

DASAR-DASAR PROGRAMAN

```
1  #include <iostream.h>
2  #include <conio.h>
3
4  void main()
5  {
6      int nilai['n'];
7      int temp, n, a, b;
8      cout << " ---PROGRAM BUBBLE SORT---\n";
9      cout << " Banyak data dalam array : ";
10     cin >> n;
11     cout << endl;
12     for (a=1; a<=n; a++) {
13         cout << " input nilai[" << a << "] : ";
14         cin >> nilai[a];
15     }
16     cout << "\n Data sebelum diurutkan\n";
17     for (a=1; a<=n; a++) {
18         cout << " " << nilai[a];
19     }
20     for (a=n-1; a>=1; a--) {
21         cout << "\n -> Looping ke-" << a;
22         for (b=1; b<=a; b++) {
23             if (nilai[b] > nilai[b + 1]) {
24                 temp = nilai[b + 1];
25                 nilai[b + 1] = nilai[b];
26                 nilai[b] = temp;
27             }
28         }
29     }
30 }
```

DASAR-DASAR PEMROGRAMAN

Penulis:

Shinta Esabella
Miftahul Haq

ISBN:

978-623-96935-0-3

Editor:

Fahmi Yuliono

Penyunting:

Miftahul Haq

Desain Sampul dan Tata letak:

Miftahul Haq

Penerbit:

Olat Maras Publising (OMP)

Redaksi

PT. OMP

*Jl. Raya Olat Maras, Dusun Batu Alang Kec. Moyo Hulu, Sumbawa Besar-
NTB Tel +62371 2629906, Fax +62371 2620676, Pos 84371
Email: lppm@uts.ac.id*

Cetakan Pertama, Mei 2021

iii + 139 hal. Ukuran 15,5cm x 23cm (Unesco)

Hak Cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun
tanpa ijin tertulis dari penerbit

KATA PENGANTAR

Segala puji bagi Allah Subhanahu Wata'ala yang telah melimpah kepada hamba-Nya nikmat Iman, Islam, juga nikmat kesempatan kepada kita semua. Sholawat serta salam semoga selalu tercurahkan kepada baginda Nabi Muhammad Shalallahu Alaihi Wasallam yang telah berjuang menyebarkan Islam sehingga kita dapat merasakan nikmat Islam. Penulis sangat bersyukur dan berterima kasih kepada berbagai pihak yang telah membantu dan mendukung, sehingga buku "*Dasar-dasar Pemrograman*" ini dapat terselesaikan.

Buku Dasar-dasar Pemrograman ini merupakan salah satu media belajar pendukung untuk memperkuat mata kuliah dasar-dasar pemrograman yang diajarkan di kelas secara teori dan praktik. Dengan adanya buku ini, diharapkan mahasiswa dapat dengan mudah mempelajari, memahami, dan mempraktikkan materi-materi yang telah diajarkan pada mata kuliah dasar-dasar pemrograman.

Semoga Allah SWT merahmati kita semua. Dan semoga buku ini dapat bermanfaat bagi semua kalangan pembaca.

Sumbawa, 10 Mei 2021

Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI	ii
BAGIAN 1 PENGENALAN DASAR-DASAR ALGORITMA DAN PEMROGRAMAN	1
BAGIAN 2 KONSEP DASAR ALGORITMA	5
BAGIAN 3 KONSEP DASAR <i>FLOWCHART</i>	7
BAGIAN 4 KONSEP DASAR <i>PSEUDO-CODE</i>	12
BAGIAN 5 TIPE DATA, EKSPRESI, <i>OPERATOR</i> , DAN <i>OPERAND</i>	15
PRAKTIKUM PERTAMA	21
BAGIAN 6 OPERASI SELEKSI	22
BAGIAN 7 OPERASI PERULANGAN	27
BAGIAN 8 PENGENALAN BAHASA C++	32
BAGIAN 9 PEMROGRAMAN MODULAR (PROSEDUR DAN FUNGSI)	42
BAGIAN 10 PEMROGRAMAN <i>ARRAY</i>	46
PRAKTIKUM KEDUA.....	50
EVALUASI PEMAHAMAN	52
BAGIAN 11 ALGORITMA <i>SEQUENTIAL</i> (LANJUTAN).....	55
BAGIAN 12 ALGORITMA <i>BRANCHING</i> (LANJUTAN)	61
BAGIAN 13 ALGORITMA <i>LOOPING</i> (LANJUTAN)	69
PRAKTIKUM KETIGA.....	74
BAGIAN 14 PEMROGRAMAN MODULAR (LANJUTAN) DAN REKURSIF	76
BAGIAN 15 PEMROGRAMAN <i>ARRAY</i> (LANJUTAN).....	87
BAGIAN 16 PEMROGRAMAN <i>STRUCT</i>	94
BAGIAN 17 PEMROGRAMAN <i>ARRAY OF STRUCT</i>	97
BAGIAN 18 PEMROGRAMAN <i>SEARCHING</i>	100
BAGIAN 19 PEMROGRAMAN <i>SORTING</i>	108

PRAKTIKUM KEEMPAT	121
BAGIAN 20 <i>POINTER</i> DALAM PEMROGRAMAN C++	124
BAGIAN 21 <i>FILE</i> (BERKAS) DALAM PEMROGRAMAN C++.....	128
EVALUASI PEMAHAMAN	131
DAFTAR PUSTAKA.....	135
TENTANG PENULIS	138

BAGIAN 1

PENGENALAN DASAR-DASAR ALGORITMA DAN PEMROGRAMAN

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu mengenal dan memahami tentang konsep dari dasar algoritma dan pemrograman.

Teori:

A. ALGORITMA

Definisi Algoritma adalah urutan langkah-langkah untuk memecahkan masalah. Algoritma dibutuhkan untuk memerintah komputer mengambil langkah-langkah tertentu dalam menyelesaikan masalah.

Dalam kehidupan sehari-hari, kita sudah melakukan penyusunan algoritma untuk menyelesaikan permasalahan atau tantangan yang dihadapi. Sebagai contoh, pada saat diminta untuk membuat telur dadar. Sebelum membuat algoritmanya, kita perlu mendefinisikan masukan (*input*) dan keluaran (*output*) terlebih dahulu, dimana input berupa telur mentah, dan *output* berupa telur dadar yang sudah matang. Susunan algoritmanya sebagai berikut:

1. Nyalakan api kompor
2. Tuangkan minyak ke dalam wajan
3. Pecahkan telur ayam ke dalam mangkok
4. Tambahkan garam secukupnya
5. Aduk campuran telur dan garam
6. Tuang adonan telur ke dalam wajan
7. Masak telur hingga matang

Algoritma akan lebih baik jika ditulis secara sistematis menggunakan beberapa skema, dalam buku ini akan dibahas mengenai skema *Flowchart* dan *Pseudocode*.

Contoh algoritma lainnya:

1. Algoritma menukar_isi_bejana
{Diberikan dua buah bejana A dan B. Bejana A berisi larutan berwarna merah, bejana B berisi larutan berwarna biru}
Logika pertama:
 - Tuangkan larutan dari bejana A ke dalam bejana B
 - Tuangkan larutan dari bejana B ke dalam bejana A

Logika kedua (bejana C sebagai perantara):

- Tuangkan larutan dari bejana A ke dalam bejana C
- Tuangkan larutan dari bejana B ke dalam bejana A
- Tuangkan larutan dari bejana C ke dalam bejana B

2. Algoritma membuat_minuman_kopi

- Masukkan satu sendok makan gula ke dalam cangkir.
- Masukkan satu sendok teh kopi kedalam cangkir.
- Tuangkan air panas ke dalam cangkir hingga penuh.
- Aduk isi cangkir selama 30 detik.

B. PROGRAM

Program adalah formulasi sebuah algoritma dalam bentuk bahasa pemrograman, sehingga siap untuk dijalankan pada mesin komputer. Membuat program seperti memberitahukan apa yang harus dilakukan kepada orang lain. Sebagai contoh, pada saat kita memberitahukan algoritma membuat telur dadar kepada orang lain, kita sudah melakukan pemrograman.

Pemrograman membuat telur dadar kepada orang lain akan lebih mudah karena orang tersebut sudah mengetahui apa itu telur dadar. Pada langkah yang ke-3 diminta untuk memecahkan telur ke dalam mangkok, bagaimana cara orang tersebut memecahkan telur tentunya sudah diketahui dan kita tidak perlu menjelaskan terlalu detail.

Lain halnya jika kita harus menyuruh komputer untuk melakukan apa yang kita inginkan. Komputer sebenarnya hanyalah sebuah mesin yang tidak memiliki emosi dan kemampuan bersosialisasi. Oleh karena itu, untuk membuatnya menjadi mudah, diperlukan penyusunan algoritma yang benar.

Mendesain algoritma yang benar dan menterjemahkannya ke dalam bahasa pemrograman bukanlah hal yang mudah karena bahasa pemrograman memiliki tata penulisan tersendiri.

C. BAHASA PEMROGRAMAN

Bahasa pemrograman adalah bahasa buatan yang digunakan untuk mengendalikan perilaku dari sebuah mesin, biasanya berupa mesin komputer, sehingga dapat digunakan untuk memberitahu komputer tentang apa yang harus dilakukan.

Struktur bahasa ini memiliki kemiripan dengan bahasa natural manusia, karena juga tersusun dari elemen-elemen dasar seperti: kata benda dan kata kerja serta mengikuti aturan untuk menyusunnya menjadi kalimat.

D. TINGKATAN BAHASA PEMROGRAMAN

Bahasa pemrograman adalah bahasa buatan yang digunakan untuk mengendalikan perilaku dari sebuah mesin, biasanya berupa mesin komputer, sehingga dapat digunakan untuk memberitahu komputer tentang apa yang harus dilakukan.

1. Bahasa Pemrograman Tingkat Tinggi.
Merupakan bahasa tingkat tinggi yang mempunyai ciri-ciri mudah dimengerti karena kedekatannya terhadap bahasa sehari-hari. Sebuah pernyataan program diterjemahkan kepada sebuah atau beberapa mesin dengan menggunakan *compiler*. Sebagai contoh adalah: JAVA, C++, .NET.
2. Bahasa Pemrograman Tingkat Menengah.
Dimana penggunaan instruksi telah mendekati bahasa sehari-hari, walaupun masih cukup sulit untuk dimengerti karena menggunakan singkatan-singkatan seperti STO yang berarti simpan (STORE) dan MOV yang artinya pindah (MOVE). Yang tergolong dalam bahasa ini adalah Fortran.
3. Bahasa Pemrograman Tingkat Rendah.
Bahasa pemrograman generasi pertama. Bahasa jenis ini sangat sulit dimengerti karena instruksinya menggunakan bahasa mesin. Disebut juga dengan bahasa *assembly* merupakan bahasa dengan pemetaan satu-persatu terhadap instruksi komputer. Setiap instruksi *assembly* diterjemahkan dengan menggunakan *assembler*.

E. ASPEK PENTING DARI ALGORITMA

1. *Finiteness*
Algoritma harus berhenti *after a finite number of steps*.
2. *Definiteness*
Setiap langkah harus didefinisikan secara tepat, tidak boleh membingungkan (ambigu).
3. *Input*
Sebuah algoritma memiliki nol atau lebih *input* yang diberikan kepada algoritma sebelum dijalankan.
4. *Output*
Sebuah algoritma memiliki nol atau lebih *output*, yang biasanya bergantung pada *input*.

5. *Effectiveness*

Setiap algoritma diharapkan memiliki sifat sederhana.

Soal Latihan:

Jawablah pertanyaan berikut dengan benar:

1. Jelaskan yang anda pahami tentang Algoritma!
2. Bedakan apa itu algoritma, program, dan bahasa pemrograman!
3. Sebutkan dan jelaskan tentang tingkatan bahasa pemrograman!
4. Buatlah algoritma untuk memilih bilangan terbesar dari 3 buah bilangan acak!

BAGIAN 2

KONSEP DASAR ALGORITMA

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami konsep dasar algoritma
2. Memahami penulisan algoritma dalam bahasa natural, *flowchart*, dan *pseudocode*

Teori:

Berikut merupakan macam-macam penulisan algoritma:

A. BAHASA NATURAL

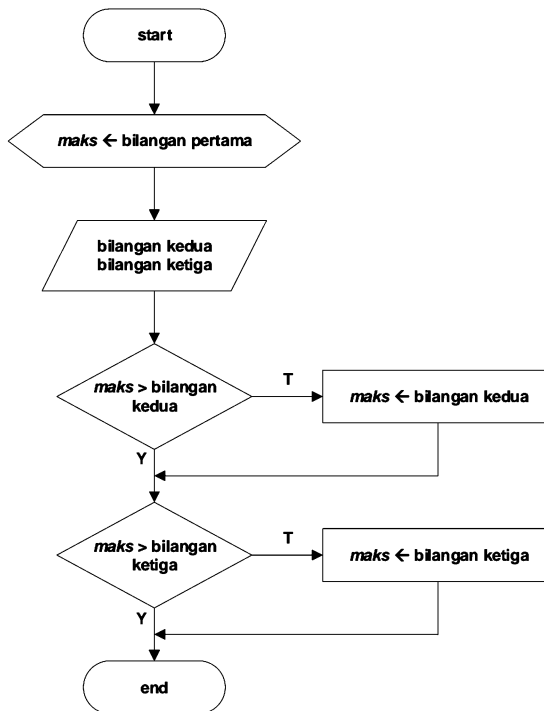
Seperti halnya bahasa Indonesia, bahasa Inggris, dsb. Tapi dalam penulisan bahasa natural sering membingungkan (ambigu).

1. Ambil bilangan pertama dan set *maks* sama dengan bilangan pertama
2. Ambil bilangan kedua dan bandingkan dengan *maks*
3. Apabila bilangan kedua lebih besar dari *maks*, set *maks* sama dengan bilangan kedua
4. Ambil bilangan ketiga dan bandingkan dengan *maks*
5. Apabila bilangan ketiga lebih besar dari *maks*, set *maks* sama dengan bilangan ketiga
6. Variabel *maks* berisi bilangan terbesar. Tampilkan hasilnya.

B. FLOWCHART

Algoritma *flowchart* adalah suatu bagan atau diagram dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program.

Penulisan menggunakan *flowchart* adalah model penulisan dengan menggunakan bentuk penyusunan bangun ruang *flowchart* atau bisa disebut juga dengan bagan alir. *Flowchart* merupakan bagian yang menunjukkan aliran atau runtutan algoritma. Penulisan ini bagus secara visual, akan tetapi bisa menjadi rumit jika algoritma yang dibuat terlalu panjang (tidak efisien).



Gambar. Contoh Penulisan dengan *Flowchart*

C. PSEUDOCODE

Pseudocode merupakan penulisan bahasa yang sudah lebih dekat ke bahasa pemrograman, namun sulit dimengerti oleh orang yang tidak mengerti pemrograman. Contoh penulisan menggunakan *pseudocode*:

1. *Maks* ← bilangan pertama
2. If (*Maks* < bilangan kedua)
3. *Maks* ← bilangan kedua
4. If (*Maks* < bilangan ketiga)
5. *Maks* ← bilangan ketiga

Soal Latihan:

1. Jelaskan perbedaan penulisan algoritma dalam bahasa natural, *flowchart*, dan *pseudocode*!
2. Buatlah sebuah algoritma sederhana menghasilkan *output* dengan tema “Menghitung Penjumlahan” dengan *input* bilangan1 dan bilangan2! (Gunakan *flowchart* dan *pseudocode*)

BAGIAN 3

KONSEP DASAR *FLOWCHART*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami algoritma dengan *flowchart* yang umumnya digunakan.
2. Menyelesaikan kasus dan latihan yang umum digunakan serta menterjemahkannya kedalam bentuk *flowchart*.

Teori:

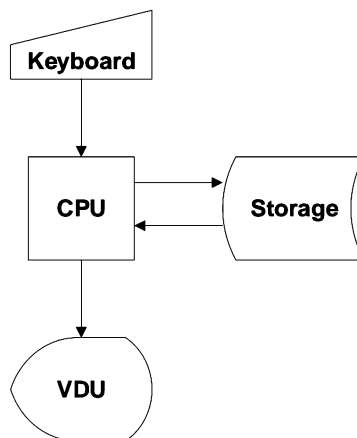
Dalam membuat algoritma, diperlukan suatu mekanisme atau alat bantu untuk menuangkan hasil pemikiran mengenai langkah-langkah penyelesaian masalah yang sistematis dan teratur. Pada dasarnya untuk bisa menyusun solusi diperlukan kemampuan *problem-solving* yang baik. Oleh karena itu, sebagai sarana untuk melatih kemampuan tersebut terdapat sebuah *tool* (alat) yang dapat digunakan, yakni *flowchart*.

Secara formal, *flowchart* didefinisikan sebagai skema penggambaran dari algoritma atau proses. Adapun tujuan dari *flowchart* adalah menggambarkan suatu tahapan penyelesaian masalah.

A. JENIS-JENIS *FLOWCHART*

1. *System Flowchart*

Menggambaran suatu sistem perangkat komputer yang digunakan dalam proses pengolahan data serta hubungan antar perangkat tersebut.



Gambar. Contoh Penggunaan *System Flowchart*

2. Program Flowchart

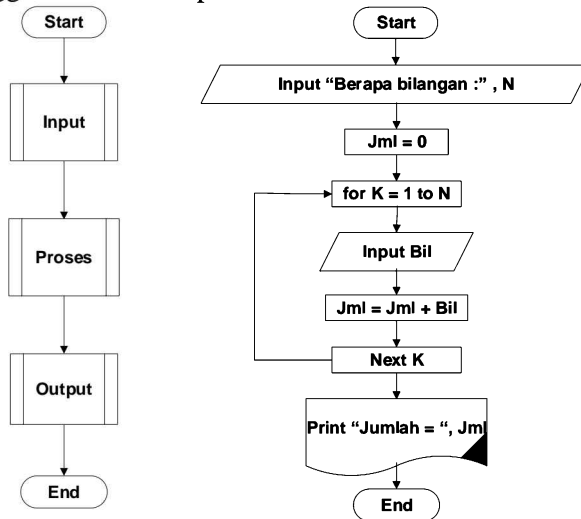
Menggambarkan urutan logika dari suatu prosedur pemecahan masalah. Terdapat dua jenis metode penggambaran *Program Flowchart* yaitu:

- *Conceptual Flowchart*

Menggambarkan alur pemecahan masalah secara global.

- *Detail Flowchart*

Menggambarkan alur pemecahan masalah secara rinci.





Gambar. Contoh program *flowchart* (1) *Conceptual flowchart*, dan (2) *Detail flowchart*

B. SIMBOL-SIMBOL *FLOWCHART*

1. Flow Direction Symbols

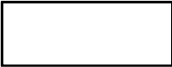

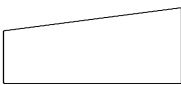
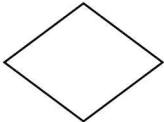

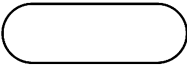
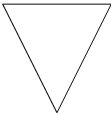

Digunakan untuk menghubungkan simbol satu dengan yang lain, disebut juga *connecting line*.

	Simbol arus / flow Untuk menyatakan jalannya arus suatu proses.
	Simbol Communication Link Untuk menyatakan transmisi data dari satu lokasi ke lokasi lain.

	Simbol Connector Untuk menyatakan sambungan dari proses ke proses lainnya dalam halaman yang sama.
	Simbol Offline Connector Untuk menyatakan sambungan dari proses ke proses lainnya dalam halaman yang berbeda.



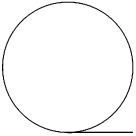

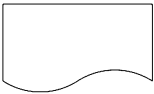
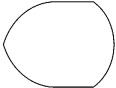
2. *Processing Symbol*

Menunjukkan jenis operasi pengolahan dalam suatu proses/prosedur.

	Simbol Process Untuk menyatakan suatu tindakan (proses) yang dilakukan oleh komputer
	Simbol Manual Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer
	Simbol Manual Input Untuk memasukkan data secara manual dengan menggunakan online keyboard
	Simbol Decision Untuk menunjukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban: YA / TIDAK
	Simbol Predefined Process Menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal
	Simbol Terminal Menyatakan permulaan atau akhir suatu program
	Simbol Offline-strong Untuk menunjukan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu
	Simbol Procedure (GoSub) Menyatakan sekumpulan langkah (proses) yang dituliskan sebagian suatu prosedur (Subran/Subprogram)

3. *Input/output Symbols.*

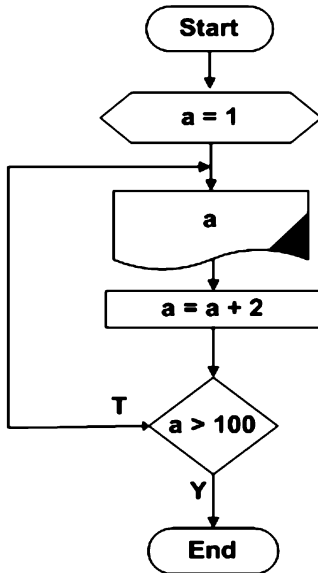
Menunjukkan jenis peralatan yang digunakan sebagai media *input* atau *output*.

	Simbol <i>Input/output</i> Untuk menyatakan proses input atau <i>output</i> tanpa tergantung jenis peralatannya
	Simbol <i>Punched Card</i> Untuk menunjukan input berasal dari kartu atau <i>output</i> ditulis ke kartu
	Simbol <i>Magnetic tape</i> Untuk menyatakan input berasal dari pita magnetis atau <i>output</i> disimpan ke pita magnetis dengan menggunakan online keyboard
	Simbol <i>Disk Storage</i> Menyatakan input berasal dari disk atau <i>output</i> disimpan ke disk
	Simbol <i>Document</i> Untuk mencetak keluaran dalam bentuk dokumen (melalui printer)
	Simbol <i>Display</i> Untuk mencetak keluaran dalam layar monitor

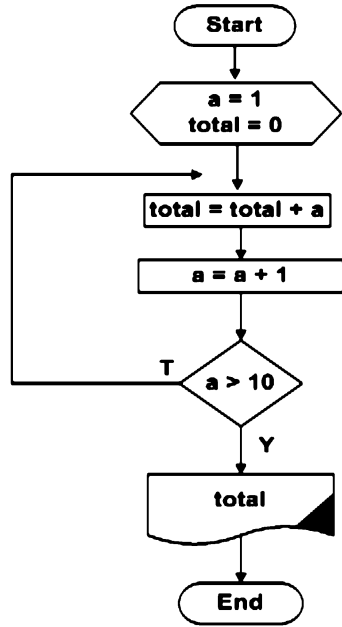
Soal Latihan:

Tulislah *output* (keluaran) dari *flowchart* berikut ini?

1.



2.



4. Buatlah *Flowchart* untuk menampilkan bilangan **0 2 4 6 8 !**
5. Buatlah *Flowchart* untuk menampilkan **nama_anda** Sebanyak **10 kali!**
6. Buatlah *Flowchart* untuk menampilkan **usia_anda**, dimana tahun lahir merupakan variabel yang harus diinput!

Latihan dikerjakan dan silakan lihat video berikut

<https://www.youtube.com/watch?v=u1huXRBkSbE>

BAGIAN 4

KONSEP DASAR *PSEUDO-CODE*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami algoritma dengan *flowchart* dan *pseudo-code*.
2. Mampu menuliskan algoritma pemrograman dalam bentuk *pseudo-code*
3. Mampu menerjemahkan *flowchart* ke dalam bentuk *pseudo-code*.

Teori:

Skema lain yang dapat digunakan untuk menyusun algoritma adalah *pseudo-code*. *Pseudo-code* adalah bentuk informal untuk mendeskripsikan algoritma yang mengikuti struktur bahasa pemrograman tertentu. Tujuan dari penggunaan *pseudo-code* adalah supaya:

1. Lebih mudah dibaca oleh manusia.
2. Lebih mudah untuk dipahami.
3. Lebih mudah dalam menuangkan ide/hasil pemikiran.

Pseudocode sering digunakan dalam buku-buku tentang ilmu komputer ataupun publikasi ilmiah untuk menjelaskan urutan proses atau metode tertentu. Seorang programmer yang ingin menerapkan algoritma tertentu, terutama yang kompleks atau algoritma baru, biasanya akan memulainya dengan membuat deskripsi dalam bentuk *pseudo-code*. Setelah *pseudo-code* tersebut jadi, maka langkah selanjutnya hanya tinggal menterjemahkannya ke bahasa pemrograman tertentu. *Pseudo-code* ini biasanya disusun dalam bentuk yang terstruktur dengan pendekatan sekuensial (berurutan) dari atas ke bawah.

Silakan lihat video berikut

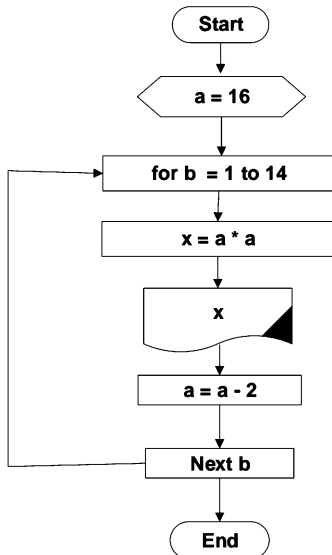
<https://www.youtube.com/watch?v=zqhQMGjNCmw>

Penulisan *Pseudocode*:

- **Judul:** Menjelaskan judul dari Algoritma yang dibuat
- **Deklarasi:** Menjelaskan variabel apa saja yang digunakan dan apa tipe datanya (mengarah ke bahasa pemrograman yang digunakan)
- **Deskripsi:** Menjelaskan setiap langkah langkah penyelesaian masalah tersebut

Contoh penulisan algoritma menggunakan *flowchart* ke *pseudocode*:

1.



Judul:

{Menampilkan bilangan kuadrat sebanyak 14 suku yang dimulai dari bilangan 16, dimana bilangan dikurangi 2}

Deklarasi:

integer a, b, x ;

Deskripsi:

1. start

2. $a \leftarrow 16$

3. FOR b = 1 TO 14

$x \leftarrow a * a$

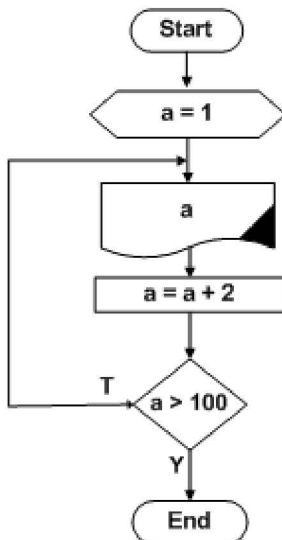
cetak "x"

$a \leftarrow a - 2$

NEXT b

4. end

2.



Judul:

{Mencetak bilangan ganjil yang kurang dari 100}

Deklarasi:

integer a;

Deskripsi:

1. start

2. $a \leftarrow 1$

3. cetak "a"

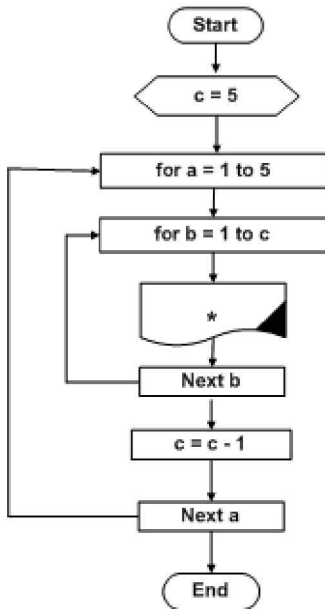
4. $a \leftarrow a + 2$

5. jika $a > 100$ maka ke no 5

jika tidak maka ke no 2

6. end

3.



Judul:

{ Mencetak bintang }

Deklarasi:

integer a, b, c;

Deskripsi:

1. start

2. $c \leftarrow 5$

3. FOR a = 1 to 5

FOR b = 1 to c

cetak " * "

NEXT b

c = c - 1

NEXT a

4. end

Soal Latihan:

Buatlah algoritma dalam bentuk *flowchart* dan *pseudocode* untuk menyelesaikan kasus berikut:

1. Menampilkan *output* bilangan bulat berikut menggunakan perulangan *for-next*: **4 5 6 7 8**
2. Menampilkan *output* **luas lingkaran**, dimana input panjang dan lebar dari piranti luar (*keyboard*)!
3. Menampilkan *output* total bayar dari soal kasus berikut:

Pak Ahmad membuka usaha untuk pemasangan kabel *coaxial*. Untuk setiap instalasi pada 1 lokasi Pak Ahmad memasang biaya tarif berupa "Biaya Pelayanan Dasar" sebesar Rp.70.000,- dan "Per-meter kabel biayanya" sebesar Rp.5.000,-. Tampilkan total yang harus dibayar oleh Pelanggan! (*sebagai input, jumlah lokasi dan panjang kabel yang digunakan*)

BAGIAN 5

TIPE DATA, EKSRESI, *OPERATOR*, DAN *OPERAND*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami tipe data, ekspresi, *operator*, dan *operand*
2. Mampu menggunakan tipe data yang sesuai dengan kebutuhan program
3. Mampu menggunakan ekspresi, *operator*, dan *operand* dalam pembuatan program

Teori:

Program dibuat untuk mengolah data menjadi informasi. Data manipulasi → disimpan ke dalam memori komputer. Disimpan dalam bentuk variabel. Variabel atau konstanta harus mempunyai nama tertentu dan tipe data tertentu. Tipe data juga sesuatu yang menyatakan pola penyajian data dalam memori komputer.

Tipe data adalah himpunan nilai yang dapat dimiliki oleh sebuah data. Tipe data menentukan apakah sebuah nilai dapat dimiliki sebuah data atau tidak, serta operasi apa yang dapat dilakukan pada data tersebut. Dalam peubah (variabel) yang akan digunakan dalam program harus ditentukan (dideklarasikan) tipe datanya.

Variabel adalah suatu lokasi di memori yang diberi nama khas untuk menampung suatu data atau mengambil kembali nilai tersebut. Dalam deklarasinya dibuat bersamaan dengan tipe data. Tipe variabel ditentukan oleh jenis data yang akan disimpan.

Bentuk umum:

```
tipe_data nama_variabel1;  
tipe_data nama_variabel2 = value;
```

Menentukan tipe data variabel artinya juga harus menentukan batasan nilai variabel tersebut dan jenis operasi yang bisa dikenakan padanya. Karena masing-masing tipe data memiliki batasan nilai dan jenis operasi yang berbeda-beda. Tipe data dapat dikelompokkan menjadi 2 macam, yaitu: 1) tipe data dasar, dan 2) tipe data bentukan.

A. TIPE DATA DASAR

<i>char</i> (2 bytes)	<i>extended character set (ISO Unicode standard)</i> seperti: huruf A..Z,a..z, 0..9, *, !.
<i>byte</i> (1 byte)	-128 to +127
<i>short</i> (2 bytes)	-32,768 to + 32,767
<i>int</i> (4 bytes)	-2,147,483,648 to + 2,147,483,647
<i>long</i> (8 bytes)	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
<i>float</i> (4 bytes)	-3.40292347E+38 to + 3.40292347E+38 (<i>IEEE standard</i>)
<i>double</i> (8 bytes)	-1.79769313486231570E+308 to ... (<i>IEEE standard</i>)
<i>boolean</i> (1 byte)	<i>true</i> atau <i>false</i>

B. TIPE DATA BENTUKAN

Tipe data bentukan merupakan sekumpulan variabel-variabel dengan tipe berbeda dan bentuk berbeda pula yang dapat dibentuk sendiri sesuai kebutuhan untuk program yang akan dibuat. Tipe yang dibentuk oleh programmer itu sendiri. Tipe data bentukan merupakan kumpulan dari tipe data lainnya dimana struktur terdiri dari data yang disebut field. Field-field tersebut digabungkan menjadi satu tujuan untuk kemudahan dalam operasi. Tipe data bentukan di luar dari tipe data dasar. Bentuk umum tipe data bentukan:

```
typedef struct {  
    tipe_data variabel1= value;  
    tipe_data variabel2= value;  
} nama_struktur;
```

C. KONSTANTA

Konstanta adalah sebuah *operand* dengan nilai yang tetap dan pasti. Nilai dari konstanta tidak dapat berubah (tetap) setelah didefinisikan, ketika program dijalankan. Pemberian nilai sebuah konstanta dilakukan di awal program. Isi sebuah konstanta tidak dapat diubah selama program berjalan. Konstanta mirip dengan variabel, namun memiliki nilai tetap dengan deklarasi nama variabel konstanta disertai dengan nilainya. Konstanta dapat berupa nilai Integer, Float, Karakter dan String.

D. OPERATOR

Dalam bahasa pemrograman, terdapat istilah *operand* dan *operator*. *Operand* adalah nilai asal yang digunakan didalam proses operasi, sedangkan *operator* adalah instruksi yang diberikan untuk mendapatkan hasil dari proses tersebut. Berikut adalah beberapa istilah yang harus ketahui dalam operator:

$x = 2 + 8$

Maka:

x	disebut dengan variabel
$=$	disebut dengan <i>operator assignment</i>
2 dan 8	disebut dengan <i>operand</i>
$2 + 8$	disebut dengan ekspresi
$+$	disebut dengan operator aritmatika (penjumlahan)
$X = 2 + 8$	disebut dengan statemen aritmatika

Operator adalah simbol atau karakter khusus yang menghasilkan suatu nilai. *Operator* meliputi:

1. Operator Aritmatika

Operator ini digunakan utk perhitungan aritmatika. Yang termasuk operator ini:

$*$	(Perkalian)
$/$	(Pembagian)
$\%$	(sisanya pembagian/modulus)
$+$	(Pertambahan)
$-$	(Pengurangan)

2. Operator Penambahan & Pengurangan

Operator penambahan untuk menaikkan satu nilai ($++$)

Operator pengurangan untuk menurunkan satu nilai ($--$)

Contoh:

```
y = x++;  
nilai y = x, x=x+1  
y = ++x;  
nilai y=x+1, x=x+1
```

Hasil dari x dan y berikut:

x semula	Pernyataan	Hasil y	Hasil x
6	$y = x++$	6	7

6	$y = ++x$	7	7
6	$y = x--$	6	5
6	$y = --x$	5	5

3. Operator Bit

& (dan)

| (atau untuk biner)

^ (atau eksklusif)

~ (bukan untuk biner/kebalikan *operand*)

<< (geser kiri) pergeseran ini identik dengan pengalihan pada bilangan 2. contoh : $x \ll n$ maka $x * 2^n$

>> (geser kanan) pergeseran ini identik dengan pembagian pada bilangan 2. contoh : $x \gg n$ maka $x / 2^n$

Operator &

Bit 1	Bit 2	Hasil
0	0	0
0	1	0
1	0	0
1	1	1

Operator |

Bit 1	Bit 2	Hasil
0	0	0
0	1	1
1	0	1
1	1	1

Operator ^

Bit 1	Bit 2	Hasil
0	0	0
0	1	1
1	0	1
1	1	0

4. Operator Penugasan

Operator ini digunakan untuk memberikan nilai ke variabel

Operator	Keterangan	Contoh
=	Pemberian nilai	$x = 2$
+=	Penambahan bilangan	$x += 2$ sama dengan $x = x + 2$
-=	Pengurangan bilangan	$x -= 2$ sama dengan $x = x - 2$
*=	Pengalian bilangan	$x *= 2$ sama dengan $x = x * 2$
/=	Pembagian bilangan	$x /= 2$ sama dengan $x = x / 2$
% =	Pemerolehan sisa bagi	$x \% = 2$ sama dengan $x = x \% 2$

5. Operator Pembandingan

Operator yang digunakan untuk membandingkan dua nilai

- > Lebih besar
- >= Lebih besar sama dengan
- < Kurang dari
- <= Kurang dari sama dengan
- = Sama dengan
- != Tidak sama dengan

Contoh:

$5 > 6$ hasilnya salah (0)

E. EKSPRESI

Ekspresi adalah suatu bentuk yang menghasilkan suatu nilai. Ekspresi dapat berupa variabel atau melibatkan *operator* dan *operand*. Contoh:

```
int sum = 5;
int finalsum;
finalsum = sum;
int a = 1+2;
```

Macam-macam ekspresi:

1. Ekspresi Aritmatika

$(A * B, x \leftarrow (k * I) \bmod 2)$

- *Operand*: numerik
- Hasilnya: numerik

2. Ekspresi Relasi

$(<, >, < >, >=, <=, \text{NOT}, \text{AND}, \text{OR})$
--

- *Operand*: numerik, string
- Hasilnya: Boolean

3. Ekspresi String

- Ekspresi string dengan operator “+” berarti penyambungan string
- $A \leftarrow \text{“Universitas Teknologi Sumbawa”}$
- $B \leftarrow \text{“(UTS)”}$
- $A + B = \text{“Universitas Teknologi Sumbawa (UTS)”}$

Soal Latihan:

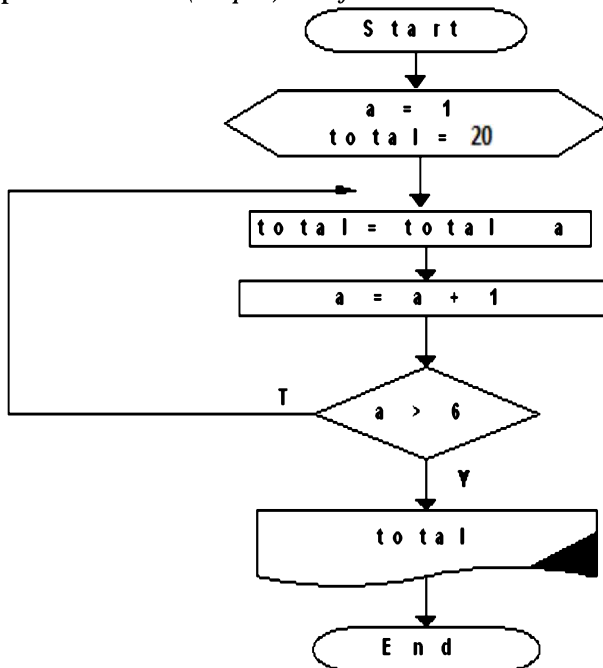
Jawablah soal dibawah ini:

1. Apa yang anda ketahui tentang definisi tipe data, ekspresi, *operator* dan *operand*?
2. Jelaskan jenis-jenis tipe data, ekspresi, *operator* dan *operand*!
3. Berikan contoh algoritma dengan *pseudocode* untuk penulisan teks tipe data, ekspresi, *operator* dan *operand*! (boleh satu contoh atau lebih)

PRAKTIKUM PERTAMA

Soal Praktikum:

1. Uraikan yang anda ketahui mengapa pentingnya mempelajari Dasar-Dasar Pemrograman dalam Program Studi Teknik Informatika!
2. Jelaskan perbedaan *System Flowchart* dengan *Program Flowchart*? berikan contoh!
3. Apakah keluaran (*output*) dari *flowchart* di bawah ini:



4. Buatlah *pseudocode* untuk menampilkan bilangan ganjil dari 1-100!
5. Jelaskan tentang tipe data, ekspresi, *operator* dan *operand*!

BAGIAN 6

OPERASI SELEKSI

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami algoritma dalam operasi seleksi
2. Mampu membedakan dan menggunakan operasi seleksi *if* dan *case* pada kondisi yang sesuai.
3. Mampu mengilustrasikan operasi seleksi dalam bentuk *flowchart*.

Teori:

Operasi seleksi adalah suatu struktur dasar algoritma yang memiliki satu atau lebih kondisi tertentu dimana sebuah instruksi dilaksanakan jika sebuah kondisi/persyaratan terpenuhi. Ada beberapa bentuk struktur dasar percabangan adalah 1) *if* dan 2) *switch case*.

A. IF

Sebuah pernyataan yang dapat dipakai untuk mengambil keputusan berdasarkan suatu kondisi. Bentuk pernyataan *if* ada tiga macam, yaitu:

1. If

Pernyataan dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak program tidak memberikan hasil apa-apa.

Bentuk Umum:

```
if (kondisi)
    Pernyataan;
```

Contoh:

```
if (nilai >= 60) then
    cout<< "Anda lulus mata kuliah dasar-dasar
    pemrograman");
```

2. If Else

Pernyataan1 dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi. Jika tidak, lakukan pernyataan2.

Jika Anda tidak mempergunakan pernyataan *else* program tidak akan *error*, namun jika anda mempergunakan pernyataan *else* tanpa didahului pernyataan *if*, maka program akan *error*.

Bentuk umum:

```
if (kondisi)
    Pernyataan 1;
else
    Pernyataan 2;
```

Contoh:

```
if (nilai >= 60) then
    cout<< "Anda Lulus Mata Kuliah Dasar-Dasar
Pemrograman");
else
    cout<< "Anda Tidak Lulus Mata Kuliah Dasar-Dasar
Pemrograman");
```

3. *If Bersarang*

Pernyataan1 dilaksanakan jika dan hanya jika kondisi1 yang diinginkan terpenuhi. Jika tidak, pernyataan2 dilaksanakan jika dan hanya jika kondisi2 yang diinginkan terpenuhi. Jika tidak, pernyataan3 dilaksanakan jika dan hanya jika kondisi3 yang diinginkan terpenuhi. Jika tidak, pernyataan4 dilaksanakan.

Bentuk umum:

```
if (kondisi1)
    Pernyataan 1;
else
    if (kondisi2)
        Pernyataan 2;
    else
        if (kondisi3)
            Pernyataan 3;
        else
            Pernyataan 4;
```

Contoh:

```
if (nilai >= 80) then
    cout<< "Anda mendapatkan Nilai A";
else
    if (nilai >= 70) then
        cout<< "Anda mendapatkan Nilai B";
```

```

else
    if (nilai >= 60) then
        cout<< "Anda mendapatkan Nilai C";
    else
        cout<< "Anda mendapatkan Nilai D";

```

B. SWITCH CASE

Pernyataan *switch* adalah pernyataan yang digunakan untuk menjalankan salah satu pernyataan dari beberapa kemungkinan pernyataan, berdasarkan nilai dari sebuah ungkapan dan nilai penyeleksian.

Bentuk pernyataan ini ada tiga macam:

1. Case Satu Pernyataan

Bentuk umum:

```

switch (ekspresi)
    case konstanta 1:
        pernyataan 1;
        break;

```

Contoh satu pernyataan:

```

int nilai;
switch (nilai)
    case 60:
        cout<< "Anda Lulus Mata Kuliah Dasar-Dasar
        Pemrograman";
        break;

```

2. Case Dua Pernyataan

Bentuk umum:

```

switch (ekspresi)
    case konstanta1:
        pernyataan1;
        break; //akhir dari suatu case

    default:
        pernyataan2;

```

Contoh dua pernyataan:

```

int nilai;

```

```

switch (nilai)
    case 60:
        cout<< "Anda Lulus Mata Kuliah Dasar-Dasar
Pemrograman";
        break;

    default:
        cout<< "Anda Tidak Lulus Mata Kuliah Dasar-Dasar
Pemrograman";

```

3. Case Tiga Pernyataan

Bentuk umum:

```

switch (ekspresi)
    case konstanta1:
        pernyataan1;
        break; //akhir dari case konstanta1

    case konstanta2:
        pernyataan2;
        break; //akhir dari case konstanta2

    case konstanta3:
        pernyataan3;
        break; //akhir dari case konstanta3

    default:
        pernyataan4;

```

Contoh tiga pernyataan:

```

int nilai;
switch (nilai)
    case 80:
        cout<< "Anda Dapat Nilai A";
        break;

    case 70:
        cout<< "Anda Dapat Nilai b";
        break;

    case 60:

```

```
cout<< "Anda Dapat Nilai C";  
break;  
  
default:  
cout<< "Anda Dapat Nilai D";
```

Soal Latihan:

1. Buatlah *flowchart* yang menggunakan percabangan satu dan dua kondisi dengan tema bebas
2. Buatlah *flowchart* dalam bentuk menu (menggunakan percabangan *switch*) yang mampu menghitung:
 - Luas dan keliling trapesium
 - Luas dan keliling persegi panjang
 - Luas dan keliling lingkaran
3. Buatlah *flowchart* untuk menyeleksi suatu bilangan dengan ketentuan sebagai berikut:
0 <= nilai < 30 : Nilai rendah
30 <= nilai < 60 : Nilai sedang
60 <= nilai <=100 : Nilai tinggi

BAGIAN 7

OPERASI PERULANGAN

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami algoritma dalam operasi perulangan
2. Mengimplementasikan operasi perulangan dalam algoritma bahasa C++.

Teori:

Operasi perulangan adalah sebuah/sekelompok instruksi yang diulang untuk jumlah pengulangan tertentu. Baik yang terdefinisi sebelumnya ataupun tidak. Struktur pengulangan terdiri atas dua bagian:

- Kondisi pengulangan yaitu ekspresi *boolean* yang harus dipenuhi untuk melaksanakan pengulangan.
- Isi atau badan pengulangan yaitu satu atau lebih pernyataan (aksi) yang akan diulang.

Terdapat 3 perintah atau notasi dalam struktur pengulangan adalah *for*, *while*, dan *do while*.

A. PERULANGAN *FOR*

Struktur perulangan *for* biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan *for* tampaknya lebih efisien karena susunannya lebih simpel dan sederhana.

Bentuk umumnya:

```
//untuk pengulangan yang melakukan proses increment
for (variabel=nilai_awal; kondisi; variabel_peningkatan)
{
    Pernyataan_yang_akan_diulang;
}

//untuk pengulangan yang melakukan proses decrement
for (nama_variabel=nilai_awal; kondisi; variabel_penurunan)
{
    Pernyataan_yang_akan_diulang;
}
```

Contoh perulangan *for*:

```
//perulangan yang menggunakan proses ascending
int a;
for (a = 1; a <= 10; a++)
{
    cout << a << "\n";
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

```
//perulangan yang menggunakan proses descending
int a;
for (a = 10; a >= 1; a--)
{
    cout << a << "\n";
}
```

Output:

```
10
9
8
7
6
5
4
3
2
1
```

Nested For, adalah suatu perulangan *for* didalam perulangan *for* lainnya. Didalam penggunaan *nested for*, perulangan yang didalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang diluar diselesaikan.

Bentuk umumnya:

```
for (variabel=nilai_awal; kondisi; variabel_perubahan)
{ //for utama
    for (variabel=nilai_awal; kondisi; variabel_perubahan)
    { //nested for
        Pernyataan/perintah_yang_akan_diulang;
    }
}
```

Contoh perulangan *nested for*:

```
//menggunakan ascending
int a, b;
for (a = 1; a <= 5; a++)
{
```

```
//menggunakan descending
int a, b;
for (a = 5; a >= 1; a--)
{
```

```

    for(b = 1; b<= 3; b++)
    {
        cout<< a<< " ";
    }
    cout << "\n";
}

```

output:

```

1 1 1
2 2 2
3 3 3
4 4 4
5 5 5

```

```

    for (b = 3; b>= 1; b--)
    {
        cout<< a<< " ";
    }
    cout << "\n";
}

```

output:

```

5 5 5
4 4 4
3 3 3
2 2 2
1 1 1

```

B. PERULANGAN *WHILE*

Perulangan *while* banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (*true*) dan akan berhenti bila kondisinya bernilai salah (*false*).

Bentuk umumnya:

```

//struktur pengulangan while
nama_variabel=nilai_awal;
while (kondisi)
{
    pernyataan_yang_akan_diulang;
    variabel_peningkatan;
}

```

Contoh perulangan *while*:

```

//menggunakan ascending
int e=1;
while (e <= 5)
{
    cout << e <<" ";
    e++;
}
output:
1 2 3 4 5

```

```

//menggunakan descending
int c=5;
while (c >= 1)
{
    cout << c <<" ";
    c--;
}
output:
5 4 3 2 1

```

C. PERULANGAN *DO...WHILE*

Pada dasarnya struktur perulangan *do...while* sama saja dengan struktur *while*, hanya saja pada proses perulangan dengan *while*, seleksi berada di *while* yang letaknya di atas sementara pada perulangan *do...while*, seleksi *while* berada di bawah batas perulangan. Jadi dengan menggunakan struktur *do...while* sekurang-kurangnya akan terjadi satu kali perulangan.

Bentuk umumnya:

```
//struktur pengulangan do.. while
nama_variabel=nilai_awal;
do {
    pernyataan_yang_akan_diulang;
    variabel_peningkatan;
}
while (kondisi);
```

Contoh perulangan *do... while*:

```
//menggunakan ascending
int e=1;
do {
    cout << e <<" ";
    e++;
}
while (e <= 7);

output:
1 2 3 4 5 6 7
```

```
//menggunakan descending
int f=7;
do {
    cout << f <<" ";
    f--;
}
while (f >= 1);

output:
7 6 5 4 3 2 1
```

Soal Latihan:

1. Jelaskan perbedaan dari perulangan *while* dan *do..while*!
2. Buatlah *flowchart* untuk menampilkan *output* berikut menggunakan algoritma perulangan!

```
10  8  6  4  2
```

3. Buatlah *flowchart* untuk menampilkan *output* berikut menggunakan algoritma perulangan!

```
5 1 2 3 4 5
4 1 2 3 4 5
3 1 2 3 4 5
2 1 2 3 4 5
```

1 1 2 3 4 5

4. Buatlah *flowchart* untuk menampilkan *output* berikut menggunakan algoritma perulangan!

PROGRAMMING 5 3 1 PROGRAMMING 5 3 1 PROGRAMMING 5 3 1

5. Tuliskanlah *output* dari program sederhana berikut!

<pre>#include <iostream.h> #include <conio.h> void main() { clrscr(); int a; for (a=1; a<=5; a++) cout << a <<" "; getche(); }</pre>

6. Tuliskanlah *output* dari program sederhana berikut!

<pre>#include <iostream.h> #include <conio.h> void main() { clrscr(); int a, b; for (a=1; a<=5; a++) { b=a*a; cout << b <<" "; } getche(); }</pre>

BAGIAN 8

PENGENALAN BAHASA C++

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

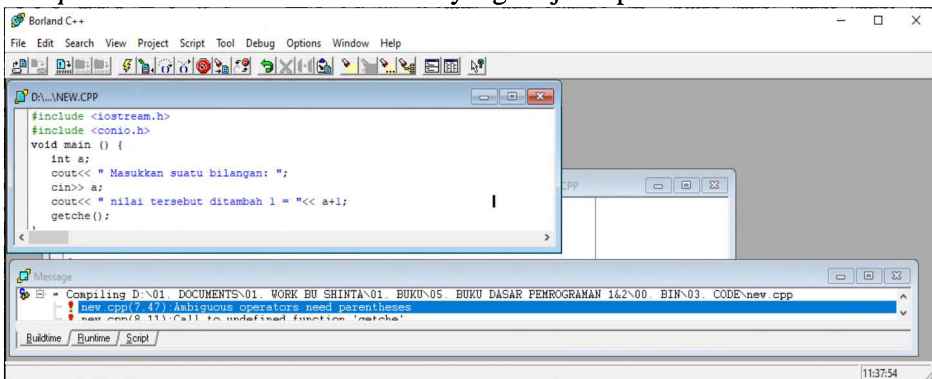
1. Memahami konsep bahasa pemrograman C++
2. Memahami penulisan sintaks dasar, dan struktur penulisan, *input-output*, komentar, dan operator dalam bahasa C++.

Teori:

Bahasa C++ diciptakan oleh Bjarne Stroustrup di AT&T Bell Laboratories awal tahun 1980-an berdasarkan C ANSI (*American National Standard Institute*). Pertama kali, prototype C++ muncul sebagai C yang diperanggih dengan fasilitas kelas. Bahasa tersebut disebut C dengan kelas (*C with class*). Selama tahun 1983-1984, C dengan kelas disempurnakan dengan menambahkan fasilitas pembebanan lebih operator dan fungsi yang kemudian melahirkan apa yang disebut C++. Symbol ++ merupakan operator C untuk operasi penaikan, muncul untuk menunjukkan bahwa bahasa baru ini merupakan versi yang lebih canggih dari C.

Borland International merilis *compiler* Borland C++ dan Turbo C++. Kedua *compiler* ini sama-sama dapat digunakan untuk mengkompilasi kode C++. Bedanya, Borland C++ selain dapat digunakan dibawah lingkungan DOS, juga dapat digunakan untuk pemrograman basis Windows. Selain *Borland International*, beberapa perusahaan lain juga merilis *compiler* C++, seperti *Topspeed C++* dan *Zortech C++*.

Seluruh contoh, soal dan praktik pemrograman dalam buku ini menggunakan *Compiler Borland C++ Version 5.02* yang berjalan pada Windows.



A. STRUKTUR PENULISAN C++

Program C++ tersusun atas sejumlah fungsi. Minimal dalam satu program bahasa C++ ada satu Fungsi. Contoh fungsi:

```
main() {  
    ...  
}
```

Setiap fungsi terdiri dari satu atau beberapa pernyataan (*statement*). *Statement* adalah suatu pernyataan yang akan diproses oleh compiler C++ sebagai suatu perintah untuk menjalankan logika tertentu. Penulisan suatu fungsi diawali dengan { dan diakhiri dengan }. C++ disebut bahasa terstruktur karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian.

Fungsi dari **main()**:

- Merupakan fungsi istimewa
- Fungsi yang harus ada pada program C++
- Fungsi ini menjadi titik awal dan titik akhir eksekusi program
- Biasa ditempatkan pada posisi paling atas dari program
- Memudahkan untuk mencari program utama.

```
main() {  
    statement;  
}
```



Fungsi utama

```
fungsi_lain() {  
    statement;  
    statement2;  
}
```



Fungsi-fungsi lain
yang ditulis oleh
programmer

B. PREPROCESSOR DIRECTIVE (#INCLUDE)

#include merupakan salah satu jenis pengarah preprosesor (*preprocessor directive*) atau sering disebut dengan Header. Dipakai untuk membaca file yang diantaranya berisi deklarasi fungsi dan definisi konstanta. Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Beberapa file judul disediakan dalam C++ nama filenya diakhiri dengan ekstensi .h Contoh:

```
#include <iostream.h>
```

Menyatakan pada kompiler agar membaca file bernama **iostream.h** saat pelaksanaan kompilasi. Bentuk umumnya:

```
#include <iostream.h>
void main() {
    statement;
}
```

Berikut adalah fungsi dari *header* yang akan sering digunakan:

- 1. **stdio.h:** Merupakan singkatan dari standar input output header yang digunakan sebagai standar input output operasi yang digunakan oleh bahasa C, akan tetapi bisa juga digunakan dalam bahasa C++. Fungsi-fungsi yang ada didalam stdio.h antara lain sebagai berikut:

printf()	Merupakan fungsi keluaran yang digunakan untuk menampilkan informasi/pesan kelayar secara terformat (menentukan tipe data yang akan dikeluarkan).
puts()	Merupakan fungsi keluaran yang digunakan untuk menampilkan informasi/pesan yang bertipe data string (tanpa harus melakukan penentuan tipe data terlebih dahulu).
scanf()	merupakan fungsi masukan yang digunakan untuk menginputkan data numerik, karakter, dan string secara terformat (menentukan tipe data yang akan dimasukan).
gets()	merupakan fungsi masukan yang khusus untuk menerima masukan tipe data string (tanpa harus melakukan penentuan tipe data terlebih dahulu).

- 2. **iostream.h:** Merupakan singkatan dari input outout stream header yang digunakan sebagai standar input output operasi yang digunakan oleh bahasa C++. Fungsi-fungsi yang ada didalam iostream.h antara lain sebagai berikut:

cout	Merupakan fungsi keluaran pada C++ yang menampilkan data dengan tipe data apapun kelayar.
cin	Merupakan fungsi masukan pada C++ yang bisa memasukan data berupa numerik dan karakter.
endl	Merupakan suatu fungsi yang manipulator yang digunakan untuk melakukan perintah <i>Newline</i> atau pindah baris.

- 3. **conio.h:** Merupakan *header file* yang berfungsi untuk menampilkan hasil antarmuka kepada pengguna. Fungsi -fungsi yang ada didalam conio.h antara lain sebagai berikut:

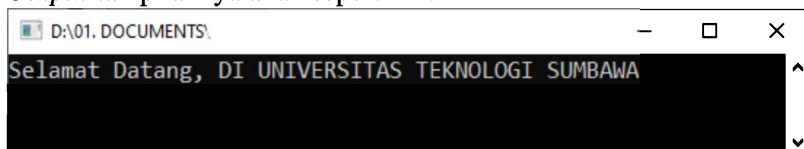
clrscr()	Merupakan singkatan dari clear screen yang digunakan untuk membersihkan layar windows.
getch()	Merupakan singkatan dari get character and echo yang digunakan untuk menahan (pause) output suatu program dan akan kembali mengeksekusi setelah melakukan inputan baik itu tombol enter atau tombol lainnya dan inputan tersebut tidak ditampilkan dalam window.
getche()	Secara fungsi sama dengan getch() akan tetapi ketika melakukan inputan, inputan tersebut tampil dalam window.

C. PENGENALAN FUNGSI CETAK

Untuk menampilkan suatu keluaran pada layar/monitor. Melibatkan file `iostream.h`. Contoh: Menampilkan kalimat “Selamat Datang, DI UNIVERSITAS TEKNOLOGI SUMBAWA”.

```
#include <iostream.h>
#include <conio.h>
void main() {
    cout << "Selamat Datang, DI UNIVERSITAS TEKNOLOGI SUMBAWA";
    getche();
}
```

Output tampilannya akan seperti ini:



Tanda ‘\’ atau *escape sequence* dapat digunakan pada fungsi cetak:

- \n menyatakan karakter baris baru
- \" menyatakan karakter petik ganda
- \\ menyatakan karakter *backslash* ‘\’
- \t menyatakan karakter tab.

Contoh dalam penulisan program:

```
#include <iostream.h>
#include <conio.h>
void main() {
    cout << "No : " << "10";
```

```

    cout << "\nNama : " << "Ali Imron";
    cout << "\nNilai : " << 80.5;
    cout << "\nHuruf : " << 'A';
    getch();
}

```

Tampilan *output*nya akan seperti ini:



```

D:\01. DOCUMENTS'
No : 10
Nama : Ali Imron
Nilai : 80.5
Huruf : A

```

D. KOMENTAR DALAM PROGRAM

Untuk menampilkan suatu keluaran pada layar/monitor. Melibatkan file `iostream.h`. Contoh: (Menampilkan kalimat “Selamat datang di UTS”) Digunakan untuk keperluan dokumentasi. Dimulai dengan tanda “/*” dan diakhiri dengan tanda “*/”. Untuk komentar yang hanya satu baris ditulis dengan diawali tanda “//”. Contoh program:

```

#include <iostream.h>
#include <conio.h>

void main()
{
    cout<< "Selamat Datang! \n"; //komentar satu baris
    cout<< "Di UNIVERSITAS TEKNOLOGI SUMBAWA\n"; /*Tanda ini
    adalah komentar tidak masuk dalam eksekusi program */
    getch();
}

```

Tampilan *output*nya akan seperti ini:



```

D:\01. DOCUMENTS'
Selamat Datang!
Di UNIVERSITAS TEKNOLOGI SUMBAWA

```

E. OPERATOR

Operator adalah simbol atau karakter khusus yang menghasilkan suatu nilai. *Operator* meliputi:

1. *Operator Aritmatika*

Operator ini digunakan untuk perhitungan aritmatika. Berikut adalah contoh *operator* perhitungan aritmatika.

- * (Perkalian)
- / (Pembagian)
- % (Sisa pembagian/modulus)
- +
- (Pengurangan)

Catatan: *Operator* % digunakan untuk mencari sisa pembagian antara dua bilangan.

2. *Operator Hubungan (Perbandingan)*

Operator Hubungan digunakan untuk membandingkan hubungan antara dua buah *operand* (sebuah nilai atau variabel). Berikut adalah *operator* hubungan dalam bahasa C++:

Operator	Arti	Contoh
<	<i>Less than</i>	$x < y$
<=	<i>Less than or equal to</i>	$x <= y$
>	<i>Greater than</i>	$x > y$
>=	<i>Greater than or equal to</i>	$x >= y$
==	<i>Equal to</i>	$x == y$
!=	<i>Not equal to</i>	$x != y$

3. *Operator Logika*

Jika *operator* hubungan membandingkan hubungan antara dua buah *operand*, maka *operator* logika digunakan untuk membandingkan logika hasil dari *operator-operator* hubungan.

Operator logika ada tiga macam, yaitu:

Operator	Arti
& &	Logika <i>AND</i> (Dan)
	Logika <i>OR</i> (Atau)
!	Logika <i>NOT</i> (Ingkaran)

4. *Operator Penugasan*

Operator ini digunakan untuk memberikan nilai ke variabel

Operator	Keterangan	Contoh
----------	------------	--------

=	Pemberian nilai	$x = 2$
+=	Penambahan bilangan	$x += 2$ sama dengan $x = x + 2$
-=	Pengurangan bilangan	$x -= 2$ sama dengan $x = x - 2$
*=	Pengalian bilangan	$x *= 2$ sama dengan $x = x * 2$
/=	Pembagian bilangan	$x /= 2$ sama dengan $x = x / 2$
%=	Pemerolehan sisa bagi	$x \% = 2$ sama dengan $x = x \% 2$

5. Operator Perbandingan

Operator yang digunakan untuk membandingkan dua nilai

- > Lebih besar
- >= lebih besar sama dengan
- < kurang dari
- <= kurang dari sama dengan
- == sama dengan
- != tidak sama dengan

Contoh:

9 < 12	hasilnya benar (1)
14 < 5	hasilnya salah (0)

F. INPUT DAN OUTPUT

Dalam ANSI C, operasi input dan *output* dilakukan dengan menggunakan fungsi-fungsi yang ada di header file *iostream.h*. contohnya untuk input dan *output* ke layar monitor digunakan perintah seperti *cin*, *cout*, *endl*, dsb. Untuk *input* dan *output* ke file digunakan perintah seperti *fread*, *fwrite*, *fputc*, dsb.

C++ mempunyai teknik input dan *output* yang baru, yaitu menggunakan *stream*. *Header file* untuk input dan *output* stream adalah *iostream.h* dan beberapa file lain, seperti *strstream.h*, *fstream.h*, dan *constream.h*.

Stream adalah suatu logika *device* (peralatan logika) yang menghasilkan dan menerima informasi atau suatu wadah yang digunakan untuk menampung keluaran dan menampung aliran data. *Stream* adalah nama umum untuk menampung aliran data (contoh: *file*, *keyboard*, *mouse*), maupun untuk

keluaran (contoh: *layer*, *printer*). Dalam C++ input berarti membaca dari stream dan *output* berarti menulis ke stream.

Dalam bahasa C++ proses memasukkan suatu data bisa menggunakan beberapa fungsi pustaka yang telah tersedia. Beberapa fungsi pustaka yang bisa digunakan adalah:

A. *Cout* <<

Fungsi ini digunakan untuk mengeluarkan *output* data berupa data numerik, karakter dan string.

Bentuk umumnya:

```
cout << ekspresi;
```

Contoh:

```
#include <iostream.h>
void main () {
    int x;
    cout << "Masukkan sebuah bilangan: ";
    cin >> x;
    cout << "Bilangan yang dimasukkan adalah: "<< x;
    getch();
}
```

B. *Cin* >>

Fungsi ini digunakan untuk menginput data berupa data numerik, karakter dan string.

Bentuk umumnya:

```
cin >> variable;
```

Contoh:

```
#include <iostream.h>
#include <conio.h>
void main () {
    int a;
    cout<< "Masukkan suatu bilangan: ";
    cin>> a;
    getch();
}
```

C. *Getchar*()

Fungsi *getchar()* digunakan untuk membaca data yang bertipe karakter. Harus diakhiri dengan penekanan tombol enter. Karakter yang dimasukkan terlihat pada layar. Pergantian baris secara otomatis.

D. *Getch()* dan *Getche()*

Fungsi *getch()* dan *getche()* digunakan untuk membaca data karakter. Karakter yang dimasukkan tidak perlu diakhiri dengan penekanan tombol enter. Tidak memberikan efek pergantian baris secara otomatis. Jika menggunakan fungsi *getch()* karakter yang dimasukkan tidak akan ditampilkan pada layar sehingga sering digunakan untuk meminta inputan berupa password. Sedangkan pada *getche()* karakter yang dimasukkan akan ditampilkan pada layar. Contoh program:

```
#include <iostream.h>
#include <conio.h>
void main () {
    clrscr();
    cout<< "Selamat Datang!";
    getch();
}
```

Soal Latihan:

1. Jelaskan sejarah singkat tentang bahasa pemrograman C++ !
2. Jelaskan dengan singkat struktur bahasa pemrograman C++ !
3. Buatlah program sederhana menggunakan *flowchart* untuk menampilkan *output* berikut dengan mengimplementasikan algoritma perulangan!

324 256 196 144 100

4. Tuliskanlah *output* dari program sederhana berikut!

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int a=1, n, t=0;
    for (n=1; n<=3; n++)
    {
        cout << a <<" ";
        t=t+a;
        a=a+2;
    }
    cout << "\nTotal = " << t;
    getch();
}
```

5. Buatlah sebuah *source code* program yang terdiri dari *input*, proses dan *output* serta *getche()*! (selain dari contoh yang telah diberi)
6. Buatlah sebuah *source code* program yang didalamnya terdapat *input*, proses operator aritmatika, dan menampilkan hasil dari proses aritmatika tersebut!

BAGIAN 9

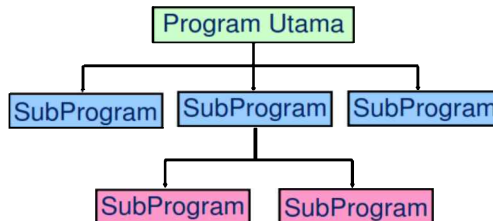
PEMROGRAMAN MODULAR (PROSEDUR DAN FUNGSI)

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

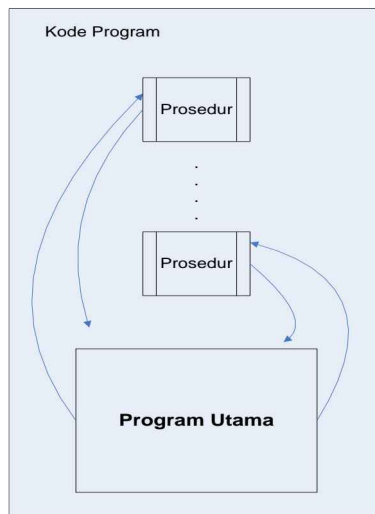
1. Memahami tentang konsep pemrograman modular
2. Mampu membedakan serta mengimplementasikan pemrograman modular dengan prosedur dan fungsi ke dalam bahasa pemrograman C++.

Teori:



Pemrograman modular menggunakan perancangan program menyerdehanakan persoalan didalam program dengan memecah atau membagi persoalan tersebut menjadi sub-sub persoalan yang lebih kecil agar mudah diselesaikan. Secara umum dikenal dua cara yang dapat digunakan untuk memecahkan persoalan dalam modul-modul, yaitu dengan menggunakan prosedur dan fungsi.

A. PROSEDUR



Prosedur adalah sebuah blok program tersendiri, yang merupakan bagian dari program lain yang lebih besar. Contoh:

```
#include <iostream.h>
#include <conio.h>
void info_program();

void main() {
    info_program();
    getch();
    cout << "\n\n";
    info_program();
    getch();
}

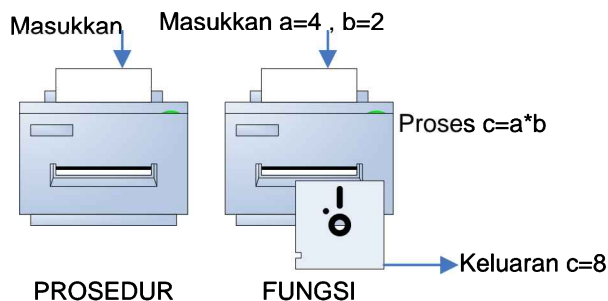
void info_program() {
    cout<< "Selamat Belajar\n";
    cout<< "Di Universitas Teknologi Sumbawa";
}
```

Output program diatas adalah:



B. FUNGSI

Fungsi adalah sebuah blok program tersendiri yang merupakan bagian dari program lain yang lebih besar sama halnya dengan prosedur hanya saja fungsi memiliki hasil keluaran.



Contoh program fungsi pertama:

```
#include <iostream.h>
#include <conio.h>
int kuadrat(int b);
void main() {
    cout<< "kuadrat 2 adalah "<< kuadrat(2);
    cout<< "\nkuadrat 3 adalah "<< kuadrat(3);
    cout<< "\nkuadrat 4 adalah "<< kuadrat(4);
    getch();
}
int kuadrat(int b) {
    int z;
    z = b*b;
    return z;
}
```

Output program diatas adalah:



```
D:\01. DOCUMENTS\
kuadrat 2 adalah 4
kuadrat 3 adalah 9
kuadrat 4 adalah 16
```

Contoh program fungsi kedua:

```
#include <iostream.h>
#include <conio.h>
int pangkat (int j, int k);
void main() {
    int x, a, hasil;
    cout<< "Program Bilangan Perpangkatan \n";
    cout<< "Masukkan nilai x = "; cin>> x;
    cout<< "\nMasukkan nilai a = "; cin>> a;
    cout<< "\nHasil pemangkatan "<<x <<"^" <<a <<" = " <<
    pangkat(x,a);
    getch();
}
int pangkat (int j, int k) {
    int i, hasil= 1;
    for (i=1; i<=k; i++)
        hasil = hasil*j;
}
```

Output program diatas adalah:

A screenshot of a Windows command prompt window. The title bar shows the path 'D:\01. DOCUMENTS\'. The window has standard minimize, maximize, and close buttons. The command prompt displays the following text: 'Program Bilangan Perpangkatan', 'Masukkan nilai x = 5', 'Masukkan nilai a = 3', and 'Hasil pemangkatan 5^3 = 125'.

```
D:\01. DOCUMENTS\  
Program Bilangan Perpangkatan  
Masukkan nilai x = 5  
Masukkan nilai a = 3  
Hasil pemangkatan 5^3 = 125
```

Soal Latihan:

1. Buatlah program sederhana menggunakan pemrograman modular prosedur! (*selain dari contoh*)
2. Buatlah program sederhana dengan tema “Aplikasi Penghitung” yang didalamnya menggunakan program modular fungsi untuk mencari:
 - a. Luas persegi panjang
 - b. Volume bola

BAGIAN 10

PEMROGRAMAN ARRAY

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami konsep pemrograman *array* dalam bahasa C++
2. Memahami penulisan sintaks *array* dan mampu mengimplementasikan konsep pemrograman *array* dalam bahasa C++.

Teori:

Array merupakan kumpulan dari nilai-nilai data yang terstruktur dan merujuk kepada sebuah atau sekumpulan elemen yang mempunyai tipe data yang sama melalui indeks. *Array* biasanya disebut juga sebagai tabel, vektor atau larik. Elemen dari *array* dapat diakses langsung jika dan hanya jika indeks terdefinisi (telah ditentukan nilainya sesuai dengan domain yang didefinisikan untuk indeks tersebut). Struktur data *array* disimpan dengan urutan yang sesuai dengan definisi indeks secara kontinu (berurutan) dalam memori komputer. Karena itu indeks haruslah merupakan suatu tipe data yang memiliki keterurutan (ada suksesor dan predesesor), misal tipe integer dan karakter.

Dilihat dari dimensinya, *array* dapat dibagi menjadi *Array* Satu Dimensi, *Array* Dua Dimensi dan *Array* Multi-Dimensi.

A. ARRAY SATU DIMENSI

Array satu dimensi diakses melalui indeksnya. Misal akan disiapkan *array* satu dimensi A bertipe integer dengan 5 elemen yang diberi nomor indeks dari 0 sampai 4, yang dapat diilustrasikan sebagai berikut:

	Nilai[0]	Nilai[1]	Nilai[2]	Nilai[3]	Nilai[4]
int Nilai[5]:	70	80	82	60	75

Karena *array* tersebut mempunyai nama yang sama, yaitu A, maka setiap elemen diberi sebutan nama yang berbeda dengan memberikan nomor indeks, sehingga masing-masing menjadi: A[0], A[1], sampai dengan A[4], yang dapat dibaca dengan:

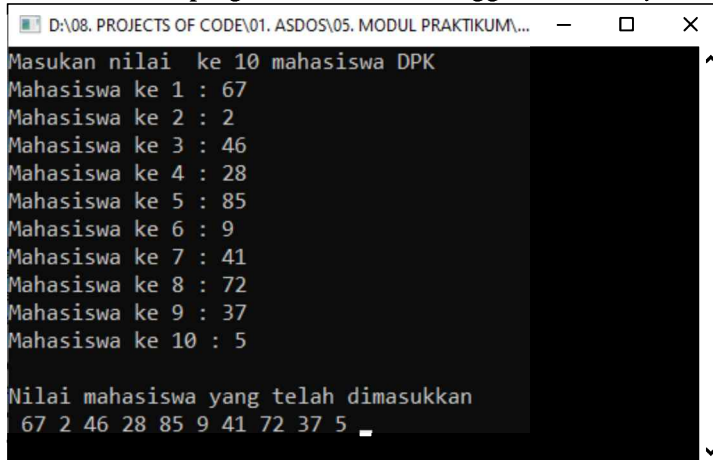
- A dengan indeks 0 atau A [nol]
- A dengan indeks 1 atau A [satu]
- A dengan indeks 2 atau A [dua]

- A dengan indeks 3 atau A [tiga]
- Dan seterusnya...

Bentuk umum penulisan array satu dimensi:

```
tipe_array nama_array[ukuran];
```

Contoh sebuah program sederhana menggunakan *array*:



```
D:\08. PROJECTS OF CODE\01. ASDOS\05. MODUL PRAKTIKUM\... - □ ×
Masukan nilai ke 10 mahasiswa DPK
Mahasiswa ke 1 : 67
Mahasiswa ke 2 : 2
Mahasiswa ke 3 : 46
Mahasiswa ke 4 : 28
Mahasiswa ke 5 : 85
Mahasiswa ke 6 : 9
Mahasiswa ke 7 : 41
Mahasiswa ke 8 : 72
Mahasiswa ke 9 : 37
Mahasiswa ke 10 : 5

Nilai mahasiswa yang telah dimasukkan
67 2 46 28 85 9 41 72 37 5 _
```

Source code program di atas adalah sebagai berikut:

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int index, nilai[10];
    cout << "Masukan nilai ke 10 mahasiswa DPK \n";
    for (index=0; index<10; index++) {
        cout << "Mahasiswa ke " << index+1 << " : ";
        cin >> nilai[index];
    }
    cout << "\nNilai mahasiswa yang telah dimasukkan ";
    cout << "\n ";
    for (index=0; index<10; index++) {
        cout << nilai[index] << " ";
    }
    getch();
}
```

B. ARRAY DUA DIMENSI

Array dua dimensi merupakan *array* yang terdiri dari m buah baris (*row*) dan n buah kolom (*column*). Bentuk *array* semacam ini menggunakan 2 (dua) buah kelompok indeks yang masing-masing direpresentasikan sebagai indeks baris dan kolom. Jika ingin memasukkan atau membaca sebuah nilai pada matriks maka, harus diketahui terlebih dahulu indeks baris dan kolomnya.

Bentuk umum:

```
tipe_data nama_array[baris][kolom];
```

Contoh penulisan *array* mendeklarasikan sebuah variabel *array* 2 dimensi dengan menggunakan tipe data integer dengan nama *a* memiliki ukuran 3 elemen baris dan 4 kolom.

```
int a[3][4];
```

Inisialisasi nilai *array* 2 dimensi dapat dibuat dengan cara manual ataupun dapat diinput oleh pengguna saat program dijalankan. Karena *array* 2 dimensi mempunyai lebih dari satu bentuk index *array*, maka ketika kita melakukan inisialisasi nilai (secara manual) perlu menggunakan tanda {...} untuk membentuk baris *array*. Contoh:

```
int a[3][4] = {{3,4,8,0},{3,9,2,1},{6,3,0,2}};
```

Inisialisasi *array* yang dibuat tersebut dapat digambarkan lewat tabel berikut:

A	Kolom Ke 0	Kolom Ke 1	Kolom Ke 2	Kolom Ke 3
Baris Ke 0	A [0] [0]	A [0] [1]	A [0] [2]	A [0] [3]
Baris Ke 1	A [1] [0]	A [1] [1]	A [1] [2]	A [1] [3]
Baris Ke 2	A [2] [0]	A [2] [1]	A [2] [2]	A [2] [3]

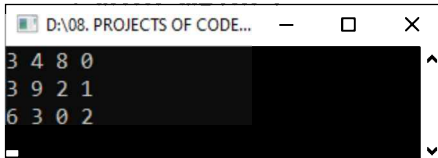
A	Kolom Ke 0	Kolom Ke 1	Kolom Ke 2	Kolom Ke 3
Baris Ke 0	3	4	8	0
Baris Ke 1	3	9	2	1
Baris Ke 2	6	3	0	2

Karena index *array* dimulai dari 0 maka setiap elemem baris dan kolom dimulai dari 0. Nilai 3 akan menempati pada baris ke-0 dan kolom ke-0, nilai 4 menempati baris ke-0 kolom ke-1 nilai, 8 pada baris ke-0 dan kolom ke-2 dan seterusnya.

Mengakses nilai *array* dapat dilakukan dengan cara manual dimana kita langsung mencetak sesuai dengan index baris dan kolom pada *array* tersebut. Sebagai contoh jika kita ingin mengakses nilai 1 maka index *array* yang kita akses adalah `a[1][3]`; karena 1 berada pada elemen baris ke-1 dan pada kolom ke-3. Berikut adalah contoh program sederhana *array* dua dimensi:

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int b, k;
    int a[3][4] = {{3,4,8,0}, {3,9,2,1}, {6,3,0,2}};
    for (b=0; b<3; b++) {
        for (k=0; k<4; k++) {
            cout << a[b][k] << " ";
        }
        cout << "\n";
    }
    getch();
}
```

Output program array di atas:



```
D:\08. PROJECTS OF CODE...
3 4 8 0
3 9 2 1
6 3 0 2
```

C. ARRAY MULTI-DIMENSI

Dalam menggambarkan *array* multidimensi, hanya terbatas hingga dimensi ke-3, yakni dengan menggunakan bangun ruang, namun dalam kenyataannya, tipe data *array* ini dapat dibentuk menjadi lebih dari tiga dimensi atau menjadi *n* dimensi.

Soal Latihan:

1. Apakah yang dimaksud dengan pemrograman *array*?
2. Apa perbedaan *array* satu dimensi dengan *array* dua dimensi?
3. Buatlah sebuah program, yang didalamnya menggunakan pemrograman *array*! (*selain kasus yang serupa dengan contoh*)

PRAKTIKUM KEDUA

Soal Praktikum:

1. (Soal Kasus) Sebuah program penghitung tarif pada salah satu klinik gigi di Sumbawa. Buatlah program sederhana untuk menampilkan *output* dibawah ini!

PENGHITUNGAN BIAYA KLINIK GIGI BAROKAH	

Pilihan Pengobatan;	
[1] Pemeriksaan Gigi	[4] Bersihkan Kapur Gigi
[2] Cabut Gigi	[5] Pemasangan Kawat Gigi
[3] Tambal Gigi	[6] Operasi Ringan
Masukan Kode Pengobatan	: _____
Masukkan Jumlah Obat	: _____
Jumlah Biaya Yang Dibayar	: Rp. _____
Jumlah Uang Anda	: Rp. _____ //input
Uang Kembalian	: Rp. _____

Ketentuan:

Biaya Periksa Gigi	= Rp. 125.000
Biaya Cabut Gigi	= Rp. 150.000
Biaya Tambal Gigi	= Rp. 225.000
Biaya Bersih Kapur Gigi	= Rp. 300.000
Biaya Pasang Kawat Gigi	= Rp. 3.500.000
Biaya Operasi Ringan	= Rp. 5.000.000

Jika obat 0-2, maka diskon 15%

Jika obat 3-5, maka diskon 20%

Jika obat lebih dari 5, maka diskon 25%

Rumus:

$\text{biaya yg dibayar} = \text{biaya pengobatan} - (\text{biaya pengobatan} \times \text{diskon})$
--

2. Buatlah program sederhana dengan pemrograman C++ untuk menampilkan *output*:

1 4 9 16 25
Total = 55

3. Buatlah program sederhana dengan pemrograman C++ untuk menampilkan *output*:

0 9 25 49 81

4. Tuliskan *output* dari program di bawah ini:

```
#include <iostream.h>
#include <conio.h>
int pangkat(int bilangan, int pangkat);
void main()
{
    clrscr();
    int bil=11, pangkt=3;
    cout<< "Bilangan Anda adalah " << bil;
    cout<< "\nPangkat yang digunakan adalah "<< pangkt;
    cout<< "\nHasil dari "<< bil<< " ^ "<< pangkt<< " =
" << pangkat(bil, pangkt);
    getch();
}

int pangkat(int bilangan, int pangkat)
{
    int i, hasil=1;
    for (i=1; i<=pangkat; i++)
        hasil = hasil*bilangan;
}
```

EVALUASI PEMAHAMAN

Soal Evaluasi:

1. (Soal Kasus) *Bank Shinta* dalam perhitungan deposito ditentukan dari golongan dan jangka waktu (periode/bulan). Buatlah program dengan menggunakan struktur penyeleksian kondisi dalam bahasa C++! Dengan *output* seperti berikut:

```
*****
PERHITUNGAN DEPOSITO BANK SHINTA
*****
Masukkan Golongan : _____ //input
Masukkan Periode  : _____ //input

Bunga              : Rp_____ atau ____ %
Jumlah Uang Awal  : Rp_____
Jumlah Uang Akhir  : Rp_____
```

Ketentuan:

```
Golongan A jum_awal = Rp 2.000.000
Golongan B jum_awal = Rp 1.750.000
Golongan C jum_awal = Rp 1.250.000
Golongan D jum_awal = Rp 1.000.000
Selain dari itu jum_awal = Rp 750.000
```

Diketahui:

```
Periode 1 – 4, bunga 18% per tahun
Periode 5 – 8, bunga 22% per tahun
Periode 9 – 12, bunga 25% per tahun
Periode > 12, bunga 27% per tahun
```

Rumus perhitungan deposito:

```
dana_bunga = ((periode/12)*persentase_bunga)*jum_awal
jum_akhir = dana_bunga+jum_awal
```

2. Tuliskan *output* dari program sederhana berikut!

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int a, b, c=1, d, e;
    cout << "ALGORITMA" << "\n";
    for (a=1; a<=2; a++) {
        cout << "ALGORITMA" << " ";
        for (b=1; b<=c; b++) {
            cout << "PEMROGRAMAN" << " ";
        }
        cout << "\n";
        c=c+1;
    }
    for (d=1; d<=1; d++) {
        cout << "ALGORITMA" << " ";
        for (e=1; e<=1; e++) {
            cout << "PEMROGRAMAN" << "\n";
        }
        cout << "ALGORITMA";
    }
    getch();
}
```

3. Tuliskan *output* dari program sederhana berikut!

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int matriks[2][4] = {{12,31,52,71}, {51,61,71,81}};
    int baris, kolom, jumlah=0;
    for (baris=0; baris<2; baris++) {
        for (kolom=0; kolom<4; kolom++) {
            jumlah = jumlah + matriks[baris][kolom];
        }
    }
    cout << "Jumlah dari setiap nilai pada Array 2 dimensi  
adalah: " <<jumlah;
    getch();
}
```

4. Tuliskan *output* dari program sederhana berikut!

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    cout << "Program Menampilkan Bintang: \n";
    int a, b;
    for (a=1; a<=4; a++) {
        for (b=1; b<=a; b++) {
            cout << "*";
        }
        cout << "\n";
    }
    for (a=1; a<=5; a++) {
        for (b=5; b>=a; b--) {
            cout << "*";
        }
        cout << "\n";
    }
    getch();
}
```

BAGIAN 11

ALGORITMA *SEQUENTIAL* (LANJUTAN)

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu menguasai dan mengembangkan konsep algoritma runtutan/*sequential*.

Teori:

ALGORITMA *SEQUENTIAL* (LANJUTAN)

Algoritma runtutan (*sequential*) merupakan salah satu struktur dasar algoritma yang bisa dikatakan cukup sederhana jika dibandingkan dengan struktur algoritma yang lain. Algoritma *sequential* bekerja dengan cara mengeksekusi setiap instruksi secara berurutan. Setiap instruksi akan dikerjakan satu per satu pada setiap barisnya dari awal hingga akhir, sesuai dengan urutan penulisan instruksi tersebut. Jadi, jika terdapat algoritma dengan urutan perintahnya adalah:

instruksi 1
instruksi 2
instruksi 3
instruksi 4
... dan seterusnya

Algoritma *sequential* akan mengeksekusi instruksi 1, kemudian setelah itu instruksi 2 dan seterusnya. Setiap instruksi dikerjakan satu persatu dan hanya sekali (tidak ada instruksi yang diulang) sampai instruksi ke-*n* sebagai instruksi terakhir merupakan akhir dari proses algoritmanya.

Algoritma *sequential* biasanya digunakan untuk program yang sederhana seperti program menghitung luas lingkaran & luas segitiga atau program lainnya yang hanya terdiri dari proses komputer dan *input/output* dari user. Berikut ini beberapa contoh penerapan algoritma sekuensial ke dalam code C++.

Contoh Program:

1. Algoritma Menghitung Kelereng

Soni mempunyai kelereng yang jumlahnya 10 buah lebih banyak dari kelereng Adi. Sedangkan Anis memiliki kelereng sebanyak 2 x kelereng Soni dan Adi. Luki memiliki kelereng sebanyak 5 buah lebih sedikit dari jumlah kelereng

Soni, Adi dan Anis. Carilah banyak kelereng Adi, Anis dan Luki, jika diketahui jumlah kelereng Soni! Buatlah program sederhana dari masalah tersebut dengan pseudocode dan coding C++!

Program Menghitung Kelereng dengan pseudocode:

Judul: {Aplikasi menghitung kelereng Adi, Anis dan Luki}

Deklarasi: k_soni, k_adi, k_anis, k_luki = integer

Deskripsi:

1. start
2. cin (k_soni)
3. $k_adi \leftarrow k_soni - 10$
4. $k_anis \leftarrow 2 * (k_soni + k_adi)$
5. $k_luki \leftarrow k_soni + k_adi + k_anis - 5$
6. cout (k_adi, k_anis, k_luki)
7. end

Program Menghitung Kelereng dengan Code C++:

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

main()
{
    int k_soni, k_adi, k_anis, k_luki;
    cout << "APLIKASI MENGHITUNG KELERENG\n";
    cout << "Kelereng Soni : ";
    cin >> k_soni;
    k_adi = k_soni - 10;
    k_anis = 2 * (k_soni + k_adi);
    k_luki = k_soni + k_adi + k_anis - 5;
    cout << "\nKelereng Adi   : " << k_adi;
    cout << "\nKelereng Anis  : " << k_anis;
    cout << "\nKelereng Luki   : " << k_luki;
    getch();
}
```

Output program di atas adalah:

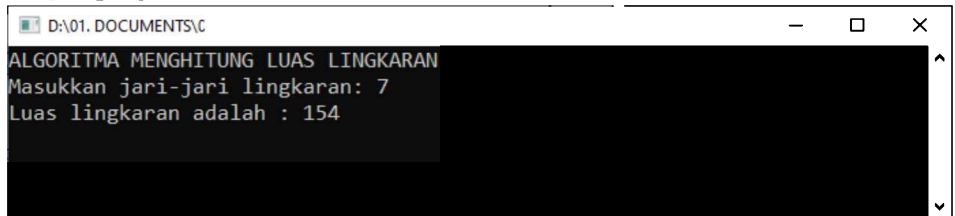


2. Algoritma Menghitung Luas Lingkaran

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

void main()
{
    float phi, jari, luas;
    phi=22.000/7;
    cout << "ALGORITMA MENGHITUNG LUAS LINGKARAN\n";
    cout << "Masukkan jari-jari lingkaran: ";
    cin >> jari;
    luas=phi*jari*jari;
    cout << "Luas lingkaran adalah : " << luas;
    getch();
}
```

Output program di atas adalah:

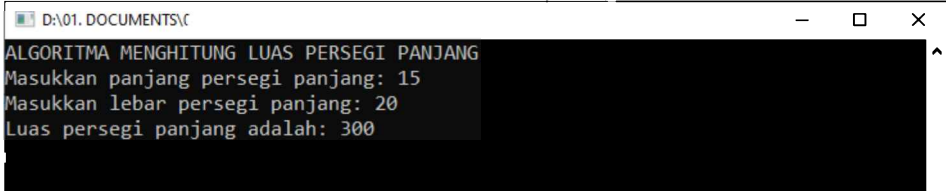
A screenshot of a Windows command prompt window. The title bar shows the file path 'D:\01. DOCUMENTS\C'. The window contains the following text: 'ALGORITMA MENGHITUNG LUAS LINGKARAN', 'Masukkan jari-jari lingkaran: 7', and 'Luas lingkaran adalah : 154'. The text is displayed in a monospaced font on a black background with white text. The window has standard Windows controls (minimize, maximize, close) in the top right corner.

3. Algoritma Menghitung Luas Persegi Panjang

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

void main()
{
    int panjang, lebar, hasil;
    cout << "ALGORITMA MENGHITUNG LUAS PERSEGI PANJANG\n";
    cout << "Masukkan panjang persegi panjang: ";
    cin >> panjang;
    cout << "Masukkan lebar persegi panjang: ";
    cin >> lebar;
    hasil=panjang*lebar;
    cout << "Luas persegi panjang adalah: " << hasil;
    getch();
}
```

Output program di atas adalah:



```
D:\01. DOCUMENTS\C
ALGORITMA MENGHITUNG LUAS PERSEGI PANJANG
Masukkan panjang persegi panjang: 15
Masukkan lebar persegi panjang: 20
Luas persegi panjang adalah: 300
```

4. Algoritma Menghitung Keliling Persegi

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

void main()
{
    int sisi, keliling;
    cout << "ALGORITMA MENGHITUNG KELILING PERSEGI\n";
    cout << "Masukkan sisi persegi: ";
    cin >> sisi;
    keliling=sisi*4;
    cout << "Keliling persegi adalah: " << keliling;
    getch();
}
```

Output program di atas adalah:



```
D:\01. DOCUMENTS\C
ALGORITMA MENGHITUNG KELILING PERSEGI
Masukkan sisi persegi: 7
Keliling persegi adalah: 28
```


Soal Latihan:

1. (Soal Kasus) Buatlah program dengan tampilan sebagai berikut:

```
*****
      PERHITUNGAN BIAYA FIRMA HUKUM
*****

Pilih Kode Pengacara;
[A] Nilawati, S.H           [C] DR. Ridwan Basri, M.H
[B] Romeo, M.H             [D] DR. Siska Selvia, M.H

Proses Rincian Biaya;
Masukkan Kode Pengacara   : _____
Masukkan Lama Konsultasi  : _____ jam

Total Biaya Konsultasi    : Rp _____
Potongan Biaya            : Rp _____
Total Bayar               : Rp _____

Jumlah Uang               : Rp _____ //input
Kembali                   : Rp _____
```

Ketentuan:

Biaya konsultasi per/jam Nilawati, S.H	= Rp.200.000
Biaya konsultasi per/jam Romeo, M.H	= Rp.350.000
Biaya konsultasi per/jam DR. Ridwan	= Rp.500.000
Biaya konsultasi per/jam DR. Siska	= Rp.650.000
Jika Konsultasi 1 – 3 jam, maka potongan biaya 0%	
Jika Konsultasi 4 – 6 jam, maka potongan biaya 10%	
Jika lebih dari 6 jam, maka potongan biaya 25%	

Rumus:

$\text{total_biaya_konsultasi} = \text{biaya_konsul} * \text{lama_konsul}$
$\text{potongan_biaya} = \text{total_biaya_konsultasi} * \text{persentase_potongan_biaya}$
$\text{total_bayar} = \text{total_biaya_konsultasi} - \text{potongan_biaya}$

2. (Soal Kasus) Seorang pedagang mangga menjual dagangannya yang setiap Kg mangga dihargai dengan harga tertentu. Setiap pembeli membayar harga mangga yang dibelinya berdasarkan berat. Tentukan perintah apa yang diberikan pedagang mangga kepada komputer agar ia dapat dengan mudah menentukan harga yang harus dibayar pembelinya!

3. Buatlah program sederhana menggunakan algoritma sequential C++, berikut:
 - a. Menghitung usia seseorang, dengan input nama dan tahun lahir!
 - b. Menghitung volume bola, dengan input radius!
 - c. Menghitung jarak tempuh kendaraan (meter), dengan input kecepatan (km/h) dan waktu (jam)!

BAGIAN 12

ALGORITMA *BRANCHING* (LANJUTAN)

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu menguasai dan mengembangkan konsep algoritma percabangan/*branching*.

Teori:

ALGORITMA *BRANCHING* (LANJUTAN)

Algoritma percabangan (*branching*) merupakan algoritma yang tidak selamanya akan berjalan dengan mengikuti struktur berurutan, kadang-kadang perlu merubah urutan pelaksanaan program dan menghendaki agar pelaksanaan program meloncat ke baris tertentu. Pada struktur percabangan, program akan berpindah urutan pelaksanaan jika suatu kondisi yang disyaratkan dipenuhi. Fungsi algoritma percabangan ini pada adalah untuk memproses keputusan yang tepat dan sesuai dengan yang keinginan pengguna sistem berdasarkan beberapa kondisi yang terjadi pada sistem yang digunakan tersebut.

A. Percabangan Untuk 1 Kondisi

Pada percabangan jenis ini, hanya ada satu kondisi yang menjadi syarat untuk melakukan satu buah atau satu blok instruksi. Format umum dari algoritma percabangan dengan satu kondisi adalah sebagai berikut:

```
if (kondisi) then  
instruksi;  
end if
```

Arti dari format di atas, jika “kondisi” bernilai benar atau tercapai, maka aksi dikerjakan. Sedangkan jika bernilai salah, maka instruksi tidak dikerjakan dan proses langsung keluar dari percabangan dan kembali lagi ke kondisi awal.

Contoh Program:

Program percabangan untuk 1 kondisi:

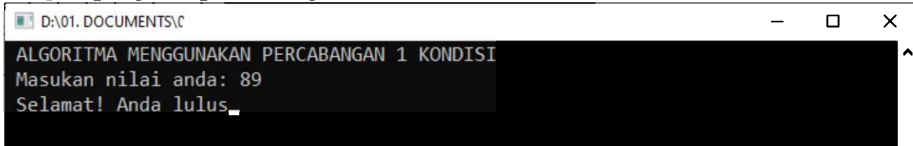
```
#include <iostream.h>  
#include <conio.h>  
  
void main()  
{  
    int nilai;
```

```

    cout << " ALGORITMA MENGGUNAKAN PERCABANGAN 1 KONDISI
\n";
    cout << " Masukkan nilai anda: ";
    cin >> nilai;
    if (nilai > 60)
        cout << " Selamat! Anda lulus";
    getch();
}

```

Output program percabangan 1 kondisi di atas adalah:



```

D:\01.DOCUMENTS\C
ALGORITMA MENGGUNAKAN PERCABANGAN 1 KONDISI
Masukkan nilai anda: 89
Selamat! Anda lulus_

```

B. Percabangan Untuk 2 Kondisi

Pada percabangan jenis ini, ada dua kondisi yang menjadi syarat untuk dikerjakannya salah satu dari dua instruksi. Kondisi ini bisa bernilai benar atau salah. Bentuk umum dari percabangan dengan dua kondisi adalah sebagai berikut:

```

if (kondisi) then
    instruksi 1;
else
    instruksi 2;
end if

```

Arti dari format di atas, jika “kondisi” bernilai benar maka instruksi 1 yang akan dikerjakan. Sedangkan jika bernilai salah), maka instruksi 2 yang akan dikerjakan. Perbedaannya dengan percabangan untuk satu kondisi terletak pada adanya dua instruksi untuk dua kondisi, yaitu kondisi bernilai benar dan kondisi bernilai salah.

Contoh Program:

Sebuah peraturan untuk menonton sebuah film tertentu adalah sebagai berikut, jika usia penonton lebih dari 17 tahun makan penonton diperbolehkan dan apabila 17 tahun kebawah maka penonton tidak diperbolehkan menonton. Buatlah algoritma untuk permasalahan tersebut!

```

#include <iostream.h>
#include <conio.h>

```

```

void main()
{
    int umur;
    cout << "~~~~~ PERATURAN MENONTON FILM ~~~~~\n";
    cout << " Silakan masukan umur anda: ";
    cin >> umur;
    if (umur > 17)
        cout << " Anda boleh menonton film ini, SELAMAT
MENONTON!";
    else
        cout << " Maaf, umur anda masih " << umur << "
tahun. Anda belum diizinkan menonton!";
    getch();
}

```

Output program di atas jika input umur dibawah 17 tahun:

```

D:\01. DOCUMENTS\
~~~~~ PERATURAN MENONTON FILM ~~~~~
Silakan masukan umur anda: 11
Maaf, umur anda masih 11 tahun. Anda belum diizinkan menonton!

```

Output program di atas jika input umur diatas 17 tahun:

```

D:\01. DOCUMENTS\
~~~~~ PERATURAN MENONTON FILM ~~~~~
Silakan masukan umur anda: 23
Anda boleh menonton film ini, SELAMAT MENONTON!

```

C. Percabangan Untuk 3 Kondisi Atau Lebih

Algoritma percabangan untuk tiga kondisi atau lebih adalah bentuk pengembangan dari dua macam algoritma percabangan yang telah dibahas sebelumnya. Karena itu, percabangan jenis ini akan memiliki banyak variasi. Secara umum, format percabangannya dapat dituliskan sebagai berikut:

```

if (kondisi 1) then
    instruksi 1;
else if (kondisi 2) then
    instruksi 2;
else
    instruksi 3;
end if

```

Maksud dari algoritma di atas, instruksi 1 akan dikerjakan jika “kondisi 1” bernilai benar. Jika bernilai salah, pemeriksaan dilanjutkan ke “kondisi 2”. Jika “kondisi 2” bernilai benar, maka instruksi 2 dikerjakan. Jika tidak, pemeriksaan dilanjutkan pada kondisi-kondisi lainnya. Pemeriksaan ini akan terus dilakukan terhadap semua kondisi yang ada. Jika tidak ada satu pun kondisi yang bernilai benar maka pernyataan yang dikerjakan adalah instruksi 3 atau instruksi (n+1) pada percabangan lebih dari 3 kondisi.

Contoh Program:

Sebuah sistem penilaian, mengkonversikan nilai angka mahasiswa menjadi nilai huruf. Dilakukan algoritma 3 seleksi kondisi lebih didalamnya. Berikut adalah contoh program sederhananya, dengan menggunakan satu *input* nilai dan menghasilkan satu *output* nilai tersebut dalam bentuk nilai huruf.

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int nilai;
    cout << "Masukkan Nilai Anda : ";
    cin >> nilai;
    if (nilai >= 0 && nilai <= 60) {
        cout << "Nilai Anda adalah E." << endl;
    }
    else if (nilai >= 61 && nilai <= 70) {
        cout << "Nilai Anda adalah D." << endl;
    }
    else if (nilai >= 71 && nilai <= 80) {
        cout << "Nilai Anda adalah C." << endl;
    }
    else if (nilai >= 81 && nilai <= 90) {
        cout << "Nilai Anda adalah B." << endl;
    }
    else if (nilai >= 91 && nilai <= 100) {
        cout << "Nilai Anda adalah A." << endl;
    }
    else {
        cout << "Anda salah menginput nilai";
    }
    getch();
}
```

Output program percabangan 3 kondisi lebih diatas adalah:



```
D:\01. DOCUMENTS\  
Masukkan Nilai Anda : 79  
Nilai Anda adalah C.
```

D. Percabangan Bersarang

Percabangan bersarang adalah instruksi yang terdiri dari adanya percabangan yang lain di dalam percabangan, atau di dalam percabangan ada percabangan lagi. Format penulisan untuk percabangan bersarang adalah sebagai berikut:

```
if (kondisi 1) then  
    if (kondisi 2) then  
        instruksi 1;  
    else  
        instruksi 2;  
else  
if (kondisi 3) then  
    instruksi 3;  
else  
    instruksi 4;  
end if
```

Jika kondisi berjumlah lebih dari 3 kondisi, polanya tetap sama. Untuk kondisi ke 2 dan seterusnya, penulisannya menggunakan “ELSE IF kondisi THEN”, sedangkan untuk kondisi terakhir cukup menggunakan ELSE saja. Mulanya, “kondisi1” dicek nilai kebenarannya. Jika benar, maka dicek nilai kebenaran “kondisi2”. Jika “kondisi2” benar, maka dikerjakan Instruksi1. Jika tidak, dikerjakan Instruksi2. Sedangkan jika “kondisi1” tidak benar, maka akan dicek nilai kebenarannya. Jika “kondisi3” bernilai benar, maka dikerjakan Instruksi3. Jika tidak, maka akan dikerjakan Instruksi4.

Contoh Program:

Contoh program percabangan bersarang:

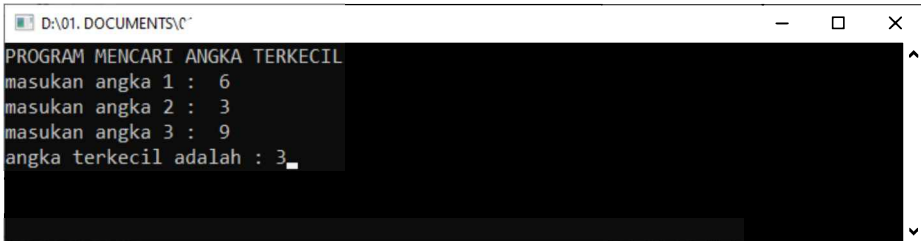
```
#include <iostream.h>  
#include <conio.h>  
void main()  
{  
    int A, B, C;  
    cout << "PROGRAM MENCARI ANGKA TERKECIL\n";
```

```

cout << "masukan angka 1 : ";
cin >> A;
cout << "masukan angka 2 : ";
cin >> B;
cout << "masukan angka 3 : ";
cin >> C;
if (A < B)
{
    if (A < C)
        cout << "angka terkecil adalah : " << A;
    else
        cout << "angka terkecil adalah : " << C;
}
else if (B < C)
    cout << "angka terkecil adalah : " << B;
else
    cout << "angka terkecil adalah : " << C;
getche();
}

```

Output program percabangan bersarang di atas adalah:



```

D:\01. DOCUMENTS\C\"
PROGRAM MENCARI ANGKA TERKECIL
masukan angka 1 : 6
masukan angka 2 : 3
masukan angka 3 : 9
angka terkecil adalah : 3

```

E. Percabangan *Switch Case*

Selain menggunakan format yang dijelaskan pada poin 3, percabangan tiga kondisi atau lebih bisa juga menggunakan format “*Switch Case*”. Format ini memiliki kegunaan yang sama, tetapi format ini digunakan untuk memeriksa data yang bertipe karakter atau integer. Secara umum format penulisannya adalah sebagai berikut:

```

switch (ekspresi) {
case konstanta-1:
    instruksi 1;
    break;
case konstanta-2:
    instruksi 2;

```



```
    break;
default:
    instruksi 3;
}
```

Contoh Program:

Program percabangan *switch case*:

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int hari;
    cout << "Input No Hari [1-7] : ";
    cin >> hari;
    cout << "Anda menginput hari : ";
    switch (hari)
    {
        case 1:
            cout << "Ahad";
            break;
        case 2:
            cout << "Senin";
            break;
        case 3:
            cout << "Selasa";
            break;
        case 4:
            cout << "Rabu";
            break;
        case 5:
            cout << "Kamis";
            break;
        case 6:
            cout << "Jum'at";
            break;
        case 7:
            cout << "Sabtu";
            break;
        default:
            cout << "Error, nomor input antara 1 sampai 7";
    }
    getch();
}
```

Output program percabangan *switch case* diatas adalah:



```
D:\01. DOCUMENTS\<
Input No Hari [1-7] : 5
Anda menginput hari : Kamis
```

Soal Latihan:

1. Jelaskan perbedaan algoritma percabangan *if* dan *switch case*! Dan berikan penjelasan kapan penggunaan yang tepat dari masing-masing *if* dan *switch case*!
2. Apa fungsi *break*; pada sintaks notasi percabangan *switch case*?
3. Dalam suatu perhitungan awal

nilai $P = X + Y$

Jika hasil P adalah positif, maka nilai $Q = X * Y$.

Sedangkan jika negative, maka nilai $Q = X/Y$.

Buatlah program sederhana menggunakan algoritma percabangan dengan ketentuan berikut!

Input : Menginput nilai X dan Y

Proses : Memeriksa kondisi dilakukan pada nilai P apakah positif (termasuk 0) ataukah negatif

Output : Menampilkan nilai Q .

4. Sebuah usaha fotokopi mempunyai aturan sebagai berikut:
 - Jika yang fotokopi statusnya adalah langganan, maka berapa lembar pun dia fotokopi, harga perlembar Rp. 75,-
 - Jika yang fotokopi bukan langganan, maka jika dia fotokopi kurang dari 100 lembar harga perlembar Rp. 100,-
 - Sedangkan jika lebih atau sama dengan 100 lembar, maka harga perlembar menjadi Rp. 85,-

Buatlah program sederhana menggunakan algoritma percabangan untuk menghitung **Total Harga** yang harus dibayar jika seseorang memfotokopi sejumlah x lembar dan dengan ketentuan berlangganan atau tidak berlangganan!

BAGIAN 13

ALGORITMA *LOOPING* (LANJUTAN)

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu menguasai dan mengembangkan konsep algoritma perulangan/*looping*.

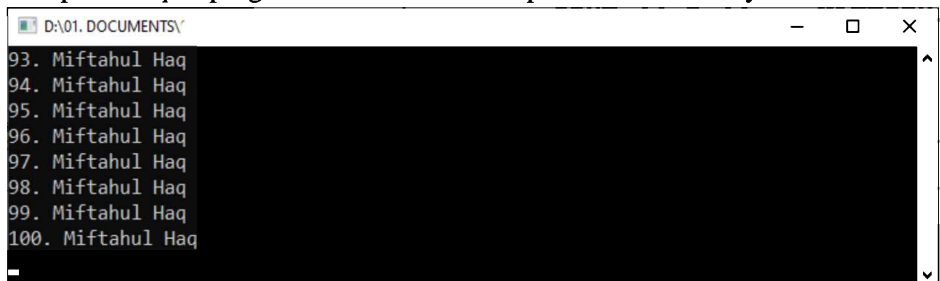
Teori:

ALGORITMA *LOOPING* (LANJUTAN)

Algoritma perulangan (*looping*) merupakan algoritma yang digunakan untuk mengulang satu atau sekelompok perintah berkali-kali agar memperoleh hasil yang diinginkan, selama kondisi yang dijadikan acuan terpenuhi atau bernilai benar. Pada perulangan terbagi menjadi dua yaitu perulangan naik dan menurun. Perulangan naik adalah perulangan yang dimulai dari nilai terkecil menuju nilai terbesar, pun sebaliknya dengan perulangan menurun yaitu dari terbesar menuju terkecil. Jenis perulangan pada pemrograman C++ ada tiga yaitu *for*, *while* dan *do while*.

Contoh Program:

Tampilan *output* program sederhana menampilkan nama sebanyak 100 kali:



```
D:\01. DOCUMENTS\
93. Miftahul Haq
94. Miftahul Haq
95. Miftahul Haq
96. Miftahul Haq
97. Miftahul Haq
98. Miftahul Haq
99. Miftahul Haq
100. Miftahul Haq
```

1. Contoh program menampilkan nama menggunakan perulangan *for*:

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

void main()
{
    int a;
    cout << "PROGRAM MENAMPILKAN NAMA 100 KALI:\n";
    cout << "MENGUNAKAN FOR\n";
    for (a = 1; a <= 100; a++)
    {
```

```

        cout << a << ". Miftahul Haq\n";
    }
    getche();
}

```

2. Contoh program menampilkan nama menggunakan perulangan *while*:

```

#include <iostream.h>
#include <conio.h>

void main()
{
    int a=1;
    cout << "PROGRAM MENAMPILKAN NAMA 100 KALI:\n";
    cout << "MENGUNAKAN WHILE\n";
    while (a <= 100)
    {
        cout << a << ". Miftahul Haq\n";
        a++; //increment
    }
    getche();
}

```

3. Contoh program menampilkan nama menggunakan perulangan *do while*:

```

#include <iostream.h>
#include <conio.h>

void main()
{
    int a=1;
    cout << "PROGRAM MENAMPILKAN NAMA 100 KALI:\n";
    cout << "MENGUNAKAN DO WHILE\n";
    do
    {
        cout << a << ". Miftahul Haq\n";
        a++; //increment
    }
    while (a <= 100);
    getche();
}

```

Selanjutnya adalah perulangan lanjutan, dimana pada perulangan ini menggabungkan beberapa logika perulangan sehingga dapat menampilkan/ memberikan *output* sesuai dengan yang diinginkan.

Contoh Program:

4. Program menampilkan bilangan 0 9 25 49 81:

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

void main()
{
    clrscr();
    cout << "MENAMPILKAN ANGKA DENGAN LOOPING\n";
    int i, b = 0, a, bil;
    int x = 1;

    for (i = 1; i <= 5; i++)
    {
        cout << b << " ";
        x = x + 2;
        b = x * x;
    }
    getch();
}
```

Output dari program menampilkan bilangan di atas:



5. Program menampilkan bintang segitiga:

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

void main()
{
    clrscr();
    cout<< "Program Menampilkan Bintang Segitiga: \n";
    int a, b;
    for (a=1; a<=4; a++) {
        for (b=1; b<=a; b++) {
            cout<< "*";
        }
    }
}
```

```

        cout<< "\n";
    }

    for (a=1; a<=5; a++) {
        for (b=5; b>=a; b--) {
            cout<< "*";
        }
        cout<< "\n";
    }
    getch();
}

```

Output dari program menampilkan bintang di atas:

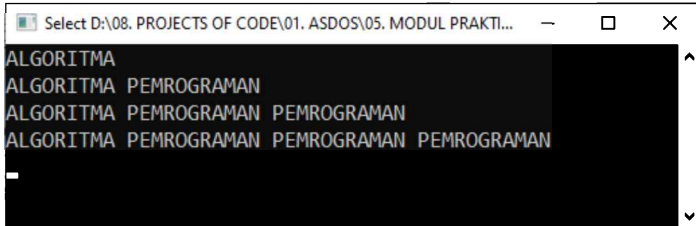
```

D:\01. DOCUMENTS
Program Menampilkan Bintang:
*
**
***
****
*****
****
***
**
*

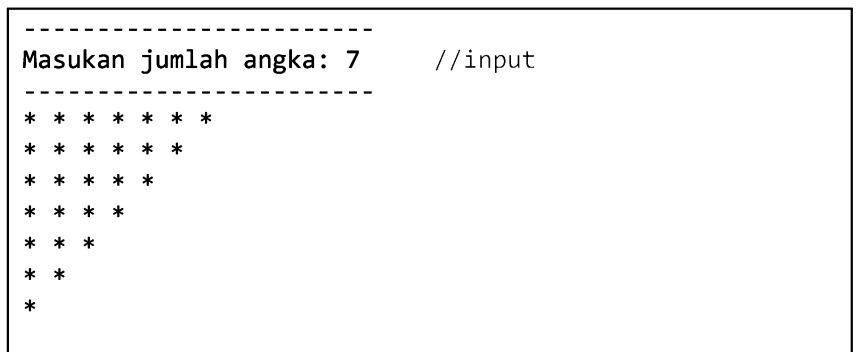
```

Soal Latihan:

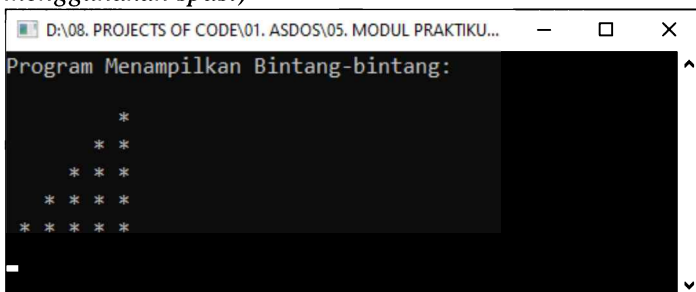
1. Buatlah program sederhana menggunakan algoritma perulangan untuk menampilkan *output*: **0 27 125 343 729**
2. Buatlah program dengan tampilan *output* sebagai berikut! (*menggunakan algoritma perulangan*)



3. Buatlah program sederhana menggunakan algoritma perulangan lanjut dengan sebuah *input* angka yaitu '7' dan menampilkan *output* sebagai berikut! (*jarak antar bintang menggunakan spasi*)



4. Buatlah program sederhana menggunakan algoritma perulangan lanjut dengan tampilan *output* sebagai berikut! (*jarak antar bintang menggunakan spasi*)



PRAKTIKUM KETIGA

Soal Praktikum:

1. (Soal Kasus) Buatlah program dengan tampilan sebagai berikut:

```
*****
SISTEM PERHITUNGAN BIAYA RS. AISYAH
*****
Pilih Kode Dokter;
[A] dr. Nilawati Sp.M   [C] dr. Ridwan Basri Sp.PD
[B] dr. Aria Sukma     [D] dr. Siska Selvia Sp.OG

Proses Rincian Biaya;
Masukkan Kode Dokter   : _____ //input
Masukkan Jumlah Obat   : _____ //input

Biaya Periksa dan Obat  : Rp_____
Potongan Diskon         : Rp_____
Total Bayar            : Rp_____

Jumlah Uang             : Rp_____ //input
Kembali                 : Rp_____
```

Ketentuan:

Biaya pemeriksaan dr. Nilawati = Rp. 175.000
Biaya pemeriksaan dr. Aria S. = Rp. 150.000
Biaya pemeriksaan dr. Ridwan = Rp. 225.000
Biaya pemeriksaan dr. Siska S. = Rp. 250.000
Harga per/obat = Rp. 75.000

Jika jumlah obat 1-3, maka diskon 0%
Jika jumlah obat 4-6, maka diskon 10%
Jika jumlah obat lebih dari 6, maka diskon 15%

Rumus:

$harga_obat = 75000 * jumlah_obat$
 $potongan_diskon = (biaya_periksa + harga_obat) * persen_diskon$
 $total_bayar = (biaya_periksa + harga_obat) - potongan_diskon$

2. Buatlah program sederhana menggunakan algoritma *looping* (perulangan) dengan sebuah *input* angka yaitu '3' dan menampilkan *output* berikut:

```
Masukkan jumlah angka: 3 //input
DASAR
DASAR DASAR
DASAR DASAR DASAR
DASAR DASAR DASAR PEMROGRAMAN PEMROGRAMAN
DASAR DASAR DASAR
DASAR DASAR
DASAR
```

3. Tuliskan *output* dari program sederhana berikut, jika dilakukan *input* yaitu angka '7' !

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int i, nilai;
    cout<< "Silakan masukan nilai i : ";
    cin>> nilai;
    if (nilai %2 != 0) { //operasi modulus
        nilai--;
    }
    for (i=nilai; i>=0; i-=2) {
        cout<< i<< " ";
    }
    getch();
}
```

BAGIAN 14

PEMROGRAMAN MODULAR (LANJUTAN) DAN REKURSIF

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu memahami serta mengembangkan cara berpikir menggunakan algoritma pemrograman modular (lanjutan) yang terdiri dari prosedur, fungsi dan rekursif.

Teori:

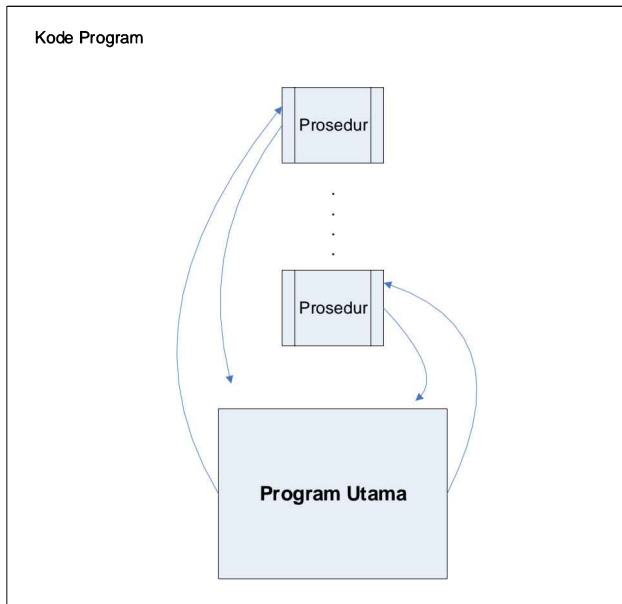
A. PEMROGRAMAN MODULAR

Pemrograman modular adalah suatu teknik pemrograman dimana program yang biasanya cukup besar dibagi-bagi menjadi beberapa bagian program yang lebih kecil sehingga akan mudah dipahami dan dapat digunakan kembali, baik untuk program itu sendiri maupun program lain yang memiliki proses yang sama.

Pemrograman modular dapat diklasifikasikan menjadi dua jenis, yaitu pemrograman prosedur (*procedure*) dan fungsi (*function*).

1. Pemrograman Prosedur (*Procedure*)

Pemrograman prosedur adalah sebuah blok program tersendiri, yang merupakan bagian dari program lain yang lebih besar.



Contoh Program:

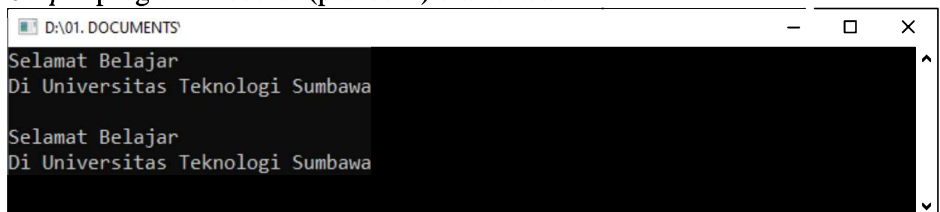
Program Modular (Prosedur)

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

void info_program(); //deklarasi prosedur
void main() {
    info_program();
    getch();
    cout << "\n\n";
    info_program();
    getch();
}

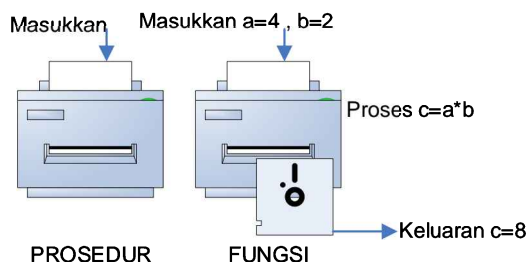
void info_program() { //program prosedur
    cout<< "Selamat Belajar\n";
    cout<< "Di Universitas Teknologi Sumbawa";
}
```

Output program modular (prosedur) diatas adalah:



2. Pemrograman Fungsi (*Function*)

Pemrograman fungsi adalah sebuah blok program tersendiri yang merupakan bagian dari program lain yang lebih besar, sama halnya dengan prosedur hanya saja pada program fungsi memiliki hasil keluaran.



Contoh Program:

Program Modular (Fungsi)

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

int kuadrat(int b); //deklarasi fungsi
void main()
{
    cout << "PROGRAM MENGGUNAKAN FUNGSI \n";
    cout << "kuadrat 2 adalah : " << kuadrat(2);
    cout << "\nkuadrat 3 adalah : " << kuadrat(3);
    cout << "\nkuadrat 4 adalah : " << kuadrat(4);
    getch();
}

int kuadrat(int b) //program fungsi
{
    int z;
    z = b * b;
    return z; //mengembalikan nilai
}
```

Output dari program modular (fungsi) diatas adalah:



B. PEMROGRAMAN REKURSIF

Pemrograman rekursif merupakan proses memanggil dirinya sendiri yang biasa dilakukan oleh fungsi atau prosedur pada pemrograman modular. Rekursif akan terus berjalan sampai kondisi berhenti terpenuhi, mirip dengan perulangan/*looping*. Ada beberapa masalah yang akan lebih mudah jika dipecahkan menggunakan fungsi rekursif. Disamping itu kode program yang menggunakan fungsi rekursif akan lebih mudah dipahami (lebih ringkas) dari pada versi iterasinya yang menggunakan metode perulangan/*looping*.

Contoh Program:

Program rekursif perpangkatan:

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

```

int pangkat(int j, int k); //deklarasi rekursif
void main()
{
    int x, a, hasil;
    cout << "Program Rekursif Perpangkatan \n" ;
    cout << "Masukkan nilai x = ";
    cin >> x;
    cout << "Masukkan nilai a = "; cin >> a ;
    cout << "Hasil pemangkatan " << x << "^" << a << " = "
    << pangkat(x, a);
    getch();
}

int pangkat(int j, int k) //program rekursif
{
    int i, hasil = 1;
    for (i = 1; i <= k; i++)
        hasil = hasil * j;
    return hasil;
}

```

Output dari program rekursif perpangkatan diatas adalah:

```

D:\01. DOCUMENTS\
PROGRAM REKURSIF PERPANGKATAN
Masukkan nilai x = 5
Masukkan nilai a = 3
Hasil pemangkatan 5^3 = 125

```

Contoh Program dari Pengembangan Modular:

1. Program Menentukan Nilai Faktorial dengan Pemrograman Rekursif

Faktorial dari bilangan asli n adalah hasil perkalian antara bilangan bulat positif yang kurang dari atau sama dengan n . Faktorial ditulis sebagai $n!$ dan disebut n faktorial, tanda (!) disebut dengan notasi faktorial.

Jika n bilangan asli maka n faktorial ($n!$) didefinisikan dengan:

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \dots \times 3 \times 2 \times 1$$

Dari definisi itu, maka kita juga memperoleh:

$$n! = n(n-1)!$$

Contoh nilai faktorial 1-10:

$$0! = 1$$

$$1! = 1$$

$$2! = 1 \times 2 = 2$$

$$3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24$$

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

$$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720$$

$$7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040$$

$$8! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 = 40320$$

$$9! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 = 362880$$

$$10! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 10 = 3628800$$

Program rekursif untuk penghitungan faktorial:

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

int faktor(int n); //deklarasi rekursif
void main()
{
    clrscr();
    int n;
    cout << "MENENTUKAN NILAI FAKTORIAL \n";
    cout << "Input Nilai : ";
    cin >> n;
    cout << "Faktorial {" << n << "} = " << faktor(n) <<
    "\n";
    getch();
}

int faktor(int n) // program rekursif
{
    int i, x = 1;
    for (i=1; i<=n; i++)
    {
        x=x*i;
    }
    return x;
}
```

Output dari program rekursif untuk penghitungan faktorial diatas adalah:

```

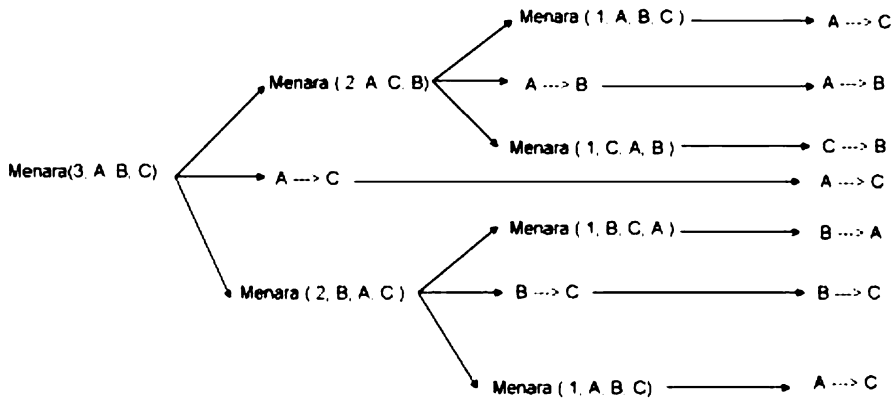
D:\01. DOCUMENTS
MENENTUKAN NILAI FAKTORIAL
Input Nilai : 6
Faktorial {6!} = 720

```

2. Program Permainan Menara Hanoi dengan Pemrograman Rekursif

Permainan menara hanoi merupakan sebuah permainan dengan konsep mindah memindahkan beberapa piringan disk dari menara 1 ke menara yang lain, tanpa merubah bentuk atau urutan dari disk tersebut. dalam permainan menara hanoi terdapat 3 menara dan beberapa disk.

Logika Algoritma Menara Hanoi adalah sebagai berikut:



Program rekursif untuk program Permainan Menara Hanoi:

```

#include <iostream.h>
#include <conio.h>
void menarahanoi(int n, char a, char b, char c);
void main()
{
    clrscr(); int n;
    cout << "PROGRAM MENARA HANOI";
    cout << "\n-----\n";
    cout << "\nMasukan Banyak Piringan : ";
    cin >> n;
    cout << "\nLangkah yang dilakukana adalah : \n";
    menarahanoi(n, 'A', 'B', 'C'); //manggil program
    getch();
}

```

```

void menarahanoi(int n, char a, char b, char c) //program
menara hanoi
{
    if (n == 1)
        cout << "Pindahkan piringan dari " << a << " ke " <<
c << endl;
    else
    {
        menarahanoi(n-1, a, c, b);
        menarahanoi(1, a, b, c);
        menarahanoi(n-1, b, a, c);
    }
}

```

Output dari program permainan Menara hanoi diatas adalah:

```

D:\01. DOCUMENTS\
PROGRAM MENARA HANOI
-----
Masukan Banyak Piringan : 3

Langkah yang dilakukana adalah :
Pindahkan piringan dari A ke C
Pindahkan piringan dari A ke B
Pindahkan piringan dari C ke B
Pindahkan piringan dari A ke C
Pindahkan piringan dari B ke A
Pindahkan piringan dari B ke C
Pindahkan piringan dari A ke C

```

3. Program Konversi Suhu

Program modular konversi suhu:

```

#include <iostream.h>
#include <conio.h>
// deklarasi program modular
void fahrenheit(float carifar, float c2);
void reamur(float carirea, float c2);
void kelvin(float cerikel, float c2);

// program utama
void main()
{
    clrscr();
    int hasil, c;
    cout << " PROGRAM KONVERSI SUHU\n";
    cout << " Masukkan suhu dalam celcius : ";

```

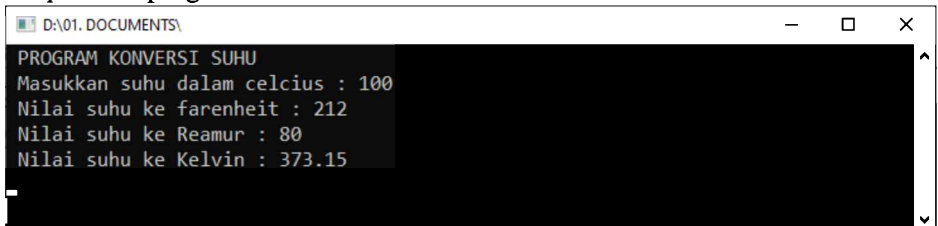


```

        cin >> c;
        fahrenheit(hasil, c);
        cout << "\n";
        reamur(hasil, c);
        cout << "\n";
        kelvin(hasil, c);
        cout << "\n";
        getch();
    }
    // program konversi ke fahrenheit
    void fahrenheit(float cari_far, float c2) {
        cari_far = c2 * 1.8 + 32;
        cout << " Nilai suhu ke Farenheit : " << cari_far;
    }
    // program konversi ke reamur
    void reamur(float cari_rea, float c2) {
        cari_rea = c2 * 0.8;
        cout << " Nilai suhu ke Reamur : " << cari_rea;
    }
    // program konversi ke kelvin
    void kelvin(float cari_kel, float c2) {
        cari_kel = c2 + 273.15;
        cout << " Nilai suhu ke Kelvin : " << cari_kel;
    }
}

```

Output dari program modular konversi suhu diatas adalah:



```

D:\01. DOCUMENTS\
PROGRAM KONVERSI SUHU
Masukkan suhu dalam celcius : 100
Nilai suhu ke fahrenheit : 212
Nilai suhu ke Reamur : 80
Nilai suhu ke Kelvin : 373.15

```

4. Program Bintang Belah Ketupat

Program bintang belah ketupat dengan modular:

```

#include <iostream.h>
#include <conio.h>
//deklarasi program modular
int bintang_atas(int bil);
int bintang_bawah(int bil);

//program utama
void main()
{

```

```

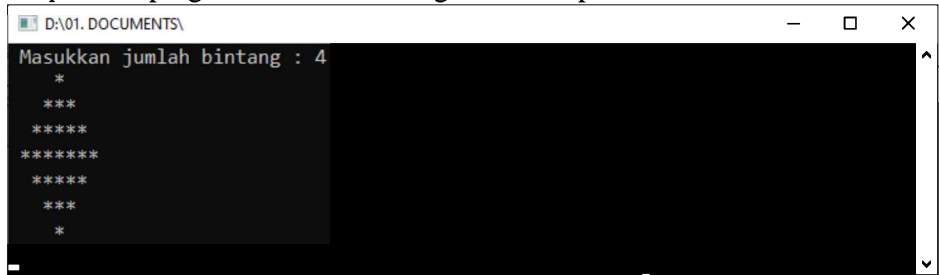
    clrscr();
    int bil;
    cout << " Masukkan jumlah bintang : ";
    cin >> bil;
    bintang_atas(bil);
    bintang_bawah(bil);
    getch();
}

//program modular atas
int bintang_atas(int bil)
{
    int a, b;
    for (a=1; a<=bil; a++) {
        for (b=bil; b>=a; b--)
            cout << " ";
        for (b=a; b<2*a; b++) {
            cout << "*";
        }
        for (b=2*(a-1); b>=a; b--) {
            cout << "*";
        }
        cout << "\n";
    }
}

//program modular bawah
int bintang_bawah(int bil)
{
    int a, b;
    for (a=bil-1; a>=1; a--) {
        for (b=a; b<=bil; b++)
            cout << " ";
        for (b=2*a; b>a; b--) {
            cout << "*";
        }
        for (b>a; b<=2*(a-1); b++) {
            cout << "*";
        }
        cout << "\n";
    }
}

```

Output dari program modular bintang belah ketupat diatas adalah:



```
D:\01. DOCUMENTS\
Masukkan jumlah bintang : 4
 *
 ***
 *****
 *****
 *****
 ***
 *
```

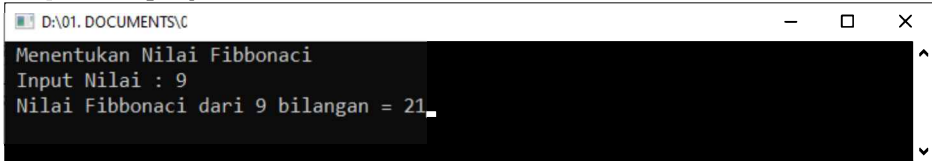
5. Program Menentukan Nilai Fibonacci

Program menentukan nilai fibonacci dengan modular:

```
#include <iostream.h>
#include <conio.h>
int fibo(int n);
void main()
{
    clrscr();
    int n;
    cout << " Menentukan Nilai Fibbonaci\n";
    cout << " Input Nilai : ";
    cin >> n;
    cout << " Nilai Fibbonaci dari " << n << " bilangan = "
    << fibo(n);
    getch();
}

int fibo(int n) //program modular fibo
{
    int i, next, a=0, b=1;
    for (i=0; i<n; i++)
    {
        if (i<=1) {
            next = 1;
        }
        else {
            next = a+b;
            a = b;
            b = next;
        }
    }
}
```

Output dari program modular menentukan nilai fibonacci diatas adalah:



```
D:\01. DOCUMENTS\>
Menentukan Nilai Fibonacci
Input Nilai : 9
Nilai Fibonacci dari 9 bilangan = 21
```

6. Program Deret Fibonacci

Program deret fibonacci:

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    int i, n, a[100];
    cout << "----- PROGRAM DERET FIBONACCI -----\\n";
    cout << "Masukkan jumlah deret : ";
    cin >> n;
    for (i=0; i<n; i++) {
        if (i<2)
            a[i] = i;
        else
            a[i] = a[i-2] + a[i-1];
    }

    cout << "Fibonacci " << n << " deret adalah \\n";
    for (i=0; i<n; i++) {
        cout << a[i] <<" ";
    }
    getch();
}
```

Output dari program deret fibonacci diatas adalah:



```
D:\01. DOCUMENTS\>
----- PROGRAM DERET FIBONACCI -----
Masukkan jumlah deret : 8
Fibonacci 8 deret adalah
0 1 1 2 3 5 8 13
```

BAGIAN 15

PEMROGRAMAN ARRAY (LANJUTAN)

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami konsep *array* dan penyimpanan dalam *array*
2. Menguasai penggunaan variabel *array* berdimensi satu
3. Memahami penggunaan variabel *array* berdimensi dua
4. Mampu mengembangkan konsep pemrograman *array* pada program sederhana.

Teori:

Elemen-elemen *array* dapat diakses oleh program menggunakan suatu indeks tertentu secara random ataupun berurutan. Pengisian dan pengambilan nilai pada indeks tertentu dapat dilakukan dengan mengeset nilai atau menampilkan nilai pada indeks yang dimaksud. Dalam C++, tidak terdapat *error handling* terhadap batasan nilai indeks, apakah indeks tersebut berada di dalam indeks *array* yang sudah didefinisikan atau belum. Hal ini merupakan tanggung jawab programmer, sehingga jika programmer mengakses indeks yang salah, maka nilai yang dihasilkan akan berbeda atau rusak karena mengakses alamat memori yang tidak sesuai. *Array* terbagi dalam dua jenis yaitu *array* satu dimensi dan dua dimensi atau multi dimensi.

A. ARRAY SATU DIMENSI

Pada *array* satu dimensi hanya terdapat satu baris dan banyak kolom. Deklarasi *Array* satu dimensi secara umum:

```
tipe_data nama_var_array [ukuran];
```

Keterangan:

tipe_data	: Menyatakan jenis tipe data elemen larik (int, char, float, dll)
nama_var_array	: Menyatakan nama variabel yang dipakai
ukuran	: Menunjukkan jumlah maksimal elemen larik.

Dalam menginisialisasi *array* sama dengan memberikan nilai awal *array* pada saat didefinisikan, dengan cara sebagai berikut:

```
int nilai[5] = {70,80,82,60,75};
```

Keterangan:

	Nilai[0]	Nilai[1]	Nilai[2]	Nilai[3]	Nilai[4]
int Nilai[5]:	70	80	82	60	75

Contoh diatas berarti anda memesan tempat di memori komputer sebanyak 5 tempat dengan indeks dari 0-5, dimana indeks ke-0 bernilai 70, ke-1 bernilai 80, ke-2 bernilai 82, dan seterusnya. Semua elemennya bertipe data integer.

0	1	2	3	4	5	6	7	Indeks
10	44	2	76	0	56	70	7	value
1d2	1d4	1d6	1d8	1da	2dc	2de	1e0	alamat

Tiga hal penting dalam array yang harus dipahami yaitu:

1. *Index*, bisa disebut juga dengan urutan, index pada array secara default dimulaia dari 0 (nol).
2. *Value*, adalah nilai yang mengisi di setiap elemen array, dan elemen-elemen ini dapat diakses oleh program menggunakan suatu indeks tertentu secara random ataupun berurutan.
3. *Reference*, adalah alamat memori dari masing-masing elemen array.

B. ARRAY DUA DIMENSI

Array dua dimensi yang sering digambarkan sebagai sebuah matrik dimana array dua dimensi memiliki banyak baris dan banyak kolom. Deklarasi array dua dimensi secara umum:

```
type_data nama_var_array [batas_baris] [batas_kolom];
```

Keterangan:

type_data : Menyatakan jenis tipe data elemen larik (int, char, float, dll)

nama_var_array : Menyatakan nama variabel yang dipakai

batas_baris : Menyatakan jumlah/ukuran baris pada *array*

batas_kolom : Menyatakan jumlah/ukuran kolom pada *array*

Contoh:

```
int nilai[3][2];
```

Keterangan contoh: array dua dimensi dengan nama nilai bertipe data integer dengan jumlah baris 3 dan jumlah kolom 2. Hasil pendefinisian diatas dapat dinyatakan sebagai berikut:

(baris)	(kolom 0)	(kolom 1)
0	nilai(0,0) = 8	nilai(0,1) = -3
1	nilai(1,0) = 9	nilai(1,1) = 0
2	nilai(2,0) = 5	nilai(2,1) = 2

Contoh lain membaca susunan *array* dua dimensi:

- Deklarasi nilai *array*:

```
char huruf[3][5];
```

- Sama dengan matriks berukuran 3x5, dengan 3 baris dan 5 kolom, dan *index* dimulai dari angka 0:

	0	1	2	3	4
0					
1					
2					

- Pada kenyataan di *memory*:

a[0][0]	a[0][4]	a[1][0]	a[1][4]	a[2][0]	a[2][4]
Baris ke 0		Baris ke 1		Baris ke 2	

Contoh Program:

1. Program dengan *array* satu dimensi:

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

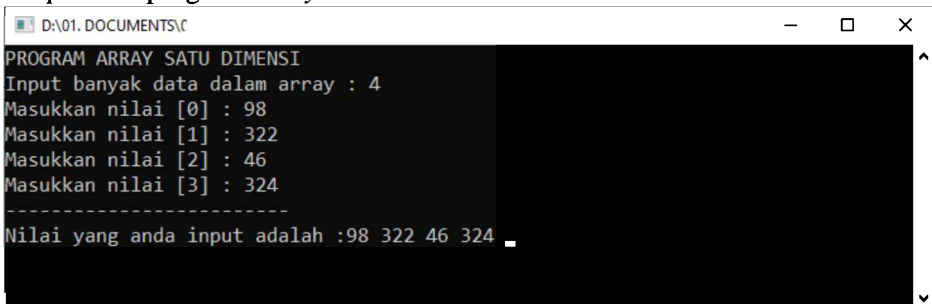
void main()
{
    int nilai['n'];
    int n, a;
    cout << "PROGRAM ARRAY SATU DIMENSI\n";
    cout << "Input banyak data dalam array : ";
    cin >> n;
```

```

for (a=0; a<n; a++) {
    cout << "Masukkan nilai [" << a << "]" : ";
    cin >> nilai[a];
}
cout << "-----\n";
cout << "Nilai yang anda input adalah : ";
for (a=0; a<n; a++) {
    cout << nilai[a] << " ";
}
getche();
}

```

Output dari program *array* satu dimensi diatas adalah:



```

D:\01. DOCUMENTS\C
PROGRAM ARRAY SATU DIMENSI
Input banyak data dalam array : 4
Masukkan nilai [0] : 98
Masukkan nilai [1] : 322
Masukkan nilai [2] : 46
Masukkan nilai [3] : 324
-----
Nilai yang anda input adalah :98 322 46 324

```

2. Program dengan *array* dua dimensi:

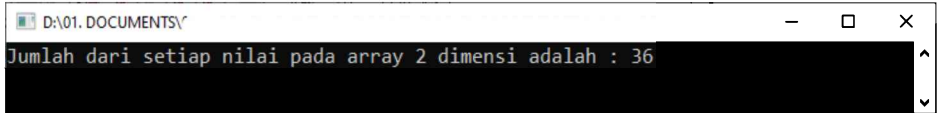
```

#include <iostream.h>
#include <conio.h>

void main()
{
    clrscr();
    int matriks[2][4] = {{4,3,2,1},{5,6,7,8}};
    int baris, kolom, jumlah=0;
    for (baris=0; baris<2; baris+=1) {
        for (kolom=0; kolom<4; kolom+=1) {
            jumlah += matriks[baris][kolom];
        }
    }
    cout << "Jumlah dari setiap nilai pada array 2 dimensi
adalah : "<<jumlah;
    getche();
}

```


Output dari program *array* dua dimensi diatas adalah:



```
D:\01. DOCUMENTS'
Jumlah dari setiap nilai pada array 2 dimensi adalah : 36
```

3. Program dengan *array* dua dimensi dan tabel visual:

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h> //library untuk tabel

void main()
{
    clrscr();
    int i, j;
    int data_jual[4][4];
    cout << " DATA PENJUALAN PT. ASTRA \n";
    for (i=1; i<=3; i++)
    {
        for (j=1; j <= 3; j++)
        {
            cout << " Data ke " << i << "." << j;
            cout << " Jumlah Penjualan : ";
            cin >> data_jual[i][j];
        }
    }

    cout << " \n Data penjualan pertahun \n";
    cout << " ~~~~~~\n";
    cout << " No.  2010   2011   2012 \n";
    cout << " ~~~~~~\n";
    for (i=0; i<4; i++)
    {
        cout << setiosflags(ios::left) << setw(3);
        cout << " " << i;
        for (j=1; j<=3; j++)
        {
            cout << setiosflags(ios::right) << setw(7);
            cout << data_jual[i][j];
        }
        cout << "\n";
    }
    cout << " ~~~~~~";
    getch();
}
```

Output dari program *array* dua dimensi dan tabel visual diatas adalah:

```
D:\01. DOCUMENTS\
DATA PENJUALAN PT. ASTRA
Data ke 1.1 Jumlah Penjualan : 78
Data ke 1.2 Jumlah Penjualan : 447
Data ke 1.3 Jumlah Penjualan : 34
Data ke 2.1 Jumlah Penjualan : 68
Data ke 2.2 Jumlah Penjualan : 438
Data ke 2.3 Jumlah Penjualan : 8645
Data ke 3.1 Jumlah Penjualan : 35
Data ke 3.2 Jumlah Penjualan : 67
Data ke 3.3 Jumlah Penjualan : 45

Data penjualan pertahun
~~~~~
No.  2010  2011  2012
-----
1    78    447   34
2    68    438  8645
3    35    67   45
```

Soal Latihan:

1. Program sederhana berikut merupakan program yang mengimplementasikan *array* dua dimensi. Tuliskan *output* dari program dibawah!

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

void main()
{
    clrscr();
    int matriks_a[2][4] = {{30, 40, 50, 60}, {10, 20, 30, 40}};
    int matriks_b[2][4] = {{6, 7, 8, 9}, {3, 4, 5, 6}};
    int a, aa, b, bb, d, dd, jumlah=0;

    cout << "--- MATRIKS A ---\n";
    for (a=0; a<2; a++) {
        for (aa=0; aa<4; aa++) {
            cout << matriks_a[a][aa] << " ";
        }
        cout << "\n";
    }

    cout << "\n\n--- MATRIKS B ---\n";
```

```

    for (b=0; b<2; b++) {
        for (bb=0; bb<4; bb++) {
            cout << matriks_b[b][bb] << " ";
        }
        cout << "\n";
    }

    cout << "\n\n--- PENJUMLAHAN MATRIKS ---\n";
    for (d=0; d<2; d++) {
        for (dd=0; dd<4; dd++) {
            jumlah = matriks_a[d][dd] +
matriks_b[d][dd];
            cout << matriks_a[d][dd] << "+" <<
matriks_b[d][dd] << "=" << jumlah << " ";
        }
        cout << "\n";
    }
    getch();
}

```

BAGIAN 16

PEMROGRAMAN *STRUCT*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu memahami serta mengembangkan cara berpikir menggunakan algoritma pemrograman struktur (*struct*).

Teori:

Pemrograman struktur (*struct*) merupakan kumpulan elemen data yang digabungkan menjadi satu kesatuan data. Juga merupakan data yang dapat menyimpan variabel-variabel dalam 1 nama, dapat memiliki tipe data yang berbeda ataupun sama. Variabel-variabel tersebut memiliki kaitan satu sama yang lain. Struktur secara logik membuat suatu tipe data baru yang dapat dipergunakan untuk menampung informasi/data yang bersifat majemuk. Bentuk umum dari struktur adalah sebagai berikut:

```
typedef struct {  
    type_data1 element1;    //anggota elemen dari struktur  
    type_data2 element2;    //  
} nama_struktur;           //nama struk  
nama_struk nama_objek;     //objek struk
```

Deklarasi *struct* bisa dengan dua cara. Berikut adalah contoh masing-masing:

1. Deklarasi Pertama

```
struct {  
    char nim[8];    //  
    char nama[50]; //anggota elemen dari struktur  
    float ipk;      //  
} mhs;              //nama+objek struk
```

2. Deklarasi Kedua

```
typedef struct {  
    char nim[8];    //  
    char nama[50]; //anggota elemen dari struktur  
    float ipk;      //  
} mahasiswa;       //nama struk  
mahasiswa mhs;     //objek struk
```

Contoh Program:

1. Program Menulis Nama dengan *Struct*

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    struct {
        char nama[40];
        short umur;
    } saya; //nama+objek struk

    cout << " Nama : ";
    cin >> saya.nama;
    cout << " Umur : ";
    cin >> saya.umur;
    cout << " \n -- DATA --\n";
    cout << saya.nama << "\n";
    cout << saya.umur;
    getch();
}
```

Output program *struct* diatas adalah:



```
D:\01. DOCUMENTS\
Nama : Miftahul
Umur : 21

-- DATA --
Miftahul
21
```

2. Program Menulis Stok Barang dengan *Struct*

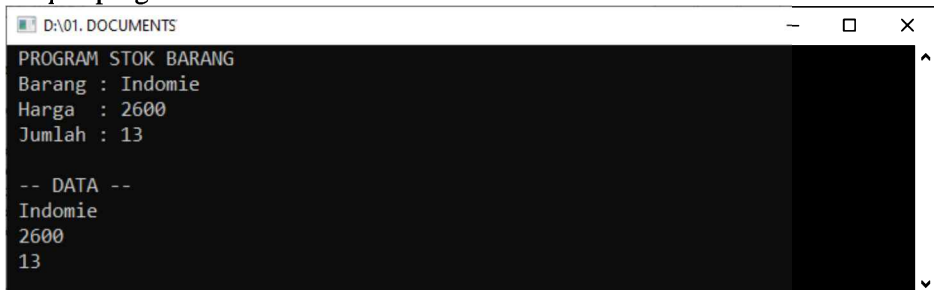
```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    cout << " PROGRAM STOK BARANG\n";
    typedef struct {
        char barang[40];
        int harga;
        short jumlah;
    } stok; //nama struk
    stok persediaan; //objek struk
```

```

    cout << " Barang : ";
    cin >> persediaan.barang;
    cout << " Harga : ";
    cin >> persediaan.harga;
    cout << " Jumlah : ";
    cin >> persediaan.jumlah;
    cout << " \n -- DATA --\n ";
    cout << persediaan.barang << "\n ";
    cout << persediaan.harga << "\n ";
    cout << persediaan.jumlah;
    getch();
}

```

Output program struct diatas adalah:



```

D:\01. DOCUMENTS
PROGRAM STOK BARANG
Barang : Indomie
Harga : 2600
Jumlah : 13

-- DATA --
Indomie
2600
13

```

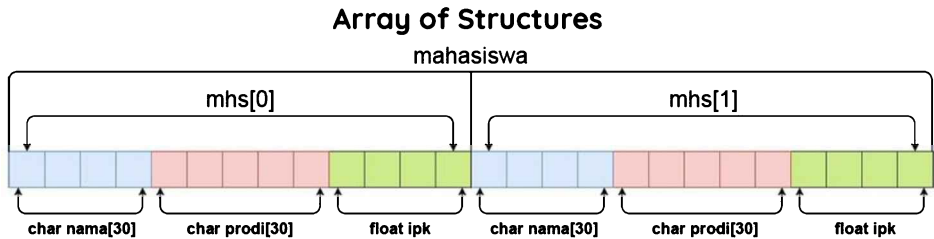
BAGIAN 17

PEMROGRAMAN *ARRAY OF STRUCT*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu memahami serta mengembangkan cara berpikir menggunakan algoritma pemrograman array dalam array yang terstruktur (*array of struct*).

Teori:



Array struktur di C++ dapat didefinisikan sebagai kumpulan beberapa variabel struktur di mana setiap variabel berisi informasi tentang entitas yang berbeda. Digunakan untuk menyimpan informasi tentang banyak entitas dari tipe data yang berbeda. *Array of struct* juga dikenal sebagai kumpulan dari struktur. Contoh deklarasi dari *array of struct* adalah sebagai berikut:

```
typedef struct {  
    char nama[30]; //  
    char prodi[30]; // elemen dari struktur mahasiswa  
    float ipk; //  
} mahasiswa; //nama struktur  
  
mahasiswa mhs[10]; //array of struct
```

Artinya *struct* mahasiswa digunakan dalam *array* untuk *struct* mhs[0], mhs[1], mhs[2], mhs[3], dan seterusnya hingga mhs[9].

Contoh Program:

1. Program Sederhana dengan *Array of Structs*

```
#include <iostream.h>  
#include <conio.h>  
  
typedef struct { //deklarasi struct diluar program utama  
    char nama[30];
```

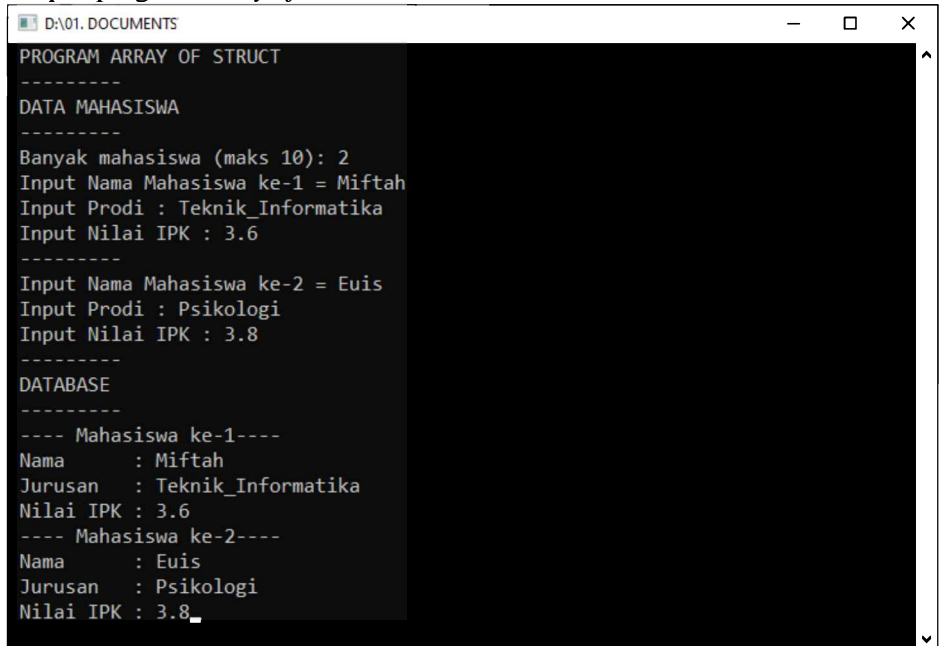
```

    char prodi[30];
    float ipk;
} mahasiswa;

void main() //program utama
{
    clrscr();
    mahasiswa mhs[10]; //objek struk
    int i, banyak;
    cout << " PROGRAM ARRAY OF STRUCT\n";
    cout << " ----- \n";
    cout << " DATA MAHASISWA\n";
    cout << " ----- \n";
    cout << " Banyak mahasiswa (maks 10): ";
    cin >> banyak;
    for (i=0; i<banyak; i++)
    {
        cout << " Input Nama Mahasiswa ke-" << i+1 << " = ";
        cin >> mhs[i].nama;
        cout << " Input Prodi : ";
        cin >> mhs[i].prodi;
        cout << " Input Nilai IPK : ";
        cin >> mhs[i].ipk;
        cout << " ----- \n";
    }
    cout << " DATABASE\n";
    cout << " -----";
    for (i=0; i<banyak; i++)
    {
        cout << "\n ---- Mahasiswa ke-" << i+1 << "----";
        cout << "\n Nama      : " << mhs[i].nama;
        cout << "\n Jurusan   : " << mhs[i].prodi;
        cout << "\n Nilai IPK  : " << mhs[i].ipk;
    }
    getch();
}

```


Output program array of struct diatas adalah:



```
PROGRAM ARRAY OF STRUCT
-----
DATA MAHASISWA
-----
Banyak mahasiswa (maks 10): 2
Input Nama Mahasiswa ke-1 = Miftah
Input Prodi : Teknik_Informatika
Input Nilai IPK : 3.6
-----
Input Nama Mahasiswa ke-2 = Euis
Input Prodi : Psikologi
Input Nilai IPK : 3.8
-----
DATABASE
-----
---- Mahasiswa ke-1----
Nama      : Miftah
Jurusan   : Teknik_Informatika
Nilai IPK : 3.6
---- Mahasiswa ke-2----
Nama      : Euis
Jurusan   : Psikologi
Nilai IPK : 3.8_
```

BAGIAN 18

PEMROGRAMAN *SEARCHING*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami dan menguasai konsep algoritma *searching* data pada *array*
2. Menerapkan algoritma *searching* pada program sederhana dari setiap metodenya (*sequential* dan *binary*).

Teori:

Pemrograman *Searching* yaitu merupakan metode pencarian data dengan membandingkan setiap elemen larik satu per satu secara urut (beruntun), mulai dari elemen pertama sampai dengan elemen yang terakhir. Ada 2 macam pencarian beruntun, yaitu pencarian pada *array* yang sudah terurut, dan pencarian pada *array* yang belum terurut. Dalam materi ini akan dijelaskan 2 metode pencarian yaitu pencarian dengan metode *Sequential search* (pencarian beruntun) dan *Binary search* (pencarian bagi dua).

A. *SEQUENTIAL SEARCH* (PENCARIAN BERUNTUN)

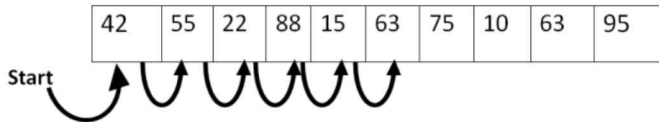
Sequential Search (pencarian beruntun) merupakan metode pencarian data dalam *array* dengan cara membandingkan data yang dicari dengan data yang ada di dalam *array* secara berurutan. Pencarian data dengan metode *sequential search* efektif untuk mencari data yang dalam posisi tidak terurut atau acak.

Prosesnya *sequential search* bisa dijelaskan seperti berikut:

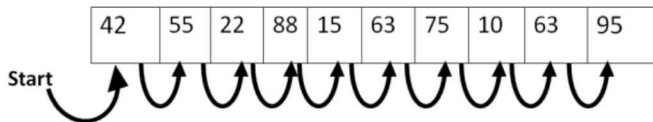
- a. Menentukan data yang dicari
- b. Membaca data *array* satu per satu secara sekuensial
- c. Mulai dari data pertama sampai dengan data terakhir, kemudian data yang dicari tadi dibandingkan dengan masing-masing data yang ada di dalam *array*.
 - Jika data yang dicari ditemukan, maka kita dapat membuat *statement* bahwa data berhasil ditemukan.
 - Jika data yang dicari tidak ditemukan, maka kita dapat membuat *statement* bahwa data tidak berhasil ditemukan.

Contoh prosesnya, nilai pencarian yang diinginkan dibandingkan dengan nilai pertama dalam daftar. Jika nilai yang diperlukan cocok dengan nilai pertama, operasi pencarian dinyatakan berhasil dan dihentikan. Misalkan kita ingin mencari nilai 63 dalam daftar nilai seperti yang ditunjukkan pada gambar di

bawah ini. Nilai 63 ada di lokasi indeks 5 dan 8. Operasi pencarian dihentikan di posisi 5 dan langsung diberikan *statement* bahwa data berhasil ditemukan.



Jika nilai yang diinginkan tidak cocok dengan nilai pertama daftar, itu dibandingkan dengan nilai kedua. Siklus pencarian berlanjut hingga nilai ditemukan atau akhir daftar tercapai. Sekarang asumsikan kita perlu mencari nilai 72 dalam daftar nilai seperti yang muncul pada gambar dibawah. Nilai ini tidak ada dalam daftar. Operasi pencarian berakhir menjelang akhir daftar dan dihentikan serta diberikan *statement* bahwa data tidak berhasil ditemukan.



Pencarian berurutan (*sequential search*) dalam prosesnya sangat lambat dan biasanya hanya digunakan untuk daftar data yang kecil. Metode ini tidak disarankan untuk data dalam jumlah yang besar, karena ada beberapa metode yang lebih efisien tersedia untuk pencarian yang besar dan kompleks.

Contoh Studi Kasus:

Menemukan data yang dicari dalam sebuah array 1 dimensi yang terdapat data di dalamnya dengan menggunakan metode *sequential searching*. Jika data yang dicari dapat ditemukan di dalam *array*, maka kemudian akan ditampilkan letak dari masing-masing indexnya.

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

void main()
{
    //deklarasi variable
    int data[10], index[10], i, j = 0, x, n;
    //proses penginputan data
    cout << " PROGRAM SEQUENTIAL SEARCHING \n";
    cout << " Masukkan jumlah data [max10] : ";
    cin >> n;
    for (i=0; i<n; i++) {
        cout << " Masukkan Data ke - " << (i+1) << " = ";
```

```

        cin >> data[i];
    }
    //memasukkan data yang akan dicari ke dalam x
    cout << " =====\n Masukkan data yang anda akan
cari : ";
    //inisialisasi jumlah data yang dicari sebelum pencarian
    cin >> x;
    //proses pencarian data
    for (i=0; i<n; i++) {
        if (data[i] == x) {
            index[j] = i;
            j++;
        }
    }

    //jika data ditemukan dalam array
    if (j > 0) {
        cout << " Data " << x << " yang dicari ada " << j <<
" buah\n";
        cout << " Data tersebut terdapat pada data-ke = ";
        for (i=0; i<j; i++) {
            cout << "[" << index[i]+1 << " ] ";
        }
        cout << endl;
    }

    //jika tidak ditemukan
    else {
        cout << " Data tidak ditemukan dalam array\n";
    }
    getch();
}

```

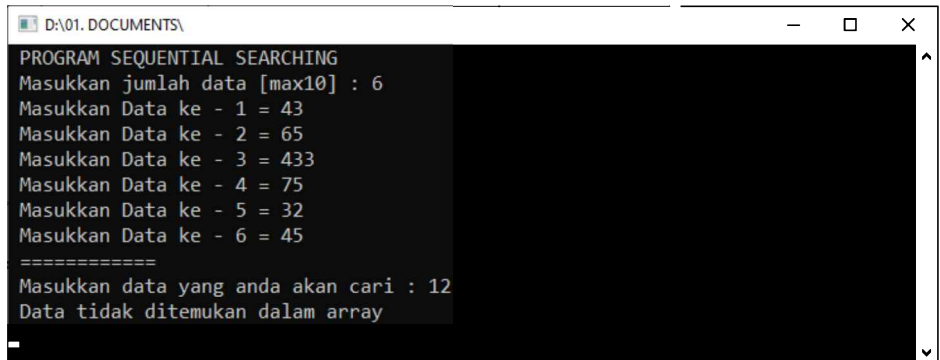
Output dari program sequential search di atas jika pencarian ditemukan:

```

D:\01. DOCUMENTS
PROGRAM SEQUENTIAL SEARCHING
Masukkan jumlah data [max10] : 5
Masukkan Data ke - 1 = 12
Masukkan Data ke - 2 = 45
Masukkan Data ke - 3 = 79
Masukkan Data ke - 4 = 34
Masukkan Data ke - 5 = 12
=====
Masukkan data yang anda akan cari : 12
Data 12 yang dicari ada 2 buah
Data tersebut terdapat pada data-ke = [1] [5]

```

Output dari program *sequential search* di atas jika **pencarian tidak ditemukan**:



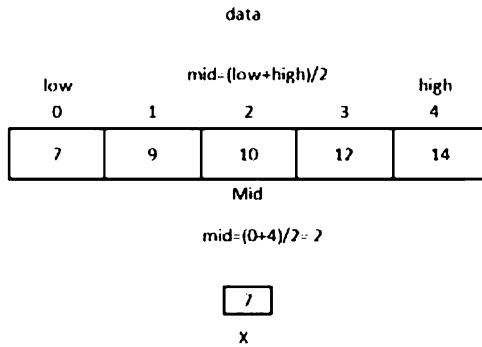
```
D:\01. DOCUMENTS\
PROGRAM SEQUENTIAL SEARCHING
Masukkan jumlah data [max10] : 6
Masukkan Data ke - 1 = 43
Masukkan Data ke - 2 = 65
Masukkan Data ke - 3 = 433
Masukkan Data ke - 4 = 75
Masukkan Data ke - 5 = 32
Masukkan Data ke - 6 = 45
=====
Masukkan data yang anda akan cari : 12
Data tidak ditemukan dalam array
```

B. *BINARY SEARCH* (PENCARIAN BAGI DUA)

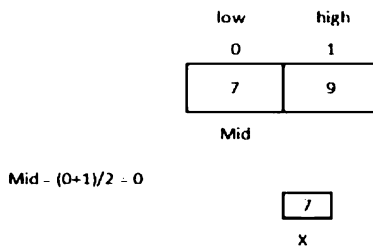
Binary Search (pencarian bagi dua) merupakan metode pencarian yang mencari data dengan melakukan mengelompokkan *array* menjadi bagian-bagian. Pencarian bagi dua ini hanya dapat diimplementasikan pada data yang telah terurut baik *ascending* maupun *descending* dalam suatu *array*.

Proses *Binary Search* yang urutan datanya *ascending*:

1. Pertama buat perulangan lalu menentukan posisi *low* yaitu posisi yang menandakan index paling rendah kemudian menentukan posisi *high*. Kemudian mencari posisi $mid = (high + low)/2$
2. Lalu membandingkan data yang dicari dengan nilai yang ada di posisi *mid*.
3. Jika data yang dicari sama dengan nilai yang ada di posisi *mid* berarti data ditemukan.
4. Jika data yang dicari lebih kecil dari nilai yang ada di posisi *mid* maka pencarian data akan dilakukan dibagian kiri *mid* dengan melakukan perbandingan. dengan kondisi posisi *high* berubah yaitu $(mid - 1)$ dan posisi *low* tetap.
5. Jika data yang dicari lebih besar dari nilai yang ada *mid*, maka pencarian data akan dilakukan di bagian kanan dari *mid* dengan posisi *low* yang berubah yaitu $(mid + 1)$ dan posisi *high* tetap.



Setelah menentukan *low* dan *high* kemudian menentukan *mid*. perhitungan $mid = (low + high) / 2$ jadi $mid = (0 + 4) / 2$, artinya *mid* berada di data[2]. Kemudian data yang dicari yang ditampung di variabel *x* dibandingkan dengan data/nilai yang berada di *mid*. data yang dicari ialah $7 < 10$ kemudian pencarian data dilakukan di bagian kiri *mid*.



Pencarian di bagian kiri *mid* masih dalam perulangan yang sama, namun indeksnyanya yang dipersempit. Karena data yang dicari lebih kecil dari data *mid* yang sebelumnya maka posisi *high* yang berubah yaitu $(mid - 1)$ yang sebelumnya dan *low* tetap pada posisi yang sama. Kemudian menentukan $mid = (0 + 1) / 2 = 0$

Artinya *mid* sekarang terletak di data[0]. lalu data yang dicari dibandingkan dengan *mid*. $7 = 7$ artinya data telah ditemukan

Kelebihan dan kekurangan *binary search*:

1. Kelebihannya yaitu tidak perlu membandingkan data yang dicari dengan seluruh data array yang ada, cukup melalui titik tengah kemudian kita bisa menentukan ke mana selanjutnya mencari data yang ingin dicari.
2. Kekurangan implementasi agak sedikit lebih rumit karena tidak bisa digunakan pada data array yang masih acak. Jadi harus melakukan sorting terlebih dahulu dalam implementasinya.

Contoh Studi Kasus:

Menemukan data dalam sebuah array 1 dimensi, dimana data diinput secara acak ke dalamnya. Dan akan dicari menggunakan metode *binary searching*. Jika data yang dicari ditemukan di dalam *array* kemudian maka program akan memberikan statemen bahwa data yang dicari berhasil ditemukan dalam *array*, dan jika tidak, maka akan statemen berubah bahwa data tidak berhasil ditemukan dalam *array*.

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

void main()
{
    //deklarasi variable
    int data[20], f, n, i, j, x, t, low, high, mid, flag;
    cout << " PROGRAM BINARY SEARCHING \n";
    cout << " Masukkan jumlah data (maks 20): ";
    cin >> n;
    for (i=0; i<n; i++) {
        cout << " Masukkan Data ke - " << (i+1) << " = ";
        cin >> data[i];
    }
    cout << " Masukkan data yang anda akan cari : ";
    cin >> x;
    //proses pengurutan data
    for (i=0; i<n; i++) {
        for (j=i+1; j<n; j++) {
            if (data[i] > data[j]) {
                t = data[i];
                data[i] = data[j];
                data[j] = t;
            }
        }
    }
    cout << " --- DATA SETELAH DI URUTKAN ---\n";
    for (i=0; i<n; i++) {
        cout << " " << data[i] ;
    }
    //proses pencarian data
    flag = 0;
    high = n;
    low = 0;
    cout << "\n --- PROSES PENCARIAN ---\n";
    while (low<=high) {
        mid = (low+high) / 2;
```

```

        cout << "  -> Low: " << low << " -> Mid: " << mid <<
" -> High: " << high; // menampilkan low, mid, high
        if (data[mid] == x) {
            flag++;
        }
        if (data[mid] < x) {
            low = mid + 1;
        }
        else {
            high = mid-1;
        }
        cout << "   ~~~ flag: " << flag << "\n"; //show flag
    }
    if (flag > 0) {
        cout << " ----- \n";
        cout << " Berhasil. Data " << x << " yang dicari ada
dalam array \n";
    }
    //jika tidak ditemukan
    else {
        cout << " ----- \n";
        cout << " Maaf. Data yang dicari, tidak ditemukan
dalam array \n";
    }
    getch();
}

```

Output dari program binary search di atas jika pencarian ditemukan:

```

Select D:\01. DOCUMENTS
PROGRAM BINARY SEARCHING
Masukkan jumlah data (maks 20): 9
Masukkan Data ke - 1 = 65
Masukkan Data ke - 2 = 223
Masukkan Data ke - 3 = 57
Masukkan Data ke - 4 = 32
Masukkan Data ke - 5 = 7
Masukkan Data ke - 6 = 34
Masukkan Data ke - 7 = 84
Masukkan Data ke - 8 = 21
Masukkan Data ke - 9 = 3
Masukkan data yang anda akan cari : 21
--- DATA SETELAH DI URUTKAN ---
3 7 21 32 34 57 65 84 223
--- PROSES PENCARIAN ---
-> Low: 0 -> Mid: 4 -> High: 9   ~~~ flag: 0
-> Low: 0 -> Mid: 1 -> High: 3   ~~~ flag: 0
-> Low: 2 -> Mid: 2 -> High: 3   ~~~ flag: 1
-----
Berhasil. Data 21 yang dicari ada dalam array

```


Output dari program *binary search* di atas jika **pencarian tidak ditemukan**:

```
D:\01. DOCUMENTS\
PROGRAM BINARY SEARCHING
Masukkan jumlah data (maks 20): 10
Masukkan Data ke - 1 = 56
Masukkan Data ke - 2 = 7
Masukkan Data ke - 3 = 24
Masukkan Data ke - 4 = 536
Masukkan Data ke - 5 = 44
Masukkan Data ke - 6 = 78
Masukkan Data ke - 7 = 235
Masukkan Data ke - 8 = 821
Masukkan Data ke - 9 = 636
Masukkan Data ke - 10 = 54
Masukkan data yang anda akan cari : 20
--- DATA SETELAH DI URUTKAN ---
7 24 44 54 56 78 235 536 636 821
--- PROSES PENCARIAN ---
-> Low: 0 -> Mid: 5 -> High: 10      ~~~~ flag: 0
-> Low: 0 -> Mid: 2 -> High: 4      ~~~~ flag: 0
-> Low: 0 -> Mid: 0 -> High: 1      ~~~~ flag: 0
-> Low: 1 -> Mid: 1 -> High: 1      ~~~~ flag: 0
-----
Maaf. Data yang dicari, tidak ditemukan dalam array
```

BAGIAN 19

PEMROGRAMAN *SORTING*

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami dan menguasai konsep algoritma *sorting* data pada *array*
2. Menerapkan algoritma *sorting* disetiap metodenya (*bubble*, *selection* dan *insertion*)

Teori:

Pemrograman *sorting* adalah proses mengatur sekumpulan objek menurut aturan atau susunan tertentu. Urutan objek tersebut dapat menaik atau disebut juga *ascending* (dari data kecil ke data lebih besar) ataupun menurun atau disebut dengan *descending* (dari data besar ke data kecil).

Metode *sorting* ada 3, yaitu:

- *Bubble sort*/pengurutan gelembung
- *Selection sort*/pengurutan maksimum-minimum
- *Insertion sort*/pengurutan sisipan.

A. BUBBLE SORT (PENGURUTAN GELEMBUNG)

Bubble sort (pengurutan gelembung) ini merupakan suatu metode pengurutan gelembung yang diinspirasi oleh gelembung sabun yang ada di dalam permukaan air, karena berat jenis gelembung sabun lebih ringan daripada berat jenis air maka gelembung sabun akan selalu megapung.

Prinsip pengapungan ini juga dipakai pada pengurutan gelembung. Elemen yang berharga paling kecil “diapungkan”, yang artinya diangkat ke atas (atau ke ujung paling kiri) melalui pertukaran. Proses pengapungan ini dilakukan N kali langkah. Pada langkah ke 1, *array*[1....N] akan terdiri dari 2 bagian yaitu :

1. Bagian yang sudah terurut yaitu *L*[1]...*L*[*i*]
2. Bagian yang belum terurut *L*[*i*+1]..*L*[*n*]

Contoh Program:

Mengurutkan data yang diinput secara acak menggunakan metode pengurutan gelembung (*bubble sort*).

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

void main()
{
```

```

int nilai['n'];
int temp, n, a, b;
cout << " ---PROGRAM BUBBLE SORT---\n";
cout << " Banyak data dalam array : ";
cin >> n;
cout << endl;
for (a=1; a<=n; a++) {
    cout << " input nilai[" << a << "] : ";
    cin >> nilai[a];
}
cout << "\n Data sebelum diurutkan\n";
for (a=1; a<=n; a++) {
    cout << " " << nilai[a];

//proses pengurutan bubble
for (a=n-1; a>=1; a--) {
    cout << "\n -> Looping ke-" << a ; // untuk
menampilkan proses looping 'A'.
    for (b=1; b<=a; b++) {
        if (nilai[b] > nilai[b + 1]) {
            temp = nilai[b+1];
            nilai[b+1] = nilai[b];
            nilai[b] = temp;
        }
        //////////MENAMPILKAN PROSES SORTING //////////
        cout << "\n    -> " << b; // untuk menampilkan
proses looping 'B'.
        cout << " - [";
        for (int c=1; c<=n; c++) { // looping untuk
menampilkan proses sorting. one-by-one :)
            if (c==n) {
                cout << nilai[c];
            }
            else {
                cout << nilai[c] << ",";
            }
        }
        cout << "]"";
    }
}
cout << "\n\n Data setelah diurutkan (ascending)\n";
for (a=1; a<=n; a++) {
    cout << " " << nilai[a];

}
cout << "\n\n Data setelah diurutkan (descending)\n";

```

```

    for (a=n; a>=1; a--) {
        cout << " " << nilai[a];
    }
    getche();
}

```

Output dari program pengurutan *bubble sort* di atas adalah:

```

D:\01. DOCUMENTS\
---PROGRAM BUBBLE SORT---
Banyak data dalam array : 7

input nilai[1] : 98
input nilai[2] : 14
input nilai[3] : 76
input nilai[4] : 3
input nilai[5] : 66
input nilai[6] : 91
input nilai[7] : 36

Data sebelum diurutkan
98 14 76 3 66 91 36
-> Looping ke-6
-> 1 - [14,98,76,3,66,91,36]
-> 2 - [14,76,98,3,66,91,36]
-> 3 - [14,76,3,98,66,91,36]
-> 4 - [14,76,3,66,98,91,36]
-> 5 - [14,76,3,66,91,98,36]
-> 6 - [14,76,3,66,91,36,98]
-> Looping ke-5
-> 1 - [14,76,3,66,91,36,98]
-> 2 - [14,3,76,66,91,36,98]
-> 3 - [14,3,66,76,91,36,98]
-> 4 - [14,3,66,76,91,36,98]
-> 5 - [14,3,66,76,36,91,98]
-> Looping ke-4
-> 1 - [3,14,66,76,36,91,98]
-> 2 - [3,14,66,76,36,91,98]
-> 3 - [3,14,66,76,36,91,98]
-> 4 - [3,14,66,36,76,91,98]
-> Looping ke-3
-> 1 - [3,14,66,36,76,91,98]
-> 2 - [3,14,66,36,76,91,98]
-> 3 - [3,14,36,66,76,91,98]
-> Looping ke-2
-> 1 - [3,14,36,66,76,91,98]
-> 2 - [3,14,36,66,76,91,98]
-> Looping ke-1
-> 1 - [3,14,36,66,76,91,98]

Data setelah diurutkan (ascending)
3 14 36 66 76 91 98

Data setelah diurutkan (descending)
98 91 76 66 36 14 3_

```

B. SELECTION SORT (PENGURUTAN MAKSIMUM-MINIMUM)

Selection sort (pengurutan maksimum-minimum) ini merupakan metode pengurutan yang didasarkan pada pemilihan elemen maksimum atau minimum kemudian mempertukarkan elemen maksimum-minimum tersebut dengan elemen terujung larik (elemen ujung kiri atau elemen ujung kanan), selanjutnya elemen terujung itu kita “isolasi” dan tidak diikuti sertakan pada proses selanjutnya. Karena proses utama dalam pengurutan adalah pemilihan elemen maksimum/minimum, maka metode ini disebut metode pemilihan (*selection*).

Contoh Program:

Mengurutkan data yang diinput secara acak menggunakan metode pengurutan maksimum-minimum (*selection sort*).

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

void main()
{
    clrscr();
    int j, jmax, k, i, temp, n;
    cout << " ---PROGRAM SELECTION SORT---\n";
    cout << " Banyak data dalam array : ";
    cin >> n;
    int A['n']; //deklarasi ukuran array
    int u=n-1;
    for (i=0; i<n; i++)
    {
        cout << " input nilai A [" << i << "] : ";
        cin >> A[i];
    }
    ///sebelum diurutkan///
    cout << "\n -----";
    cout << "\n Nilai sebelum diurutkan";
    cout << "\n ----- \n";
    for (i=0; i<n; i++)
    {
        cout << " " << A[i];
    }
    cout << "\n -----";
    //proses pengurutan
    for (j=0; j<n; j++)
    {
        jmax=0;
        for (k=1; k<=u; k++)
        {
```

```

        if (A[k] > A[jmax])
        {
            jmax = k;
            cout << "\n ditemukan Jmax pada index: [" <<
jmax << "] nilai berisi: " << A[jmax];
        }
    }
    temp = A[u];
    A[u] = A[jmax];
    A[jmax] = temp;
    cout << "\n tukar index[" << u << "] ke Jmax[" <<
jmax << "];
    cout << "\n --> data sekarang [";
    for (int c=0; c<n; c++)
    { //menampilkan hasil tukar
        if (c==n-1) {
            cout << A[c];
        }
        else {
            cout << A[c] << ",";
        }
    }
    cout << "];
    u--;
    cout << "\n -----";
}
//menampilkan nilai setelah diurutkan
cout << "\n Nilai setelah diurutkan (ascending)\n";
for (i=0; i<n; i++)
{
    cout << " " << A[i];
}
cout << "\n Nilai setelah diurutkan (descending)\n";
for (i=n-1; i>=0; i--)
{
    cout << " " << A[i];
}
getche();
}

```

Output dari program pengurutan *selection sort* di atas adalah:

```
D:\01. DOCUMENTS\
---PROGRAM SELECTION SORT---
Banyak data dalam array : 7
input nilai A [0] : 98
input nilai A [1] : 12
input nilai A [2] : 755
input nilai A [3] : 34
input nilai A [4] : 8
input nilai A [5] : 41
input nilai A [6] : 603

-----
Nilai sebelum diurutkan
-----
98 12 755 34 8 41 603
-----
ditemukan Jmax pada index: [2] nilai berisi: 755
tukar index[6] ke Jmax[2]
--> data sekarang [98,12,603,34,8,41,755]
-----
ditemukan Jmax pada index: [2] nilai berisi: 603
tukar index[5] ke Jmax[2]
--> data sekarang [98,12,41,34,8,603,755]
-----
tukar index[4] ke Jmax[0]
--> data sekarang [8,12,41,34,98,603,755]
-----
ditemukan Jmax pada index: [1] nilai berisi: 12
ditemukan Jmax pada index: [2] nilai berisi: 41
tukar index[3] ke Jmax[2]
--> data sekarang [8,12,34,41,98,603,755]
-----
ditemukan Jmax pada index: [1] nilai berisi: 12
ditemukan Jmax pada index: [2] nilai berisi: 34
tukar index[2] ke Jmax[2]
--> data sekarang [8,12,34,41,98,603,755]
-----
ditemukan Jmax pada index: [1] nilai berisi: 12
tukar index[1] ke Jmax[1]
--> data sekarang [8,12,34,41,98,603,755]
-----
tukar index[0] ke Jmax[0]
--> data sekarang [8,12,34,41,98,603,755]
-----
Nilai setelah diurutkan (ascending)
8 12 34 41 98 603 755
Nilai setelah diurutkan (descending)
755 603 98 41 34 12 8
```

C. INSERTION SORT (PENGURUTAN SISIPAN)

Insertion sort (pengurutan sisipan) ini merupakan metode Pengurutan sisip adalah metode pengurutan dengan cara menyisipkan elemen larik pada posisi

yang tepat. Pencarian posisi yang tepat dilakukan dengan pencarian beruntun. Selama pencarian posisi yang tepat dilakukan pergeseran elemen larik.

Misalkan ada 6 kartu tersusun acak diatas meja. Dan bayangkan bahwa kartu-kartu tersebut sebagai *array* yang memiliki urutan 2, 8, 5, 3, 9, 4.

2	8	5	3	9	4
---	---	---	---	---	---

Algoritma *insertion sort* dimulai dari index array ke 1 yang berarti dimulai dari angka 8. Setiap perulangan kita akan membandingkan angka disebelah kirinya hingga angka tersebut lebih besar daripada angka disebelah kirinya. disini angka 8 lebih besar daripada angka 2 jadi tidak ada perubahan.

2	8	5	3	9	4
---	---	---	---	---	---

Lanjut masuk ke angka 5, disini angka 5 lebih kecil dibanding angka disebelah kirinya yaitu 8. Tapi dia lebih besar dibanding 2, jadi angka 5 akan masuk diantara 2 dan 8. sedangkan angka 8 bergeser ke kanan.

		5			
2	8	8	3	9	4

Selanjutnya angka 3, disini angka 3 lebih kecil dibanding 8 dan 5 tapi lebih besar dibanding 2, maka angka 3 akan bergeser 2 langkah ke kiri, sedangkan angka 5 dan 8 akan bergeser 1 langkah ke kanan.

		3			
2		5	8	9	4

Selanjutnya angka 9, karena ia lebih besar daripada 8, maka posisinya tidak berubah.

2	3	5	8	9	4
---	---	---	---	---	---

Dan terakhir angka 4, disii angka 4 lebih kecil dibanding angka 5, 8, 9 maka ia bergeser 3 langkah ke kiri dan angka 5, 8, 9 bergeser 1 langkah ke kanan.

Jadi algoritma pengurutan menggunakan metode ini sama halnya seperti kita mengurutkan kartu diatas meja, dimana kita menggeser kartunya hingga menemukan posisi yang pas.

Contoh Program:

Mengurutkan *ascending* dan *descending* data yang diinput secara acak menggunakan metode pengurutan sisipan (*insertion sort*).

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

void main()
{
    clrscr();
    int b, i, j, k, m, n, x, z;
    int data['n'];
    cout << " ---PROGRAM INSERTION SORT---\n";
    cout << " Banyak Data : ";
    cin >> n;
    //proses input data
    for (i=0; i<n; i++) {
        cout << " input data ke-" << i << " : ";
        cin >> data[i];
    }
    //menampilkan data sebelum diurutkan
    cout << "\n Data sebelum diurutkan" << endl;
    for (i=0; i<n; i++) {
        cout << " #" << i << " - " << data[i] << endl;
    }
    //menampilkan proses pengurutan ascending
    cout << "\n Proses pengurutan ascending" << endl;
    for (i=0; i<n; i++) {
        for (j=0; j<=i; j++) {
            if (data[i] < data[j]) {
                m = data[i];
                data[i] = data[j];
                data[j] = m;
            }
        }
        cout << " #" << i << " :";
        for (k=0; k<=i; k++) {
            cout << " " << data[k];
        }
        cout << endl;
    }
    //menampilkan data setelah diurutkan ascending
    cout << "\n Data setelah diurutkan\n";
    for (i=0; i<n; i++)
    {
        cout << " " << i+1 << " - " << data[i] << endl;
    }
}
```

```

}
//menampilkan proses pengurutan descending
cout << "\n Proses pengurutan descending\n";
for (i=0; i<n; i++) {
    for (j=0; j<=i; j++) {
        if (data[i] > data[j]) {
            m = data[i];
            data[i] = data[j];
            data[j] = m;
        }
    }
    cout << " #" << i << " :";
    for (k=0; k<=i; k++) {
        cout << " " << data[k];
    }
    cout << endl;
}
//menampilkan data setelah diurutkan descending
cout << "\n Data setelah diurutkan\n";
for (i=0; i<n; i++) {
    cout << " " << i+1 << " - " << data[i] << endl;
}
getche();
}

```

Output dari program pengurutan *ascending* dan *descending* pada *insertion sort* di atas:

```

D:\01. DOCUMENTS\
---PROGRAM INSERTION SORT---
Banyak Data : 7
input data ke-0 : 78
input data ke-1 : 227
input data ke-2 : 54
input data ke-3 : 37
input data ke-4 : 579
input data ke-5 : 54
input data ke-6 : 773

Data sebelum diurutkan
#0 - 78
#1 - 227
#2 - 54
#3 - 37
#4 - 579
#5 - 54
#6 - 773

```

```

Proses pengurutan ascending
#0 : 78
#1 : 78 227
#2 : 54 78 227
#3 : 37 54 78 227
#4 : 37 54 78 227 579
#5 : 37 54 54 78 227 579
#6 : 37 54 54 78 227 579 773

Data setelah diurutkan
1 - 37
2 - 54
3 - 54
4 - 78
5 - 227
6 - 579
7 - 773

Proses pengurutan descending
#0 : 37
#1 : 54 37
#2 : 54 54 37
#3 : 78 54 54 37
#4 : 227 78 54 54 37
#5 : 579 227 78 54 54 37
#6 : 773 579 227 78 54 54 37

Data setelah diurutkan
1 - 773
2 - 579
3 - 227
4 - 78
5 - 54
6 - 54

```

Contoh Kasus:

1. Sebuah Usaha Dagang milik Mentari berhasil melakukan penjualan barang dagangan nya setiap hari. Data-data yang terjual setelah dicatat ternyata belum berurutan. Maka dibuatkanlah program sederhana yang dapat mengurutkan kode barang yang terjual tiap harinya, sehingga proses rekapitulasi penjualan dapat berjalan efisien tiap harinya. Berikut adalah *source code* programnya.

```

#include <iostream.h>
#include <conio.h>

void main()
{
    clrscr();
    int i, j, temp, jmax;

```

```

int data_jual[7];
int u=6;
cout << "DATA PENJUALAN UD. MENTARI\n";
cout << "Input KODE Data Barang Laku Harian \n";
for (i=0; i<7; i++)
{
    cout << "\nMasukkan Kode Penjualan Ke- " << i+1
<< " : ";
    cin >> data_jual[i];
}
cout << "\nKODE Data Barang Laku Hari Ini adalah =
\n";
for (i=0; i<7; i++)
{
    cout << data_jual[i] << " ";
}
for (i=0; i<7; i++)
{
    jmax=0;
    for (j=1; j <= u; j++)
        if (data_jual[j] > data_jual[jmax])
        {
            jmax=j;
        }
    temp = data_jual[u];
    data_jual[u] = data_jual[jmax];
    data_jual[jmax] = temp;
    u--;
}
cout << "\nKODE Data Barang Laku Hari Ini Setelah di
URUTKAN adalah = \n";
for (i=0; i<7; i++)
{
    cout << data_jual[i] << " ";
}
getche();
}

```

Tampilan *output* dari program *sorting* diatas adalah:



```
D:\01. DOCUMENTS\
DATA PENJUALAN UD. MENTARI
Input KODE Data Barang Laku Harian
Masukkan Kode Penjualan Ke- 1 : 456
Masukkan Kode Penjualan Ke- 2 : 2327
Masukkan Kode Penjualan Ke- 3 : 546
Masukkan Kode Penjualan Ke- 4 : 32468
Masukkan Kode Penjualan Ke- 5 : 56
Masukkan Kode Penjualan Ke- 6 : 46
Masukkan Kode Penjualan Ke- 7 : 288

KODE Data Barang Laku Hari Ini adalah =
456 2327 546 32468 56 46 288
KODE Data Barang Laku Hari Ini Setelah di URUTKAN adalah =
46 56 288 456 546 2327 32468
```

2. Sebuah Bank ABC memiliki layanan pengaduan nasabah. Pengaduan nasabah tersebut mencatat setiap nomor rekening nasabah yang bermasalah tiap harinya. Maka dibuatkanlah program sederhana untuk mengurutkan nomor rekening nasabah tersebut agar memudahkan proses pengarsipan aktivitas layanan pengaduan nasabah Bank ABC.

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

void main()
{
    clrscr();
    int a, b, t, n;
    int data[100];
    cout << "DATA PENGADUAN HARIAN NASABAH BANK ABC \n";
    cout << "Input jumlah pengaduan harian: ";
    cin >> n;
    for (int a=0; a<n; a++)
    {
        cout << "\nMasukkan No. Rek Pengadu ke-" <<
(a+1) << " = ";
        cin >> data[a];
    }

    cout << "\nList No. Rek nasabah yang melakukan
pengaduan hari ini adalah = \n";
    for (a=0; a<n; a++)
    {
        cout << data[a] << " ";
    }
    for (a=n-1; a>=1; a--)
```

```

{
    for (b=0; b<a; b++)
    {
        if (data[b] > data[b+1])
        {
            t = data[b+1];
            data[b+1] = data[b];
            data[b] = t;
        }
    }
}

cout << "\nData No. Rek nasabah setelah diurutkan
adalah =\n";
for (a=0; a<n; a++)
{
    cout << data[a] << " ";
}
getche();
}

```

Tampilan *output* dari program *sorting* diatas adalah:

```

D:\01. DOCUMENTS\'
DATA PENGADUAN HARIAN NASABAH BANK ABC
Input jumlah pengaduan harian: 6
Masukkan No. Rek Pengadu ke-1 = 345
Masukkan No. Rek Pengadu ke-2 = 7773
Masukkan No. Rek Pengadu ke-3 = 325775
Masukkan No. Rek Pengadu ke-4 = 345754
Masukkan No. Rek Pengadu ke-5 = 799534
Masukkan No. Rek Pengadu ke-6 = 18343

List No. Rek nasabah yang melakukan pengaduan hari ini adalah =
345 7773 325775 345754 799534 18343
Data No. Rek nasabah setelah diurutkan adalah =
345 7773 18343 325775 345754 799534

```

PRAKTIKUM KEEMPAT

Soal Praktikum:

1. (Soal Kasus) Sebuah Tim Pengacara Hukum di Sumbawa memiliki layanan program berupa jasa konsultasi. Umpamakan Anda adalah seorang *client* yang menggunakan jasa konsultasi hukum ini. Buatlah program dengan menggunakan struktur penyeleksian kondisi dalam bahasa C++ ! Dengan *output* seperti berikut:

```
*****
PROGRAM BIAYA KONSULTASI
TIM PENGACARA SUMBAWA
*****
Kode Pengacara;
1. Shinta Esabella, M.H
2. Rodianto, M.H
3. DR. Saiful. M.H
4. Prof. Widiarta, M.H
5. DR. Miftahul, M.H

Kode Konsultasi;
A. Masalah Pertanahan
B. Masalah Sengketa
C. Masalah Perceraian
D. Masalah Harta Waris
E. Masalah Hukum Kerjasama

Masukkan Nama Klien      : _____
Masukkan Kode Pengacara  : _____
Masukkan Kode Konsultasi : _____
Masukkan Lama Konsultasi : _____ //jam

-----
Detail Biaya Konsultasi
-----
Nama Klien                : _____
Nama Pengacara            : _____
Konsultasi Pilihan        : _____
Lama Konsultasi           : _____ jam
Total Biaya Konsultasi    : Rp_____

Uang Yang Dibayar         : _____ //input
Uang Kembalian            : _____
```

Ketentuan:

Tarif pengacara per/jam konsultasi

- | | |
|-------------------------|---------------|
| 1. Shinta Esabella, M.H | = Rp. 125.000 |
| 2. Rodianto, M.H | = Rp. 125.000 |
| 3. DR. Saiful. M.H | = Rp. 350.000 |
| 4. DR. Wilia, M.H | = Rp. 350.000 |
| 5. Prof. Widiarta, M.H | = Rp. 700.000 |

Rumus:

$\text{biaya_konsultasi} = \text{tarif_pengacara} * \text{lama_konsultasi}$

2. Buatlah program sederhana dalam bahasa pemrograman C++ menggunakan pemrograman rekursif dengan 3 buah *function* untuk menampilkan *output* di bawah ini:

```
*
* *
* * *
* * * *
* * * *
* * * *
* * *
* *
*
```

3. Tuliskanlah *output* dari *source-code* di bawah ini! (dengan memberikan input sesuai apa yang diminta oleh program)

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

typedef struct {
    char nama[30];
    short umur;
    char jurusan[50];
    float ipk;
} mahasiswa;

void main()
{
    clrscr();
    mahasiswa saya[3];
    int i;
    cout << "DATA SISWA DAN UMURNYA\n";
```



```

    for (i=0; i<=2; i++)
    {
        cout << "Masukkan nama siswa ke- " << i+1 << "
= ";
        cin >> saya[i].nama;
        cout << "Masukkan umur = ";
        cin >> saya[i].umur;
        cout << "Masukkan Jurusan = ";
        cin >> saya[i].jurusan;
        cout << "Masukkan IPK = ";
        cin >> saya[i].ipk;
    }

    cout << "\nDATABASE\n";
    cout << "=====\n";
    for (i=0; i<=2; i++)
    {
        cout << "\nNama " << saya[i].nama << " berumur
" << saya[i].umur;
        cout << "\nJurusan " << saya[i].jurusan << "
IPK " << saya[i].ipk;
    }
    getche();
}

```

BAGIAN 20

POINTER DALAM PEMROGRAMAN C++

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami dan menguasai konsep *pointer* dalam pemrograman C++
2. Menerapkan konsep *pointer* dalam program sederhana.

Teori:

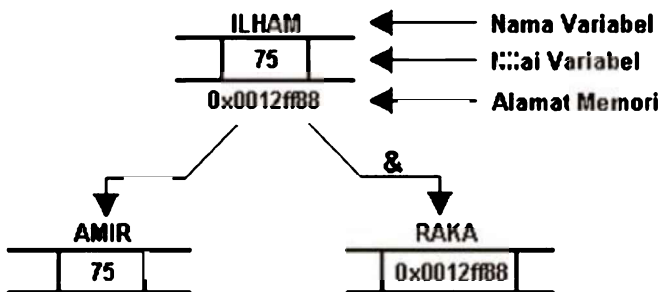
Pointer merupakan sebuah variabel yang berisi alamat dari variabel lain. Suatu *pointer* dimaksudkan untuk menunjukan ke suatu alamat memori sehingga alamat dari suatu variabel dapat diketahui dengan mudah. Arti *pointer* dalam bahasa sehari-hari adalah petunjuk atau bisa dibilang penentu atau *pointer* secara sederhana bisa diartikan sebagai tipe data yang nilainya mengarah pada nilai yang terdapat pada sebuah area memori (alamat memori).

Fungsi *pointer* yang utama adalah untuk menyimpan alamat memori dari sebuah variabel. Selain menyimpan alamat dari sebuah variabel, *pointer* juga berfungsi untuk menyimpan alamat memori dari sebuah fungsi.

A. OPERATOR DEREFERENCE (&)

Operator ini biasa disebut dengan “*address of*” atau *operator* alamat. Dengan menggunakan *operator dereference* ini, suatu variabel akan menghasilkan/ menampilkan alamat lokasi memori.

```
ILHAM = 75;
AMIR = ILHAM;    // AMIR sama dengan ILHAM (75)
RAKA = &ILHAM;  // RAKA sama dengan Address Of ILHAM (0x0012ffb8)
```



Jika dilihat dari gambar, sudah ditentukan dimana nilai variabel dari ILHAM adalah 75, sedangkan nilai variabel AMIR sama dengan nilai variabel ILHAM yaitu 75 (**AMIR=ILHAM**), dan nilai variabel RAKA **sama dengan alamat memori** dari nilai variabel ILHAM (**RAKA=&ILHAM**) karena menggunakan

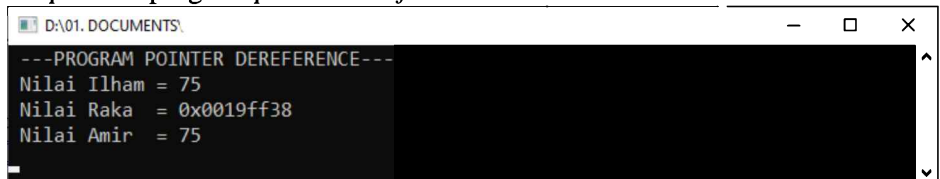
operator dereference yang ditandai dengan tanda (&). Jadi yang *output* untuk nilai variabel RAKA adalah hanya alamat dari memori nilai variabel ILHAM yaitu 0x0019ff38.

Contoh Program:

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

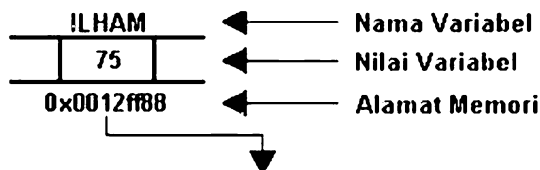
void main()
{
    int ilham, *raka, amir;
    ilham = 75;
    raka = &ilham; /*Nilai variabel raka adalah merupakan
alamat memori dari nilai ilham*/
    amir = ilham;
    cout << " ---PROGRAM POINTER DEREFERENCE---\n";
    cout << " Nilai Ilham = " << ilham << endl;
    cout << " Nilai Raka  = " << raka << endl;
    cout << " Nilai Amir  = " << amir << endl;
    getch();
}
```

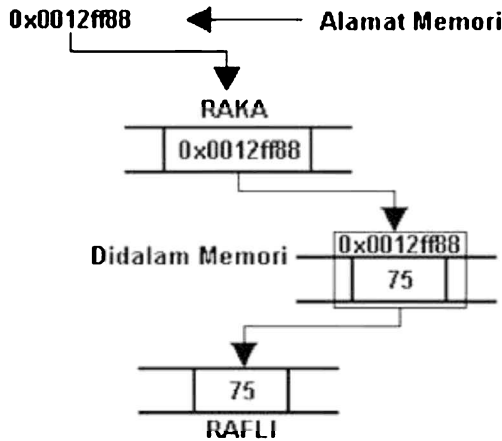
Output dari program *pointer dereference* di atas:



B. OPERATOR REFERENCE (*)

Operator Reference atau biasa disebut dengan “*value pointed by*”. Bedanya dengan *Operator Dereference*, operator ini akan menampilkan nilai yang terdapat dalam suatu alamat memori, bukan menampilkan alamat memorinya.





```

ilham = 75;
raka = &ilham;
rafli = *raka;

```

Dari gambar di atas dimana nilai variabel ILHAM (Ilham=75) sudah ditentukan sebelumnya yaitu 75 dengan alamat memori **0x0019ff38**. Sedangkan nilai variabel RAKA adalah merupakan **alamat memori** dari nilai variabel ILHAM (RAKA=&ILHAM) karena menggunakan *operator pointer deference* (diawali dengan tanda "&") yaitu **0x0019ff38** dan untuk nilai variabel RAFLI merupakan **nilai yang terdapat dalam alamat memori** dari nilai variabel RAKA (RAFLI=*RAKA) karena menggunakan *operator pointer reference* (diawali dengan tanda "*") yaitu 75.

Contoh Program:

```

#include <iostream.h>
#include <conio.h>
void main()
{
    int ilham, rafli, *raka;
    ilham = 75;
    raka = &ilham;
    rafli = *raka;
    cout << " ---PROGRAM POINTER REFERENCE---\n";
    cout << "Nilai Ilham adalah : " << ilham << endl;
    cout << "Nilai Raka adalah : " << raka << endl;
    cout << "Nilai Rafli adalah : " << rafli << endl;
    getche();
}

```

Output dari program *pointer reference* di atas:



A screenshot of a Windows command prompt window. The title bar shows the file path "D:\01. DOCUMENTS\" and standard window controls (minimize, maximize, close). The command prompt has a black background with white text. The output of a program is displayed, showing three lines of data. The first line is a separator "---PROGRAM POINTER REFERENCE---". The second line shows "Nilai Ilham adalah : 75". The third line shows "Nilai Raka adalah : 0x0019ff38". The fourth line shows "Nilai Rafli adalah : 75". A vertical scrollbar is visible on the right side of the text area.

```
---PROGRAM POINTER REFERENCE---  
Nilai Ilham adalah : 75  
Nilai Raka adalah : 0x0019ff38  
Nilai Rafli adalah : 75
```

BAGIAN 21

FILE (BERKAS) DALAM PEMROGRAMAN C++

Tujuan:

Setelah menyelesaikan bagian ini, diharapkan mampu:

1. Memahami dan menguasai konsep *file* (berkas) dalam pemrograman C++
2. Menerapkan konsep *file* (berkas) dalam program sederhana.

Teori:

File disini dapat dikatakan sebagai penyimpanan data eksternal yang bersifat permanen. Operasi-operasi terhadap *file* berkaitan dengan input dan juga *output* serta hal lain seperti mengecek keberadaan suatu *file*, ukuran *file*, dan lain-lain. Contoh program ini akan menggunakan header “*fstream*”.

- *ofstream* untuk menulis file
- *ifstream* untuk membaca file
- *fstream* untuk keduanya (menulis dan membaca)

Contoh Program:

1. Program sederhana untuk menulis sesuatu kedalam *file* (berkas)

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
#include <stdlib.h>


void main()
{
    //stream untuk menulis file
    ofstream myfile;
    //membuka file
    myfile.open("TEST.txt", ios::app);
    cout << "OPERASI FILE 1\n";
    cout << "-----\n";
    //fail() untuk memeriksa suatu kesalahan pada operasi
    file
    if (!myfile.fail()) { //menulis ke dalam file
        myfile << "Belajar OPERASI FILE\n";
        myfile.close(); //menutup file
        cout << "Text telah ditulis ke dalam file\n";
    }
}
```

```

    else {
        cout << "File tidak ditemukan\n";
    }
    getche();
}

```

Output dari program menulis *file* (berkas) di atas:



```

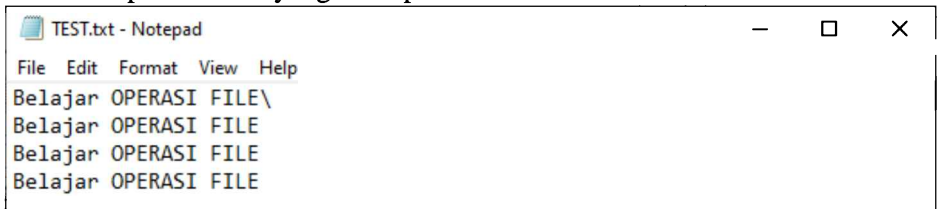
D:\01. DOCUMENTS\
OPERASI FILE 1
-----
Text telah ditulis ke dalam file

```

File (berkas) pada folder:

Name	Date modified	Type
TEST.txt	05-Mar-21 00:41	Text Document
bcwdef.csm	22-Feb-21 22:20	CSM File

Isi tulisan pada berkas yang disimpan:



```

TEST.txt - Notepad
File Edit Format View Help
Belajar OPERASI FILE\
Belajar OPERASI FILE
Belajar OPERASI FILE
Belajar OPERASI FILE

```

Contoh Program:

2. Program sederhana untuk membaca file (berkas)

```

#include <iostream.h>
#include <conio.h>
#include <fstream.h>
#include <stdlib.h>

void main()
{
    //stream untuk menulis file
    ifstream myfile;
    char sv_text;
    //membuka file
    myfile.open("TEST.txt", ios::nocreate);
    cout << "OPERASI FILE 1\n"

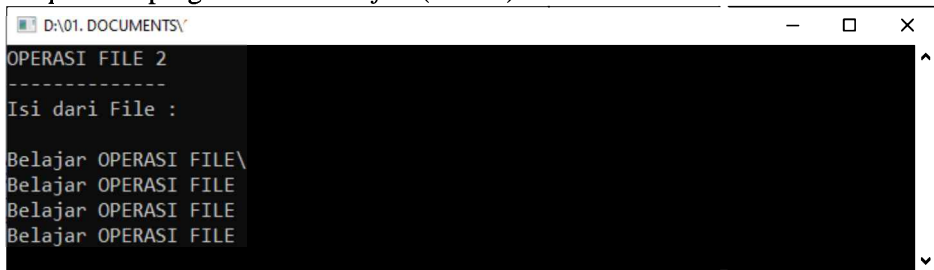
```

```

        << "-----\n";
//fail() untuk memeriksa suatu kesalahan pada operasi
file
    if (!myfile.fail()) {
        cout << "Isi dari File :\n\n";
        while (!myfile.eof()) {
            myfile.get(sv_text);
            cout << sv_text;
        }
        myfile.close();
    }
    else {
        cout << "File tidak ditemukan\n";
    }
    getch();
}

```

Output dari program membaca *file* (berkas) di atas:



```

D:\01. DOCUMENTS\
OPERASI FILE 2
-----
Isi dari File :
Belajar OPERASI FILE\
Belajar OPERASI FILE
Belajar OPERASI FILE
Belajar OPERASI FILE

```


EVALUASI PEMAHAMAN

Soal Evaluasi:

1. (Soal Kasus) Buatlah program sederhana dalam bahasa pemrograman C++ menggunakan algoritma struktur array (*array of struct*) untuk Menampilkan data lagu yang berisi tentang judul lagu, penyanyi, tahun produksi, nomor *track* dan kode album seperti *output* di bawah ini!

```
*****
PROGRAM KOLEKSI LAGU RBT (RING BACK TONE)
*****
Berapa data lagu yang anda input? 2 //input

Data ke-1:
    Judul Lagu      : Bersamamu      //input
    Penyanyi       : Vierra          //input
    Tahun Produksi  : 2009            //input
    Nomor Track    : 03               //input
    Kode Album     : MFL              //input

Data ke-2:
    Judul Lagu      : It Will Rain   //input
    Penyanyi       : Bruno Mars      //input
    Tahun Produksi  : 2011            //input
    Nomor Track    : 01               //input
    Kode Album     : BMR              //input

-----
Data lagu yang anda miliki:
    Data 1:
        Penyanyi    : Vierra
        Judul Lagu  : Bersamamu
        Kode RBT    : 03-MFL
        Tahun       : 2009
    Data 2:
        Penyanyi    : Bruno Mars
        Judul Lagu  : It Will Rain
        Kode RBT    : 01-BMR
        Tahun       : 2011
-----
```

Ketentuan:

- a. Program ini akan memiliki dua buah *struct*, yaitu *struct* lagu dan *struct* kodeRBT.
- b. Jumlah data yang diinputkan dinamis (maks. 20 lagu).

2. Buatlah program sederhana dalam bahasa pemrograman C++ dengan menggunakan pemrograman *array* dua dimensi untuk menampilkan *output* di bawah ini:

```
*****
PROGRAM PENGURANGAN DUA MATRIKS
*****
---MATRIKS A---
20  30  40  50
10  20  30  40

---MATRIKS B---
1  2  3  4
4  3  2  1

=== MATRIKS A DIKURANGI MATRIKS B ===
20-1=19    30-2=28    40-3=37    50-4=46
10-4=16    20-3=17    30-2=28    40-1=39

TOTAL PENJUMLAHAN DARI MATRIKS (A-B) SELURUHNYA = 230
```

3. Dari *source-code* program di bawah ini ketika *compile* masih ditemukan 5 item *sintax* yang *error*, perbaikilah dan tuliskan kembali *sourcecode* program, sehingga program dapat berjalan dengan benar pada bahasa pemrograman C++!

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

void main()
{
    clrscr();
    int i, j, temp, jmax;
    int data_jual[7];
    int u=6;
    cout << "DATA PENJUALAN UD. BERLIAN\n";
    cout << "Input kode data barang laku harian \n";
    for (i=0; i<7; i++)
```

```

    {
        cout << "\nMasukkan Kode Penjualan Ke- " << i+1
        << " : ";
        cin >> data_jual[j];
    }
    cout << "\nKODE Data Barang Laku Hari Ini adalah =
\n";
    for (i=0; i<7; i++)
    {
        cout << data_jual[i] << " ";
    }
    for (i=0; i<7; i++)
    {
        jmax=0;
        for (j=1; j<=u; j++)
            if (data_jual[j] > data_jual[jmax])
            {
                jmax=j
            }
        temp = data_jual[u];
        data_jual[u] = data_jual[jmax];
        data_jual[jmax] = temp;
        u++;
    }
    cout << "\nKODE Data Barang Laku Hari Ini Setelah
di URUTKAN adalah = \n";
    for (i=0; i<7; i++)
    {
        cout << data_jual[i] << " ";
    }
    getch();
}

```

4. Setelah Anda memperbaiki *sintax* yang *error* dari program di atas, Tuliskanlah *output* dari hasil perbaikan *code* yang telah Anda kerjakan, dimana *code* data barang yang terjual adalah berikut!

89	71	8	24	17	24	12
----	----	---	----	----	----	----

5. Buatlah program dalam bahasa pemrograman C++ dengan menggunakan algoritma *array* multi-dimensi dan *sorting* untuk menampilkan *output* seperti di bawah ini!

```
*****
      NILAI MATA KULIAH DASAR-DASAR PEMROGRAMAN
      TEKNIK INFORMATIKA
*****
Input jumlah mahasiswa : 10    //input

      Praktikum      UTS      UAS      Rata2
-----
Mahasiswa ke-1      87      96      70      84.33
Mahasiswa ke-2      68      87      90      81.67
Mahasiswa ke-3      94      100     90      94.67
Mahasiswa ke-4      100     81      82      87.67
Mahasiswa ke-5      83      65      85      77.67
Mahasiswa ke-6      78      87      65      76.67
Mahasiswa ke-7      85      75      83      81.00
Mahasiswa ke-8      91      94      100     95.00
Mahasiswa ke-9      76      72      84      77.33
Mahasiswa ke-10     87      93      73      84.33
-----
Nilai terendah adalah   =    65
Nilai tertinggi adalah  =   100
Nilai rata2 kelas adalah =  84.03
```

DAFTAR PUSTAKA

- Almon, C. (2008). *Scientific Programming with Borland C++ Builder and Codegear's Turbo C++*.
- Antonius, R. (2010). *Algoritma dan Pemrograman Dengan Bahasa C*. Andi, Yogyakarta.
- Ardiansyah, H., Amalia, R., & Prasetyo, A. B. (2019). *Algoritma dan Pemrograman 1*.
- Blanchette, J., & Summerfield, M. (2006). *C++ GUI programming with Qt 4*. Prentice Hall Professional.
- Brassard, Gilles (1999), *Fundamentals of algorithma*, PrinteceHall.
- Busbee, K. L. (2013). *Programming Fundamentals: A Modular Structured Approach Using C++*.
- Dale, N. B., & Weems, C. (2014). *Programming and problem solving with C++*. Jones & Bartlett Publishers.
- Donovan, S. (2001). *C++ by Example*. Que Publishing.
- Farrell, J. (2008). *Object-oriented programming using C++*. Cengage Learning.
- Fishwick, P. A. (1992, December). Simpack: getting started with simulation programming in C and C++. In *Proceedings of the 24th conference on Winter simulation* (pp. 154-162).
- Friedman, F. L., & Koffman, E. B. (2011). *Problem Solving, Abstraction, and Design Using C++*. Pearson.
- Goodrich, M. T., Tamassia, R., & Mount, D. M. (2011). *Data structures and algorithms in C++*. John Wiley & Sons.
- Gregor, D., Järvi, J., Siek, J., Stroustrup, B., Dos Reis, G., & Lumsdaine, A. (2006, October). Concepts: Linguistic support for generic programming in C++. In *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications* (pp. 291-310).
- Hanief, S., Jepriana, I. W., & Kom, S. (2020). *Konsep Algoritme dan Aplikasinya dalam Bahasa Pemrograman C++*. Penerbit Andi.
- Hartono, Jogianto. (1992). *Konsep dasar pemrograman bahasa C++*. Yogyakarta: Andi Yogyakarta.

- Heriyanto, Abdul Kadir. (2006). *Algoritma Pemrograman Menggunakan C++*. Yogyakarta: Andi.
- Kadir, Abdul. (2012). *Buku Pintar C++ untuk Pemula*. Yogyakarta: Penerbit Andi.
- Kamthane, A. (2003). *Object-oriented Programming with ANSI and Turbo C++*. Pearson Education India.
- Kotur, P. B. (2012). *Object Oriented Programming with C++* (Vol. 1). Sapna Book House (P) Ltd..
- Langsam, Y., Augenstein, M. J., & Tenenbaum, A. M. (2000). *Data Structures using C and C++*. Prentice-Hall of India.
- Mallia, A., & Zoffoli, F. (2019). *C++ Fundamentals: Hit the ground running with C++, the language that supports tech giants globally*. Packt Publishing Ltd.
- Oualline, S. (2003). *Practical C++ programming*. " O'Reilly Media, Inc."
- Prata, S. (2011). *C++ Primer Plus*. Addison-Wesley Professional.
- Pohl, I. (1993). *Object-oriented programming using C++*. Addison-Wesley.
- Rachmat, C. A. (2010). *Algoritma dan Pemrograman dengan Bahasa C; Konsep Teori, dan implementasi*.
- Raharjo, B. (2006). *Pemrograman C++ Mudah dan Cepat Menjadi Master C++ dengan Mengungkap Rahasia-rahasia pemrograman Dalam C++*. Bandung: Informatika.
- Rinaldi, Munir, and Leony, Lidya (2016). *Algoritma dan Pemograman Dalam Bahasa Pascal, C, dan C++*. Bandung: Informatika
- Ropianto, M. (2018). *Algoritma & Pemrograman*. Deepublish.
- Sari, Y. (2017). *Logika Algoritma, Pseudocode, Flowchart, dan C++*. Perahu Litera.
- Satir, G., & Brown, D. (1995). *C++: The Core Language*. " O'Reilly Media, Inc."
- Satrio, E., & Pakpahan, S. (2019). *Pemrograman C++ Untuk Pembelajar Mandiri* (Vol. 1). Penerbit Cahaya INFORMATIKA.
- Sianipar, R. H. (2015). *Struktur Data C++ Dengan Pemrograman Generik* (Vol. 1). Penerbit ANDI.
- Sianipar, R. H. (2015). *Soal & Penyelesaian C++* (Vol. 1). Penerbit INFORMATIKA.

- Sianipar, R. H. (2014). *Pemrograman C++ Untuk Pemula* (Vol. 1). Penerbit INFORMATIKA.
- Sianipar, R. H., & Mangiri, H. S. (2013). *C++ Untuk Programmer* (Vol. 1). Penerbit INFORMATIKA.
- Sitorus, L. (2015). *Algoritma dan pemrograman*. Penerbit Andi.
- Sismoro, H. (2005). *Pengantar Logika Informatika. Algoritma dan Pemrograman Komputer*. Penerbit Andi.
- Sjukani, M. (2010). *Algoritma: Algoritma dan Struktur Data 1 dengan C, C++ dan Java Edisi 6*. Jakarta: Mitra Wacana Media.
- Sjukani, M. (2012). *Struktur Data (Algoritma dan Struktur data 2 dengan C, C++)*. Jakarta: Mitra Wacana Media.
- Solichin, A. (2003). *Pemrograman Bahasa C dengan Turbo C*. IlmuKomputer.Org.
- Stroustrup, B., & Stroustrup, B. (2014). *The Essence of C++.*. Video. *Edinburgh: The University of Edinburgh*.
- Suryadi, H. S., & Sumin, A. (1997). *Pengantar Algoritma dan Pemrograman Teknik Diagram Alur dan Bahasa Basic Dasar*. Depok: Gunadarma
- Tosin, Rijianto. (1997). *Flowchart untuk siswa dan mahasiswa*. Jakarta: Dinastindo.
- Utami, E. (2005). *10 Langkah Belajar Logika dan Algoritma. menggunakan Bahasa C dan C++ di GnuLinux*. Penerbit Andi.
- Yatini, B. Indra. (2001), *Pemrograman Terstruktur*. Yogyakarta: J & J Learning.
- Yevick, D. (2005). *A First Course in Computational Physics and Object-Oriented Programming with C++ Hardback with CD-ROM*. Cambridge University Press.

TENTANG PENULIS

Shinta Esabella



Shinta Esabella, lahir di Sumbawa Besar pada 14 Juli 1986, menyelesaikan gelar Sarjana Teknik Informatika (S.T) di Sekolah Tinggi Teknik (STT) PLN Jakarta tahun 2008. Meraih gelar Magister Teknik Informatika (M.TI) di Universitas Bina Nusantara Jakarta pada tahun 2015. Memiliki minat bidang Pengembangan Rekayasa Perangkat Lunak serta Algoritma Pemrograman. Pernah bekerja menjadi staff di PT. PLN (Persero) Cabang Sumbawa (2009-2010). Menjadi Dosen di Universitas Cordova Indonesia, Kabupaten Sumbawa Barat (2010-2014). Dosen Program Studi Teknik Informatika Universitas Teknologi Sumbawa (2015 hingga sekarang). Menjadi Tim *Smart City* Kabupaten Sumbawa dan Anggota Relawan TIK Cabang Sumbawa. Pernah meraih beberapa penghargaan, 1) Penghargaan Sebagai Tim Penyusun *Masterplan Smart City* Kabupaten Sumbawa Tahun 2018, 2) Peneliti Terbaik dalam Seminar Hasil Penelitian Dosen Pemula Tahun 2019, 3) Pemakalah Terbaik dalam Kegiatan Seminar Nasional dengan Tema “Inovasi Hasil Penelitian dan Pengabdian Kepada Masyarakat dalam Menunjang Era Industri 4.0” Tahun 2020. Pernah melakukan penelitian pendanaan Kementrian RISTEKDIKTI 2018 dengan judul “Aplikasi Ensiklopedia Kebudayaan Sumbawa Berbasis Web”. Penelitian Kementrian RISTEKDIKTI 2019 dengan judul “Aplikasi Portal Oleh-oleh Khas Sumbawa”. Dan penelitian Pendanaan Ideathon Indonesia KEMENTRIAN RISTEK-BRIN 2020 dengan judul “*Gig Economy VS COVID-19*”. Akses link portfolio lengkap: <https://www.linkedin.com/in/shinta-esabella/>.

TENTANG PENULIS

Miftahul Haq



Miftahul Haq, anak ke-dua dari empat bersaudara. Kelahiran Bekasi, pada 28 Desember 1997. Seorang Santri Pondok Modern Darussalam Gontor – Ponorogo, Jawa Timur. Lulus dari pondok tahun 2016. Melanjutkan studi di UNIDA (Universitas Darussalam Gontor) mengambil Program Studi Ilmu Komunikasi Fakultas Humaniora hingga 2017. Melanjutkan kuliah di UTS (Universitas Teknologi Sumbawa) dengan Program Studi Teknik Informatika, Fakultas Teknik. Mengambil fokus peminatan studi di bidang RPL (Rekayasa Perangkat Lunak). Memiliki hobi di bidang desain grafis visual. Pekerjaan selain menjadi mahasiswa, adalah menjadi staff direksi Jurnal Hexagon (Jurnal Teknik dan Sains) Fakultas Teknik UTS, dan pekerjaan sampingan menjadi *freelance graphics designer*. Pengalaman organisasi, menjadi Staff Kominfo BEM (Badan Eksekutif Mahasiswa) UTS periode 2019-2020. Pengalaman di perkuliahan, menjadi Asisten Dosen mata kuliah Dasar-dasar Pemrograman dari tahun 2018 hingga 2021. Memiliki sertifikasi lulus kursus Dicoding ([dicoding.com](https://www.dicoding.com)) 1) Belajar Membuat Aplikasi Android untuk Pemula, 2) Belajar Dasar Pemrograman Web, 3) Belajar Dasar-Dasar Azure Cloud, 4) Memulai Pemrograman Dengan Dart. Akses link portfolio lengkap: <https://www.linkedin.com/in/mfth12/>.

DASAR-DASAR PEMROGRAMAN

BUKU INI MERUPAKAN SALAH SATU MEDIA BELAJAR PENDUKUNG UNTUK MEMPERKUAT MATA KULIAH DASAR-DASAR PEMROGRAMAN YANG DIAJARKAN DI KELAS SECARA TEORI DAN PRAKTIK. BUKU INI MEMUAT KUMPULAN MATERI, DASAR LOGIKA BERPIKIR, PRAKTIKUM, DAN SOAL EVALUASI. DENGAN ADANYA BUKU INI, DIHARAPKAN MAHASISWA DAPAT DENGAN MUDAH MEMPELAJARI, MEMAHAMI, DAN MEMPRAKTIKAN MATERI-MATERI YANG TELAH DIAJARKAN PADA MATA KULIAH DASAR-DASAR PEMROGRAMAN.