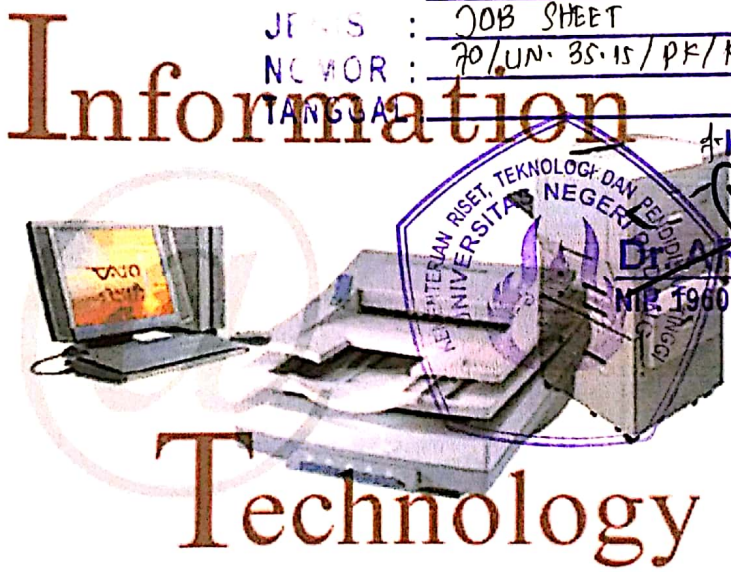
JOB SHEET
Mata Kuliah Praktikum Pemograman komputer



Disusun oleh:

IKA PARMA DEWI,S.Pd,M.Pd.T

Jurusan Teknik Elektronika
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2019

JOB SHEET
Mata Kuliah Praktikum Pemograman komputer

Disusun oleh:

IKA PARMA DEWI,S.Pd,M.Pd.T

Jurusan Teknik Elektronika
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2019

KATA PENGANTAR

Sebagai salah satu persyaratan untuk melengkapi mata kuliah teori dan praktek yang harus dimiliki oleh mahasiswa Pendidikan Teknik Informatika Jurusan Teknik Elektronika Universitas Negeri Padang adalah praktikum pemograman komputer.

Praktikum pemograman komputer adalah pengenalan secara aplikatif tentang ruang lingkup pemrograman, dasar-dasar pemrograman, statement kendali, proses pengulangan/loop(for,while dan do, while), array, fungsi dan struktur. Sehingga dengan sendirinya kelulusan mahasiswa/i terhadap kedua mata kuliah teori dan praktek ini menjadi syarat untuk mengikuti perkuliahan praktikum pemrograman komputer.

Penulis menyadari Job Sheet ini jauh dari kesempurnaan. Penulis sangat menghargai masukan dari pada pembaca. Dengan berbagai kekurangan dan kelemahannya, semoga Job Sheet ini akan membawa manfaat yang sebesar-besarnya untuk proses pembelajaran dan dalam rangka ikut mencerdaskan bangsa Indonesia.

Padang, September 2019

Ika Parma Dewi, S.Pd, M.Pd.T
NIP. 198400413 201404 2 001

DAFTAR ISI

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 1**
Kode : ELA1.61.2306	**Judul : Ruang Lingkup Pemrograman**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

1) Mengenal lingkungan salah satu compiler bahasa pemrograman C.
2) Menggunakan compiler tersebut untuk menyelesaikan kasus sederhana.

B. Alat dan Bahan

1. Personal komputer
2. Aplikasi Turbo C++

C. Teori Singkat

Bahasa C diciptakan oleh Dennis Ritchie tahun 1972 di Bell Laboratories.

Kelebihan Bahasa C:

a) Bahasa C tersedia hampir di semua jenis komputer.
b) Kode bahasa C sifatnya adalah portable dan fleksibel untuk semua jenis komputer.
c) Bahasa C hanya menyediakan sedikit kata-kata kunci. hanya terdapat 32 kata kunci.
d) Proses executable program bahasa C lebih cepat.
e) Dukungan pustaka yang banyak.
f) C adalah bahasa yang terstruktur
g) Bahasa C termasuk bahasa tingkat menengah penempatan ini hanya menegaskan bahwa C bukan bahasa pemrograman yang berorientasi pada mesin, yang merupakan ciri bahasa tingkat rendah, melainkan berorientasi pada obyek tetapi dapat dinterprestasikan oleh mesin dengan cepat, secepat bahasa mesin. Inilah salah satu kelebihan C yaitu memiliki kemudahan dalam menyusun programnya semudah bahasa tingkat tinggi namun dalam mengesekusi program secepat bahasa tingkat rendah.

Kekurangan Bahasa C:

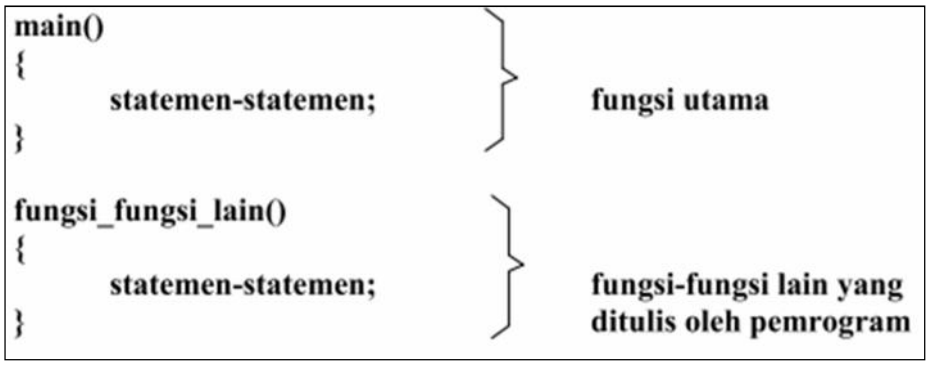
a) Banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
b) Bagi pemula pada umumnya akan kesulitan menggunakan pointer.

Suatu source program C baru dapat dijalankan setelah melalui tahap kompilasi dan penggabungan. Tahap kompilasi dimaksudkan untuk memeriksa source-program sesuai dengan kaidah-kaidah yang berlaku di dalam bahasa pemrograman C. Tahap kompilasi akan menghasilkan relocatable object file. File-file objek tersebut kemudian digabung dengan perpustakaan-fungsi yang sesuai. untuk menghasilkan suatu executable-program.

Shortcut yang digunakan untuk mengkompile:

a) ALT + F9 : dipakai untuk melakukan pengecekan jika ada error pada program yang telah kita buat.
b) CTRL + F9 : dipakai untuk menjalankan program yang telah kita buat.

Struktur dari program C :



D. Percobaan

1. Percobaan 1 : Hello Word

Buatlah sebuah file C baru, simpan dengan nama **Hello.cpp** :

```
/* program C pertamaku */

#include <stdio.h>

main()
{
    printf("Hello Word...!!!");
}
```

E. Latihan

1. Hello World!

Buat file C baru dengan nama : [NamaAnda]. Hasil dari program yang harus tampil dilayar :

```
Welcome to C Programming [NamaAnda]!!!
```

2. The Tree

Buat file C baru dengan nama : TheTree. Hasil dari program yang harus tampil dilayar :

```
I think that I shall never see,
a poem as lovely as a tree.
A tree whose hungry mouth is pressed
Against the Earth’s sweet flowing breast.
```

Selamat ngoding ☺

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 2 dan 3**
Kode : ELA1.61.2306	**Judul : Dasar- dasar Pemrograman**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

1. Mengidentifikasi bagian dasar dari program C.
2. Membedakan mana yang termasuk ke dalam tipe data dasar, tipe variabel, pengidentifikasian dan operator.
3. Mengembangkan program C sederhana.
4. Menganalisa program C.

B. Alat dan Bahan

1. Personal komputer
2. Aplikasi Turbo C++

C. Teori Singkat

Tipe Data Dasar

Data merupakan suatu nilai yang bias dinyatakan dalam bentuk konstanta atau variable. Konstanta menyatakan nilai yang tetap, sedangkan variable menyatakan nilai yang dapat diubah-ubah selama eksekusi berlangsung. Data berdasarkan jenisnya dapat dibagi menjadi lima kelompok, yang dinamakan sebagai tipe data dasar. Kelima tipe data dasar adalah :

- Bilangan bulat (integer)
- Bilangan real presisi-tunggal (float)
- Bilangan real presisi-ganda (double)
- Karakter (char)
- Tak-bertipe (void)

Tipe	Total bit	Kawasan	Keterangan
char	8	-128 s/d 127	karakter
int	32	-2147483648 s/d 2147483647	bilangan integer
float	32	1.7E-38 s/d 3.4E+38	bilangan real presisi-tunggal
double	64	2.2E-308 s/d 1.7E+308	bilangan real presisi-ganda

Tabel1. Ukuran memori untuk tipe data

Format menampilkan argument dari setiap tipe data :

%d	untuk menampilkan bilangan bulat (integer)
%f	untuk menampilkan bilangan titik-mengambang (pecahan)
%c	untuk menampilkan sebuah karakter
%s	untuk menampilkan sebuah string

D. Percobaan

Percobaan 1 : Menampilkan Data Variabel

```
#include <stdio.h>

main()
{
    int value = 10;
    char x;
    x = 'A';

    printf("Value : %d\n", value);
    printf("The value of x = %c", x);
}
```

Percobaan 2 : Penggunaan Operator Aritmatika

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

main()
{
    int i = 37;
    int j = 42;
    float x = 27.475;
    float y = 7.22;

    printf("Variable values...\n");
    printf("    i = %d\n", i);
    printf("    j = %d\n", j);
    printf("    x = %f\n", x);
    printf("    y = %f\n\n", y);


    //penjumlahan angka
    printf("Adding...\n");
    printf("    i + j = %d\n", (i + j));
    printf("    x + y = %f\n\n", (x + y));
```

```
        //pengurangan angka
        printf("Subtracting...\n");
        printf("    i - j = %d\n", (i - j));
        printf("    x - y = %f\n\n", (x - y));

        //perkalian angka
        printf("Multiplying...\n");
        printf("    i * j = %d\n", (i * j));
        printf("    x * y = %f\n\n", (x * y));

        //pembagian angka
        printf("Dividing...\n");
        printf("    i / j = %f\n", (i / j));
        printf("    x / y = %f\n\n", (x / y));

        //tipe penggabungan
        printf("Mixing tipes...\n");
        printf("    j + y = %f\n", (j + y));
        printf("    i * x = %f\n", (i * x));
}
```

Percobaan 3 : Penggunaan Operator Relasi

```
#include <stdio.h>

main()
{
        int i = 37;
        int j = 42;
        int k = 42;

        printf("Nilai variabel...\n");
        printf("    i = %d\n", i);
        printf("    j = %d\n", j);
        printf("    k = %d\n\n", k);

        //lebih besar dari
        printf("Lebih besar dari...\n");
        printf("    i > j = %d\n", (i > j)); //false
        printf("    j > i = %d\n", (j > i)); //true
        printf("    k > j = %d\n\n", (k > j)); //false

        //lebih besar atau sama dengan
        printf("Lebih besar dari atau sama dengan...\n");
        printf("    i >= j = %d\n", (i >= j)); //false
```

```
        printf("    j >= i = %d\n", (j >= i)); //true
        printf("    k >= j = %d\n\n", (k >= j)); //true

         //lebih kecil dari
        printf("Lebih kecil dari...\n");
        printf("    i < j = %d\n", (i < j)); //true
        printf("    j < i = %d\n", (j < i)); //false
        printf("    k < j = %d\n\n", (k < j)); //false

         //lebih kecil atau sama dengan
        printf("Lebih kecil dari atau sama dengan...\n");
        printf("    i <= j = %d\n", (i <= j)); //true
        printf("    j <= i = %d\n", (j <= i)); //false
        printf("    k <= j = %d\n\n", (k <= j)); //true

        //sama dengan
        printf("Sama dengan...\n");
        printf("    i == j = %d\n", (i == j)); //false
        printf("    k == j = %d\n\n", (k == j)); //true

        //tidak sama dengan
        printf("Tidak sama dengan...\n");
        printf("    i != j = %d\n", (i != j)); //true
        printf("    k != j = %d\n", (k != j)); //false
}
```

Percobaan 4 : Penggunaan Operator Kondisi (?:)

```
#include <stdio.h>
main()
{
        int   score = 0;
        char  answer = 'a';

        score = (answer == 'a') ? 10 : 0;
        printf("Score = %d", score );
}
```

E. Latihan

1. **Mendeklarasikan dan mencetak variable**

   Diberikan tabel dibawah ini, deklarasikan variabel yang terdapat didalamnya dengan tipe data yang sesuai dan berikan nilai inisialisasi. Tampilkan hasil outputnya yaitu nama variabel dan nilainya.

Variable name	*Data tipe*	*Initial value*
Number 1	integer	10
letter	character	a
Number 2	float	8.22

Berikut ini merupakan tampilan yang diharapkan sebagai hasil eksekusi program,

```
Number1 = 10
letter = a
Number2 = 8.22
```

2. Mendapatkan nilai rata-rata dari tiga angka

Buatlah program yang menghasilkan output nilai rata-rata dari tiga angka. Nilai dari masing-masing tiga angka tersebut adalah 10, 20 dan 45. Tampilan Output yang diharapkan adalah,

```
numbe  1 = 10
numbe  2 = 20
numbe  3 = 45
Average is = 25
```

Selamat ngoding ☺

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 4 dan 5**
Kode : ELA1.61.2306	**Judul : Statement Kendali (Lanjutan)**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

a) Membuat program C dengan adanya statement kendali

b) Membuat program C dengan penggunaan if, if – else.

c) Membuat program C dengan penggunaan switch – case.

B. ALAT dan BAHAN

1. Komputer/laptop
2. Editor (Dev-C++, TurboC, Codeblocks, Netbeans, eclipse)

C. Teori Singkat

Banyak persoalan yang diperlukan untuk membuat keputusan. Contoh yang sederhana berupa cara mengatur agar komputer bisa menyimpulkan bahwa suatu bilangan merupakan bilangan genap atau bilangan ganjil. Untuk keperluan pengambilan keputusan semacam itu, C menyediakan beberapa jenis pernyataan, berupa

- Pernyataan *if*
- Pernyataan *if-else*, dan
- Pernyataan *switch*

Pernyataan-pernyataan tersebut memerlukan suatu kondisi, sebagai basis dalam pengambilan keputusan. Kondisi umum yang dipakai berupa keadaan benar dan salah.

Operator yang digunakan untuk menghasilkan kondisi benar dan salah, bisa berupa operator relasi dan bisa juga berupa operator logika. Berikut ini dibahas masing-masing jenis operator serta tabel prioritas masing-masing operator.

1. Pernyataan if

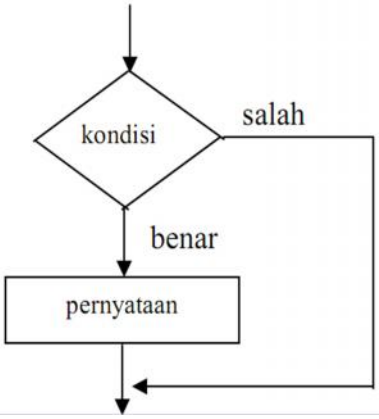
Pernyataan *if* mempunyai bentuk umum :

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

Bentuk ini menyatakan :

- jika kondisi yang diseleksi adalah benar (bernilai logika = 1), maka pernyataan yang mengikutinya akan diproses.
- Sebaliknya, jika kondisi yang diseleksi adalah tidak benar (bernilai logika = 0), maka pernyataan yang mengikutinya tidak akan diproses.

Mengenai kodisi harus ditulis diantara tanda kurung, sedangkan pernyataan dapat berupa sebuah pernyataan tunggal, pernyataan majemuk atau pernyataan kosong. Diagram alir dapat dilihat seperti gambar 3.

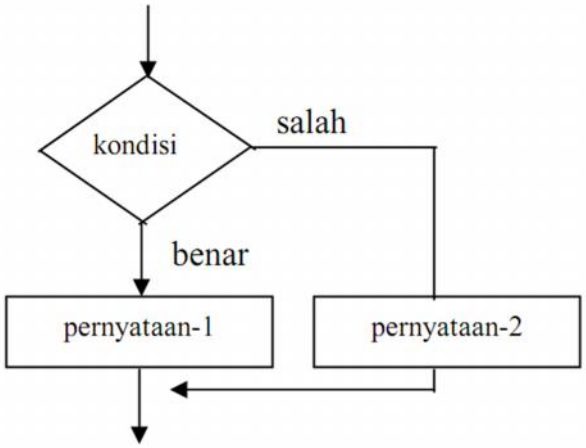


2. Pernyataan if – else

Pernyataan ***if-else*** memiliki bentuk :

```
if (kondisi)
   pernyataan-1;
else
   pernyataan-2;
```

Diagram alir dapat dilihat seperti gambar



Arti dari pernyataan *if-else* :

- Jika kondisi benar, maka **pernyataan-1** dijalankan.
- Sedangkan bila kondisi bernilai salah, maka **pernyataaan-2** yang dijalankan.

Masing-masing **pernyataan-1** dan **pernyataan-2** dapat berupa sebuah pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong.

Contoh penggunaan pernyataan *if-else* adalah untuk menyeleksi nilai suatu bilangan pembagi. Jika nilai bilangan pembagi adalah nol, maka hasil pembagian dengan nilai nol akan mendapatkan hasil tak berhingga. Jika ditemui nilai pembaginya nol, maka proses pembagian tidak akan dilakukan.

3. Pernyataan if dalam if

Di dalam suatu pernyataan *if* (atau *if-else*) bisa saja terdapat pernyataan *if* (atau *if-else)* yang lain. Bentuk seperti ini dinamakan sebagai *nested if*. Secara umum, bentuk dari pernyataan ini adalah sebagai berikut :

```
if (kondisi-1)
    if (kondisi-2)
    .
    .
        if(kondisi-n)
            pernyataan;
        else
            pernyataan;
            .
            .
    else
```

pernyataan;

- Kondisi yang akan diseleksi pertama kali adalah kondisi yang terluar (**kondisi-1**). Jika **kondisi-1** bernilai salah, maka statemen *else* yang terluar (pasangan *if* yang bersangkutan) yang akan diproses. Jika *else* (pasangannya tsb) tidak ditulis, maka penyeleksian kondisi akan dihentikan.
- Jika **kondisi-1** bernilai benar, maka kondisi berikutnya yang lebih dalam (**kondisi-2**) akan diseleksi. Jika **kondisi-2** bernilai salah, maka statemen *else* pasangan dari *if* yang bersangkutan yang akan diproses. Jika *else* (untuk **kondisi-2**) tidak ditulis, maka penyeleksian kondisi akan dihentikan.
- Dengan cara yang sama, penyeleksian kondisi akan dilakukan sampai dengan **kondisi-n,** jika kondisi-kondisi sebelumnya bernilai benar.

4. Pernyataan switch

Pernyataan switch merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah alternative. Diantaranya untuk menggantikan pernyataan if bertingkat.

Bentuk umum pernyataan switch :

```
switch (ekspresi)
{
case konstanta-1 :
    pernyataan-1
    break;
case konstanta-2 :
    pernyataan-2
    break;
.
.
.
case konstanta-n :
    pernyataan-n
    break;
default :
    pernyataan-n
}
```

Dengan *ekspresi* dapat berupa ungkapan bernilai integer atau bertipe karakter. Setiap *konstanta-i* (*konstanta-1, konstanta-2,…*) dapat berupa konstanta integer atau karakter.

Setiap *pernyataan-i* (*pernyataan-1,…., pernyataan-n*) dapat berupa sebuah atau beberapa pernyataan.

D. Percobaan

Percobaan 1 : Menghitung Diskon

```
#include <stdio.h>

main()
{
	double total_pembelian, diskon;

	printf("Total pembelian : Rp ");
	scanf("%lf", &total_pembelian);

	diskon = 0;
	if(total_pembelian >= 50000)
		diskon = 0.05 * total_pembelian;

	printf("Besarnya diskon : Rp %.2lf\n", diskon);
}
```

Percobaan 2 : Menentukan Bilangan Ganjil Atau Genap

```
#include <stdio.h>

main()
{
	int bilangan;

	printf("Masukkan sebuah bilangan bulat : ");
	scanf("%d", &bilangan);

	if (bilangan % 2)
	{
		printf("Nilai %d tidak habis dibagi 2\n", bilangan);
		puts("Karena itu termasuk sebagai bilangan GANJIL");
	}
	else
	{
		printf("Nilai %d habis dibagi 2\n", bilangan);
		puts("Karena itu termasuk sebagai bilangan GENAP");
	}
}
```

Percobaan 3 : Menentukan Bilangan Positif Atau Negatif

```
#include <stdio.h>

main()
{
	float x, y;

	printf("Masukkan nilai x dan y : ");
	scanf("%f %f", &x, &y);

	if (x > 0)
		if (y > 0)
			puts("x dan y positif");
		else
			puts("Diantara x dan y ada yang bernilai negatif");
}
```

Percobaan 4 : Menentukan Nama Hari 1

```
#include <stdio.h>

main()
{
	int kode_hari;
	puts("Menentukan hari\n");
	puts("1 = SENIN   3 = RABU   5 = JUMAT   7 = MINGGU");
	puts("2 = SELASA  4 = KAMIS  6 = SABTU");
	printf("\nMasukkan 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 JUMAT");
	else if (kode_hari == 6)
		puts("Hari SABTU");
	else if (kode_hari == 7)
		puts("Hari MINGGU");
	else
		puts("Kode yang anda masukkan salah!"); }
```

Percobaan 5 : Menentukan Nilai 1

```
#include <stdio.h>

main()
{
    double grade = 92.0;
    if( grade >= 90 ){
        printf( "Excellent!" );
    }
    else if( (grade < 90) && (grade >= 80)){
        printf("Good job!" );
    }
    else if( (grade < 80) && (grade >= 60)){
        printf("Study harder!" );
    }
    else{
        printf("Sorry, you failed.");
    }
}
```

Percobaan 6 : Menentukan Jenis Inputan

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

main()
{
    char karakter;
    printf("Masukkan sebuah karakter : ");
    karakter = getche();

    printf("\nSaya tahu. Yang anda ketikkan ");
    if ((karakter >= 'A') && (karakter <= 'Z'))
        puts("adalah HURUF KAPITAL");

    else if ((karakter >= 'a') && (karakter <= 'z'))
        puts("adalah HURUF KECIL");
    else if ((karakter >= '0') && (karakter <= '9'))
        puts("adalah ANGKA");
    else if (karakter == ' ')
        puts("adalah SPASI");
    else
    {
        puts("tidak termasuk HURUF KAPITAL, ");
        puts("HURUF KECIL, ANGKA maupun SPASI");
    }
```

```
}
```

Percobaan 7 : Menentukan Nama Hari 2

```
#include <stdio.h>

main()
{
    int kode_hari;

    puts("Menentukan hari\n");
    puts("1 = SENIN   3 = RABU   5 = JUMAT   7 = MINGGU");
    puts("2 = SELASA  4 = KAMIS  6 = SABTU");
    printf("\nMasukkan kode hari (1 - 7) : ");
    scanf("%d", &kode_hari);

    switch (kode_hari)
    {
        case 1 :
            puts("Hari SENIN");
            break;
        case 2 :
            puts("Hari SELASA");
            break;

        case 3 :
            puts("Hari RABU");
            break;
        case 4 :
            puts("Hari KAMIS");
            break;
        case 5 :
            puts("Hari JUMAT");
            break;
        case 6 :
            puts("Hari SABTU");
            break;
        case 7 :
            puts("Hari MINGGU");
            break;
        default :
        puts("Kode yang anda masukkan salah!");
    }
}
```

Percobaan 8 : Menentukan Nilai 2

```
#include <stdio.h>

main()
{
    int grade = 92;
    switch(grade) {
    case 100:
        printf( "Excellent!" );
        break;

    case 90:
        printf("Good job!" );
        break;

    case 80:
        printf("Study harder!" );
        break;

    default:
        printf("Sorry, you failed.");
    }
}
```

E. Latihan

1. Membaca Bilangan

Ambil sebuah angka sebagai input dari user, dan outputnya berupa kata yang sesuai dengan angka. Angka yang dimasukkan antara 1-10. Jika user memasukkan nilai yang tidak sesuai berikan output “Invalid number”.

1. Gunakan statement if-else untuk menyelesaikan
2. Gunakan statement switch untuk menyelesaikan

2. Menghitung Rata-rata

Buatlah program untuk menghitung nilai rata-rata seorang siswa, dengan ketentuan sebagai berikut:

- Nama Siswa, Nilai Pertandingan I, Nilai Pertandingan II, Nilai Pertandingan III diinput.
- Nilai rata-rata merupakan hasil dari Nilai Pertandingan I, II, dan III dibagi 3.
- Ketentuan mendapatkan hadiah dari pertandingan:
  - Jika Nilai Rata-Rata >= 85, maka mendapat hadiah seperangkat komputer

- Jika Nilai Rata-Rata >= 70, maka mendapat hadiah uang sebesar Rp. 500.000
- Jika Nilai Rata-Rata < 70, maka mendapat hadiah hiburan

) Tampilan yang diinginkan sebagai berikut:

Layar Masukkan

```
PROGRAM HITUNG NILAI RATA-RATA

Nama Siswa : ... <diinput>
Nilai Pertandingan I   : ... <diinput>
Nilai Pertandingan II  : ... <diinput>
Nilai Pertandingan III : ... <diinput>
```

Layar Keluaran

```
Siswa yang bernama ... <tampil data>
Memperoleh nilai rata-rata <hasil proses> dari hasil
perlombaan yang diikutinya.
Hadiah yang didapat adalah ... <hasil proses>
```

Selamat ngoding ☺

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 6 dan 7**
Kode : ELA1.61.2306	**Judul : Proses Pengulangan/Loop(for, while)**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

- Membuat program dengan proses pengulangan dengan menggunakan pernyataan for
- Membuat program dengan proses pengulangan dengan menggunakan pernyataan while

B. Alat dan Bahan

1. Personal Komputer
2. Aplikasi Turbo C++

C. Teori Singkat

1. Pernyataan for

Mengulang suatu proses merupakan tindakan yang banyak dijumpai dalam pemrograman. Pada semua bahasa pemrograman, pengulangan proses ditangani dengan suatu mekanisme yang disebut *loop*. Dengan menggunakan *loop*, suatu proses yang berulang misalnya menampilkan tulisan yang sama seratus kali pada layar dapat diimpelementasikan dengan kode program yang pendek.

Pernyataan pertama yang digunakan untuk keperluan pengulangan proses adalah pernyataan *for*. Bentuk pernyataan ini :

```
for (ungkapan1; ungkapan2; ungkapan3)
  pernyataan;
```

Kegunaan dari masing-masing ungkapan pada pernyataan *for*.

- Ungkapan1 : digunakan untuk memberikan inisialisasi terhadap variabel pengendali *loop*.
- Ungkapan2 : dipakai sebagai kondisi untuk keluar dari *loop*.
- Ungkapan3 : dipakai sebagai pengatur kenaikan nilai variabel pengendali *loop*.

Ketiga ungkapan dalam *for* tersebut harus dipisahkan dengan tanda titik koma (;). Dalam hal ini pernyatan bisa berupa pernyataan tunggal maupun jamak. Jika pernyataannya

berbentuk jamak, maka pernyataan-pernyataan tersebut harus diletakkan di antara kurung kurawal buka ({) dan kurung kurawal tutup (}), sehingga formatnya menjadi :

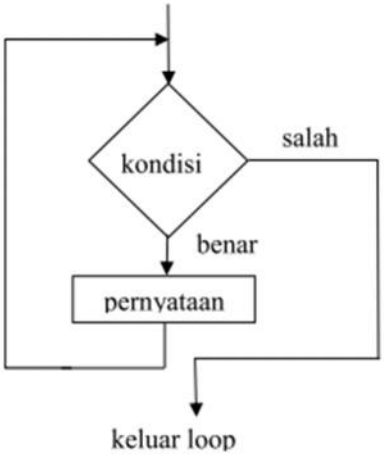
```
for (ungkapan1; ungkapan2; ungkapan3)
{
   pernyataan;
   pernyataan;
   .
   .
   .
}
```

2. Pernyataan while

Pada pernyataan *while*, pengecekan terhadap loop dilakukan di bagian awal (sebelum tubuh loop). Lebih jelasnya, bentuk pernyataan *while* adalah sebagai berikut :

```
while (kondisi)
   pernyataan;
```

dengan pernyataan dapat berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Proses pengulangan terhadap pernyataan dijelaskan pada gambar berikut :



Gambar 1 Diagram alir *while*

Dengan melihat gambar 1 tampak bahwa ada kemungkinan pernyataan yang merupakan tubuh loop tidak dijalankan sama sekali, yaitu kalau hasil pengujian kondisi *while* yang pertama kali ternyata bernilai salah.

D. Percobaan

Percobaan 1 : Perulangan dengan menggunakan for

```
#include <stdio.h>

main()
{
	int bilangan;

	for (bilangan = 20; bilangan <= 100; bilangan += 10)
		printf("%d\n", bilangan);
}
```

Percobaan 2 : Perulangan dengan menggunakan for

```
#include <stdio.h>

main()
{
	int bilangan;

	for (bilangan = 60; bilangan >= 10; bilangan -= 10)
		printf("%d\n", bilangan);
}
```

Percobaan 3 : Perulangan dengan menggunakan for

```
#include <stdio.h>

main()
{
	int bilangan = 60;

	for ( ; bilangan >= 10; )
	{
		printf("%d\n", bilangan);
		bilangan -= 10;
	}
}
```

Percobaan 4 : Perulangan dengan menggunakan for

```
#include <stdio.h>

main()
{
	int bil ;
```

```
        for(bil = 0; bil <= 10; bil++)
                printf("%d\n", bil);
}
```

Percobaan 5 : Perulangan dengan menggunakan while

```
#include <stdio.h>
main()
{
        int i = 0;
        while (i <= 10)
        {
                printf("%d\n", i);
                i++;
        } }
```

Percobaan 6 : Perulangan dengan menggunakan while

```
//membaca Y atau T dengan while

#include <stdio.h>

main()
{
        char pilihan;
        //diberi nilai salah lebih dahulu
        int sudah_benar = 0;

        printf("Pilihan Y atau T.\n");

        //program dilanjutkan jika tombol Y,y,T,t ditekan
        while (!sudah_benar)
        {
                pilihan = getchar(); //baca tombol
                sudah_benar = (pilihan == 'Y') || (pilihan == 'y') ||
                              (pilihan == 'T') || (pilihan == 't');
        }
        //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");
        }
}
```

Percobaan 7 : Perulangan dengan menggunakan while

```
//menghitung jmlh karakter dlm suatu kalimat

#include <stdio.h>

main()
{
	char kar;
	int jumkar = 0, jumspasi = 0;

	puts("Masukkan sebuah kalimat dan akhiri dengan ENTER.\n");
	puts("Saya akan menghitung jumlah karakter ");
	puts("pada kalimat tersebut.\n");

	while((kar = getchar()) != '\n')
	{
		jumkar++;
		if (kar == ' ') jumspasi++;
	}

	printf("\nJumlah karakter  = %d", jumkar);
	printf("\nJumlah spasi     = %d\n\n", jumspasi);
}
```

Percobaan 8 : Perulangan dengan menggunakan while

```
#include <stdio.h>
main()
{
	int pencacah = 0;
		while (pencacah < 10)
	{
		puts("TURBO C++");
		pencacah++;
	}
}
```

E. Latihan

1. **Cetak nama 100 kali**

   Buatlah program untuk mencetak nama masing-masing sebanyak 100 kali dengan menggunakan pernyataan perulangan for dan while.

2. **Cetak angka**

   Gunakan perulangan for dan while untuk menghasilkan tampilan dari angka 100 sampai angka 1.

Selamat ngoding ☺

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 8**
Kode : ELA1.61.2306	**Judul : Proses Pengulangan/Loop(do, while)**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

1. Membuat program dengan proses pengulangan dengan menggunakan pernyataan do-while
2. Menggunaan pernyataan break
3. Menggunaan pernyataan continue
4. Menggunaan pernyataan goto
5. Membuat program dengan proses loop di dalam loop (nested loop)
6. Penggunaan exit( ) untuk menghentikan ekseskusi program

B. Alat dan Bahan

1. Personal computer
2. Aplikasi Turbo C++

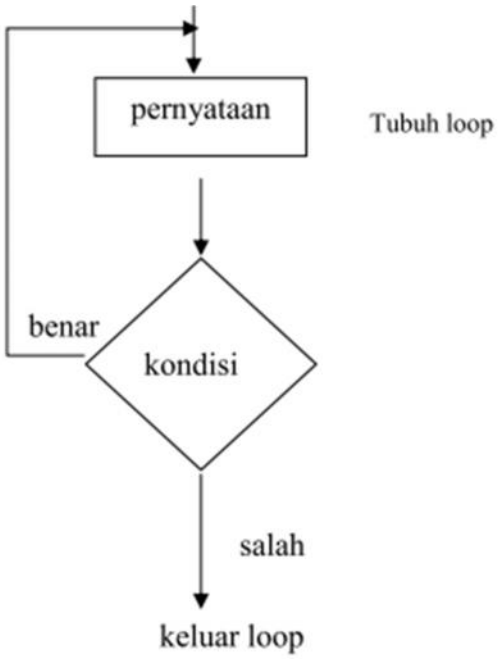
C. Teori Singkat

1. Pernyataan do-while

Bentuk pernyataan ***do-while***

```
do
   pernyataan;
while (kondisi)
```

Pada pernyataan *do-while*, tubuh *loop* berupa pernyataan,dengan pernyataan bisa berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong. Pada pernyataan *do*, mula-mula pernyataan dijalankan. Selanjutnya, kondisi diuji. Sendainya kondisi bernilai benar, maka pernyataan dijalankan lagi, kemudian kondisi diperiksa kembali, dan seterusnya. Kalau kondisi bernilai salah pada saat dites, maka pernyataan tidak dijalankan lagi. Untuk lebih jelasnya dapat dilihat pada Gambar 1 . Berdasarkan Gambar 1 terlihat bahwa tubuh *loop* minimal akan dijalankan sekali.



Gambar 1 . Diagram alir *do-while*

2. Pernyataan break

Pernyataan break sesungguhnya telah diperkenalkan pada pernyataan *switch*. Pernyataan ini berfungsi untuk keluar dari *loop for*, *do-while* dan *while*. Sedangkan pada *switch* yaitu untuk menuju ke akhir (keluar dari) struktur *switch*. Sebagai contoh dapat dilihat pada gambar 2 . Kalau pernyataan *break* dijalankan maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir tubuh *loop for*.

```
for ( ; ; )
{
      .
      .
  if ( …… )
    break;
      .
      .
}                    /* akhir tubuh loop for */

puts("\nSelesai…");
```

2

Gambar 4.4 Ilustrasi pengaruh break

3. Pernyataan continue

Pernyataan *continue* digunakan untuk mengarahkan eksekusi ke iterasi (proses) berikutnya pada *loop* yang sama. Pada *do-while* dan *while*, pernyataan *continue* menyebabkan eksekusi menuju ke kondisi pengujian pengulangan, seperti yang dilukiskan pada Gambar 3. Pada *loop for*, pernyataan contunue menyebabkan bagian penaik variabel pengendali *loop* dikerjakan (ungkapan3 pada struktur *for*) dan kondisi untuk keluar dari *loop for* (ungkapan2 pada struktur *for*) diuji kembali.

Program ini digunakan untuk memasukkan data harus diulangi dan hal ini dikendalikan dengan *continue*. Untuk mengakhiri pemasukan data, data yang dimasukkan harus bernilai kurang dari 0. Perlu diketahui kondisi bernilai 1.

```
while (kondisi) do
{
    continue;
}
```

```
{
    continue;
} while (kondisi)
```

Gambar 3 Pengaruh *continue* pada *while* dan *do-while*

Menyatakan bahwa kondisi selalu dianggap benar. Untuk keluar dari *loop*, pernyataan yang digunakan berupa *break*.

4. Pernyataan goto

Pernyataan *goto* merupakan intruksi untuk mengarahkan eksekusi ke pernyataan yang diawali dengan suatu label. Label sendiri berupa suatu pengenal (*identifier*) yang diikuti dengan tanda titik dua (:)

Contoh pemakaian *goto* ditujukan pada program dibawah ini:

Pernyataan

```
goto cetak;
```

Mengisyaratkan agar eksekusi dilanjutkan ke pernyataan yang diawali dengan label

```
cetak:
```

Pernyataan

```
if (++pencacah <= 10)
     goto cetak;
```

Mempunyai arti :

- Naikkan nilai **pencacah** sebesar 1
- Kemudian, jika **pencacah** kurang dari atau sama dengan 10 maka eksekusi menuju ke label **cetak.**

Penerapan *goto* biasanya dilakukan pada *loop* di dalam *loop* (*nested loop*), dengan tujuan memudahkan untuk keluar dari *loop* terdalam menuju ke pernyataan yang terletak di luar *loop* terluar.

5. Menggunakan exit( ) untuk menghentikan eksekusi program

Suatu eksekusi program dapat dihentikan (secara normal) melalui pemanggilan fungsi *exit( )*. Hal ini biasa dilakukan, jika di dalam suatu eksekusi terdapat suatu kondisi yang tak dikehendaki. Prototipe dari fungsi *exit()* didefinisikan pada file **stdlib.h**, yang memiliki deklarasi sebagai berikut :

```
void exit(int status);
```

Menurut kebiasaan, nilai nol diberikan pada argumen *exit()* untuk menunjukkan penghentian program yang normal. Sedangkan untuk menunjukkan kesalahan, nilai yang diberikan pada argumen fungsi diisi dengan nilai bukan-nol.

D. Percobaan

Percobaan 1 : Perulangan dengan menggunakan do-while

```
#include <stdio.h>

main()
{
     int i = 0;

     do
```

```
        {
                printf("%d\n", i);
                i++;
        }
        while (i <= 10);
}
```

Percobaan 2 : Perulangan dengan menggunakan do-while

```
//menampilkan tulisan sebyk 10 x dengan do-while

#include <stdio.h>

main()
{
        int pencacah = 0;

        do
        {
                puts("TURBO C++");
                pencacah++;
        }
        while (pencacah < 10);
}
```

Percobaan 3 : Perulangan dengan menggunakan do-while

```
//membaca Y atau T dengan do-while

#include <stdio.h>

main()
{
        char pilihan;
        //diberi nilai salah lebih dahulu
        int sudah_benar = 0;

        printf("Pilihan Y atau T.\n");

        //program dilanjutkan jika tombol Y,y,T,t ditekan
        do
        {
                pilihan = getchar(); //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");
        }
}
```

Percobaan 4 : Menggunakan pernyataan continue

```
//menampilkan bil. ganjil antara 7 -25 kecuali 15

#include <stdio.h>

main()
{
        int x;

        for (x = 7; x <= 25; x += 2)
        {
                if (x == 15)
                        continue;
                printf("%4d", x);
        }
        printf("\n");
}
```

Percobaan 5 : Menggunakan pernyataan break

```
//pemakaian break untuk keluar dari looping

#include <stdio.h>

main()
{
        char kar;

        printf("Ketik sembarang kalimat");
        printf(" dan akhiri dengan ENTER.\n");

        for ( ; ;)
        {
```

```
            kar = getchar();
            if (kar == '\n')
                break;
        }
        printf("Selesai\n");
    }
```

Percobaan 6 : Nested loop (perulangan di dalam perulangan)

```
    //loop for bersarang untuk membuat tabel perkalian

    #include <stdio.h>
    #define MAKS 8

    main()
    {
        int baris, kolom, hasil_kali;

        for (baris = 1; baris <= MAKS; baris++)
        {
            for (kolom = 1; kolom <= MAKS; kolom++)
            {
                hasil_kali = baris * kolom;
                printf("%3d", hasil_kali);
            }
            printf("\n"); //pindah baris
        }
    }
```

Percobaan 7 : Pemakaian exit( ) untuk menghentikan eksekusi program

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

    main()
    {
        char kar;

        printf("Tekanlah X untuk menghentikan program.\n");

            while(((kar = getchar()) == 'x'))
                exit(0);
    }
```

E. Latihan

1. Mencetak nama

Buatlah program untuk mencetak nama masing-masing sebanyak 100 kali dengan menggunakan pernyataan do-while.

2. Mencetak angka

Dengan menggunakan perulangan for, while, do-while buatlah program yang hasil tampilannya seperti di bawah ini :

```
1
22
333
4444
55555
```

Selamat ngoding ☺

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 10 dan 11**
Kode : ELA1.61.2306	**Judul : Array**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

a) Membuat program dengan adanya unsur array berdimensi satu.

b) Membuat program dengan adanya unsur array berdimensi dua.

B. Alat dan Bahan

1. Personal computer
2. Aplikasi Turbo C++

C. Teori Singkat

Dalam beberapa literatur, array sering disebut (diterjemahkan) sebagai larik. Array adalah kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data di suatu array disebut dengan elemen-elemen array. Letak urutan dari elemen-elemen array ditunjukkan oleh suatu *subscript* atau indeks.

Array bisa berupa array berdimensi satu, dua, tiga atau lebih. Array berdimensi satu *(one-dimensional array)* mewakili bentuk suatu vektor. Array berdimensi dua *(two-dimensional array)* mewakili bentuk dari suatu matriks atau table

. Array berdimensi tiga *(three-dimensional array)* mewakili bentuk suatu ruang.

1. Array Berdimensi Satu

Suatu array berdimensi satu dideklarasikan dalam bentuk umum berupa :

```
tipe_data nama_var[ukuran];
```

dengan :

- tipe_data : untuk menyatakan tipe dari elemen array, misalnya *int, char, float*.
- nama_var : nama variabel array
- ukuran : untuk menyatakan jumlah maksimal elemen array.

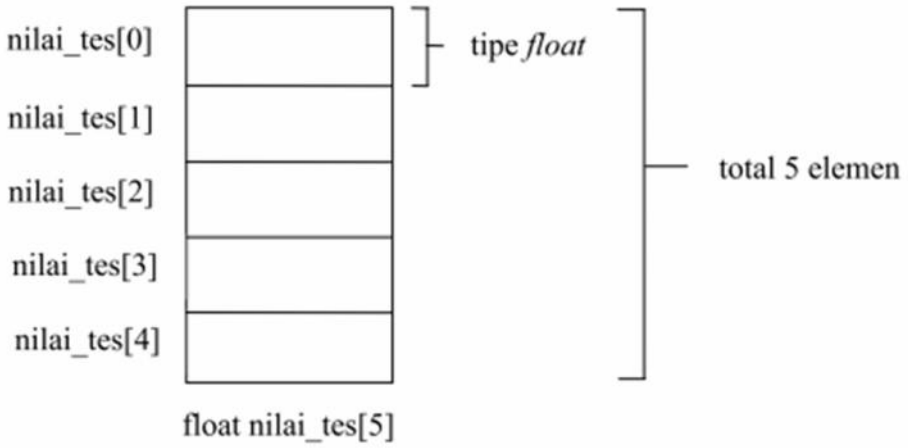
Contoh pendeklarasian array :

```
float nilai_tes[5];
```

menyatakan bahwa array **nilai_tes** mengandung 5 elemen bertipe *float*.

Mengakses Array Berdimensi Satu

Pada C, data array akan disimpan dalam memori yang berurutan. Elemen pertama mempunyai indeks bernilai 0. Jika variabel **nilai_tes** dideklarasikan sebagai array dengan 5 elemen, maka elemen pertama memiliki indeks sama dengan 0, dan elemen terakhir memiliki indeks 4. Gambar 6.1 di bawah ini menjelaskan urutan komponen dalam array.



Bentuk umum pengaksesan array adalah sbb :

nama_var[indeks]

sehingga, untuk array **nilai_tes**, maka :

nilai_tes[0] → elemen pertama dari **nilai_tes**

nilai_tes[4] → elemen ke-5 dari **nilai_tes**

Contoh :

```
nilai_tes[0] = 70;                    /* contoh 1 */
scanf("%f", &nilai_tes[2]);           /* contoh 2 */
```

Contoh pertama merupakan pemberian nilai 70 ke **nilai_tes[0]**. Sedangkan contoh 2 merupakan perintah untuk membaca data bilangan dari keyboard dan diberikan ke **nilai_tes[2]**. Pada contoh 2 ini

```
&nilai_tes[2]
```

berarti "alamat dari **nilai_tes[2]**". Perlu diingat bahwa *scanf()* memerlukan argumen berupa alamat dari variabel yang digunakan untuk menyimpan nilai masukan.

Inisialisasi Array Berdimensi Satu

Sebuah array dapat diinisialisasi sekaligus pada saat dideklarasikan. Untuk mendeklarasikan array, nilai-nilai yang diinisialisasikan dituliskan di antara kurung kurawal ({}) yang dipisahkan dengan koma.

Contoh :

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

merupakan instruksi untuk mendeklarasikan array **jum_hari** yang memiliki 12 elemen yang bersifat statis dan sekaligus melakukan inisialisasi terhadap masing-masing elemen array.

Beberapa Variasi Mendeklarasikan Array

Ada beberapa variasi cara mendeklarasikan sebuah array (dalam hal ini yang berdimensi satu), di antaranya adalah sebagai berikut :

- `int  numbers[10];`
- `int numbers[10] = { 34, 27, 16 };`
- `int numbers[] = { 2, -3, 45, 79, -14, 5, 9, 28, -1, 0 };`
- `char text[] = "Welcome to New Zealand.";`
- `float radix[12] = { 134.362, 1913.248 };`
- `double radians[1000];`

2. Array Berdimensi Dua

Data seperti yang disajikan pada Tabel 6-1, dapat disimpan pada sebuah array berdimensi dua. Dimensi pertama dari array digunakan untuk menyatakan kode program kursus dan dimensi kedua untuk menyatakan tahun kursus.

Tabel 6-1. Data Kelulusan Siswa Pada Sebuah Kursus Komputer

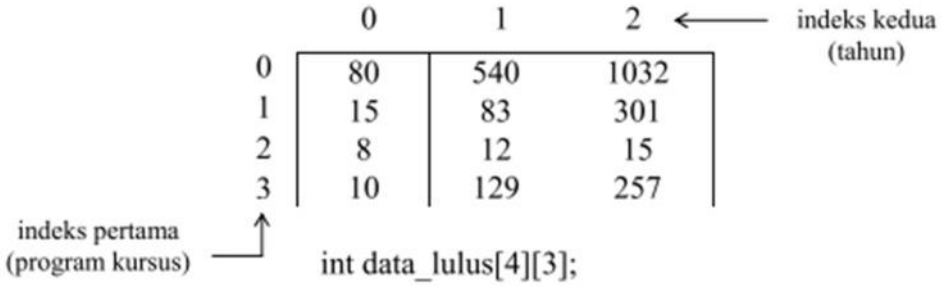
Tahun Program	1998	1999	2000
1. (INTRO)	80	540	1032
2. (BASIC)	15	83	301
3. (PASCAL)	8	12	15
4. (C)	10	129	257

Mendeklarasikan Array Berdimensi Dua

Pendeklarasian yang diperlukan untuk menyimpan data kelulusan siswa pada Tabel 6-1 adalah:

```
int data_lulus[4][3];
```

Nilai 3 untuk menyatakan banyaknya tahun dan 4 menyatakan banyaknya program kursus. Gambar 6.2 memberikan ilustrasi untuk memudahkan pemahaman tentang array berdimensi dua.



Gambar 6.2 Array berdimensi dua

Mengakses Array Berdimensi Dua

Array seperti **data_lulus** dapat diakses dalam bentuk

data_lulus[indeks pertama, indeks kedua]

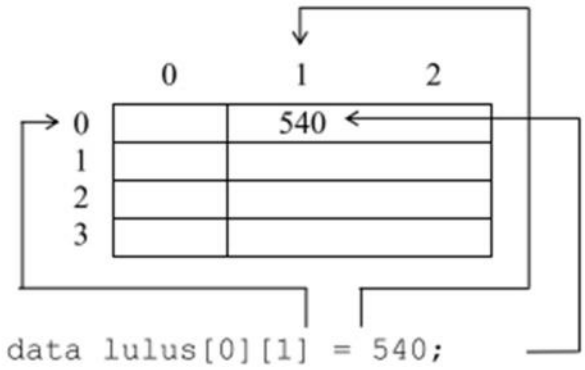
Contoh :

(1) `data_lulus[0][1] = 540;`

merupakan instruksi untuk memberikan nilai 540 ke array **data_lulus** untuk indeks pertama = 0 dan indeks kedua bernilai 1.

(2) `printf("%d",data_lulus[2][0]);`

merupakan perintah untuk menampilkan elemen yang memiliki indeks pertama = 2 dan indeks kedua = 0.



Inisialisasi Array Berdimensi Dua

Gambar berikut memberikan penjelasan tentang inisialisasi yang dilakukan terhadap array berdimensi dua :

0	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0
1	1	1	1	1	1	1	0
1	1	0	0	0	0	1	0
1	1	0	0	0	0	1	0
1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0

Untuk itu, deklarasi dan inisialisasi yang dilakukan berupa :

```
int huruf_A[8][8] = {
        { 0, 1, 1, 1, 1, 1, 0, 0 } ,
        { 0, 1, 0, 0, 0, 1, 0, 0 } ,
        { 0, 1, 0, 0, 0, 1, 0, 0 } ,
        { 1, 1, 1, 1, 1, 1, 1, 0 } ,
        { 1, 1, 0, 0, 0, 0, 1, 0 } ,
        { 1, 1, 0, 0, 0, 0, 1, 0 } ,
        { 1, 1, 0, 0, 0, 0, 1, 0 } ,
        { 0, 0, 0, 0, 0, 0, 0, 0 }
    };
```

atau bisa juga ditulis sebagai berikut :

```
int huruf_A[8][8] =
    {  0, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0
    };
```

D. Percobaan

Percobaan 1

```
#include <stdio.h>

main()
{
    int i;
    int values[] = {1,2,3,4,5,6,7,8,9};
    char word[] = {'H', 'E', 'L', 'L', 'O'};

    for(i = 0; i < 9; i++)
        printf("Values[%d] is %d\n", i, values[i]);

    printf("\n");
    for(i = 0; i < 5; i++)
        printf("Word[%d] is %c\n", i, word[i]);
}
```

Percobaan 2

```
#include <stdio.h>
#define MAKS 5
```

```
main()
{
    int i;
    float total = 0, rata;
    float nilai_tes[MAKS];  //deklarasi array

    for(i = 0; i < MAKS; i++)  //pemasukan data nilai_tes
    {
        printf("Nilai tes ke-%d : ", i+1);
        scanf("%f", &nilai_tes[i]);
        //menghitung jumlah seluruh nilai
        total = total + nilai_tes[i];
    }
    rata = total / MAKS;  //hitung nilai rata-rata

    //cetak nilai rata-rata
    printf("\nNilai rata-rata = %.2f\n", rata);
}
```

Percobaan 3

```
#include <stdio.h>

main()
{
    int bulan, tahun, jhari;
    int jum_hari[12] = {31, 28, 31, 30, 31, 30,
                        31, 31, 30, 31, 30, 31};

    puts("MEMPEROLEH JUMLAH HARI");
    puts("PADA SUATU BULAN DAN SUATU TAHUN");
    puts("--------------------------------");
    printf("Masukkan bulan (1..12) : ");
    scanf("%d", &bulan);
    printf("Masukkan tahunnya : ");
    scanf("%d", &tahun);

    if(bulan == 2)
        if(tahun % 4 == 0)
            jhari = 29;
        else
            jhari = 28;
    else
        jhari = jum_hari[bulan-1];

    printf("\nJumlah hari dalam bulan %d tahun %d adalah %d hari\n",
            bulan, tahun, jhari); }
```

Percobaan 4

```
#include <stdio.h>

main()
{
    int tahun, kode_program;
    int data_lulus[4][3];

    //memberikan data ke array
    data_lulus[0][0] = 22;
    data_lulus[0][1] = 8;
    data_lulus[0][2] = 89;

    data_lulus[1][0] = 21;
    data_lulus[1][1] = 7;
    data_lulus[1][2] = 87;

    data_lulus[2][0] = 24;
    data_lulus[2][1] = 9;
    data_lulus[2][2] = 28;

    data_lulus[3][0] = 22;
    data_lulus[3][1] = 21;
    data_lulus[3][2] = 24;

    //proses utk memperoleh informasi jml siswa yg lulus
    printf("Masukkan tahun dari data yang ingin anda ketahui");
    printf("(2012..2014) : ");
    scanf("%d", &tahun);

    printf("(1 = INTRO, 2 = BASIC, 3 = PASCAL, 4 = C) : ");
    scanf("%d", &kode_program);
    printf("\nTotal kelulusan program tersebut = %d\n",
                    data_lulus[kode_program-1][tahun - 2012]);
}
```

E. Latihan

Buatlah program yang mempunyai variabel array yang menyimpan data dari nama masing-masing dan program tersebut menampilkan setiap huruf dari nama masing-masing (seperti percobaan 1).

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 12 dan 13**
Kode : ELA1.61.2306	**Judul : Fungsi**

A. LEARNING OUTCOMES PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu

1. Memecah program menjadi fungsi-fungsi yang sederhana

B. Alat dan Bahan

1. Personal computer
2. Aplikasi Turbo C++

C. Teori Singkat

Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya. Elemen utama dari program bahasa C berupa fungsi-fungsi, dalam hal ini program dari bahasa C dibentuk dari kumpulan fungsi pustaka (standar) dan fungsi yang dibuat sendiri oleh pemrogram. Fungsi banyak digunakan pada program C dengan tujuan :

a. Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Dengan memisahkan langkah-langkah detail ke satu atau lebih fungsi-fungsi, maka fungsi utama (*main()*) menjadi lebih pendek, jelas dan mudah dimengerti.

b. dapat mengurangi pengulangan (duplikasi) kode. Langkah-langkah program yang sama dan dipakai berulang-ulang di program dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi-fungsi. Selanjutnya bagian program yang membutuhkan langkah-langkah ini tidak perlu selalu menuliskannya, tetapi cukup memanggil fungsi-fungsi tersebut.

Dasar Fungsi

Fungsi standar C yang mengemban tugas khusus contohnya adalah ;

- *printf()* , yaitu untuk menampilkan informasi atau data ke layar.
- *scanf()* , yaitu untuk membaca kode tombol yang diinputkan.

Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai argumen. Nilai masukan ini akan diolah oleh fungsi. Hasil akhir fungsi berupa sebuah nilai (disebut sebagai *return value* atau nilai keluaran fungsi). Oleh karena itu fungsi sering digambarkan sebagai "kotak gelap" seperti ditunjukkan pada gambar di bawah ini.

Penggambaran sebagai kotak gelap di antaranya menjelaskan bahwa bagian dalam fungsi bersifat pribadi bagi fungsi. Tak ada suatu pernyataan di luar fungsi yang bisa mengakses bagian dalam fungsi, selain melalui parameter (atau variabel eksternal yang akan dibahas belakangan). Misalnya melakukan *goto* dari pernyataan di luar fungsi ke pernyataan dalam fungsi adalah tidak diperkenankan.

Bentuk umum dari definisi sebuah fungsi adalah sebagai berikut ;

```
tipe-keluaran-fungsi  nama-fungsi (deklarasi argumen)
{
     tubuh fungsi
}
```

Keterangan :

- **tipe-keluaran-fungsi**, dapat berupa salah satu tipe data C, misalnya *char* atau *int* . Kalau penentu tipe tidak disebutkan maka dianggap bertipe *int* (secara *default*).
- **tubuh fungsi** berisi deklarasi variabel (kalau ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan. Tubuh fungsi ini ditulis di dalam tanda kurung kurawal buka dan kurung kurawal tutup.

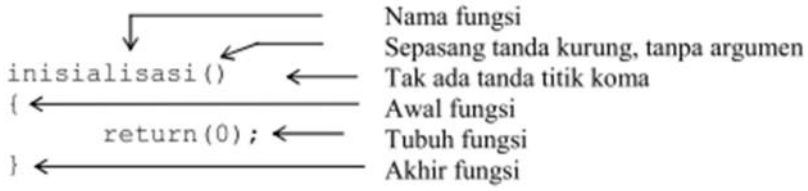
Sebuah fungsi yang sederhana bisa saja tidak mengandung parameter sama sekali dan tentu saja untuk keadaan ini deklarasi parameter juga tidak ada. Contoh :

```
int inisialisasi()
{
     return(0);
}
```

```
inisialisasi()
{
      return(0);
}
```

Pada fungsi di atas :

- tipe keluaran fungsi tidak disebutkan, berarti keluaran fungsi ber tipe *int*.
- `inisialisasi` adalah nama fungsi
- Tanda `()` sesudah nama fungsi menyatakan bahwa fungsi tak memiliki parameter.
- Tanda `{` dan `}` adalah awal dan akhir fungsi
- `return(0)` merupakan sebuah pernyataan dalam tubuh fungsi.



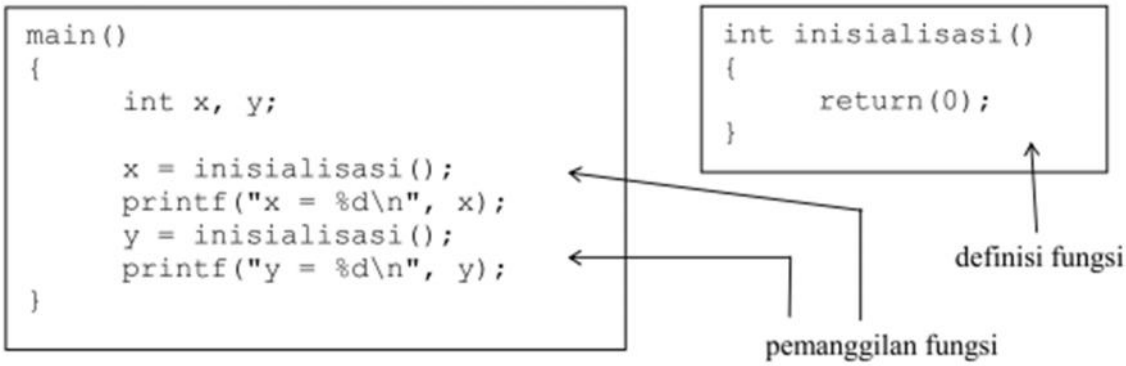
Memberikan Nilai Keluaran Fungsi

Suatu fungsi dibuat untuk maksud menyelesaikan tugas tertentu. Suatu fungsi dapat hanya melakukan suatu tugas saja tanpa memberikan suatu hasil keluaran atau melakukan suatu tugas dan kemudian memberikan hasil keluaran. Fungsi yang hanya melakukan suatu tugas saja tanpa memberikan hasil keluaran misalnya adalah fungsi untuk menampilkan hasil di layar.

Dalam tubuh fungsi, pernyataan yang digunakan untuk memberikan nilai keluaran fungsi berupa *return.* Sebagai contoh, pada fungsi **inisialisasi()** di atas terdapat pernyataan

```
return(0);
```

merupakan pernyataan untuk memberikan nilai keluaran fungsi berupa nol.



Program di atas sekaligus menjelaskan bahwa suatu fungsi cukup didefinisikan satu kali tetapi bisa digunakan beberapa kali. Pada keadaan semacam ini seandainya tubuh fungsi mengandung banyak pernyataan, maka pemakaian fungsi dapat menghindari duplikasi kode dan tentu saja menghemat penulisan program maupun kode dalam memori.

Misalnya pada saat pernyataan

```
x = inisialisasi();
```

ijalankan, mula-mula eksekusi akan diarahkan ke fungsi **inisialisasi()**, selanjutnya suatu nilai keluaran (hasil fungsi) akhir fungsi diberikan ke **x**. Proses yang serupa, dilakukan untuk pernyataan

```
y = inisialisasi();
```

Bagi suatu fungsi, jika suatu pernyataan *return* dieksekusi, maka eksekusi terhadap fungsi akan berakhir dan nilai pada parameter *return* akan menjadi keluaran fungsi. Untuk fungsi yang tidak memiliki pernyataan *return*, tanda } pada bagian akhir fungsi akan menyatakan akhir eksekusi fungsi.

Di bawah ini diberikan contoh sebuah fungsi yang mengandung dua buah pernyataan *return*. Fungsi digunakan untuk memperoleh nilai minimum di antara 2 buah nilai yang menjadi parameternya.

```
int minimum(int x, int y)
{
     if (x < y)
          return(x);
     else
          return(y);
}
```

Pada fungsi di atas terdapat dua buah parameter berupa **x** dan **y**. Oleh karena itu fungsi juga mengandung bagian untuk mendeklarasikan parameter, yang menyatakan **x** dan **y** bertipe *int*. Adapun penentuan nilai keluaran fungsi dilakukan pada tubuh fungsi, berupa pernyataan

```
if (x < y)
     return(x);
else
     return(y);
```

yang menyatakan :

- jika x < y maka nilai keluaran fungsi adalah sebesar nilai x.
- untuk keadaan lainnya (x >= y) maka keluaran fungsi adalah sebesar y.

Fungsi Dengan Keluaran Bukan Integer

Untuk fungsi yang mempunyai keluaran bertipe bukan integer, maka fungsi haruslah didefiniskan dengan diawali tipe keluaran fungsinya (ditulis di depan nama fungsi). Sebagai contoh untuk menghasilkan nilai terkecil di antara dua buah nilai real, maka definisinya berupa :

```
float minimum(float x, float y)
{
     if (x < y)
          return(x);
     else
          return(y);
}
```

Perhatikan, di depan nama **minimum** diberikan tipe keluaran fungsi berupa *float*. Seluruh parameter sendiri juga didefinisikan dengan tipe *float*.

Khusus untuk fungsi yang dirancang tanpa memberikan nilai keluaran (melainkan hanya menjalankan suatu tugas khusus) biasa didefinisikan dengan diawali kata kunci *void* (di depan nama fungsi).

Prototipe Fungsi

Prototipe fungsi digunakan untuk menjelaskan kepada kompiler mengenai :

- tipe keluaran fungsi
- jumlah parameter
- tipe dari masing-masing parameter.

Bagi kompiler, informasi dalam prototipe akan dipakai untuk memeriksa keabsahan (validitas) parameter dalam pemanggilan fungsi. Salah satu keuntungannya adalah, kompiler akan melakukan konversi seandainya antara tipe parameter dalam fungsi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukan kesalahan bila jumlah parameter dalam definisi dan saat pemanggilan berbeda.

Contoh prototipe fungsi;

```
float jumlah (float x, float y);
```

atau

```
float jumlah (float, float);
```

Penjelasannya adalah sbb :

```
float jumlah (float, float);
```

Nama fungsi
Diakhiri dengan titik koma
Tipe parameter kedua
Tipe parameter pertama
Tipe keluaran fungsi

Untuk fungsi yang tidak memiliki argumen (contoh program **void.c**), maka deklarasinya adalah

```
void info_program(void);
```

menyatakan bahwa **info_program()** tidak memiliki parameter

Catatan :

- Untuk fungsi-fungsi pustaka, prototipe dari fungsi-fungsi berada di file-file judulnya *(header file)*. Misalnya fungsi pustaka *printf()* dan *scanf()* prototipenya berada pada file dengan nama **stdio.h**
- Untuk fungsi pustaka pencantuman pada prototipe fungsi dapat dilakukan dengan menggunakan *preprocessor directive* **#include**.

D. Percobaan

Percobaan 1

```
#include <stdio.h>

int inisialisasi()
{
    return(0);
}

main()
{
    int x, y;

    x = inisialisasi();
    printf("x = %d\n", x);
    y = inisialisasi();
    printf("y = %d\n", y);
}
```

Percobaan 2

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

```
int minimum(int x, int y)
{
	if(x < y)
		return(x);
	else
		return(y);
}


main()
{
	int a = 20;
	int b = 44;
	int hasil;

	hasil = minimum(a, b);
	printf("Nilai terkecil = %d\n", hasil);

	printf("Nilai terkecil = %d\n", minimum(3, 2));
}
```

Percobaan 3

```
#include <stdio.h>

float jumlah (float, float);

main()
{
	int a = 6;
	int b = 3;
	float c;

	c = jumlah(a, b);
	printf("c = a + b = %.2f\n", c);
	printf("hasil penjumlahan = %.2f\n", jumlah(20.1, 0.9));
}

float jumlah(float x, float y)
{
	return(x + y);
}
```

Percobaan 4

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

```
void fungsi(void);

main()
{
	int i = 20;

	fungsi();
	printf("\ni dalam main() = %d", i);
}

void fungsi(void)
{
	int i = 1;
	printf("i dalam fungsi() = %d", i);
}
```

Percobaan 5

```
#include <stdio.h>

int i = 273;

void tambah(void);

main()
{
	printf("Nilai awal i = %d\n", i);
	i = i +7;
	printf("Nilai i kini= %d\n", i);
	tambah();
	printf("Nilai i kini= %d\n", i);
	tambah();
	printf("Nilai i kini= %d\n", i);
}

void tambah(void)
{
	i++;
}
```

Percobaan 6

```
#include <stdio.h>

void tampil10kali()
{
	int i;
```

```
        for(i = 0; i < 10; i++)
        {
                printf("I Love C\n");
        }
}

main()
{
        printf("\n");
        tampil10kali();
}
```

E. Latihan

Buatlah program untuk menghitung luas persegi dimana program tersebut terdiri dari 2 buah fungsi yaitu fungsi main( ) dan fungsi hitung_luas( ). Pada fungsi hitung_luas( ) terdapat inputan nilai sisi persegi dan rumus untuk menghitung luas persegi. Pada fungsi main( ) terdapat pemanggilan fungsi hitung_luas( ) dan menampilkan hasil dari luas persegi tersebut.

Selamat ngoding ☺

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 14**
Kode : ELA1.61.2306	**Judul : Struktur**

A. LEARNING OUTCOMES PRAKTIKUM

1. Membuat program dengan menggunakan struktur.

B. Alat dan Bahan

1. Personal computer
2. Aplikasi Turbo C++

C. Teori Singkat

Untuk menyimpan banyak data pada satu variable, biasanya kita menggunakan array. Array mampu menyimpan banyak data dalam satu variable, yaitu dengan ditempatkan dalam index yang berbeda. Tapi kita juga bisa menyimpan banyak data dalam satu index, bahkan dengan tipe data yang berbeda. Untuk dapat melakukan hal ini, kita menggunakan structure.

Structure adalah kumpulan dari beberapa variable yang dikelompokan dalam satu nama untuk kemudahan dalam penggunaan variable-variable tersebut. Structure mampu mengelompokan variable dengan tipe data yang bebrbeda dalam satu nama. Structure sangat diperlukan terutama dalam program-program yang besar. Karena, biasanya program-program yang besar menggunakan data yang kompleks, yang harus diolah dan digunakan secara bersamaan. Strucutre pada beberapa bahasa pemrograman lain disebut "record", misalnya pascal.

Contoh penggunaan structure misalnya pada seorang mahasiswa. Dalam program, kita bisa membuat mahasiswa sebagai sebuah variable. tapi dalam dunia nyata mahasiswa memiliki banyak data, antara lain alamat, nama, umur, NIM, dll. Pada program yang besar, data-data yang dimiliki mahasiswa harus dikelola secara bersamaan. Dan pada program, kita akan sangat kerepotan apabila menggunakan satu buah variable saja. Untuk mengatasi masalah ini, kita memerlukan structure untuk mengelompokan data dan mempermudah dalam mengolahnya.

Operasi yang bisa dilakukan terhadap structure antara lain, penggandaan (copy), sebagai parameter fungsi, dan sebagai nilai kembalian dari fungsi. Structure juga bisa digabungkan dengan array, yaitu dengan memberi index pada nama structure.

Format deklarasi structure adalah sebagai berikut:

```
struct                                                    <NamaStructure>
{
member
member
member
}
```

Struct merupakan suatu struktur data yang menggabungkan beberapa data dengan berbagai tipe data yang memiliki ukuran yang berbeda (heterogen) di kelompokan dalam satu deklarasi unik dan saling berkaitan, dengan format sbb :

```
struct model_name {
type1 element1;
type2 element2;
type3 element3;
.
.
} object_name;
```

Dimana model_name adalah nama untuk model tipe stukturnya dan parameter optional object_name merupakan identifier yang valid untuk objek struktur. Diantara kurung kurawal { } berupa tipe dan sub-identifier yang mengacu ke elemen pembentuk struktur. Jika pendefinisian stuktur menyertakan parameter model_name (optional), maka parameter tersebut akan menjadi nama tipe yang valid ekuivalen dengan struktur. Contoh :

```
struct products {
char name [30];
float price;
} ;
products apple;
products orange, melon;
```

Didefinisikan model struktur products dengan dua field : name danprice, dengan tipe yang berbeda. Kemudian tipe struktur tadi (products) digunakan untuk mendeklarasikan tiga objek : apple, orangedan melon. Ketika dideklarasikan, products menjadi nama tipe yang valid seperti tipe dasar int, char atau short dan dapat mendeklarasikan objects (variables) dari tipe tersebut. Optional field yaitu object_name dapat dituliskan pada akhir deklarasi struktur untuk secara langsung mendeklarasikan object dari tipe struktur. Contoh :

```
struct products {
char name [30];
float price;
} apple, orange, melon;
```

Sangat penting untuk membedakan antara structure model,dan structure object. Model adalah type, dan object adalah variable. Kita dapat membuat banyak objects (variables) dari satu model (type).Struct dapat dideklarasikan secara bertingkat, yaitu salah satu field struct bertipe struct lainnya (nested Structure) . Selain itu struct juga dapat digabungkan dengan array, struct yang field-nya berupa array atau array yang setiap elemennya berupa structure.

D. Percobaan

Percobaan 1

```
#include <stdio.h>

main()
{
      struct titik
      {
      int x;
      int y;
      };

   struct titik koordinat;
   koordinat.x = 5;
   koordinat.y = 10;

   printf("Titik terletak pada: %d, %d\n", koordinat.x, koordinat.y);
   printf("\n --- end of code --- \n");
}
```

Percobaan 2

```
#include <stdio.h>

main()
{
      int a,b,c;

       struct info
       {
              int nomor;
              int nim;
              char nama[20];
              char nilai[2];
       };
```

```
        struct info mahasiswa[10];
        printf("Masukkan Jumlah Data Mahasiswa (Maks. 10 orang) : ");
        scanf ("%d", &a);

        for(b=0;b<a;b++)
        {
                mahasiswa[b].nomor=b+1;
                printf("\n%d.\n", b+1);
                printf("NIM : ");
                scanf ("%d", &mahasiswa[b].nim);
                printf("Nama : ");
                scanf("%s", &mahasiswa[b].nama);
                printf("Nilai : ");
                scanf("%s", &mahasiswa[b].nilai);
        }

        printf("\n%d Data Mahasiswa Telah Dimasukkan", b);
        printf("\nNOMOR\t NIM\t Nama\t\t Nilai\n");

        for (c=0;c<b;c++)
        {
                printf("\n %d\t %d\t %s\t\t %s\t\n", mahasiswa[c].nomor,
                                        mahasiswa[c].nim,
                                        mahasiswa[c].nama,
                                        mahasiswa[c].nilai);
        }
}
```

Percobaan 3

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

main()
{
        struct data_tanggal   //definisi tipe data_tanggal
        {
                int tanggal;
                int bulan;
                int tahun;
        };

        struct data_rekan   ///definisi tipe data_rekan
        {
                char nama[31];
                struct data_tanggal tgl_lahir;
        };
```

```
        //deklarasi variabel struktur info_rekan
        struct data_rekan info_rekan;

        //pemasukan data ke elemen variabel struktur
        strcpy(info_rekan.nama, "AURORA");
        info_rekan.tgl_lahir.tanggal = 14;
        info_rekan.tgl_lahir.bulan = 2;
        info_rekan.tgl_lahir.tahun = 2000;

        //menampilkan elemen variabel struktur
        printf("Nama         : %s\n", info_rekan.nama);
        printf("Tanggal lahir : %d-%d-%d\n",
                                                info_rekan.tgl_lahir.tanggal,
                                                info_rekan.tgl_lahir.bulan,
                                                info_rekan.tgl_lahir.tahun);
    }
```

Percobaan 4

```
    #include <stdio.h>
    main()
    {
        struct zodiak  //definisi tipe struktur
        {
                char nama[20];
                int tgl_awal;
                int bln_awal;
                int tgl_akhir;
                int bln_akhir;
        };

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

        int tg_lhr, bl_lhr, th_lhr;

        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 adalah %s\n", bintang.nama);
        else
                printf("Bintang anda bukanlah %s\n", bintang.nama);
    }
```

Fakultas Teknik UNP Padang	**Waktu : 3 x 50”**
Jurusan : Teknik Elektronika	**Mata Kuliah : Praktikum Pemograman komputer**
Prodi : Pendd.Teknik Informatika	**Topik : Job Sheet 15 dan 16**
Kode : ELA1.61.2306	**Judul : Struktur**

A. LEARNING OUTCOMES PRAKTIKUM

1. Membuat program dengan menggunakan struktur.

B. Alat dan Bahan

1. Personal computer
2. Aplikasi Turbo C++

C. Teori Singkat

Untuk menyimpan banyak data pada satu variable, biasanya kita menggunakan array. Array mampu menyimpan banyak data dalam satu variable, yaitu dengan ditempatkan dalam index yang berbeda. Tapi kita juga bisa menyimpan banyak data dalam satu index, bahkan dengan tipe data yang berbeda. Untuk dapat melakukan hal ini, kita menggunakan structure.

Structure adalah kumpulan dari beberapa variable yang dikelompokan dalam satu nama untuk kemudahan dalam penggunaan variable-variable tersebut. Structure mampu mengelompokan variable dengan tipe data yang bebrbeda dalam satu nama. Structure sangat diperlukan terutama dalam program-program yang besar. Karena, biasanya program-program yang besar menggunakan data yang kompleks, yang harus diolah dan digunakan secara bersamaan. Strucutre pada beberapa bahasa pemrograman lain disebut "record", misalnya pascal.

Contoh penggunaan structure misalnya pada seorang mahasiswa. Dalam program, kita bisa membuat mahasiswa sebagai sebuah variable. tapi dalam dunia nyata mahasiswa memiliki banyak data, antara lain alamat, nama, umur, NIM, dll. Pada program yang besar, data-data yang dimiliki mahasiswa harus dikelola secara bersamaan. Dan pada program, kita akan sangat kerepotan apabila menggunakan satu buah variable saja. Untuk mengatasi masalah ini, kita memerlukan structure untuk mengelompokan data dan mempermudah dalam mengolahnya.

Operasi yang bisa dilakukan terhadap structure antara lain, penggandaan (copy), sebagai parameter fungsi, dan sebagai nilai kembalian dari fungsi. Structure juga bisa digabungkan dengan array, yaitu dengan memberi index pada nama structure.

Format deklarasi structure adalah sebagai berikut:

```
Struct   <NamaStructure>
```

```
{
member
member
member
}
```

Struct merupakan suatu struktur data yang menggabungkan beberapa data dengan berbagai tipe data yang memiliki ukuran yang berbeda (heterogen) di kelompokan dalam satu deklarasi unik dan saling berkaitan, dengan format sbb :

```
struct model_name {
type1 element1;
type2 element2;
type3 element3;
.
.
} object_name;
```

Dimana model_name adalah nama untuk model tipe stukturnya dan parameter optional object_name merupakan identifier yang valid untuk objek struktur. Diantara kurung kurawal { } berupa tipe dan sub-identifier yang mengacu ke elemen pembentuk struktur. Jika pendefinisian stuktur menyertakan parameter model_name (optional), maka parameter tersebut akan menjadi nama tipe yang valid ekuivalen dengan struktur. Contoh :

```
struct products {
char name [30];
float price;
} ;
products apple;
products orange, melon;
```

Didefinisikan model struktur products dengan dua field : name danprice, dengan tipe yang berbeda. Kemudian tipe struktur tadi (products) digunakan untuk mendeklarasikan tiga objek : apple, orangedan melon. Ketika dideklarasikan, products menjadi nama tipe yang valid seperti tipe dasar int, char atau short dan dapat mendeklarasikan objects (variables) dari tipe tersebut. Optional field yaitu object_name dapat dituliskan pada akhir deklarasi struktur untuk secara langsung mendeklarasikan object dari tipe struktur. Contoh :

```
struct products {
char name [30];
float price;
} apple, orange, melon;
```

Sangat penting untuk membedakan antara structure model, dan structure object. Model

adalah type, dan object adalah variable. Kita dapat membuat banyak objects (variables) dari satu model (type).Struct dapat dideklarasikan secara bertingkat, yaitu salah satu field struct bertipe struct lainnya (nested Structure) . Selain itu struct juga dapat digabungkan dengan array, struct yang field-nya berupa array atau array yang setiap elemennya berupa structure.

D. Percobaan

Percobaan 1

```
#include <stdio.h>

main()
{
      struct titik
      {
      int x;
      int y;
      };

   struct titik koordinat;
   koordinat.x = 5;
   koordinat.y = 10;

   printf("Titik terletak pada: %d, %d\n", koordinat.x, koordinat.y);
   printf("\n --- end of code --- \n");
}
```

Percobaan 2

```
#include <stdio.h>

main()
{
      int a,b,c;

       struct info
       {
                int nomor;
                int nim;
                char nama[20];
                char nilai[2];
       };
      struct info mahasiswa[10];
      printf("Masukkan Jumlah Data Mahasiswa (Maks. 10 orang) : ");
      scanf ("%d", &a);
```

```
        for(b=0;b<a;b++)
        {
                mahasiswa[b].nomor=b+1;
                printf("\n%d.\n", b+1);
                printf("NIM : ");
                scanf ("%d", &mahasiswa[b].nim);
                printf("Nama : ");
                scanf("%s", &mahasiswa[b].nama);
                printf("Nilai : ");
                scanf("%s", &mahasiswa[b].nilai);
        }

        printf("\n%d Data Mahasiswa Telah Dimasukkan", b);
        printf("\nNOMOR\t NIM\t Nama\t\t Nilai\n");

        for (c=0;c<b;c++)
        {
                printf("\n %d\t %d\t %s\t\t %s\t\n", mahasiswa[c].nomor,
                                        mahasiswa[c].nim,
                                        mahasiswa[c].nama,
                                        mahasiswa[c].nilai);
        }
}
```

Percobaan 3

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

main()
{
        struct data_tanggal   //definisi tipe data_tanggal
        {
                int tanggal;
                int bulan;
                int tahun;
        };

        struct data_rekan   ///definisi tipe data_rekan
        {
                char nama[31];
                struct data_tanggal tgl_lahir;
        };

        //deklarasi variabel struktur info_rekan
        struct data_rekan info_rekan;
```

```
        //pemasukan data ke elemen variabel struktur
        strcpy(info_rekan.nama, "AURORA");
        info_rekan.tgl_lahir.tanggal = 14;
        info_rekan.tgl_lahir.bulan = 2;
        info_rekan.tgl_lahir.tahun = 2000;

        //menampilkan elemen variabel struktur
        printf("Nama          : %s\n", info_rekan.nama);
        printf("Tanggal lahir : %d-%d-%d\n",
                                                info_rekan.tgl_lahir.tanggal,
                                                info_rekan.tgl_lahir.bulan,
                                                info_rekan.tgl_lahir.tahun);
    }
```

Percobaan 4

```
    #include <stdio.h>

    main()
    {
        struct zodiak  //definisi tipe struktur
        {
                char nama[20];
                int tgl_awal;
                int bln_awal;
                int tgl_akhir;
                int bln_akhir;
        };

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

        int tg_lhr, bl_lhr, th_lhr;

        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 adalah %s\n", bintang.nama);
        else
                printf("Bintang anda bukanlah %s\n", bintang.nama);
    }
```

Selamat ngoding ☺

DAFTAR PUSTAKA

Abdul Kadir. 2014. Pemrograman Dasar Turbo C Untuk IBM. Yogyakarta : Andi Yogyakarta.