

# UNIT 1

## Pengantar Bahasa Pemrograman C

Tujuan:

1. Mampu memahami sistematika penulisan bahasa pemrograman C beserta tipe data, variabel, operator, dan fungsi input/output
2. Mampu menuliskan bahasa pemrograman C dengan aplikasi CodeBlock
3. Mampu mempraktikkan deklarasi data dan variabel dalam bahasa pemrograman C
4. Mampu mempraktikkan perintah input dan output dalam bahasa pemrograman C
5. Mampu mempraktikkan jenis-jenis operator dan ekspresi dalam bahasa pemrograman C

Dasar Teori:

Struktur Bahasa Pemrograman C

<pre>#include &lt;stdio.h&gt;  int main() {     Pernyataan_1;     Pernyataan_2;     ....     ...     Pernyataan_n;      return 0; }</pre>	<pre>#include --&gt; preprosesor &lt;stdio.h&gt; --&gt; berkas header int --&gt; tipe data main() --&gt; fungsi utama program {} --&gt; badan isi fungsi Pernyataan_n; --&gt; isi dari fungsi  return 0; --&gt; pernyataan untuk memberi kode keluaran program</pre>
---	--

- Berkas *header* (berekstensi .h) adalah berkas yang berisi prototipe fungsi, definisi makro, definisi konstanta, definisi variabel, dan definisi tipe.
- Preprosesor *#include* adalah suatu perintah yang digunakan untuk mengatur kompiler agar membaca berkas *header* yang disertakan di belakang kata *include* saat pelaksanaan kompilasi.
- *main()* adalah fungsi yang akan dijalankan pertama kali ketika program dieksekusi.
- Nilai balik program ditentukan oleh pernyataan *return*. Pada program di atas, *return 0* menyatakan bahwa nilai balik program adalah nol, nilai nol biasanya digunakan untuk menyatakan bahwa program berhasil melaksanakan tugas yang dibebankan.
- Setiap pernyataan ditulis dengan akhiran tanda titik-koma (;).

## Variabel

Variabel adalah program yang digunakan untuk menyimpan suatu nilai tertentu di mana nilai tersebut dapat berubah-ubah. Setiap variabel mempunyai tipe dan hanya data yang bertipe sama dengan tipe variabel yang dapat disimpan di dalam variabel tersebut. Setiap variabel mempunyai nama dan pemisahan antar variabel (jika lebih dari 2) dilakukan dengan memberikan tanda koma.

Contoh:

```
int jumlah;  
float harga_per_unit, total_biaya;
```

Dari contoh di atas, variabel jumlah hanya boleh menerima data yang bertipe data integer (bulat), tidak boleh menerima data bertipe lainnya. Variabel harga\_per\_unit dan total\_biaya hanya bisa diisi dengan bilangan pecahan (*float*).

## Konstanta

Berbeda dengan variabel, isi di dalam konstanta tidak dapat diubah-ubah selama eksekusi program berlangsung. Dapat menggunakan pernyataan `const` lalu disertai dengan tipe data dan variabel atau menggunakan pernyataan `#define` yang umumnya diletakkan di posisi preprosesor.

Contoh:

```
#define phi 3.14  
const int m = 0;
```

## Tipe Data

Bahasa pemrograman C memiliki empat macam tipe data dasar, yaitu tipe data bilangan bulat, tipe data pecahan, tipe data karakter, tipe data logika. Lebih lengkapnya berikut adalah tabel tipe data beserta ukuran dan jangkauannya:

Tipe	Lebar	Jangkauan Nilai	
		Dari	Sampai dengan
int	16 bit	-32768	-32767
unsigned int	16 bit	0	65535
short int	16 bit	-32768	-32767
long int	32 bit	-2147483648	-2147483647
unsigned long int	32 bit	0	4294967295
float	32 bit	3.4E-38	3.4E+38
double	64 bit	1.7E-308	1.7E+308
long double	80 bit	3.4E-4932	3.4E+4932
char	8 bit	-128	128
unsigned char	8 bit	0	255

## Operator

Operator adalah simbol atau tanda yang jika diletakkan pada dua buah operand dapat menghasilkan sebuah hasil, operator berupa simbol yang digunakan untuk menyusun suatu ekspresi dengan melibatkan satu atau beberapa operand. Terdapat tiga jenis operator yang paling umum digunakan yaitu: operator aritmatika, operator pembanding, dan operator logika.

Operator Aritmatika	Operator Pembanding	Operator Logika
+	>	&&
-	$\geq$	$\parallel$
*	<	!
/	$\leq$	
%	$\equiv$	
++	$\neq$	
--	?:	
()		

## Fungsi printf() dan scanf()

Pernyataan printf() digunakan untuk menampilkan data ke layar. Dengan menggunakan fungsi ini, tampilan dapat diatur (diformat) dengan mudah. Bentuk umumnya adalah:

```
printf("string kendali", argumen1, argumen2, ...);
```

String kendali dapat berupa keterangan beserta penentu format (seperti %d, %f). argumen adalah data yang akan ditampilkan, dapat berupa variabel, konstanta, maupun ungkapan.

Contoh:

```
/* Program pertama */
#include <stdio.h>
main()
{
    printf("Halo dunia!");
}
```

Sedangkan fungsi scanf() merupakan fungsi yang digunakan untuk menampilkan data yang dimasukkan dari keyboard. Dikenal juga sebagai fungsi masukan atau *input* dalam bahasa pemrograman C.

### Contoh 1:

Menghitung luas dan keliling lingkaran dengan jari-jari yang dimasukkan melalui keyboard

```
#include<stdio.h>
#define phi 3.14
int main()
{
    float jari-jari, luas, keliling;
    printf("Masukan nilai jari-jari lingkaran: ");
    scanf("%f", &jari-jari);

    luas = phi*jari-jari*jari-jari;
    keliling = 2*phi*jari-jari;
    printf("\nLuas lingkaran dengan jari-jari %f adalah: %f ", jari-jari, luas);
    printf("\nKeliling lingkaran dengan jari-jari %f adalah: %f", jari-jari, keliling);

    return 0;
}
```

### Contoh 2:

Menukar dua buah nilai dari dua buah variabel. Misalnya, sebelum pertukaran nilai a adalah 5 dan b adalah 3, maka setelah penukaran nilai a = 3 dan b = 5.

```
#include<stdio.h>
int main()
{
    int a,b,c;
    printf("Program Menukar 2 Buah Nilai \n\n");
    printf("Sebelum ditukar\n");
    printf("-----\n");
    printf("Bilangan pertama = ");
    scanf("%i", &a);
    printf("Bilangan kedua = ");
    scanf("%i", &b);
    c=a;
    a=b;
    b=c;
    printf("Setelah ditukar\n");
    printf("-----\n");
    printf("Bilangan pertama = %i \n", a);
    printf("Bilangan kedua = %i \n", b);

    return 0;
}
```

### Contoh 3:

Mengkonversikan total detik menjadi jam menit detik

Petunjuk: 1 menit = 60 detik dan 1 jam = 3600 detik

```
#include<stdio.h>
int main()
{
    int totdet, jam, sisa, menit, detik;
    printf("Program Konversi Detik menjadi Jam Menit Detik\n");
    printf("-----\n");
    printf("Bilangan total detik = ");
    scanf("%i", &totdet);
    jam=totdet/3600;
    sisa=totdet%3600;
    menit=sisa/60;
    detik=sisa%60;
    printf("Jumlah jam = %i jam \n", jam);
    printf("Jumlah menit = %i menit \n", menit);
    printf("Jumlah detik = %i detik \n", detik);

    return 0;
}
```

### Tugas Unit 1:

1. Buatlah algoritma (flowchart dan pseudocode) beserta program (bahasa C) dengan spesifikasi sebagai berikut:
  - Menampilkan tulisan “Halo, siapa nama Anda?”, lalu
  - Meminta pengguna untuk memasukkan namanya, dan akhirnya
  - Menuliskan pesan “Senang berteman denganmu, <nama>”, yang dalam hal ini <nama> adalah string yang dibaca sebelumnya
2. Seorang *user* di warnet mulai menggunakan internet pada pukul 11 dan selesai pada pukul 14. Bila tarif penggunaan warnet tersebut 1 jamnya adalah Rp8.000, maka buatlah algoritma (flowchart dan pseudocode) beserta programnya (bahasa C) untuk menghitung lama pemakaian (dalam jam menit detik) dan biaya yang harus dibayar oleh *user* tersebut.
3. Berikan penjelasan analisis program pada setiap program yang dibuat pada poin 1 dan 2.

## UNIT 2

### Percabangan

Tujuan:

1. Mampu memahami struktur penulisan percabangan menggunakan IF dan SWITCH
2. Mampu membuat program menggunakan kendali percabangan
3. Mampu menganalisis program yang memiliki percabangan di dalamnya

Dasar Teori:

Pada algoritma, percabangan merupakan suatu fungsi kendali ada program. Percabangan hanya mengenal keadaan benar atau salah, *true* atau *false*, 1 atau 0 atau yang dikenal dengan istilah boolean. Pada percabangan, jika suatu pernyataan menunjukkan nilai benar, maka suatu kondisi akan dieksekusi, namun jika menunjukkan nilai yang salah maka akan dieksekusi kondisi yang lainnya.

Terdapat 2 jenis percabangan yaitu percabangan IF ELSE dan percabangan SWITCH. Bentuk umum dari kedua kondisi tersebut adalah sebagai berikut:

```
if (kondisi_1) {
    Pernyataan jika kondisi_1 benar;
}
else if (kondisi_2) {
    Pernyataan jika kondisi_2 benar;
}
else {
    Pernyataan jika seluruh kondisi salah;
}
```

Bentuk umum SWITCH adalah sebagai berikut:

```
switch (kondisi) {
    case konstanta1:
        Pernyataan 1; break;
    case konstanta2:
        Pernyataan 2; break;
    .
    .
    default:
        Pernyataan jika seluruh case tidak terpenuhi;
        break;
}
```

## Contoh Program

Contoh 1. Program untuk menentukan diskon berdasarkan total penjualan. Masukan yang dibutuhkan adalah harga barang dan jumlah pembelian. Menghasilkan total penjualan:

1. Bila total penjualan di atas 100.000, maka mendapat diskon 20% dari total
2. Bila total penjualan lebih dari 50.000, maka mendapat diskon 10% dari total
3. Selain itu, tidak mendapatkan diskon

```
#include<stdio.h>
int main()
{
    float harga, total, diskon;
    int jumlah;

    printf("Program hitung diskon\n");
    printf("-----\n");
    printf("Harga barang Rp");
    scanf("%f", &harga);
    printf("Jumlah beli: ");
    scanf("%d", &jumlah);

    total = harga * jumlah;
    printf("\nTotal harga adalah Rp%f\n", total);
    printf("Mendapat diskon Rp");

    if(total > 100000) {
        diskon = total * 0.2;
    }
    else if (total > 50000) {
        diskon = total * 0.1;
    }
    else {
        diskon = 0;
    }

    printf("%f\n\n", diskon);
    printf("Maka, Total yang harus dibayar adalah Rp%f\n", total-diskon);
    return 0;
}
```

Jalankan program di atas, isikan harga barang dan jumlah beli. Lihat apa hasil dari program tersebut. Isikanlah dengan beberapa skenario nilai, dari yang akan menghasilkan total di atas 100000, di atas 50000, dan di bawah 50000. Silakan dianalisis program tersebut.

Contoh 2. Program sederhana *nested if*

```
#include<stdio.h>
int main()
{
    int id;
    char ukuran;
    long harga = 0;

    printf("\t\tToko Butik\n");
    printf("=====*\n");
    printf("1. Baju Kaos\n");
    printf("2. Celana Kaos\n");
    printf("=====*\n");
    printf("Pilih no: ");
    scanf("%d",&id);

    printf("Pilih Ukuran [S/M/L]: ");
    scanf("%c",&ukuran);

    if(id==1) {
        if(ukuran == 'S' || ukuran == 's') {
            harga = 58000;
        } else if(ukuran == 'M' || ukuran == 'm') {
            harga = 65000;
        } else {
            harga = 87000;
        }
    }
    else if(id==2) {
        if(ukuran == 'S' || ukuran == 's') {
            harga = 96000;
        } else if(ukuran == 'M' || ukuran == 'm') {
            harga = 112000;
        } else {
            harga = 130000;
        }
    }
    else {
        printf("Item tidak terdaftar");

        printf("Harga: %l", harga);
    }
}
```

Contoh 3. Program sederhana *switch*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    char nama[15], ket[30], kode;
    printf("Masukkan nama mahasiswa: ");
    scanf("%s", &nama);
    printf("Pilih kode Program Studi [A/B/C/D]: ");
    kode=getche();

    switch (kode) {
        case 'A':
            strcpy(ket, "Program Studi Teknik Perkapalan");
            break;
        case 'B':
            strcpy(ket, "Program Studi Teknik Industri");
            break;
        case 'C':
            strcpy(ket, "Program Studi Teknik Mesin");
            break;
        case 'D':
            strcpy(ket, "Program Studi Teknik Elektro");
            break;
        case 'a':
            strcpy(ket, "Program Studi Teknik Perkapalan");
            break;
        case 'b':
            strcpy(ket, "Program Studi Teknik Industri");
            break;
        case 'c':
            strcpy(ket, "Program Studi Teknik Mesin");
            break;
        case 'd':
            strcpy(ket, "Program Studi Teknik Elektro");
            break;
        default:
            strcpy(ket, "Program Studi Tidak Ditemukan");
            break;
    }
    printf("\n\nNama Mahasiswa: %s \n", nama);
    printf("Kode Program Studi: %c \n", kode);
    printf("Nama Program Studi: %s \n", ket);
    getch();
    return 0;
}
```

Tugas Unit 2:

1. Misalkan karyawan PT. Indonesia Sejahtera Bersama dikelompokkan berdasarkan golongannya. Upah per jam setiap karyawannya bergantung pada golongannya, dengan ketentuan:

Golongan	Upah per jam
A	Rp52.000
B	Rp65.000
C	Rp78.000
D	Rp112.000

Jumlah jam kerja normal selama satu minggu adalah 32 jam. Kelebihan jam kerja dianggap sebagai lembur dengan upah lemburnya adalah Rp45.000 per-jamnya untuk semua golongan karyawan. Buatlah algoritma pseudocode dan program untuk menghitung gaji karyawan bulanan. Data yang dimasukkan dari keyboard adalah nama karyawan, golongan, dan jumlah jam kerja. Data yang dicetak ke layar adalah nama karyawan dan gaji perbulannya. Program dibuat dengan menggunakan struktur IF dan SWITCH CASE.

2. Buatlah analisis program di atas.

## UNIT 3

### **Perulangan (*Looping*)**

Tujuan:

1. Mampu memahami struktur penulisan perulangan menggunakan FOR, WHILE, dan DO WHILE
2. Mampu membuat program menggunakan kendali perulangan
3. Mampu menganalisis program yang memiliki perulangan di dalamnya

Dasar Teori:

Pada algoritma, perulangan merupakan suatu fungsi kendali pada program. Fungsi perulangan bertujuan untuk mengulang alir program sebanyak kondisi yang ditentukan. Perintah ini sangat sering digunakan pada program yang mengharuskan pembacaan data yang jumlahnya banyak, menampilkan data, melakukan pencarian, pengurutan data, dan lain sebagainya.

Terdapat 3 jenis perulangan yaitu perulangan FOR, WHILE, dan DO WHILE. Ketiga perulangan tersebut memiliki bentuk umum adalah sebagai berikut. Pertama adalah bentuk umum FOR:

```
for (nilai_awal; ekspresi_kondisi; peubah_kondisi)
{
    Pernyataan_yang_akan_diulang;
}
```

Kedua, bentuk umum WHILE adalah sebagai berikut:

```
while (ekspresi_kondisi)
{
    Pernyataan_yang_akan_diulang;
    Counter;
}
```

Terakhir adalah perulangan DO WHILE yang mana serupa dengan struktur perulangan WHILE, hanya pernyataan dieksekusi pada perintah DO sedangkan perintah WHILE menentukan kondisi kapan perintah DO dilaksanakan. Bentuk umumnya adalah sebagai berikut:

```
do {
    pernyataan_yang_akan_diulang;
}
while (ekspresi_kondisi);
```

### Contoh 1: Perulangan FOR sederhana

```
/* Program mencari data terbesar dan terkecil */
#include <stdio.h>
int main () {
    int n, i, max, min, bil;
    printf("Program mencari data terbesar dan terkecil \n\n");
    printf("Masukkan banyaknya data: ");
    scanf("%d",&bil);
    max = bil;
    min = bil;

    for(i = 2; i <= n; i++) {
        printf("Masukkan bilangan ke-%d: ",i);
        scanf("%d",&bil);
        if(bil > max) {
            max = bil;
        }
        else if(bil < min) {
            min = bil;
        }
    }
    printf("\nData terbesar adalah %d\n",max);
    printf("\nData terkecil adalah %d\n",min);

    return 0;
}
```

### Contoh 2: Struktur FOR bersarang bagian 1

```
/* Program membentuk persegi */
#include <stdio.h>
int main () {
    int x, y;

    for(y = 1; y <= 4; y++) {
        for(x = 1; x <= 0; x++) {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

### Contoh 3: Struktur FOR bersarang bagian 2

```
/* Program membentuk persegi */
#include <stdio.h>
int main () {
    int n, x, y;
    printf("Masukkan tinggi segitiga: ");
    scanf("%d", &n);

    for(x = 1; x <= n; x++) {
        for(y = 1; y <= x; y++) {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

### Contoh 4: Perulangan WHILE

```
/* Program menghitung rata-rata bilangan */
#include <stdio.h>
int main () {
    int n = 0;
    float jumlah = 0, bil, rata;
    char lagi = 'Y';
    while ((lagi=='Y') || (lagi=='y')) {
        printf("Masukkan bilangan: ");
        scanf("%f", &bil);
        jumlah = jumlah+bil;
        n = n+1;
        printf("Apakah Anda akan memasukkan data lagi? [Y/T]: ");
        scanf("%s", &lagi);
        printf("\n");
    }
    rata = jumlah/n;
    printf("\nBanyaknya bilangan: %i \n", n);
    printf("Rata-rata bilangan: %.2f \n", rata);
    return 0;
}
```

### Contoh 5: Perulangan DO WHILE

```
include <stdio.h>
int main () {
    int pilih;
    do {
        printf("DAFTAR MENU MAKANAN\n");
        printf("-----\n");
        printf("1.\tPaket Ayam Goreng\n");
        printf("2.\tPaket Ayam Bakar\n");
        printf("3.\tPaket Ayam Geprek\n");
        printf("4.\tPaket Ayam Penyet\n");
        printf("\nMasukkan pilihan Anda (0=selesai): ");
        scanf("%d", &pilih);

        switch(pilih) {
            case 1:
                printf("Harga Paket Ayam Goreng Rp20.000\n");
                break;
            case 2:
                printf("Harga Paket Ayam Bakar Rp20.000\n");
                break;
            case 3:
                printf("Harga Paket Ayam Bakar Rp20.000\n");
                break;
            case 4:
                printf("Harga Paket Ayam Bakar Rp20.000\n");
                break;
            case 0:
                printf("Selesai\n");
                break;
            default:
                printf("Pilihan hanya 1 - 4, pilih 0 jika ingin selesai");
                break;
        }
    } while (pilih != 0);
    return 0;
}
```

Tugas Unit 3:

1. Buatlah analisis program untuk setiap contoh program di atas.
2. Buatlah algoritma pseudocode dan program bahasa C untuk menghitung jumlah N buah bilangan genap atau ganjil pertama (user memilih opsi apakah menghitung jumlah N buah bilangan genap pertama atau jumlah N buah bilangan ganjil pertama). Catatan: N adalah bilangan bulat tidak negatif.
3. Buatlah analisis program nomor 2 di atas.

## UNIT 4

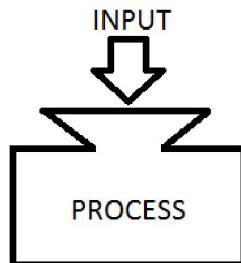
### Fungsi dan Prosedur

Tujuan:

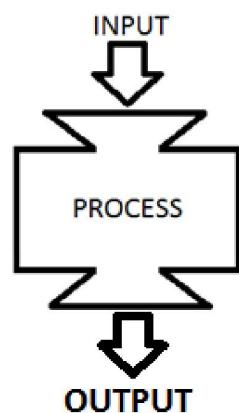
1. Mampu memahami struktur penulisan Fungsi dan Prosedur pada bahasa pemrograman C
2. Mampu membuat program menggunakan Fungsi dan Prosedur
3. Mampu menganalisis program yang memiliki Fungsi dan Prosedur di dalamnya

Dasar Teori:

Prosedur atau *subroutine* merupakan blok rangkaian kode program yang mengerjakan sesuatu sebagai bagian pembentuk suatu program yang tidak memberikan nilai kembalian pada pemanggilnya. Sedangkan Fungsi merupakan blok kode program yang mengerjakan sesuatu untuk menghasilkan suatu nilai yang akan diterima secara *assignment* pada pemanggilannya. Ilustrasi dari Fungsi dan Prosedur dapat ditunjukkan pada gambar di bawah ini:



Gambar 1. Ilustrasi Prosedur



Gambar 2. Ilustrasi Fungsi

Bentuk umum dari Prosedur adalah sebagai berikut:

```
void namasubroutine([parameter])
{
    [Perintah-perintah/Blok Kode Program];
}
```

Sedangkan bentuk umum Fungsi adalah sebagai berikut:

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

Baik Fungsi maupun Prosedur harus dideklarasikan terlebih dahulu di atas sebelum fungsi *int main()*, karena sistem pembacaan *compiler* adalah dari atas ke bawah. Jika, suatu fungsi ditempatkan di bawah fungsi *int main()*, maka sebelum fungsi *main* harus dideklarasikan terlebih dahulu juga fungsi tersebut.

Selain Fungsi dan Prosedur. Terdapat juga variabel lokal, variabel global, dan variabel statis, yang dijelaskan pada sintaks di bawah ini.

```
int fungsi();
int i, j;                                // Variabel Global

int main () {
    int k, l;                                // Variabel Lokal
}

int fungsi() {
    static int m, n;                          // Variabel Statis
}
```

Contoh 1:

```
#include <stdio.h>
int lebihbesar(int a, int b) {
    if (a>b) {
        return a;
    }else {
        return b;
    }
}
int main()
{
    int x, y, nilai;
    printf("Masukkan nilai X: "); scanf("%d",&x);
    printf("Masukkan nilai Y: "); scanf("%d",&y);
    nilai = lebihbesar(x,y);
    printf("Nilai terbesar adalah: %d\n", nilai);

    return 0;
}
```

Contoh 2:

```
#include <stdio.h>
int jumlah(int x, int y);
int main()
{
    int a,b,c;
    printf("A = "); scanf("%d", &a);
    printf("B = "); scanf("%d", &b);
    c = jumlah(a,b);
    printf("Jumlah kedua bilangan A dan B adalah: %d \n", c);

    return 0;
}
int jumlah(int a, int b) {
    int hasil;
    hasil = x + y;
    return (hasil);
}
```

Contoh 3:

```
#include <stdio.h>
int checkPrimeNumber(int n);
int main() {
    int n1, n2, i, flag;
    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);
    printf("Prime numbers between %d and %d are: ", n1, n2);
    for (i = n1 + 1; i < n2; ++i) {
        flag = checkPrimeNumber(i);

        if (flag == 1)
            printf("%d ", i);
    }
    return 0;
}

int checkPrimeNumber(int n) {
    int j, flag = 1;
    for (j = 2; j <= n / 2; ++j) {
        if (n % j == 0) {
            flag = 0;
            break;
        }
    }
    return flag;
}
```

Contoh 4:

```
#include <stdio.h>
int power(int n1, int n2);
int main() {
    int base, a, result;
    printf("Enter base number: ");
    scanf("%d", &base);
    printf("Enter power number(positive integer): ");
    scanf("%d", &a);
    result = power(base, a);
    printf("%d^%d = %d", base, a, result);
    return 0;
}

int power(int base, int a) {
    if (a != 0)
        return (base * power(base, a - 1));
    else
        return 1;
}
```

Tugas Unit 4:

1. Buatlah analisis untuk program Contoh 3 dan Contoh 4.
2. Buatlah kembali program yang ada di Unit 2 dan Unit 3 menggunakan Fungsi, tanpa perlu membuat kembali algoritma baik flowchart maupun pseudocode.
3. Buatlah analisis untuk program di poin nomor 2.

## UNIT 5

### Larik / Array

Tujuan:

1. Mampu memahami struktur penulisan larik atau *array*
2. Mampu membuat program menggunakan larik atau *array*
3. Mampu menganalisis program yang memiliki larik atau *array* di dalamnya

Dasar Teori:

Larik merupakan suatu tipe data yang terstruktur dan dapat digunakan untuk menyimpan data yang memiliki tipe data yang sama. Dengan kata lain, larik adalah kumpulan data yang memiliki tipe data yang sama. Larik dapat diibaratkan seperti sebuah tabel. Larik dapat digunakan untuk menyimpan data yang berjumlah banyak, namun masih memiliki suatu hubungan atau terdapat kesamaan antar data tersebut.

Terdapat 2 jenis larik, yaitu larik 1 dimensi dan larik 2 dimensi. Perbedaan mendasar antara kedua jenis larik ini adalah, larik dengan 1 dimensi merupakan larik yang dapat digambarkan sebuah baris. Selain itu, dalam larik 1 dimensi, elemen yang ada di dalamnya dapat diakses hanya dengan menggunakan 1 indeks saja. Sedangkan larik 2 dimensi merupakan larik yang dapat digambarkan seperti sebuah matrik. Selain itu, elemen yang ada dalam larik 2 dimensi dapat diakses dengan menggunakan 2 indeks, yaitu indeks kolom dan juga indeks baris. Berikut format pendeklarasian larik:

tipe\_data nama\_larik[jumlah\_elelen]

adapun elemen dalam larik dituliskan sebagai berikut: (contoh pada 8 elemen larik 1 dimensi)



Contoh 1: Larik Dimensi 1 - Program memasukkan data dengan larik

```
#include<stdio.h>
int main() {
    char nama[10];
    char nim[10];
    int nilai[10];
    int i,n;
/* menentukan banyaknya data yang akan diinputkan */
    printf("Banyak data: ");
    scanf("%d", &n);
    printf("\n");
/* memasukkan data sesuai dengan banyaknya data yang ditentukan */
    for(i = 1; i<=n;i++) {
        printf("Data ke-%d \n",i);
        printf("Nama: "); scanf("%s",&nama[i]);
        printf("NIM: "); scanf("%s",&nim[i]);
        printf("Nilai: "); scanf("%d",&nilai[i]);
        printf("\n");
    }
/* menampilkan data sesuai dengan yang diinput */
    printf("-----\n");
    for(i = 1; i<=n;i++) {
        printf("Data ke-%d \n",i);
        printf("Nama: %s\n",nama[i]);
        printf("NIM: %s\n",nim[i])
        printf("Nilai: %d\n",nilai[i])
        printf("\n");
    }
}
```

Contoh 2: Larik Dimensi 1 – Menghitung nilai rata-rata menggunakan larik 1 dimensi

```
#include<stdio.h>
#include<conio.h>
int main() {
    int nim[5];
    float uts[5];
    float uas[5];
    float kuis[5];
    float tugas[5];
    float rata[5];
    int i, banyak;

    printf("Masukkan jumlah mahasiswa: "); scanf("%d",&banyak);

    for(i=0;i<banyak;i++) {
        printf("\nData mahasiswa ke-%d",i+1);
        printf("\nMasukkan NIM mahasiswa: "); scanf("%d", &nim[i]);
        printf("Masukkan nilai kuis: "); scanf("%f", &kuis[i]);
        printf("Masukkan nilai tugas: "); scanf("%f", &tugas[i]);
        printf("Masukkan nilai UTS: "); scanf("%f", &uts[i]);
        printf("Masukkan nilai UAS: "); scanf("%f", &uas[i]);

        rata[i]=(kuis[i]+tugas[i]+uts[i]+uas[i])/4;
    }
    printf("\n-----");
    for(i=0;i<banyak;i++) {
        printf("\nNIM: ", nim[i]);
        printf("\nNilai kuis: ", kuis[i]);
        printf("\nNilai tugas: ", tugas[i]);
        printf("\nNilai UTS: ", uts[i]);
        printf("\nNilai UAS: ", uas[i]);
        printf("\nNilai rata: ", rata[i]);
        printf("\n");
    }
    return 0;
}
```

### Contoh 3: Larik Berdimensi Dua – Program Membuat Matrik

```
#include<stdio.h>
int main() {
    int a[100][100];
    int m,n,i,j;

/* menentukan banyaknya baris & kolom matriks */
    printf("Matriks berordo m x n \n");
    printf("-----\n\n");
    printf("Masukkan banyaknya baris (m): "); scanf("%d",&m);
    printf("Masukkan banyaknya kolom(n): "); scanf("%d",&n);
    printf("\n");

/* input elemen matriks */
    for(i = 0; i<m;i++) {
        for(j=0;j<n;j++) {
            printf("Elemen matrik A[%d %d]: ",i+1,j+1);
            scanf("%d",&a[i][j]);
        }
    }

/* menampilkan elemen matriks */
    printf("\n");
    printf("Matriks A = \n\n");
    for(i = 0; i<m;i++) {
        for(j=0;j<n;j++) {
            printf("%3d",a[i][j]);
        }
        printf("\n");
    }
}
```

### Tugas Unit 5:

1. Buatlah sebuah program yang dapat mencetak bilangan prima yang ada dalam bilangan 1 – 30 dengan menggunakan larik, di mana user memasukkan nilai dari 1 – 30. Buatlah analisis dari program tersebut.
2. Buatlah sebuah program untuk menjumlahkan dua buah matriks menggunakan larik dua dimensi dengan ketentuan:
  - Syarat dari penjumlahan matrik adalah jumlah ordo matrik A = jumlah ordo matrik B
  - User yang menginputkan ordo matrik
  - User yang menginputkan nilai dari tiap elemen matrik yang ada