

LAPORAN FINAL PROJECT
PRAKTIKUM ALGORITMA & PEMROGRAMAN 2021



Dosen Pengampu :

Dra. Luh Gede Astuti,M.Kom.

Disusun Oleh :

Wahyu Vidiadivani	2008561082
Kompiang Gede Sukadharma	2008561083
I Gusti Bgs Darmika Putra	2008561094
I Made Teja Sarmandana	2008561098
Anak Agung Hendra Widiastana Tusan	2008561102
I Putu Agus Arya Wiguna	2008561109

INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS UDAYANA
2021

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Ida Sang Hyang Widhi Wasa, karena atas rahmat-Nya, penulis diberikan kelancaran dalam pembuatan Laporan Praktikum Algoritma Pemrograman yang bertujuan untuk memenuhi kriteria penilaian tugas dengan tepat waktu yaitu 30 Mei 2021. Terima kasih kami ucapkan kepada Dosen Pengampu Mata Kuliah Algoritma dan Pemrograman ini yaitu **Dra. Luh Gede Astuti, M.Kom.** yang telah membimbing dalam mata kuliah ini. Tidak terlepas juga saya ucapkan terima kasih kepada Kakak **Tryana Mertayasa** dan **Ni Putu Sintia Wati** selaku Asisten Dosen yang telah membantu keberlangsungan proses belajar mengajar mata kuliah ini.

Selama pembuatan laporan ini, tim penulis tidak lepas dari banyak kesulitan dalam pembuatannya. Penulis mendapat banyak masukan dan bimbingan dari berbagai pihak sehingga tidak lupa tim penulis mengucapkan terimakasih sebesar-besarnya kepada saudara, pembimbing dan orang tua yang selalu memberi motivasi dan dukungan, serta teman-teman mahasiswa Program Studi Informatika yang juga mengikuti mata kuliah Algoritma Pemrograman ini.

Tim penulis menyadari bahwa di dalam Laporan Final Project Praktikum Algoritma Pemrograman ini masih terdapat banyak kesalahan dan jauh dari kata sempurna. Maka dari itu, tim penulis memohon maaf apabila terdapat kesalahan kata maupun penulisan di dalam Laporan Final Project Praktikum Algoritma Pemrograman ini. Dan kami mohon untuk memberi masukan dan saran yang dapat membangun agar tugas kedepannya dapat kami selesaikan dengan lebih baik lagi. Semoga makalah ini dapat bermanfaat bagi para pembacanya dan dapat menambah wawasan bagi kita semua.

Denpasar, 26 Mei 2021

Tim Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
BAB I LATAR BELAKANG	1
BAB II LANDASAN TEORI	2
2.1. Pendahuluan	2
2.2. Operasi File	2
2.3. Penyeleksian Kondisi	7
2.4. Linked List.....	8
2.5. Struct.....	9
2.6. Modular Function	11
BAB III DESAIN & METODE	12
3.1. Flowchart.....	12
3.2. Metode - metode penyusunan program	12
BAB IV HASIL & IMPLEMENTASI.....	18
4.1. Kode	18
BAB IV PENUTUP	50
4.1. Kesimpulan.....	50
4.2. Saran	51
DAFTAR PUSTAKA	52

BAB I

LATAR BELAKANG

Daftar tugas-tugas yang harus dikerjakan pada suatu rentang waktu. Biasanya dibuat harian, mingguan, atau bulanan. Ini adalah bagian dari perencanaan yang dapat dibuat dengan *to do list*. Penerapan pemrograman dalam dunia kerja dapat dilihat seperti program *to do list*. Program ini adalah program yang dapat meningkatkan produktivitas pengguna dengan menggunakan sistem *to do list*. Dengan menggunakan *to do list* pengguna dapat manajemen pekerjaan yang akan dilakukan dalam jangka waktu beberapa hari. *To do list* bisa dibuat untuk tim, atau untuk pribadi. Tentu saja daftar data-data ini tidak harus dibuat dalam bentuk tertulis. Kita bisa saja mengagendakan pekerjaan-pekerjaan kita kedalam sebuah program, tanpa mencatatnya secara manual. Pada program yang kami buat ini memiliki berbagai menu untuk membuat *to do list* dengan properti : nama tugas, kelompok tugas (berfungsi untuk mengelompokkan tugas), prioritas, *deadline* (informasi tentang waktu tersisa), mengedit properti *to do list*, menghapus *to do list*, melihat *to do list* dengan beberapa cara yaitu : melihat semua *to do list*, mencari *to do list*, dan yang terakhir menu yang tersedia yaitu penyimpanan *to do list* agar data pengguna tidak hilang saat program dihentikan. Kegunaan dari prioritas yaitu *user* diharapkan dapat dengan mudah memilih pekerjaan yang akan diselesaikan. Maka dengan menggunakan *to do list* pengguna dapat manajemen pekerjaan yang akan dilakukan dengan jangka waktu yang efektif.

BAB II

LANDASAN TEORI

2.1.Pendahuluan

Seringkali untuk program-program aplikasi sistem informasi, data perlu disimpan secara permanen untuk keperluan lebih lanjut. Data dapat disimpan di external memory seperti di disk, floppy disk, dan UFD (USB Flash Disk) atau pun di internal memory sendiri pada penyimpanan internal memory mempunyai sifat volatile dan relatif lebih kecil dibandingkan dengan external memory. Pada bab ini akan membahas tentang operasi input/output file dan manipulasi file. Operasi i/o file melibatkan pembacaan dari file atau perekaman ke file. Manipulasi file melibatkan operasi pengecekan keberadaan file di disk, mengganti nama, menghapus file, dan lain- lain.

2.2.Operasi File

- **Pendahuluan**

File adalah sebuah organisasi dari sejumlah record. Masing-masing record bisa terdiri dari satu atau beberapa field. Setiap field terdiri dari satu atau beberapa byte.

- **Membuka File**

Untuk membuka atau mengaktifkan file, fungsi yang digunakan adalah fungsi `fopen()`. File dapat berupa file biner atau file teks. File biner adalah file yang pola penyimpanan di dalam disk dalam bentuk biner, yaitu seperti bentuk pada memori (RAM) komputer. File teks adalah file yang pola penyimpanan datanya dalam bentuk karakter. Penambahan yang perlu dilakukan untuk menentukan mode teks atau biner adalah “t” untuk file teks dan “b” untuk file biner. Prototype fungsi `fopen()` ada di header fungsi “`stdio.h`”

- **Bentuk Umum**

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

Keterangan :

- namafile adalah nama dari file yang akan dibuka/diaktifkan.
- mode adalah jenis operasi file yang akan dilakukan terhadap file.

Mode	Keterangan
“r”	Membuka file yang telah ada untuk dibaca. Jika file belum ada, pembukaan file tidak akan berhasil dan fungsi fopen akan bernilai NULL.
“w”	Membuat file baru untuk ditulis. Jika file telah ada maka file lama akan dihapus.
“a”	Membuka file yang telah ada untuk ditambah dengan data baru yang akan diletakan di akhir file. Jika file belum ada akan dibuat file baru.
“r+”	Sama dengan “r” tetapi selain file dapat dibaca juga dapat ditulis.
“w+”	Sama dengan “w” tetapi selain file dapat ditulis juga dapat dibaca.
“a+”	Sama dengan “a” tetapi selain file dapat ditulis file juga dapat dibaca

- **Menutup File**

Untuk menutup file kita dapat menggunakan fungsi `fclose()`. Fungsi ini digunakan untuk menutup file apabila file sudah tidak diproses lagi. Karena adanya keterbatasan jumlah file yang dapat dibuka secara serentak, maka file harus ditutup.

- Prototype fungsi `fclose()` ada di header file “`stdio.h`”
- Bentuk Umum :

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

atau

```
int fcloseall(void);
```

- **Menulis File**

Write file artinya kita membuka file dalam mode write (siap untuk ditulis). Dengan menggunakan format yang sama dengan diatas, kita akan mencoba membuat file baru dengan isi **NAMA#UMUR. Menyimpan data ke file bisa menggunakan perintah** `fprintf(namavariabelfile, format);`

- **Membaca File**

Sesuai namanya, disini kita akan membuka file sesuai namanya lalu membaca isi filenya. Untuk membaca isi file lalu menyimpannya ke dalam variabel, bisa dengan menggunakan `fscanf(namavariabelfile, format);`

- **Menambahkan File**

Append artinya menambahkan data pada file baris terakhir. Jika belum ada data/filenya, maka append akan membuatkan file baru. Contoh :

```
int main() {
```

```
char nama[100];

int umur;

printf("Masukkan nama : "); scanf("%[^\\n]", &nama);
fflush(stdin);

printf("Masukkan umur : "); scanf("%d", &umur);
fflush(stdin);


FILE *out=fopen("test.txt","a");

fprintf(out,"%s#%d\\n",nama, umur);

fclose(out);

printf("Sukses menambah data.");

getchar();

return 0;

}
```


- **Meletakkan Data Ke Penyangga**

Bahasa C membedakan lima macam bentuk data untuk diletakkan di penyangga (buffer), yaitu karakter, integer, string, terformat, dan blok data. Untuk masing-masing data ini fungsi pustaka yang digunakan berbeda yaitu sebagai berikut:

Fungsi pustaka	Penjelasan
fputc()	Meletakkan sebuah nilai karakter ke buffer untuk direkam ke file.
fgetc()	Membaca sebuah nilai karakter dari file untuk diletakkan di buffer.
putw()	Meletakkan sebuah nilai integer ke buffer untuk direkam ke file.
getw()	Membaca sebuah nilai integer dari file untuk diletakkan di buffer.
fputs()	Meletakkan sebuah nilai string ke buffer untuk direkam ke file.
fgets()	Membaca sebuah nilai string dari file untuk diletakkan di buffer.
fprintf()	Meletakkan sebuah data terformat di buffer untuk direkam ke file.
fscanf()	Membaca sebuah data terformat dari file untuk diletakkan di buffer.
fwrite()	Meletakkan sebuah blok data ke buffer untuk direkam ke file.
fread()	Membaca sebuah struktur data dari file untuk diletakkan di buffer.

2.3. Penyeleksian Kondisi

Pada umumnya, suatu permasalahan yang kompleks mengandung suatu penyeleksian kondisi atau dikatakan permasalahan tersebut memiliki beberapa alternatif pelaksanaan aksi. Dengan menyeleksi suatu kondisi, maka selanjutnya dapat ditentukan tindakan apa yang harus dilakukan, tergantung pada hasil kondisi yang diseleksi tersebut. Maka, penyeleksian kondisi merupakan cara untuk menyelesaikan suatu permasalahan yang kompleks dalam bahasa C. Dengan cara mengarahkan perjalanan suatu proses yang bersifat terstruktur artinya yaitu suatu aksi hanya dikerjakan apabila persyaratan atau kondisi tertentu terpenuhi. Metode – metode percabangan (Muhardian, 2019) :

- a. If : Percabangan if merupakan percabangan yang hanya memiliki satu blok pilihan saat kondisi bernilai benar.
- b. If ... else : Percabangan if/else merupakan percabangan yang memiliki dua blok pilihan. Blok pilihan pertama untuk kondisi benar, dan pilihan kedua untuk kondisi salah (else).
- c. If- else if- else : Percabangan if/else/if merupakan percabangan yang memiliki lebih dari dua blok pilihan.
- d. Nested if : Percabangan nested if merupakan percabangan if dengan struktur yang lebih kompleks. dimana didalam sebuah pernyataan if terdapat pernyataan if lainnya, dengan kata lain terdapat sebuah kondisi if didalam if. penggunaan struktur if bercabang biasa digunakan untuk pemilihan beberapa pernyataan bertingkat, ketika sebuah pernyataan if dijalankan dan bernilai true maka akan terdapat pernyataan if lainnya pada blok tersebut.
- e. Penyeleksian dengan operator ternary : Operator Ternary merupakan operator yang melibatkan tiga buah operand. Operator ini dilambangkan dengan tanda ?: dan berguna untuk

melakukan pemilihan terhadap nilai tertentu dimana pemilihan tersebut didasarkan atas ekspresi tertentu.

- f. Switch.. case : Percabangan switch-case adalah bentuk lain dari percabangan if/else/if percabangan code program dimana kita membandingkan isi sebuah variabel dengan beberapa nilai. Jika proses membandingkan tersebut benar, maka program yang terdapat di dalamnya akan berjalan. . Jadi, terdapat dua perintah di sini, yaitu : switch dan case. Di sini seorang programmer dapat membuat blok kode (case) sebanyak yang diinginkan di dalam blok switch. Pada <value>, kita bisa isi dengan nilai yang nanti akan dibandingkan dengan varabel. Setiap case harus diakhiri dengan break. Khusus untuk default, tidak perlu diakhiri dengan break karena dia terletak di bagian akhir. Pemberian break bertujuan agar program berhenti mengecek case berikutnya saat sebuah case terpenuhi.

2.4.Linked List

1. Definisi

Linked List atau dikenal juga dengan sebutan senarai berantai adalah struktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada Linked List disebut Node.

Linked List dalam ilmu komputer merupakan sebuah struktur data yang digunakan untuk menyimpan sejumlah objek data biasanya secara terurut sehingga memungkinkan penambahan, pengurangan, dan pencarian data atas elemen data yang tersimpan dalam senarai dilakukan secara lebih efektif. Pada praktiknya sebuah struktur data memiliki elemen yang digunakan untuk saling menyimpan rujukan antara satu rujukan dengan lainnya sehingga membentuk sebuah senarai abstrak, tiap – tiap elemen yang terdapat pada senarai abstrak ini seringkali

disebut sebagai node karena mekanisme rujukan yang saling terkait inilah disebut sebagai senarai berantai (Dharma, 2017).

2. Operasi pada Linked List

Double linked list memiliki beberapa operasi dasar pada list, misalkan penyisipan, penghapusan, menampilkan maju, dan menampilkan mundur.

1. Insert = Istilah Insert berarti menambahkan sebuah simpul baru ke dalam suatu linked list.
2. Konstruktor = Fungsi ini membuat sebuah linked list yang baru dan masih kosong.
3. IsEmpty = Fungsi ini menentukan apakah linked list kosong atau tidak.
4. Find First = Fungsi ini mencari elemen pertama dari linked list
5. Find Next = Fungsi ini mencari elemen sesudah elemen yang ditunjuk now.
6. Retrieve = Fungsi ini mengambil elemen yang ditunjuk oleh now. Elemen tersebut lalu dikembalikan oleh fungsi.
7. Update = Fungsi ini mengubah elemen yang ditunjuk oleh now dengan isi dari sesuatu.
8. Delete Now = Fungsi ini menghapus elemen yang ditunjuk oleh now. Jika yang dihapus adalah elemen pertama dari linked list (head), head akan berpindah ke elemen berikut.

2.5.Struct

1. Definisi

Struct adalah tipe data bentukan yang berisi kumpulan variabel-variabel yang bernaung dalam satu nama yang sama. Berbeda dengan array yang berisi kumpulan variabel yang bertipe data sama, struct dapat memiliki variabel-variabel yang bertipe data sama atau berbeda, bahkan bisa menyimpan variabel yang bertipe

data array atau struct. Variabel-variabel yang menjadi anggota struct disebut dengan elemen struct.

2. Karakteristik Struct

- Sekumpulan data
- Tipe datanya boleh berbeda
- Menggunakan nama yang sama
- Dibedakan melalui nama fieldnya
- Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah kesatuan (dalam bahasa PASCAL, struktur disebut dengan record).
- Variabel-variabel yang membentuk suatu struktur, selanjutnya disebut sebagai elemen dari struktur atau field.

3. Ilustrasi Struct

- Struct bisa diumpamakan sebagai sebuah class, misalnya: Mahasiswa.
- Struct Mahasiswa memiliki properti atau atribut atau variabel yang melekat padanya:
 - a. NIM yaitu karakter sejumlah 9
 - b. Nama yaitu karakter
 - c. IPK yaitu bilangan pecahan

4. Deklarasi Struct

- Suatu struktur didefinisikan dengan menggunakan kata kunci struct.

```
struct date {
    int  month;
    int  day;
    int  year;
};

struct date {
    int month, day, year;
};
```

- Contoh di atas mendefinisikan sebuah tipe data struktur bernama struct date yang memiliki tiga buah elemen (field) berupa: month, day, year.

2.6.Modular Function

Modular Function merupakan salah satu dari bentuk Pemrograman Modular. Pemrograman Modular sendiri merupakan suatu teknik pemrograman dimana program yang bersifat kompleks atau besar dibagi-bagi menjadi beberapa program yang lebih sederhana atau kecil. Penggunaan cara ini adalah untuk memudahkan program agar bisa dimengerti, jika terdapat kesalahan dalam proses pengolahan data, akan mudah untuk mencari error atau kesalahannya.

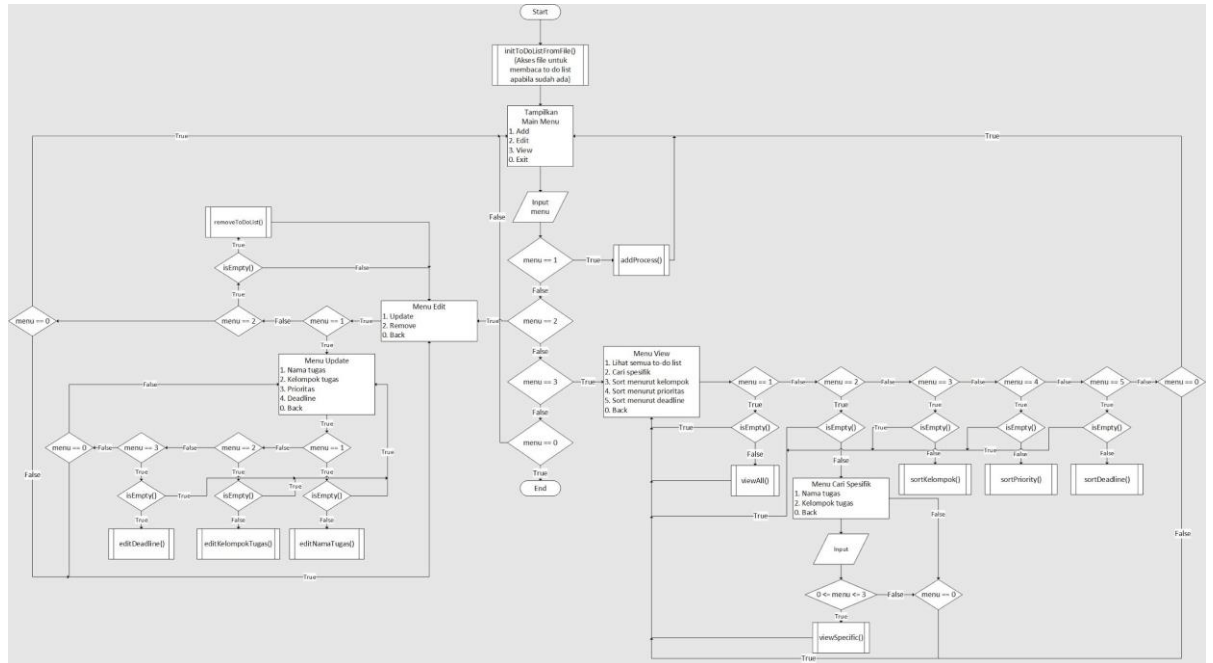
Fungsi secara umum terbagi menjadi fungsi yang sudah disediakan oleh suatu bahasa pemrograman (Standard Library) yang dimana fungsi ini, kita harus mendeklarasikan terlebih dahulu library yang akan digunakan, yaitu dengan menggunakan preprosesor direktif dan fungsi yang dibuat oleh programmer sendiri (User Define Function) yang dimana fungsi ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan dijadikan satu ke dalam suatu library buatan programmer yang kemudian dimasukan jika ingin menggunakannya. Sebuah nama fungsi harus dideklarasikan tipe datanya sehingga fungsi tersebut dapat dikatakan mengembalikan sebuah nilai.

Dalam pembuatan fungsi, hal-hal yang perlu diperhatikan adalah data yang diperlukan sebagai masukan, informasi apa yang harus diberikan oleh fungsi dibuat ke pemanggilnya, dan algoritma apa yang harus digunakan untuk mengolah data menjadi informasi.

BAB III

DESAIN & METODE

3.1. Flowchart



3.2. Metode - metode penyusunan program

Dalam penyusunan program ini, adapun metode - metode yang digunakan di dalam bahasa C yaitu:

1. Tipe Data Dasar

Sesuai dengan namanya, tipe data dasar adalah tipe data paling dasar yang tersedia di dalam bahasa pemrograman C. Terdapat 3 jenis tipe data dasar:

- Char: tipe data yang berisi 1 huruf atau 1 karakter.
- Integer: tipe data untuk menampung angka bulat.
- Float: tipe data untuk menampung angka pecahan.

- Tipe data dasar disebut juga sebagai Primary Data Type, Fundamental Data Types atau Basic Data Type.

Implementasi dalam program:

```
void addProcess(To_Do_List_Node **main_node) {
    To_Do_List_Node temp_input;
    date date_now;
    char temp_text_prioritas[50], temp_text_date[50], *token;
    int back, back_to = 1;
    getDate(&date_now);
```

2. Tipe Data Turunan

Tipe data turunan berasal dari tipe data dasar yang dikelompokkan atau di modifikasi. Terdapat 3 tipe data turunan di dalam bahasa pemrograman C:

- Array: Tipe data yang terdiri dari kumpulan tipe data dasar. Tipe data tersebut harus 1 jenis.
- Structure: Tipe data yang terdiri dari kumpulan tipe data dasar. Tipe data tersebut bisa lebih dari 1 jenis.
- Pointer: Tipe data untuk mengakses alamat memory secara langsung.
- Tipe data turunan disebut juga sebagai Derived Data Type.

Implementasi dalam program:

```
1  #ifndef __DATA_TYPE_
2  #define __DATA_TYPE_
3  typedef struct To_Do_List_Node {
4      struct To_Do_List_Node *next;
5      char nama_tugas[50], kelompok_tugas[50];
6      int prioritas;
7      int dl_dd, dl_mm, dl_yyyy;
8  } To_Do_List_Node;
9
10
11  // Tambahkan ADT yang kalian perlukan
12
13  typedef struct date {
14      int dd, mm, yyyy;
15  } date;
16  #endif
```


3. Function :

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilmnya. Dalam penggunaan bahasa C pada suatu kompilator fungsi dapat dibagi menjadi dua : fungsi pustaka (fungsi yang disediakan oleh kompilator) dan fungsi yang didefinisikan atau dibuat oleh programmer.

Implementasi dalam program:

```

7 > void editNamaTugas(To_Do_List_Node **main_node){ ...
29
30
31 > void editKelompokTugas(To_Do_List_Node **main_node){ ...
57
58 > void editPrioritas(To_Do_List_Node **main_node){ ...
86
87 > void editDeadline(To_Do_List_Node **main_node){ ...
128
129 > void delete(To_Do_List_Node **main_node){ ...
167
168 > void overwriteFile(To_Do_List_Node **main_node){ ...
177
178
179 |

```

4. Penyeleksian kondisi:

Dengan menyeleksi suatu kondisi, maka selanjutnya dapat ditentukan tindakan apa yang harus dilakukan, tergantung pada hasil kondisi yang diseleksi tersebut. Maka, penyeleksian kondisi merupakan cara untuk menyelesaikan suatu permasalahan yang kompleks dalam bahasa C.

Implementasi dalam program:

```

128     if (back_to > 0) back_to--;
129
130     while (1) {
131         if (back == 0 && back_to == 0) {
132             printf(" * Nama Tugas : ");
133             scanf("%s", temp_input.nama_tugas);
134             getchar();
135         }
136         if (strcmp(temp_input.nama_tugas, "-h") == 0) {
137             helpAddProcess(1);
138             continue;
139         }
140         else if (strcmp(temp_input.nama_tugas, "-b") == 0) {
141             back = 1;
142             break;
143         }
144         break;
145     }
146
147     if (back == 1) break;
148     if (back_to > 0) back_to--;

```

5. Operasi file:

File adalah sebuah organisasi dari sejumlah record. Masing-masing record bisa terdiri dari satu atau beberapa field. Setiap field terdiri dari satu atau beberapa byte.

Implementasi dalam program:

```
95 void saveToFile(To_Do_List_Node *temp_input) {
96     FILE *write_to_file = fopen("file\\to_do_list.txt", "a");
97     fprintf(write_to_file, "%s|s|d|d/%d/%d\n", temp_input->nama_tugas, temp_input->kelompok_tugas, \
98         temp_input->prioritas, temp_input->dl_dd, \
99         temp_input->dl_mm, temp_input->dl_yyyy);
100     fclose(write_to_file);
101 }
```

```
21 void initToDoListFromFile(To_Do_List_Node **main_node) {
22     FILE *init_node_from_file = fopen("file\\to_do_list.txt", "r");
23     if (init_node_from_file != NULL) {
24         To_Do_List_Node temp_node;
25         char temp[200];
26         char *token;
```

6. Linked List:

adalah suatu struktur data linier. Berbeda dengan array yang juga merupakan struktur data linier dan tipe data komposit, linked list dibentuk secara dinamik. Pada saat awal program dijalankan elemen linked list belum data. Elemen linked list (disebut node) dibentuk sambil jalan sesuai instruksi. Apabila setiap elemen array dapat diakses secara langsung dengan menggunakan indeks, sebuah node linked list diakses dengan menggunakan pointer yang mengacu (menunjuk) ke node tersebut. Awal atau kepala linked list harus diacu sebuah pointer yang biasa diberi nama head. Pointer current (disingkat curr) digunakan untuk memindahkan pengacuan kepada node tertentu.

Implementasi dalam program:

```
> To_Do_List_Node *makeNode(To_Do_List_Node temp_input) {
    To_Do_List_Node *head;

    head = (To_Do_List_Node *) malloc(sizeof(To_Do_List_Node));
    head->next = NULL;
    strcpy(head->nama_tugas, temp_input.nama_tugas);
    strcpy(head->kelompok_tugas, temp_input.kelompok_tugas);
    head->prioritas = temp_input.prioritas;
    head->dl_dd = temp_input.dl_dd;
    head->dl_mm = temp_input.dl_mm;
    head->dl_yyyy = temp_input.dl_yyyy;
    return head;
}
```

```

47 void insert(To_Do_List_Node **main_node, To_Do_List_Node temp_input) {
48     To_Do_List_Node *tail_input_node, *temp_main_node = *main_node;
49     tail_input_node = makeNode(temp_input);
50     while(temp_main_node->next != NULL) temp_main_node = temp_main_node->next;
51     temp_main_node->next = tail_input_node;
52 }

```

7. Struct:

Struct adalah tipe data bentukan yang berisi kumpulan variabel-variabel yang bernaung dalam satu nama yang sama. Berbeda dengan array yang berisi kumpulan variabel yang bertipe data sama, struct dapat memiliki variabel-variabel yang bertipe data sama atau berbeda, bahkan bisa menyimpan variabel yang bertipe data array atau struct. Variabel-variabel yang menjadi anggota struct disebut dengan elemen struct.

Implementasi dalam program:

```

1  #ifndef __DATA_TYPE
2      #define __DATA_TYPE__
3  typedef struct To_Do_List_Node {
4      struct To_Do_List_Node *next;
5      char nama_tugas[50], kelompok_tugas[50];
6      int prioritas;
7      int dl_dd, dl_mm, dl_yyyy;
8  } To_Do_List_Node;
9
10
11     // Tambahkan ADT yang kalian perlukan
12
13  typedef struct date {
14      int dd, mm, yyyy;
15  } date;
16  #endif

```

8. Modular Function

Modular Function merupakan salah satu dari bentuk Pemrograman Modular. Pemrograman Modular sendiri merupakan suatu teknik pemrograman dimana program yang bersifat kompleks atau besar dibagi-bagi menjadi beberapa program yang lebih sederhana atau kecil.

Implementasi dalam program:

```
lib > C create.h > ...
1  #ifndef __CREATE_
2      #define __CREATE_
3      #include "abstract_data_type.h"
4      To_Do_List_Node *makeNode(To_Do_List_Node temp_input);
5      void initToDoListFromFile(To_Do_List_Node **main_node);
6      void insert(To_Do_List_Node **main_node, To_Do_List_Node temp_input);
7      void helpAddProcess(int menu);
8      void saveToFile(To_Do_List_Node *temp_input);
9      void addProcess(To_Do_List_Node **main_node);
10 #endif
```

```
1  #ifndef __EDIT_
2      #define __EDIT_
3      #include "abstract_data_type.h"
4      #include "universal_function.h"
5      void editNamaTugas(To_Do_List_Node **main_node);
6      void editKelompokTugas(To_Do_List_Node **main_node);
7      void editPrioritas(To_Do_List_Node **main_node);
8      void editDeadline(To_Do_List_Node **main_node);
9      void delete(To_Do_List_Node **main_node);
10     void overwriteFile(To_Do_List_Node **main_node);
11 #endif
12
```

```
lib > C universal_function.h > ...
1  #ifndef __UNIVERSAL_FUNCTION_
2      #define __UNIVERSAL_FUNCTION_
3      #include "abstract_data_type.h"
4      void clear();
5      void getDate(date *date_now);
6      char *lowerTheSentence(char *sentence);
7  #endif
```

```
1  #ifndef __VIEW_
2      #define __VIEW_
3      #include "abstract_data_type.h"
4      #include "universal_function.h"
5      int dayLeft(To_Do_List_Node *temp_main_node);
6      void viewAll(To_Do_List_Node *main_node, int menu);
7      void viewSpecific(To_Do_List_Node *main_node, int menu);
8      void swapValueLinkedList(To_Do_List_Node **node1, To_Do_List_Node **node2);
9      void copyLinkedList(To_Do_List_Node **copied_main_node, To_Do_List_Node *temp);
10     void selectionSortList(To_Do_List_Node **main_node, int menu);
11     void sortKelompok(To_Do_List_Node *main_node);
12     void sortPriority(To_Do_List_Node *main_node);
13     void sortDeadline(To_Do_List_Node *main_node);
14 #endif
```

```
lib > C our_module.h > ...
1  #ifndef __OUR_MODULE_
2      #define __OUR_MODULE_
3      #include "abstract_data_type.h"
4      #include "create.h"
5      #include "universal_function.h"
6      #include "view.h"
7      #include "edit.h"
8  #endif
```

BAB IV

HASIL & IMPLEMENTASI

4.1. Kode

- main.c

```
#include <stdio.h>
#include <string.h>
#include "lib/our_module.h"

int main() {
    int menu;
    To_Do_List_Node *main_node = NULL;
    date date_now;

    initToDoListFromFile(&main_node);
    while(1) {
        clearTheScreen();
        puts("=====");
        puts("| MENU UTAMA |");
        puts("=====");
        puts(" 1. Add");
        puts(" 2. Edit");
        puts(" 3. View");
        puts(" 0. Exit");
        puts("=====");
        printf("Input : ");
        scanf("%d", &menu);
        getchar();

        if (menu == 1) {
            clearTheScreen();
            addProcess(&main_node);
            tahan();
        }
        else if (menu == 2) {
            while (1) {
                clearTheScreen();
                puts("=====");
                puts("| MENU EDIT |");
                puts("=====");
                puts(" 1. Update");
                puts(" 2. Remove");
                puts(" 0. Back");
                puts("=====");
                printf("Input : ");
                scanf("%d", &menu);
                getchar();
```

```

if (menu == 1) {
    while (1) {
        clearTheScreen();
        puts("=====");
        puts("|      MENU UPDATE  |");
        puts("=====");
        puts(" 1. Nama tugas");
        puts(" 2. Kelompok tugas");
        puts(" 3. Prioritas");
        puts(" 4. Deadline");
        puts(" 0. Back");
        puts("=====");
        printf("Input : ");
        scanf("%d", &menu);
        getchar();
        if (menu == 1) {
            if (isEmpty(main_node)) {
                tahan();
                continue;
            }
            editNamaTugas(&main_node);
            tahan();
        }
        else if (menu == 2) {
            if (isEmpty(main_node)) {
                tahan();
                continue;
            }
            editKelompokTugas(&main_node);
            tahan();
        }
        else if (menu == 3) {
            if (isEmpty(main_node)) {
                tahan();
                continue;
            }
            editPrioritas(&main_node);
            tahan();
        }
        else if (menu == 4) {
            if (isEmpty(main_node)) {
                tahan();
                continue;
            }
            editDeadline(&main_node);
            tahan();
        }
        else if (menu == 0) {
            break;
        }
    }
}

```

```

                else {
                    puts("\nPilih antara angka 0
hingga 4");

                                getchar();
                                continue;
                }
            }
        }
    else if (menu == 2) {
        if (isEmpty(main_node)) {
            tahan();
            continue;
        }
        removeToDoList(&main_node);
        tahan();
    }
    else if (menu == 0) {
        break;
    }
    else {
        puts("\nPilih antara angka 0 hingga
2");

                                getchar();
                                continue;
        }
        overwriteFile(&main_node);
    }
}
else if (menu == 3) {
while (1) {
    clearTheScreen();
    puts("=====");
    puts("|      MENU VIEW      |");
    puts("=====");
    puts(" 1. Lihat semua to-do list");
    puts(" 2. Cari spesifik");
    puts(" 3. Sort menurut kelompok");
    puts(" 4. Sort menurut prioritas");
    puts(" 5. Sort menurut deadline");
    puts(" 0. Back");
    puts("=====");
    printf("Input : ");
    scanf("%d", &menu);
    getchar();

    if (menu == 1) {
        clearTheScreen();
        if (isEmpty(main_node)) {
            tahan();
            continue;
        }
    }
}
}

```

```

        viewAll(main_node, 0);
        tahan();
    }
    else if (menu == 2) {
        clearTheScreen();
        if (isEmpty(main_node)) {
            tahan();
            continue;
        }
        while(1) {
            clearTheScreen();
            puts("=====");
            puts("|      Cari Spesifik      |");
            puts("=====");
            puts(" 1. Nama tugas");
            puts(" 2. Kelompok tugas");
            puts(" 0. Back");
            puts("=====");
            printf("Input : ");
            scanf("%d", &menu);
            getchar();

            if (menu > 0 && menu < 3) {
                clearTheScreen();
                viewSpecific(main_node, menu);
                tahan();
            }
            else if (menu == 0) {
                break;
            }
            else {
                puts("\nPilih antara angka 0
                hingga 2");

                tahan();
                continue;
            }
        }
    }
    else if (menu == 3) {
        clearTheScreen();
        if (isEmpty(main_node)) {
            tahan();
            continue;
        }
        sortKelompok(main_node);
        tahan();
    }
    else if (menu == 4) {
        clearTheScreen();
        if (isEmpty(main_node)) {
            tahan();

```



```

        continue;
    }
    sortPriority(main_node);
    tahan();
}
else if (menu == 5) {
    clearTheScreen();
    if (isEmpty(main_node)) {
        tahan();
        continue;
    }
    sortDeadline(main_node);
    tahan();
}
else if (menu == 0) break;
else {
    puts("\nPilih antara angka 0 hingga
5");

    tahan();
    continue;
}
}
}
else if (menu == 0) break;
else {
    puts("\nPilih antara angka 0 hingga 3");
    tahan();
    continue;
}
}
return 0;
}

```

- **abstract_data_type.h**

```

#ifndef __DATA_TYPE_
#define __DATA_TYPE_
typedef struct To_Do_List_Node {
    struct To_Do_List_Node *next;
    char nama_tugas[50], kelompok_tugas[50];
    int prioritas;
    int dl_dd, dl_mm, dl_yyyy;

} To_Do_List_Node;

typedef struct date {
    int dd, mm, yyyy;
} date;
#endif

```

- **our_module.h**

```
#ifndef __OUR_MODULE_
#define __OUR_MODULE_
#include "abstract_data_type.h"
#include "create.h"
#include "universal_function.h"
#include "view.h"
#include "edit.h"
#include "delete.h"
#endif
```

- **create.h**

```
#ifndef __CREATE_
#define __CREATE_
#include "abstract_data_type.h"
To_Do_List_Node *makeNode(To_Do_List_Node temp_input);
void initToDoListFromFile(To_Do_List_Node
**main_node);
void insert(To_Do_List_Node **main_node,
To_Do_List_Node temp_input);
void helpAddProcess(int menu);
void saveToFile(To_Do_List_Node *temp_input);
void addProcess(To_Do_List_Node **main_node);
#endif
```

- **create.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "create.h"
#include "universal_function.h"

To_Do_List_Node *makeNode(To_Do_List_Node temp_input) {
    To_Do_List_Node *head;

    head = (To_Do_List_Node *)
    malloc(sizeof(To_Do_List_Node));
    head->next = NULL;
    strcpy(head->nama_tugas, temp_input.nama_tugas);
    strcpy(head->kelompok_tugas,
temp_input.kelompok_tugas);
    head->prioritas = temp_input.prioritas;
    head->dl_dd = temp_input.dl_dd;
    head->dl_mm = temp_input.dl_mm;
    head->dl_yyyy = temp_input.dl_yyyy;
    return head;
}
```

```

}

void initToDoListFromFile(To_Do_List_Node **main_node) {
    FILE *init_node_from_file =
fopen("file\\to_do_list.txt", "r");
    if (init_node_from_file != NULL) {
        To_Do_List_Node temp_node;
        char temp[200];
        char *token;
        while(fgets(temp, 200, init_node_from_file) !=
NULL) {
            token = strtok(temp, "|");
            strcpy(temp_node.nama_tugas, token);
            token = strtok(NULL, "|");
            strcpy(temp_node.kelompok_tugas, token);
            token = strtok(NULL, "|");
            temp_node.prioritas = atoi(token);
            token = strtok(NULL, "|");
            token = strtok(token, "/");
            temp_node.dl_dd = atoi(token);
            token = strtok(NULL, "/");
            temp_node.dl_mm = atoi(token);
            token = strtok(NULL, "/");
            temp_node.dl_yyyy = atoi(token);
            *main_node == NULL ? *main_node =
makeNode(temp_node) : insert(main_node, temp_node);
        }
    }
    fclose(init_node_from_file);
}

void insert(To_Do_List_Node **main_node, To_Do_List_Node
temp_input) {
    To_Do_List_Node *tail_input_node, *temp_main_node =
*main_node;
    tail_input_node = makeNode(temp_input);
    while(temp_main_node->next != NULL) temp_main_node =
temp_main_node->next;
    temp_main_node->next = tail_input_node;
}

void helpAddProcess(int menu) {
    if (menu == 1) {
        puts("\n\n== Nama Tugas ==");
        puts("Field ini hendaknya diisi");
        puts("nama tugas yang akan dikerjakan");
        puts("Contoh : Membuatkan pacar kejutan\n\n");
    }
    else if (menu == 2) {
        puts("\n\n== Kelompok Tugas ==");
    }
}

```

```

        puts("Field ini dibuat ntuk memudahkan Anda
dalam");
        puts("mengelompokkan tugas yang memiliki");
        puts("kesamaan menurut Anda dalam segi");
        puts("asal tugas, nama, dll");
        puts("Contoh : Alpro");
        puts("\n== CATATAN ==");
        puts("Pengelompokan ini tidak sensitive case");
        puts("Alpro dan alpro akan dianggap sama");
        puts("\n");
    }
    else if (menu == 3) {
        puts("\n\n== Prioritas ==");
        puts("Untuk membantu Anda dalam mengatur
jadwal");
        puts("prioritas ini dapat menyimpan tingkat");
        puts("prioritas agar dapat ditampilkan
nantinya");
        puts("== Tingkat-tingkat prioritas dari 1 - 4");
        puts("1. Penting & mendesak");
        puts("2. Tidak penting & mendesak");
        puts("3. Penting & tidak mendesak");
        puts("4. Tidak penting & tidak mendesak\n\n");
    }
    else if (menu == 4) {
        puts("\n\n== Deadline ==");
        puts("Deadline membantu anda untuk menyimpan");
        puts("batas akhir dari pengumpulan atau batas
selesai");
        puts("tugas Anda");
        puts("Format yang dapat diterima program ini
adalah");
        puts("dd/mm/yy");
        puts("Contoh : 22/05/2021\n\n");
    }
}

void saveToFile(To_Do_List_Node *temp_input) {
    FILE *write_to_file = fopen("file\\to_do_list.txt",
"a");
    fprintf(write_to_file, "%s|%s|%d|%d/%d/%d\n",
temp_input->nama_tugas, temp_input->kelompok_tugas, \

temp_input->prioritas, temp_input->dl_dd, \

temp_input->dl_mm, temp_input->dl_yyyy);
    fclose(write_to_file);
}

void addProcess(To_Do_List_Node **main_node) {
    To_Do_List_Node temp_input, *temp_travel;

```

```

    date date_now;
    char    temp_text_prioritas[50],    temp_text_date[50],
*token;
    int back, back_to = 1;
    getTheDate(&date_now);

    printf(" ===== Guide ===== \n");
    printf("Selamat datang di add proses\n");
    printf("apabila Anda bingung tentang\n");
    printf("properti yang akan diisi,\n");
    printf("Anda dapat menginputkan -h\n");
    printf("untuk memanggil menu help\n\n");
    printf("Apabila anda ingin kembali\n");
    printf("ke menu sebelumnya, Anda\n");
    printf("dapat menginputkan -b untuk\n");
    printf("kembali ke menu sebelumnya\n\n");

    printf("=====\n");
    printf("| Input to-do list |\n");
    printf("=====\n");

    while (1) {
        temp_travel = *main_node;
        back = 0;

        if (back_to > 0) back_to--;

        while (1) {
            if (back == 0 && back_to == 0) {
                printf(" * Nama Tugas : ");
                scanf("%s", temp_input.nama_tugas);
                getchar();
                while (temp_travel != NULL) {
                    if
(!strcmp(lowerTheSentence(temp_travel->nama_tugas),
lowerTheSentence(temp_input.nama_tugas))) {

                        puts("\n=====");
                        puts("=== MASUKKAN NAMA TUGAS YANG
BERBEDA ===");
                        puts("=== NAMA TUGAS INI SUDAH
DIGUNAKAN ===");

                        puts("=====\n");
                        back = 1;
                        break;
                    }
                    temp_travel = temp_travel->next;
                }
                if (back == 1) {
                    back = 0;

```

```

        continue;
    }
}
if (strcmp(temp_input.nama_tugas, "-h") == 0) {
    helpAddProcess(1);
    continue;
}
else if (strcmp(temp_input.nama_tugas, "-b") ==
0) {
    back = 1;
    break;
}
break;
}

if (back == 1) break;
if (back_to > 0) back_to--;

while (1) {
    if (back == 0 && back_to == 0) {
        printf(" * Kelompok Tugas : ");
        scanf("%[^\\n]",
temp_input.kelompok_tugas);
        getchar();
    }
    if (strcmp(temp_input.kelompok_tugas, "-h") == 0)
{
        helpAddProcess(2);
        continue;
    }
    else if (strcmp(temp_input.nama_tugas, "-b") ==
0) {
        back = 1;
        back_to = 1;
        break;
    }
    break;
}

if (back == 1) continue;
if (back_to > 0) back_to--;

while(1) {
    if (back == 0 && back_to == 0) {
        printf(" * Prioritas (1-4): ");
        scanf("%[^\\n]", temp_text_prioritas);
        getchar();
        temp_input.prioritas
=
atoi(temp_text_prioritas);
        if (!(temp_input.prioritas >= 1 &&
temp_input.prioritas <= 4)) {

```

```

        puts("\nMasukan angka antara 1 hingga
4\n");
        continue;
    }
}
if (strcmp(temp_text_prioritas, "-h") == 0) {
    helpAddProcess(3);
    continue;
}
else if (strcmp(temp_text_prioritas, "-b") == 0)
{
    back = 1;
    back_to = 2;
    break;
}
break;
}

if (back == 1) continue;
if (back_to > 0) back_to--;

while (1) {
if (back == 0 && back_to == 0) {
    printf(" * Deadline (dd/mm/yyyy) : ");
    scanf("%[^%\n]", temp_text_date);
    getchar();

    token = strtok(temp_text_date, "/");
    temp_input.dl_dd = atoi(token);
    token = strtok(NULL, "/");
    temp_input.dl_mm = atoi(token);
    token = strtok(NULL, "/");
    temp_input.dl_yyyy = atoi(token);

    if ((temp_input.dl_dd < date_now.dd ||
temp_input.dl_mm < date_now.mm) \
        && temp_input.dl_yyyy <
date_now.yyyy) {

        puts("\n=====");
        puts("=== TIDAK BOLEH MEMASUKAN
TANGGAL ===");
        puts("=== SEBELUM TANGGAL HARI INI
===");

        puts("=====\n");
        continue;
    }
}
if (strcmp(temp_text_date, "-h") == 0) {
    helpAddProcess(4);

```

```

        continue;
    }
    else if (strcmp(temp_text_date, "-b") == 0) {
        back = 1;
        back_to = 3;
        break;
    }
    break;
}

if (back == 1) continue;
break;
}
if (back == 0) {
    saveToFile(&temp_input);
    *main_node == NULL ? *main_node =
makeNode(temp_input) : insert(main_node, temp_input);
}
}

```

- **edit.h**

```

#ifndef __EDIT__
#define __EDIT__
#include "abstract_data_type.h"
#include "universal_function.h"
void editNamaTugas (To_Do_List_Node **main_node);
void
    editKelompokTugas (To_Do_List_Node
**main_node);
void editPrioritas (To_Do_List_Node **main_node);
void editDeadline (To_Do_List_Node **main_node);
#endif

```

- **edit.c**

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "edit.h"
#include "universal_function.h"
#include "our_module.h"

void editNamaTugas (To_Do_List_Node **main_node) {
    To_Do_List_Node *temp = *main_node;
    char new[50], huruf[50];
    printf("Masukkan nama tugas yang ingin dicari: ");
    scanf("%s", &huruf);

    while(temp!=NULL) {
        if (strcmp(temp->nama_tugas, huruf)==0) {
            printf("Masukkan nama tugas baru: ");
            scanf("%s", &new);
            getchar();

```



```

        strcpy(temp->nama_tugas, new);
        puts("Data berhasil disimpan");
        return ;
    }
    temp=temp->next;
}

if (temp == NULL) {
    puts("DATA yang Anda cari tidak ada!");
}
}

void editKelompokTugas(To_Do_List_Node **main_node){
    To_Do_List_Node *temp = *main_node;
    char new[50], huruf[50];

    printf("Masukkan nama tugas yang ingin dicari: ");
    scanf(" %49[^\n]", &huruf);

    while(temp!=NULL){
        if (strcmp(temp->nama_tugas,huruf)==0){
            printf("Masukkan kelompok tugas baru: ");
            scanf(" %49[^\n]",&new);
            strcpy(temp->kelompok_tugas, new);
            puts("Data berhasil disimpan");
            return;
        }
        temp=temp->next;
    }

    if (temp == NULL) {
        puts("DATA yang Anda cari tidak ada!");
    }
}

void editPrioritas(To_Do_List_Node **main_node){
    To_Do_List_Node *temp = *main_node;
    int new;
    char huruf[50];

    printf("Masukkan nama tugas yang ingin dicari: ");
    scanf(" %49[^\n]", &huruf);

    while (temp!=NULL){
        if (strcmp(temp->nama_tugas,huruf)==0){
            printf("Masukkan prioritas tugas baru (1-
4): ");

```



```

                                puts("Data          berhasil
disimpan!");
                                break;
                                }
                                }
                                return;

                                }
                                temp=temp->next;

                                }

                                if (temp == NULL) {
                                    puts("DATA yang Anda cari tidak ada!");
                                }
                                }

```

- delete.h

```

#ifndef __DELETE__
#define __DELETE__
#include "abstract_data_type.h"
void removeToDoList(To_Do_List_Node **main_node);
#endif

```

- delete.c

```

#include <stdio.h>
#include <string.h>
#include "delete.h"
#include "view.h"
#include "universal_function.h"
#include "abstract_data_type.h"

void removeToDoList(To_Do_List_Node **main_node) {
    To_Do_List_Node *prev = NULL, *temp = *main_node;
    char will_delete[50];

    viewAll(temp, 0);

    printf("Masukan nama tugas yang ingin dihapus : ");
    scanf("%s", will_delete);
    getchar();

    while(temp != NULL && strcmp(temp->nama_tugas,
will_delete)) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {

```

```

        puts("\n Nama tugas yang anda masukkan tidak
ada");
        return;
    }

    if (prev == NULL) *main_node = (*main_node)->next;
    else prev->next = temp->next;
}

```

- view.h

```

#ifndef __VIEW__
#define __VIEW__
int dayLeft(To_Do_List_Node *temp_main_node);
void viewAll(To_Do_List_Node *main_node, int menu);
void viewSpecific(To_Do_List_Node *main_node, int
menu);
void swapValueLinkedList(To_Do_List_Node **node1,
To_Do_List_Node **node2);
void copyLinkedList(To_Do_List_Node
**copied_main_node, To_Do_List_Node *temp);
void selectionSortList(To_Do_List_Node
**main_node, int menu);
void sortKelompok(To_Do_List_Node *main_node);
void sortPriority(To_Do_List_Node *main_node);
void sortDeadline(To_Do_List_Node *main_node);
#endif

```

- view.c

```

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

int dayLeft(To_Do_List_Node *temp_main_node) {
    int day_left;
    date date_now;
    getTheDate(&date_now);
    day_left = (temp_main_node->dl_dd - date_now.dd) + \
                30 * (temp_main_node->dl_mm -
date_now.mm) + \
                365 * (temp_main_node->dl_yyyy -
date_now.yyyy);
    return day_left;
}

void viewAll(To_Do_List_Node *main_node, int menu) {
    To_Do_List_Node *temp = main_node;
    int nomor = 1, day_left, prioritas = 0, print = 1;

```

```

while (temp != NULL) {
    // Menu 3 (Sort menurut kelompok)
    if (menu == 3 && print == 1) {
        puts("\n=====");
        printf("Kelompok      %s\n",      temp->kelompok_tugas);
        puts("=====\\n");
        nomor = 1;
        print = 0;
    }
    else if (menu == 3 && temp->next != NULL &&
strcmp(temp->kelompok_tugas,      temp->next->kelompok_tugas)) {
        print = 1;
    }
    // Menu 4 (Sort menurut prioritas)
    while(menu == 4 && temp->prioritas !=
prioritas) {
        prioritas++;
        nomor = 1;
        print = 1;
    }
    if (menu == 4 && print == 1) {
        puts("\n=====");
        printf("Prioritas %d\\n", prioritas);
        puts("=====\\n");
        print = 0;
    }
    }

    day_left = dayLeft(temp);
    printf("%d. Nama tugas      : %s\\n", nomor, temp->nama_tugas);
    printf("      Nama kelompok : %s\\n", temp->kelompok_tugas);
    printf("      Prioritas      : %d\\n", temp->prioritas);
    printf("      Deadline      : %02d/%02d/%d\\n",
temp->dl_dd, temp->dl_mm, temp->dl_yyyy);
    if (day_left >= 0) printf("      Sisa Waktu      :
%d hari\\n", day_left);
    else printf("      Sisa Waktu      : TERLAMBAT %d
hari\\n", abs(day_left));
    temp = temp->next;
    nomor++;
}
}

void copyLinkedList(To_Do_List_Node
**copied_main_node, To_Do_List_Node *temp) {
    while (temp != NULL) {

```

```

        *copied_main_node == NULL ? *copied_main_node
= makeNode(*temp) : insert(copied_main_node, *temp);
        temp = temp->next;
    }
}

void viewSpecific(To_Do_List_Node *main_node, int menu)
{
    To_Do_List_Node *temp = main_node;
    To_Do_List_Node *copied_main_node = NULL;
    copyLinkedList(&copied_main_node, temp);
    int day_left, nomor = 1;
    char will_find[50];

    if (menu == 1) {
        printf("Masukkan nama tugas : ");
        scanf("%[^\\n]", will_find);
        getchar();
        while (temp != NULL) {
            if
            (strcmp(lowerTheSentence(copied_main_node-
>nama_tugas), lowerTheSentence(will_find)) == 0) {
                day_left = dayLeft(temp);

                puts("\\n=====");
                printf("%d. Nama tugas      : %s\\n",
nomor, temp->nama_tugas);
                printf("    Nama kelompok : %s\\n", temp-
>kelompok_tugas);
                printf("    Prioritas      : %d\\n", temp-
>prioritas);
                printf("    Deadline       : %d/%d/%d\\n",
temp->dl_dd, temp->dl_mm, temp->dl_yyyy);
                if (day_left >= 0) printf("    Sisa Waktu
: %d hari\\n", day_left);
                else printf("    Sisa Waktu      :
TERLAMBAT %d hari\\n", abs(day_left));

                puts("=====\\n");
                nomor++;
            }
            copied_main_node = copied_main_node->next;
            temp = temp->next;
        }
    }
    else if (menu == 2) {
        printf("Masukkan nama kelompok tugas : ");
        scanf("%[^\\n]", will_find);
        getchar();
        while (temp != NULL) {

```

```

        if
        (strcmp(lowerTheSentence(copied_main_node-
>kelompok_tugas), lowerTheSentence(will_find)) == 0) {
            day_left = dayLeft(temp);
            printf("\n%d. Nama tugas      : %s\n",
nomor, temp->nama_tugas);
            printf("    Nama kelompok : %s\n", temp-
>kelompok_tugas);
            printf("    Prioritas      : %d\n", temp-
>prioritas);
            printf("    Deadline      : %d/%d/%d\n",
temp->dl_dd, temp->dl_mm, temp->dl_yyyy);
            if (day_left >= 0) printf("    Sisa Waktu
: %d hari\n", day_left);
            else printf("        Sisa Waktu      :
TERLAMBAT %d hari\n", abs(day_left));
            nomor++;
        }
        copied_main_node = copied_main_node->next;
        temp = temp->next;
    }
}

if (temp == NULL && nomor == 1) puts("\nTo-do list
yang anda cari tidak ada");
}

void swapValueLinkedList(To_Do_List_Node **node1,
To_Do_List_Node **node2) {
    To_Do_List_Node temp;

    strcpy(temp.nama_tugas, (*node1)->nama_tugas);
    strcpy(temp.kelompok_tugas, (*node1)-
>kelompok_tugas);
    temp.prioritas = (*node1)->prioritas;
    temp.dl_dd = (*node1)->dl_dd;
    temp.dl_mm = (*node1)->dl_mm;
    temp.dl_yyyy = (*node1)->dl_yyyy;

    strcpy((*node1)->nama_tugas, (*node2)-
>nama_tugas);
    strcpy((*node1)->kelompok_tugas, (*node2)-
>kelompok_tugas);
    (*node1)->prioritas = (*node2)->prioritas;
    (*node1)->dl_dd = (*node2)->dl_dd;
    (*node1)->dl_mm = (*node2)->dl_mm;
    (*node1)->dl_yyyy = (*node2)->dl_yyyy;

    strcpy((*node2)->nama_tugas, temp.nama_tugas);
    strcpy((*node2)->kelompok_tugas,
temp.kelompok_tugas);
    (*node2)->prioritas = temp.prioritas;

```

```

        (*node2)->dl_dd = temp.dl_dd;
        (*node2)->dl_mm = temp.dl_mm;
        (*node2)->dl_yyyy = temp.dl_yyyy;
    }

void selectionSortList(To_Do_List_Node **main_node,
int menu) {
    To_Do_List_Node *min;
    for (To_Do_List_Node *current = *main_node ;
current != NULL ; current = current->next) {
        min = current;
        for (To_Do_List_Node *index = current->next ;
index != NULL ; index = index->next) {
            if (menu == 1 && min->prioritas > index-
>prioritas) min = index;
            else if (menu == 2 && dayLeft(min) >
dayLeft(index)) min = index;
        }
        swapValueLinkedList(&min, &current);
    }
}

void sortKelompok(To_Do_List_Node *main_node) {
    int banyak_kelompok = 0;
    To_Do_List_Node *temp = main_node;
    To_Do_List_Node *copied_main_node = NULL;
    copyLinkedList(&copied_main_node, temp);

    for (To_Do_List_Node *current = copied_main_node ;
current != NULL ; current = current->next) {
        for (To_Do_List_Node *index = current->next ;
index != NULL ; index = index->next) {
            if(!strcmp(current->kelompok_tugas, index-
>kelompok_tugas) && current->next != NULL) {
                swapValueLinkedList(&index, &(current-
>next));
                break;
            }
        }
    }

    viewAll(copied_main_node, 3);
}

void sortPriority(To_Do_List_Node *main_node) {
    To_Do_List_Node *temp = main_node;
    To_Do_List_Node *copied_main_node = NULL;

    copyLinkedList(&copied_main_node, temp);
}

```



```

        selectionSortList(&copied_main_node, 1);

        viewAll(copied_main_node, 4);
    }

void sortDeadline(To_Do_List_Node *main_node) {
    To_Do_List_Node *copied_main_node = NULL;
    To_Do_List_Node *temp = main_node;

    copyLinkedList(&copied_main_node, temp);
    selectionSortList(&copied_main_node, 2);

    viewAll(copied_main_node, 0);
}

```

- **universal_function.h**

```

#ifndef __UNIVERSAL_FUNCTION_
#define __UNIVERSAL_FUNCTION_
#include "abstract_data_type.h"
void clear();
void getTheDate(date *date_now);
char *lowerTheSentence(char *sentence);
int isEmpty(To_Do_List_Node *main_node);
void clearTheScreen();
void tahan();
void red();
void reset();
void overwriteFile(To_Do_List_Node **main_node);
#endif

```

- **universal_function.c**

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
#include "universal_function.h"

void clear() {
    #ifdef _WIN32
        system("cls");
    #elif __linux__
        system("clear");
    #endif
}

void getTheDate(date *date_now) {
    time_t time_now;
    time(&time_now);
    struct tm *time_now_converted =
        localtime(&time_now);
}

```

```

        date_now->dd = time_now_converted->tm_mday;
        date_now->mm = time_now_converted->tm_mon + 1;
        date_now->yyyy = time_now_converted->tm_year +
1900;
    }

    char *lowerTheSentence(char *sentence) {
        for (int i = 0 ; sentence[i] ; i++) sentence[i] =
tolower(sentence[i]);
        return sentence;
    }

    int isEmpty(To_Do_List_Node *main_node) {
        if (main_node == NULL) {
            puts("To-do list masih kosong");
            return 1;
        }
        return 0;
    }

    void clearTheScreen() {
        clear();
        puts("Program To-do list\n");
    }

    void tahan() {
        puts("\nTekan enter untuk melanjutkan...");
        getchar();
    }

    void overwriteFile(To_Do_List_Node **main_node) {
        To_Do_List_Node *temp = *main_node;
        FILE *read_and_write =
fopen("file\\to_do_list.txt", "w+");
        while(temp != NULL) {
            fprintf(read_and_write,
"%s|%s|%d|%d/%d/%d\n",temp->nama_tugas, temp-
>kelompok_tugas,\

temp->prioritas, temp->dl_dd,\

temp->dl_mm, temp->dl_yyyy);
            temp = temp->next;
        }
        fclose(read_and_write);
    }

```

4.2. Penjelasan Kode

Program ini merupakan program yang dibuat dengan menyebar fungsi-fungsi yang didefinisikan oleh user ke beberapa file, disebut juga

pemrograman modular. Semua modul yang telah kami definisikan dan deklarasikan tersimpan di folder *lib*.

Pada fungsi *main*, kami memanggil library yang kami gunakan yang semuanya tersimpan di file header yang bernama *our_module.h* yang lokasinya terletak pada folder *lib*.

Pada program ini kamu menggunakan struktur data utama, yaitu *linked list*. Kami memilih *linked list* karena kita tidak akan mengetahui berapa banyak to-do list yang akan dibuat oleh user. Oleh karena itu, kami menggunakan *linked list* dengan menggunakan tipe data yang didefinisikan oleh user yang kami beri nama *To_Do_List_Node* dengan nama variabel utama *main_node*.

Ketika program pertama kali dijalankan, program akan mengecek apakah terdapat file yang bernama *to_do_list.txt*. Apabila file tersebut ada, program akan membaca data-data dari file yang ada di dalamnya dan akan membuat *linked list* yang berdasarkan dengan data dari file tersebut. Proses tersebut. Proses tersebut kami buat ke dalam fungsi yang bernama *initToDoListFromFile()*.

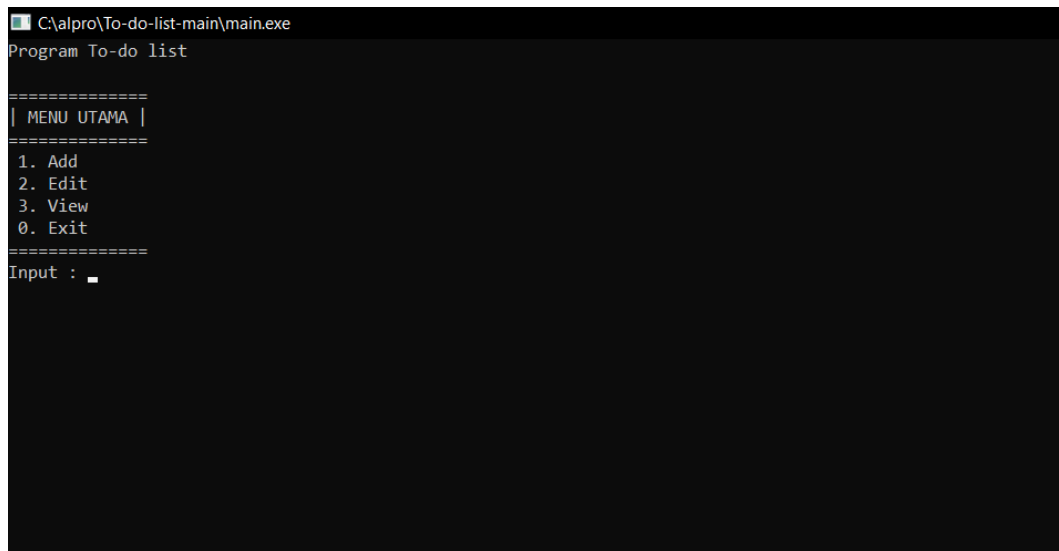
Setelah itu user akan masuk ke dalam menu edit, user dapat memilih proses apa saja yang dapat dilakukan. Ketika proses *add* dipilih, maka program akan menjalankan fungsi *addProcess()* yang deklarasinya terdapat dalam file *create.c*, fungsi ini akan menambahkan node ke dalam *linked list* dan menghubungkannya.

Ketika *edit* dipilih, akan muncul dua pilihan, yaitu *update* dan *remove*. Pilihan *edit* ini merepresentasikan proses *Update* and *Delete* dalam *CRUD*. User dapat mengubah semua properti to-do list dalam menu *update*. User juga dapat menghapus to-do list dengan memilih menu *remove*. Ketika salah satu menu dipilih, program akan menjalankan fungsi yang sudah kami buat secara modular.

Representasi dari operasi *Read* dalam program ini adalah menu *view*. Menu *view* dapat menampilkan to-do list dalam beberapa format yang sudah kami sediakan.

Demikian program CRUD yang menggunakan metode pemrograman modular.

4.3. Screenshot *run* dari program



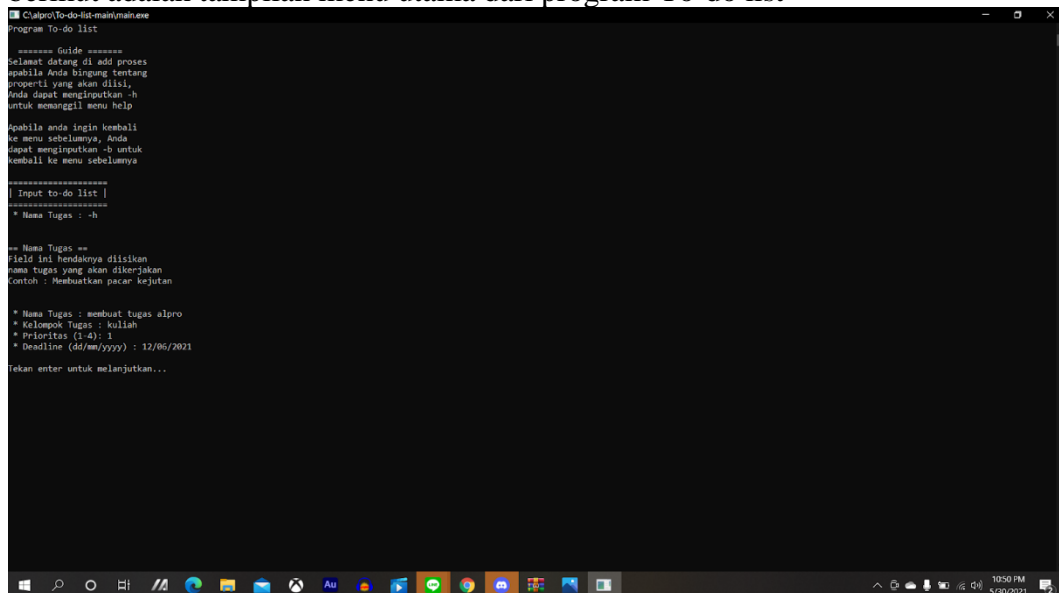
```

C:\alpro\To-do-list-main\main.exe
Program To-do list

=====
| MENU UTAMA |
=====
1. Add
2. Edit
3. View
0. Exit
=====
Input : 

```

berikut adalah tampilan menu utama dari program To-do list



```

C:\alpro\To-do-list-main\main.exe
Program To-do list

===== Guide =====
Selamat datang di add proses
apabila Anda bingung tentang
properti yang akan diisi,
Anda dapat menginputkan -h
untuk memanggil menu help

Apabila anda ingin kembali
ke menu sebelumnya, Anda
dapat menginputkan -b untuk
kembali ke menu sebelumnya

=====
| Input to-do list |
=====
* Nama Tugas : -h

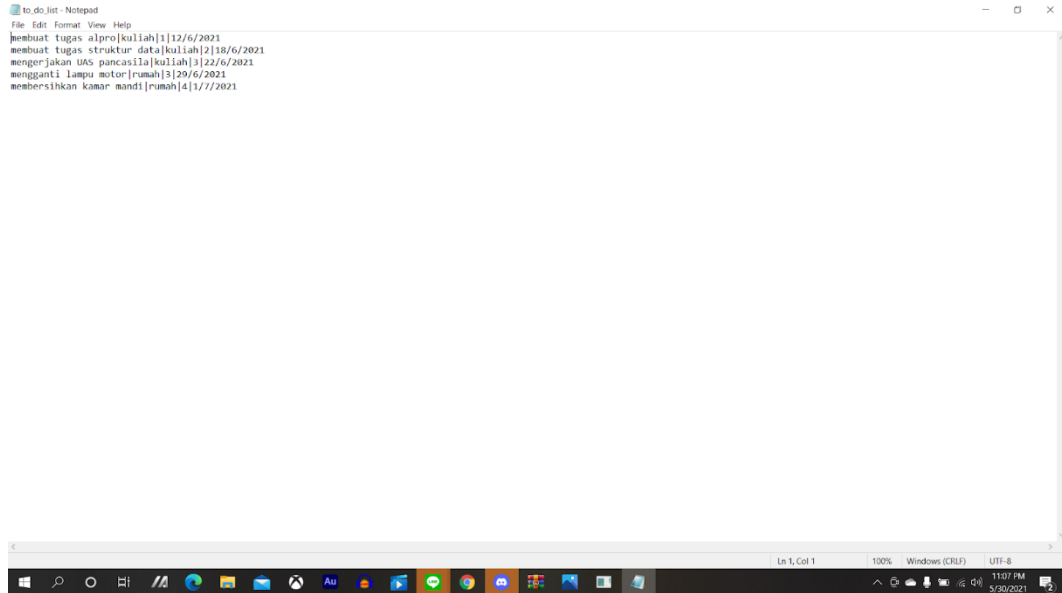
-- Nama Tugas --
Field ini hendaknya diisi
nama tugas yang akan dikerjakan
Contoh - Masukkan pacar kejutan

* Nama Tugas : membuat tugas alpro
* Kelompok Tugas : kuliah
* Prioritas (1-4): 1
* Deadline (dd/mm/yyyy) : 12/06/2021

Tekan enter untuk melanjutkan...

```

Jika kita memilih menu add, kita akan menginputkan data tugas yang ingin dikerjakan. dan jika kesulitan juga kita bisa mendapatkan bantuan dengan menginput -h. dan saya juga telah menambahkan beberapa tugas yang harus dikerjakan di program tersebut. dibawah ini merupakan data-data tugas yang telah diinput.

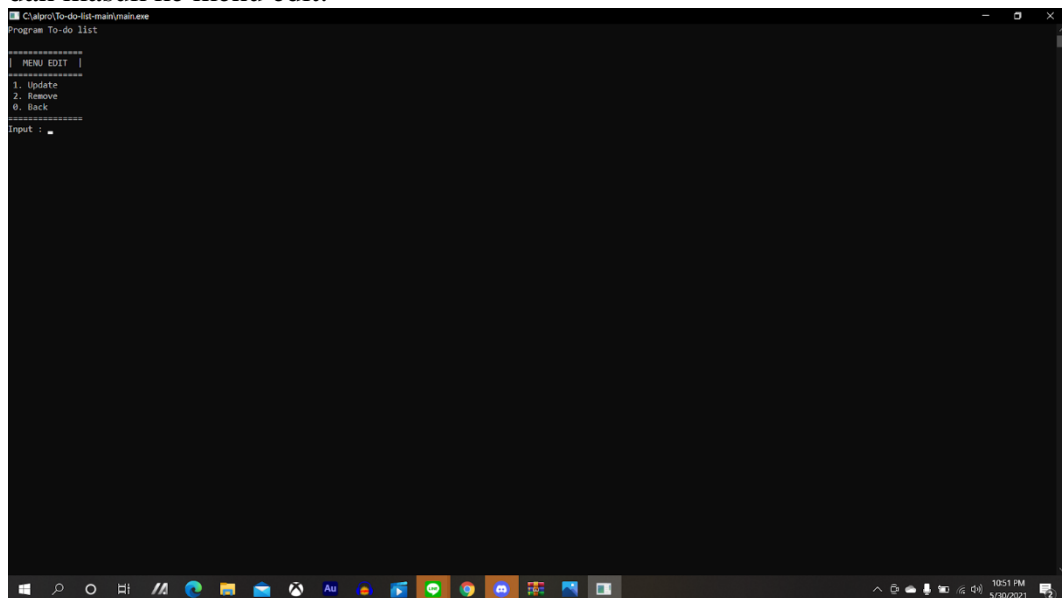


```

File Edit Format View Help
membuat tugas alpro[kuliah1]12/6/2021
membuat tugas struktur data[kuliah]21/8/6/2021
mengerjakan uts pancasila[kuliah]3/22/6/2021
mengganti lampu motor[rumah]3/29/6/2021
membersihkan kamar mandi[rumah]4/1/7/2021

```

kemudian jika kita ingin mengedit data-data tugas kita kembali ke menu utama dan masuk ke menu edit.

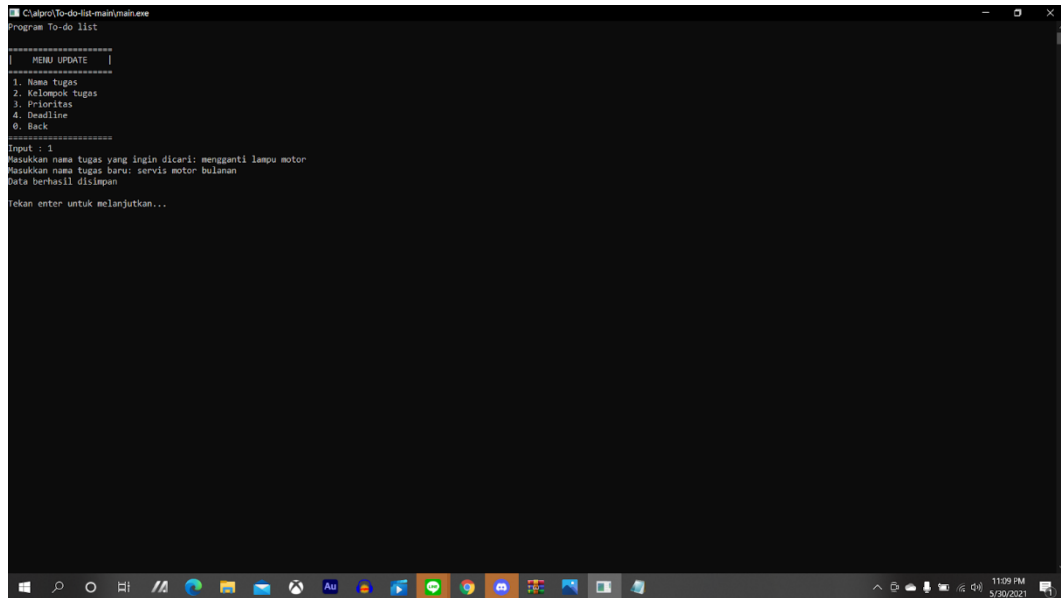


```

C:\proj\To-do-list-main\main.exe
Program To-do list
=====
| MENU EDIT |
=====
1. Update
2. Remove
0. Back
=====
Input : 

```

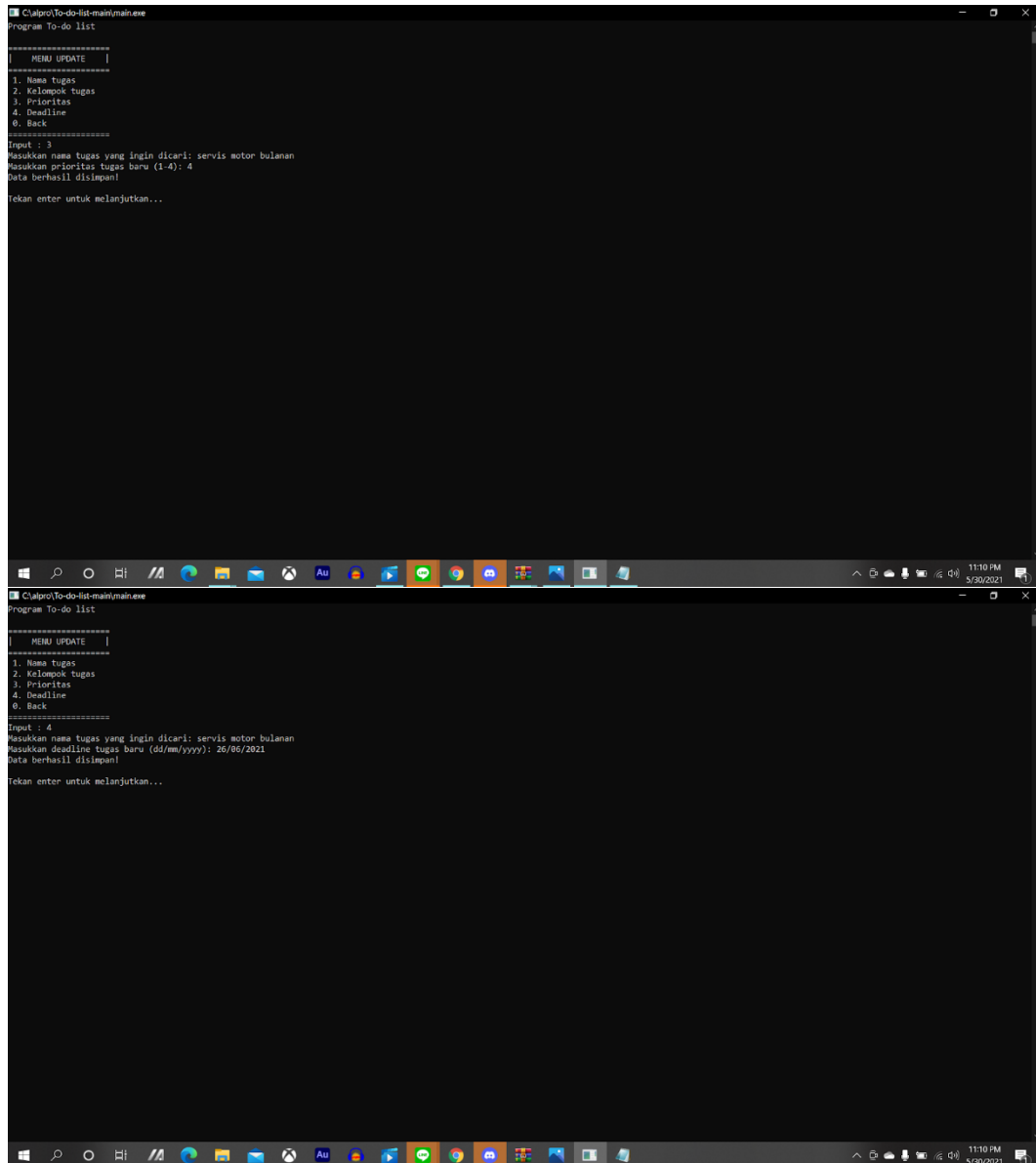
gambar diatas merupakan tampilan dari menu edit. saya ingin mengupdate tugas yang sudah saya tambahkan kemudian masuk ke menu update dan saya ingin mengganti nama tugas.



```
C:\proj\To-do-list-main\main.exe
Program To-do list

=====
| MENU UPDATE |
=====
1. Nama tugas
2. Kelompok tugas
3. Prioritas
4. Deadline
0. Back
=====
Input : 1
Masukkan nama tugas yang ingin dicari: mengganti lampu motor
Masukkan nama tugas baru: servis motor bulanan
Data berhasil disimpan
Tekan enter untuk melanjutkan...
```

disitu saya juga sudah mengupdate beberapa menurut prioritasnya dan deadlinenya.



```
C:\proj\To-do-list-main\main.exe
Program To-do list

=====
| MENU UPDATE |
=====
1. Nama tugas
2. Kelompok tugas
3. Prioritas
4. Deadline
0. Back
=====
Input : 3
Masukkan nama tugas yang ingin dicari: servis motor bulanan
Masukkan prioritas tugas baru (1-4): 4
Data berhasil disimpan!
Tekan enter untuk melanjutkan...

C:\proj\To-do-list-main\main.exe
Program To-do list

=====
| MENU UPDATE |
=====
1. Nama tugas
2. Kelompok tugas
3. Prioritas
4. Deadline
0. Back
=====
Input : 4
Masukkan nama tugas yang ingin dicari: servis motor bulanan
Masukkan deadline tugas baru (dd/mm/yyyy): 26/06/2021
Data berhasil disimpan!
Tekan enter untuk melanjutkan...
```

dan setelah mengupdate tugas saya ingin menghapus tugas yang mungkin bisa tidak jadi saya kerjakan dan saya kembali ke menu edit dan memilih menu remove

```

C:\pro\To-do-list-main\main.exe
Program To-do list

=====
| MENU EDIT |
=====
1. Update
2. Remove
0. Back
=====
Input : 2
1. Nama tugas : membuat tugas alpro
   Nama kelompok : kuliiah
   Prioritas : 1
   Deadline : 12/06/2021
   Sisa Waktu : 12 hari
2. Nama tugas : membuat tugas struktur data
   Nama kelompok : kuliiah
   Prioritas : 2
   Deadline : 18/06/2021
   Sisa Waktu : 18 hari
3. Nama tugas : mengerjakan uas pancasila
   Nama kelompok : kuliiah
   Prioritas : 3
   Deadline : 22/06/2021
   Sisa Waktu : 22 hari
4. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/06/2021
   Sisa Waktu : 26 hari
5. Nama tugas : membersihkan kamar mandi
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 01/07/2021
   Sisa Waktu : 31 hari
Masukan nama tugas yang ingin dihapus :

```

dari gambar diatas diperlihatkan beberapa tugas-tugas dan saya ingin menghapus tugas membersihkan kamar mandi

```

C:\pro\To-do-list-main\main.exe
Program To-do list

=====
| MENU EDIT |
=====
1. Update
2. Remove
0. Back
=====
Input : 2
1. Nama tugas : membuat tugas alpro
   Nama kelompok : kuliiah
   Prioritas : 1
   Deadline : 12/06/2021
   Sisa Waktu : 12 hari
2. Nama tugas : membuat tugas struktur data
   Nama kelompok : kuliiah
   Prioritas : 2
   Deadline : 18/06/2021
   Sisa Waktu : 18 hari
3. Nama tugas : mengerjakan uas pancasila
   Nama kelompok : kuliiah
   Prioritas : 3
   Deadline : 22/06/2021
   Sisa Waktu : 22 hari
4. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/06/2021
   Sisa Waktu : 26 hari
5. Nama tugas : membersihkan kamar mandi
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 01/07/2021
   Sisa Waktu : 31 hari
Masukan nama tugas yang ingin dihapus : membersihkan kamar mandi
Tekan enter untuk melanjutkan...

```

nah jika sudah berarti data tersebut sudah terhapus dan kita kembali lagi ke main menu dan kita ingin mengecek data kita sekali lagi dengan memilih menu view pada main menu dan akan ditampilkan menu seperti dibawah ini


```

C:\alpro\To-do-list-main\main.exe
Program To-do list

=====
|      MENU VIEW      |
=====
1. lihat semua to-do list
2. Cari spesifik
3. Sort menurut kelompok
4. Sort menurut prioritas
5. Sort menurut deadline
6. Back
=====
Input : _

```

untuk mengecek semua tugas kita pilih menu yang pertama yaitu lihat semua to-do list

```

C:\alpro\To-do-list-main\main.exe
Program To-do list

1. Nama tugas : membuat tugas alpro
   Nama kelompok : kuliiah
   Prioritas : 1
   Deadline : 12/06/2021
   Sisa Waktu : 12 hari
2. Nama tugas : membuat tugas struktur data
   Nama kelompok : kuliiah
   Prioritas : 2
   Deadline : 18/06/2021
   Sisa Waktu : 18 hari
3. Nama tugas : mengerjakan uas pancasila
   Nama kelompok : kuliiah
   Prioritas : 3
   Deadline : 22/06/2021
   Sisa Waktu : 22 hari
4. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/06/2021
   Sisa Waktu : 26 hari

Tekan enter untuk melanjutkan...

```

dan jika kita ingin mencari data secara spesifik bisa kita gunakan menu yang ke dua di menu view tersebut yaitu cari spesifik dan akan muncul menu seperti dibawah

ini

```

ClaproTo-do-list-main\main.exe
Program To-do list
=====
| Cari Spesifik |
=====
1. Nama tugas
2. Kelompok tugas
0. Back
=====
Input :

```

kita coba berdasarkan nama tugasnya dengan memilih nomor 1 dan masukan nama tugas dan saya mencari “membuat tugas alpro” dan akan muncul seperti gambar dibawah ini

```

ClaproTo-do-list-main\main.exe
Program To-do list
Masukan nama tugas : membuat tugas alpro
1. Nama tugas : membuat tugas alpro
Nama kelompok : kuliiah
Prioritas : 1
Deadline : 12/6/2021
Sisa Waktu : 12 hari
Tekan enter untuk melanjutkan...

```

sedangkan jika kita hendak mencari berdasarkan kelompoknya kita memilih menu nomor 2 dan masukan kelompok tugas dan saya masukan disini yaitu kelompok ‘rumah’ dan muncul seperti dibawah ini

```

C:\alpro\To-do-list-main\main.exe
Program To-do list

Masukkan nama kelompok tugas : rumah

1. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/6/2021
   Sisa Waktu : 25 hari

Tekan enter untuk melanjutkan...

```

kemudian jika data sudah dicek, lalu kita lanjut ke menu sort data. disini kita bisa menyortir data berdasarkan kelompok, prioritas dan deadlinenya yang dimana menu tersebut bisa kita pilih di menu view yang sebelumnya saya jelaskan dan saya ingin mengurutkan data berdasarkan kelompoknya dan akan muncul seperti gambar dibawah ini

```

C:\alpro\To-do-list-main\main.exe
Program To-do list

Kelompok kuliah
1. Nama tugas : membuat tugas alpro
   Nama kelompok : kuliah
   Prioritas : 1
   Deadline : 12/06/2021
   Sisa Waktu : 12 hari
2. Nama tugas : membuat tugas struktur data
   Nama kelompok : kuliah
   Prioritas : 2
   Deadline : 18/06/2021
   Sisa Waktu : 18 hari
3. Nama tugas : mengerjakan uas pancasila
   Nama kelompok : kuliah
   Prioritas : 3
   Deadline : 22/06/2021
   Sisa Waktu : 22 hari

Kelompok rumah
4. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/06/2021
   Sisa Waktu : 25 hari

Tekan enter untuk melanjutkan...

```

nah data diatas itu sudah disortir berdasarkan kelompok dan jika kita ingin menyortir data berdasarkan menurut prioritas kita pilih menu keempat dan akan muncul tampilan seperti gambar dibawah ini

```

C:\alpro\To-do-list-main\main.exe
Program To-do list

Prioritas 1
1. Nama tugas : membuat tugas alpro
   Nama kelompok : kuliiah
   Prioritas : 1
   Deadline : 12/06/2021
   Sisa Waktu : 12 hari

Prioritas 2
1. Nama tugas : membuat tugas struktur data
   Nama kelompok : kuliiah
   Prioritas : 2
   Deadline : 13/06/2021
   Sisa Waktu : 18 hari

Prioritas 3
1. Nama tugas : mengerjakan uas pancasila
   Nama kelompok : kuliiah
   Prioritas : 3
   Deadline : 22/06/2021
   Sisa Waktu : 22 hari

Prioritas 4
1. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/06/2021
   Sisa Waktu : 26 hari

Tekan enter untuk melanjutkan...

```

dan yang terakhir jika kita sort menurut deadline nya akan muncul tampilan seperti gambar dibawah ini dimana disortir berdasarkan renten harinya.

```

C:\alpro\To-do-list-main\main.exe
Program To-do list

1. Nama tugas : membuat tugas alpro
   Nama kelompok : kuliiah
   Prioritas : 1
   Deadline : 12/06/2021
   Sisa Waktu : 12 hari
2. Nama tugas : membuat tugas struktur data
   Nama kelompok : kuliiah
   Prioritas : 2
   Deadline : 13/06/2021
   Sisa Waktu : 18 hari
3. Nama tugas : mengerjakan uas pancasila
   Nama kelompok : kuliiah
   Prioritas : 3
   Deadline : 22/06/2021
   Sisa Waktu : 22 hari
4. Nama tugas : servis motor bulanan
   Nama kelompok : rumah
   Prioritas : 4
   Deadline : 26/06/2021
   Sisa Waktu : 26 hari

Tekan enter untuk melanjutkan...

```

BAB IV

PENUTUP

4.1. Kesimpulan

Di dalam dunia programmer, operasi file menjadi hal yang sangat penting sekali. Dimana dalam pembuatan sebuah program nantinya, kita sangat perlu akan keberadaan dari data – data dalam melakukan operasi program. Maka dari itu, fungsi file disini dapat digunakan sebagai tempat penampung atau tempat penyimpanan sementara. Selain itu juga digunakan untuk memastikan data di dalam file adalah benar, kemudian meminimalisir atau bahkan menghilangkan potensi kehilangan data.

Disamping itu, masih banyak lagi kegunaan ataupun manfaat lainnya dari operasi file ini, diantaranya adalah : untuk memberikan dukungan berupa masukan (input) dan keluaran (output) kepada banyak pengguna (user) pada sistem multiuser, menyediakan sekumpulan rutin antar muka input dan output, hingga memenuhi kebutuhan dari manajemen data bagi user atau operator komputer. Dapat disimpulkan, bahwa pentingnya pengetahuan dasar ini dimiliki oleh seorang programmer, melihat begitu banyaknya manfaat yang didapat ketika kita berhasil untuk menguasai pengetahuan akan operasi file ini.

Akan tetapi, sebelum masuk ke dalam operasi file, pemahaman dan pengetahuan akan dasar - dasar pemrograman harus lebih dahulu untuk diketahui. Karena, tanpa mengenal dan memahami dasar - dasar pemrograman, kita tidak dapat menggunakan dan mengimplementasikan operasi file tersebut. Di samping itu, struktur data juga sangat penting untuk dikuasai, ketika kita sudah memiliki pondasi atau dasar dalam menggunakan bahasa pemrograman, karena seperti yang kita ketahui bahwa struktur data sangat penting digunakan untuk alokasi memori lebih efisien dan kinerja program yang optimal.

4.2. Saran

Tim Penulis menyadari bahwa makalah ini masih banyak kekurangan. Untuk kedepannya tim penulis akan berusaha untuk membuat makalah dengan lebih baik lagi. Kritik dan saran yang membangun dari para pembaca sangat dibutuhkan penulis untuk dapat memperbaiki kekurangan yang ada dan dapat memberikan inovasi untuk menjadi lebih baik lagi. Saran tim penulis adalah pentingnya konsep operasi file untuk dipahami serta diperdalam lagi, karena penggunaanya begitu banyak di dalam dunia informatika. Agar pekerjaan tertentu dapat diselesaikan lebih sederhana lagi dengan menggunakan operasi file.

DAFTAR PUSTAKA

- Anonim.2012.OperasiFile.<https://pramitananda.wordpress.com/pemrograman-c/operasi-file/> (diakses tanggal 26 mei 2021).
- Ade,Muh Krisna.2018. Daftar Pustaka Menggunakan Operasi File Bahasa Pemrograman C.<https://muhammadadekrisna.wordpress.com/> (diakses tanggal 26 April 2021).
- Ade,Muh Krisna.2018. Jenis-jenis Operasi File Pada Bahasa C. <https://muhammadadekrisna.wordpress.com/2018/10/30/jenis-jenis-operasi-file-pada-bahasa-c/> (diakses tanggal 27 mei 2021).
- Andre.2019. Tutorial Belajar Bahasa C. <https://www.duniailkom.com/tutorial-belajar-c-perulangan-for-bahasa-c/>(diakses tanggal 27 mei 2021).
- Dharma, Abdi. 2017. Aplikasi Pembelajaran Linked List Berbasis Mobile Learning. Riau Journal Of Computer Science, 4(1), 1-11.
- Dianta Ava, Indra. 2019. “Logika dan Algoritma Pemrograman”. Semarang: Sekolah Tinggi Elektronika dan Komputer PAT Semarang.
- Dimas, Setiawan 2020. Contoh Program C IF Bersarang (Nested IF). <https://kelasprogrammer.com/contoh-program-c-if-bersarang-nested-if/>(diakses tanggal 27 mei 2021).
- Jeffrey, Hermanto Array dalam Bahasa C. Tutorial Pemrograman Komputer Sederhana. Published Februari 12, 2008. Dikunjungi maret 17, 2021. <https://tutorialpemrograman.wordpress.com/2008/02/12/array-dalam-bahasa-c/>