

**LAPORAN TUGAS MINI PROJECT PEMROGRAMAN
BERORIENTASI OBJEK
“FIM PROJECT”**

FIM Project GitHub



Dosen Pengampu:

Prof. Dr. Drs. Opim Salim Sitompul M.Sc

Reza Taqyuddin S.Kom

Disusun Oleh :

KELOMPOK 1

R Khairu Wahyutama	(221402051)
Grant Gabriel Tambunan	(221402057)
Jeremy Sharon Tarigan	(221402107)
Khalil Ramzy Nasution	(221402110)

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA MEDAN

TAHUN AJARAN 2022/2023

KATA PENGANTAR

Puji dan syukur senantiasa kita panjatkan kepada Tuhan Yang Maha Esa dengan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan penyusunan laporan dengan judul “FIM Project” dalam rangka pemenuhan tugas mini project mata kuliah Pemrograman Berorientasi Objek.

Tidak lupa penulis mengucapkan rasa terima kasih kepada Prof. Dr. Drs. Opim Salim Sitompul M.Sc dan Reza Taqyuddin S.Kom selaku dosen pengampu dan asisten dosen untuk mata kuliah pemrograman berorientasi objek yang telah membantu penulis dalam mengerjakan laporan ini. Penulis juga mengucapkan terima kasih kepada teman-teman yang telah memberikan masukan dalam pembuatan laporan ini.

Penulis menyadari bahwa laporan ini masih jauh dari kata sempurna dan masih terdapat beberapa kekurangan, oleh karena itu penulis sangat mengharapkan saran dan kritik yang membangun dari pembaca untuk penyempurnaan laporan ini. Semoga laporan mini project ini dapat bermanfaat bagi pembaca dan juga untuk penulis sendiri.

Medan, 09 Juni 2023

Penulis

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI	ii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Tujuan.....	2
1.3 Hasil yang diharapkan	2
BAB II FLOWCHART DAN DIAGRAM ARSITEKTUR PROGRAM	3
2.1 Flowchart Diagram Program	3
2.1.1 Penjelasan Flowchart Diagram Program.....	4
2.2 Diagram Arsitektur Program	12
2.2.1 Penjelasan Diagram Arsitektur Program	12
BAB III TAMPILAN MENU	14
3.1 Tampilan Menu Program	14
BAB IV HASIL IMPLEMENTASI DAN SCREESHOOT TAMPILAN	15
4.1 Fitur Create New File	15
4.2 Fitur Read and Open File	16
4.3 Fitur Edit a Line From File.....	16
4.4 Fitur Delete File.....	20
4.5 Fitur Compile Code (only C++)	21
4.6 Fitur Run File	22
4.7 Tampilan code	23
4.7.1 Main Code	23
4.7.2 Struct Linked List Code	24
4.7.3 Class Code.....	24
BAB V PENUTUP.....	26
5.1 Kesimpulan.....	26
5.2 Saran	26

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era perkembangan teknologi informasi yang pesat, penggunaan text editor telah menjadi bagian penting dalam kegiatan pengembangan perangkat lunak dan penulisan kode program. Text editor adalah salah satu alat yang digunakan untuk mengedit dan mengelola file teks. Namun, dalam industri perangkat lunak, terdapat berbagai macam text editor yang tersedia dengan fitur dan fungsionalitas yang berbeda-beda.

Salah satu text editor populer yang telah dikenal di kalangan pengembang perangkat lunak adalah Vim. Vim adalah text editor yang berada di dalam terminal dan memiliki penggunaan serta konfigurasi yang cukup berbeda dari text editor lainnya. Vim memungkinkan pengguna untuk melakukan pengeditan teks dengan efisiensi tinggi melalui berbagai perintah dan pintasan keyboard yang canggih.

Dalam konteks ini, kami memilih untuk mengembangkan text editor FIM sebagai tugas mini proyek dari mata kuliah pemrograman berorientasi objek. Kami terinspirasi oleh keunikan Vim dan kebutuhan akan text editor yang handal dan mudah digunakan dalam pengembangan perangkat lunak. FIM bertujuan untuk memberikan pengalaman pengeditan teks yang intuitif, efisien, dan fleksibel, dengan menggunakan prinsip-prinsip desain dan konsep pemrograman berorientasi objek.

1.2 Tujuan

Tujuan dari proyek FIM ini adalah:

1. Mengembangkan text editor yang memiliki antarmuka pengguna yang intuitif dan mudah digunakan.
2. Menyediakan fitur-fitur yang mendukung pengeditan teks yang efisien dan produktif.
3. Mengimplementasikan konsep-konsep pemrograman berorientasi objek dalam desain dan pengembangan FIM.
4. Meningkatkan pemahaman dan penerapan konsep pemrograman berorientasi objek dalam pengembangan perangkat lunak.
5. Memenuhi tugas mini project yang diberikan sebagai tugas project akhir mata kuliah “Pemrograman Berorientasi Object”.

1.3 Hasil yang diharapkan

Dengan dikembangkannya FIM, diharapkan akan memberikan manfaat sebagai berikut:

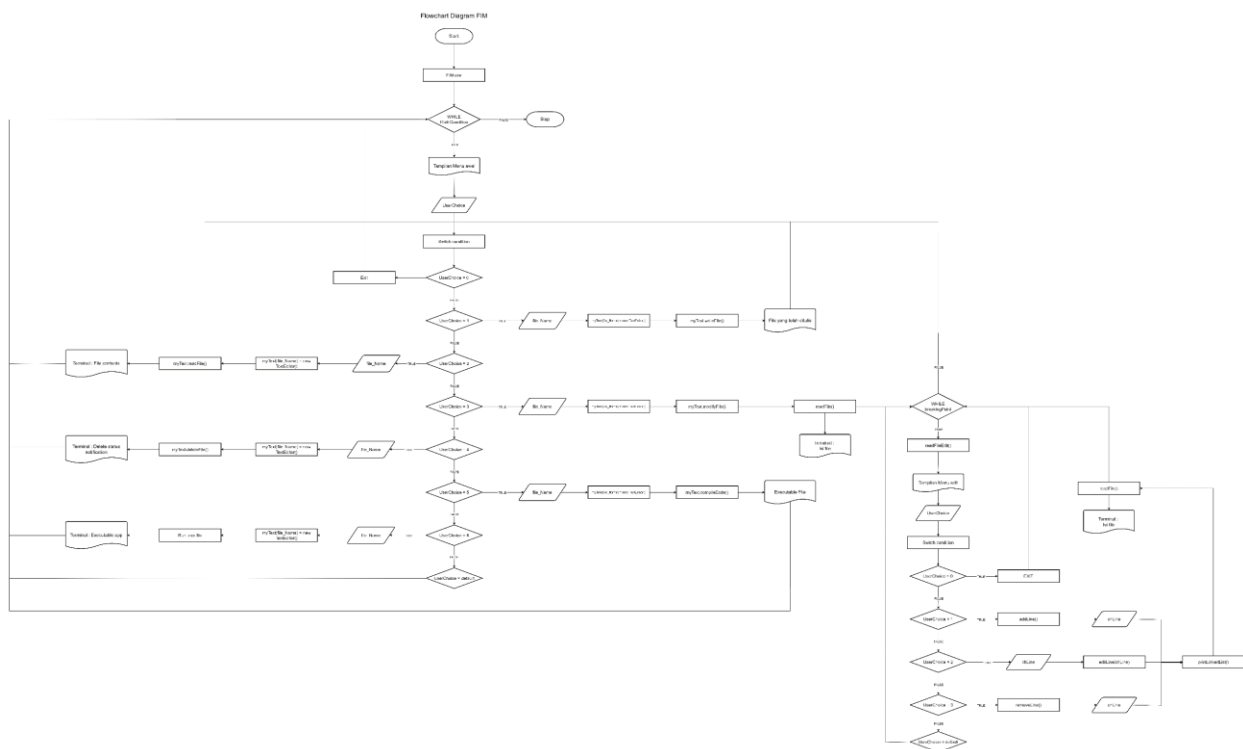
1. Meningkatkan efisiensi dan produktivitas pengguna dalam pengeditan teks dan pengembangan perangkat lunak.
2. Menyediakan pengalaman pengeditan teks yang intuitif dan sesuai dengan kebutuhan pengguna.
3. Memperkaya pemahaman pengembang tentang konsep-konsep pemrograman berorientasi objek.
4. Mendapatkan nilai yang baik dari hasil kerja keras kami.

Dengan demikian, penelitian ini memiliki nilai penting dalam bidang pengembangan perangkat lunak dan konsep pemrograman berorientasi objek. Melalui pengembangan FIM, diharapkan pengguna dapat mengoptimalkan proses pengeditan teks dan meningkatkan efisiensi dalam pengembangan perangkat lunak.

BAB II

FLOWCHART DAN DIAGRAM ARSITEKTUR PROGRAM

2.1 Flowchart Diagram Program

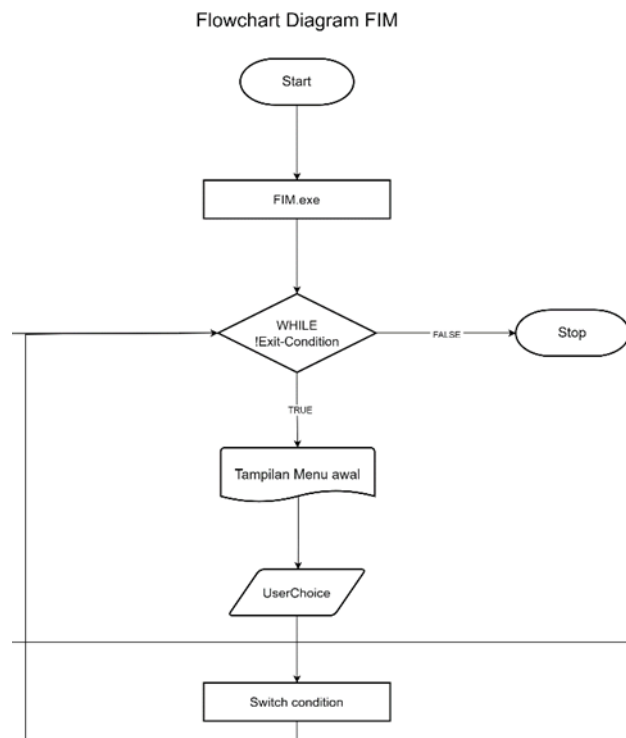


Link Flowchart Diagram Program:

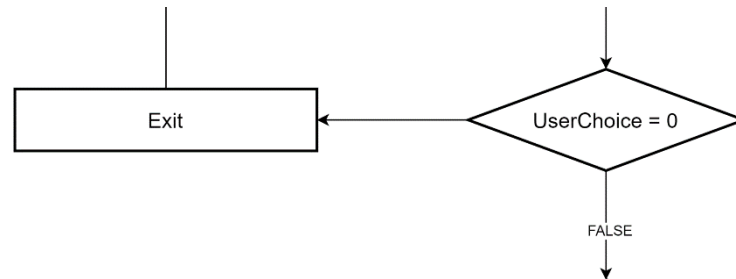
- [FIM-Fixed-Flowchart-png](#)
- [FIM-Fixed-Flowchart.svg](#)

2.1.1 Penjelasan Flowchart Diagram Program

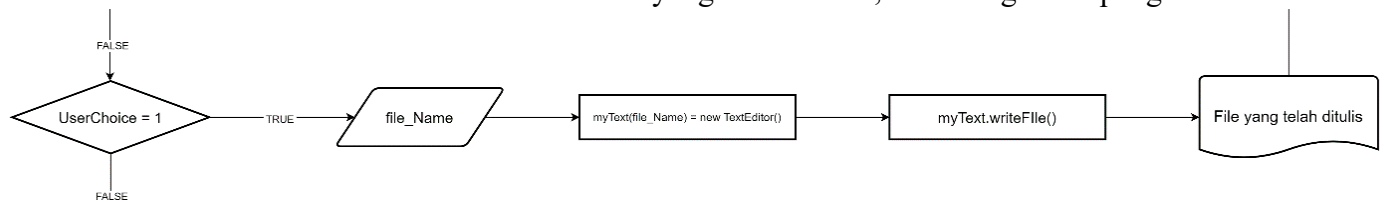
Pertama-tama, permulaan dari menjalankan FIM ialah menjalankan aplikasi executable FIM.exe. Selanjutnya file tersebut akan diarahkan ke perulangan while, yang dimana sebelum terjadi exit-condition, dia akan mengarahkan ke tampilan menu awal dari fungsi publik 'main()'. Selanjutnya user akan diarahkan untuk memasukkan (Input) sebuah angka, yang akan ditentukan arahnya dari switch-case.



Apabila user memilih angka ‘0’, maka program akan mengubah nilai *exit-condition* menjadi sebaliknya, dan akan menjalankan ‘break;’ yang membuat perulangan berhenti. Setelah perulangan berhenti akhirnya fungsi ‘main()’ mencapai exit conditionnya di ‘return 0;’

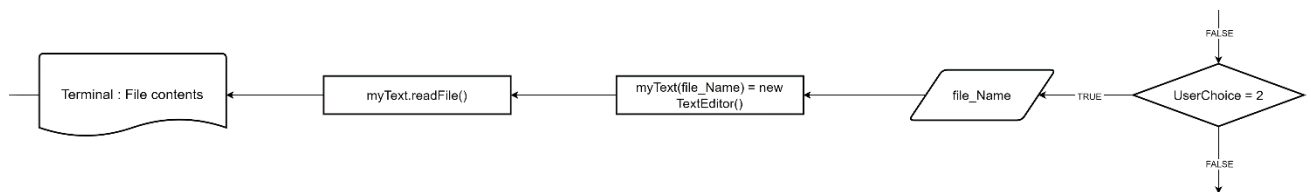


Apabila user memilih angka ‘1’, maka user akan diarahkan untuk memasukkan sebuah inputan berupa *string*, dimana inputan tersebut akan dijadikan sebuah nama file. Inputan tersebut harus memiliki ekstensi dari tipe file yang ingin dibuat. Setelahnya program akan menurunkan object ‘myText’ dari class ‘TextEditor’ dengan membawa argumen berupa nama file yang telah di input. Selanjutnya akan dijalankan fungsi ‘myText.writeFile()’. Lalu user akan diarahkan untuk menulis isi file yang telah dibuat, dan mengakhiri pengisian file

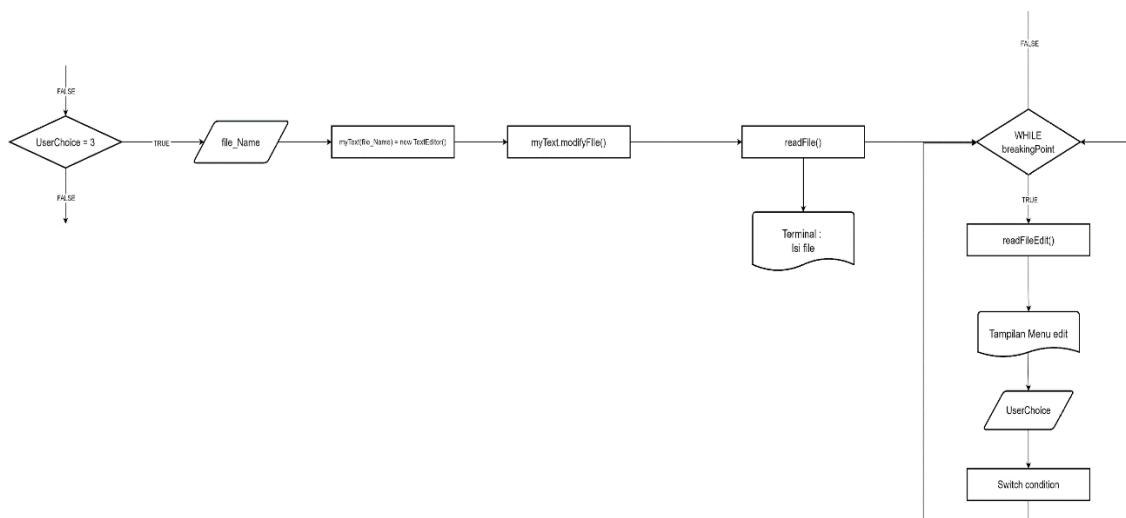


tersebut dengan membuat baris baru dengan kalimat ‘END.’. Lalu, inputan berupa tulisan tersebut akan disimpan dan dimasukkan kedalam file yang telah di inisialisasikan. Terakhir, akan ada ‘break;’ yang akan mengarahkan program kembali ke perulangan dengan nilai *exit-condition* tetap / tidak berubah.

Apabila user memilih angka ‘2’, maka program akan menjalankan fungsi publik ‘askFileName()’ dan user akan diarahkan untuk memasukkan sebuah inputan berupa *string* , dimana inputan tersebut akan dijadikan sebuah nama file. Inputan tersebut harus memiliki ekstensi dari tipe file yang ingin dibuka. Setelahnya program akan menurunkan object ‘myText’ dari class ‘TextEditor’ dengan membawa argumen berupa nama file yang telah di input. Selanjutnya akan dijalankan fungsi ‘myText.readFile()’ yang akan membaca baris-perbaris dari isi file tersebut, dan mengeluarkannya ke *standard output* dengan format khusus. Terakhir, akan ada ‘break;’ yang akan mengarahkan program kembali ke perulangan dengan nilai *exit-condition* tetap / tidak berubah, dan perulangan pun berlanjut.



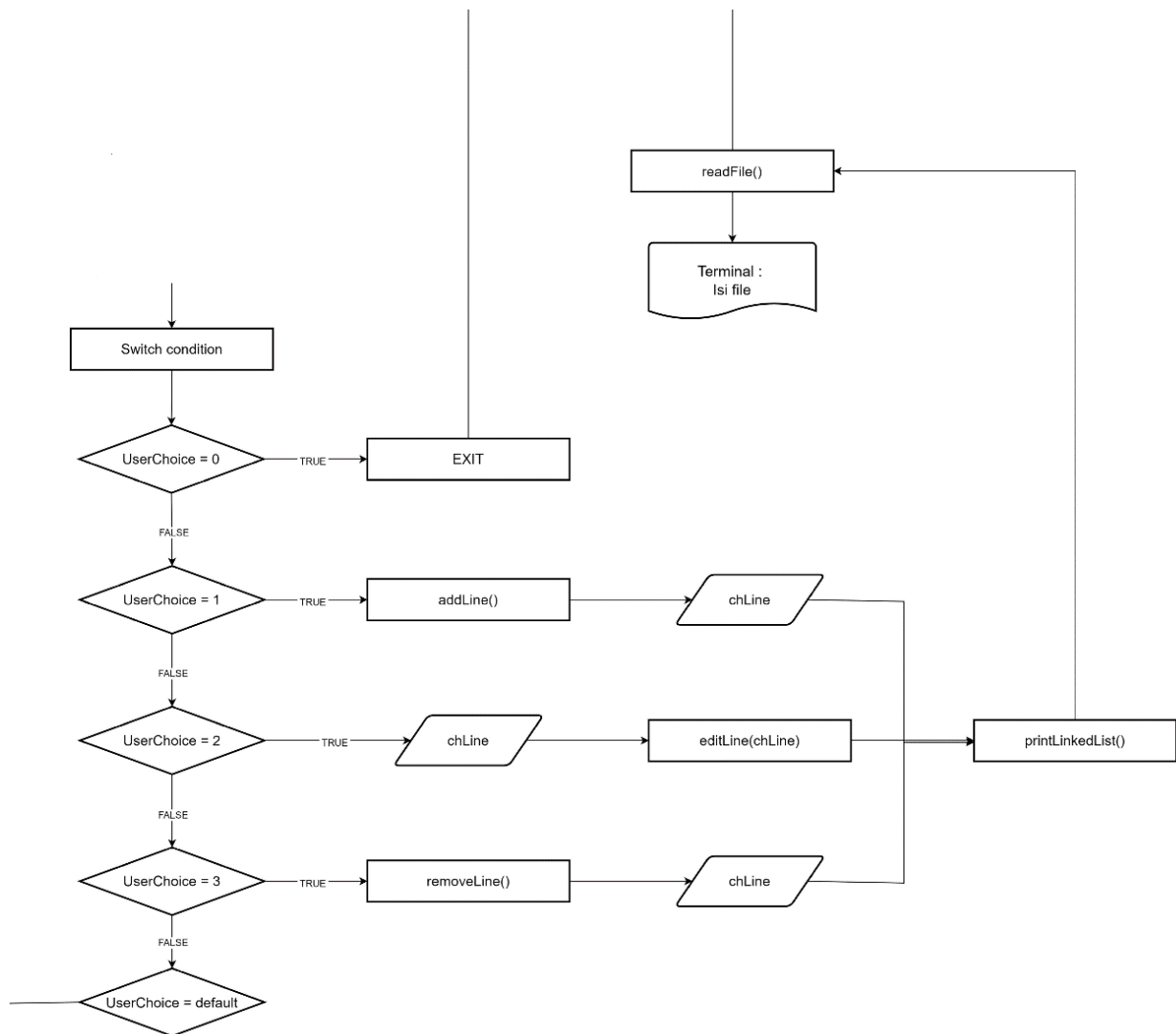
Apabila user memilih angka ‘3’, maka program akan menjalankan fungsi publik ‘askFileName()’ dan user akan diarahkan untuk memasukkan sebuah inputan berupa *string* , dimana inputan tersebut akan dijadikan sebuah nama file. Inputan tersebut harus memiliki ekstensi dari tipe file yang ingin diedit. Setelahnya program akan menurunkan object ‘myText’ dari class ‘TextEditor’ dengan membawa argumen berupa nama file yang telah di input. Selanjutnya akan dijalankan fungsi ‘myText.modifyFile()’ yang akan pertamanya menjalankan fungsi ‘readFile()’ dengan tujuan menampilkan isi file yang ingin di edit ke standard output.



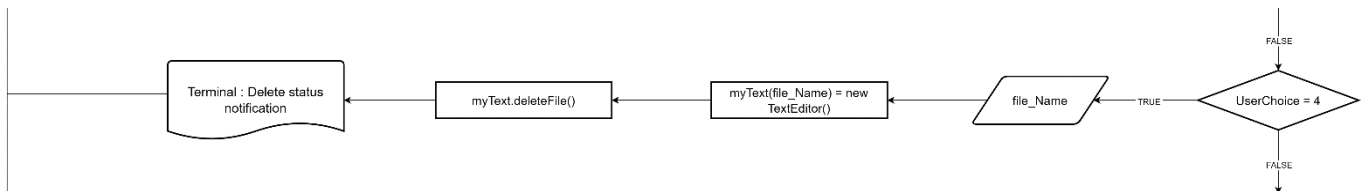
Selanjutnya akan ada perulangan *while* yang dimana apabila belum mencapai 'breakingPoint', akan tetap berulang. Dalam perulangan ini akan dijalankan fungsi 'readFileEdit()' yang akan memindahkan isi file, baris-perbaris ke dalam sebuah linkedlist, dimana setiap baris dari file tersebut adalah sebuah node. Hal ini bertujuan agar memudahkan dalam melakukan operasi edit dalam text editor. Selanjutnya akan dijalankan fungsi 'editMenu()' yang akan menampilkan menu dan meminta inputan dari user. Setelah inputan diterima, akan masuk kedalam *switch-case*.

Didalam *switch-case* untuk menu edit, apabila user memilih angka 0, maka akan mengubah nilai 'breakingPoint' menjadi sebaliknya dan akan membuat perulangan berhenti. Apabila user memilih angka '1' di menu edit, maka akan dijalankan fungsi 'addLine()' yang akan meminta inputan berupa dibaris berapa sebuah text akan ditambahkan. Selanjutnya fungsi itu juga akan meminta sebuah string berupa pesan / isi yang ingin kita tambahkan didalam text. Selanjutnya ia akan menyimpan isi tersebut di linkedlist sesuai dengan node yang terasosiasi. Lalu akan dijalankan fungsi 'printLinkedList()' yang akan mengupdate dengan me-rewrite semua isi file, sesuai dengan isi linkedlist. Perlu diingat bahwa setiap node dalam linkedlist tersebut ialah sebuah baris dalam sebuah file. Lalu isi file akan ditampilkan lagi ke standard output / terminal dengan fungsi 'readFile()', dan akan kembali ke perulangan tanpa mengubah nilai 'breakingPoint', yang berarti perulangan tetap dijalankan. Apabila user memilih angka '2' di menu edit, maka user akan dimintai inputan berupa angka sebagai penanda baris mana yang ingin diedit. Selanjutnya akan dijalankan fungsi 'editLine()' yang akan menerima parameter berupa angka tadi, dan melakukan edit isi linkedlist sesuai dengan node yang ingin diedit. Lalu akan dijalankan fungsi 'printLinkedList()' yang akan mengupdate dengan me-rewrite semua isi file, sesuai dengan isi linkedlist. Lalu isi file akan ditampilkan lagi ke standard output / terminal dengan fungsi 'readFile()', dan akan kembali ke perulangan tanpa mengubah nilai 'breakingPoint', yang berarti perulangan tetap dijalankan. Apabila user memilih angka '3' di menu edit, maka fungsi 'removeLine()' akan dijalankan. Fungsi tersebut akan meminta inputan berupa baris yang akan dihapus, lalu program akan menghapus node yang berisi baris yang dituju.

Lalu akan dijalankan fungsi 'printLinkedList()' yang akan mengupdate dengan me-rewrite semua isi file, sesuai dengan isi linkedlist. Lalu isi file akan ditampilkan lagi ke standard output / terminal dengan fungsi 'readFile()', dan akan kembali ke perulangan tanpa mengubah nilai 'breakingPoint', yang berarti perulangan tetap dijalankan.



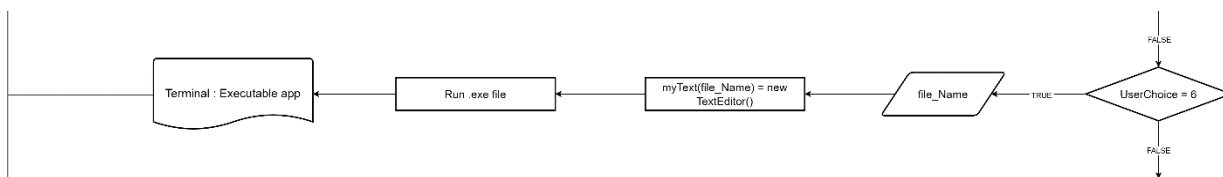
Apabila user memilih angka '4', maka program akan menjalankan fungsi publik 'askFileName()' dan user akan diarahkan untuk memasukkan sebuah inputan berupa *string*, dimana inputan tersebut akan dijadikan sebuah nama file. Inputan tersebut harus memiliki ekstensi dari tipe file yang ingin di hapus. Setelahnya program akan menurunkan object 'myText' dari class 'TextEditor' dengan membawa argumen berupa nama file yang telah di input. Selanjutnya akan dijalankan fungsi 'deleteFile()'. Didalam fungsi tersebut akan dijalan kan fungsi 'remove()' yang akan menerima parameter berisi nama file yang akan didelete. Selanjutnya apabila delete berhasil, program akan mengeluarkan pesan berupa status penghapusan. Jika berhasil maka akan keluar pesan "File '{file_Name}' deleted successfully." dan jika penghapusan gagal akan keluar pesan "Failed to delete file '{file_Name}'.". Terakhir, akan ada 'break;' yang akan mengarahkan program kembali ke perulangan dengan nilai *exit-condition* tetap / tidak berubah, dan perulangan pun berlanjut.



Apabila user memilih angka '5', maka program akan menjalankan fungsi publik 'askFileName()' dan user akan diarahkan untuk memasukkan sebuah inputan berupa *string*, dimana inputan tersebut akan dijadikan sebuah nama file. Inputan tersebut harus memiliki ekstensi dari tipe file yang tepat, yakni file C++ (.cpp). Setelahnya program akan menurunkan object 'myText' dari class 'TextEditor' dengan membawa argumen berupa nama file yang telah di input. Selanjutnya program akan menjalankan fungsi 'myText.compileCode()'. Di dalam fungsi ini, filename tadi akan digunakan dalam mengkonstruksi sebuah perintah yang akan dijalankan ke system. Konstruksi tersebut berupa "g++ -o " + file_Name.substr(0, file_Name.find_last_of('.')) + " " + file_Name;' Sebagai analogi, apabila nama file yang akan kita compile bernama 'Testing.cpp', maka konstruksi tersebut akan menghasilkan *string* 'g++ -o Testing Testing.cpp'. Selanjutnya string itu akan dijalankan dengan fungsi bawaan 'system()' yang akan menjalankan perintah nya. Dengan begitu compiler akan terjalankan dan menghasilkan sebuah *executable file* dengan ekstensi .exe, terakhir, akan ada 'break;' yang akan mengarahkan program kembali ke perulangan dengan nilai *exit-condition* tetap / tidak berubah, dan perulangan pun berlanjut.



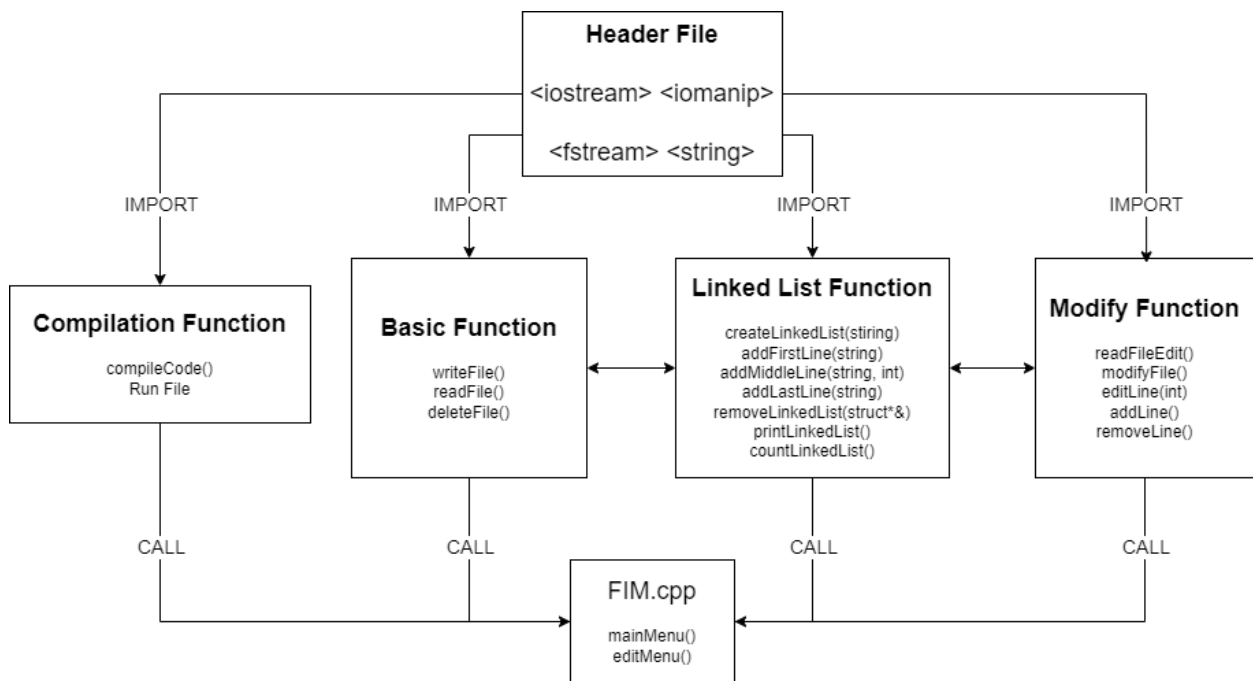
Apabila user memilih angka '6', maka program akan menjalankan fungsi publik 'askFileName()' dan user akan diarahkan untuk memasukkan sebuah inputan berupa *string*, dimana inputan tersebut akan dijadikan sebuah nama file. Inputan tersebut tidak memerlukan ekstensi file, namun perlu digaris-bawahi bahwa file yang hanya bisa dijalankan hanya file yang berekstensi .exe dan juga file .apk, selainnya tidak bisa. Setelahnya program akan menurunkan object 'myText' dari class 'TextEditor' dengan membawa argumen berupa nama file yang telah di input. Selanjutnya program akan menggunakan fungsi bawaan 'system()' dan akan dijalankan lah file tersebut. Sebagai analogi, apabila kita memiliki aplikasi yang dapat dieksekusi bernama 'Testing.exe', maka system akan memasukkan 'Testing' ke CLI dan file akan dieksekusi / dijalankan. Terakhir, akan ada 'break;' yang akan mengarahkan program kembali ke perulangan dengan nilai *exit-condition* tetap / tidak berubah, dan perulangan pun berlanjut.



Apabila user memasukkan selain angka-angka tersebut, fungsi *switch* akan menganggap itu sebagai default dan melakukan perintah 'break;'. Lalu program akan kembali ke perulangan dan berjalan lagi dikarenakan *exit-condition* belum terpenuhi.



2.2 Diagram Arsitektur Program



2.2.1 Penjelasan Diagram Arsitektur Program

a. Header Files:

- `<iostream>`: Digunakan untuk mengakses fungsi-fungsi input/output standar seperti `cout` dan `cin` yang digunakan untuk input dan output data.
- `<string>`: Digunakan untuk mengakses kelas `string` yang menyediakan fitur-fitur dan fungsi-fungsi untuk memanipulasi `string`.
- `<iomanip>`: Digunakan untuk mengatur tampilan output dengan manipulator seperti `setw`, `setprecision`, dan lainnya.
- `<fstream>`: Digunakan untuk mengakses fungsi-fungsi yang berkaitan dengan operasi file, seperti membaca dari file dan menulis ke file.

b. Fungsi Utama (main):

- Fungsi utama `main` merupakan entry point dari program. Di dalam fungsi `main`, alur program utama dijalankan.
- Fungsi `main` dapat menerima argumen baris perintah jika diperlukan, seperti `int main(int argc, char *argv[])`.

- c. Variabel dan Objek:
- Variabel dan objek yang dibutuhkan dalam program dideklarasikan dan diinisialisasi di dalam fungsi main atau di tempat yang sesuai.
 - Misalnya, variabel untuk menyimpan input dari pengguna, variabel yang memegang nilai dari file yang sedang dibaca, dan objek yang digunakan untuk operasi pada string.
- d. Input dan Output:
- Fungsi `cout` digunakan untuk menampilkan pesan atau data ke layar.
 - Fungsi `cin` digunakan untuk menerima input dari pengguna.
 - Fungsi `setw` dari `<iomanip>` digunakan untuk mengatur lebar tampilan output.
 - Fungsi `setprecision` dari `<iomanip>` digunakan untuk mengatur presisi desimal output.
- e. Penggunaan `<fstream>`:
- Objek `ifstream` digunakan untuk membaca dari file. File yang akan dibaca harus dihubungkan dengan objek `ifstream` menggunakan metode `open()` sebelum membaca data dari file tersebut.
 - Objek `ofstream` digunakan untuk menulis ke file. File yang akan ditulis harus dihubungkan dengan objek `ofstream` menggunakan metode `open()` sebelum menulis data ke file tersebut.
 - Metode `close()` digunakan untuk memutuskan koneksi antara objek file dengan file yang dioperasikan setelah selesai membaca atau menulis.
- f. Struktur Program:
- Struktur program dapat menggunakan kontrol alur seperti pengulangan (`for`, `while`) dan percabangan (`if-else`, `switch-case`) untuk mengendalikan alur eksekusi program.
 - Deklarasi dan definisi fungsi-fungsi dapat ditambahkan di atas fungsi main atau menggunakan header file terpisah jika diperlukan.

BAB III

TAMPILAN MENU

3.1 Tampilan Menu Program

```
FIM - Fi Improved
Version 1.0.0

FIM is open source and freely distributed.
Feel free to check our github repos!

1. Create a new file
2. Read and open from an existing file
3. Edit a line from a file
4. Delete file
5. Compile code (C++)
6. Run an app file
0. Exit
Enter your choice . . . █
```

Di dalam Tampilan Menu tersebut mencakup enam fitur yang terbagi menjadi empat fitur utama dan dua fitur tambahan.

Fitur Utama terdiri dari:

- Create New File
- Read and Open File
- Edit Line
- Delete File

Fitur Tambahan terdiri dari:

- Compile Code (only C++)
- Run File File

BAB IV

HASIL IMPLEMENTASI DAN SCREESHOOT TAMPILAN

4.1 Fitur Create New File

Dalam Tampilan Menu proyek FIM, terdapat fitur "Create" yang memungkinkan pengguna untuk membuat file baru dengan ekstensi yang sesuai. Pengguna dapat dengan mudah membuat file baru dengan format yang diinginkan. Perhatikan *Gambar 1.1*.

```
1. Create a new file
2. Read and open from an existing file
3. Edit a line from a file
4. Delete file
5. Compile code (C++)
6. Run an app file
0. Exit
Enter your choice . . .
$ 1
Enter the name file and its extension to create : tes.cpp
```

Gambar 1.1

Sebagai tambahan, penting untuk dicatat bahwa isi file yang sudah diketik harus diakhiri dengan kata kunci "END". Kata kunci ini berfungsi sebagai penanda akhir dari konten yang dimasukkan pengguna ke dalam file. Dengan penanda ini, sistem dapat memastikan bahwa isi file sudah selesai dan siap untuk diproses atau disimpan. Perhatikan *Gambar 1.2*.

```
Write your file, end it with "END."

#include <iostream>

int main() {
    for (int i = 1; i <= 100; i++) {
        if (i % 3 == 0 && i % 5 == 0) {
            std::cout << "FizzBuzz" << std::endl;
        } else if (i % 3 == 0) {
            std::cout << "Fizz" << std::endl;
        } else if (i % 5 == 0) {
            std::cout << "Buzz" << std::endl;
        } else {
            std::cout << i << std::endl;
        }
    }

    return 0;
}
END.
```

Gambar 1.2

4.2 Fitur Read and Open File

Dalam Tampilan Menu proyek FIM, terdapat fitur "Read and Open File" yang memungkinkan pengguna untuk membaca dan membuka file yang telah dibuat sebelumnya. Dengan menggunakan fitur "Read and Open File", pengguna harus menulis nama file beserta ekstensi yang telah dibuat sebelumnya. Perhatikan *Gambar 2.1*.

```
1. Create a new file
2. Read and open from an existing file
3. Edit a line from a file
4. Delete file
5. Compile code (C++)
6. Run an app file
0. Exit
Enter your choice . . .
$ 2
Enter the name file and its extension to open : tes.cpp
```

Gambar 2.1

Setelah di-ENTER akan muncul tampilan sebagai berikut. Perhatikan *Gambar 2.2*.

```
1|
2| #include <iostream>
3|
4| int main() {
5|     for (int i = 1; i <= 100; i++) {
6|         if (i % 3 == 0 && i % 5 == 0) {
7|             std::cout << "FizzBuzz" << std::endl;
8|         } else if (i % 3 == 0) {
9|             std::cout << "Fizz" << std::endl;
10|        } else if (i % 5 == 0) {
11|            std::cout << "Buzz" << std::endl;
12|        } else {
13|            std::cout << i << std::endl;
14|        }
15|    }
16|
17|    return 0;
18| }
Press any key to continue . . .
```

Gambar 2.2

4.3 Fitur Edit a Line From File

Dalam Tampilan Menu proyek FIM, terdapat fitur "Edit Line" yang memberikan kemampuan kepada pengguna untuk melakukan perubahan atau modifikasi pada baris-baris yang terdapat dalam file mereka. Fitur ini memudahkan pengguna dalam mengedit konten file dengan cepat dan efisien. Dengan menggunakan fitur "Edit a Line From File", pengguna harus menulis nama file beserta ekstensi yang ingin diedit. Perhatikan *Gambar 3.1*.

```

1. Create a new file
2. Read and open from an existing file
3. Edit a line from a file
4. Delete file
5. Compile code (C++)
6. Run an app file
0. Exit
Enter your choice . . .
$ 3
Enter the name file and its extension to open : tes.cpp

```

Gambar 3.1

Fitur "Edit a Line From File" memiliki tiga fitur pengeditan utama, yaitu:

- "Add New Line" yang berguna untuk menambahkan baris baru dalam file yang sedang diedit. Fitur ini memudahkan pengguna dalam menyisipkan konten baru pada posisi yang diinginkan dalam file yang sedang mereka edit. Untuk menggunakan fitur ini, pengguna perlu menentukan nomor baris tempat mereka ingin menambahkan konten baru. Setelah menentukan nomor baris, pengguna dapat langsung memasukkan teks atau kode yang ingin ditambahkan pada baris tersebut. Perhatikan *Gambar 3.2.1*.

```

1|
2| #include <iostream>
3|
4| int main() {
5|     for (int i = 1; i <= 100; i++) {
6|         if (i % 3 == 0 && i % 5 == 0) {
7|             std::cout << "FizzBuzz" << std::endl;
8|         } else if (i % 3 == 0) {
9|             std::cout << "Fizz" << std::endl;
10|         } else if (i % 5 == 0) {
11|             std::cout << "Buzz" << std::endl;
12|         } else {
13|             std::cout << i << std::endl;
14|         }
15|     }
16|
17|     return 0;
18| }
Press any key to continue . . .
1. Add new line
2. Edit Line
3. Remove Line
0. Back to menu
Enter your choices . . . 1
Enter which line number you want to add: 3
Insert the new line: using namespace std;

```

Gambar 3.2.1

Setelah di-ENTER akan muncul tampilan sebagai berikut. Perhatikan *Gambar 3.2.2*.

```
1|
2| #include <iostream>
3| using namespace std;
4|
5| int main() {
6|     for (int i = 1; i <= 100; i++) {
7|         if (i % 3 == 0 && i % 5 == 0) {
8|             std::cout << "FizzBuzz" << std::endl;
9|         } else if (i % 3 == 0) {
10|             std::cout << "Fizz" << std::endl;
11|         } else if (i % 5 == 0) {
12|             std::cout << "Buzz" << std::endl;
13|         } else {
14|             std::cout << i << std::endl;
15|         }
16|     }
17|
18|     return 0;
19| }
Press any key to continue . . . █
```

Gambar 3.2.2

- "Edit Line" yang berguna untuk mengedit atau memodifikasi baris dalam file. Fitur ini memberikan kemampuan kepada pengguna untuk mengubah konten dari baris yang telah ada. Untuk menggunakan fitur ini, pengguna perlu menentukan nomor baris yang ingin mereka edit. Setelah menentukan nomor baris, pengguna dapat memasukkan teks atau kode baru yang ingin menggantikan konten yang sudah ada pada baris tersebut. Perhatikan *Gambar 3.3.1*.

```
1|
2| #include <iostream>
3| using namespace std;
4|
5| int main() {
6|     for (int i = 1; i <= 100; i++) {
7|         if (i % 3 == 0 && i % 5 == 0) {
8|             cout << "FizzBuzz" << endl;
9|         } else if (i % 3 == 0) {
10|             std::cout << "Fizz" << std::endl;
11|         } else if (i % 5 == 0) {
12|             std::cout << "Buzz" << std::endl;
13|         } else {
14|             std::cout << i << std::endl;
15|         }
16|     }
17|
18|     return 0;
19| }
Press any key to continue . . .
1. Add new line
2. Edit Line
3. Remove Line
0. Back to menu
Enter your choices . . . 2
Enter which line number you want to edit: 10
Insert the new content: █cout << "Fizz" << endl;
```

Gambar 3.3.1

Setelah di-ENTER akan muncul tampilan sebagai berikut. Perhatikan *Gambar 3.3.2*.

```
1|
2| #include <iostream>
3| using namespace std;
4|
5| int main() {
6|     for (int i = 1; i <= 100; i++) {
7|         if (i % 3 == 0 && i % 5 == 0) {
8|             cout << "FizzBuzz" << endl;
9|         } else if (i % 3 == 0) {
10|             cout << "Fizz" << endl;
11|         } else if (i % 5 == 0) {
12|             std::cout << "Buzz" << std::endl;
13|         } else {
14|             std::cout << i << std::endl;
15|         }
16|     }
17|
18|     return 0;
19| }
Press any key to continue . . .
```

Gambar 3.3.2

- "Remove Line" yang berguna untuk menghapus baris yang tidak diinginkan dalam file. Fitur ini memberikan pengguna kemampuan untuk dengan mudah menghilangkan baris-baris tertentu yang tidak diperlukan dalam proses pengeditan atau pengelolaan file. Untuk menggunakan fitur ini, pengguna hanya perlu memilih nomor baris yang ingin dihapus dari daftar baris yang ada. Dengan menentukan nomor baris tersebut, pengguna dapat secara langsung menghapus baris tersebut dari file. Perhatikan *Gambar 3.4.1*.

```
1|
2| #include <iostream>
3| using namespace std;
4|
5| int main() {
6|     for (int i = 1; i <= 100; i++) {
7|         if (i % 3 == 0 && i % 5 == 0) {
8|             cout << "Fizz Buzz" << endl;
9|         } else if (i % 3 == 0) {
10|             cout << "Fizz" << endl;
11|         } else if (i % 5 == 0) {
12|             cout << "Buzz" << endl;
13|         } else {
14|             cout << i << endl;
15|         }
16|     }
17|
18|     return 0;
19| }
Press any key to continue . . .
1. Add new line
2. Edit Line
3. Remove Line
0. Back to menu
Enter your choices . . . 3
Enter the line number you want to remove: 1

Line successfully removed.
Press any key to continue . . .
```

Gambar 3.4.1

Setelah di-ENTER akan muncul tampilan berikut. Perhatikan *Gambar 3.4.2*.

```
1| #include <iostream>
2| using namespace std;
3|
4| int main() {
5|     for (int i = 1; i <= 100; i++) {
6|         if (i % 3 == 0 && i % 5 == 0) {
7|             cout << "Fizz Buzz" << endl;
8|         } else if (i % 3 == 0) {
9|             cout << "Fizz" << endl;
10|        } else if (i % 5 == 0) {
11|            cout << "Buzz" << endl;
12|        } else {
13|            cout << i << endl;
14|        }
15|    }
16|
17|    return 0;
18| }
```

Press any key to continue . . .

Gambar 3.4.2

Dengan adanya fitur pengeditan ini, pengguna memiliki kontrol penuh atas isi file mereka. Mereka dapat menyesuaikan dan mengubah konten file dengan mudah sesuai dengan kebutuhan proyek atau tugas yang sedang dijalankan.

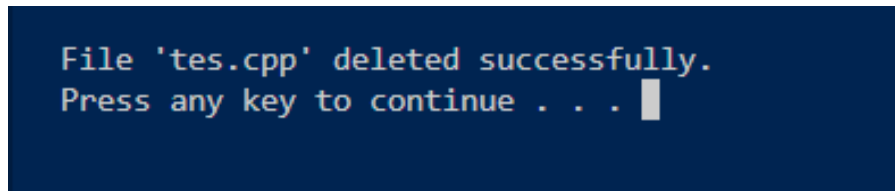
4.4 Fitur Delete File

Dalam Tampilan Menu proyek FIM, terdapat fitur "Delete File" yang memberikan kemampuan kepada pengguna untuk menghapus file yang tidak diperlukan lagi. Fitur ini memungkinkan pengguna untuk melakukan pembersihan dan penghapusan file yang sudah tidak relevan atau tidak dibutuhkan dalam proyek mereka. Dengan menggunakan fitur "Delete File", pengguna harus menulis nama file beserta ekstensi yang ingin di hapus. Perhatikan *Gambar 4.1*.

```
1. Create a new file
2. Read and open from an existing file
3. Edit a line from a file
4. Delete file
5. Compile code (C++)
6. Run an app file
0. Exit
Enter your choice . . .
$ 4
Enter the name file and its extension to delete : tes.cpp
```

Gambar 4.1

Setelah di-ENTER akan muncul tampilan berikut. Perhatikan *Gambar 4.2*.

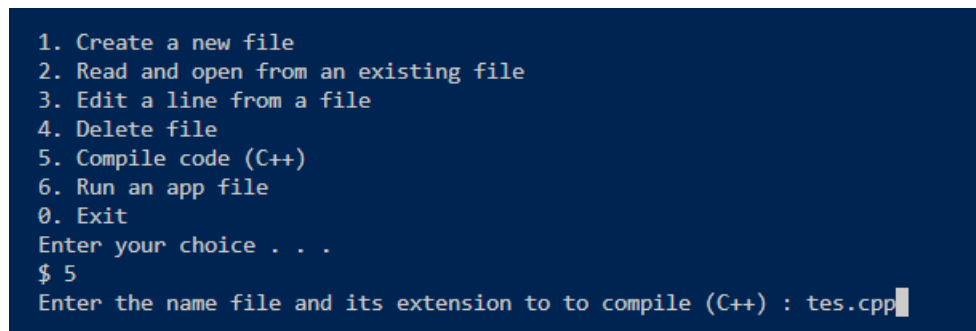
A screenshot of a terminal window with a dark blue background. The text is white and reads: "File 'tes.cpp' deleted successfully. Press any key to continue . . .". A white cursor is visible at the end of the second line.

```
File 'tes.cpp' deleted successfully.  
Press any key to continue . . .
```

Gambar 4.2

4.5 Fitur Compile Code (only C++)

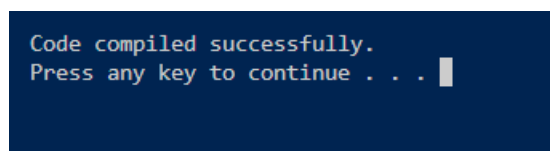
Dalam Tampilan Menu proyek FIM, terdapat fitur "Compile Code (only c++)" yang secara khusus dirancang untuk file dengan ekstensi (.cpp). Fitur ini memberikan kemampuan kepada pengguna untuk mengompilasi file kode program dalam bahasa C++ menjadi file objek yang dapat dieksekusi. Perhatikan *Gambar 5.1*.

A screenshot of a terminal window with a dark blue background. The text is white and shows a menu with options 1 through 6 and 0. The user has entered '5' as their choice. The prompt asks for the file name and extension to compile, with 'tes.cpp' entered.

```
1. Create a new file  
2. Read and open from an existing file  
3. Edit a line from a file  
4. Delete file  
5. Compile code (C++)  
6. Run an app file  
0. Exit  
Enter your choice . . .  
$ 5  
Enter the name file and its extension to to compile (C++) : tes.cpp
```

Gambar 5.1

Setelah di-ENTER akan muncul tampilan berikut. Perhatikan *Gambar 5.2*.

A screenshot of a terminal window with a dark blue background. The text is white and reads: "Code compiled successfully. Press any key to continue . . .". A white cursor is visible at the end of the second line.

```
Code compiled successfully.  
Press any key to continue . . .
```

Gambar 5.2

Maka proses Compile berhasil.

4.6 Fitur Run File

Dalam Tampilan Menu proyek FIM, terdapat fitur "Run File" yang memungkinkan pengguna untuk menjalankan berkas yang telah di-compile menjadi file eksekusi (.exe). Fitur ini memberikan pengguna kemampuan untuk secara langsung menjalankan program atau aplikasi yang telah selesai dikembangkan. Dengan menggunakan fitur "Run File", pengguna dapat memilih file eksekusi (.exe) yang ingin dijalankan dari daftar berkas yang tersedia. Setelah dipilih, program akan dijalankan dan pengguna dapat melihat hasil dari program tersebut. Perhatikan *Gambar 6.1*.

```
1. Create a new file
2. Read and open from an existing file
3. Edit a line from a file
4. Delete file
5. Compile code (C++)
6. Run an app file
0. Exit
Enter your choice . . .
$ 6
Enter the app file name to run : tes.exe
```

Gambar 6.1

Setelah di-ENTER akan muncul Output dari program Fizz Buzz C++ berikut. Perhatikan *Gambar 6.2*.

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
Fizz Buzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
Fizz Buzz
31
32
Fizz
34
Buzz
Fizz
```

Gambar 6.2

4.7 Tampilan code

4.7.1 Main Code

```
int main() {
    bool breakingPoint = true;

    while(breakingPoint) {
        switch(menu()) {
            case 1: {
                string file_Name = askFileName("create");
                TextEditor myText(file_Name);
                myText.writeFile();
                break;
            }

            case 2: {
                string file_Name = askFileName("open");
                TextEditor myText(file_Name);
                myText.readFile();
                break;
            }

            case 3: {
                string file_Name = askFileName("open");
                TextEditor myText(file_Name);
                myText.modifyFile();
                break;
            }

            case 4: {
                string file_Name = askFileName("delete");
                TextEditor myText(file_Name);
                myText.deleteFile();
                system("PAUSE");
                break;
            }

            case 5: {
                string file_Name = askFileName("to compile (C++)");
                TextEditor myText(file_Name);
                myText.compileCode();
                break;
            }

            case 6: {
                string file_Name;
                cout << "Enter the app file name to run : ";
                cin.ignore();
                getline(cin, file_Name);
                system("CLS");
                system(file_Name.c_str()); // Basically run the program into the
system.
```

```

        cout << endl;
        system("PAUSE");
        break;
    }

    case 0:
        breakingPoint = false; // Exit condition.
        break;

    default:
        break;
}

}

return 0;
}

```

4.7.2 Struct Linked List Code

```

// {{ Linklist For Edit Start }}

struct Filenote {
    string line;
    Filenote* next;
};

Filenote *head = NULL, *tail = NULL, *cur, *newNode, *nextNode, *del;

// {{ Linklist For Edit End }}

```

4.7.3 Class Code

```

class TextEditor {
private:
    string file_Name;
    ofstream ofile;
    ifstream ifile;

public:
    TextEditor(string file_Name = "file.txt") {this->file_Name = file_Name;} //
Constructor
    // Default function (Create and read)
    void writeFile(); // Method for writing a
file. [[-> Khalil Ramzy]]
    bool readFile(); // Method for read a
file. [[-> Jeremy Sharon]]

    // Edit function /w linkedlists
    void createLineLinkedList(string); // Make the first node of linked
list. [[-> R. Khairu]]

```

```

        void addFirstLine(string);           // Add the node to the first linked
list, used for add line.                    [[-> R. Khairu]]
        void addMiddleLine(string, int);     // Add the node at the position by
the parameter.                             [[-> R. Khairu]]
        void addLastLine(string);           // Add the node at the last
position of linked list.                   [[-> R. Khairu]]
        void removeLinkedList(Filenote*&);   // Removing all node from linked
list.                                       [[-> R. Khairu]]
        void printLinkedList();              // Print linked list to
file.                                     [[-> R. Khairu]]
        int countLinkedList();               // Count node in linked
list.                                     [[-> R. Khairu]]

        // Modify (Edit and Delete) and run it.
        Void readFileEdit();                // Read a file to add into
linkedlist.                               [[-> Grant Gabriel]]
        void modifyFile();                  // A controller method for
modifying file.                           [[-> Grant Gabriel]]
        void editLine(int);                 // Edit line at position passed by
parameter.                               [[-> Grant Gabriel]]
        void addLine();                     // Menu controller for adding a
line.                                    [[-> Grant Gabriel]]
        void removeLine();                  // Removing a
line.                                    [[-> Grant Gabriel]]
        void deleteFile();                  // Method for deleting a
file.                                    [[-> Khalil Ramzy]]
        void compileCode();                 // Method for compiling a C++
file.                                    [[-> Jeremy Sharon]]
        ~TextEditor(){}; // Destructors
};

```

Untuk penjelasan code yang lebih detail, bisa dilihat di link GitHub berikut :

[FIM-Project-Github](#)

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan dari project FIM ini ialah sebagai berikut :

- Pengguna dapat menggunakan aplikasi FIM, dengan membaca dokumentasi lebih dalam mengenai proyek ini.
- Pengguna dapat menggunakan FIM, text editor berbasis CLI untuk membuat sebuah file dengan ekstensi tersendiri.
- Pengguna dapat menggunakan FIM untuk menulis sebuah konten / isi kedalam sebuah file, dan juga membaca isi file tersebut langsung dari terminal.
- Pengguna dapat menggunakan aplikasi FIM untuk melakukan edit dalam sebuah file tertentu langsung di terminal, yang dimana pengguna dapat melakukan perubahan, maupun menghapus sebuah baris langsung dari terminal.
- Pengguna dapat menggunakan aplikasi FIM untuk melakukan perintah compile sebuah program C++ langsung dari terminal.
- Pengguna dapat menggunakan aplikasi FIM untuk menjalankan sebuah aplikasi yang telah di compile maupun aplikasi yang telah tersedia.
- Aplikasi FIM telah memenuhi persyaratan dan memenuhi ketentuan tugas yang diberikan dalam mata kuliah “Pemrograman Berorientasi Objek”.

5.2 Saran

Saran untuk project FIM ini adalah sebagai berikut :

- Developer butuh mengatasi penggunaan memori berlebihan dari aplikasi FIM yang disebabkan oleh ketidakefisienan aplikasi.
- Developer butuh memberikan desain interaksi yang lebih menarik dan juga sepadan untuk pengguna nantinya.
- User harus mengusahakan memahami dokumentasi secara mendalam agar terbiasa dalam penggunaan aplikasi FIM ini.