

Hanya dipergunakan di lingkungan Fakultas Teknik Elektro



MODUL PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN

Lab RAID
S1 Teknik Komputer
Fakultas Teknik Elektro

DAFTAR PENYUSUN

- Casi Setianingsih, S.T., M.T
- Burhanuddin Dirgantoro, Ir. MT
- Ashri Dinimaharawati, ST. MT
- Asisten Laboratorium Software Engineering and Application

LEMBAR PERNYATAAN

Yang bertanda dibawah ini:

Nama : Casi Setianingsih, S.T., M.T.

NIP : 19890019

Jabatan : Dosen Pembina Laboratorium *Realm of Artifical Intelligence and Design*

Menerangkan dengan sesungguhnya bahwa modul praktikum ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Akademik 2022/2023 di Laboratorium *Realm of Artifical Intelligence and Design* Fakultas Teknik Elektro Universitas Telkom.

Bandung, 2 Februari 2023


Mengetahui,
Ketua Kelompok Keahlian



Dr. Yudha Purwanto, S.T., M.T

NIP. 02770066

Dosen Pembina Laboratorium *Realm of Artifical Intelligence and Design*



Casi Setianingsih, S.T., M.T

NIP. 19890019

STRUKTUR ORGANISASI
LABORATORIUM SOFTWARE ENGINEERING AND
APPLICATION

TAHUN AJARAN 2022/2023

Dosen Pembina Lab:

Casi Setianingsih, S.T., M.T	NIP. 19890019
------------------------------	---------------

Koordinator Asisten Praktikum:

Harvan Nurluthfi Irawan	1103204038
-------------------------	------------

Asisten Praktikum:

Evan Pradipta Hardinatha	1103204160
--------------------------	------------

Gunawan Tri Mardani	1103201261
---------------------	------------

Ibrahim Maulana	1103201247
-----------------	------------

Jaisy Malikulmulki Arasy	1103202201
--------------------------	------------

Mohammad Rizki Ramdhan	1103240126
------------------------	------------

Ario Syawal Muhammad	1103201243
----------------------	------------

Jean Jeasenn Timotius Zipazi	1103201257
------------------------------	------------

Zaidan Luthfi	1103203238
---------------	------------

Giovanni Nathaniel	1103202211
--------------------	------------

Muhammad Tharreq An Nahl	1103204040
--------------------------	------------

Muhammad Irfan Al Rasyid	1103200080
Rizky Ramadhani Syam	1103204086
Achmad Rionov Faddillah Ramadhan	1103204030
Edwin Malik Makarim	1103202079

Visi & Misi

Fakultas Teknik Elektro

VISI:

Menjadi fakultas berstandar internasional yang berperan aktif dalam pengembangan pendidikan, riset, dan entrepreneurship di bidang teknik elektro dan teknik fisika, berbasis teknologi informasi.

MISI:

1. Menyelenggarakan sistem pendidikan yang berstandar internasional di bidang teknik elektro dan teknik fisika berbasis teknologi informasi.
2. Menyelenggarakan, menyebarluaskan, dan memanfaatkan hasil-hasil riset berstandar internasional di bidang teknik elektro dan fisika.
3. Menyelenggarakan program entrepreneurship berbasis teknologi bidang teknik elektro dan teknik fisika di kalangan sivitas akademika untuk mendukung pembangunan ekonomi nasional.
4. Mengembangkan jejaring dengan perguruan tinggi dan industri terkemuka dalam dan luar negeri dalam rangka kerjasama pendidikan, riset, dan entrepreneurship.
5. Mengembangkan sumberdaya untuk mencapai keunggulan dalam Pendidikan, riset, dan entrepreneurship.

Visi & Misi

Jurusan Teknik Komputer

VISI:

“Menjadi Program Studi S1 Teknik Komputer berstandar internasional yang menghasilkan lulusan di bidang komputer.”

MISI:

1. Menyelenggarakan Pendidikan yang berstandar internasional untuk menghasilkan lulusan yang menguasai ilmu dan teknologi komputer.
2. Menyelenggarakan penelitian berkualitas internasional di bidang Sistem Komputer berbasis teknologi ilmu dan teknologi komputer dengan melibatkan mahasiswa secara aktif.
3. Menjalankan pengabdian masyarakat dengan prinsip menyebarluaskan ilmu dan teknologi komputer hasil penelitian kepada masyarakat luas dengan melibatkan mahasiswa secara aktif.
4. Membekali mahasiswa ilmu dan pengetahuan yang praktis, agar mampu bekerja, dan mengembangkan diri dan berwirausaha di bidang teknologi informasi dan komunikasi komputer.



ATURAN LABORATORIUM FAKULTAS TEKNIK ELEKTRO TELKOM UNIVERSITY

Setiap Mahasiswa Fakultas Teknik Elektro yang akan menggunakan Fasilitas Laboratorium, **WAJIB**

Mematuhi Aturan sebagai berikut :

1. Menggunakan seragam resmi Telkom University, dan membawa Kartu Tanda Mahasiswa (KTM) yang masih berlaku.
2. Rambut rapih.
3. Dilarang **merokok** dan **makan minum** didalam ruangan, dan membuang sampah ada tempatnya.
4. Dilarang menyimpan barang-barang milik pribadi di Laboratorium tanpa seijin Fakultas.
5. Dilarang menginap di Laboratorium tanpa seijin Fakultas.
6. Jam Kerja Laboratorium dan Ruang Riset adalah 06.30 WIB sampai 22.00 WIB.
7. Mahasiswa yang akan menggunakan Laboratorium dan atau ruang riset diluar jam kerja, harus mengajukan ijin kepada Fakultas.

Bandung, 2 Februari 2023

Dekan Fakultas Teknik Elektro

Dr. Bambang Setia Nugroho, S.T., M.T.

DAFTAR ISI

DAFTAR PENYUSUN	1
LEMBAR PERNYATAAN	2
STRUKTUR ORGANISASI	3
ATURAN LABORATORIUM FAKULTAS TEKNIK ELEKTRO.....	7
DAFTAR ISI.....	8
DAFTAR GAMBAR	10
DAFTAR TABEL.....	12
Modul 1 Konsep Algoritma.....	14
1.1 Tujuan:.....	14
1.2 Alat dan Bahan:	14
1.3 Dasar Teori	14
1.3.1 Mengenal Algoritma dan Pseudocode	14
1.3.2 Struktur Bahasa Pemrograman C	15
1.3.3 Variabel dan Konstanta	17
1.3.4 Tipe Data.....	18
1.3.5 Tipe Data Bentuk.....	22
1.3.6 String	23
1.3.7 Operator Aritmatika	24
1.3.8 Komentar.....	26
1.3.9 Contoh Program	27
1.3.10 Format Tipe Data	31
1.3.11 If.....	33
1.3.12 If - else.....	35
1.3.13 If – else if	36
1.3.14 Nested If.....	37
1.3.15 Switch Case.....	39
Modul 2 Array	41
2.1 Tujuan praktikum	41
2.2 Alat dan Bahan	41
2.3 Dasar Teori	41
2.3.1 Definisi.....	41
2.3.2 Deklarasi	42

2.3.3	Jenis-jenis Array.....	43
2.3.4	Looping Array	45
Modul 3	Prosedur, Fungsi, dan Pointer	48
3.1	Tujuan Praktikum	48
3.2	Alat dan Bahan	48
3.3	Dasar Teori	48
3.3.1	Pointer	48
3.3.2	Fungsi.....	51
3.3.3	Prosedur	53
Modul 4	Sorting.....	58
4.1	Tujuan.....	58
4.2	Alat dan Bahan	58
4.3	Dasar Teori	58
4.3.1	Bubble Sort	58
4.3.2	Insertion Sort.....	61
4.3.3	Selection Sort	64
Modul 5	Searching	67
5.1	Tujuan Praktikum	67
5.2	Alat dan Bahan	67
5.3	Dasar Teori	67
5.3.1	Linear Search	67
5.3.2	Binary Search	69
Modul 6	Operasi File.....	73
6.1	Tujuan Praktikum	73
6.2	Alat dan Bahan	73
6.3	Dasar Teori	73
6.3.1	Pengertian File	73
6.3.2	Operasi File	73
6.3.3	CRUD.....	77
Daftar Pustaka	80

DAFTAR GAMBAR

Gambar 1.1 Output.....	30
Gambar 1.2 Source Code Penggunaan Konstanta.....	31
Gambar 1.3 Output Penggunaan Konstanta	31
Gambar 1.4 Syntax If	34
Gambar 1.5 Contoh If	34
Gambar 1.6 Output Program If	34
Gambar 1.7 Syntax if-else.....	35
Gambar 1.8 Source Code If-else	35
Gambar 1.9 Output If-else.....	36
Gambar 1.10 Syntax If-Else if	36
Gambar 1.11 Source Code If-Else if	37
Gambar 1.12 Output If-else if	37
Gambar 1.13 Syntax Nested If.....	38
Gambar 1.14 Source Code Nested If	38
Gambar 1.15 Output Nested if	38
Gambar 1.16 Syntax Switch Case.....	39
Gambar 1.17 Source Code Switch	39
Gambar 1.18 Output Switch.....	39
Gambar 2.1 Tabel Array	42
Gambar 2.2 Deklarasi Array	42
Gambar 2.3 <i>Syntax Array</i> Satu Dimensi	43
Gambar 2.4 <i>Source Code</i> Progam Array Satu Dimensi.....	43
Gambar 2.5 <i>Output</i> Program Array Satu Dimensi	44
Gambar 2.6 <i>Syntax Array Dua Dimensi</i>	44
Gambar 2.7 <i>Analogi Array Dua Dimensi</i>	44
Gambar 2.8 <i>Source Code</i> Program Array Dua Dimensi	45
Gambar 2.9 <i>Output</i> Program Array Satu Dimensi	45
Gambar 2.10 <i>Source Code</i> Program <i>Looping Array</i>	46
Gambar 2.11 <i>Output</i> Program <i>Looping Array</i>	47
Gambar 3.1 <i>Deklarasi Pointer</i>	48
Gambar 3.2 <i>Source Code Pointer Ampersand</i>	49
Gambar 3.3 Output Pointer Ampersand.....	49
Gambar 3.4 <i>Source Code</i> Nilai Alamat pada <i>Pointer</i>	49
Gambar 3.5 Output Nilai Alamat pada <i>Pointer</i>	50
Gambar 3.6 <i>Source Code Pointer Lanjutan</i>	50
Gambar 3.7 <i>Output</i> Contoh <i>Pointer Lanjutan</i>	50
Gambar 3.8 Deklarasi sebuah fungsi	51
Gambar 3.9 Syntax Memanggil Sebuah Fungsi.....	52
Gambar 3.10 Source Code Penggunaan Fungsi.....	52
Gambar 3.11 <i>Output</i> Penggunaan Fungsi	53
Gambar 3.12 Syntax prosedur tanpa parameter	53
Gambar 3.13 Syntax prosedur dengan parameter	54
Gambar 3.14 <i>Source Code</i> Prosedur Parameter Masukan	55

Gambar 3.15 <i>Output</i> Prosedur Parameter Masukan.....	55
Gambar 3.16 <i>Source Code</i> Prosedur Parameter Keluaran	56
Gambar 3.17 <i>Output</i> Prosedur Parameter Keluaran.....	56
Gambar 3.18 <i>Source Code</i> Prosedur Parameter	57
Gambar 3.19 <i>Output</i> Prosedur Parameter Masukan dan keluaran	57
Gambar 4.1 Coding Bubble Sort.....	60
Gambar 4.2 Output Coding Bubble Sort.....	61
Gambar 4.3 Output Coding Selection Sort	66
Gambar 4.4 Output Coding Selection Sort	66
Gambar 5.1 Ilustrasi Linear Search.....	68
Gambar 5.2 Contoh program yang menggunakan linear search	68
Gambar 5.3 Jika data ditemukan	69
Gambar 5.4 Langkah pertama contoh binary search.....	69
Gambar 5.5 Langkah kedua contoh binary search	70
Gambar 5.6 Langkah ketiga contoh binary search	70
Gambar 5.7 contoh program dengan binary search	71
Gambar 5.8 Jika nilai ada.....	71
Gambar 5.9 Jika nilai tidak ditemukan	72
Gambar 6.1 Contoh Membuka File.....	74
Gambar 6.2 Contoh Menulis File.....	75
Gambar 6.3 Contoh Membaca File	75
Gambar 6.4 Contoh Bentuk String.....	75
Gambar 6.5 Program Menulis File.....	76
Gambar 6.6 Output Program Menulis File.....	76
Gambar 6.7 Contoh Algoritma CRUD	78
Gambar 6.8 Output CRUD.....	79

DAFTAR TABEL

Tabel 1.1 Struktur Bahasa Pemrograman C	15
Tabel 1.2 Source code Integer.....	18
Tabel 1.3 Tipe data integer, ukuran, dan panjangnya	18
Tabel 1.4 Source Code Float.....	19
Tabel 1.5 Source Code Double	20
Tabel 1.6 Tipe data double, ukuran, dan panjangnya	20
Tabel 1.7 Source Code Char	20
Tabel 1.8 Tipe data char, ukuran, dan panjangnya	21
Tabel 1.9 Source Code Boolean.....	21
Tabel 1.10 Penggunaan Struct.....	22
Tabel 1.11 Source Code String	24
Tabel 1.12 Source Code Perkalian	24
Tabel 1.13 Source Code Pembagian	25
Tabel 1.14 Source Code Penjumlahan	25
Tabel 1.15 Source Code Pengurangan	25
Tabel 1.16 Source Code Modulo	26
Tabel 1.17 Source Code Single-Line Comment	27
Tabel 1.18 Source Code Multi-Line Comment.....	27
Tabel 1.19 Output ke console.....	27
Tabel 1.20 Source Code Deklarasi dan Inisialisasi Variabel	28
Tabel 1.21 Mendapatkan input dari user.....	29
Tabel 1.22 Penggunaan Konstanta	30
Tabel 1.23 Format tipe data	32
Tabel 1.24 Operasi Logika.....	33
Tabel 1.25 Operasi Relasi	33
Tabel 1.26 Penggunaan If	34
Tabel 1.27 Source Code If-else	35
Tabel 1.28 Source Code If-Else if.....	36
Tabel 2.1 Contoh Deklarasi <i>Array</i>	42
Tabel 2.2 Program <i>Array</i> Satu Dimensi	43
Tabel 2.3 Program <i>Array</i> Dua Dimensi	45
Tabel 2.4 Program <i>Looping Array</i>	46
Tabel 3.1 <i>Pointer – Operator Ampersand</i>	49
Tabel 3.2 <i>Pointer – Nilai dari Suatu Alamat pada Pointer</i>	49
Tabel 3.3 <i>Pointer – Contoh Lanjutan</i>	50
Tabel 3.4 Contoh Penggunaan Fungsi	52
Tabel 3.5 Contoh Program Parameter Masukan	54
Tabel 3.6 Contoh Program Parameter Keluaran	55
Tabel 3.7 Contoh program parameter Masukan dan Keluaran	56

Tabel 4.1 Bubble Sort	58
Tabel 4.2 Coding Bubble Sort.....	60
Tabel 4.3 Insertion Sort.....	61
Tabel 4.4 Coding Insertion Sort.....	62
Tabel 4.5 Selection Sort	64
Tabel 4.6 Coding Selection Sort	65
Tabel 5.1 Contoh Program Linear Search.....	68
Tabel 5.2 <i>Contoh Program Binary Search</i>	71
Tabel 6.1 Mode atau operasi dalam file	74
Tabel 6.2 Contoh Program Write FILE	76

Modul 1 Konsep Algoritma

1.1 Tujuan:

1. Memahami konsep pseudocode, ditambah algoritma pemrograman dan implementasinya dalam Bahasa C.
2. Memahami tipe data pada algoritma pemrograman dan implementasinya dalam Bahasa C.
3. Memahami konsep percabangan dalam pemrograman
4. Membuat program sederhana menggunakan percabangan dalam Bahasa pemrograman C
5. Memahami struktur perulangan dan dapat membuat program sederhana dengan konsep tersebut dalam Bahasa pemrograman C

1.2 Alat dan Bahan:

1. PC yang sudah terinstall Code Blocks

1.3 Dasar Teori

1.3.1 Mengenal Algoritma dan Pseudocode

Algoritma (algorithm) adalah sekumpulan langkah-langkah terurut yang dilakukan oleh sebuah program komputer untuk menyelesaikan sebuah permasalahan tertentu. Dalam pemrograman, algoritma didefinisikan sebagai metode yang terdiri dari langkah-langkah terstruktur untuk mencari solusi suatu masalah dengan bantuan komputer. Tahapan dalam menyelesaikan permasalahan menggunakan algoritma adalah terdiri dari tiga bagian yaitu menentukan permasalahan (ide), pemecahan masalah, dan solusi (hasil).

Algoritma dapat disajikan dalam dua bentuk, yaitu pseudocode dan flowchart. Pseudocode sendiri merupakan penyajian algoritma yang informal dan dapat dibuat dengan kaidah yang ditentukan sendiri. Dengan kata lain, pseudocode merupakan urutan logika yang bertujuan untuk dipahami manusia dengan mudah.

Pseudocode bukan merupakan bahasa pemrograman, sehingga Bahasa yang kita gunakan untuk menulis tidak menjadi masalah. Umumnya,

pseudocode ditulis dengan bahasa Inggris untuk memudahkan konversi ke bahasa pemrograman. Namun, Bahasa Indonesia pun dapat digunakan.

```
begin
  numeric nCode
  display "ENTER THE DAY CODE : "
  accept nCode
  switch(nCode)
  begin
    case 1 : display "MONDAY"
      break;
    case 2 : display "TUESDAY"
      break;
    case 3 : display "WEDNESDAY"
      break;
    case 4 : display "THURSDAY"
      break;
    case 5 : display "FRIDAY"
      break;
    case 6 : display "SATURDAY"
      break;
    case 7 : display "SUNDAY"
      break;
    default : display "OUT OF RANGE"
  end
end
```

Gambar 1.1 Contoh Pseudocode

1.3.2 Struktur Bahasa Pemrograman C

Sebagai permulaan, kita akan berkenalan terlebih dahulu dengan struktur dasar dari sebuah program yang ditulis menggunakan bahasa pemrograman C. Strukturnya adalah sebagai berikut:

Tabel 1.1 Struktur Bahasa Pemrograman C

Source Code



```
// Standard I/O Library (Wajib)
#include <stdio.h>
//Standard Library (Wajib)
#include <stdlib.h>

// Main Functions -> Program Utama
int main(){
    // Statement
    printf("Hello World!\n");
    //Terminator
    return 0;
}
```

Gambar 1.2 Struktur Bahasa Pemrograman C

Penjelasan:

Pada source code di atas, ada beberapa bagian yang harus ada dalam program dengan bahasa C, antara lain:

a. Library

Library adalah sekumpulan fungsi yang sudah built-in dalam bahasa pemrograman C yang digunakan untuk menjalankan sebuah tugas tertentu. Contohnya adalah fungsi `printf()`. Fungsi `printf()` diperoleh dari library `stdio.h`. Jika library tersebut tidak kita memasukkan (`include`) ke dalam program, maka program akan error.

b. Main Function

Main function atau fungsi utama adalah salah satu hal penting yang harus ada dalam program. Main function adalah

entry point (gerbang masuk) dari sebuah program. Saat program di-*compile*, *compiler* akan mencari **main function** terlebih dahulu untuk dapat masuk dan meng-**compile** program menjadi **bahasa mesin** dan akhirnya dapat dijalankan.

c. Statement

Statement adalah baris kode program yang kita tulis. Statement ini akan dijalankan secara urut dari atas ke bawah.

d. Terminator

Terminator (return 0) adalah sebagai penanda akhir dari sebuah program. Return 0 menandakan bahwa program kita berjalan tanpa adanya error.

1.3.3 Variabel dan Konstanta

Variabel adalah sebuah tempat yang digunakan untuk menyimpan sebuah nilai yang nantinya akan diolah dalam program. Nilai suatu **variabel** dapat berubah-ubah sewaktu-waktu selama program berjalan. Sedangkan **konstanta** adalah jenis variabel yang tidak dapat diubah selama program berjalan.

Ada beberapa hal yang harus diperhatikan saat membuat sebuah variabel, antara lain adalah sebagai berikut:

- a. Harus diawali dengan huruf alfabet (A-Z, a-z)
- b. Dapat berupa huruf, digit atau karakter garis bawah (_)
- c. Panjang maksimal adalah 32 karakter, jika lebih maka yang dianggap adalah 32 karakter awal
- d. Tidak boleh menggunakan spasi
- e. Tidak boleh menggunakan operator aritmatika (+ - / * %)
- f. Tidak boleh menggunakan karakter-karakter khusus seperti : , ; # @ \$ & dan ()

Perhatikan, bahasa C bersifat **case-sensitive**, sehingga huruf kapital dan non-kapital dibedakan. Contohnya ketika anda

membuat variabel dengan nama name dan Name, kedua variabel tersebut adalah variabel yang berbeda.

1.3.4 Tipe Data

Tipe data adalah sesuatu yang menentukan jenis nilai yang dapat ditampung oleh sebuah variabel. Berikut adalah beberapa tipe data yang dapat digunakan dalam bahasa pemrograman C, antara lain:

a. Integer

Integer adalah tipe data yang dapat menampung bilangan bulat. Tipe data integer memiliki ukuran 32 bit. Contoh:

Tabel 1.2 Source code Integer

Source Code
The image shows a screenshot of a code editor with a dark background. At the top left, there are three colored circles (red, yellow, green). Below them, there are three lines of C code: <code>int puluhan = 10;</code> , <code>int ratusan = 100;</code> , and <code>int negatif = -5;</code> . The code is color-coded: <code>int</code> is blue, <code>puluhan</code> , <code>ratusan</code> , and <code>negatif</code> are green, and the values <code>10</code> , <code>100</code> , and <code>-5</code> are red.
Gambar 1.3 Source code Integer

Tipe data integer dibagi lagi berdasarkan ukuran dan rentang nilai yang dapat disimpan. Perhatikan tabel berikut!

Tabel 1.3 Tipe data integer, ukuran, dan panjangnya

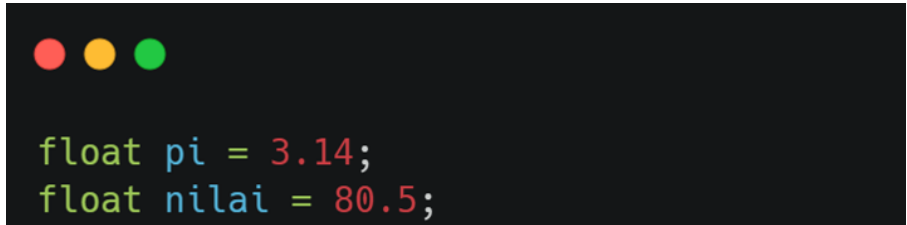
Tipe Data	Ukuran	Panjang
int	16 atau 32	-32.767 sampai 32.767
unsigned int	16 atau 32	0 sampai 65.535
signed int	16 atau 32	-32.767 sampai 32.767
short int	16	-32.767 sampai 32.767

unsigned short int	16	0 sampai 65.535
signed short int	16	-32.767 sampai 32.767
long int	32	-2.147.483.647 sampai 2.147.483.647
long long int	64	$-(2^{63}-1)$ to $2^{63}-1$
signed long int	32	-2.147.483.647 sampai 2.147.483.647
unsigned long int	32	0 sampai 4.294.967.295
unsigned long long int	64	$2^{64}-1$

b. Float

Float adalah tipe data yang dapat menampung bilangan desimal atau pecahan. Tipe data float memiliki ukuran 32 bit. Contoh:

Tabel 1.4 Source Code Float

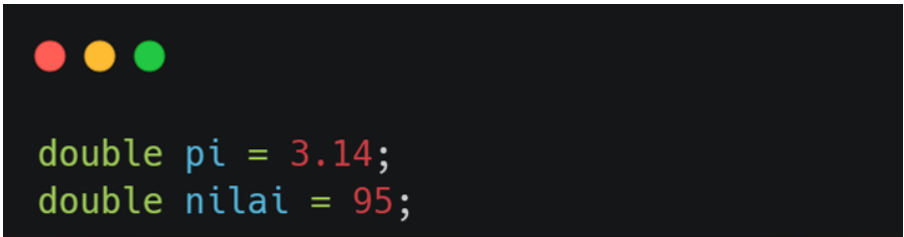
Source Code
 <pre>float pi = 3.14; float nilai = 80.5;</pre>

Gambar 1.4 Source Code Float

c. Double

Double adalah tipe data yang dapat menampung bilangan desimal atau pecahan. Tipe data double hampir sama dengan tipe data float, yang membedakannya hanya ukuran data yang dapat disimpan.

Tabel 1.5 Source Code Double

Source Code
 <pre>double pi = 3.14; double nilai = 95;</pre>
Gambar 1.5 Source Code Double

Tipe data double dibagi lagi berdasarkan ukuran dan rentang nilai yang dapat disimpan. Perhatikan tabel berikut!

Tabel 1.6 Tipe data double, ukuran, dan panjangnya

Tipe Data	Ukuran	Panjang
double	64	IE-37 sampai IE+3737 dengan 6 digit angka presisi
long double	80	IE-37 sampai IE+3737 dengan 6 digit angka presisi

d. Char

Char adalah tipe data yang dapat menampung sebuah karakter. Berikut adalah contoh penggunaan tipe char:

Tabel 1.7 Source Code Char

Source Code



```
char a = 'A';  
char f = 'F';  
char num = '5';
```

Gambar 1.6 Source Code Char

Tipe data char dibagi lagi berdasarkan ukuran dan rentang nilai yang dapat disimpan. Perhatikan tabel berikut!

Tabel 1.8 Tipe data char, ukuran, dan panjangnya

Tipe Data	Ukuran	Panjang
char	8	-127 sampai 127
unsigned char	8	0 sampai 255
signed char	8	-127 sampai 127

e. Boolean

Boolean adalah sebuah tipe data yang hanya memiliki dua macam nilai, yaitu True atau False. Tipe data boolean biasanya digunakan untuk membuat logika dalam program. Contoh:

Tabel 1.9 Source Code Boolean

Source Code



```
bool alwaysTrue = true;
bool alwaysFalse = false;
```

Gambar 1.7 Source Code Boolean

1.3.5 Tipe Data Bentukan

Tipe bentukan adalah tipe data yang dibuat sendiri oleh programmer. Tipe ini dibuat karena ada relasi antar variabel yang bila digabungkan mempunyai suatu maksud yang sama. Untuk membuat tipe data bentukan, digunakan kata kunci struct. Salah satu contoh dari tipe data bentukan adalah mahasiswa, dimana mahasiswa mempunyai nama, nim, jurusan, dan nilai. Berikut adalah contohnya:

Tabel 1.10 Penggunaan Struct

Source Code

```

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

struct Mahasiswa {
    char name[50];
    int nim;
    char jurusan[50];
    int nilai;
} mahasiswa;

// Program Utama
int main(){
    //Implementasi Struct
    strcpy(mahasiswa.name, "Rizky");
    mahasiswa.nim = 110314045;
    strcpy(mahasiswa.jurusan, "Teknik Komputer");
    mahasiswa.nilai = 90;
    //Output
    printf("%s\n%d\n%s\n%d\n", mahasiswa.name, mahasiswa.nim,
        mahasiswa.jurusan, mahasiswa.nilai);
    //Terminator
    return 0;
}

```

Gambar 1.8 Source Code Penggunaan Struct

Output

```

Rizky
110314045
Teknik Komputer
90

```

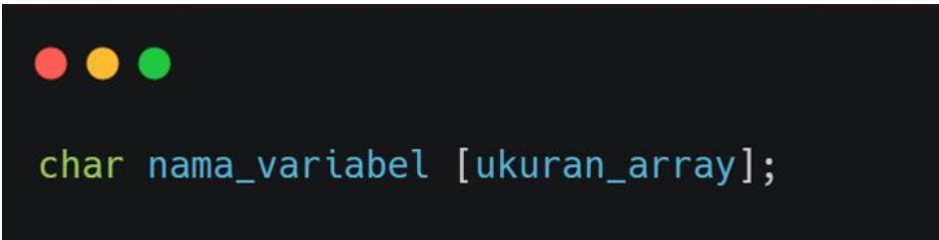
Gambar 1.9 Output Penggunaan Struct

1.3.6 String

String merupakan kumpulan dari beberapa karakter. Jadi bisa dikatakan bahwa string bukan merupakan sebuah tipe data. Dalam bahasa pemrograman C, string memang tidak dikenal. Namun, kita bisa membuat string dengan menggunakan tipe data char. Untuk lebih lengkapnya, string akan dibahas pada modul 10.

Pendeklarasian string dapat dilakukan dengan cara sebagai berikut:

Tabel 1.11 Source Code String


Source Code

Gambar 1.10 Source Code String

1.3.7 Operator Aritmatika

Operator aritmatika ini dikhususkan untuk tipe data int, float, dan double. Berikut adalah beberapa **operator aritmatika** yang dapat digunakan pada bahasa pemrograman C, antara lain:


a. Multiplication (Perkalian | *)

Tabel 1.12 Source Code Perkalian

Source Code

Gambar 1.11 Source Code Perkalian

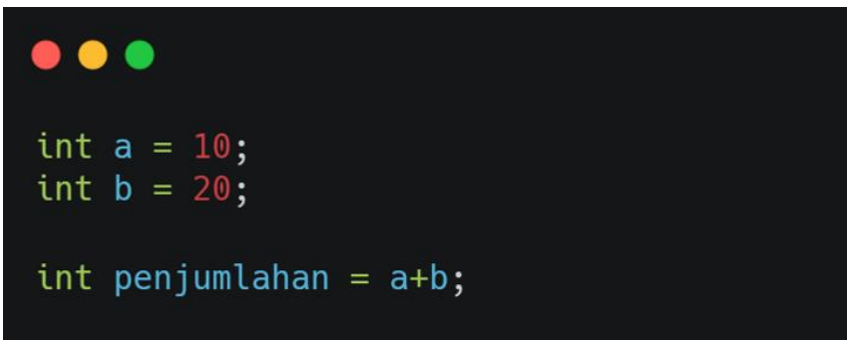
b. Division (Pembagian | /)

Tabel 1.13 Source Code Pembagian

Source Code
 <pre>int a = 10; int b = 20; int pembagian = a/b;</pre>
Gambar 1.12 Source Code Pembagian

c. Addition (Penjumlahan | +)

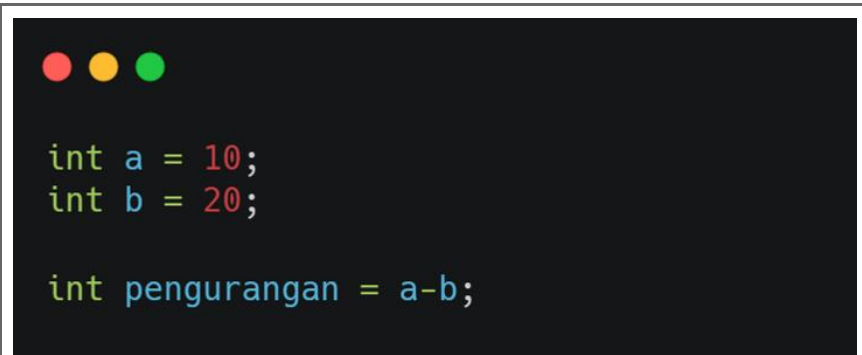
Tabel 1.14 Source Code Penjumlahan

Source Code
 <pre>int a = 10; int b = 20; int penjumlahan = a+b;</pre>
Gambar 1.13 Source Code Penjumlahan

d. Subtraction (Pengurangan | -)

Tabel 1.15 Source Code Pengurangan

Source Code



```

int a = 10;
int b = 20;


int pengurangan = a-b;

```

Gambar 1.14 Source Code Pengurangan

e. Modulo (Sisa Hasil Bagi | %)

Tabel 1.16 Source Code Modulo

Source Code
 <pre> int a = 10; int b = 20; int modulus = a%b; </pre>


Gambar 1.15 Source Code Modulo

1.3.8 Komentar

Komentar adalah sebuah fitur dalam hampir semua bahasa pemrograman termasuk bahasa pemrograman C. Komentar dibuat untuk membantu programmer dalam mendokumentasikan kode yang mereka tulis agar dapat lebih mudah untuk dibaca. Komentar pun akan sangat berguna ketika anda bekerja dengan tim. Komentar dalam bahasa pemrograman C ada dua macam, yaitu:


a. Single-Line Comment

Tabel 1.17 Source Code Single-Line Comment

Source Code
 <p>Gambar 1.16 Source Code Single-Line Comment</p>

b. Multi-Line Comment

Tabel 1.18 Source Code Multi-Line Comment

Source Code
 <p>Gambar 1.17 Source Code Multi-Line Comment</p>

1.3.9 Contoh Program

Berikut adalah beberapa referensi contoh program yang dapat anda implementasikan dan pelajari selama modul ini:

Tabel 1.19 Output ke console

Source Code



```
// Standard I/O Library (Wajib)
#include <stdio.h>
//Standard Library (Wajib)
#include <stdlib.h>

// Main Functions -> Program Utama
int main(){
    // Statement
    printf("Hello World!\n");
    //Terminator
    return 0;
}
```

Gambar 1.18 Output ke console

Keterangan:

Untuk menampilkan output ke console, kita dapat menggunakan fungsi printf(“output yang ingin ditampilkan”). Jangan lupa memberikan semicolon (;) di setiap akhir baris dalam program.

b. Deklarasi dan inisialisasi variabel

Tabel 1.20 Source Code Deklarasi dan Inisialisasi Variabel

Source Code



```
int nilai; //deklarasi
nilai = 100
```

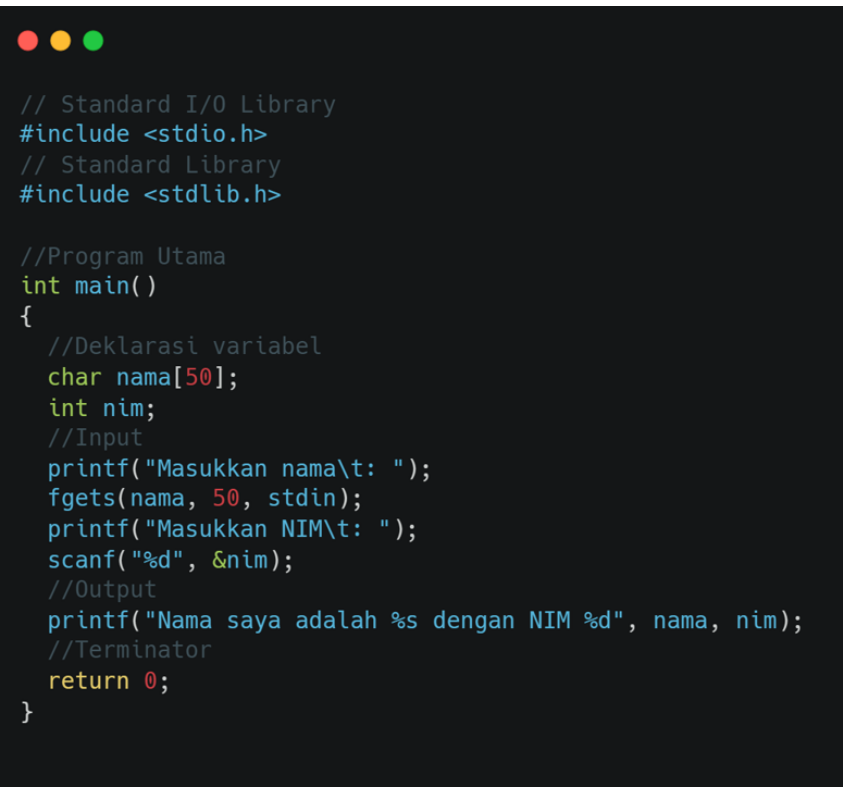
Gambar 1.19 Source Code Deklarasi dan Inisialisasi Variabel

Keterangan:

- Deklarasi : membuat dan mendaftarkan variabel baru dalam program.
- Inisialisasi : memberikan nilai awal untuk sebuah variabel dalam program.

c. Mendapatkan input dari user

Tabel 1.21 Mendapatkan input dari user

Source Code
 <pre> // Standard I/O Library #include <stdio.h> // Standard Library #include <stdlib.h> //Program Utama int main() { //Deklarasi variabel char nama[50]; int nim; //Input printf("Masukkan nama\t: "); fgets(nama, 50, stdin); printf("Masukkan NIM\t: "); scanf("%d", &nim); //Output printf("Nama saya adalah %s dengan NIM %d", nama, nim); //Terminator return 0; } </pre>
Gambar 1.20 Source Code untuk Mendapatkan input dari user
Input
SEA Lab 135623446
Output

```
Masukkan nama    : SEA Lab
Masukkan NIM     : 135623446
Nama saya adalah SEA Lab
dengan NIM 135623446
```

Gambar 1.1 Output

Keterangan:

- Untuk mendapatkan input berupa String dari user, kita dapat menggunakan fungsi `fgets(nama_variabel, ukuran, stdin)`. Jangan lupa untuk include library `string.h` terlebih dahulu.
- Untuk mendapatkan input berupa tipe data selain String (contoh: `int`, `float`, `double`, `char`, `boolean`), kita dapat menggunakan fungsi `scanf("%tipe_data", &variabel)`. Jangan lupa untuk include library `stdio.h` dan `stdlib.h` terlebih dahulu.

d. Penggunaan konstanta

Tabel 1.22 Penggunaan Konstanta

Source Code

```

// Standard I/O Library
#include <stdio.h>
// Mendefinisikan konstanta
#define phi 3.14

//Program Utama
int main()
{
    //Deklarasi variabel
    int r;
    float luas;
    //Input
    printf("Jari-jari lingkaran = ");
    scanf("%d", &r);
    //Output
    luas = phi*r*r;
    printf("Luas Lingkaran = %.2f\n", luas);
    //Terminator
    return 0;
}

```

Gambar 1.2 Source Code Penggunaan Konstanta

Output

```

Jari-jari lingkaran = 7
Luas Lingkaran = 153.86

```

Gambar 1.3 Output Penggunaan Konstanta

Keterangan:

- Syntax `#define phi 3.14` adalah cara untuk membuat sebuah konstanta dalam bahasa pemrograman C.
- Syntax `%x.f` berfungsi untuk menampilkan x angka di belakang koma dari variabel keluaran. Pada contoh diatas, `printf("Luas Lingkaran = %.2f \n", luas);` akan menampilkan 2 angka di belakang koma.

1.3.10 Format Tipe Data

Berikut adalah daftar format tipe data yang dapat membantu anda dalam menulis program pada bahasa pemrograman C, antara lain:

Tabel 1.23 Format tipe data

Tipe Data	Penentu Format
Integer	%d, %i (iterasi)
Floating Point <ul style="list-style-type: none"> - Bentuk desimal - Bentuk pangkat - Bentuk lebih pendek dari desimal dan pangkat 	%f %e %g
Double precision	%lf
Character	%c
String	%s
Unsigned integer	%u
Long integer	%ld
Long unsigned integer	%lu
Unsigned hexadecimal integer	%x
Unsigned octal integer	%o

Seleksi Kondisional

Dalam kehidupan sehari-hari, untuk melakukan sesuatu pasti akan dihadapkan dengan sebuah atau beberapa keputusan. Di dalam pemrograman bahasa C, keputusan disebut juga dengan fungsi

kondisional. Fungsi kondisional menggunakan operasi **boolean** (*true* atau *false*), operasi logika, dan operasi relasi untuk menentukan aksi (keputusan) yang akan dilakukan oleh program.

Tabel 1.24 Operasi Logika

Operasi Logika			
Program	Keterangan	Contoh	Operasi Boolean
&&	Logika AND	(6>5) && (4<5)	TRUE
	Logika OR	(11>=10) (10>=10)	TRUE
!	Logika NOT	!((6>5) && (4<5))	TRUE

Tabel 1.25 Operasi Relasi

Operasi Relasi			
Program	Keterangan	Contoh	Operasi Boolean
>	lebih besar	7 > 6	TRUE
<	lebih kecil	6 < 7	TRUE
>=	lebih besar sama dengan	7 >= 7	TRUE
<=	lebih kecil sama dengan	6 <= 6	TRUE
==	sama dengan	5 == 5 atau 'a' == 'a'	TRUE
!=	tidak sama dengan	5 != 4 atau 'a' != 'b'	TRUE

1.3.11 If

Digunakan untuk menjalankan suatu instruksi apabila kondisi terpenuhi. Jika kondisi tersebut bernilai benar, maka statement tersebut akan dijalankan. Jika kondisi tersebut bernilai salah, maka statement

tersebut akan diabaikan dan program berlanjut ke instruksi selanjutnya.
Penulisannya adalah sebagai berikut

```
if(kondisi){  
    /*statement atau aksi*/  
}
```

Gambar 1.4 Syntax If

Contoh :

Tabel 1.26 Penggunaan If

Source Code
<pre>#include <stdio.h> main(){ float nilai; printf("Masukkan nilai: "); scanf("%f", &nilai); if(nilai>65 && nilai<=100){ printf("SELAMAT, ANDA LULUS!"); } }</pre>
Gambar 1.5 Contoh If
Output
<pre>Masukan nilai :70 SELAMAT, ANDA LULUS!;</pre>
Gambar 1.6 Output Program If

1.3.12 If - else

Kita dapat menentukan apa yang akan dilakukan jika kondisi tersebut tidak terpenuhi menggunakan *else*. Jika kondisi tersebut bernilai salah, maka program akan menjalankan statement selain kondisi yang diinginkan.

```
if(kondisi){
    /*statement atau aksi*
}
else{
    /*statement atau aksi*
}
```

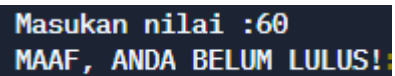
Gambar 1.7 Syntax if-else

Contoh :

Tabel 1.27 Source Code If-else

Source Code
<pre>#include <stdio.h> main(){ float nilai; printf("Masukkan nilai: "); scanf("%f", &nilai); if(nilai>65 && nilai<=100){ printf("SELAMAT, ANDA LULUS!"); } else{ printf("MAAF, ANDA BELUM LULUS!") } }</pre>
Output

Gambar 1.8 Source Code If-else



```
Masukan nilai :60
MAAF, ANDA BELUM LULUS!
```

Gambar 1.9 Output If-else

1.3.13 If – else if

If – else if digunakan untuk memeriksa beberapa kondisi. *Statement 2* akan dijalankan jika kondisi 1 tidak terpenuhi dan kondisi 2 terpenuhi. Namun jika semua kondisi tidak terpenuhi, maka program akan menjalankan *statement 3*.

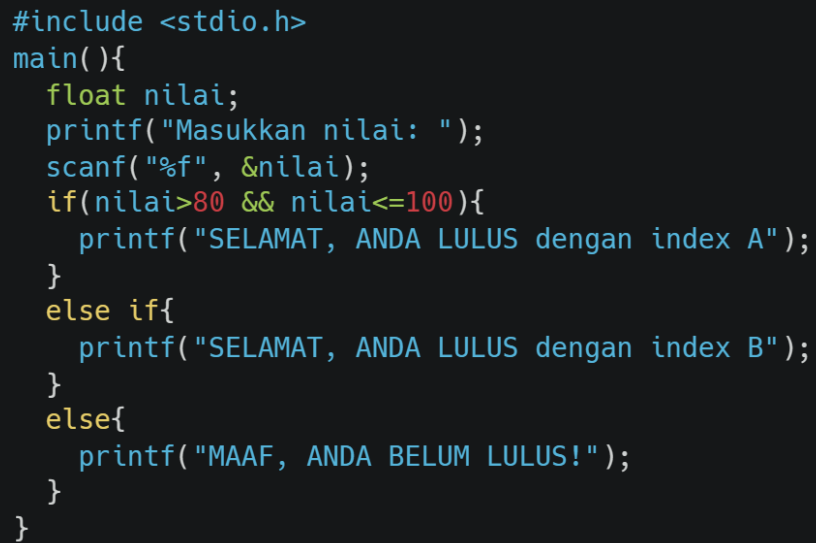


```
if(kondisi 1){
    /*statement atau aksi*
}
else if{
    /*statement atau aksi*
}
else{
    /*statement atau aksi*
}
```

Gambar 1.10 Syntax If-Else if

Tabel 1.28 Source Code If-Else if

Source Code



```

#include <stdio.h>
main(){
    float nilai;
    printf("Masukkan nilai: ");
    scanf("%f", &nilai);
    if(nilai>80 && nilai<=100){
        printf("SELAMAT, ANDA LULUS dengan index A");
    }
    else if{
        printf("SELAMAT, ANDA LULUS dengan index B");
    }
    else{
        printf("MAAF, ANDA BELUM LULUS!");
    }
}

```

Gambar 1.11 Source Code If-Else if

Output



```

Masukan nilai :65
SELAMAT, ANDA LULUS dengan index B!

```

Gambar 1.12 Output If-else if

1.3.14 Nested If

Nested if digunakan jika dalam suatu kondisi membutuhkan kondisi lainnya agar terpenuhi. Kondisi 1 harus bernilai benar, jika kondisi 2 juga bernilai benar maka program akan dijalankan. Namun jika salah satu kondisi bernilai salah, maka program akan mengambil instruksi selanjutnya.

```

if(kondisi 1){
    if(kondisi 2){
        /*statement atau aksi
    }
}

```

Gambar 1.13 Syntax Nested If

Contoh :

Source Code
<pre> #include <stdio.h> main(){ float nilai; int sks; printf("Masukkan nilai: "); scanf("%f", &nilai); scanf("%d",&sks); if(nilai>80 && nilai<=100){ printf("SELAMAT, ANDA LULUS dengan index A!"); if(sks>90){ printf("SELAMAT, ANDA LULUS!") } } else{ printf("MAAF, ANDA BELUM LULUS!"); } } </pre>
Gambar 1.14 Source Code Nested If
Output
<pre> Masukan nilai :84 Masukan sks :95 SELAMAT, ANDA LULUS! </pre>
Gambar 1.15 Output Nested if

1.3.15 Switch Case

Switch berfungsi untuk memeriksa beberapa kondisi dan akan mengerjakan *statement* sesuai dengan *case* yang terpenuhi.

Syntax :

```
switch(variabel){  
    case value 1: /*statement value 1*/; break;  
    case value 2: /*statement value 2*/; break;  
    case value 3: /*statement value 3*/; break;  
    default: /*statement selain value-value di atas*/; break;  
}
```

Gambar 1.16 Syntax Switch Case

Source Code
<pre>#include <stdio.h> main(){ int nilai; char index; printf("Masukkan nilai: "); scanf("%d", &nilai); switch(nilai){ case 80 ... 100: index = 'A'; break; case 60 ... 79: index = 'B'; break; case 40 ... 59: index = 'C'; break; case 10 ... 39: index = 'D'; break; default: index = 'E'; break; } printf("Index = %c", index) }</pre>
Output
<pre>Masukan nilai :57 Index = C</pre>

Gambar 1.17 Source Code Switch

Gambar 1.18 Output Switch

Yang perlu diperhatikan saat menggunakan *switch case* dalam bahasa C adalah:

1. Syarat variabel dalam menggunakan *switch case* adalah harus bertipe *int*, *char*, dan *enum*.
2. Peletakan *default*: bebas
3. Tidak boleh ada duplikat pada *value case*
4. Keyword *break* berfungsi untuk menghentikan perintah yang ada pada case. Jika tidak menyertakan *break*, maka *case* selanjutnya akan dilakukan
5. “...” 3 titik pada program diatas digunakan untuk menandakan *range/interval*. Misal: 1 sampai 5 bisa kita tuliskan menjadi 1 ... 5.
6. *Nested switch case* diperbolehkan.

Modul 2 Array

2.1 Tujuan praktikum

1. Mengetahui dan memahami penggunaan Array dalam Bahasa pemrograman.
2. Memahami jenis-jenis Array dan perbedaan masing-masing.
3. Memahami implementasi Array.

2.2 Alat dan Bahan

PC yang terinstal Code Blocks

2.3 Dasar Teori

Pada modul sebelumnya anda sudah mempelajari mengenai seleksi kondisional dan perulangan, kemudian muncul 1 masalah yang akan kalian hadapi dalam dalam pemograman, bagaimana anda dapat menyimpan data yang banyak saat perulangan ? apakah kita perlu membuat variable yang banyak untuk menampung semua itu?

Pertanyaan itu dapat kalian selesai menggunakan Array dengan mudah. Pada modul ini anda akan dijelaskan bagaimana cara kerja array beserta jenis – jenis array dalam pemrograman.

2.3.1 Definisi

Array merupakan sebuah jenis struktur data yang dapat menyimpan ukuran dari elemen yang mempunyai tipe data sama. Array merupakan konsep yang penting dalam pemrograman, karena array menyimpan data dalam jumlah banyak dan terindeks.

Misalkan ada kumpulan data bertipe int yaitu angka 1, 2, 3, 4, dan 5. Kumpulan data ini dapat disajikan dalam bentuk Array karena memiliki tipe data yang sama yaitu int. Misal kumpulan data tadi diberi nama Angka sehingga jika disajikan dalam bentuk array akan menjadi `int Angka[] = {1, 2, 3, 4, 5}` atau `int Angka[5] = {1, 2, 3, 4, 5}`. Pada sebuah array, index array dimulai dari indeks ke- 0, sehingga pada array `Angka[]`, angka 1 berada di indeks ke-0 (`Array[0]`), angka 2 berada di indeks ke-1 (`Array[1]`), dan seterusnya. Sedangkan pada pendeklarasian Array, `int Angka [5]` berarti Array Angka dapat menampung 5 masukan nilai int.

Nilai ke-1	Nilai ke-2	...	Nilai ke-N	→ Nilai Elemen Array
Alamat ke-1	Alamat ke-2	...	Alamat ke-N	→ Alamat Elemen Array
0	1	...	N-1	→ Indeks Elemen Array

Gambar 2.1 Tabel Array

2.3.2 Deklarasi

Dalam pendeklarasian array, harus menentukan jenis elemen dan jumlah elemen yang dibutuhkan oleh sebuah array, sistematikanya sebagai berikut :

```
type nameArray[ukuran Array];
```

Gambar 2.2 Deklarasi Array

Keterangan :

type = tipe data sebuah array (int, float, char, dll).

NamaArray = nama sebuah array (mahasiswa, dosen , matakuliah, ...).

UkuranData = ukuran data array, harus bilangan integer yang lebih besar dari 0 contoh: (4, 32 , 256 , ...).

Contoh pendeklarasian array :

Tabel 2.1 Contoh Deklarasi Array

int a[7] = {1,2,3,4};	deklarasi int[5] inisialisasi menjadi 1,2,3,4,0,0,0
char str[4] = "abc";	deklarasi char[4] inisialisasi menjadi 'a','b','c','0'
float xy[11];	xy merupakan sebuah array dari 11 data float
float *ptr[17];	ptr merupakan sebuah array dari 17 pointers menjadi floats

2.3.3 Jenis-jenis Array

Array dibedakan menjadi beberapa jenis, yaitu :

2.3.3.1 Array satu dimensi

Array satu dimensi adalah suatu array yang terdiri dari satu subscript, yaitu jumlah data maksimum. Sistematikanya sebagai berikut :

```
tipe_data nama_Array[ukuran Array];
```

Gambar 2.3 Syntax Array Satu Dimensi

‘Type’ dapat bertipe data bebas yang sesuai dalam bahasa C .

‘namaArray’ disesuaikan dengan compiler.

‘x’ harus bilangan integer yang lebih dari 0.

Contoh program :

Tabel 2.2 Program Array Satu Dimensi

SOURCE CODE
<pre>int main() { int array[5] = {1, 2, 3, 4, 5}; // deklarasi array dengan tipe int dengan ukuran 5 printf("element yang ada pada array:\n"); printf("[%d %d %d %d %d]\n", array[0], array[1], array[2], array[3], array[4]); return 0; }</pre>
OUTPUT

Gambar 2.4 Source Code Program Array Satu Dimensi

```

> make -s
> ./main
element yang ada pada array:
[1 2 3 4 5]
> 

```

Gambar 2.5 Output Program Array Satu Dimensi

2.3.3.2 Array 2 Dimensi

Array 2 dimensi adalah tipe data dalam bahasa pemrograman yang memungkinkan Anda untuk menyimpan data dalam bentuk matriks, atau grid dengan baris dan kolom. Setiap elemen dalam array memiliki indeks unik berupa pasangan indeks baris dan kolom. Array 2 dimensi dideklarasikan dengan menentukan tipe data elemen, nama array, dan jumlah baris dan kolom yang ingin disimpan.

```

int arr[3][4];

```

Gambar 2.6 Syntax Array Dua Dimensi

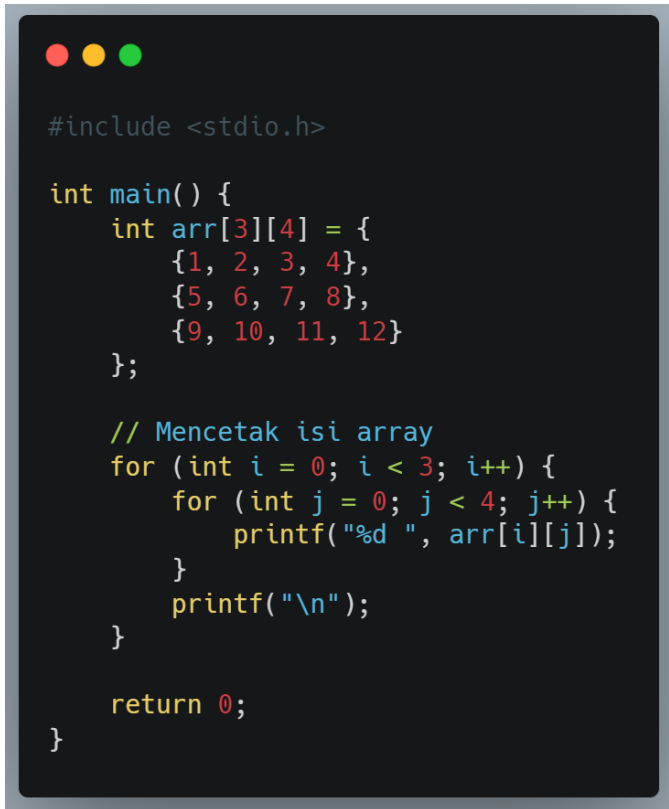
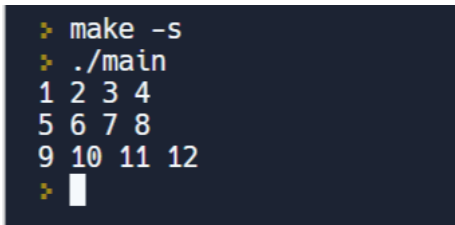
Ini akan membuat array 2 dimensi dengan nama "arr" yang menyimpan tiga baris dan empat kolom elemen integer.

Array dua dimensi dapat dianggap sebagai table yang memiliki 'x' jumlah baris dan 'y' jumlah kolom. Berikut yang berisi 2 baris dan 3 kolom yang dapat ditampilkan sebagai berikut.

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

Gambar 2.7 Analogi Array Dua Dimensi

Tabel 2.3 Program Array Dua Dimensi


SOURCE CODE
 <pre> #include <stdio.h> int main() { int arr[3][4] = { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} }; // Mencetak isi array for (int i = 0; i < 3; i++) { for (int j = 0; j < 4; j++) { printf("%d ", arr[i][j]); } printf("\n"); } return 0; } </pre>
Gambar 2.8 Source Code Program Array Dua Dimensi
OUTPUT
 <pre> > make -s > ./main 1 2 3 4 5 6 7 8 9 10 11 12 > </pre>
Gambar 2.9 Output Program Array Satu Dimensi

2.3.4 Looping Array

Looping array digunakan untuk menyimpan data dalam bentuk matriks atau grid dengan lebih dari dua dimensi. Setiap elemen dalam array memiliki indeks unik yang terdiri dari beberapa pasangan indeks. Array multidimensi dideklarasikan dengan menentukan tipe data elemen, nama array, dan jumlah dimensi serta ukuran masing-masing dimensi yang ingin disimpan.

Meskipun array multidimensi jarang digunakan dalam pemrograman, mereka dapat membantu memecahkan masalah yang memerlukan representasi data dalam bentuk matriks dengan lebih dari dua dimensi. Misalnya, sebuah array tiga dimensi dapat digunakan untuk menyimpan data seperti gambar 3D atau volumetrik.

Tabel 2.4 Program *Looping Array*

SOURCE CODE
 <pre>#include <stdio.h> int main() { int arr[2][3][4] = { { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} }, { {13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24} } }; // Mencetak isi array for (int i = 0; i < 2; i++) { for (int j = 0; j < 3; j++) { for (int k = 0; k < 4; k++) { printf("%d ", arr[i][j][k]); } printf("\n"); } printf("\n"); } return 0; }</pre>
Gambar 2.10 Source Code Program <i>Looping Array</i>
OUTPUT

```
> make -s
> ./main
1 2 3 4
5 6 7 8
9 10 11 12

13 14 15 16
17 18 19 20
21 22 23 24

> █
```

Gambar 2.11 *Output* Program *Looping Array*

Modul 3 Prosedur, Fungsi, dan Pointer

3.1 Tujuan Praktikum

1. Memahami dan mampu mengimplementasikan Fungsi dan Prosedur pada Bahasa C
2. Memahami konsep pointer pada Bahasa C

3.2 Alat dan Bahan

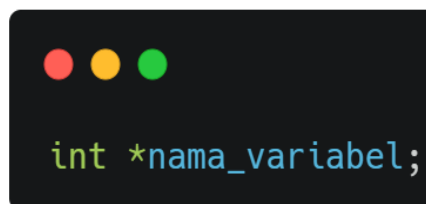
PC yang terinstal code blocks

3.3 Dasar Teori

3.3.1 Pointer

Pointer adalah tipe data yang menyimpan alamat memori dari sebuah variabel. Ini berarti bahwa pointer tidak menyimpan nilai sebenarnya dari variabel, melainkan menyimpan alamat memori tempat variabel itu disimpan. Oleh karena itu, pointer bisa digunakan untuk mengakses dan memanipulasi data yang disimpan pada alamat memori tersebut.

Untuk membuat pointer, kita harus menentukan tipe data dari variabel yang akan di referensikan oleh pointer dan menambahkan tanda *asterisk* (*) sebelum nama variabel. Misalnya, jika kita ingin membuat pointer untuk variabel integer, maka kita dapat mendeklarasikan pointer sebagai berikut:


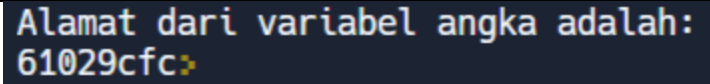


```
int *nama_variabel;
```

Gambar 3.1 Deklarasi Pointer


Selain operator *asterisk* (*), juga ada operator *ampersand* (&) pada implementasi *pointer*. Operator *ampersand* (&) digunakan untuk mengetahui sebuah address (alamat pada memory) dari suatu variabel *pointer*. Alamat memori akan selalu berupa 6 karakter *hexadecimal*. Contoh implementasi operator *ampersand* adalah sebagai berikut:

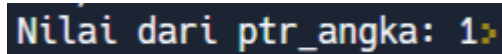
Tabel 3.1 *Pointer – Operator Ampersand*

SOURCE CODE
 <pre>#include <stdio.h> int main() { int angka = 1; printf("Alamat dari variabel angka adalah:\n"); printf("%x", &angka); return 0; }</pre>
<p>Gambar 3.2 Source Code Pointer Ampersand</p>
OUTPUT
 <pre>Alamat dari variabel angka adalah: 61029cfc></pre>
<p>Gambar 3.3 Output Pointer Ampersand</p>

Untuk mendapatkan nilai dari suatu alamat yang terdapat pada isi pointer, maka perintah yang digunakan adalah menambah tanda *Asterisk* “*” di depan variabel. Adapun contoh penggunaannya :

Tabel 3.2 *Pointer – Nilai dari Suatu Alamat pada Pointer*

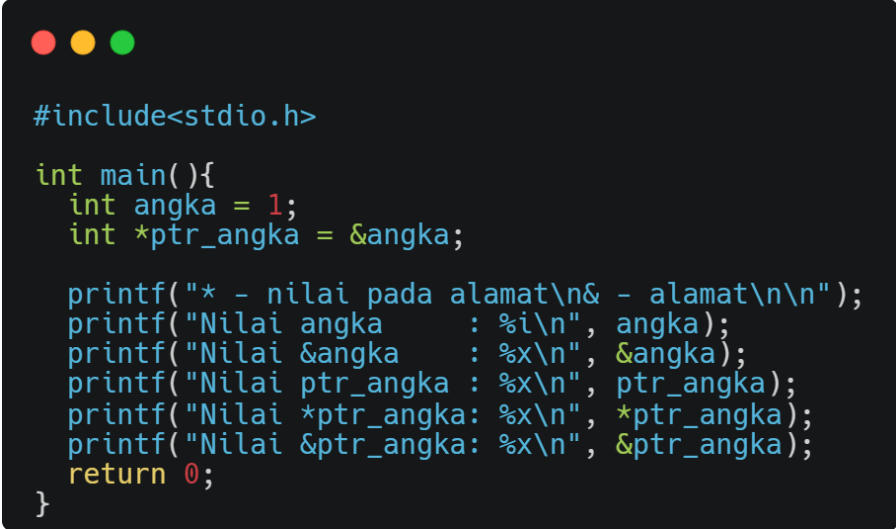
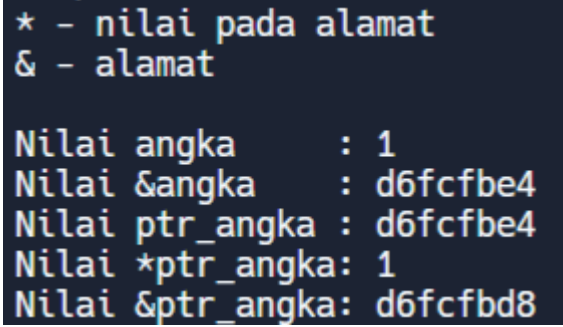
SOURCE CODE
 <pre>#include <stdio.h> int main() { int angka = 1; int *ptr_angka = &angka; printf("Nilai dari ptr_angka: %x", *ptr_angka); return 0; }</pre>
<p>Gambar 3.4 Source Code Nilai Alamat pada Pointer</p>
OUTPUT



Gambar 3.5 Output Nilai Alamat pada *Pointer*

Berikut contoh lanjutan dari implementasi pointer:

Tabel 3.3 Pointer – Contoh Lanjutan

SOURCE CODE
 <pre>#include<stdio.h> int main(){ int angka = 1; int *ptr_angka = &angka; printf("* - nilai pada alamat\n& - alamat\n\n"); printf("Nilai angka : %i\n", angka); printf("Nilai &angka : %x\n", &angka); printf("Nilai ptr_angka : %x\n", ptr_angka); printf("Nilai *ptr_angka: %x\n", *ptr_angka); printf("Nilai &ptr_angka: %x\n", &ptr_angka); return 0; }</pre>
OUTPUT
 <pre>* - nilai pada alamat & - alamat Nilai angka : 1 Nilai &angka : d6fcfbe4 Nilai ptr_angka : d6fcfbe4 Nilai *ptr_angka : 1 Nilai &ptr_angka : d6fcfbd8</pre>

Gambar 3.7 Output Contoh *Pointer* Lanjutan

Untuk lebih jelasnya coba perhatikan keterangan berikut:

- angka – sudah dapat diketahui nilai dan *type* datanya yaitu 1 (*type* data *integer*).
- &angka – alamat dari variabel “angka” adalah d6fcfbe4 (*hexadecimal*).

- `ptr_angka` – nilai dari alamat *pointer* “`ptr_angka`” adalah alamat dari variabel “angka”, karena perintah atau logika (`int *ptr_angka = &angka`), maka nilai “`ptr_angka`” akan sama dengan alamat variabel “angka”.
- `*ptr_angka` – nilai pada alamat yang ditunjuk *pointer* “`ptr_angka`”, karena “`ptr_angka`” mengandung alamat “angka”, nilai pada alamat yang ditunjuk adalah nilai variabel “angka”.
- `&ptr_angka` – alamat dari “`ptr_angka`” adalah `d6fcfbd8` (*hexadecimal*).

3.3.2 Fungsi

Fungsi adalah sebuah blok kode terorganisir yang dapat digunakan kembali atau *reusable* untuk melakukan sebuah aksi / tindakan. Fungsi terpisah dalam blok sendiri yang berfungsi sebagai *sub-program* yang merupakan sebuah program kecil untuk memproses sebagian dari pekerjaan program utama. Tujuan pentingnya adalah untuk membuat program tersebut mudah dipahami atau dibaca alur programnya.

Penggunaan fungsi membuat program menjadi lebih efisien karena mengurangi pengulangan penulisan kode yang sama. Hal ini karena jika sekelompok kode sudah dibuat menjadi sebuah fungsi maka selanjutnya kita tinggal memanggil nama fungsinya beserta dengan argumen yang menyertainya (jika terdapat parameter pada fungsi tersebut). Sebuah fungsi dideklarasikan diluar dari blok program utama (`int main()`). Untuk mendeklarasikan sebuah fungsi adalah sebagai berikut:

```
TipeData Nama_Fungsi(/*Parameter*/){
    //Statement kode
    return nilai; //nilai yang ingin dikeluarkan
}
```

Gambar 3.8 Deklarasi sebuah fungsi


Pada fungsi terdapat statement *return* yang berfungsi untuk mengeluarkan sebuah *value* (nilai) dari fungsi yang dijalankan. Untuk pemanggilan sebuah fungsi yang telah dibuat, dilakukan di dalam program utama (*int main()*). bentuk umumnya adalah seperti berikut.



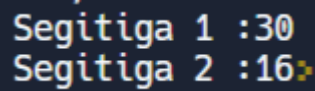
```
Nama_Fungsi(/*Parameter*/); //Memanggil Fungsi
```

Gambar 3.9 Syntax Memanggil Sebuah Fungsi

Tabel 3.4 Contoh Penggunaan Fungsi

SOURCE CODE
 <pre>#include <stdio.h> #include <stdlib.h> // fungsi untuk menghitung luas segitiga int luas_segitiga(int alas, int tinggi){ int luas; luas = (0.5*alas)*tinggi; return luas; // mengeluarkan nilai "luas" } int main(){ int hasil; // memanggil sebuah fungsi cara 1 hasil = luas_segitiga(10,6); printf("Segitiga 1 :%d \n",hasil); //memanggil sebuah fungsi cara 2 printf("Segitiga 2 :%d \n",luas_segitiga(4,8)); return 0; }</pre>
OUTPUT

Gambar 3.10 Source Code Penggunaan Fungsi



```
Segitiga 1 :30
Segitiga 2 :16
```

Gambar 3.11 *Output* Penggunaan Fungsi

3.3.3 Prosedur

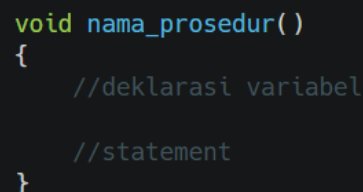
Prosedur adalah suatu program yang terpisah dalam blok sendiri yang berfungsi sebagai subprogram yang bersifat aktivitas. Parameter adalah nama variabel yang dideklarasikan pada bagian header prosedur. Tiap data ditransfer antara parameter aktual dan parameter formal yang bersesuaian. Parameter aktual adalah yang disertakan pada waktu pemanggil prosedur, sedangkan parameter formal adalah parameter yang dideklarasikan di dalam bagian header prosedur.

A. Syntax

Prosedur bukan program yang berdiri sendiri, jadi ia tidak dapat dieksekusi secara langsung. Ini berarti, instruksi-instruksi di dalam prosedur baru dapat dilaksanakan hanya bila prosedur tersebut diakses. Berikut adalah jenis prosedur berdasarkan dengan parameternya:

a. Tanpa Parameter

prosedur tanpa parameter adalah sebuah blok kode yang dapat dipanggil pada suatu bagian dari program tanpa memerlukan input dari user atau input lain. Prosedur ini memiliki sintaks seperti berikut:



```
void nama_prosedur( )
{
    //deklarasi variabel

    //statement
}
```

Gambar 3.12 Syntax prosedur tanpa parameter

b. Menggunakan Parameter

prosedur Menggunakan parameter adalah sebuah blok kode yang dapat dipanggil pada suatu bagian dari program dan memerlukan input dari user atau input lain. Prosedur ini memiliki sintaks seperti berikut:

```
void nama_prosedur(deklarasi prosedur) //jika ada
{
    //deklarasi variabel

    //statement
}
```

Gambar 3.13 Syntax prosedur dengan parameter

B. Parameter

Pada sebuah prosedur variabel yang mencatat atau mewakili suatu nilai untuk melakukan suatu proses disebut parameter. Kebanyakan program memerlukan pertukaran informasi antara prosedur dan titik dimana ia dipanggil. Ketika prosedur dipanggil, parameter aktual berkoresponden dengan parameter formal (pada sebuah prosedur). Tiap parameter aktual berpasangan dengan parameter formal yang bersesuaian.

a. Parameter Masukan

Pada parameter ini, nilai (*value*) parameter aktual diisikan ke dalam parameter formal yang bersesuaian, dapat dilihat pada contoh berikut:

Tabel 3.5 Contoh Program Parameter Masukan

SOURCE CODE

```

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

void bola(int jari)
{
    float volume;
    volume=(4/3)*3.14*jari*jari*jari;
    printf("maka volume bola adalah : %f\n",volume);
}

int main(){
    int jari2;
    printf("Masukkan jari-jari bola : ");
    scanf("%d",&jari2);
    bola(jari2);
    return 0;
}

```

Gambar 3.14 *Source Code* Prosedur Parameter Masukan

OUTPUT

```

Masukkan jari-jari bola : 11
maka volume bola adalah : 4179.339844

```

Gambar 3.15 *Output* Prosedur Parameter Masukan

b. Parameter Keluaran

Prosedur dapat menghasilkan satu atau lebih output yang dapat kita gunakan panggil kedalam program utama dengan menggunakan program pemanggil. Maka nilai keluaran tersebut ditampung di dalam parameter keluaran, dapat dilihat pada contoh berikut:

Tabel 3.6 Contoh Program Parameter Keluaran

SOURCE CODE


```

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

void bola(int jari, float *volume)
{
    *volume=(4/3)*3.14*jari*jari*jari;
}

int main(){
    int jari2;
    float v;
    printf("Masukkan jari-jari bola : ");
    scanf("%d",&jari2);
    bola(jari2,&v);
    printf("maka volume bola adalah : %f\n",v);
    return 0;
}

```

Gambar 3.16 *Source Code* Prosedur Parameter Keluaran

OUTPUT

```

Masukkan jari-jari bola : 11
maka volume bola adalah : 4179.339844

```

Gambar 3.17 *Output* Prosedur Parameter Keluaran

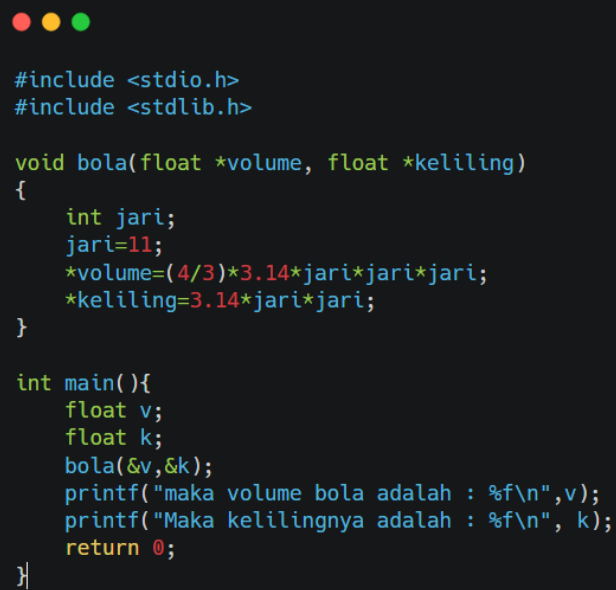
Dilihat dari program diatas, parameter pada prosedur menggunakan pointer sehingga dapat menampung nilai keluaran dari prosedurnya.

c. Parameter Masukan dan Keluaran

Parameter ini merupakan gabungan dari dua jenis parameter sebelumnya dan dapat berperan sebagai masukan sekaligus keluaran. Dapat dilihat pada contoh berikut:

Tabel 3.7 Contoh program parameter Masukan dan Keluaran

SOURCE CODE



```

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

void bola(float *volume, float *keliling)
{
    int jari;
    jari=11;
    *volume=(4/3)*3.14*jari*jari*jari;
    *keliling=3.14*jari*jari;
}

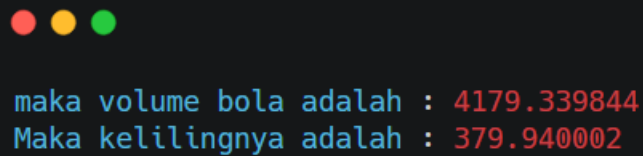
int main(){
    float v;
    float k;
    bola(&v,&k);
    printf("maka volume bola adalah : %f\n",v);
    printf("Maka kelilingnya adalah : %f\n", k);
    return 0;
}

```

Gambar 3.18 *Source Code* Prosedur Parameter

Masukan dan Keluaran

OUTPUT



```

maka volume bola adalah : 4179.339844
Maka kelilingnya adalah : 379.940002

```

Gambar 3.19 *Output* Prosedur Parameter Masukan dan keluaran

Modul 4 Sorting

4.1 Tujuan

1. Mengetahui dan Memahami Konsep Sorting.
2. Mengetahui Jenis-Jenis Sorting dan Perbedaan Masing-masing Sorting.
3. Memahami Bagaimana Implementasi Sorting.

4.2 Alat dan Bahan

PC yang sudah terinstal code blocks

4.3 Dasar Teori

Dalam pemrograman dengan bahasa C memiliki cara untuk mengurutkan data. Konsep Sorting dalam pemrograman dengan bahasa C dapat memudahkan mengurutkan data yang acak. Ada beberapa metode sorting dalam Bahasa C.

4.3.1 Bubble Sort

Bubble sort adalah algoritma pengurutan yang membandingkan dua elemen yang berdekatan dan menukarnya hingga berada dalam urutan yang diinginkan. Sama seperti pergerakan gelembung udara di dalam air yang naik ke permukaan, setiap elemen array bergerak ke ujung di setiap iterasi. Oleh karena itu, ini disebut bubble sort. Algoritma ini tidak cocok untuk kumpulan data besar karena membutuhkan waktu proses yang lama.

Tabel 4.1 Bubble Sort

Bubble Sort
Sebuah array memiliki nilai [5, 1, 4, 2, 8]

Iterasi 1

indeks 0 akan membandingkan dengan indeks 1
nilai akan ditukar jika indeks 0 lebih besar dari indeks 1

5	1	4	2	8
---	---	---	---	---

karna lebih besar maka nilai akan ditukar

1	5	4	2	8
---	---	---	---	---

selanjutnya indeks 1 akan membandingkan dengan indeks 2
nilai akan ditukar jika indeks 1 lebih besar dari indeks 2

1	5	4	2	8
---	---	---	---	---

karena lebih besar maka nilai akan ditukar

1	4	5	2	8
---	---	---	---	---

lakukan langkah - langkah tersebut hingga indeks terakhir

1	4	5	2	8
---	---	---	---	---

ditukar karena lebih besar

1	4	2	5	8
---	---	---	---	---

1	4	2	5	8
---	---	---	---	---

tidak ditukar karena lebih kecil

1	4	2	5	8
---	---	---	---	---

Iterasi 2

karena indeks terakhir sudah diurutkan maka pengurutan akan berhenti di indeks terakhir yang belum diurutkan

langkah - langkah sama seperti pada iterasi 1

1	4	2	5	8
---	---	---	---	---

tidak ditukar karena lebih kecil

1	4	2	5	8
---	---	---	---	---

1	4	2	5	8
---	---	---	---	---

ditukar karena lebih besar

1	2	4	5	8
---	---	---	---	---

1	2	4	5	8
---	---	---	---	---

tidak ditukar karena lebih kecil

1	2	4	5	8
---	---	---	---	---

Iterasi 3

1	2	4	5	8
---	---	---	---	---

tidak ditukar karena lebih kecil				
1	2	4	5	8

1	2	4	5	8
tidak ditukar karena lebih kecil				
1	2	4	5	8

Tabel 4.2 Coding Bubble Sort

Source Code



```

// Bubble sort in C
#include <stdio.h>

int main() {
    int data[] = {5, 1, 4, 2, 8};
    int size = sizeof(data) / sizeof(data[0]);

    // Bubble Sort
    for (int step = 0; step < size - 1; ++step) {
        for (int i = 0; i < size - step - 1; ++i) {

            // Ubah < menjadi > untuk pengurutan dari nilai terbesar.
            if (data[i] > data[i + 1]) {
                int temp = data[i];
                data[i] = data[i + 1];
                data[i + 1] = temp;
            }
        }
    }
    printf("Hasil pengurutan dari nilai terkecil:\n");
    for (int i = 0; i < size; ++i) {
        printf("%d ", data[i]);
    }
}

```

Gambar 4.1 Coding Bubble Sort

Output

```

> make -s
> ./main
Hasil pengurutan dari nilai terkecil:
1 2 4 5 8 >

```

Gambar 4.2 Output Coding Bubble Sort

4.3.2 Insertion Sort

Insertion Sort adalah algoritma pengurutan sederhana yang bekerja mirip dengan cara Anda mengurutkan kartu remi di tangan Anda. *Array* secara virtual dibagi menjadi bagian yang diurutkan dan bagian yang tidak disortir. Nilai dari bagian yang tidak disortir diambil dan ditempatkan pada posisi yang benar di bagian yang diurutkan.

Contoh :

Tabel 4.3 Insertion Sort

Insertion Sort

Sebuah array memiliki nilai [9, 5, 1, 4, 3]

Iterasi 1

indeks 1 akan membandingkan dengan indeks 0

nilai indeks 1 disimpan didalam key

indeks 1 di isi dengan nilai indeks 0

indeks 0 akan di isi dengan nilai dalam key

key	9	5	1	4	3
5	9	9	1	4	3
	5	9	1	4	3

Iterasi 2

indeks 2 akan membandingkan dengan indeks sebelumnya

langkah - langkah sama seperti pada iterasi 1

key	5	9	1	4	3
1	5	9	9	4	3
	5	5	9	4	3
	1	5	9	4	3

nilai - nilai akan terus dipindahkan dan **akan berhenti jika**
 indeks 2 lebih besar dari indeks yang dibandingkan
 indeks 2 menempati di awal urutan

Iterasi 3

indeks 3 akan membandingkan dengan indeks sebelumnya

key	1	5	9	4	3
4	1	5	9	9	3
	1	5	5	9	3
	1	4	5	9	3

indeks 0 (1) tidak dipindahkan karena nilai lebih kecil dari indeks 3 (4)

maka indeks 1 diisi dengan key (4)

Iterasi 4

indeks 4 akan membandingkan dengan indeks sebelumnya

key	1	4	5	9	3
3	1	4	5	9	9
	1	4	5	5	9
	1	4	4	5	9
	1	3	4	5	9

indeks 0 (1) tidak dipindahkan karena nilai lebih kecil dari indeks 4 (3)

maka indeks 1 diisi dengan key (3)

Tabel 4.4 Coding Insertion Sort

Source Code

```

// Insertion sort pada C
#include <stdio.h>

int main() {
    int data[] = {9, 5, 1, 4, 3};
    int size = sizeof(data) / sizeof(data[0]);

    // Insertion Sort
    for (int step = 1; step < size; step++) {
        int key = data[step];
        int j = step - 1;

        // Ubah key<data[j] menjadi key>data[j] untuk pengurutan dari nilai
        terbesar.
        while (key < data[j] && j >= 0) {
            data[j + 1] = data[j];
            --j;
        }
        data[j + 1] = key;
    }

    printf("Hasil pengurutan dari nilai terkecil:\n");
    for (int i = 0; i < size; ++i) {
        printf("%d ", data[i]);
    }
}

```

Gambar 4.3 Coding Insertion Sort

Output

```

❯ make -s
❯ ./main
Hasil pengurutan dari nilai terkecil:
1 3 4 5 9 ❯

```

Gambar 4.4 Output Coding Insertion Sort

4.3.3 Selection Sort

Selection Sort adalah algoritma pengurutan yang memilih elemen terkecil dari daftar yang tidak diurutkan di setiap iterasi dan menempatkan elemen tersebut di awal daftar yang tidak diurutkan. Proses ini diulangi untuk sisa bagian daftar yang tidak diurutkan hingga seluruh daftar diurutkan.

Contoh :

Tabel 4.5 Selection Sort

Selection Sort

Sebuah array memiliki nilai [20, 12, 10, 15, 2]

Iterasi 1

menentukan nilai "minimum" awal dengan memasukan nilai indeks 0

*merah menandakan nilai sedang dibandingkan mana yang lebih kecil

*biru menandakan nilai minimum

nilai minimum akan dibandingkan dengan nilai indeks array yang belum diurutkan

nilai paling kecil akan dimasukan kedalam nilai "minimum"

20	12	10	15	2
20	12	10	15	2
20	12	10	15	2
20	12	10	15	2
20	12	10	15	2

nilai 2 dalam indeks 4 adalah minimum

maka nilai 2 akan ditukar dengan nilai pada indeks 0 (indeks paling awal yang belum diurutkan)

2	12	10	15	20
---	----	----	----	----

Iterasi 2

indeks 0 adalah array yang telah diurutkan maka tidak akan masuk dalam perbandingan nilai

nilai "minimum" dimasukan dengan nilai indeks 1 (indeks paling awal array yang belum diurutkan)

2	12	10	15	20
2	12	10	15	20
2	12	10	15	20
2	12	10	15	20

nilai 10 di indeks 2 adalah minimum

indeks 1 (indeks paling awal array yang belum diurutkan) akan ditukar dengan indeks 2 (minimum)

2	10	12	15	20
---	----	----	----	----

Iterasi 3

indeks 0 dan 1 adalah array yang sudah diurutkan
ulangi langkah - langkah seperti pada iterasi sebelumnya

2	10	12	15	20
2	10	12	15	20
2	10	12	15	20

karena nilai minimum di indeks 2 sudah benar posisi urutannya maka tidak ada yang berubah

2	10	12	15	20
---	----	----	----	----

Iterasi 4

2	10	12	15	20
2	10	12	15	20

karena nilai minimum di indeks 3 sudah benar posisi urutannya maka tidak ada yang berubah

2	10	12	15	20
---	----	----	----	----

Tabel 4.6 Coding Selection Sort

Source Code

```

// Selection sort pada C
#include <stdio.h>

int main() {
    int data[] = {20, 12, 10, 15, 2};
    int size = sizeof(data) / sizeof(data[0]);
    int temp = 0;

    // Selection Sort
    for (int step = 0; step < size - 1; step++) {
        int min_idx = step;
        for (int i = step + 1; i < size; i++) {
            // Ubah < menjadi > untuk pengurutan dari nilai terbesar.
            if (data[i] < data[min_idx])
                min_idx = i;
        }

        temp = data[min_idx];
        data[min_idx] = data[step];
        data[step] = temp;
    }

    printf("Hasil pengurutan dari nilai terkecil:\n");
    for (int i = 0; i < size; ++i) {
        printf("%d ", data[i]);
    }
}

```

Gambar 4.3 Output Coding Selection Sort

Output

```

❯ make -s
❯ ./main
Hasil pengurutan dari nilai terkecil:
2 10 12 15 20 ❯

```

Gambar 4.4 Output Coding Selection Sort

Modul 5 Searching

5.1 Tujuan Praktikum

1. Mengetahui mengenai logika pencarian dengan metode sequential dan interval.
2. Mampu mengimplementasikan logika pencarian kedalam program dalam bahasa C.
3. Mampu melakukan analisis pada algoritma searching yang dibuat.

5.2 Alat dan Bahan

PC yang terinstal IDE Code Blocks

5.3 Dasar Teori

Algoritma pencarian adalah algoritma yang digunakan untuk mencari sebuah nilai dari sebuah array. Algoritma ini pada dasarnya merupakan algoritma perulangan sampai data tersebut ditemukan atau data tersebut dinyatakan tidak ada didalam array tersebut.

Algoritma pencarian umumnya dibagi menjadi 2 jenis:

1. **Sequential Search**: algoritma ini memeriksa setiap data / nilai dari indeks pertama array tersebut sampai indeks terakhir indeks tersebut.

Contoh: **Linear Search**

2. **Interval Search** algoritma ini lebih efisien karena tidak harus mengecek semua indeks dalam array tersebut, namun array tersebut harus diurukan terlebih dahulu. Contoh: **Binary Search**

5.3.1 Linear Search

Linear searching adalah metode yang bekerja dengan cara mengecek satu persatu isi array tersebut dari awal hingga akhir (secara berurutan). Hingga nilai tersebut ditemukan atau tidak ditemukan

Kelebihan algoritma ini adalah jika nilai yang dicari ada diawal array tersebut sehingga cepat ditemukan. Sedangkan kelemahannya adalah ketika nilai yang dicari terdapat pada akhir array, sehingga proses pencariannya akan lama dan bertambah lama jika array tersebut memiliki data yang sangat banyak.

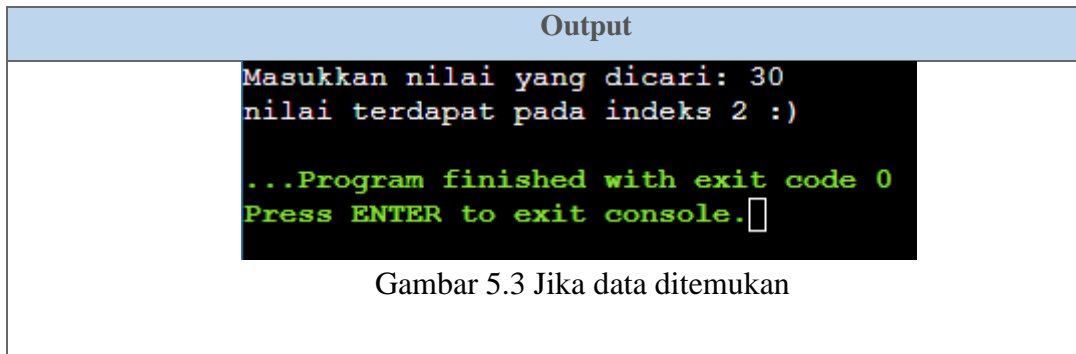


Gambar 5.1 Ilustrasi Linear Search

Contoh : Array A = [2,4,7,6,5] , data yang dicari adalah 7 , maka program akan mengecek satu persatu dan data ditemukan di Indeks ke-2

Tabel 5.1 Contoh Program Linear Search

Source Code
<pre> #include <stdio.h> int main() { int daftarData[5] = {107, 52, 30, 69, 43}; //input nilai target int targetNilai; printf("Masukkan nilai yang dicari: "); scanf("%d", &targetNilai); //deklarasi variabel indeks target, //jika nilainya tetap -1, berarti targetNilai tidak ditemukan //jika nilainya berubah, maka targetNilai ditemukan int indeksTarget = -1; //algoritma linear search for (int i = 0; i < 5; i++) { //pengecekan nilai satu persatu if (targetNilai == daftarData[i]) { indeksTarget = i; } } if (indeksTarget >= 0) { printf("nilai terdapat pada indeks %d :", indeksTarget); } else { printf("nilai 404 not found :("); } } </pre>
Gambar 5.2 Contoh program yang menggunakan linear search



5.3.2 Binary Search

Metode searching dengan algoritma binary hanya bisa digunakan untuk data yang telah diurutkan (sorting). Karena algoritma ini bekerja dengan cara menetapkan indeks dengan nilai tengah (mid). Kemudian nilai mid tersebut dibandingkan dengan nilai yang dicari

Algoritma binary searching adalah seperti berikut:

1. Deklarasi / input data array sebanyak N, dan target nilai yang dicari.
2. Menetapkan indeks low (0), high (N – 1) dan mid ((low + high) / 2).
3. Membandingkan target nilai dengan isi indeks mid:
 - a. Jika target nilai = nilai indeks mid, maka perulangan berhenti.
 - b. Jika target nilai > nilai indeks mid, maka low = mid + 1
 - c. Jika target nilai < nilai indeks mid, maka high = mid - 1
4. Langkah 3 diulang terus, sampai target nilai ditemukan atau dinyatakan tidak ditemukan.

Algoritma ini memiliki keunggulan dari segi fleksibilitas yang dimana algoritma ini hanya menggunakan sedikit waktu ketimbang Metode Sequential Search.

Sebagai contoh:

Banyak Data :	10									
Indeks :	0	1	2	3	4	5	6	7	8	9
Data Array A :	2	5	6	10	11	12	17	21	22	25

Gambar 5.4 Langkah pertama contoh binary search

Dari data di atas diketahui N (Banyak data) = 10. Jika dicari nilai 5, maka tentukan dulu nilai awalnya sebagai berikut:

- Low: 0
- High: $N - 1 \Rightarrow 10 - 1 \Rightarrow 9$
- Mid: $(9 + 0) / 2 \Rightarrow 4.5 \Rightarrow 4$

Indeks :	0	1	2	3	4	5	6	7	8	9
	LOW				MID				HIGH	
Loop 1 - Data Array A :	2	5	6	10	11	12	17	21	22	25

Gambar 5.5 Langkah kedua contoh binary search

Program akan mengecek apakah nilai di Indeks 4 apakah sama dengan nilai yang dicari, dan ternyata $5 \text{ (cari)} < 11 \text{ (Indeks ke-4)}$. Maka perulangan akan tetap berjalan dan terjadilah perubahan nilai sebagai berikut :

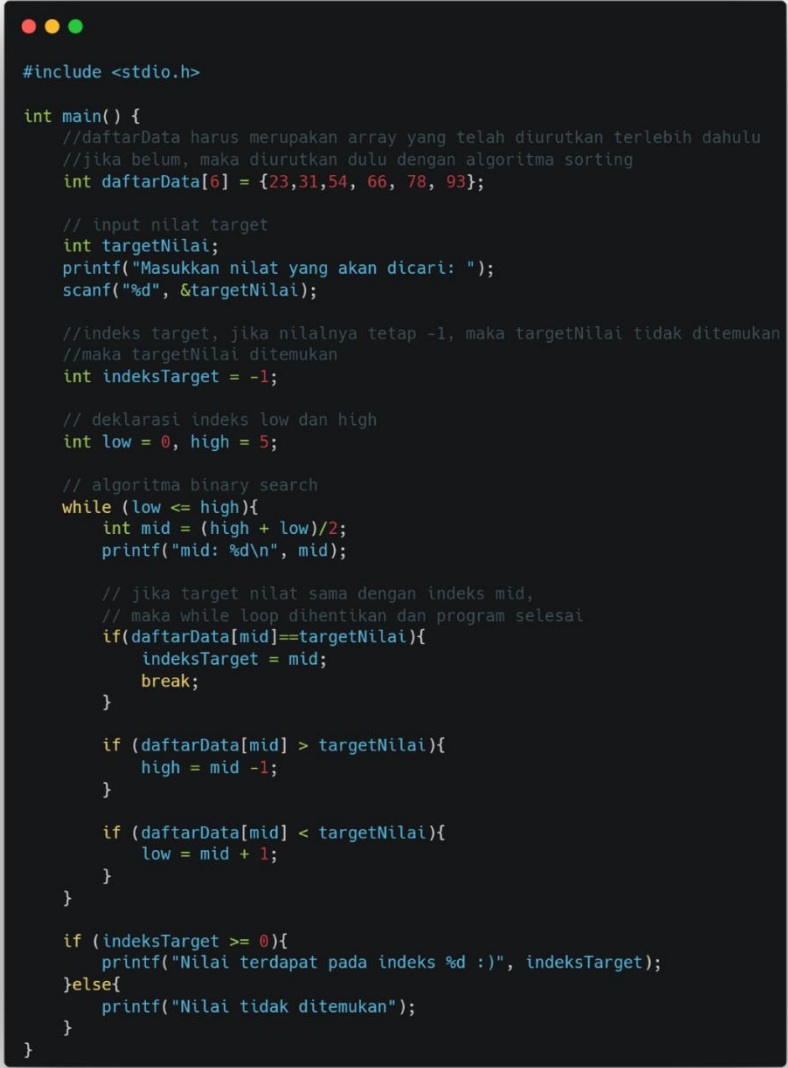
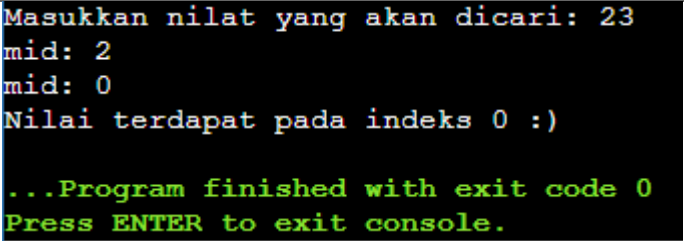
- Low: 0
- High: $\text{Mid} - 1 \Rightarrow 4 - 1 \Rightarrow 3$
- Mid: $(3 + 0) / 2 \Rightarrow 1.5 \Rightarrow 1$

Indeks :	0	1	2	3	4	5	6	7	8	9
	LOW	MID		HIGH						
Loop 2 - Data Array A :	2	5	6	10	11	12	17	21	22	25

Gambar 5.6 Langkah ketiga contoh binary search

Program akan mengecek apakah nilai di Indeks 1 apakah sama dengan nilai yang dicari, dan ternyata $5 \text{ (cari)} == 5 \text{ (Indeks ke-1)}$. Maka program menetapkan bahwa nilai yang dicari ada di Indeks ke-1.

Tabel 5.2 Contoh Program Binary Search

Source Code
 <pre> #include <stdio.h> int main() { //daftarData harus merupakan array yang telah diurutkan terlebih dahulu //jika belum, maka diurutkan dulu dengan algoritma sorting int daftarData[6] = {23,31,54, 66, 78, 93}; // input nilai target int targetNilai; printf("Masukkan nilai yang akan dicari: "); scanf("%d", &targetNilai); //indeks target, jika nilainya tetap -1, maka targetNilai tidak ditemukan //maka targetNilai ditemukan int indeksTarget = -1; // deklarasi indeks low dan high int low = 0, high = 5; // algoritma binary search while (low <= high){ int mid = (high + low)/2; printf("mid: %d\n", mid); // jika target nilai sama dengan indeks mid, // maka while loop dihentikan dan program selesai if(daftarData[mid]==targetNilai){ indeksTarget = mid; break; } if (daftarData[mid] > targetNilai){ high = mid -1; } if (daftarData[mid] < targetNilai){ low = mid + 1; } } if (indeksTarget >= 0){ printf("Nilai terdapat pada indeks %d :)", indeksTarget); }else{ printf("Nilai tidak ditemukan"); } } </pre>
<p>Gambar 5.7 contoh program dengan binary search</p>
Output
 <pre> Masukkan nilai yang akan dicari: 23 mid: 2 mid: 0 Nilai terdapat pada indeks 0 :) ...Program finished with exit code 0 Press ENTER to exit console. </pre> <p>Gambar 5.8 Jika nilai ada</p>


```
Masukkan nilai yang akan dicari: 0
mid: 2
mid: 0
Nilai tidak ditemukan

...Program finished with exit code 0
Press ENTER to exit console. □
```

Gambar 5.9 Jika nilai tidak ditemukan

Modul 6 Operasi File

6.1 Tujuan Praktikum

1. Memahami cara membuat, membuka, dan menutup file.
2. Memahami cara read dan write pada file.
- 3.

6.2 Alat dan Bahan

PC yang terinstal code blocks

6.3 Dasar Teori

6.3.1 Pengertian File

File adalah identitas dari data yang disimpan di dalam berkas sistem yang dapat diakses dan diatur oleh pengguna. Penggunaan dan pengoperasian file pasti selalu dibutuhkan terutama bagi seorang programmer untuk mengolah sebuah data pada file. Penggunaan operasi **FILE** dalam sebuah program sangat dibutuhkan dalam pembuatan program yang sesungguhnya. Kita membutuhkan file sebagai tempat penampung data-data selama operasi program. Jika kamu menggunakan variabel biasa, data yang biasanya diolah program hanya akan tersimpan sementara dalam memory dan akan hilang ketika program close. Berbeda dengan *memory*, penyimpanan data berbasis file akan tersimpan terus walaupun program telah di-*close* maupun komputer telah di *shutdown*.

6.3.2 Operasi File

Secara umum operasi file dibagi menjadi 3, yaitu: Membuat File, Membuka dan Menutup File, Membaca dan Menulis File.

1. **Membuat File**
- 2.

Dalam membuat file kita memerlukan nama file terlebih dahulu sehingga deklarasi yang diperlukan sebagai berikut:

```
FILE *nama_file;
```

2. Membuka dan Menutup File

3.

Untuk membuka file kita membutuhkan fungsi **fopen**("letak_file*/nama_file, dan format",jenis_operasi), untuk menutup file kita bisa menggunakan **fclose(nama_file)**. Jika kita melakukan operasi yang membutuhkan untuk membuka file **lebih dari satu** maka bisa menggunakan **fcloseall(void)** untuk menutup semua file. Fopen merupakan fungsi dari header "stdio.h".

Contoh penggunaan:



```
FILE *fp; //deklarasi variabel fp atau membuat file
fp = fopen("file.txt", "w+"); //variabel fp sekarang berisi file.txt
```

Gambar 6.1 Contoh Membuka File

Adapun beberapa mode atau operasi dalam file diantaranya:

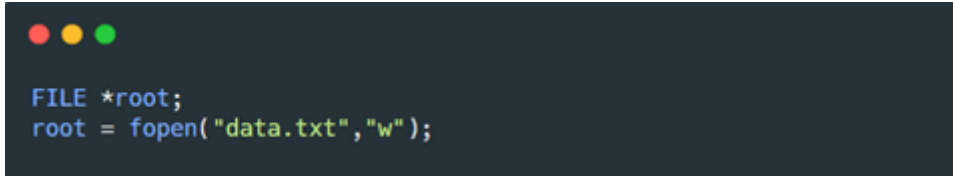
Tabel 6.1 Mode atau operasi dalam file

Mode	Arti
r	Membaca file (file harus sudah ada)
w	Menulis file (file yang sudah ada akan dihapus)
a	Membuka file yang sudah ada dan pada prosesnya dilakukan penambahan saat menulis (jika file belum ada, otomatis akan dibuat)
r+	Membaca file, tetapi juga memiliki fungsi lain yaitu dapat menulis
w+	Menulis file tetapi juga dapat membaca (file yang sudah ada akan dihapus)
a+	Membuka file yang sudah ada dan prosesnya dilakukan penambahan saat menulis dan dapat membaca file

3. Read and Write File

a. Write FILE

Untuk menulis file di bahasa C kamu bisa menggunakan sebuah kata kunci “w” untuk menulis saja atau “w+” untuk menulis , setelah menulis terus dibaca file tersebut . dibawah ini contoh *syntax* menulis file :



```
FILE *root;  
root = fopen("data.txt","w");
```

Gambar 6.2 Contoh Menulis File

b. **Read FILE**

Untuk membaca file di bahasa C kamu bisa menggunakan kata kunci “r” untuk membaca saja dan “r+” untuk membaca serta dapat menulis file setelah dibaca.



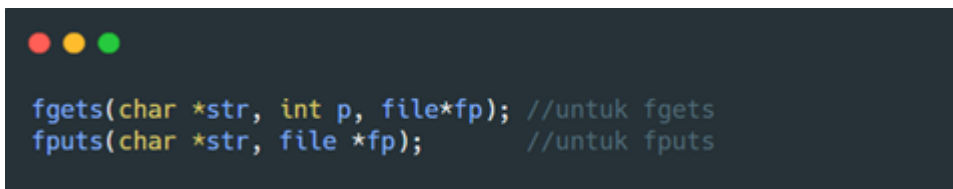
```
FILE *root;  
root = fopen("data.txt","r");
```

Gambar 6.3 Contoh Membaca File

c. **Bentuk String**

Sebelumnya kita harus tahu bahwa string merupakan tipe data yang memuat sebuah list atau array dari char sehingga apabila char hanya bisa menerima input sebanyak 1 karakter maka string bisa menerima lebih dari satu karakter atau bisa terangkai menjadi sebuah kata. Jadi string memiliki cara tersendiri dalam menuliskan nilainya ke dalam sebuah file yaitu dengan cara : fgets() dan fputs(). Jika diperhatikan maka pada setiap akhir kata get atau put di tambahkan “f” yang berarti file dan “s” yang berarti string.

Penulisannya sebagai berikut:



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

Gambar 6.4 Contoh Bentuk String

Catatan:

- char *str menyesuaikan dengan nama variabel string yang ingin kita masukkan kedalam file atau yang akan kita baca dari file.
- int p merupakan panjang dari string atau filenya.
- file *fp merupakan variabel file yang ingin kita akses.

Contoh Program Write FILE :

Tabel 6.2 Contoh Program Write FILE

Source Code
 <pre> #include <stdio.h> int main() { FILE *root; root = fopen("data.txt", "w"); if(root == NULL) { printf("error"); } else { fprintf(root, "masuk pak echo!!!"); printf("\n data berhasil masuk ke file coba cek file \n"); } return 0; } </pre>
Gambar 6.5 Program Menulis File
Output
 <pre> data berhasil masuk ke file coba cek file > </pre>
Gambar 6.6 Output Program Menulis File

6.3.3 CRUD

CRUD adalah singkatan dari Create, Read, Update, dan Delete). CRUD biasanya digunakan dalam database. Database aplikasi atau web sangat memerlukan algoritma CRUD. Tanpa algoritma CRUD database website tidak dapat berjalan.

Contoh Program:

Table 6.1 Source Code CRUD

Source Code

```

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

int main( )
{
    FILE *file ;

    printf( "==> MENULIS DATA <==\n" ) ;
    char data[50] = "Algoritma dan Pemrograman";
    file = fopen("Test.txt", "w" ) ;

    if ( file == NULL )
    {
        printf( "Test.txt file gagal dibuka" ) ;
    }
    else
    {
        if ( strlen ( data ) > 0 )
        {
            fputs(data, file) ;
            fputs("\n", file) ;
        }
        fclose(file) ;

        printf("Data berhasil ditulis di file Test.txt\n");
    }

    printf( "\n==> MEMBACA DATA DARI FILE <==\n" ) ;
    char bacadata[50];
    file = fopen("Test.txt", "r" ) ;

    if ( file == NULL )
    {
        printf( "File Test.txt gagal dibuka" ) ;
    }
    else
    {
        while( fgets ( bacadata, 50, file ) != NULL )
        {
            printf( "%s" , bacadata ) ;
        }
        fclose(file) ;

        printf("Data diatas dari file Test.txt\n");
    }

    printf("\n==> MENAMBAHKAN DATA KE FILE <==\n");
    file = fopen("Test.txt", "a");
    char bacalagi[50];
    fprintf(file, "Pemrograman Berorientasi Objek");
    fclose(file) ;

    printf("\n==> MENGHAPUS ISI DARI FILE <==\n");
    int t = 0;
    int i;

    for(i = 0; i < 100; i++) {
        if(t == 0) {
            file = fopen("Test.txt", "w");
            t++;
        }
        else {
            file = fopen("Test.txt", "a");
        }
    }
    fclose(file) ;

    return 0;
}

```

Gambar 6.7 Contoh Algoritma CRUD

Output

```

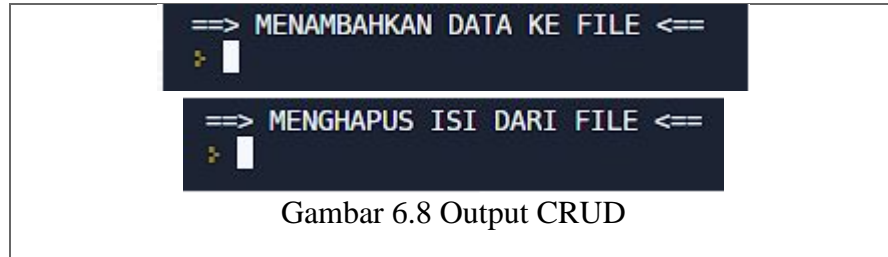
==> MENULIS DATA <==
Data berhasil ditulis di file Test.txt

```

```

==> MEMBACA DATA DARI FILE <==
Algoritma dan Pemrograman
Data diatas dari file Test.txt

```



Gambar 6.8 Output CRUD

Daftar Pustaka

Asisten Praktikum Lab. RAID, C. S. S., 2022. *MODUL PRAKTIKUM ALGORITMA DAN PEMROGRAMAN*. Bandung: Laboratorium RAID.