+F, pilih save ketik nama Lat2d.c
6. Alt+F, pilih directories, ganti semua dengan C:\TC,lalu di esc
7. Alt+F, pilih Compile to OBJ
8. Alt+F, pilih Compile to EXE
9. Alt+X, pada tampilan Command Prompt ketik lat2c.exe lalu tekan enter

IV. PERTANYAAN

1. Apa yang dimaksud dengan perintah jamak dan tunggal? Jelaskan dengan singkat dan jelas.
2. Sebuah toko membuka usaha fotocopy dengan ketentuan, jika jumlah total lembar yang difotocopy >= 100, maka harga perlembar Rp. 50, tetapi jika kurang dari 100, maka harga perlembar Rp. 60. Buatlah flowchart dan programnya.
3. Pengambangan dari soal di atas, ternyata perusahaan itu cukup maju dan mempunyai banyak langganan. Untuk langganan biaya fotocopynya adalah

Rp. 50. Modifikasilah flowchart dan program d atas sehingga ketentuan tersebut bisa dipenuhi.

4. Diketahui persamaan uadrat $ax^2 + bx + c = 0$. Buatlah program untuk menghitung nilai-nilai akarnya dengan ketentuan nilai deskriminasinya sebagai berikut :

   $D = b^2 - 4ac$

   D = 0, maka nilai akarnya x1 = x2 = $\frac{-b}{2a}$

   D>0, maka nilai akarnya x1 = -b + $x = \frac{\sqrt{D}}{2a}$

   x2 = -b - $\frac{\sqrt{D}}{2a}$

   D < 0, maka akarnya imajiner (Tampilkan keterangan “Akar imajiner”).

BAB 3

STRUKTUR PERCABANGAN BERTINGKAT

I. TUJUAN

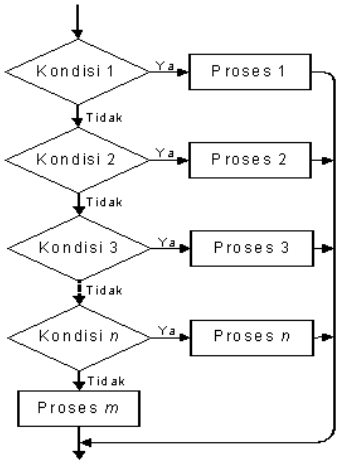
- Mengetahui dan menjelaskan tentang struktur algoritma percabangan bersyarat (if-then-else bertingkat, case).
- Membuat algoritma dalam bentuk flowcharts dan pseudocode yang melibatkan permasalahan-permasalahan percabangan bersyarat(if-then-else bertingkat, case)dimana komputer harus melakukan pemilihan untuk menuju ke salah satu cabang berdasarkan kondisi tertentu.
- Mengimplementasikan algoritma yang telah dibuat ke dalam bahasa pemrograman dengan menggunakan bahasa Pascal.

II. DASAR TEORI

Salah satu proses didalam suatu program adalah proses seleksi atau kondisional. Dalam bahasa C ada beberapa antara lain.

1. if-else bertingkat

♦ Struktur diagram alir :



Bentuk umum If Else bertingkat

```
If (kondisi 1) then
    Perintah 1;
Else
        If (kondisi 2)
          Perintah 2
  Else
.....
```

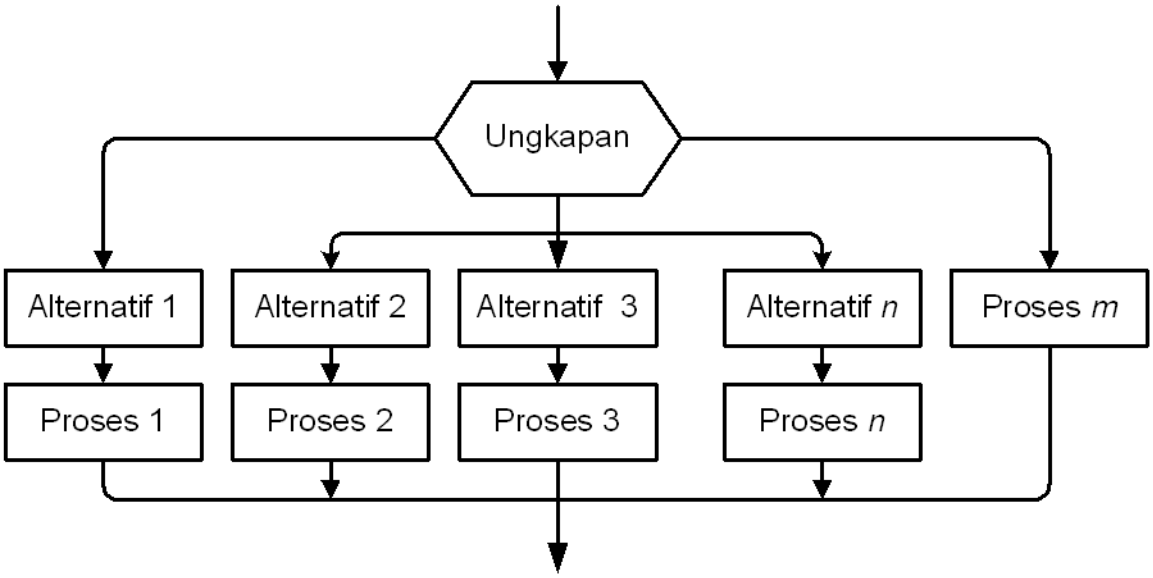
Catatan :

KONDISI adalah ungkapan yang digunakan untuk menyatakan suatu yang disyaratkan (ungkapan akan bernilai boolean(kondisi true atau false))

2. **Pernyataan Switch**

Pernyataan switch merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah alternative.

- Struktur diagram alir :



Bentuk Umum pernyataan Switch

```
Switch (ungkapan)
{
  Case konstanta_1;
        Pernyataan_1;
```

Break;

Case konstanta_2;

Pernyataan_2;

Break;

……….

Case konstanta_n;

Pernyataan_n;

Break;

Default;

Pernyataan_m

}

Keterangan :

Ungkapan atau *ekpresi* dapat berupa ungkapan bernilai integer atau bertipe karakter. Setiap *konstanta-I (konstanta_1, konstanta_2, konstanta_3…… konstanta_n)* dapat berupa konstanta integer atau karakter. Setiap *pernyataan_I (pernyataan_1, pernyataan_2, pernyataan _3 …. Pernyataan_n)* dapat berupa pernyataan tunggal atau pernyataan jamak.

Seperti halnya pengujian dalam **IF-ELSE**, pengujian dalam **SWITCH** juga dimulai dari konstanta_1 . Bila suatu nilai ungkapan sesuai dengan nilai konstanta, maka proses yang ada dalam konstanta tersebut yang akan dikerjakan dan diakhiri dengan kata kunci *break* (berakhir). Bila nilai ungkapan tidak ada yang cocok dengan semua nilai konstanta maka proses default yang dikerjakan.

III. LANGKAH KERJA

a. Menentukan hari dalam seminggu

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/*-------------------------------------------*
 * Program Lat5a.c                 *
 * contoh pemakaian if-else bertingkat      *
 * Untuk menentukan hari dalam seminggu *
```

```
*---------------------------------------------*/
#include <stdio.h>
main()
{
int kode_hari;
puts("=============================================");
puts("Menentukan hari                        ");
puts("1=SENIN   2=SELASA  3=RABU         4=KAMIS   ");
puts("5=JUM'AT  6=SABTU        7=MINGGU      ");
puts("=============================================");
printf("Masukkan kode hari (1..7)      :     ");
scanf("%d", &kode_hari);
if (kode_hari == 1)
   puts("Hari SENIN");
else
     if (kode_hari == 2)
    puts("Hari SELASA");
      else
     if (kode_hari == 3)
         puts("Hari RABU");
     else
          if (kode_hari == 4)
       puts("Hari KAMIS");
           else
        if (kode_hari == 5)
            puts("Hari JUM'AT");
        else
             if (kode_hari == 6)
              puts("Hari SABTU");
             else
            if (kode_hari == 7)
                puts("Hari MINGGU");
            else
                puts("Kode yang anda masukkan salah!");
 getch();
}
```

5. Alt+F, pilih save ketik nama lat3a.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc

7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Menentukan jumlah hari dari bulan dalam setahun

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :
5. Alt+F, pilih save ketik nama lat3b.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc

7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Buatlah program yang dapat digunakan untuk menjumlahkan atau megalikan atau mengurangkan atau membagi dua bilangan. (sekali dijalankan hanya mengerjakan salah satunya).
2. Anda serang mengetahui perintah if – else secara bertingkat. Coba terangkan jalannya eksekus program jika menemui instruksi (perintah tersebut).
3. Demikian juga untuk perintah switch, coba terangkan jalannya eksekusi program jika menemui instruksi (perintah tersebut).
4. Menurut pendapat anda mengapa semua permasalahan percabangan yang dapat diselesaikan dengan struktur if-else tidak mesti dapat diselesaikan menggunakan struktur switch? Jelaskan dengan singkat dan jelas dan sertai pula dengan contoh programnya.

BAB 4

STRUKTUR PERULANGAN

I. TUJUAN

1. Mengetahui dan memahami tentang perintah perulangan *(loop/repitive)* dalam bahasa C.
2. Menjelaskan tentang bentuk umum atau struktur dari perintah-perintah perulangan tersebut.
3. Membuat progrma yang melibatkan permasalahan-permasalahan yang membutuhkan proses perulangan.

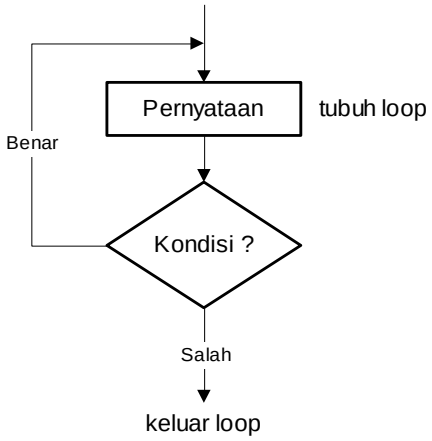
II. DASAR TEORI

Salah satu proses di dalam suatu program adalah proses perulangan (loop). Dalam bahasa C ada beberapa antara lain:

1. Do-while

Pada pernyataan do-while, tubuh loop (berupa pernyataan) berada diawal, sedangkan kondisi berada dibelakang. Berikut ini low chart dan bentuk umum pernyataan do-while:

Struktur diagram alirnya:



Bentuk umum struktur DO-WHILE

do

aksi /tubuh perulangan

Catatan :

- KONDISI adalah ungkapan yang digunakan untuk menyatakan suatu yang diisyaratkan (ungkapan akan bernilai boolean (kondisi true atau false))
- Pernyataan adalah perintah-perintah yang dikerjakan selama loop berlangsung. Biasanya perulangan dalam d0-while pasti berupa pernyataan jamak.

Contoh penggunaan do-while, untuk mengunci pemakai agar yang ditekan hanya tombol ‘Y’, ‘y’, ‘T’, atau ‘t’ (bila tombol lain yang ditekan maka loop akan berjalan terus):

```
cacah←1 { inisialisasi variable counter }
do
output (“Selamat Belajar Algoritma”)
cacah←cacah+1 {increment variabel counter}
while(cacah ≤ 100)
```

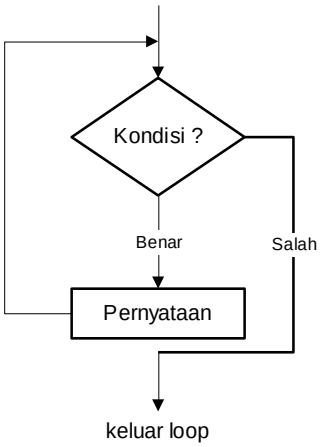
Contoh 2 penggunaan DO-WHILE, untuk mengunci pemakai agar yang ditekan hanya tombol ‘Y’,’y’, ‘T’, atau ‘t’ (bila tombol lain yang ditekan maka loop akan berjalan terus):

```
do
pilih ← input(karakter)  /* baca tombol */
sdh_bnr ← (pilih =’Y’) OR (pilih =’y’) OR
        (pilih =’T’) OR (pilih =’t’)
while(NOT sdh_bnr)
```

2. Pernyataan while

Berbeda dengan *do-while*, pada pernyataan while pengujian terhadap *loop* dilakukan dibagian awal (sebelum tubuh *loop*), untuk lebih jelasnya perhatikan *flow chart* dan bentuk umum pernyataan while berikut :

Struktur diagram alirnya :



Bentuk umum struktur WHILE

while(kondisi)

aksi/tubuh perulangan

Keterangan :

Dari gambar disamping terlihat bahwa kemungkinan pernyataan sebagai tubuh loop tidak akan pernah dijalankan selama kondisi yang disyaratkan oleh *while* tidak terpenuhi.

Contoh 1. Penggunaan WHILE untuk proses perulangan yang jumlah perulangannya dapat diketahui.

```
cacah←1 { inisialisasi variable counter }

while(cacah ≤ 100)
```

```
    output ("Selamat Belajar Algoritma")
    cacah←cacah+1 {increment variabel counter}
endwhile
```

Contoh 2 penggunaan *WHILE*, untuk kasus yang sama seperti permasalahan di atas:

```
sdh_bnr ← 0 { diberi nilai awal dahulu }
while(NOT sdh_bnr)
   pilih←input(karakter)  { baca tombol }
   sdh_bnr ← (pilih ='Y') OR (pilih ='y') OR
             (pilih ='T') OR (pilih ='t')
  Endwhile
```

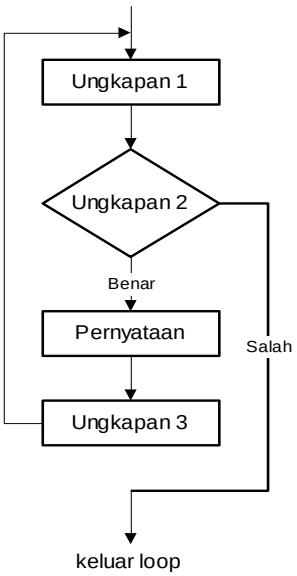
3. Struktur REPEAT

Secara prinsip kerja struktur REPEAT sama dengan struktur Do-WHILE. Perbedaan mendasar antara keduanya adalah pada masalah logika pengulangan, yaitu perulangan dalam struktur REPEAT akan berjalan selama kondisi bernilai salah sedang pada struktur Do-WHILE adalah sebaliknya.

4. Struktur FOR

Struktur perulangan yang lain adalah struktur FOR. Bentuk umum dan flow chart dari struktur FOR adalah sebagai berikut :

Struktur diagram alirnya :



Bentuk umum Struktur FOR

<u>for</u> pencacah ← nilai_awal <u>to</u> atau <u>downto</u> nilai_akhir <u>do</u>

aksi

<u>Keterangan :</u>

- pencacah: dipakai sebagai variabel pengendali loop (variabel counter).
- Nilai_awal: diberikan untuk nilai awal variabel counter
- Nilai_akhir: dipakai sebagai terminator proses perulangan

Contoh pemakaian FOR untuk kasus di atas:

for cacah ← 1 to 100 do

```
output ("Selamat Belajar Algoritma")
```

5. Nested loop (loop dalam loop)

Dalam suatu loop bisa terkandung loop yang lain. Loop seperti ini dikenal dengan istilah loop dalam loop (*nested loop*). Berikut ini contoh permasalahan yang membutuhkan *nested loop* untuk membuat tabel perkalian :

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	4	6	8	10	12	14	16
3	3	6	9	12	15	18	21	24
4	4	8	12	16	20	24	28	32
5	5	10	15	20	25	30	35	40
6	6	12	18	24	30	36	42	48
7	7	14	21	28	35	42	49	56
8	8	16	24	32	40	48	56	64

```
for baris ← 1 to maks_baris do
    for kolom← 1 to maks_kolom do
     Hasil_kali = baris * kolom
     output(hasil_kali)
```

III. LANGKAH KERJA

a. Langkah menampilkan "TURBO C" sebanyak 10 kali :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/* ---------------------------------------------*
 * Program Lat6a.c                  *
 * contoh pemakaian do-while             *
 * menampilkan tulisan "Turbo C" 10 kali*
 *---------------------------------------------*/
 #include<stdio.h>
```

```
main()
{
int pencacah;
pencacah=0;
do
   {
    puts("TURBO C");
    pencacah++;
   }
while(pencacah < 10);
printf("\nSelesai.........");
getch();
}
```

5. Alt+F, save dengan nama Lat4a.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

b. Membaca tombol Y atau T :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/* ---------------------------------------*
 * Program Lat6b.c                       *
 * contoh pemakaian do-while             *
 * Untuk membaca tombol Y atau T         *
 *---------------------------------------*/
#include <stdio.h>
#include <conio.h>
main()
{
int sudah_benar;
char pilihan;
printf("Pilihlah Y atau T : ");
/*program hanya mau dilanjutkan kalau tombol
   Y, y, T atau t yang ditekan*/

do
{
pilihan = getch (); /*baca tombol*/
sudah_benar = (pilihan=='Y') || (pilihan=='y') ||
         (pilihan=='T') || (pilihan=='t');
}
while(!sudah_benar);

/*memberi keterangan tentang pilihan*/
switch (pilihan)
 {
```

```
 case 'Y' :
 case 'y' :
  puts("\nPilihan anda adalah Y");
  break;
 case 'T' :
 case 't' :
  puts("\nPilihan anda adalah T");
  break;
 }
 getch();
}
```

5. Alt+F, save dengan nama Lat4b.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

c. Menghitung jumlah kata dan karakter dalam suatu kaliamt

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/* ---------------------------------------------*
 * Program Lat6e.c                  *
 * contoh pemakaian do-while            *
 * menghitung jumlah kata dan karakter       *
 * dalam suatu kalimat                  *
 *----------------------------------------------*/
 #include <stdio.h>
 #include <conio.h>
 #define ENTER '\r' /*karakter CR*/
 #define SPASI ' ' /*karakter spasi*/

 main()
 {
 char karakter;
 int jumkar = 0;
 int jumspasi    = 1;
 clrscr();
 puts("Ketikkan sebuah kalimat dan akhiri dengan ENTER");
 puts("Saya  akan  menghitung  jumlah  karakter  dan  kata  pada
kalimat tersebut\n");
 while( (karakter = getche ()) !=ENTER)
   {
    jumkar++;
    if(karakter ==SPASI)jumspasi++;
   }
 printf("\nJumlah karakter : %d",jumkar);
```

```
printf("\nJumlah kata       : %d",jumspasi);
getch();
}
```

5. Alt+F, save dengan nama Lat4e.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

d. Mencari rata-rata dari sejumlah bilangan denga struktur *for* :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/* ---------------------------------------------*
 * Program Lat6f.c                    *
 * contoh pemakaian for                    *
 * Mencari rata-rata dari sejumlah bilangan   *
 *---------------------------------------------*/
main()
{
 int i,n;   /*n adalah jumlah bilangan*/
 float bil,rata,total=0;  /*bil adalah bilangan yang akan
                  Anda masukkan*/

 printf("\nJumlah bilangan : ");
 scanf("%d",&n);

 for(i=1;i<=n;i++)
 {
  printf("Bilangan ke %d : ",i);
  scanf("%f",&bil);
  total+=bil;
 }

 rata=total/n;
 printf("Rata-rata bilangan tersebut : %f",rata);

}
```

5. Alt+F, save dengan nama Lat4f.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

e. Membuat tabel perkalian studi kasus *nested loop* dengan struktur *for* :

1. Start → Program → Accesories → Command Prompt

2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/* ------------------------------------*
 * Program Lat6g.c              *
 * contoh pemakaian nested loop      *
 * membuat tabel perkalian      *
 *------------------------------------*/
 #include <stdio.h>
 #include <conio.h>
 main()
 {
 int baris, kolom, hasil_kali, maks = 0;
 clrscr();
 printf("masukkan jumlah maksimal baris / kolomnya : ");
 scanf("%d",&maks);
 /*kalang nested loop*/
 for(baris = 1;baris <=maks ; baris++)
  {
   for(kolom = 1;kolom <=maks ; kolom++)
    {
    hasil_kali = baris * kolom;
    printf("%2d", hasil_kali);
    }
  printf("\n"); /*pindah baris*/
  }
 getch();
 }
```

5. Alt+F, save dengan nama Lat4g.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

f. Membuat tabel perkalian studi kasus pemakaian pernyataan *break* :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/* ------------------------------------------*
 * Program Lat6h.c                  *
 * contoh pemakaian break dan continue        *
 * Program menu sederhana           *
 * Operasi dua bilangan                  *
 *------------------------------------------*/
 #include <stdio.h>
 main()
 {
```

```
char x ='1',ket;
float bil1,bil2.hasil;
int i;
clrscr();

while(1)
{
  /*Cetak menu*/
  for(i=1;i<20;++i)
 printf("\xDB");
 printf("\n\n");

  printf("\rOPERASI DUA BILANGAN");
  printf("\n1.Penjumlahan");
  printf("\n2.Selisih");
  printf("\n3.Perkalian");
  printf("\n4.Pembagian");
  printf("\n5.Selesai");

  printf("\n\n");
  for(i=1;i<20;++i)
 printf("\xDB");

  printf("\n\nPilihan Anda :");
  x=getche();

  if(x=='5');
    break;
    clrscr();

  /*Pilihan yang salah*/
  if (x<'1' || x>'5')
     {
   printf("Anda salah memilih");
   printf("\nTekan sembarang tombol");
   getch();
   continue; /*Kembali ke Menu*/
     }

  /*Pilihan yang benar*/
  printf("\nInputkan Bilangan pertama : ");
  scanf("%f",&bil);
  printf("\nInputkan Bilangan kedua : ");
  scanf("%f",&bil2);

  switch(x)
  {
case '1':
  hasil=bil1+bil2;
  ket='+';
  break;
case '2':
  hasil=bil1-bil2;
```

```
      ket='-';
      break;
    case '3':
      hasil=bil1*bil2;
      ket='*';
      break;
    case '4':
      hasil=bil4:bil2;
      ket=:';
      break;
      }
      /*Cetak hasil*/
      print("\n\n%f %c %f = %f",bil1,ket,bil2,hasil);
      printf("\n\nTekan sembarang tombol");
      getch();
      clrscr();
    }
  }
```

5. Alt+F, save dengan nama Lat4h.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Apa yang anda ketahui tentang iterasi? Jelaskan dengan singkat dan jelas !
2. Penyataan *do-while* dan *while* mempunyai kesamaan yaitu sama-sama mengetes kondisi. Jelaskan perbedaan keduanya dan disertai dengan contoh program!
3. Jelaskan tentang kinerja pernyataan *break* dan continue yang dilibatkan dalam struktur perulangan!
4. Apa yang kamu ketahui tentang label? Jelaskan dan disertai pula dengan contoh programnya !
5. Apa guna dari fungsi *exit()* ?

- sedang pada struktur WHILE adalah sebaliknya. Di antara struktur tersebut kita dapat memilih struktur mana yang tepat untuk menyelesaikan masalah kita. Apabila masalah kita dapat diselesaikan dengan beberapa struktur yang ada, maka kita tinggal memilih struktur mana yang lebih simple atau lebih sederhana.

BAB 5

FUNGSI

I. TUJUAN

1. Melakukan pemanggilan fungsi dengan nilai dan referensi.
2. Menggolongkan variabel berdasarkan kelas penyimpanan.
3. Menerapkan fungsi rekursi

II. DASAR TEORI

Fungsi merupakan bagian (blok) dari kode program yang dirancang untuk melaksanakan tugas tertentu. Fungsi merupakan bagian penting dalam pemrograman C, dengan tujuan:

1. Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan.
2. Dapat mengurangi pengulangan kode (duplikasi kode).

Ada beberapa contoh fungsi standar yang sudah pernah kita gunakan dan memiliki tugas khusus, seperti: printf(),getch(), puts() dan masih banyak lagi fungsi lain yang bisa kita pakai sesuai tugasnya. Disamping itu kita juga dapat membuat fungsi sendiri sesuai dengan kebutuhan kita. Dalam bekerja dengan fungsi kita memerlukan masukan yang dinamakan sebagai argumen dan parameter. Masukan ini selanjutnya diolah oleh fungsi dan hasil keluarannya dapat berupa sebuah nilai (nilai balik fungsi). Bentuk umum sebuah fungsi adalah sebagai berikut:

```
Penentu_tipe nama_fungsi (daftar parameter)
Deklarasi parameter
{
    Tubuh fungsi
}
```

Penentu tipe digunakan untuk menentukan tipe keluaran fungsi, defaultnya int.

a) Melewatkan Parameter

Ada 2 cara melewatkan parameter ke dalam fungsi :

1. Pemanggilan dengan nilai *(call by value)*
2. Pemanggilan dengan referensi *(call by reference)*

Call by Value	Call by Reference
Pemanggilan dengan nilai, nilai dari parameter aktual akan disalin ke parameter formal. Dengan cara ini nilai parameter aktual tidak bisa berubah sekalipun nilai parameter formal berubah.	Pemanggilan dengan referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel di luar fungsi dengan pelaksanaan pengubahan dilakukan di dalam fungsi

b) Penggolongan Variabel Berdasarkan Kelas Penyimpanan

Suatu variabel disamping dapat digolongkan berdasarkan jenis/tipe data juga dapat diklasifikasikan berdasarkan kelas penyimpanan berupa :

- Variabel lokal
- Variabel eksternal
- Variabel statis
- Variabel register

Variabel lokal adalah variabel yang dideklarasikan dalam fungsi.

Contoh:

```
Void fung_x(void)
{
  Int x;
  .....
  .....
}
```

x adalah variabel lokal bagi fung_x()

variabel eksternal merupakan variabel yang dideklarasikan di luar fungsi. Variabel statis dapat berupa variabel internal (didefinisikan di dalam fungsi) maupun variabel eksternal.

Contoh :

```
Staticz;  /* statis eksternal */

Voidfung_y(void)

{

  Staticint y; /* statis internal */

  ……..

  …….

}
```

Variabel register adalah variabel yang nilainya disimpan dalam register bukan pada memori RAM. Diterapkan pada variabel yang lokal atau parameter formal, yang bertipe char atau int.

III. LANGKAH KERJA

a. Langkah pemanggilan dengan nilai

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F→ New
4. Ketik :

```
/* ------------------------------------*
 * Program Pr8_1.C             *
 * Dibuat oleh : Raras              *
 * Tanggal     : 06 November 2011   *
 * Topik       : pemanggilan            *
 *      dg nilai     *
 * ------------------------------------*/
 #include<stdio.h>
 #include<conio.h>

 void tukar (int, int);

 main()
 {
  int a, b;

   a = 88;
   b = 77;

   printf("nilai sebelum pemanggilan fungsi\n");
   printf("a = %d b = %d\n\n", a, b);

   tukar(a, b);

   printf("nilai sesudah pemanggilan fungsi\n");
   printf("a = %d b = %d\n\n", a, b);
   getch();
```

```
}

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

5. Alt+F, save dengan nama Lat5a.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

b. Langkah melakukan pemanggilan dengan referensi :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F→ New
4. Ketik :

```
/* -------------------------------------*
 * Program Pr8_2.c             *
 * Dibuat oleh : Raras              *
 * Tanggal     : 06 November 2011   *
 * Topik       : pemanggilan        *
 *      dg referensi      *
 * -------------------------------------*/
 #include <stdio.h>
 #include <conio.h>

 void tukar (int *px, int *py);

 main()
 {

  int a, b;

  a = 88;
  b = 77;

  printf("nilai sebelum pemanggilan fungsi\n");
  printf("a = %d b = %d\n\n", a, b);

  tukar(&a, &b); /*parameter : alamat a dan alamat b*/

  printf("nilai sesudah pemanggilan fungsi\n");
  printf("a = %d b = %d\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 = %d *py = %d\n\n", *px, *py);
}
```

5. Alt+F, save dengan nama Lat5b.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

c. Langkah menghasilkan fungsi rekursi (fungsi yang memanggil dirinya sendiri)

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F→ New
4. Ketik :

```
/* ------------------------------------ *
 * Program Pr8_3.c             *
 * Dibuat oleh : Raras              *
 * Tanggal     : 06 November 2011   *
 * Topik       : fungsi rekursi     *
 *      pada faktorial         *
 * ------------------------------------ */
#include <stdio.h>
#include <conio.h>
int faktorial (int);

main()
{
 int x;

 puts("MENCARI FAKTORIAL DARI X!");
 printf("Masukkan nilai x (bulat positip) : ");
 scanf("%d", &x);
 printf("Faktorial dari %d = %d\n", x, faktorial (x));
 getch();
}

int faktorial (int m)    /*definisi fungsi*/
{
```

```
if( m == 1 )
  return(1);
else
  return(m * faktorial(m-1));
```

5. Alt+F, save dengan nama Lat5c.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C →compile to OBJ
8. Alt+C → compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Apa yang dimaksud dengan call by value dan call by reference?
2. Jelaskan tentang penggolongan variabel berdasarkan kelas penyimpanan!
3. Buat program menggunakan fungsi untuk mnyelesaikan soal berikut dan simpan dalam file lat5d.c :

   Menampilkan menu untuk menghitung :

   1) Rumus Phytagoras
   2) Rumus abc
   3) Keluar

   Gunakan struktur case dan masing-masing perhitungan dibuat fungsinya sehingga bisa dipanggil dengan kode program di dalam struktur case!

BAB 6

PENGENALAN ARRAY

I. TUJUAN

1. Mengetahui dan memahami tentang larik *(array)* dalam bahasa C.
2. Mendeklarasikan dan mengakses elemen larik.
3. Membuat program yang melibatkan permasalahan-permasalahan yang membutuhkan pengaman tipe data menggunakan larik.

II. DASAR TEORI

Dalam beberapa literatur, array sering disebut atau diterjemahkan sebagai larik. Array merupakan koleksi data dengan setiap elemen data menggunakan nama yang sama dan masing-masing lemen data bertipe sama. Setiap elemen array dapat diakses dan dibedakan melalui indeks array. Adapun array sendiri digolongkan menjadi :

1. array berdimensi satu,
2. array berdimensi dua,
3. array berdimensi banyak, dan
4. array tak berukuran.

1. Array Berdimensi Satu

a. Bentuk umum deklarasi array berdimensi satu

```
nama_array : array [ukuran] of tipe
```

dengan :

tipe : untuk menyatakan jenis elemen array (misal char, int, float).

ukuran : untuk menyatakan jumlah maksimal elemen array

Contoh :

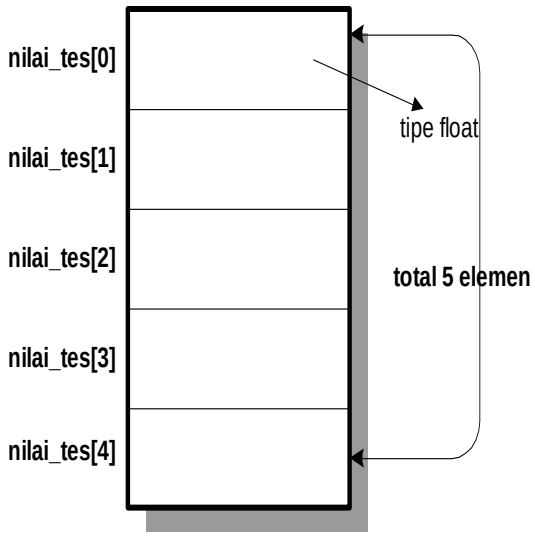
```
nilai tes : array[1..5] of float
```

Deklarasi di atas menyatakan bahwa array **nilai tes** mengandung 5 elemen bertipe **float**.

CATATAN :

Elemen aaray akan disimpan dalam memori secara berurutan dengan indeks pertama 0 (nol), elemen kedua nomor indeksnya 1 (satu), dan seterusnya.

b. Mengakses elemen array



c. Inisialisasi array berdimensi satu

Contoh :

```
static int jum_hari[12] =
                        { 31, 28, 31, 30, 31, 30 31, 31, 30, 31, 30, 31 }
```

Contoh di atas merupakan instruksi untuk mendeklarasikan array jum_hari yang bersifat statis dan sekaligus melakukan inisialisasi (pemberian nilai) pada masing-masing elemen array. Adapun perintah di atas berarti :

jum_hari[0] bernilai 31

jum_hari[1] bernilai 28

jum_hari[2] bernilai 31

dan seterusnya.

2. <u>Array Berdimensi Dua</u>

a. Bentuk umum deklarasi array berdimensi dua

tipe nama_var[ukuran1][ukuran2]

dengan :

tipe : untuk menyatakan jenis elemen array (misal char, int, float)

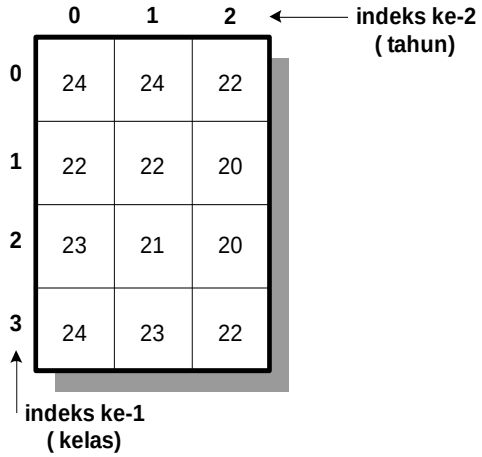
ukuran 1,2 : untuk menyatakan jumlah maksimal elemen array.

Contoh : tabel data jumlah siswa kelas 1 :

Kelas	Tahun I	Tahun II	Tahun III
IK 1A	24	24	22
IK 1B	22	22	20
IK 1C	23	21	20
IK 1D	24	23	22

b. Pengaksesan elemen array

Data tersimpan dalam array berdimensi dua :



cara mengakses array berdimensi dua adalah dengan bentuk berikut :

data_siswa[indeks ke-1][indeks ke-2]

c. Inisialisasi array berdimensi dua

Contoh :

```
Static int data_siswa[4][3] =
                              { 24, 24, 22,
                                22, 22, 20,
                                23, 21, 20,
                                24, 23, 22 };
```

Pengaksesan data array di atas dapat dilakukan dengan perintah sebagai berikut :

data_siswa[0][0] bernilai 24
data_siswa[0][1] bernilai 24
data_siswa[0][2] bernilai 22
data_siswa[1][0] bernilai 22
dan seterusnya.

3. <u>Array Berdimensi Banyak</u>

Bahasa C memungkinkan untuk membuat array yang dimensinya lebih dari dua. Bentuk umum pendeklarasian array berdimensi banyak adalah sebagai berikut :

tipe nama_var[ukuran1] [ukuran2]....[ukuran n]

dengan :

tipe : untuk menyatakan jenis elemen array (misal char, int, float).

ukuran 1,2..n : untuk menyatakan jumlah maksimal elemen array.

Contoh deklarasi :

int data_matrik[

```
data_siswa =

array[1..4, 1..3] of
integer
```

contoh inisialisasi array berdimensi banyak :

```
static int data_matrik[2][4][4] =
                                { 10, 20, 15, 20,
                                  12, 30, 22, 35,
                                  21, 45, 32, 50,
                                  45, 28, 12, 55,
                                  90, 60, 65, 75,
                                  72, 25, 80, 26,
                                  99,  35,  62,  78,
                                  33, 66, 88, 99 }
```

atau dapat juga ditulis

```
static int data_matrik[2][4][4] =
                              {{{  10, 20, 15, 20 },
                                 { 12, 30, 22, 35 },
                                 { 21, 45, 32, 50 },
                                 { 45, 28, 12, 55 },
                               },
                                {{ 90, 60, 65, 75 },
                                 { 72, 25, 80, 26 },
                                 { 99, 35, 62, 78 },
                                 { 33, 66, 88, 99 }}};
```

4. **Array tak berukuran**

Inisialisasi array ta berukuran dapat dilakukan untuk aray berdimensi satu atau lebih. Untuk array yang berdimensi lebi dari satu, hanya dimensi terkiri yang boleh tak berukuran. Dengan cara ini tabel dalam array dapat diperluas atau dikurangi tanpa mengubah ukuran array. Sebagai contoh :

```
int skala[ ] = {1, 2, 4, 6, 8 };
```

merupakan pendeklarasian array berdimensi satu yang tak berukuran, maka secara otomatis :
skala [0] bernilai 1

skala [1] bernilai 2
skala [2] bernilai 4
dan seterusnya

contoh lain:

```
char konversi[][2] = {   'A', 'T'
                         'E', 'M'
                         'I', 'V'
                         'O', 'S'
                         'U', 'J'        };
```

Contoh array di atas berarti :

konversi[0][0] bernilai 'A'
konversi[0][1] bernilai 'T'
konversi[1][0] bernilai 'E'
.....

III. LANGKAH KERJA

a. Langkah untuk menentukan rata-rata nilai tes siswa :

10. Start → Program → Accesories → Command Prompt
11. Jalankan turbo C
12. Alt+F → New
13. Ketik :

```
/*-------------------------------------------------  *
 * Program Lat10a.c                                 *
 * Contoh pemakaian array berdimensi satu           *
 * Untuk menghitung rata rata nilai siswa           *
 *-------------------------------------------------  */

 #include <stdio.h>
 #define MAKS_TES 5
 main ()
 {
 int i;
 float total_nilai, rerata;
 float nilai_tes[MAKS_TES];

 clrscr ();
 for (i=0; 1<MAKS_TES; i++)
    {
     printf ("Nilai tes ke- %d: ", i+1);
     scanf ("%f", &nila_tes [i]);
    }
 total_nilai = 0
 for (i=0; i<MAKS_TES; i++)
    total_nilai += nilai_tes[i];

 rerata = total_nilai / MAKS_TES;
```

```
printf ("Nilai rata rata = %.2f", rerata);
getch();
}
```

14. Alt+F, save dengan nama Lat6a.c
15. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
16. Alt+C → compile to OBJ
17. Alt+C → compile to EXE
18. Alt+R

b. Langkah menentukan jumlah hari dari bulan dalam setahun :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/*-----------------------------------------------*
 * Program Lat10b.C                              *
 * Contoh inisialisasi arra berdimensi satu   *
 * Untuk memperoleh jumlah hari suatu bulan   *
 *-----------------------------------------------*/

 #include <stdio.h>
 main ()
 {
  static jum_hari[12] =
       { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
  int bulan, tahun, jhari;

  clrscr();

puts("=====================================================");
puts(" 1 = JANUARI 2 = FEBRUARI 3 = MARET    4 = APRIL
    ");
puts(" 5 = MEI       6 = JUNI     7 = JULI     8 = AGUSTUS  ");
puts(" 9 = SEPTEMBER 10 = OKTOBER 11 = NOVEMBER   12 =  DESEMBER
");
puts("=====================================================");
);

  printf("Masukan bulan (1..12)     : ");
  scanf("%d", &bulan);
  printf("Masukan tahun        : ");
  scanf("%d", &tahun);

  if (bulan == 2)
    if((tahun % 4 == 0) && (tahun % 100 != 0))
      jhari = 29;
    else
      jhari = 28;
  else
    jhari = jum_hari[bulan];
```

```
  printf(" Jumlah hari bulan %d tahun %d = %d", bulan, tahun,
jhari);

 }
```

5. Alt+F, save dengan nama Lat6b.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C →compile to EXE
9. Alt+R

c. Langkah penjumlahan dua buah matrik :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C
3. Alt+F → New
4. Ketik :

```
/*-----------------------------------------*
 *Program Lat10c.c                         *
 *contoh penggunaan array berdimensi dua        *
 *Untuk menjumlahkan dua buah array        *
 *-----------------------------------------*/
 #include<stdio.h>
 #define maks_kolom 5
 #define maks_baris 5

 /*prototipe fungsi*/
 void entri_data_matrik(int matriks[][], int, int);
 void jumlah_dua_matrik(int matriks1[][],int matriks2[][],
       int mat_hasil[][], int, int);
 void tampil_data_matrik(int matriks[][], int, int);

 main()
 {
 int jum_kolom,jum_baris;
 int mat1[maks_baris][maks_kolom],/*deklarasi array dimensi 2*/
     mat2[maks_baris][maks_kolom],
     mat_hasil[maks_baris][maks_kolom];

 clrscr();
 puts("=====================================================");
 puts("       Operasi penjumlah dua buah matrik            ");
 puts("-----------------------------------------------------");
 printf("Banyak baris dari elemen matrik(...%d):",maks_baris);
 scanf("%d", &jum_baris);
 printf("Banyak kolom dari elemen matrik(...%d):",maks_kolom);
 scanf("%d", &jum_kolom);

 puts("\nData Matrik 1\n");
 entri_data_matrik(mat1,jum_baris,jum_kolom);
 puts("\nData Matrik 2\n");
 entri_data_matrik(mat2,jum_baris,jum_kolom);
```

```
 puts("\nPenjumlahan Dua Matrik\n");
 jumlah_dua_matrik(mat1,mat2,mat_hasil,jum_baris,jum_kolom);
 puts("\nPenampilan hasil jumlahan Dua Matrik\n");
 tampil_data_matrik(mat_hasil,jum_baris,jum_kolom);
 getche();
 }

 void      entri_data_matrik(int      matriks[][maks_kolom],int
jum_baris,int jum_kolom)
 {
  int i, j;
   for(i=0 ; i<jum_baris;i++)
    for(j=0 ; j<jum_kolom;j++)
      {
      printf("Data baris %d kolom %d :",i+1,j+1);
      scanf("%d",&matriks[i][j]);
      }
 }

 void      jumlah_dua_matrik(int      matriks1[][maks_kolom],int
matriks2[][maks_kolom],
        int   mat_hasil[][maks_kolom],   int   jum_baris,   int
jum_kolom)
  {
  int i, j;
  for(i=0; i<jum_baris; i++)
   for(j=0; j<jum_kolom; j++)
    mat_hasil[i][j]=matriks1[i][j]+ matriks2[i][j];
  }

 void    tampil_data_matrik(int    matriks[][maks_kolom],    int
jum_baris, int jum_kolom)
  {
  int i, j;
  for(i=0;i<jum_baris;i++)
   {
   for(j=0;j<jum_kolom;j++)
   printf("%7d",matriks[i][j]); puts(" ");
   }
  }
```

5. Alt+F, save dengan nama Lat6c.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C →compile to EXE
9. Alt+R

d. Langkah program penyandian data :

1. Start → Program → Accesories → Command Prompt
2. Jalankan turbo C

3. Alt+F → New
4. Ketik :

```
/*-----------------------------------------*
 *Program Lat10c.c                         *
 *contoh penggunaan array berdimensi dua    *
 *Untuk menjumlahkan dua buah array         *
 *-----------------------------------------*/
 #include<stdio.h>
 #define ENTER 13
 #define jum_kolom 2
 main()
 {
 static char konversi[][jum_kolom]={'A','T',
                     'a','t',
                     'E','M',
                     'e','m',
                     'I','V',
                     'i','v',
                     'O','S',
                     'o','s',
                     'U','J',
                     'u','j'};
 char kalimat[256],karakter;
 int i=0,j=0, jum_kar,jum_sandi;
   clrscr();

puts("============================================================
======");
   puts("  Masukkan sebuah kalimat dan akhiri dengan ENTER !
");
   puts("     Maka kalimat tersebut akan saya sandikan     ");
   puts("-----------------------------------------------------
---------");
   /*proses memasukkan kalimat*/
   while((kalimat[i]=getche())!=ENTER)
  i++;

   /*sandikan dan tampilkan kelayar*/
   puts("\n\nHasil penyandian:");
   jum_sandi=sizeof(konversi)/(jum_kolom*sizeof(char));
   for(i=0;i<jum_sandi;i++)
    {
    karakter=kalimat[i];
    for(j=0;j<jum_sandi;j++)
       {
       if(karakter==konversi[j][0])
   {
    karakter=konversi[j][1];
    break;/*keluar dari for terdalam*/
   }
       if(karakter==konversi[j][1])
   {
    karakter=konversi[j][0];
```

```
        break;/*keluar dari for terdalam*/
       }
           }
        putchar(karakter);
        }
     puts(" ");
   }
```

5. Alt+F, save dengan nama Lat6d.c
6. Alt+O, pada directories change semua menjadi C:\TC,lalu esc
7. Alt+C → compile to OBJ
8. Alt+C →compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Berikan ulasan terhadap program lat6c.c, tentang model array yang dipakai, proses pelewatan array sebagai parameter dan kinerja program tersebut.
2. Buatlah program yang dapat digunakan mengalikan atau mengurangkan dua buah matrik.
3. Suatu array dalam fungsi main() dideklarasikan sebagai berikut :
   static int grafik[8] = {10, 5, 8, 12, 15, 20, 25, 30};
   berdasarkan array di atas, nuatlah program yang menghasilkan tampilan sebagai berikut :

```
10      |   **********
 5      |   *****
 8      |   ********
12      |   ************
15      |   ***************
20      |   ********************
25      |   *************************
30      |   ******************************
```

4. Berikanlah kesimpulan dari praktik yang telah anda lakukan!

BAB 7

STRUKTUR

I. TUJUAN

- ✓ Mengetahui dan memahami tentang perintah untuk membuat struktur dalam bahasa C.
- ✓ Menjelaskan tentang bentuk umum atau struktur dari perintah-perintah yang terkait dengan struktur.
- ✓ Membuat program yang melibatkan permasalahan-permasalahan yang membutuhkan struktur.

II. DASAR TEORI

Struktur

1 Pendefinisian, Pendeklarasian, dan Pengaksesan Struktur

Struktur adalah koleksi dari variabel yang dinyatakan dengan sebuah nama, dengan sifat setiap variabel dapat memiliki tipe yang berlainan. Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah kesatuan. Struktur dapat digambarkan seperti berikut :

Data	.....	.....	Data
Data	.....	.....	Data
Data	.....	.....	Data
Data	.....	.....	Data

Field1 Field2 ..... Field_n

Suatu struktur dideklarasikan dengan menggunakan format sebagai berikut :

```
struct nama_struktur
   {
  tipe NamaField_1;
  tipe NamaField_2;
  ..... . . … . .
  tipe NamaField_n;
   };
```

Contoh :

```
struct data_tunggal
 {
   int tanggal;
   int bulan;
   int tahun;
  };
```

Untuk mendeklarasikan dengan menggunakan format sebagai berikut :

```
struct data_tanggal tgl_lahir
```

keterangan :

- `data_tanggal` : nama tipe struktur
- `tgl_lahir` : variabel struktur

Cara lain untuk pendefinisian dan pendeklarasian struktur dapat dialkukan dengan cara sebagai berikut :

```
struct data_tanggal
     {
      int tanggal;
      int bulan;
      int tahun;
   }
```

Keterangan : tgl_lahir merupakan variabel struktur

Suatu struktur juga dapat digunakan dalam pendefinisian suatu struktur lain.
Contoh :

```
struct data_tanggal
   {
      int tanggal;
      int bukan;
      in tahun;
   }

struct data_mhs
 {
  char nama[20];
  struct data_tanggal tgl_lahir;
 }info mhs;
```

Keterangan :

Pada contoh di atas, variabel info_mhs bertipe struktur data_mhs yang mempunyai field :

- nama
- tgl_lahir

sedangkan tgl_lahir sendiri bertipe struktur data_tanggal, yang mempunyai field :

- tanggal
- bulan
- tahun

Suatu elemen data struktur dapat diakses dengan bentuk umum sebagai berikut :

Variabel_struktur.nama_filed

Antara variabel_struktur dengan nama_field dipisahkan dengan tanda titik (.)
Contoh :

```
info_mhs.tgl_lahir.tanggal = 30; /*mengisikn nilai 30 ke field tanggal*/
tgl = info_mhs.tgl_lahir; /*memperoleh isi field tanggal*/
puts(info_mhs.nama); /*menampilkan isi field nama*/
```

2 Inisialisasi Struktur

Suatu struktur juga bisa diinisialisasikan pada saat pendeklarsiannya. Contoh deklarsi struktur berikut melibatkan proses inisialisasi pada saat pendeklarsiannya.

```
struct zodiak
{ char nama[11]
  int tgl_awal;
  int bln_awal;
  int tgl_akhir;
  int bln_akhi;
};

struct zodiak bintang =

{"Aquarius", 20, 1, 18, 2};
```

3 Array dan Struktur

Penggunaan struktur sering dikaitkan dengan array membentuk array dari struktur. Contoh array dari struktur dapat diilustrasikan sebagai berikut :

```
# define maks 20
..... . .
```

.....

```
struct data_mhs larik_mhs[maks];
```

Setelah array larik_mhs dideklarasikan maka ruang yang disediakan sperti ilustrasi gambar berikut :

III. LANGKAH KERJA

a. Menyimpan Data Dalam Array :

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/*-------------------------------------------*
 * Program Lat11a.c                          *
 * contoh array struktur                     *
 * yang digunakan untuk memegang data mhs        *
 *-------------------------------------------*/
 #include <stdio.h>
 #include <string.h>
 #include <ctype.h>
 #include <conio.h>

 #define MAKS 20

 main()
 {
 struct data_tanggal /*definisi tipe data tanggal*/
   { int tanggal;
     int bulan;
     int tahun;
   };
 struct data_mhs /*definisi tipe data_mhs*/
   { char nama [21];
     struct data_tanggal tgl_lahir;
   };
 struct data_mhs info_mhs[MAKS]; /*deklarasi array struktur*/
 char tombol;
 int i, jum_mhs = 0;

   /*pemasukan data ke array struktur*/
   clrscr();
   puts("DATA MHS MAHASISWA:\n");
   do
      {
      printf("Nama        :");
      gets(info_mhs[jum_mhs].nama);
      printf("Tanggal lahir (XX-XX-XXXX) :");
      scanf("%d-%d-%d", &info_mhs[jum_mhs].tgl_lahir.tanggal,
            &info_mhs[jum_mhs].tgl_lahir.bulan,
            &info_mhs[jum_mhs].tgl_lahir.tahun);
      fflush(stdin);/* hapus sisa data dalam penampung keyboard
*/
```

```
      jum_mhs++;

      printf("\nMau memasukkan lagi (Y/T)? ");
      tombol= toupper(getch());
      while (!(tombol == 'T' || tombol == 'Y'))
       tombol = toupper(getch());
      printf("%c\n\n", tombol);
      }
   while (tombol == 'Y');

   /* penampilan elemen variabel struktur */
   puts("\nData mhs: NAMA-TANGGAL LAHIR\n");
   for(i=0; i<jum_mhs; i++)
      printf("%-21s %d-%d-%d\n", info_mhs[i].nama,
                  info_mhs[i].tgl_lahir.tanggal,
                  info_mhs[i].tgl_lahir.bulan,
                  info_mhs[i].tgl_lahir.tahun);
   getche();
}
```

5. Alt+F, pilih save ketik nama lat7a.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Menentukan Zodiak Berdasarkan Data Tanggal Lahir :

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/*-----------------------------------------------------*
 * Program ZODIAK.C                                   *
 * contoh inisialisasi terhadap variabel struktur     *
 *-----------------------------------------------------*/
 #include<stdio.h>

 main()
 {
 struct zodiak
      { char nama[11];
  int tgl_awal;
  int bln_awal;
  int tgl_akhir;
  int bln_akhir;
 };

 struct zodiak bintang=
 { "Sagitarius", 23, 11, 20, 12 };

 int tg_lhr, bl_lhr, th_lhr;
```

```
clrscr();
printf("Tanggal lahir anda (XX-XX-XXXX): ");
scanf("%d-%d-%d", &tg_lhr, &bl_lhr, &th_lhr);

if ( (tg_lhr >= bintang.tgl_awal &&
   bl_lhr == bintang.bln_awal ) |!
   ( tg_lhr <= bintang.tgl_akhir &&
      bl_lhr == bintang.bln_akhir ))
     printf(" Bintang anda bukanlah %s\n", bintang.nama);
else
   printf(" Bintang anda bukanlah %s\n", bintang.nama);
   getche();
}
```

5. Alt+F, pilih save ketik nama lat7b.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

Pengembangan 7b.c

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>

main()
{
  struct zodiak /* definisi tipe struktur */
       {
       char nama[11];
       int tgl_awal;
       int bln_awal;
       int tgl_akhir;
       int bln_akhir;
       };

  struct zodiak bintang[] =
       {{"CAPRICORN" ,22, 12, 20,  1},
         {"AQUARIUS" ,21,  1, 19,  2},
         {"PISCES"   ,20,  2, 20,  3},
         {"ARIES"    ,21,  3, 19,  4},
         {"TAURUS"   ,20,  4, 20,  5},
         {"GEMINI"   ,21,  5, 21,  6},
         {"CANCER"   ,22,  6, 23,  7},
         {"LEO" ,24,  7, 23,  8},
         {"VIRGO"    ,24,  8, 23,  9},
         {"LIBRA"    ,24,  9, 22, 10},
```

```
        {"SCORPIO"  ,23, 10, 22, 11},
        {"SAGITARIUS"    ,23, 11, 21, 12},
      };
  int i, tg_lhr, bl_lhr, th_lhr;

  printf("Tanggal lahir Anda (XX-XX-XXXX): ");
  scanf("%d-%d-%d", &tg_lhr, &bl_lhr, &th_lhr);

  for (i=0; ;i++)
    if( (tg_lhr >= bintang[i].tgl_awal &&
       bl_lhr == bintang[i].bln_awal ) ||
          (tg_lhr <= bintang[i].tgl_akhir &&
       bl_lhr == bintang[i].bln_akhir) )
      break;
  printf("Bintang Anda adalah %s\n", bintang[i].nama);
}
```

5. Alt+F, pilih save ketik nama lat7c.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

Pengembangan 7e.c

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
 main()
 {
  struct zodiak
  {
   char nama[11];int tgl_awal;
   int bln_awal;int tgl_akhir;int bln_akhir;
  };
  struct zodiak btg1={"Capricorn",22,12,20,1};
  struct zodiak btg2={"Aquarius",21,1,19,2};
  struct zodiak btg3={"Pisces",20,2,20,3};
  struct zodiak btg4={"Aries",21,3,19,4};
  struct zodiak btg5={"Taurus",20,4,20,5};
  struct zodiak btg6={"Gemini",21,5,21,6};
  struct zodiak btg7={"Cancer",22,6,23,7};
  struct zodiak btg8={"Leo",24,7,23,8};
  struct zodiak btg9={"Virgo",24,8,23,9};
  struct zodiak btg10={"Libra",24,9,22,10};
  struct zodiak btg11={"Scorpio",23,10,22,11};
  struct zodiak btg12={"Sagitarius",23,11,21,12};

  int tg_lhr,bl_lhr,th_lhr;
  clrscr();
  puts("\t\t\tRAMALAN BINTANG");
```

```
  printf("\n\n\tTanggal Lahir Kamu [xx-xx-xxxx] : ");
scanf("%d-%d-%d",&tg_lhr,&bl_lhr,&th_lhr);
clrscr();
puts("\t\t\tRAMALAN BINTANG");
if((tg_lhr>=btg1.tgl_awal)&&(bl_lhr==btg1.bln_awal)||
     (tg_lhr<=btg1.tgl_akhir)&&(bl_lhr==btg1.bln_akhir))
    {
     printf("\n\tBintang Kamu adalah %s\n",btg1.nama);
     printf("\n\tKamu         lahir         pada         %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Keinginanmu tercapai bulan ini");
     puts("\n\tKesehatan : Kuncinya di jadwal dan waktu");
     puts("\n\tKeuangan   :   Hati-hati   dengan   proyek   dana
bersama");
     puts("\n\tCinta : Misunderstanding");
    }
  else
  if((tg_lhr>=btg2.tgl_awal)&&(bl_lhr==btg2.bln_awal)||
     (tg_lhr<=btg2.tgl_akhir)&&(bl_lhr==btg2.bln_akhir))
    {
     printf("\n\tBintang Kamu adalah %s\n",btg2.nama);
     printf("\n\t       Kamu       lahir       pada       %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
     puts("\n\tPeruntungan : Krisis percaya diri");
     puts("\n\tKesehatan : Prima");
     puts("\n\tKeuangan : Banyak uang");
     puts("\n\tCinta : Masih mau menuggu");
    }
  else
  if((tg_lhr>=btg3.tgl_awal)&&(bl_lhr==btg3.bln_awal)||
     (tg_lhr<=btg3.tgl_akhir)&&(bl_lhr==btg3.bln_akhir))
    {
     printf("\n\tBintang Kamu adalah %s\n",btg3.nama);
     printf("\n\t       Kamu       lahir       pada       %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Pelan tapi pasti");
     puts("\n\tKesehatan : Pertahankan pola makan sehat kamu");
     puts("\n\tKeuangan : Yang penting gak berhutang!");
     puts("\n\tCinta : Malu atau nyerah");

    }
  else
  if((tg_lhr>=btg4.tgl_awal)&&(bl_lhr==btg4.bln_awal)||
     (tg_lhr<=btg4.tgl_akhir)&&(bl_lhr==btg4.bln_akhir))
    {
     printf("\n\tBintang Kamu adalah %s\n",btg4.nama);
     printf("\n\tKamu         lahir         pada         %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Nggak konsisten");
     puts("\n\tKesehatan : Tubuh udah kasih alarm");
     puts("\n\tKeuangan : Kantong tebal");
     puts("\n\tCinta : Masih kikuk tapi kalian serasi");
    }
  else
  if((tg_lhr>=btg5.tgl_awal)&&(bl_lhr==btg5.bln_awal)||
     (tg_lhr<=btg5.tgl_akhir)&&(bl_lhr==btg5.bln_akhir))
    {
     printf("\n\tBintang Kamu adalah %s\n",btg5.nama);
```

```
     printf("\n\tKamu         lahir         pada         %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan  :  Mempersiapkan  sesuatu  yang  baru
");
     puts("\n\tKesehatan : Tidur tepat waktu");
     puts("\n\tKeuangan :Butuh budget banyak buat liburan");
     puts("\n\tCinta : Hunting season is on");
    }
  else
  if((tg_lhr>=btg6.tgl_awal)&&(bl_lhr==btg6.bln_awal)||
     (tg_lhr<=btg6.tgl_akhir)&&(bl_lhr==btg6.bln_akhir))
    {
     printf("\n\tBintang Kamu adalah %s\n",btg6.nama);
     printf("\n\tKamu         lahir         pada         %d-%d-
%d",tg_lhr,bl_lhr,th_lhr); puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Ada yang baru");
     puts("\n\tKesehatan : Butuh olahraga");
     puts("\n\tKeuangan : Butuh uang");
     puts("\n\tCinta : Adem ayem");
    }
else
if((tg_lhr>=btg7.tgl_awal)&&(bl_lhr==btg7.bln_awal)||
   (tg_lhr<=btg7.tgl_akhir)&&(bl_lhr==btg7.bln_akhir))
   {
    printf("\n\tBintang Kamu adalah %s\n",btg7.nama);
    printf("\n\tKamu         lahir         pada         %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
    puts("\n\n\tRamalan Minggu Ini : ");
    puts("\n\tPeruntungan : Labil");
    puts("\n\tKesehatan : Makan yang berserat");
    puts("\n\tKeuangan : Berbagi");
    puts("\n\tCinta : Curi pandang");
   }
else
if((tg_lhr>=btg8.tgl_awal)&&(bl_lhr==btg8.bln_awal)||
   (tg_lhr<=btg8.tgl_akhir)&&(bl_lhr==btg8.bln_akhir))
  {
   printf("\n\tBintang Kamu adalah %s\n",btg8.nama);
   printf("\n\tKamu lahir pada %d-%d-%d",tg_lhr,bl_lhr,th_lhr);
   puts("\n\n\tRamalan Minggu Ini : ");
   puts("\n\tPeruntungan : Wajar kalau bosan");
   puts("\n\tKesehatan : Gangguan tenggorokan");
   puts("\n\tKeuangan : Pas-pasan");
   puts("\n\tCinta : Plenty fish in the sea and one for you");
  }
else
if((tg_lhr>=btg9.tgl_awal)&&(bl_lhr==btg9.bln_awal)||
   (tg_lhr<=btg9.tgl_akhir)&&(bl_lhr==btg9.bln_akhir))
  {
  printf("\n\tBintang Kamu adalah %s\n",btg9.nama);
  printf("\n\tKamu lahir pada %d-%d-%d",tg_lhr,bl_lhr,th_lhr);
  puts("\n\n\tRamalan Minggu Ini : ");
  puts("\n\tPeruntungan : Good month");
  puts("\n\tKesehatan : Jaga pola makan");
  puts("\n\tKeuangan : Banyak pemasukan");
  puts("\n\tCinta : mood mirip roller coaster");
  }
else
if((tg_lhr>=btg10.tgl_awal)&&(bl_lhr==btg10.bln_awal)||
   (tg_lhr<=btg10.tgl_akhir)&&(bl_lhr==btg10.bln_akhir))
```

```
  {
   printf("\n\tBintang Kamu adalah %s\n",btg10.nama);
   printf("\n\tKamu lahir pada %d-%d-%d",tg_lhr,bl_lhr,th_lhr);
   puts("\n\n\tRamalan Minggu Ini : ");
   puts("\n\tPeruntungan : Banyak kegiatan");
   puts("\n\tKesehatan : Istirahat cukup");
   puts("\n\tKeuangan : Boros");
   puts("\n\tCinta : Belajar sabar");
  }
else
if((tg_lhr>=btg11.tgl_awal)&&(bl_lhr==btg11.bln_awal)||
   (tg_lhr<=btg11.tgl_akhir)&&(bl_lhr==btg11.bln_akhir))
  {
   printf("\n\tBintang Kamu adalah %s\n",btg11.nama);
   printf("\n\tKamu lahir pada %d-%d-%d",tg_lhr,bl_lhr,th_lhr);
   puts("\n\n\tRamalan Minggu Ini : ");
   puts("\n\tPeruntungan : Exciting month");
   puts("\n\tKesehatan : Ketagihan makanan enak");
   puts("\n\tKeuangan : Hemat dong");
   puts("\n\tCinta : Tunjukkin kalu suka");
  }
else
  {
   printf("\n\tBintang Kamu adalah %s",btg12.nama);
   printf("\n\tKamu lahir pada %d-%d-%d",tg_lhr,bl_lhr,th_lhr);
   puts("\n\n\tRamalan Minggu Ini : ");
   puts("\n\tPeruntungan : karma");
   puts("\n\tKesehatan : Fit");
   puts("\n\tKeuangan : Banyak godaan");
   puts("\n\tCinta : Tke it easy");
  }
getch();
}
```

5. Alt+F, pilih save ketik nama lat7d.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

Pengambangan 7d.c

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
 main()
 {
  struct zodiak
  {
   char nama[11];int tgl_awal;
   int bln_awal;int tgl_akhir;int bln_akhir;
  };
  struct zodiak btg1={"Capricorn",22,12,20,1};
  struct zodiak btg2={"Aquarius",21,1,19,2};
  struct zodiak btg3={"Pisces",20,2,20,3};
  struct zodiak btg4={"Aries",21,3,19,4};
```

```
  struct zodiak btg5={"Taurus",20,4,20,5};
  struct zodiak btg6={"Gemini",21,5,21,6};
  struct zodiak btg7={"Cancer",22,6,23,7};
  struct zodiak btg8={"Leo",24,7,23,8};
  struct zodiak btg9={"Virgo",24,8,23,9};
  struct zodiak btg10={"Libra",24,9,22,10};
  struct zodiak btg11={"Scorpio",23,10,22,11};
  struct zodiak btg12={"Sagitarius",23,11,21,12};

  int tg_lhr,bl_lhr,th_lhr;
  clrscr();
  puts("\t\t\tRAMALAN BINTANG");
  printf("\n\n\tTanggal Lahir Kamu [xx-xx-xxxx] : ");
scanf("%d-%d-%d",&tg_lhr,&bl_lhr,&th_lhr);
clrscr();
puts("\t\t\tRAMALAN BINTANG");
if((tg_lhr>=btg1.tgl_awal)&&(bl_lhr==btg1.bln_awal)||
     (tg_lhr<=btg1.tgl_akhir)&&(bl_lhr==btg1.bln_akhir))
    {
   printf("\n\nTanggal lahir kamu adalah : %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
     printf("\n\tBintang Kamu adalah %s\n",btg1.nama);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Keinginanmu tercapai bulan
ini");
     puts("\n\tKesehatan : Kuncinya di jadwal dan waktu");
   puts("\n\tKeuangan : Hati-hati dengan proyek dana
bersama");
   puts("\n\tCinta : Misunderstanding");
    }
  else
  if((tg_lhr>=btg2.tgl_awal)&&(bl_lhr==btg2.bln_awal)||
     (tg_lhr<=btg2.tgl_akhir)&&(bl_lhr==btg2.bln_akhir))
    {
   printf("\n\nTanggal lahir kamu adalah : %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
     printf("\n\tBintang Kamu adalah %s\n",btg2.nama);
     puts("\n\tPeruntungan : Krisis percaya diri");
     puts("\n\tKesehatan : Prima");
   puts("\n\tKeuangan : Banyak uang");
   puts("\n\tCinta : Masih mau menuggu");
    }
  else
  if((tg_lhr>=btg3.tgl_awal)&&(bl_lhr==btg3.bln_awal)||
     (tg_lhr<=btg3.tgl_akhir)&&(bl_lhr==btg3.bln_akhir))
    {
   printf("\n\nTanggal lahir kamu adalah : %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
     printf("\n\tBintang Kamu adalah %s\n",btg3.nama);
     printf("\t      Kamu      lahir      pada      %d-%d-
%d",tg_lhr,bl_lhr,th_lhr);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Pelan tapi pasti");
     puts("\n\tKesehatan : Pertahankan pola makan sehat
kamu");
   puts("\n\tKeuangan : Yang penting gak berhutang!");
   puts("\n\tCinta : Malu atau nyerah");

    }
  else
  if((tg_lhr>=btg4.tgl_awal)&&(bl_lhr==btg4.bln_awal)||
```

```
     (tg_lhr<=btg4.tgl_akhir)&&(bl_lhr==btg4.bln_akhir))
    {
   printf("\n\nTanggal  lahir  kamu  adalah  :  %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
     printf("\n\tBintang Kamu adalah %s\n",btg4.nama);
     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan : Nggak konsisten");
     puts("\n\tKesehatan : Tubuh udah kasih alarm");
   puts("\n\tKeuangan : Kantong tebal");
   puts("\n\tCinta : Masih kikuk tapi kalian serasi");
    }
  else
  if((tg_lhr>=btg5.tgl_awal)&&(bl_lhr==btg5.bln_awal)||
     (tg_lhr<=btg5.tgl_akhir)&&(bl_lhr==btg5.bln_akhir))
    {
   printf("\n\nTanggal  lahir  kamu  adalah  :  %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
     printf("\n\tBintang Kamu adalah %s\n",btg5.nama);

     puts("\n\n\tRamalan Minggu Ini : ");
     puts("\n\tPeruntungan  :  Mempersiapkan  sesuatu  yang
baru ");
     puts("\n\tKesehatan : Tidur tepat waktu");
   puts("\n\tKeuangan :Butuh budget banyak buat liburan");
   puts("\n\tCinta : Hunting season is on");
    }
  else
  if((tg_lhr>=btg6.tgl_awal)&&(bl_lhr==btg6.bln_awal)||
     (tg_lhr<=btg6.tgl_akhir)&&(bl_lhr==btg6.bln_akhir))
    {
   printf("\n\nTanggal  lahir  kamu  adalah  :  %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s\n",btg6.nama);
puts("\n\n\tRamalan Minggu Ini : ");
puts("\n\tPeruntungan : Ada yang baru");
puts("\n\tKesehatan : Butuh olahraga");
puts("\n\tKeuangan : Butuh uang");
puts("\n\tCinta : Adem ayem");
    }
else
if((tg_lhr>=btg7.tgl_awal)&&(bl_lhr==btg7.bln_awal)||
   (tg_lhr<=btg7.tgl_akhir)&&(bl_lhr==btg7.bln_akhir))
  {
   printf("\n\nTanggal  lahir  kamu  adalah  :  %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s\n",btg7.nama);
printf("\tKamu lahir pada %d-%d-%d",tg_lhr,bl_lhr,th_lhr);
puts("\n\n\tRamalan Minggu Ini : ");
puts("\n\tPeruntungan : Labil");
puts("\n\tKesehatan : Makan yang berserat");
puts("\n\tKeuangan : Berbagi");
puts("\n\tCinta : Curi pandang");
    }
else
if((tg_lhr>=btg8.tgl_awal)&&(bl_lhr==btg8.bln_awal)||
   (tg_lhr<=btg8.tgl_akhir)&&(bl_lhr==btg8.bln_akhir))
  {
   printf("\n\nTanggal  lahir  kamu  adalah  :  %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s\n",btg8.nama);
puts("\n\n\tRamalan Minggu Ini : ");
```

```
puts("\n\tPeruntungan : Wajar kalau bosan");
puts("\n\tKesehatan : Gangguan tenggorokan");
puts("\n\tKeuangan : Pas-pasan");
puts("\n\tCinta : Plenty fish in the sea and one for you");
  }
else
if((tg_lhr>=btg9.tgl_awal)&&(bl_lhr==btg9.bln_awal)||
   (tg_lhr<=btg9.tgl_akhir)&&(bl_lhr==btg9.bln_akhir))
  {
   printf("\n\nTanggal lahir kamu adalah : %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s\n",btg9.nama);
puts("\n\n\tRamalan Minggu Ini : ");
puts("\n\tPeruntungan : Good month");
puts("\n\tKesehatan : Jaga pola makan");
puts("\n\tKeuangan : Banyak pemasukan");
puts("\n\tCinta : mood mirip roller coaster");
  }
else
if((tg_lhr>=btg10.tgl_awal)&&(bl_lhr==btg10.bln_awal)||
   (tg_lhr<=btg10.tgl_akhir)&&(bl_lhr==btg10.bln_akhir))
  {
   printf("\n\nTanggal lahir kamu adalah : %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s\n",btg10.nama);
puts("\n\n\tRamalan Minggu Ini : ");
puts("\n\tPeruntungan : Banyak kegiatan");
puts("\n\tKesehatan : Istirahat cukup");
puts("\n\tKeuangan : Boros");
puts("\n\tCinta : Belajar sabar");
  }
else
if((tg_lhr>=btg11.tgl_awal)&&(bl_lhr==btg11.bln_awal)||
   (tg_lhr<=btg11.tgl_akhir)&&(bl_lhr==btg11.bln_akhir))
  {
   printf("\n\nTanggal lahir kamu adalah : %d-%d-%d",
tg_lhr, bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s\n",btg11.nama);
puts("\n\n\tRamalan Minggu Ini : ");
puts("\n\tPeruntungan : Exciting month");
puts("\n\tKesehatan : Ketagihan makanan enak");
puts("\n\tKeuangan : Hemat dong");
puts("\n\tCinta : Tunjukkin kalu suka");
  }
else
{
printf("\n\nTanggal lahir kamu adalah : %d-%d-%d", tg_lhr,
bl_lhr, th_lhr);
printf("\n\tBintang Kamu adalah %s",btg12.nama);
puts("\n\n\tRamalan Minggu Ini : ");
puts("\n\tPeruntungan : karma");
puts("\n\tKesehatan : Fit");
puts("\n\tKeuangan : Banyak godaan");
puts("\n\tCinta : Tke it easy");
 }
getch();
}
```

5. Alt+F, pilih save ketik nama lat7e.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc

7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Apa yang anda ketahui tentang data/informasi, filed, record, dan struktur ? jelaskan dengan singkat dan jelas !
2. Berikanlah penjelasan dari deklarasi struktur berikut ini :

```
struct hoby { char nama_hobby[20];
              char alsan[30]; }
struct mhs { char nama[20];
             int nim;
             struct hoby myhoby;
             car kelas[4]; } data_mhs;
```

3. Buatlah program dengan tipe struktur untuk merekam data identitas mahasiswa. Adapun field-fieldnya adalah sebagai berikut :
   - nama
   - nim
   - jurusan
   - prodi
   - kelas
   - alamat
   - hoby

   tampilkan datanya dalam bentuk tabel.
4. Buatlah kesimpulan dari praktik yang telah anda lakukan !

BAB 8

OPERASI I/O

I. TUJUAN

- ✓ Menjelaskan tentang konsep string.
- ✓ Menjelaskan operasi I/O pada string.
- ✓ Menjelaskan cara mengakses elemen string.
- ✓ Menjelaskan berbagai fungsi mengenai string.

II. DASAR TEORI

String merupakan bentuk data yang dapat digunaan untuk menampung dan memanipulasi data teks. Dalam bahasa C, string bukan merupakan tipe data tersendiri, namun merupakan jenis khusus dari tipe array. Tipe string dapat digunaan sebagai konstanta, yang ditulis dengan diawali dan diakhiri tanda petik ganda. Misalnya :

" Teknik Elektro"

Konstanta string seperti di atas disimpan dalam memori secara berurutan, dengan komposisi sebagai berikut:

memori rendah → memori tinggi

T	**e**	**k**	**n**	**i**	**k**		**E**	**l**	**e**	**k**	**t**	**r**	**o**

Setiap karakter akan menempati memori sebesar 1 byte dan byte terkahir otomatis akan berisi karakter NULL.

a) Variabel String

Variabel string digunakan untuk menyimpan data string. Misalnya :

```
char nama[15];
```

Contoh di atas merupakan instruksi untuk mendeklarasikan variable tipe string dengan panjang maksimal mengandung 15 karakter (termasuk karakter NULL). Deklarasi di atas sebenarnya alah deklarasi array bertipe char. Untuk memasukkan data string ke dalam suatu variabel dapat dilakukan dengan menggunakan instruksi gets(), dengan bentuk umum pemakaiannya adalah :

```
gets(nama_array);
```

Jika menggunakan statement scanf(), maka instruksinya akan menjadi:

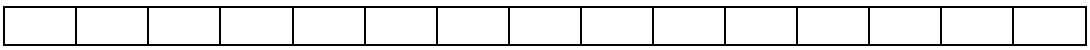
```
scanf("%",nama_array);
```

Di depan nama array tidak perlu ada operator &(operator alamat), karena nama array tanpa kurung siku sudah menaytakan alamat. Jika menggunakan scanf(), data string tidak bisa mengandung spasi. Prorotipe gets() terdapat pada file stdio.h.

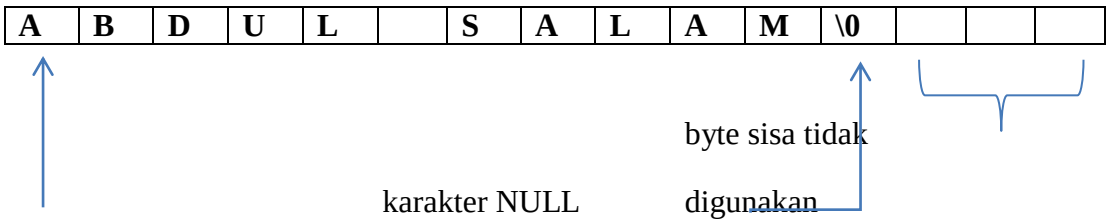
```
/* ------------------------------------------- */
/* File program : nama.c                       */
/* Contoh memasukkan data string dari keyboard   */
/* ------------------------------------------- */

#include <stdio.h>


main()
{
 char nama[15];
 printf("Nama anda : ");
 gets(nama) ;
  printf("Halo, %s. Selamat datang di Elektro.\n",nama);
}
```

Pada program di atas, setelah deklarasi char nama[15], maka komputer akan menyediakan uang pada memori sebanyak 15 arakter. Setelah pengguna memasukkan data nama : ABUL SALAM, maka data string yang dimasukkan akan diletakkan pada area memori yang sudah dipesan. Karena data yang dimasuukan kurang dari 15 katakter, maka otomatis setelah karakter terakhir akan diisi dengan karakter NULL.

ruang memori yang disediakan setelah : char nama[15]



Setelah pemasukkan data ABDUL SALAM



(menyatakan alamat dari lokasi data string)

Instruksi gets() akan membaca seluruh karakter yang diketik pada keyboard sampai tombol ENTER ditekan. Dalam ha ini tidak ada pengecekan terhadap batasan dari array yang merupakan argumennya. Jika string yang dimasuukan melebihi ukuran array, mak sisa string (panjang string dikurangi ukuran array plus karakter NULL) akan ditempatkan kelokasi sesudah bagian akhir array.

b) Inisiaisasi String

Suatu variabel string dapat diinisialisasi seperti hanya variabel array yang lain, namun pada elemen terakhir harus berupa karakter NULL. Sebagai contoh :

char kota[]=

{‘Y’, ‘o’, g‘’, y‘’, ‘a’, ‘k’, ‘a’, ‘r’, ‘t’, ‘a’, ‘\0’,};

Contoh di atas menyatakan baha variabel kota dalah variabel string dengan nilai awal berupa string “Yogyakarta”.

Bentuk inisialisasi yang lebih singkat adalah:

char kota[] = “Yogyakarta”;

Pada bentuk ini, karakter NULL tidak perlu ditulis, karena secara implisit akan disisipkan oleh kompiler.

c) Menampilkan Isi Variabel String Ke Layar

Untuk menampilkan isi variabel string, dapat digunakan salah satu dari pernyataan berikut:

- puts(var_string);
- printf(“%s”,var_string);
- printf(var_string);

Contoh program berikut ini akan menampilkan isi variabel kota, berdasarkan dua bentuk inisialiasi string.

```
/* ---------------------------------------- */
/* File program : kota.c                    */
/* Contoh menampilkan data string ke layar  */
/* ---------------------------------------- */
#include<stdio.h>
void bentuk1(void);
void bentuk2(void);

main()
{
bentuk1();
bentuk2();
}

void bentuk1(void)
{
char kota[]=
{'Y','o','g','y','a','k','a','r','t',a'','\0'};
puts(kota);
}

void bentuk2(void)
{
char kota[] = "Solo";
puts(kota);
}
```

d) Mengakses Elemen String

Variabel string merupakan bentuk khusus dari array bertipe char. Oleh karena itu, elemen dari variabel string dapat diakses seperti halnya pengaksesan elemen pada array.

Program berikut menunjukkan cara mengakses elemen array untuk menghitung total karakter dari string yang dimasukkan melalui keyboard.

```
/* ------------------------------------ */
/* File program : hitkar.c              */
/* Contoh menakses elemen string        */
/* ------------------------------------ */

#include<stdio.h>
#define maks 256

main()
{
int i, jumkar;
char teks[maks];
puts("Masukkan suatu kalimat.");
puts("Saya akan menghitung jumlah karakternya.");
gets(teks);

jumkar = 0
for(i=0;teks[i];i++)
jumkar++;
printf("Jumlah karakter = %d\n", jumkar);
}
```

Penghitungan jumlah karakter dari string teks dapat dilakukan dengan memeriks elemen dari string dimulai dari posisi yang pertama (indeks sama dengan 0) sampai ditemukannya karakter NULL. Elemen yang ke-I dari teks dinyatakan dengan :

teks[i]

Pemeriksaan terhadap teks[i] selama tidak berupa karakter NULL (dimulai dari indeks bernilai 0) dilakukan melalui:

for(i=0;teks[i];i++)
jumkar++;

Kondisi teks[i] pada for mempunai makan yang secara implisit berupa:

teks[i] != ’\0’

atau “karakter yang k-i dari teks tidak sama dengan karakter NULL”.

Contoh program berikut memperlihatkan cara penyalinan nilai ke suatu variabel string.

```
/* ------------------------------------ */
/* File program : SALINSTR.c            */
/* Contoh menyalin suatu string         */
/* ------------------------------------ */

#include<stdio.h>

main()
{
```

```
int i;
char keterangan[] = "Saya menyukai bahasa C";
char kalimat[30];

i = 0 ;
while(keterangan[i] !='\0')
{
kalimat[i] = keterangan[i];
i++;
}
kalimat[i] = '\0'; /*Beri karakter NULL*/

printf("Isi kalimat = %s\n", kalimat);
}
```

Selama keterangan[i] tidak berupa karakter NULL, maka keterangan[i] akan disalin ke kalimat [i]. Jadi dalam loop while tidak terdapat penyalinan karakter NULL dari keterangan ke kalimat. Oleh karenia itu eeluarnya dari loop while, pemberian karakter NULL ke kalimat perlu dilakukan. Bentuk yang lebih singkat untuk melakukan penyalinan eterangan ke kalimat berupa:

```
i=0
while (kalimat[i] = keterangan[i])
 i++;
```

Dengan penulisan seperti di atas, penyalinan arakter NULL juga akan dilakukan setelah menyalin karakter NULL (karena kondisi berbilai NILL) maka eksekusi terhadap loop akan dihentikan. Dengan deikian sekeluarnya dari loop tidak perlu ada pernyataan:

```
kalimat[i] = '\0'
```

e) Beberapa Fasilitas Untuk Operasi Karakter

Turbo C menyediakan sejumlah makro berargumen (semacam fungsi tetpi didefinisikan dengan menggunakan pengarah praprosesor #define) dan fungsi yang berkaitan dengan data karakter. Fasilitas ini sangat bermanfaat unutk keperluan pemanipulasian string. Beberapa diantaranta didefinisikan pada file judul (CTYPE.H, sebagai berikut):

isalnum()

Mempunyai bentuk:

```
int isalnum(int c);
```

Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf (huruf kapital maupun huruf kecil) atau berupa sebuah karakter angka (0 sampai dengan 9).

isalpaha()

Mempunyai bentuk:

int isalpaha(int c);

Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf (huruf kapital maupun huruf kecil)

isdigit()

Mempunyai bentuk:

int isdigit(int c);

Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah karakter angka (0 sampai dengan 9).

islower()

Mempunyai bentuk:

int islower(int c);

Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf kecil (‘a’ sampai dengan ‘z’)

isupper()

Mempunyai bentuk:

int isupper(int c);

Merupakan makro yang akan menghasilkan nilai benar (bukan nol) jika c adalah sebuah huruf kecil (‘A’ sampai dengan ‘Z’)

tolower()

Mempunyai bentuk:

int totlower(int c);

Jika c adalah huruf kapital,maka hasil fungsi berupa huruf kecilnya. Apabila c tidak berupa huruf kapital, keluaran fungsi sama dengan c.

toupper()

Mempunyai bentuk:

int toupper(int c);

Jika c adalah huruf kecil,maka hasil fungsi berupa huruf kapitalnya. Apabila c tidak berupa huruf kecil, keluaran fungsi sama dengan c.

strcpy()

Mempunyai bentuk:

strcpy(tujuan,asal);

Fungsi ini digunakan untuk menyalin variabel string asal ke variabel string tujuan. Dalam hal ini, variabel tujuan haruslah mempunyai ukuran yang dapat digunakan untuk menampung seluruh karakter dari string. Contoh :

strcpy(pepatah,”Kalau ada kemauan pasti ada jalan”);

merupakan instruksi untuk menyalin string ”Kalau ada kemauan pasti ada jalan” kevariabel string pepatah.

strlen()

Mempunyai bentuk:

strlen(var_string);

Digunakan untuk memperoleh julah karakter dalam variabel string yang merupakan argumennya. Karakter NULL tidak ikut dihitung.

strcat()

Mempunyai bentuk:

strcat(tujuan,sumber);

fungsi ini diggunakan untuk menambahkan variabel string sumber ke bagian akhir dari variabel string tujuan

strcmp()

Mempunyai bentuk:

var_int = strcmp(str1,str2);

Fungsi ini digunakan untuk membandingkan variabel string str1 dengan string str2. Hasil fungsi bertipe int berupa nilai

- Negatif, jika str1 kurang dari str2
- Nol , jika str1 sama dengan str2
- Positif , jika str1 lebih dari str2

Absolut hasil fungsi (kecuali jika bernilai nol) menyatakan selisih nilai ASCII dari karakter yang menyebabkan str1 berbeda dengan str2. Pembandingan dilakukan untuk karakter pada posisi yang sama dari dan str2, dimulai dari karakter terkiri. Acuan pembandingan dari dua buah karakter didasarkan oleh ASCII-nya. Misal karakter 'A' lebih kecil dari karakter 'B' dan karakter 'B' lebih kecil dari karakter. Contoh : string "HALO" lebih kecil dari string "HELO", karena karakter 'A' mempunai nilai yang lebih kecil daripada karakter 'E'. Apabila salah satu string mempunyai panjang yang lebih pendek, an sampai karakter yang terkahir dari string yang terpendek ternyata karakter antara str1 dengan str2 sama, maka string yang lebih pendek mempunyai nilai yang lebih kecil dibandingkan dengan string yang lebih panjang.

III. LANGKAH KERJA

a. Latihan 1 : memasukkan data string dari keyboard

10. Pilih menu → program → accesoris → command prompt
11. Jalankan progam turbo C
12. Alt + F, pilih New
13. Ketik :

```
/* ---------------------------------------- */
/* File program : nama.c           */
/* Contoh memasukkan data string dari keyboard   */
/* ---------------------------------------- */

#include <stdio.h>

main()
```

```
{
 char nama[15];
 printf("Nama anda : ");
 gets(nama) ;
  printf("Halo, %s. Selamat datang di Elektro.\n",nama);
}
```

14. Alt+F, pilih save ketik nama nama.c
15. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
16. Alt+C, compile to OBJ
17. Alt+C, compile to EXE
18. Alt+R

b. Latihan 2 : menampilkan data string ke layar

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ---------------------------------------- */
/* File program : kota.c            */
/* Contoh menampilkan data string ke layar      */
/* ---------------------------------------- */
#include<stdio.h>
void bentuk1(void);
void bentuk2(void);

main()
{
bentuk1();
bentuk2();
}

void bentuk1(void)
{
char kota[]=
{'Y','o','g','y','a','k','a','r','t',a'','\0'};
puts(kota);
}

void bentuk2(void)
{
char kota[] = "Solo";
puts(kota);
}
```

5. Alt+F, pilih save ketik nama kota.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE

9. Alt+R

c. Latihan 3 : mengakses elemen string

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ---------------------------------- */
/* File program : hitkar.c          */
/* Contoh menakses elemen string    */
/* ---------------------------------- */

#include<stdio.h>
#define maks 256

main()
{
int i, jumkar;
char teks[maks];
puts("Masukkan suatu kalimat.");
puts("Saya akan menghitung jumlah karakternya.");
gets(teks);

jumkar = 0;
for(i=0;teks[i];i++)
jumkar++;
printf("Jumlah karakter = %d\n", jumkar);
}
```

5. Alt+F, pilih save ketik nama hitkar.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

d. Latihan 4 : menyalin suatu string

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ---------------------------------- */
/* File program : SALINSTR.c        */
/* Contoh menyalin suatu string         */
/* ---------------------------------- */

#include<stdio.h>

main()
{
int i;
char keterangan[] = "Saya menyukai bahasa C";
char kalimat[30];
```

```
i = 0 ;
while(keterangan[i] !='\0')
{
kalimat[i] = keterangan[i];
i++;
}
kalimat[i] = '\0'; /*Beri karakter NULL*/

printf("Isi kalimat = %s\n", kalimat);
}
```

5. Alt+F, pilih save ketik nama salinstr.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

e. Latihan 5 : berbagai fungsi string

e.1. /*str51.c*/

1. Pilih menu → program → accesoris → command prompt
2. alankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* str51.c */
#include<stdio.h>
#include<string.h>
main()
{       char salam[]="Halo";
        printf("Panjang         string=%d         karakter\n",
strlen(salam));}
```

5. Alt+F, pilih save ketik nama str51.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. ‘Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

e.2. /*str52.c*/

1. Pilih menu → program → accesoris → command prompt
2. alankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* str52.c */
#include<stdio.h>
#include<string.h>
#define PJG 15
main()
{
```

```
char str1[PJG], str2[PJG];
strcpy(str1, "sala");  /* str1 diisi "sala" */
strcpy(str2, "tiga");  /* str2 diisi "tiga" */
strcat(str1, str2);    /* tambahkan str2 ke akhir str1 */
printf("str1 --> %s    str2 --> a %s\n", str1, str2);}
```

5. Alt+F, pilih save ketik nama str52.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. ‘Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

e.3. /*str53.c*/

1. Pilih menu → program → accesoris → command prompt
2. alankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* str53 */
#include<stdio.h>
#include<string.h>
main()
{      char   str1[]="HALO";   char   str2[]="Halo";   char
str3[]="HALO";
  printf("Hasil pembandingan %s dengan %s --> %d\n",
  str1, str2, strcmp(str1, str2));
  printf("Hasil pembandingan %s dengan %s --> %d\n",
  str2, str1, strcmp(str2, str1));
  printf("Hasil pembandingan %s dengan %s --> %d\n",
  str1, str3, strcmp(str1, str3));
}
```

5. Alt+F, pilih save ketik nama str53.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. ‘Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

f. Latihan 6: studi kasus

Buat program untuk menghasilkan keluaran sbb:

POLITEKNIK
POLITEKNI
POLITEKN
POLITEK
POLITE
POLIT
POLI
POL

PO
P

1. Alt+F, pilih save ketik nama politekn.c
2. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
3. Alt+C, compile to OBJ
4. Alt+C, compile to EXE
5. Alt+R

IV. PERTANYAAN

1 Ketikkan sebuah kalimat melalui keyboard dengan menggunakan *gets()* (atau *fgets()*) kemudian didapatkan keluaran berupa laporan tentang jumah huruf kecil dan dan huruf kapital dalam kalimat tsb.

2 Masukkan nama Anda, rubah ke dalam huruf besar semua, balikkan urutan hurufnya, selanjutnya tampilkan hasilnya di layar.

3 Ketikkan sebuah kalimat, hitung dan tampilkn jumlah kata dan karakternya.

4 Buatlah program utuk menguji apakah suatu kalimat itu termasuk polindrom atau bukan.

Catatan : kalmat dikatakan polindrom jika kata atau kalimat tersebut dibaca dari depan atau belakang sama bunyinya.

5 Buatlah kesimpulan dari praktik yang telah anda lakukan !

BAB 9

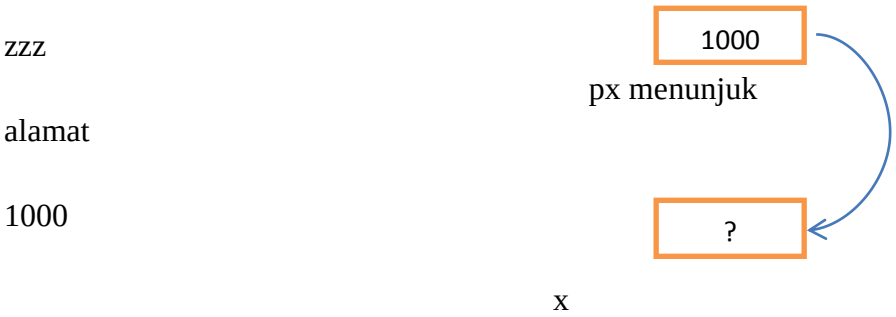
POINTER

I. TUJUAN

- ✓ Menjelaskan tentang konsep dari variabel pointer.
- ✓ Menjelaskan tentang pointer array.
- ✓ Menjelaskan tentang pointer string.
- ✓ Menjelaskan tentang array pointer.

II. DASAR TEORI

Pointer adalah suatu variabel yang menyimpan alamat dari suatu data, dan bukan menyimpan datanya sendiri, jika *px* adalah poinetr dan *x* adalah vaiabel yang ditunjuk oleh *px*, maka jika *px* berada pada alamat memori awal 1000, maka *px* akan berisi 1000, sedangkan datanya sendiri berupa nilai yang ada pada lokasi memori 1000.



a. Pendeklarasian Variabel Pointer

Variabel tipe pointer dideklarasikan dengan bentuk sebagai berikut:

```
type *nama_variabel;
```

dengan *tipe* dapat berupa sembarang tipe data, sedangkan nama_variabel adalah nama dari variabel pointer.

Sebagai contoh:

```
int px; /* contoh1 */
char pch1, pch2; /* contoh2 */
```

contoh pertama menyatakan bahwa px adalah variabel pointer yang menunjuk ke suatu data bertipe int. pada contoh kedua, masing-masing variabel *pch1* dan *pch2* adalah variabel pointer yang menunjuk ke data bertipe ***char.***

b. Mengatur Pointer Agar Menunjuk ke Variabel Lain

Agar suatu pointer menunjuk ke variabel yang lain, mula – mula pointer haus diisi dengan alamat dari variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel, dapat digunakan operator **&** (operator alamat, yang bersifat *unaray*), dengan cara menempatkan operator di depan nma variabel. Sebagai contoh, jika x dideklarsikan sebagai variabel bertipe *int,* maka **&x** , berarti "alamat dari variabel x". Adapun contoh pemberian alamat ke suatu variabel pointer *px* (yang dideklarasikan sebagai pointer yang menunjuk ke data bertipe *int*) yaitu:

Px = &x;

Pernyataan di atas berarti bahwa px diberi nilai berupa alamat dari variabel x, setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa *px* menunjuk ke variabel x.

c. Mengakses Isi Suatu Variabel Melalui Pointer

Jika suatu variabel sudah ditunjuk oleh pointer, maka variabel tersebut dapat diakses melalui variabel itu sendiri (disebut sebagau pengaksesan tak langsung), ataupun melalui pointer (isebut pengaksesan langsung). Pengaksesan tak langsung dilakukan dengan menggunakan operator *inderection* berupa simbol * (bersifat *unaray*), seperti contoh berikut: *px yang menyatakan "isi atau nilai variabel/data yang ditunjuk oleh pointer px". Sebagai contoh jika y bertipe int, maka sesudah dua penyataan berikut:

px = &x;
y = *px;

y akan berisi nilai yang sesusi dengan nilai x. Contoh program berikut memperlihatkan pemakaian variabel pointer

d. Tipe Variabel Pointer dan Tipe Obyek yang Ditunjuk

Antara tipe pointer (sesuai dengan pendeklarasian pointer) dan tipe obyek yang kan ditunjuk oleh pointer haruslah sejenis. Jika misalnya pointer *pu* dimaksudkan untuk menunjuk data bertipe *int* maka data yang akan ditunjuk oleh pointer *pu* juga harus bertipe *int*. suatu kesalahan akan terjadi jika misalnya pointer *floa*t digunakan untuk menunjuk data bertipe int.

e. Mengubah Isi Suatu Variabel Melalui Pointer

Contoh berikut memberikan gambaran tentang pengubahan isi sutu variabel secara tak langsung (yaitu melalui pointer). Mula-mula *pd* dideklarasikan sebagi *pointer* yang menunjuk ke suatu data bertipe *float* dan *d* sebagai variabel bertipe *float*. Selanjutnya dengan ungkapan

d = 54.6;

digunakan untuk mengisikan nilai 54.6 secara langsung ke variabel d. Perintah pd = &d; Digunakan untuk memberikan alamat dari *d* ke *pd*. Dengan demikian *pd* menunjuk ke variabel *d*. Sedangkan pernyataan berikut:

*pd = *pd + 10; (atau: *pd += 10;)

Merupakan instruksi untuk mengubah nilai variabel *d* secara tak langsung. Perintah di atas berarti "jumlahkan isi variabel yang ditunjuk oleh *pd* dengan nilai 10 dan simpan hasilnya ke variabel tersebut", atau identik dengan pernyataan:

D = d + 10;

Namun seandainya tidak ada instruksi
Pd = &d;
Maka pernyataan:
*pd = *pd + 10;
Tidak sama dengan
d = d + 10

f. Pointer dan Array

Hubungan antara struktur data pointer dan array dalam C sangatlah erat, sebab sesungguhnya array secara internal akan diterjemahkan dalam benruk pointer. Contoh berikut akan memberi gambaran tentang hubungan antara pointer dan array. Misalnya dalam suatu fungsi dideklarasikan:

static int tgl_lahir[3] = { 01, 09, 64}; -dan

Kemudian diberikan pernyataan

ptgl = &tgl_lahir[0];

maka *ptgl* akan berisi alamat dari elemen array *tgl_lahir* yang berindeks nol. Instruksi di atas juga dapat ditulis menjadi:

ptgl = tgl_lahir;

sebab nama array tanpa tanda kurung menyatakan alamat awal dari array. Sesudah penugasan seperti di atas, maka

*ptgl

Dengan sendirinya menyatakan elemen pertama (berindeks sama dengan nol) dari array *tgl_lahir*. Contoh di atas dapat dipahami melalui contoh program berikut ini.

```
/*-----------------------------------*/
/* File program : Pointer3.c        */
/* Pointer yang menunjuk array */
/* ----------------------------------*/
#include <stdio.h>

main()
{
static int tgl_lahir[] = {24, 6, 1965};
int *ptgl;
ptgl = tgl_lahir; /* ptgl berisi alamat array */

printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
printf("nilai dari tgl_lahir[0] = %d\n", tgl+lahir):
}
```

Jika ingin menampilkan seluruh elemen array *tgl_lahir*, maka dapat digunakan perintah

```
for(i=0; i<3; i++)
```

```
printf("%s%,tgl_lahir[i]");
```

jika diimplementasikan dengan menggunkan pointer

```
tgl_lahir[i]
```

dapat digantikan menjadi

```
*(ptgl + i)
```

Dengan terlebih dahulu mengatur *ptgl* agar menunjuk ke array *tgl_lahir*, sehingga penulisan instruksi penampilan isi array *tgl_lahir* dapat diubah menjadi:

```
ptgl = tgl_lahir;
for(i=0; i<3;i++0)
printf("%d", *(ptgl + i));
```

Secara umum operasi pointer dapat diterangkan sebagi berikut. Misalkan *a* adalah suatu array, dan *pa* adalah pointer yang menunjuk array *a*, maka

*(pa + i)

Akan menyatakan lemen array dengan indeks sama dengan *i*. Jadi

*(pa + 0) identik dengan a[0]

*(pa + 1) identik dengan a[1]

*(pa + 2) identik dengan a[2]

Ungkapan seperti

pa + i;

memiliki arti "tambahkan nilai pa (berisi alamat) dengan i kali ukuran dari obyek yang ditunjuk oleh pa".

Jika pa dideklarasikan sebagai

int *pa;

maka obyek dari pa adalah data int(berukuran 2 byte). Cara lain dalam menampilkan isi suatu array yaitu dengan menaikkan isi variabel pointer dengan menggunakan opertor ++.

*(ptgl + i)

dapat diganti menjadi

ptgl++

Misalkan suatu instruksi :

```
int *pa;
int a[3];
```

Sesudah pernyataan: **pa = a;**

Sesudah pernyataan: **pa = ++;**

g. Array dari Pointer

Suatu array dapat digunakan untuk menyimpan sejumlah pointer. Misal pertanyaan :

char *namahari[10]

merupakan pernyataan untuk mendeklarasikan array pointer. Aray namahari terdiri dari 10 elemen berupa pointer yang menunjuk ke data bertipe char.

h. Inisialisasi Array Pointer

Array pointer dapat diinisialisasi sewkatu pendeklarasian. Sebagai contoh:

Static char *namahari[] =

{“Senin”,”Selasa”,”Rabu”,”Kamis”,”Jumat”,”Sabtu”,”Minggu”};

Pada contoh di atas,

namahari[0] menunjuk ke string “Senin”

namahari[1] menunjuk ke string “Selasa”

namahari[2] menunjuk ke string “Rabu”

dan seterusnya

i. Pointer Menunjuk Pointer

Suatu pointer bisa saja menunjuk ke pointer lain. Gambar di bawah memperlihatkan contoh mengenai hal ini.

Untuk membentuk rantai pointer seperti gambar di atas, diperlukan pendeklarasian sebagai berikut :

```
int var_x;
int *ptr1;
int **ptr2;
```

Fungsi di atas dimaksudkan agar jika dipanggil, variabel yang berkaitan dengan parameter aktual dapat diubah nilainya, yaitu masing-masing dinaikkan sebesar 2. Perhatikan pada deklaasi di depan:

Contoh pemanggilan fungsinya adalah

- *var_x* adalah variabel bertipe *int.* naikkan_nilai(&a, &b);
- *ptr1* adalah variabel pointer yang menunjuk ke data bertipe *int.*
- *ptr2* adalah variabel pointer yang menunjuk ke pointer bertipe int. (itulah sebabna deklarasi berupa : *int **ptr2*;)
  Perhatikan bahwa dalam hal ini variabel *a* dan *b* harus ditulis diawali dengan operator alamat (&) yang berarti menyatakan alamat variabel, sebab parameter fungsi dalam pendefinisian berupa pointer.

Agar *ptr1* menunjuk ke variabel *var_x*,perintah yang diperlukan berupa

```
ptr1 = &var_x
```

Sedangkan supaya *ptr2* menunjuk ke *ptr1*, instruksi yang diperlukan adalah : Program 9-5

```
Ptr2 = &ptr1;
```

Contoh program berikut memberikan gambaran cara pengaksesan nlai pada *var_x* melalui pointer *ptr1* dan *ptr2*

j. Pointer dan Fungsi

<u>j.1. Pointer Sebagai Parameter Fungsi</u>

Penerapan pointer sebagai parameter fungsi yaitu jika diinginkan gar nilai suatu variabel internal dapat diubah oleh fungsi yang dipanggil. Misal pada fungsi berikut :

```
void naikkan_nilai (int *, int *y)
{
*x = *x + 2;
```

```
*y = *y + 2;
}
```

Fungsi di atas dimaksudkan agar jika dipanggil variabel yang brkaitan dengan parameter aktua dapat diubah nilainya, yaitu masing-masing dinaikkan sebesar 2.

Contoh pemanggilan fungsinya adalah

naikkan_nilai(&a, &b);

Perhatikan bahwa dalam hal ini variabel a dan b harus ditulis diawali dengan operator alamat (&) yang berarti menyatakan alamat variabel, sebab parameter fungsi dalam pendefinisian berupa pointer.

j.2. Pointer Sebagai Keluaran Fungsi

Suatu fungsi dapat dibuat agar keluarannya berupa pointer. Misalnya, suatu fungsi menghasilkan keluaran berupa pointer yang menunjuk ke string nama bulan, seperti pada contoh berikut

III. LANGKAH KERJA

a. Latihan 1 : pemakaian pointer

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ------------------------------*/
/* File program : Pointer.c    */
/* Contoh pemakaian pointer    */
/* ------------------------------*/
#include<stdio.h>

main()
{
int x,y; /* x dan y bertipe int */
int*px; /* px pointer yang menunjuk obyek bertipe int */

x = 87;
px = &x; /* px berisi alamat dari x */
y = *px; /* y berisi nilai yang citunjuk px */

printf("Alamat x = %p\n",&x);
printf("Isi px = %p\n",px);
printf("Isi x = %d\n",x);
printf("Nilai yang ditunjuk px = %d\n",*px);
printf("Nilai y = %d\n",y);
}
```

5. Alt+F, pilih save ketik nama pemakaia.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE

9. Alt+R

b. Latihan 2 : pointer menunjuk array

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ----------------------------*
 * File program : Pointer3.c   *
 * Pointer yang menunjuk array *
 * ----------------------------*/
 #include<stdio.h>
 #include<conio.h>

 void main()
 {
 static int tgl_lahir[] = { 13,9,1982 };
 int*ptgl;
 ptgl = tgl_lahir;/* ptgl berisi alamat array */
 printf("Diakses dengan pointer");
 printf("Tanggal = %i\n",*ptgl);
 printf("Bulan = %i\n",*(ptgl + 1));
 printf("Tahun = %i\n",*(ptgl + 2));

 printf("\nDiakses dengan array biasa\n");
 printf("Tanggal = %i\n",tgl_lahir[0]);
 printf("Bulan = %i\n",tgl_lahir[1]);
 printf("Tahun = %i\n",tgl_lahir[2]);
 getch();
 }
```

5. Alt+F, pilih save ketik nama array.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

c. Latihan 3 : pointer menunjuk ke pointer

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ------------------------------------*
 * File program : Ptrptr.c          *
 * Contoh pointer menunjuk pointer  *
 * ------------------------------------*/
 #include<stdio.h>
 main()
```

```
{
int var_x = 234; /*variabel int*/
int *ptr1; /*pointer int*/
int **ptr2; /*pointer menunjuk pointer int*/
ptr1 = &var_x; /*ptr1 berisi alamat var_x*/
ptr2 = &ptr1; /*ptr2 berisi alamat ptr1*/
/*Mengakses nilai var_x melalui ptr1*/ printf("Nilai var_x =
%d\n",*ptr1);
/*Mengakses nilai var_x melalui ptr2*/ printf("Nilai var_x =
%d\n",**ptr2);
getche();
}
```

5. Alt+F, pilih save ketik nama point.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

d. Latihan 4 : pointer sebagai fungsi keluaran

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ------------------------------------*
 * File program : Ptrbulan.c       *
 * Contoh fungsi dengan keluaran berupa *
 * pointer yang menunjuk string         *
 * ------------------------------------*/

#include<stdio.h>
char*nama_bulan(int n);
main()
{
int bl;
printf("Bulan(1..12):");
scanf("%d", &bl);
printf("%s", nama_bulan(bl));
}

char*nama_bulan(int n)
{
static char*bulan[] =
{"Kode                                                    bulan
salah","Januari","Pebruari","Maret","April","Mei","Juni","Juli"
,"Agustus",
  "September","November","Desember"};

return((n<1 || n>12)?bulan[0] : bulan[n]);
}
```

5. Alt+F, pilih save ketik nama luar.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

e. Latihan 5 : penukaran string via pointer

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ------------------------------------*
 * File program : Ptrptr.c          *
 * Contoh Penukaran string via pointer  *
 * ------------------------------------*/
 #include<stdio.h>
 #include<conio.h>
 char*nama1 = "SPIDERMAN";
 char*nama2 = "GATOTKACA";
 void main()
 {
 char nama;
 clrscr();
 puts("SEMULA:");
 printf("Saya suka >> %s\n", nama1);
 printf("Tapi saya juga suka >> %s\n", nama2);
 /* Penukaran string yang ditunjuk oleh pointer nama1 dan nama2
*/
 printf("SEKARANG:");
 printf("Saya suka >> %s\n", nama1);
 printf("Dan saya juga masih suka >> %s\n", nama2);
 getch();
 }
```

5. Alt+F, pilih save ketik nama tukar.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

f. Latihan 6 : studi kasus

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
/* ------------------------------------ *
 * File program : Ptrptr.c          *
 * contoh penukaran string via pointer  *
```

```
 * ---------------------------------- */
 #include<stdio.h>
 #include<conio.h>
 char *nama1 = "POLITECNIC";
 char *nama2 = "SEMARANG";
 char *nama3 = "STATE";
 void main()
 {
 char nama;
 clrscr;
 puts("Kalimat Semula : ");
 printf("%s %s %s \n \n", nama1, nama2, nama3);
 /* penukaran string yang ditunjuk oleh pointer nama1 dan nama2
*/
 printf("Kalimat Kini : ");
 printf("%s %s %s \n \n", nama3, nama2, nama1);
 getch()
 }
```

5. Alt+F, pilih save ketik nama politekc.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

IV. PERTANYAAN

1 Apa yang tercetak dari program-program berikut ini?

a. Program pertama

```
#include<stdio.h>
void misteri2(const char *);
main()
{
char *string = "polines";
misteri2(string);
putchar('\n');
return0;
}void misteri2(const char *s) {
for (; *s !='\0'; s++)
putchar(*s);
}
```

b. Program kedua

```
#include<stdio.h>
int misteri3(const char *);
void main()
{
char string[80];
printf("Ketik sebuah string : "); scanf("%s", string);
```

```
printf("%d\n", misteri3(string));
}
int misteri3(const char *s) {
int x = 0;
for(; *s !='\0' ; s++)
++x;
return x ;
}
```

2 Buatlah kesimpulan dari praktik yang telah anda lakukan

BAB 10

OPERASI FILE

I. TUJUAN

- ✓ Menjelaskan tentang struktur file
- ✓ Menjelaskan tentang tahap-tahap operasi pada file
- ✓ Menjelaskan tentang fungsi untuk penyimpanan dan pembacaan file per-karakter
- ✓ Menjelaskan tentang file biner dan file teks
- ✓ Menjelaskan tentang operasi penyimpanan dan pembacaan file per-int
- ✓ Menjelaskan tentang operasi penyimpanan dan pembacaan file per-blok
- ✓ Menjelaskan cara membaca dan menyimpan data string pada file
- ✓ Menjelaskan cara mengakes file biner secara acak
- ✓ Menjelaskan cara menghapus file
- ✓ Menjelaskan cara mengganti nama file

II. DASAR TEORI

a) Pendahuluan

1) Penyimpanan suatu data dalam disk berupa suatu file.
2) Gambar struktur file:
3) Tahapan Operasi File:
   - Membuka / mengaktifkan file
   - Melaksanakan proses file
   - Menutup file

   - Bentuk deklarasi:

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

Keterangan :

- **namafile** berupa nama dari file yang akan diaktifkan
- **mode** berupa jenis oprasi yang akan dilakukan terhadap file
- **prototipe** ada pada file stdio.h

b) Jenis Operasi File

1) **r** menyatakan file hanya akan dibaca, jika file belum ada maka tidak akan berhasil.
2) **w** menyatakan bahwa file baru diciptakan. Jika file tersebut sudah ada dalam disk, isinya yang lama akan terhapus.

3) **a** untuk membukanfile yang sudah ada untuk ditambah dengan data, jika file belum ada akan dibuat yang baru
4) **r+** sama dengan **"r"** tetapi selain file dapat dibaca, file juga dapat ditulisi.
5) **w+** sama dengan **"w"** tetapi selain file dapat ditulisi, file juga dapat dibaca.
6) **a+** sama dengan **"r"** tetapi selain file dapat ditulisi, file juga dapat dibaca.
7) Berhasil tidaknya operasi pengaktifan file dapat dilihat pada keluaran fungsi *fopen()*.
8) Jika keluaran fungsi berupa NULL (suatu makro yang didefinisikan pada file stdio.h), berarti operasi pengaktifan file gagal → misal membuka file dengan mode "r" tapi filenya belum ada
9) Contoh :

```
FILE *pf; //deklarasi variabel pf
pf = fopen("COBA.TXT" "W")
```

- Menciptakan dan mengaktifkan file bernama "COBA.TXT"
- Dengan mode yaitu "w" (mode penulisan ke file)
- Dan menempatkan pointer-ke-FILE e variabel pointer pf

```
if(pf = fopen("COBA.TXT" "w") == NULL)
{
  printf("File tidak dapat diciptakan !\n");
  exit(1);       //keluar dari program
}
```

Keterangan :
- pf akan diisi dengan keluaran dari fungsi *fopen()*.
- Jika nilainya NULL, maka akan mencetak "File tidak dapat diciptakan", setelah itu program dihentikan

c) Menutup File

1) Apabila sudah tidak diproses lagi, maka file tersebut ditutup, karena adanya keterbatasan jumlah file yang dapat dibuka secara serentak.
2) Perintah yang digunakan : fclose();
3) Bentuk deklarasi :

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

4) Bentuk deklarasi yang lain :

```
int fcloseall(void) ——————→ fcloseall();
```

Prototype yang digunakan : stdio.h

d) Operasi Penyimpanan File

1) Penyimpanan karakter ke file menggunakan perintah : fputc();
2) Bentuk deklarasi :

```
int fputc(charkar, FILE ptr_file)
```

- **ptr_file** adalah pointer-ke-FILE yang berisi keluaran dari *fopen()*,

- kar berupa karakter yang kan disimpn dalam file.

e) Operasi Pembacaan File

1) Pembacaan karakter dari suatu file memakai perintah : fgetch().
2) Bentuk deklarasi :

```
int fgetc(FILE *ptr_file)
```

PROSES PEMBACAAN FILE PER KARAKTER :

1. Buka file COBA.TXT dengan mode "r"
   Jika tidak berhasil dibuka maka
   - beri keterangan pada layar bahwa file tak ada
   - selesai
2. Baca sebuah karakter dari file
   jika karakter sama dengan EOF (tanda akhir file) maka ke langkah 4
3. Tampilkan karakter ke layar dan kembali ke langkah 2
4. Tutup file
5. Selesai

f) Jenis File

1) File Biner : file yang pola penyimpanan di dalam disk berbentuk biner, yaitu seperti bentuk pada memori RAM (komputer). Dipakai untuk menyimpan data kompleks, mis : struct.
2) File Teks : file yang pola penyimpanan datanya dalam bentuk karakter. Dipakai untuk menyimpan data seperti karakter atau string.
3) Penentuan mode teks dan mode biner :
   t : untuk mode teks
   b : untuk mode biner

Contoh :
"rt" : mode file teks dan file hendak dibaca
"rt+" : mode file teks dan file bisa dibaca dan ditulisi.
Bisa juga ditulis : "r+t"
"rb" : mode file biner dan file hendak dibaca

g) Operasi Baca dan Tulis File Per-Int

1) Perintah yang digunakan : _putw(), _getw().
2) Bentuk deklarasi :

```
int_putw(int nilai, FILE *ptr_file);
int_getw(FILE *ptr_file);
```

Kegunaan :
_getw() untuk membaca sebuah data bertipe int dari file
_putw() untuk menyimpan sebuah data bertipe *int* ke file

h) Operasi Baca dan Tulis File Per-Blok

1) Fungsi : untuk menyimpan atau membaca data file dalam bentuk kesatuan blok(sejumlah byte), misal float atau struct.
2) Perintah yang digunakan : fread() dan fwrite();
3) Bentuk deklarasi :

```
int fread(void *buffer, int n, FILE *ptr_file);
int fwrite(void *buffer, int jum_byte, FILE *ptr_file);
```

dengan :
buffer : pointer yang menunjuk ke alamat memori
jum_byte : jumlah byte yang akan dibaca atau disimpan
n : banyaknya blok data berukuran jum_byte yang akan ditulis/dibaca
ptr_file : pointer-ke-FILE yang berisi nilai keluaran dari *fopen()*

i) Operasi Baca dan Simpan Data String pada File

1) Perintah yang digunakan : *fgets()* dan *fputs()*
2) Bentuk deklarasi :

```
int fputs(char *str, FILE *ptr_file);
char fgets(char *str, int n, FILE *ptr_file);
```

Kegunaan:
fputs() : menyimpan string str ke dalam file.
fgets() : membaca string dari file sampai ditemukannya karakter baris baru '\n' atau setelh (n-1) karakter, dengan n adalah panjang maksimal string yang dibaca per waktu-baca.

Note :

- Saat *simpan, fputs()* tidak menambahkan karakter baris-baru('\n') dengan sendirinya, dan karakter tidak ikut disimpan.
- Baik *fgets()* maupun *fputs* digunakan untuk file teks.

j) Akses File Biner Secara Acak

1) Tujuan : membaca data di tengah file scr cepat.
2) Perintah yang digunakan : fseek().
3) Bentuk deklarasi :

```
int fseek(FILE *ptr_file, long int offset, int posisi);
```

dengan :
ptr_file adalah pointer yang berasal dari keluaran *fopen()*
offset menyatakan jumlah byte terhadap posisi
posisi dapat diisi suatu nilai tertera pada tabel

Konstanta	Nilai	Lokasi file
SEEK_SET	0	Awal file
SEEK_CUR	1	Posisi penunjuk file saat ini
SEEK_END	2	Akhir file

Prototype : stdio.h

Contoh Aplikasi fseek()

k) Operasi Menghapus File

Bentuk deklarasi :

```
int remove(char *namafile);
```

Namafile : pointer yang menunjuk ke nama file yang akan dihapus

- Jika operasi hapus berhasil, akan menghasilkan output = 0

  Prototype : stdio.h

l) Operasi Mengganti Nama File

Bentuk deklarasi :

```
int rename(char *namafilelama, char *namafilebaru);
```

Namafile : pointer yang menunjuk ke nama file yang akan dihapus

- Jika operasi hapus berhasil, akan menghasilkan output = 0

  Prototype : stdio.h

III. LANGKAH KERJA

1. Latihan 1 Dasar Operasi File

a. Latihan 1a : program menciptakan file

19. Pilih menu → program → accesoris → command prompt
20. Jalankan progam turbo C
21. Alt + F, pilih New
22. Ketik :

```
#include<stdio.h>
#include<conio.h>
main()
{
  FILE *pf; /* Pointer-ke-FILE */
  char kar;

  /* Ciptakan file */
  if((pf=fopen("COBA.TXT","w")) == NULL)
  {
       printf("file tak dapat diciptakan!\r\n");
       exit(1);
  }
  /*Masukkan karakter per karakter sampai ditekan ENTER*/
  while((kar=getchar())!='\n')
       fputc(kar, pf);
```

```
    fclose(pf); /* tutup file */
}
```

23. Alt+F, pilih save ketik nama 1acpt.c
24. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
25. Alt+C, compile to OBJ
26. Alt+C, compile to EXE

27. Alt+R

b. Latihan 1b : program baca file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
main()
{
   FILE *pf;
   char kar;

   if((pf=fopen("COBA.TXT","r")) == NULL) /* buka file */
   {
        printf("file tak dapat dibuka !\r\n");
        exit(1);
   }
   /*Baca karakter per karakter sampai ditemui EOF*/
   while((kar=fgetc(pf)) !=EOF)
        putchar(kar);
   printf("\n");
   fclose(pf);   /* tutup file*/
}
```

5. Alt+F, pilih save ketik nama 1bbc.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

c. Latihan 1c : program create & baca file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

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

```
main()
{
   FILE *pf; /* Pointer-ke-FILE */
   char kar;
   /* Ciptakan file */
   if((pf = fopen("COBA.TXT","r+")) == NULL )
   {
        printf("file tak dapat diciptakan !\r\n");
        exit(0);
   }
   while((kar=fgetc(pf)) !=EOF) /* baca kar dari file */
        putchar(kar);

   while((kar=getchar()) !='\n') /* baca kar dr keyboard */
        fputc(kar,pf);
   fclose(pf);
}
```

5. Alt+F, pilih save ketik nama 1ccdb.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE

9. Alt+R

2. Latihan 2 Operasi Tulis & Baca File Per-Int

a. Latihan 2a : program menciptakan file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   FILE *pf; /* ptr-ke-FILE */
   int nilai, data, i;
   char jawab;
   if((pf=fopen("BILANGAN.DAT", "wb")) == NULL)
   {
        printf("file gagal diciptakan!\n");
        exit(1);
   }
   printf("Masukkan banyaknya data : ");
   scanf("%d",&data);
   for(i=0;i<data;i++)
   {
        printf("\nBilangan yang disimpan : ");
        scanf("%d",&nilai);     /*  baca  nilai  dr  keyboard
*/
```

```
        putw(nilai, pf);   /* baca bilangan ke file */
    }
    printf("\nOke.Data sudah disimpan dalam file.\n");
    fclose(pf);   /* menutup file */
}
```

5. Alt+F, pilih save ketik nama 2acpt.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Latihan 2b : program membaca file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :
5. Alt+F, pilih save ketik nama 2bbc.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

3. Latihan 3 Operasi Tulis & Baca File Per-Blok

a. Latihan 3a : program menciptakan file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
main()
{
   FILE *f_struktur;
   char jawaban;
   int sudah_benar;

   struct
   {
   char judul[20];
   char pengarang[20];
   int jumlah;
   } buku;   /* variabel buku bertipe struktur */

   if((f_struktur = fopen("DAFBUK.DAT","wb")) == NULL)
   {
```

```
        printf("File tidak dapat diciptakan!\n");
        exit(1);
    }
    do
      {
        fflush(stdin); /* Hapus isi penampung keyboard */
        printf("Judul buku :");
        gets(buku.judul);
        printf("Nama pengarang  :");
        gets(buku.pengarang);
        printf("Jumlah buku     :");
        scanf("%d", &buku,jumlah);
        fflush(stdin); /* Hapus isi penampung keyboard */
        /* Rekam sebuah data betipe struktur */
        fwrite(&buku,sizeof(buku), 1, f_struktur);
        printf("\nMau merekam data lagi[Y/T]?");
        jawaban = getchar();
        printf("\n");
      }while(jawaban == 'Y'|| jawaban == 'y');
      fclose(f_struktur);        /* Tutup file */
}
```

5. Alt+F, pilih save ketik nama 3acpt.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Latihan 3b : program baca file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
mian()
{
FILE *f_struktur;
int i = 1;
struct
{
  char judul[20];
  char pengarang[20];
  int jumlah;
} buku      /* variabel buku bertipe struktur */

if((f_struktur = fopen("DAFBUK.DAT"."rb")) == NULL)
{
  printf("%2s. %-30s %-30s %s\n\n", "No", "Judul Buku",
"Nama Pengarang", "Jumlah");
  /* diulang selama masih ada record yg terbaca dlm file
*/
```

```
  while(fread(&buku, sizeof(buku), 1, f_struktur) == 1)
  printf("%2d.   %-30s   %-30s   %4d\n",   i++,buku.judul,
buku.pengarang, buku.jumlah);
  printf("\n");
  fclose(f-struktur);          /* Tutup file */
}
```

5. Alt+F, pilih save ketik nama 3bbc.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

4. Latihan 4 Operasi Tulis & Simpan Data String

a. Latihan 4a : program tulis file2

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
main()
{
FILE *pf; /* pointer-ke-FILE*/
int data,i;
char nama[40];
clrscr();
if((pf=fopen("latihan.txt","w"))==NULL)
{
printf("File gagal diciptakan...!\n");
exit(1);
}
printf("Masukkan banyaknya data :");
scanf("%d", &data);
for(i=1; i<=data;i++)
{
printf("\nNama ke-%d : ", i);
fflush(stdin);
gets(nama);
strcat(nama, "\n");
fputs(nama,pf);
}
printf("\nOke.Data sudah disimpan dalam file,,,,,\n");
fclose(pf);
getch();
}
```

5. Alt+F, pilih save ketik nama lat4atls.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc

7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Latihan 4b : program tulis file2

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<conio.h>
main()
{
FILE *pf; /*pointer-ke-FILE*/
int data,i=1;
char nama[40];
clrscr();
if((pf=fopen("latihan.txt","w"))==NULL)
{
printf("File gagal diciptakan....!\n");
exit(1);
}
printf("\nNama ke-%d : ",i);
fflush(stdin);
gets(nama);
fputs(nama,pf);
printf("\nOke. Data sudah disimpan dalam file.....\n");
fclocse(pf);
getch();
}
```

5. Alt+F, pilih save ketik nama lat4b.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

c. Latihan 4c : program tulis file2

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
main()
{
FILE *pf; /*pointer-ke-FILE*/
int data,i=1;
char nama[40];
clrscr();
```

```
if((pf=fopen("latihan.txt","r"))==NULL)
{
printf("File gagal dibuka...!\n");
exit(1);
}
/* baca file per string sampai ditemui EOF */
while(fgets(nama,6,pf))
printf("%s\n",nama);
fclose(pf);
getch();
}
```

5. Alt+F, pilih save ketik nama lat4ctls.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

5. Latihan 5 Contoh Program fseek()

a. Latihan 5a : program mencari karakter

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
main()
{
FILE *pf;
char kar, jawab;
int i, no_record;
long int offset_byte;
if((pf =fopen("latihan.txt", "r")) == NULL) {
    printf("File tidak dapat dibuka !\n");
    exit(1);
    }
    do
    {
  printf("\n Nomor record dr data yg mau ditampilkan : ");
  scanf("%d", &no_record);
  offset_byte = (no_record-1);
  fseek(pf, offset_byte, SEEK_SET);
  kar=fgetc(pf);     /* baca kar dari file */
  putchar(kar);      /* tampilkan ke layar */
  printf("\nMau mencoba lagi (Y/T)? ");
  jawab=getche();
    } while (jawab == 'y' || 'Y');
    printf("\n");
    fclose(pf);
```

```
}       /* Tutup file */
```

5. Alt+F, pilih save ketik nama 5ack.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Latihan 5b : program mencari string

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
main()
{
  FILE *pf;
  char jawab, nama[20];
  int i, no_record;
  long int offset_byte;
  if((pf =fopen("latihan.txt", "r")) == NULL) {
       printf("File tidak dapat dibuka !\n");
       exit(1);
  }
  do
  {
       printf("\n Nomor record dr data yg mau ditampilkan
: ");
       scanf("%d", &no_record);
       offset_byte = (no_record-1);
       fseek(pf, offset_byte, SEEK_SET);
       printf("%s\n",fgets(nama,20,pf));
       printf("\nMau coba lagi(Y/T)");
       jawab=getche();
  } while(jawab == 'y' || jawab == 'Y');
  printf("\n");
  fclose(pf);
}      /* Tutup file */
```

5. Alt+F, pilih save ketik nama 5bcs.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

6. Latihan 6 Contoh Program fseek

a. Latihan 6a : program untuk mnghapus file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
#define PJG 65
main()
{
  int kode;
  char namafile[PJG];

  printf("Nama file yang akan dihapus : ");
  gets(namafile);
  kode = remove(namafile);
  if(kode == 0)
   printf("File sudah dihapus\n");
  else
   printf("Gagal dalam menghapus\n");
}
```

5. Alt+F, pilih save ketik nama 6ahps.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

b. Latihan 6b : program untuk mengganti nama file

1. Pilih menu → program → accesoris → command prompt
2. Jalankan progam turbo C
3. Alt + F, pilih New
4. Ketik :

```
#include<stdio.h>
#include<stdlib.h>
#define PJG 65
main()
{
int kode;
char namafilelama[PJG], namafilebaru[PJG];

printf("Nama file yang akan diganti : ");
gets(namafilelama);
printf("Nama file yang baru : ");
gets(namafielbaru);

kode = rename(namafilelama, namafilebaru);
if(kode == 0)
  printf("Nama file sudah diganti\n");
else
  printf("Gagal dalam mengganti nama\n");
```

```
}
```

5. Alt+F, pilih save ketik nama 6bgnt.c
6. Alt+O, pilih directories, ganti semua dengan C:\TC, lalu esc
7. Alt+C, compile to OBJ
8. Alt+C, compile to EXE
9. Alt+R

IV. PERTANYAAN

1. Sebutkan dan jelaskan struktur suatu file !
2. Sebutkan dan jelaskan secara garis besar tahapan operasi suatu file, sertakan contohnya untuk masing-masing tahapan.

3. Sebutkan dan jelaskan jenis-jenis file.
4. Perhatikan potongan kode program berikut ini :

   ```
   FILE *pf;
   pf =fopen(“LATIHAN.TXT”, “w+”);
   ```

   Jelaskan maksud dari kode program tersebut !
5. Buatlah kesimpulan dari praktik yang telah anda lakukan!

LINKED LIST

Tujuan Pembelajaran

Setelah bekerja melalui modul ini, mahasiswa diharapkan dapat :

1. Membuat, mengkompile dan menjalankan program C sederhana menggunakan Singly linked list LIFO
2. Membuat, mengkompile dan menjalankan program C sederhana menggunakan Singly Linked List FIFO
3. Menghapus dan menyisipkan data menggunakan Linked List

Linked List

Array dengan suku-suku berupa struktur untuk mencatat sejumlah data berstruktur sama, menggunakan alokasi memori bersifat statik. Artinya, jika array tersebut dideklarasikan berjumlah 100 suku, maka alokasi memori juga berjumlah 100 memori susunan struktur.

Jika dari 100 struktur tersebut hanya digunakan 50, berarti sisanya berupa pemborosan memori karena tidak dipergunakan. Oleh sebab itu masalah pencatatan dann perekaman data terstruktur, sangat jarang sekali menggunakan array struktur, karena penggunaan memori tidak dinamik mengikuti kebutuhan.

Perihal pencatatan dan perekaman sejumlah data terstruktur dengan bentuk struktur yang sama, selalu menggunakan linked list. Linked list etrdiri dari sejumlah struktur dalam bentuk kait-mengait dari struktur satu ke struktur berikutnya.

Kaitan tersebut dijalin menggunakan pointer pada salah satu (atau lebih) elemen struktur-nya untuk mencatat adress struktur yang terletak berikutnya atau terletak sebelumnya.

Contoh bentuk paling sederhana suatu struktur yang dapat dikaitkan dalam linked list adalah sebagai berikut :

```
Struct catatan_murid {
```

```
                    Unsigned long Nomor_pokok;
                    Char nama[50];
                    Struct catatan_murid *Next;
                    }
Struct catatan_murid *Catat;
```

Wujud Next adalah suatu elemen struktur (member of struktur) berupa pointer. Artinya, Next dapat diisi address struktur sebelumnya, atau address struktur berikutnya sehingga terjalin mata rantai struktur untuk mencatat sejumlah data terstruktur secara berderet, dengan letak urutan yang pasti.

Struktur yang dilengkapi denngan elemen (member) berwujud pointer, dapat dijadikan array struktur dinamik. Yaitu jumlah struktur dan sekaligus alokasi memorinya dapat dipesan tambahannya, atau dikurangi jumlahnya sesuai dengan jumlah keperluan saja, pada saat program sedang berjalan.

Bentuk inilah yang menjadi bentuk dasar pencatat atau perekam data terstruktur pada lingkup data base. Bentuk paling sederhana adalah singly linked list (one way linked list), dengan referensi struktur awal ke arah struktur akhir secara urut. Bentuk ini cukup hanya dilengkapi satu elemen pointer di dalam strukturnya.

Bentuk struktur yang dilengkapi 2 elemen pointer disebut Doubly Linked List (Two way linked list), karena satu elemen pointer dapat digunakan mencatat address dari rah struktur awal ke struktur akhir, dan elemen pointer lainnya dapat digunakan mencatat address dari arah struktur akhir ke struktur awal.

Jumlah elemen pointer didalam suatu struktur dapat ditentukan sesuai dengan keperluan. Pemanfaatannya tidak lain untuk menjalin pencatatan address structure berikutnya atau sebelumnya. Kaitanjalinan inilah yang disebut link dan dapat disususn menjadi jala-jala pencatatan sejumlah besar data terstruktur.
Namun lingkup pencatatan data terstruktur menggunakan jala-jala kaitan pointer, termasuk lingkup lanjut dan memerlukan bidang sendiri. Bidang ini pada umumnya mencakup bidang yang khususmenelaah pengolahan data-data terstruktur, terutama bentuk data bertipe struktur.
Lingkup penjelasan didalam buku ini, hanya akan meletakkan dasar-dasar pokok tentang pencatatan data terstruktur.

Agar deretan sejumlah struktur berstruktur sama dapat dijalin menjadi mata rantai catatan, maka susunan jalinan itu harus mempunyai awal dan mempunyai akhir. Artinya perlu ditentukan pointer awal dan pointer akhir, sehingga struktur awal dan struktur akhir secara pasti dapat diketahui posisinya secara pasti.

Bentuk kaitan paling sederhana hanya menggunakan satu elemen pointer didalam struktur. Rincian mengaitkan dapat dilakukan dengan 2 macam cara:

1. Kaitan Last In First Out atau kaitan LIFO.
2. Kaitan First In First Out atau kaitan FIFO.

Kaitan LIFO berarti struktur yang disimpan terakhir (Last In) hanya dapat dimunculkan berada di urutan teratas (First Out) bilamana dilakukan pembacaan atau penyajian kembali.

Kaitan LIFO seringkali disebut STAC, karena keadaannya analog seperti tumpukan keping kayu atau tumpukan buku. Setelah keping-keping ditumpuk, maka keping teratas yang dapat diambil tanpa mengganggu keping lainnya.

Kaitan FIFO erarti struktur yang disimpan pertama kali (First Out) juga akan muncul menjadi berada di posisi pertama (First Out) bilamana dibaca dan disajikan kembai. Kaitan FIFO seringkali disebut QUE (antrian).

Tujuan 1 Setelah bekerja melalui modul ini, mahasiswa diharapkan dapat membuat, mengkompile dan menjalankan proghram C sederhana menggunakan Singly linked list LIFO

SINGLY LINKED LIST LIFO

Contoh sederhana tentang Linked List LIFO yang berupa catatan data terstruktur, dapat dituliskan dengan sejumlah catatan murid tentang Nomor Induk Mahasiswa dan Nama Mahasiswanya. Satu Nomor Induk Mahasiswa (NIM) dan satu Nama Mahasiswa yang bersangkutan ditampung didalam satu struktur.

Realisasi pemasukan data struktur secara LIFO dapat berwujud sebagai berikut:

Memasukkan data murid.

Jumlah murid? : 4

NIM : 2188009

Nama : Iji Pertama

NIM : 2288009

Nama : Dwi Kedua

NIM : 2388009

Nama : Tri Ketiga

NIM : 2488009

Nama : Opat Keempat

Jika data murid itu dimasukkan secara LIFO, kemudian dibaca dari memori dan disajikan kkembali di layar maka hasilnya akan menyajikan data terakhir menjadi urutan sajian pertama, diakhiri dengan data pertama menjadi urutan sajian terakhir sebagai berikut:

NIM	Nama
2488009	Opat Keempat
2388009	Tri Ketiga
2288009	Dwi Kedua
2188009	Iji Pertama

Berkat adanya elemen pointer didalam masing-masing struktur, maka array struktur bukan array statik melainkan array struktur dinamik.

Jumlah array dapat secara dinamik mengikuti kebutuhan pemasukan data. Pelaksanaan program berwujud dinamik mengikuti jumlah struktur pencatat data terstruktur yang dibutuhkan, dan dapat dituliskan sebagai berikut:

Memasukkan Data Murid.

NIM : 2188009

Nama : Iji Pertama

Dilanjutkan? (Y/T): Y

NIM : 2288009

Nama : Dwi Kedua

Dilanjutkan? (Y/T): Y

NIM : 2388009

Nama : Tri Ketiga

Dilanjutkan? (Y/T): Y

NIM : 2488009

Nama : Opat Keempat

Dilanjutkan? (Y/T): T

Hasil memasukkan disajikan kembali di layar:

NIM	Nama
2188009	Iji Pertama
2288009	Dwi Kedua
2388009	Tri Ketiga
2488009	Opat Keempat

Program sederhana untuk mewujudkan pelaksanaan itu dapat disusun menggunakan cara Singly Linked List, yaitu menggunakan struktur dengan salah satu elemennya (satu member) berupa pointer.

Pertama kali agar mudah dimengerti, maka susunan program dibuat non-modular terlebih dahulu. Simaklah program non-modular ini hingga dipahami benar langkah-langkah penjalinan Linked List. Susunan program berikutnya dikembangkan, dimodifikasi menjadi program modular menggunakan sejumlah vvoid function dan function yang sesuai, dan variabel pointer ke struktur dideklarasikan global.

Setelah itu dikembangkan lagi menjadi program modular menggunakan deklarasi variael pointer ke struktur ditempatkan pada deklarasi lokal. Susunan ini merupakan susunan canggih untuk melatih beriap ke arah kemampuan modularsasi dan strukturisasi penyusunan program.

Deklarasi untuk menampung struktur data perihal catatan murid dengan unsur NIM dan Nama seperti tersebut diata, sukup sederhana, yaitu:

```
struct catatan_murid
     {
          Unsigned long nomor_pokok;
          Char nama[50];
          Struct  catatan_murid *next;
     }
```

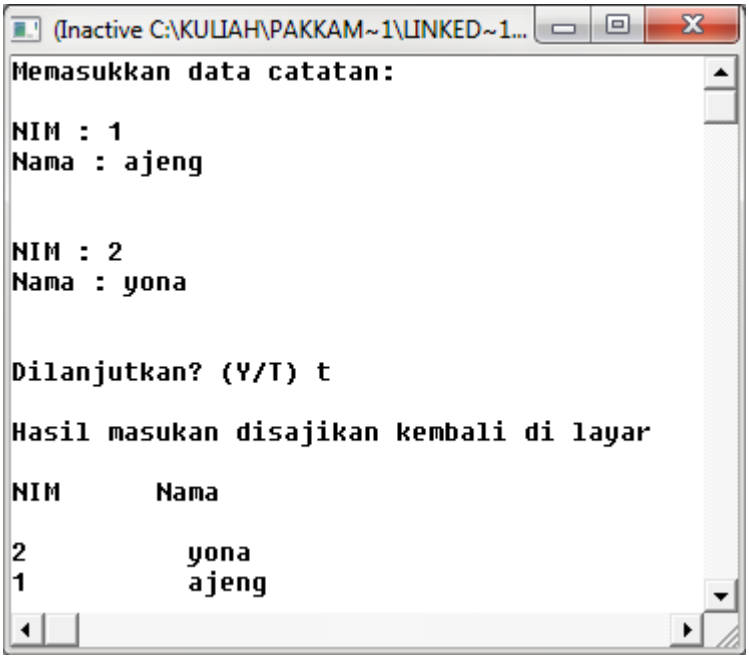
Dengan demikian terjalin pemasukan struktur data Structure catatan_murid dalam bentuk kaitan LIFO. Kaitan Linked List LIFO seringkali juga disebut STACK. Adapun susunan program seutuhnya sebagai berikut:

Contoh LIFO_1 :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
#define YA 1
#define TIDAK 0
#define NULL 0
struct catatan_murid
{
     char Nama[80];
     unsigned long NIM;
```

```
     struct catatan_murid *next;
};
main()
{
     struct catatan_murid *tampung;
     struct catatan_murid *ujung;
     char *p;
     int jawab;
     puts("Memasukkan data catatan: \n");
     ujung    =    (struct    catatan_murid*)malloc(sizeof(struct
catatan_murid));
          printf("NIM : "); scanf("%lu", &ujung->NIM);
          printf("Nama : "); scanf("%s", &ujung->Nama);
          ujung->next = NULL;
          tampung = ujung;
          puts("\n");

     while (YA)
     {
     ujung    =    (struct    catatan_murid*)malloc(sizeof(struct
catatan_murid));
          ujung->next = tampung;
          printf("NIM : "); scanf("%lu", &ujung->NIM);
          printf("Nama : "); scanf("%s", &ujung->Nama);
          tampung=ujung;
          puts("\n");
     printf("Dilanjutkan? (Y/T) ");
     jawab = getche(); puts("\n");
          if((jawab=='T')||(jawab=='t')) break;
     else if ((jawab=='Y')||(jawab=='y')) continue;
     else
     {
          puts("Jawaban harus (Y,y) atau (T,t)\n");
          continue;
     }
     }
     puts("Hasil masukan disajikan kembali di layar\n");
     puts("NIM      Nama\n");
     while(ujung!=NULL)
     {
          p=&(ujung->Nama[0]);
          printf("%-10lu %s\n", ujung->NIM,p);
          ujung=ujung->next;
     }
}
```

Hasilnya :



Hal yang sangat perlu disimak dalam contoh program tersebut adalah penggunaan 2 variabel pointer TAMPUNG dan UJUNG sebagai sarana. Pemilihan jumlah variable pointer diperbolehkan berapa saja sesuai dengan kebutuhan, dan dalam penyelesaian Linked List LIFO itu cukup dipergunakan 2 variabel pointer saja.

Program non modular hanya sekedar contoh agar masalahnya dapat dimengerti. Susunan program berbentuk non-modular, akan susah dan payah jika menggarap masalah yang besar. Susunannyamenjadi ruwet dan banyak menggunakan GOTO kemana saja asal dapat menghimpun susunan penyelesaian masalah.

Oleh karena itu, sejak masalahnya masih sederhana, maka penyelesaiannya harus telah menggunakan susunan progam modular menggunakan function dan void function yang sesuai, dilengkapi dengan pass parameter (pass argumen) yang teapt dan efisien. Susunan program modular untuk menyelesaikan perihal tersebut diatas berupa modifikasi program LIFO_1.C menjadi progam LIFO_2.C menggunakan deklarasi global bagi pointer variable TAMPUNG dan UJUNG.

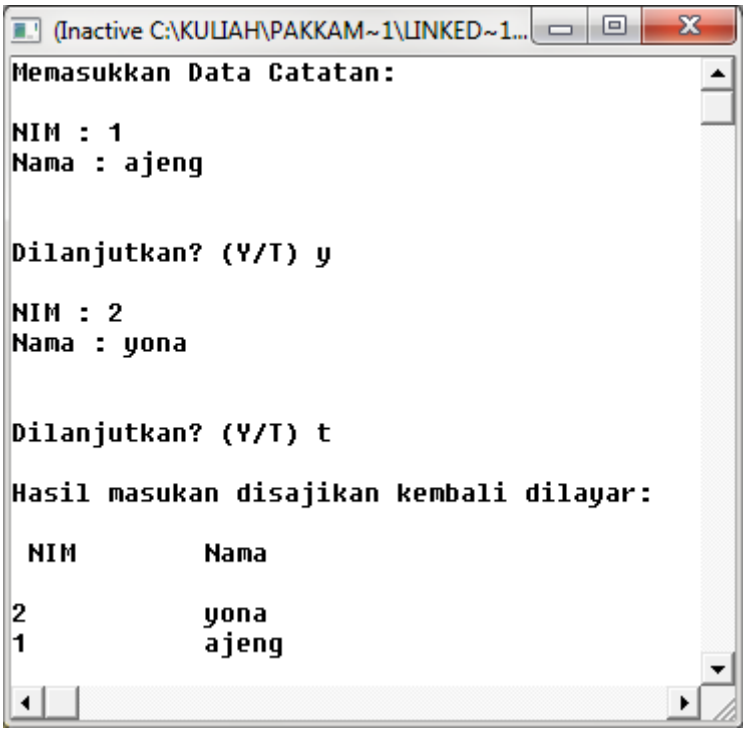
Contoh LIFO_2 :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
#define YA 1
#define TIDAK 0
#define NULL 0
struct catatan_murid
{
char nama[80];
unsigned long nim;
```

```
struct catatan_murid *next;
};
struct catatan_murid *tampung;
struct catatan_murid *ujung;

struct catatan_murid *catat()
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     puts("\n");
     if(tampung==0)
     {
          ujung->next=NULL;
          tampung= ujung;
     }
     else
     {
          ujung->next = tampung;
          tampung = ujung;
     }
     return (tampung);
}
dilanjutkan_ya_tidak()
{
     char jawab;                                                    ;
     int j;
     printf("Dilanjutkan? (Y/T) ");
          jawab = getche(); puts("\n");
     if((jawab=='T')||(jawab=='t'))j=0;
     else if((jawab=='Y')||(jawab=='y'))j=1;
     else
     {
          puts("Jawaban harus (Y,y) atau (T,t)\n");
          j=1;
     }
     return(j);
     }
void sajikan_hasil()
{
     char *p;
     puts(" NIM        Nama\n");
     while (ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu  %s\n", ujung->nim, p);
          ujung=ujung->next;
     }
}
main()
{
     int jawab;
     puts("Memasukkan Data Catatan: \n");
          tampung=NULL;
          jawab=YA;
     while (jawab==YA)
     {
          ujung   =   (struct   catatan_murid*)malloc(sizeof(struct
catatan_murid));
          tampung=catat();
          jawab = dilanjutkan_ya_tidak();
     }
```

```
    puts("Hasil masukan disajikan kembali dilayar: \n");
    sajikan_hasil();
}
```

Hasilnya :



Program LIFO_2.C memberi hasil pelaksanaan identic dengan hasil pelaksanaan susunan program non-modular LIFO_1.C dan tampak lebih mudah dkembangkan menjadi susunan program yang lebih besar mengarah ke Data Base.

Perhatikan kemudahan yang diakibatkan oleh deklarasi global bagi variable pointer TAMPUNG dan UJUNG. Emudahan itu terlihat bahwa antar function dengan main () tidak diperlukan pass parameter (pass argument). Namun kemudahan itu tidak terlepas dari kerawanan variable global yang secara beramai-ramai digunakan langsung oleh function dan void function yang ada.

Kerawanan tersebut dapat dicegah dengan deklarasi lokal bagi variabel pointer TAMPUNG dan UJUNG. Akan tetapi akibat dari deklarasi lokal ini memerlukan pass parameter antar function dengan main () yang harus dipilih setepat-tepatnya.

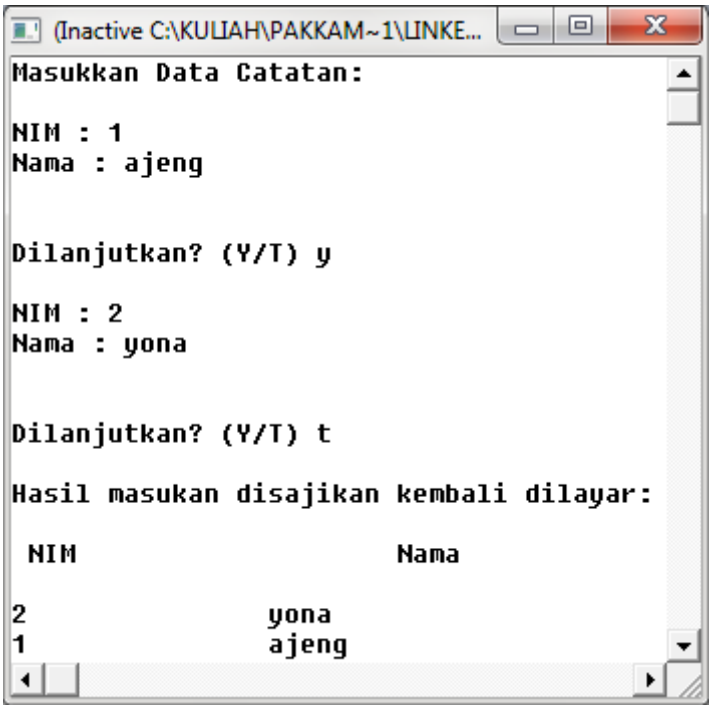
Contoh LIFO_3 :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<ctype.h>
#define YA 1
#define TIDAK 0
#define NULL 0

struct catatan_murid
{
```

```
     char nama[80];
     unsigned long nim;
     struct catatan_murid *next;
};
struct catatan_murid *catat(struct catatan_murid *ujung, struct
catatan_murid *tampung)
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     puts("\n");
     if(tampung==0)
     {
          ujung->next=NULL;
          tampung=ujung;
     }
     else
     {
          ujung->next = tampung;
          tampung = ujung;
     }
     return(tampung);
}

dilanjutkan_ya_tidak()
{
     char jawab;
     int j;
     printf("Dilanjutkan? (Y/T) ");
     if ((toupper(jawab=getche()))=='T')j=0;
     else if(toupper(jawab)=='Y') j=1;
     else
     {
          puts("\nJawaban harus (Y,y) atau (T,t)\n");
          j=1;
     }
     puts("\n");
     return(j);
}

void sajikan_hasil(struct catatan_murid *ujung)
{
     char *p;
     puts(" NIM              Nama\n");
     while (ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu%s\n", ujung->nim, p);
          ujung=ujung->next;
     }
}
main()
{
     struct catatan_murid *tampung;
     struct catatan_murid *ujung;
     int jawab;
     puts("Masukkan Data Catatan: \n");
     tampung=NULL;
     jawab = YA;
     while(jawab==YA)
     {
```

```
          ujung=(struct        catatan_murid*)malloc(sizeof(struct
catatan_murid));
          tampung = catat(ujung, tampung);
          jawab = dilanjutkan_ya_tidak();
     }
     puts("Hasil masukan disajikan kembali dilayar: \n");
     sajikan_hasil(ujung);
}
```

Hasilnya :



Function dan void function beserta pass argumen-nya perlu dikaji lebih cermat, terutama bentuk syntax dan jalinan informasi yang dikandungnya.

Terutama bentuk pointer ke function :

```
Struct catatan_murid *catat
     {
          Struct catatan_murid *UJUNG;
          Struct catatan_murid *TAMPUNG;
     }
```

Function tersebut adalah *catat (), sedangkan catat adalah pointernya, dan pointer catat inti bertype pointer ke structure catatan_murid. Oleh karena itu function *catat () menghasilkan (mengembalikan) harga berupa pointer pada :

return (TAMPUNG) ;

Jika kita menginginkan agar suatu function untuk mengembalikan pointer ke structure, maka perlu digunakan pointer ke function dengan struktur pointer ke structure. Hal yang perlu dikerjakan hanya menyusun syntax yang benar.

Agar lebih jelas simaklah kembali :

Struct catatan_murid *catat

{

```
     struct catatan_murid *UJUNG;
     struct catatan_murid *TAMPUNG;
}
{
     cputs("NIM    :"); scanf("%lu", &UJUNG->NIM);
     cputs("Nama   :"); cgets(UJUNG->Nama);
     puts("\n");
     if(TAMPUNG==0)
          {
               UJUNG->Next = NULL;     TAMPUNG=UJUNG;
          }
     Else
          {
               UJUNG->Next=TAMPUNG;    TAMPUNG=UJUNG;
          }
     return (TAMPUNG);
}
```

Tujaun 2 Setelah bekerja melalui modul ini, mahasiswa diharapkan dapat membuat, mengkompile dan menjalankan proghram C sederhana menggunakan Singly Linked List FIFO

SINGLY LINKED LIST FIFO

Linked List LIFO pada umumnya terasa janggal hasilnya karena bilamana dibaca dan dituliskan kembali, structure terakhir disajikan pada posisi pertama (teratas) dan structure paling awal disajikan pada posisi terakhir (terbawah).

Linked List FIFO akan dapat memberi hasil yang tidak janggal karena structure pertama yagn masuk akan dapat dimunculkan menjadi structure pertama di posisi teratas bilamana dibaca dan disajikan. Linked Listed FIFO seringkali juga disebut QUEUE.

Meskipun demikian diperlukan pengisian yang tepat pada elemen pointer *Next dalam menjalin kaitan yang diinginkan. Sebagai kelanjutan contoh tentang bagaimana LIFO itu, perlu disimak kembali pemasukan catatan murid dengan unsur NIM dan Nama mahasiswa seperti pada contoh program LIFO.

Secara sepintas pelaksanaan pemasukan data tetap sama seperti pada pemasukan data LIFO, akan tetapi penggarapan kaitan FIFO antar structure sangat berbeda. Hasilnya penggarapan kaitan FIFO dapat memberikan pembacaan berupa structure pertama yang masuk (First In) dapat disajikan menjadi struktur pertama yang berposisi paling awal (First Out).

Hasil urutan pembacaan dan penyajian dapat tetap seperti pada urutan pemasukan data-data structure-nya, seperti memperlakukan structure sebagai antrian. Oleh

sebab itu lazim disebut QUEUE dan tampak lebih wajar dalam urutan, tidak janggal seperti pada kaitan LIFO yang lazim disebut STACK.

Contoh realisasi pemasukan data dan penyajiaannya kembali sebagai berikut:

Memasukkan Data Murid.

NIM : 2188009

Nama : Iji Pertama

Dilanjutkan? (Y/T): Y

NIM : 2288009

Nama : Dwi Kedua

Dilanjutkan? (Y/T): Y

NIM : 2388009

Nama : Tri Ketiga

Dilanjutkan? (Y/T): Y

NIM : 2488009

Nama : Opat Keempat

Dilanjutkan? (Y/T): T

Hasil memasukkan disajikan kembali di layar:

NIM	Nama
2188009	Iji Pertama
2288009	Dwi Kedua
2388009	Tri Ketiga
2488009	Opat Keempat

Dari gambaran itu tampak bahwa pointer ke structure_1 harus diketahui secara pasti menggunakan pointer AWAL. Demikian pula elemen pointer *Next pada structure_4 sebagai structure ujung akhir, harus diisi NULL sehingga secara pasti diketahui bahwa structure_4 adalah ujung akhir.

Pointer AWAL bertugas sangat khusus, yaitu hanya mencatat address structure paling pertama saja, agar structure awal dapat selalu diketahui secara pasti bilamana diperlukan sebagai referensi awal dari kaitan Linked List.

Dengan demikian untuk mencatat address structure awal diperlukan pointer AWAL dan untuk mencatat address structure ujung diperlukan pointer UJUNG. Namun demikian karena jumlah structure secara dinamik mengikuti kebutuhan, maka UJUNG selalu bergerak, karena structure dapat bertambah atau dikurangi.

Dalam Singly Linked List FIFO ini pointer AWAL memang hanya mencatat address structure_1 (struktur paling pertama) saja. Pointer UJUNG dipilih menjadi variabel pointer struktur UJUNG agar dapat mengikuti pertambahan posisi struktur terujung. Kemudian pointer TAMPUNG akan secara sementara menampung struktur UJUNG menyongsong adanya pertambahan struktur baru yang akan menjadi struktur UJUNG yang baru. Hal yang sulit dimengerti ini sulit pula menerangkannya, apalagi menjelaskan hanya berdasar kalimat.

Perihal sangat penting adalah pengisian elemen pointer *Next pada setiap struktur. Pengisian ini harus menjalin kaitan address antara struktur yang satu dengan struktur lainnya, sehingga terjadi kaitan address berikutan.

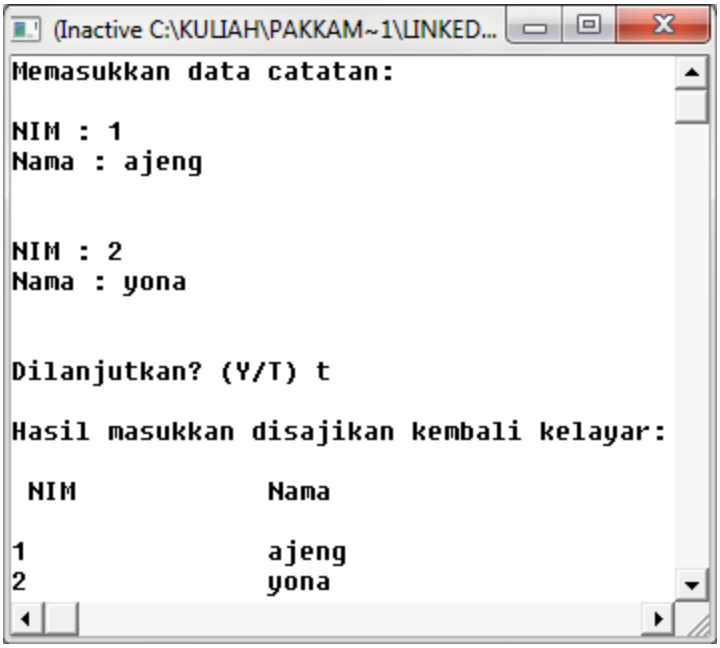
Contoh FIFO_1 :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
#define YA 1
#define TIDAK 0
#define NULL 0
struct catatan_murid
{
     char nama[80];
     unsigned long nim;
     struct catatan_murid *next;
};
main()
{
     struct catatan_murid *awal;
     struct catatan_murid *tampung;
     struct catatan_murid *ujung;
     char *p;
     int jawab;
```

```
     puts("Memasukkan data catatan: \n");
     awal=(struct              catatan_murid*)malloc(sizeof(struct
catatan_murid));
          printf("NIM : "); scanf("%lu", &awal->nim);
          printf("Nama : "); scanf("%s", &awal->nama);
          tampung=awal;
          tampung->next=awal->next;
          puts("\n");
     while (YA)
     {
          ujung=(struct        catatan_murid*)malloc(sizeof(struct
catatan_murid));
               tampung->next = ujung;
               printf("NIM : "); scanf("%lu", &ujung->nim);
               printf("Nama : "); scanf("%s", &ujung->nama);
               ujung->next=NULL;
               tampung=ujung;
               puts("\n");
          printf("Dilanjutkan? (Y/T) ");
          jawab=getche();puts("\n");
          if((jawab=='T')||(jawab=='t'))break;
          else if((jawab=='Y')||(jawab=='y'))continue;
          else
          {
               puts("Jawaban harus (Y,y) atau (T,t)\n"); continue;
          }
       }
     puts("Hasil masukkan disajikan kembali kelayar: \n");
     puts(" NIM         Nama\n");
     ujung=awal;
     while(ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu%s\n", ujung->nim, p);
```

```
            ujung=ujung->next;
        }
}
```

Hasilnya :



Contoh hasil pelaksaan eksekusi program, akan tepat berbentuk seperti yang telah dilukiskan diatas.

Program FIFO_1.C diawali dengan mengisi data pada struktur pertama yaitu struktur yang addressnya akan dicatat oleh pointer AWAL sebagai berikut :

```
    awal=(struct catatan_murid*)malloc(sizeof(struct catatan_murid));
        printf("NIM : "); scanf("%lu", &awal->nim);
        printf("Nama : "); scanf("%s", &awal->nama);
        tampung=awal;
        tampung->next=awal->next;
```

Agar dapat dikaitkan dengan struktur berikutnya, ditolong dengan pointer TAMPUNG, yaitu dengan menjadikan isi TAMPUNG = AWAL. Jika TAMPUNG -> Next diisi dengan isi pointer UJUNG maka struktur AWAL akan mengait ke struktur berikutnya :

```
while (YA)
    {
```

```
          ujung=(struct        catatan_murid*)malloc(sizeof(struct
catatan_murid));
               tampung->next = ujung;
               printf("NIM : "); scanf("%lu", &ujung->nim);
               printf("Nama : "); scanf("%s", &ujung->nama);
               ujung->next=NULL;
               tampung=ujung;
               puts("\n");
```

Demikian seterusnya, dan elemen pointer struktur ujung akhir harus diisi NULL seperti pada pernyataan UJUNG -> Next = NULL.

Namun anda tidak boleh puas hanya dengan program non modular saja, karena bila demikian maka anda tidak akan mampu menggarap masalah-masalah besar. Oleh karena itu maka program FIFO_1.C tersebut dimodifikasi menjadi modular dan sebagai tahapan termudah adalah menggunakan deklarasi global bagi pointer AWAL, TAMPUNG, dan UJUNG.

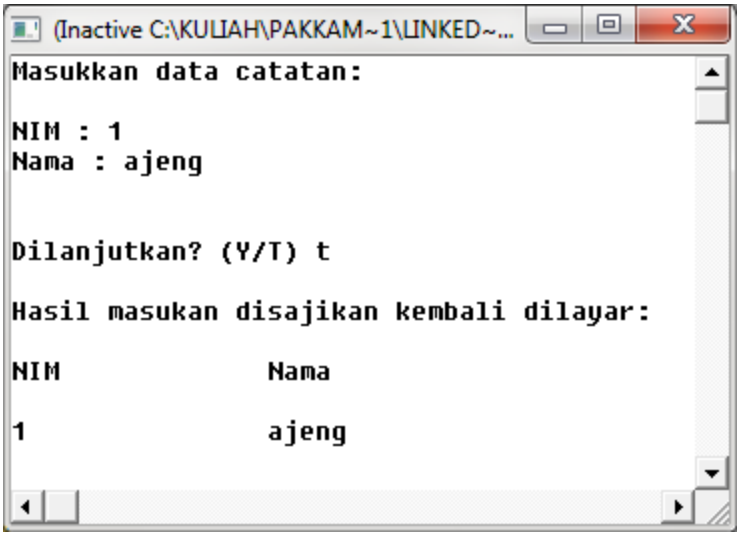
Contoh FIFO_2 :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#define YA 1
#define TIDAK 0
#define NULL 0
struct catatan_murid
{
     char nama[80];
     unsigned long nim;
     struct catatan_murid *next;
};

struct catatan_murid *awal;
struct catatan_murid *tampung;
struct catatan_murid *ujung;
```

```
struct catatan_murid *catat()
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     puts("\n");
     ujung->next=NULL;
     if(awal==0)
     {
          awal=ujung; tampung=awal;
     }
     else
     {
          tampung->next=ujung;tampung=ujung;
     }
     return (tampung);
}
dilanjutkan_ya_tidak()
{
     char jawab;
     int j;
     printf("Dilanjutkan? (Y/T) ");
     jawab=getche(); puts("\n");
     if((jawab=='T')||(jawab='t')) j=0;
     else if((jawab=='Y')||(jawab=='y')) j=1;
     else
     {
          puts("Jawaban harus (Y,y) atau (T,t)\n"); j=1;
     }
     return (j);
}
void sajikan_hasil()
{
     char *p;
```

```
     puts("NIM     Nama\n");
     ujung=awal;
     while(ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu%s\n", ujung->nim, p);
          ujung=ujung->next;
     }
}
main()
{
     int jawab;
     puts("Masukkan data catatan: \n");
     awal=NULL;
     jawab=YA;
     while(jawab==YA)
     {
          ujung=(struct         catatan_murid*)malloc(sizeof(struct
catatan_murid));
          tampung=catat();
          jawab=dilanjutkan_ya_tidak();
          puts("Hasil masukan disajikan kembali dilayar: \n");
          sajikan_hasil();
     }
}
```

Hasilnya :



Agar susunan program menjadi lebih andal menghadapi lingkungan pemrograman lebih kompleks, maka program FIFO_2 dapat dimodifikasi berbentuk tetap modular dan menggunakan deklarasi variabel pointer AWAL, TAMPUNG, dan UJUNG menjadi variabel local di main ().

Pass argument (pas parameter) hanya bersyntax agak rumit akan tetapi mekanisme kerjanya sangat sederhana. Jalan kerjanya hanya mengirim address yang dikandung pointer. Susunan program seutuhnya sebagai berikut :

Contoh FIFO_3 :

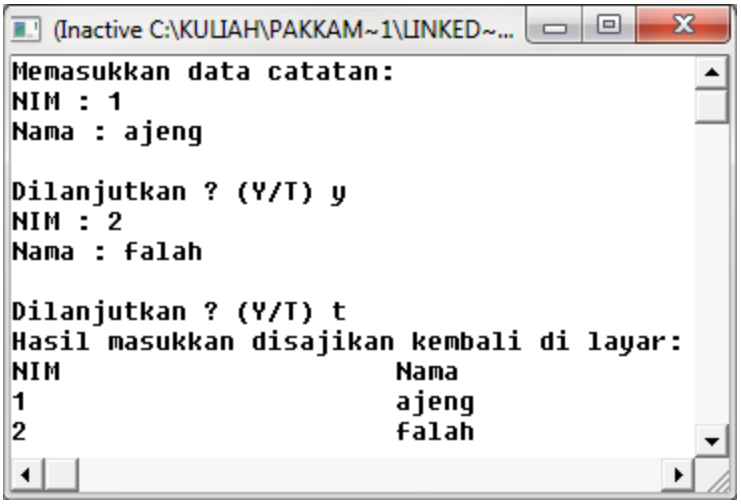
```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<ctype.h>

#define YA 1
#define TIDAK 0
#define NULL 0

struct catatan_murid
{
     char nama[80];
     unsigned long nim;
```

```
     struct catatan_murid *next;
};


struct catatan_murid *catat(struct catatan_murid *tampung, struct
catatan_murid *ujung)
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     printf("\n");
     ujung->next=NULL;
     if(tampung==NULL)
     {
          tampung = ujung;
     }
     else
     {
          tampung->next=ujung;
          tampung=ujung;
     }
     return (tampung);
}
dilanjutkan_ya_tidak()
{
     char jawab;
     int j;
     printf("Dilanjutkan ? (Y/T) ");
     if ((toupper(jawab=getche()))=='T')j=0;
     else if (toupper(jawab)=='Y')j=1;
     else
     {
          printf("\nJawaban harus (Y,y) atau (T,t)\n");
          j=1;
     }
```

```
     printf("\n");
     return (j);
}
void    sajikan_hasil(struct    catatan_murid    *ujung,    struct
catatan_murid *awal)
{
     char *p;
     printf("NIM             Nama\n");
     ujung=awal;
     while(ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu     %s\n", ujung->nim, p);
          ujung=ujung->next;
     }
}
main()
{
     struct catatan_murid *awal;
     struct catatan_murid *tampung;
     struct catatan_murid *ujung;
     int jawab;
     printf("Memasukkan data catatan: \n");
     awal=NULL;
     tampung=NULL;
     jawab=YA;
     while(jawab==YA)
     {
          ujung=(struct    catatan_murid    *)malloc(sizeof(struct
catatan_murid));
          if (awal==NULL)
          {
               awal=catat(tampung, ujung);
               tampung=awal;
```

```
          }
          else tampung=catat(tampung, ujung);
          jawab = dilanjutkan_ya_tidak();
     }
     printf("Hasil masukkan disajikan kembali di layar: \n");
     sajikan_hasil(ujung, awal);
   return 0;
}
```

Hasilnya :



Tujuan 3 Menghapus dan menyisipkan data menggunakan Linked List

MENGHAPUS DAN MENYISIPI

Setiap kali diperlukan menampung data, memori penampung dapat dipesan menggunakan malloc, asal pointernya telah dideklarasikan pada tubuh program baik deklarasi global maupun deklarasi lokal. Perihal pemesanan memori bagi struktur yang dioperasikan menggunakan pointer telah berulang kali digunakan malloc ( ) pada contoh-contoh program LIFO dan FIFO untuk menjalin Linked List.

Kebalikan malloc ( ) adalah free ( ). Artiny, free ( ) akan membuat isi pointer menjadi NULL dan memori yang dicakup oleh penunjukkan pointer tersebut menjadi bebas digunakan oleh pemakai lain.

Ambillah contoh program FIFO_3. C dengna contoh pemasukan data sebagai berikut :

Memasukkan Data Murid.
NIM : 2188009
Nama : Iji Pertama
Dilanjutkan? (Y/T): Y
NIM : 2288009
Nama : Dwi Kedua
Dilanjutkan? (Y/T): Y
NIM : 2388009
Nama : Tri Ketiga
Dilanjutkan? (Y/T): Y
NIM : 2488009
Nama : Opat Keempat
Dilanjutkan? (Y/T): T
Hasil memasukkan disajikan kembali di layar:

NIM	Nama
2188009	Iji Pertama
2288009	Dwi Kedua
2388009	Tri Ketiga
2488009	Opat Keempat
Menghapus salah satu catatan. Pilihan :	
H atau h :	Salah satu catatan
E atau e :	Selesai dan exit
H	
NIM murid yang catatannya akan dihapus NIM: 23880009 Hasilnya:	
NIM	Nama
2188009	Iji Pertama

2288009	Dwi Kedua
2488009	Opat Keempat

Penghapusan satu struktur ddidalam deretan struktur yang berkaitan linked list harus menjada kaitan pointernya agar jalinan kaitan itu tidak rusak.

Langkah penghapusan menggunakan pointer UJUNG dan pointer P keduanya mencatat address_structure_2, iyaitu UJUNG = P = address_structure_2

Dengan P = P -> Next berarti P mencatat address_structure_3

Kemudian UJUNG -> Next = P -> Next berarti mengutipkan address_structure_4 kedalam *Next structure_2

Setelah itu baru dapat dilakukan penghapusan structure_e dengan free (P) dan jalinan linked list tetap terkait dari structure_1 ke structure_2, dari structure_2 ke structure_4

Function seutuhnya harus dapat mencari mulai dari structure_1 hingga struktur terakhir. Oleh karena itu penggarapannya harus menggunakan loop.

Akan tetapi sebelum loop dilakukan, terlebih dahulu harus diperiksa apakah structure_1 yang akan dihapus. Oleh karena itu structure_1 yang dicatat addressnya menggunakan pointer AWAL mendapat perlakuan khusus, karena posisi awal sangat menentukan.

Dengan demikian susunan program harus dua tahap. Tahap pertama menggarap structure_1, karena memang harus digarap secara khusus. Tahap kedua membuat loop pencarian dan penghapusan struktur yang diinginkan.

Susunan function seutuhnya sebagai berikut :

```
hapus_1_catatan(struct catatan_murid *ujung)
{
     unsigned long h;
     struct catatan_murid *p;
     printf("\nNIM murid yang catatannya akan dihapus(?): \n");
     scanf("%lu", &h);
```

```
    if(awal->nim==h)
    {
        p=awal->next;
        free(awal);
        awal=p;
    }
    else
    {
        p=awal;
        while(p!=NULL)
        {
            ujung=p;
            p=p->next;
            if(p->nim==h)
            {
                ujung->next=p->next;
                free(p); p=ujung->next;
            }
        }
    }
    return 0;
}
```

Agar supaya pelaksanaan program penghapusan cukup akrab dan mudah digunakan, dipilih menggunakan menu :

Menghapus salah satu catatan.

Pilihan :

H atau h : hapus satu catatan

E atau e : selesai dan exit

Jika diketikkan huruf H atau huruf h akan muncul pertanyaan:

NIM yang akan dihapus(?) :

Pertanyaan ini harus dijawab dengan mengetikkan angka NIM, maka kemudian NM dan Nama yang disebut akan dihapus. Secara sepintas akan tampak bahwa penghapusan itu sangat mudah.

Kemudahan itu memang benar asal cara-cara katian pengisian elemen pointer *Next dikuasai secara mantap. Susunan program seutuhnya adalah berupa pengembangan program FIFO_3.C ditambagi function Menu_Hapus, function Pilihan, dan Function Hapus_1_Catatan.

Contoh FIFO_H :

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

#define YA 1
#define TIDAK 0
#define NULL 0

struct catatan_murid
{
     char nama[80];
     unsigned long nim;
     struct catatan_murid *next;
};
struct catatan_murid *awal;
struct catatan_murid *catat(struct catatan_murid *tampung, struct
catatan_murid *ujung)
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     printf("\n");
     ujung->next=NULL;
     if (tampung==NULL)
```

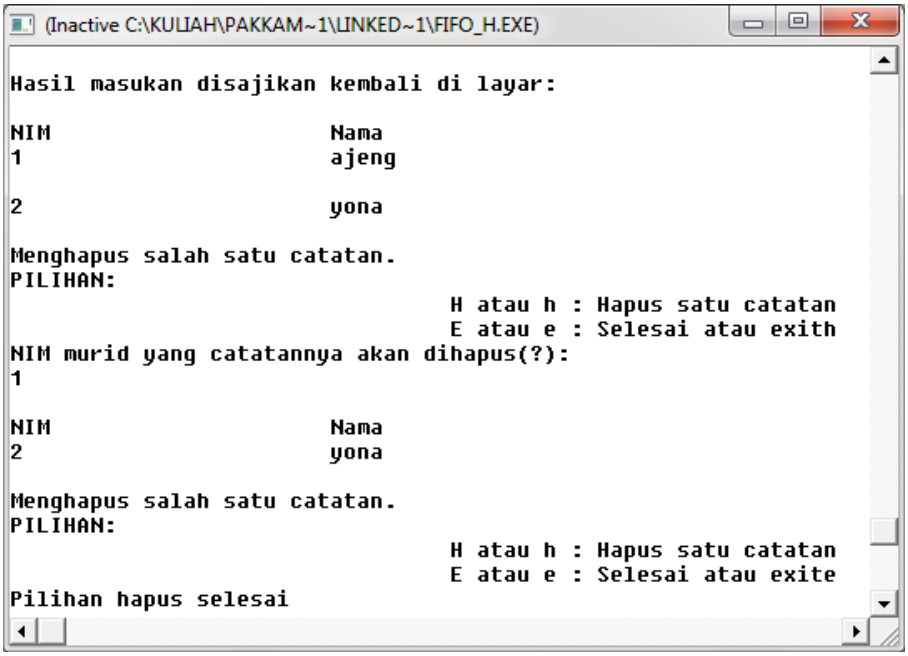
```
     {
          tampung=ujung;
     }
     else
     {
          tampung->next=ujung;
          tampung=ujung;
     }
     return (tampung);
}
dilanjutkan_ya_tidak()
{
     char jawab;
     int j;
     printf("Dilanjutkan? (Y/T) ");
     if((toupper(jawab=getche()))=='T')j=0;
     else if (toupper(jawab)=='Y')j=1;
     else
     {
          printf("\nJawaban harus (Y,y) atau (T,t)n");
          j=1;
     }
     printf("\n");
     return(j);
}
void sajikan_hasil(struct catatan_murid *ujung)
{
     char *p;
     printf("\nNIM           Nama");
     ujung=awal;
     while(ujung!=NULL)
     {
          p=&(ujung->nama[0]);
```

```
          printf("\n%-10lu        %s\n", ujung->nim, p);
          ujung=ujung->next;
     }
}
hapus_1_catatan(struct catatan_murid *ujung)
{
     unsigned long h;
     struct catatan_murid *p;
     printf("\nNIM murid yang catatannya akan dihapus(?): \n");
     scanf("%lu", &h);
     if(awal->nim==h)
     {
          p=awal->next;
          free(awal);
          awal=p;
     }
     else
     {
          p=awal;
          while(p!=NULL)
          {
               ujung=p;
               p=p->next;
               if(p->nim==h)
               {
                    ujung->next=p->next;
                    free(p); p=ujung->next;
               }
          }
     }
     return 0;
}
void menu_hapus()
```

```
{
     printf("\nMenghapus salah satu catatan.");
     printf("\nPILIHAN: ");
     printf("\n                    H   atau   h   :   Hapus   satu
catatan");
     printf("\n                    E atau e : Selesai atau exit");
}
void pilihan(struct catatan_murid *ujung)
{
     char p;
     while((tolower(p=getche()))!='e')
     {
          switch(p)
          {
               case 'h' :
               {
                    hapus_1_catatan(ujung);
                    sajikan_hasil(ujung);
                    menu_hapus(); break;
               }
               default :
               {
                    printf("\nJawaban harus (H,h) atau (E,e)\n");
                    break;
               }
          }
     }
     printf("\nPilihan hapus selesai\n");
}
main()
{
     struct catatan_murid *tampung;
     struct catatan_murid *ujung;
```

```
     int jawab;


     printf("\nMemasukkan data catatan : \n");
     awal=NULL;
     tampung=NULL;
     jawab=YA;
     while(jawab==YA)
     {
          ujung=(struct    catatan_murid    *)malloc(sizeof(struct
catatan_murid));
          if (awal==NULL)
          {
               awal=catat(tampung, ujung);
               tampung=awal;
          }
          else tampung=catat(tampung, ujung);
          jawab=dilanjutkan_ya_tidak();
     }
     printf("\nHasil masukan disajikan kembali di layar: \n");
     sajikan_hasil(ujung);
     menu_hapus();
     pilihan(ujung);
     return 0;
}
```

Hasilnya :



Program FIFO_3.C juga dapat dikembangkan untuk melayani sisipan struktur kedalam deretan struktur yang telah terjalin pada kaitan linked list. Contoh realisasi pelaksanaan dapat dilukiskan sebagai berikut :

Memasukkan Data Murid.

NIM : 2188009

Nama : Iji Pertama

Dilanjutkan? (Y/T): Y

NIM : 2288009

Nama : Dwi Kedua

Dilanjutkan? (Y/T): Y

NIM : 2388009

Nama : Tri Ketiga

Dilanjutkan? (Y/T): Y

NIM : 2488009

Nama : Opat Keempat

Dilanjutkan? (Y/T): T

Hasil memasukkan disajikan kembali di layar:

NIM	Nama
2188009	Iji Pertama
2288009	Dwi Kedua

2388009	Tri Ketiga
2488009	Opat Keempat
Menyisipkan salah satu catatan. Pilihan :	
S atau s :	Sisipkan satu catatan
E atau e :	Selesai dan exit
S	
Disisipkan setelah NIM: 23880009	
NIM dan Nama murid yang disisipkan: NIM : 2188888 Nama : Tambahan Sisipan	
NIM	Nama
2188009	Iji Pertama
2288009	Dwi Kedua
2388009	Tri Ketiga
2188888	Tambahan Sisipan
2488009	Opat Keempat

Agar supaya dapat menampung sisipan, maka harus memesan struktur baru menggunakan pointer TAMPUNG.

```
TAMPUNG = (struct Catatan_murid *) malloc(size of (*TAMPUNG))
```

Pointer P mencatat address structure_3, sebab setelah structure_3 akan disisipi structure_sisip. Structure_sisip akan mengait ke dalam jalinan Linked List dengan :

TAMPUNG -> Next = -> P->Next;

P->Next = TAMPUNG;

TAMPUNG->Next = P->Next; akan mengutipkan address_structure_4 kedalam TAMPUNG->Nexr sehingga structure sisip mengait ke structure_4

P->Next = TAMPUNG akan mengutipkan address_structure_sisip kedalam P->Next, yaitu *Next_structure_3, sehingga structure_3 mengait ke structure_sisip.

Urutan tersebut sama sekal tidak boleh terbalik atau dibalik, sebab jalinan kaitannya unik, dengan demikian maka structure_3 mengait ke structure_sisip, dan structure_sisip mengait ke structure_4, dan jalinan linked list terjamin benar.

Namun proses pengaitan tersebut harus digabung dengan pencarian dan pemilihan. Maka teknik rekayasa loop harus dicari agar proses pencarian, pemilihan, dan penjalinan sisipan dapta berlangsung diantara sekian jumlah structure yang ada di linked list.

Susunan langkah pernataan program berikut ini akan berbicara sendiri tentang bagaimana proses penyisipan dilaksakan :

```
p=awal;
while(p!=NULL)
{
     if(p->nim==h)
     {
          tampung=(struct                    catatan_murid
*)malloc(sizeof(*tampung));
          printf("NIM dan Nama murid yang disispkan\n");
          printf("NIM : "); scanf("%lu", &tampung->nim);
          printf("Nama : "); scanf("%s", &tampung->nama);
          printf("");
          tampung->next=p->next;
          p->next=tampung;
          p=tampung;
     }
     p=p->next;
```

Susunan program selengkapnya berasal dari program FIFO_3.C dengan tambahan-tambahan function, disesuaikan dengan kebutuhan pemecahan masalahnya.

Contoh FIFO_S :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<ctype.h>
#define YA 1
#define TIDAK 0
#define NULL 0

struct catatan_murid
{
     char nama[80];
     unsigned long nim;
     struct catatan_murid *next;
};
struct catatan_murid *awal;
struct catatan_murid *catat(struct catatan_murid *tampung, struct
catatan_murid *ujung)
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     printf("\n");
     ujung->next=NULL;
     if(tampung==NULL)
     {
          tampung=ujung;
     }
     else
     {
          tampung->next=ujung; tampung=ujung;
     }
     return(tampung);
}
dilanjutkan_ya_tidak()
```

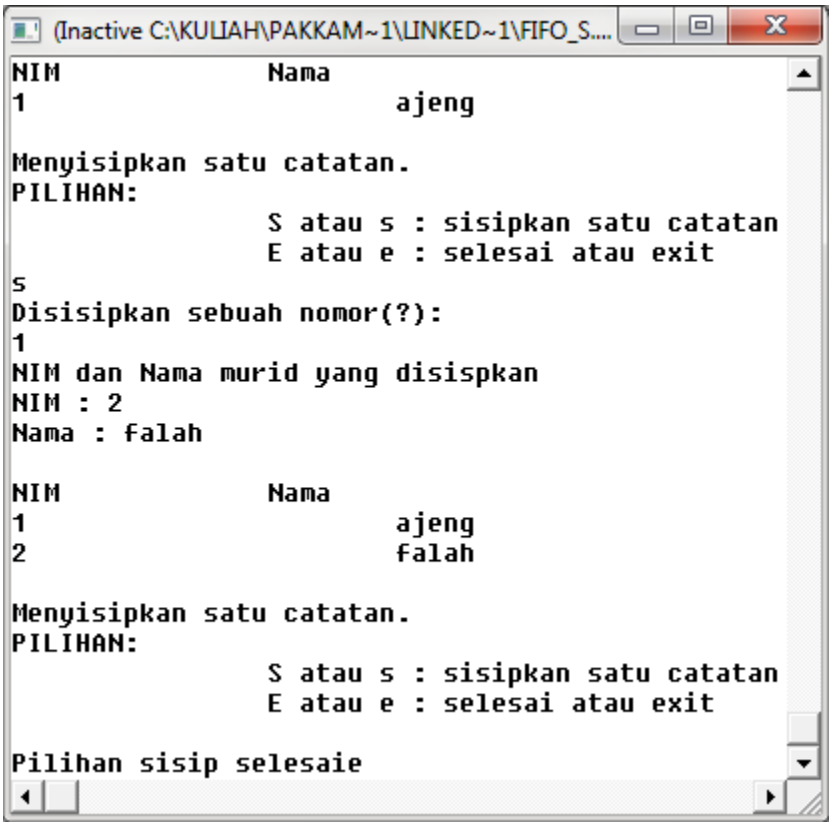
```
{
     char jawab;
     int j;
     printf("DIlanjutkan? (Y/T) ");
     if((toupper(jawab=getche()))=='T')j=0;
     else if(toupper(jawab)=='Y')j=1;
     else
     {
          printf("\nJawaban harus (Y,y) atau (T,t)\n"); j=1;
     }
     printf("\n");
     return(j);
}
void sajikan_hasil(struct catatan_murid *ujung)
{
     char *p;
     printf("\nNIM      Nama\n");
     ujung=awal;
     while(ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu     %s\n", ujung->nim, p);
          ujung=ujung->next;
     }
}
void sisipkan_1_catatan(struct catatan_murid *tampung)
{
     unsigned long h;
     struct catatan_murid *p;
     printf("\n\rDisisipkan  sebuah  nomor(?):  \n");  scanf("%lu",
&h);
     p=awal;
     while(p!=NULL)
```

```
     {
          if(p->nim==h)
          {
               tampung=(struct                         catatan_murid
*)malloc(sizeof(*tampung));
               printf("NIM dan Nama murid yang disispkan\n");
               printf("NIM : "); scanf("%lu", &tampung->nim);
               printf("Nama : "); scanf("%s", &tampung->nama);
               printf("");
               tampung->next=p->next;
               p->next=tampung;
               p=tampung;
          }
          p=p->next;
     }
}
void menu_sisip()
{
     printf("\nMenyisipkan satu catatan.\n");
     printf("PILIHAN:\n");
     printf("      S atau s : sisipkan satu catatan\n");
     printf("      E atau e : selesai atau exit\n");
}
void pilihan(struct catatan_murid *ujung)
{
     char p;
     while((tolower(p=getche()))!='e')
     {
          switch(p)
          {
               case 's':
               {
                    sisipkan_1_catatan(ujung);
```

```
                    sajikan_hasil(ujung);
                    menu_sisip(); break;
               }
               default:
               {
                    printf("\njawaban harus (S,s) atau (E,e)\n");
break;
               }
          }
     printf("\nPilihan sisip selesai");
     }
}
main()
{
     struct catatan_murid *tampung;
     struct catatan_murid *ujung;
     int jawab;
     printf("\nMemasukkan data catatan: \n");
     awal=NULL;
     tampung=NULL;
     jawab=YA;
     while (jawab==YA)
     {
          ujung=(struct    catatan_murid    *)malloc(sizeof(struct
catatan_murid));
          if(awal==NULL)
          {
               awal=catat(tampung,ujung); tampung=awal;
          }
          else tampung=catat(tampung, ujung);
          jawab=dilanjutkan_ya_tidak();
     }
     printf("Hasil masukan disajikan kembali di layar: \n");
     sajikan_hasil(ujung);
```

```
     menu_sisip();

     pilihan(tampung);

   return 0;

}
```

Hasilnya :



Program FIFO_H.C yang melayani pemasukan dan penghapusan struktur, dapat diolah dan digabung dengan program FIFO_S.C untuk melayani penyisipan struktur sekaligus. Olaha gabungan program itu dijadkan program FIFO_HS.C untuk melayani pemasukan data Linked List struktur, menghapus struktur dan menyisipkan struktur.

Disini dengan jelas dapat disimak tentang bagaimana mengembangkan program yang pada awalnya masih kecil dan sederhana, kemudian dikembangkan menjadi program makin besar dan semakin dapat menyelesaikan masalah lebih kompleks.

Kunci penyelesaian masalahnya ada tiga :

1. Menyelesaikan sub-masalah, masalah menggunakan function dan void function yang tepat guna.

2. Menyusun deklarasi variabel yang sesuai dan andal. Yaitu sesedikit mungkin dekalrarsi variabel global, dan mengusahakan sedapat mungkin menggunakan deklarasi variabel lokal.
3. Menggunakan pass parameter (pass argument) yang tepat dengan pedoman bahwa pass parameter pada umumnya adalah pointer (address) dan value parameter.

Namun kemahiran mewujudkan ketiga kunci masalah tersebut tidak semudah seperti yang dibicarakan. Penugasannya memerlukan latihan intensif dan kecermatan yang tinggi.

Berikut ini adalah hasil olahan perkembangan program, sejak pertama kali membahas Linked List struktur hingga pokok dasar Data Base dapat terwujud sederana seperti program FIFO_HS.C

Akan tetapi wujud ini baru merupakan permulaan yang paling awal dan paling sederhana, namun jika dapat dikuasai pendalamannya, akan terasa mendapatkan landasan dasar yang sangat kuat.

Contoh olahan gabungan program Linked List FIFO yang telah dibahas dapat diwujukan menjadi program FIFO_HS.C sebagai berikut :

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<ctype.h>
#define YA 1
#define TIDAK 0
#define NULL 0
struct catatan_murid
{
     char nama[80];
     unsigned long nim;
     struct catatan_murid *next;
};
struct catatan_murid *awal;
```

```
struct catatan_murid *catat(struct catatan_murid *tampung, struct
catatan_murid *ujung)
{
     printf("NIM : "); scanf("%lu", &ujung->nim);
     printf("Nama : "); scanf("%s", &ujung->nama);
     printf("\n");
     ujung->next=NULL;
     if(tampung==NULL)
     {
          tampung=ujung;
     }
     else
     {
          tampung->next=ujung; tampung=ujung;
     }
     return(tampung);
}
disajikan_ya_tidak()
{
     char jawab;
     int j;
     printf("Dilanjutkan? (Y/T)\n");
     if((toupper(jawab=getche()))=='T')j=0;
     else if(toupper(jawab)=='Y')j=1;
     else
     {
          printf("\njawaban harus (Y,y) atau (T,t)\n"); j=1;
     }
     printf("\n");
     return(j);
}
void sajikan_hasil(struct catatan_murid *ujung)
{
```

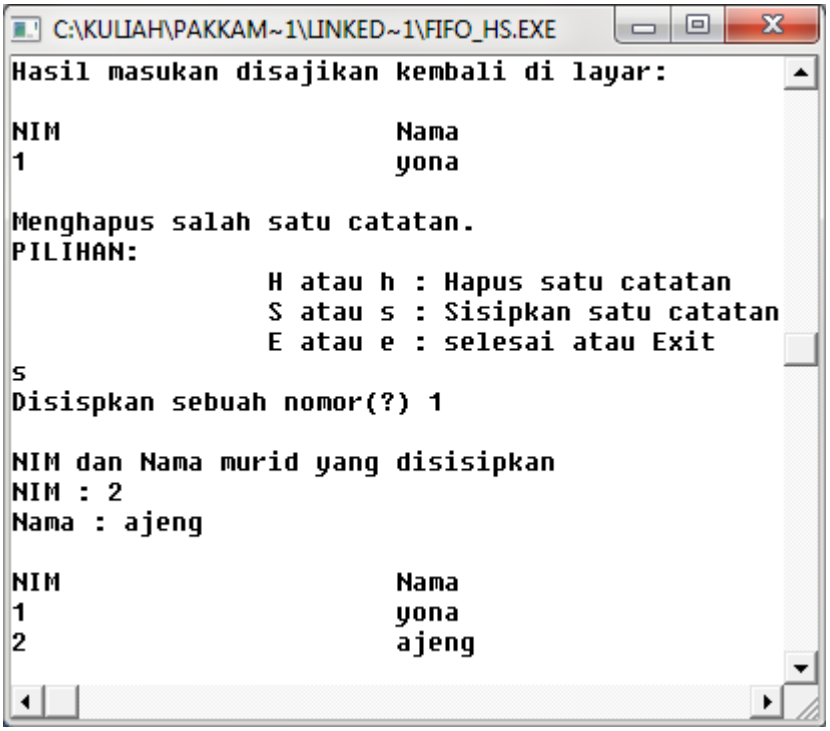
```
     char *p;
     printf("\nNIM           Nama\n");
     ujung=awal;
     while(ujung!=NULL)
     {
          p=&(ujung->nama[0]);
          printf("%-10lu     %s\n", ujung->nim,p);
          ujung=ujung->next;
     }
}
void sisipkan_1_catatan(struct catatan_murid *tampung)
{
     unsigned long h;
     struct catatan_murid *p;
     printf("\n\rDisispkan sebuah nomor(?) "); scanf("%lu", &h);
     p=awal;
     while(p!=NULL)
     {
          if(p->nim==h)
          {
               tampung=(struct                         catatan_murid
*)malloc(sizeof(*tampung));
               printf("\nNIM dan Nama murid yang disisipkan\n");
               printf("NIM : "); scanf("%lu", &tampung->nim);
               printf("Nama : "); scanf("%s", &tampung->nama);
               printf("");
               tampung->next=p->next; p->next=tampung; p=tampung;
          }
          p=p->next;
     }
}
hapus_1_catatan(struct catatan_murid *ujung)
{
```

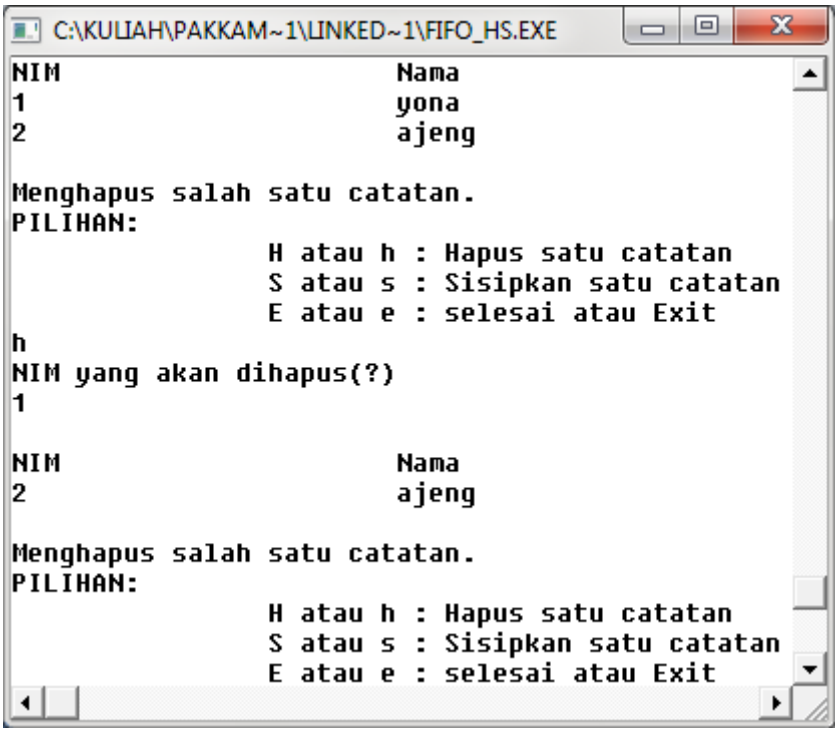
```
     unsigned long h;
     struct catatan_murid *p;
     printf("\nNIM yang akan dihapus(?)\n"); scanf("%lu", &h);
     if(awal->nim==h)
     {
          p=awal->next; free(awal); awal=p;
     }
     else
     {
          p=awal;
          while(p!=NULL)
          {
               ujung=p; p=p->next;
               if(p->nim==h)
               {
                    ujung->next=p->next;
                    free(p);
                    p=ujung->next;
               }
          }
     }
     return 0;
}
menu_sisip_hapus()
{
     printf("\nMenghapus salah satu catatan.\n");
     printf("PILIHAN:\n");
     printf("      H atau h : Hapus satu catatan\n");
     printf("      S atau s : Sisipkan satu catatan\n");
     printf("      E atau e : selesai atau Exit\n");
     return 0;
}
void pilihan(struct catatan_murid *ujung)
```

```
{
     char p;
     while((toupper(p=getche()))!='e')
     {
          switch (p)
          {
               case 's' :
               {
                    sisipkan_1_catatan(ujung);
                    sajikan_hasil(ujung);
                    menu_sisip_hapus(); break;
               }
               case 'h' :
               {
                    hapus_1_catatan(ujung);
                    sajikan_hasil(ujung);
                    menu_sisip_hapus(); break;
               }
               case 'e' :
               {
                    printf("\nPiliha   sisip   hapus   selesai\n");
break;
               }
               default :
               {
                    printf("\njawaban harus (H,h) atau (S,s) atau
(E,e)\n"); break;
               }
          }
     }
}
main()
{
     struct catatan_murid *tampung;
```

```
     struct catatan_murid *ujung;
     int jawab;
     printf("Memasukkan data catatan: \n");
     awal=NULL;
     tampung=NULL;
     jawab=YA;
     while(jawab==YA)
     {
          ujung=(struct    catatan_murid    *)malloc(sizeof(struct
catatan_murid));
          if(awal==NULL)
          {
               awal=catat(tampung,ujung); tampung=awal;
          }
          else tampung=catat(tampung, ujung);
          jawab=disajikan_ya_tidak();
     }
     printf("Hasil masukan disajikan kembali di layar: \n");
     sajikan_hasil(ujung);
     menu_sisip_hapus();
     pilihan(tampung);
}
```

Hasilnya :


C:\KULIAH\PAKKAM~1\LINKED~1\FIFO_HS.EXE
Hasil masukan disajikan kembali di layar:
NIM Nama
1 yona
Menghapus salah satu catatan.
PILIHAN:
H atau h : Hapus satu catatan
S atau s : Sisipkan satu catatan
E atau e : selesai atau Exit
s
Disispkan sebuah nomor(?) 1
NIM dan Nama murid yang disisipkan
NIM : 2
Nama : ajeng
NIM Nama
1 yona
2 ajeng


C:\KULIAH\PAKKAM~1\LINKED~1\FIFO_HS.EXE
NIM Nama
1 yona
2 ajeng
Menghapus salah satu catatan.
PILIHAN:
H atau h : Hapus satu catatan
S atau s : Sisipkan satu catatan
E atau e : selesai atau Exit
h
NIM yang akan dihapus(?)
1
NIM Nama
2 ajeng
Menghapus salah satu catatan.
PILIHAN:
H atau h : Hapus satu catatan
S atau s : Sisipkan satu catatan
E atau e : selesai atau Exit

IV. DAFTAR PUSTAKA

- Handoko, Slamet. 2005. *Pemrograman Bahasa C*. Semarang : Politeknik Negeri Semarang.
- Munir, Rinaldi. 2007. Algoritma & Pemrograman. Bandung : Informatika.
- Kadir, Abdul. 1997. *Pemrograman Dasar Turbo C*. Yogyakarta : ANDI.
- Inggriani Liem,2003,Contoh Program Kecil Dalam Bahasa C, versi Online:ITB