

LAPORAN MINI PROJECT
PEMROGRAMAN BERBASIS OBJEK
“TEXT EDITOR”



Dosen Pengampu : Prof. Dr. Drs. Opim Salim Sitompul, M.Sc
NIP : 196108171987011001

Disusun Oleh :

Kelompok 3 :

Muhammad Nayaka Putra	(221402001)
Ufy Ananda Yatna	(221402060)
Fildza Rasyika	(221402101)
Ruth Limike Puteri Sihaloho	(221402128)

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN 2023

KATA PENGANTAR

Puji syukur kehadiran Tuhan Yang Maha Esa atas segala rahmatnya sehingga laporan mini project yang berjudul ‘Text Editor’ ini dapat tersusun hingga selesai. Tidak lupa juga kami mengucapkan terima kasih Prof. Dr. Drs. Opim Salim Sitompul, M.Sc. selaku dosen pengampu mata kuliah Pemrograman Berbasis Objek yang telah membimbing dan memberi sumbangan pengetahuannya kepada kami sehingga dapat menyelesaikan laporan ini secara tepat waktu.

Kami menyadari sepenuhnya bahwa masih ada kekurangan dalam laporan dan mini project kami, baik dari segi penyusunan kalimat maupun tata bahasanya. Oleh karena itu, dengan tangan terbuka kami menerima segala bentuk saran maupun kritik dari pembaca agar kami dapat memperbaiki laporan mini project ‘Text Editor’ ini.

Akhir kata, kami berharap semoga laporan yang telah kami buat dapat bermanfaat dalam meningkatkan pengetahuan dan pemahaman pembaca terkait penerapan Pemrograman Berbasis Objek.

Medan, 10 Juni 2023

Kelompok 3

DAFTAR ISI

HALAMAN SAMPUL	i
KATA PENGANTAR	ii
DAFTAR ISI	iii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	1
BAB II PEMBAHASAN	2
2.1 Arsitektur Program	2
2.2 Flowchart	5
2.3 Desain	6
2.4 Hasil Implementasi	15
BAB III PENUTUP	41
3.1 Kesimpulan	41
3.2 Saran	41
DAFTAR PUSTAKA	42

BAB I

PENDAHULUAN

1.1 Latar Belakang

Text editor adalah jenis perangkat lunak yang sangat penting di dunia komputasi modern. Text editor digunakan oleh pemrogram dan pengembang untuk memanipulasi kode sumber teks, mengedit file konfigurasi atau dokumentasi. Text editor juga memungkinkan pengguna untuk membuat, mengedit, dan mengelola berbagai dokumen teks seperti file kode sumber, catatan, atau dokumen lainnya. Text editor memudahkan pengguna dalam membuat program yang mudah dipahami. Beberapa contoh text editor yang populer adalah Visual Studio Code (VSCode), Sublime Text, dan Notepad++.

Dalam pengembangan perangkat lunak, pendekatan pemrograman berorientasi objek (OOP) telah menjadi pendekatan yang populer dan efektif. OOP memungkinkan pengembang untuk memodelkan dunia nyata dalam struktur yang terorganisir dan modular. Saat membuat program text editor, OOP C++ bisa menjadi pilihan yang baik karena C++ adalah bahasa pemrograman yang kuat dan efisien dengan dukungan yang baik untuk OOP.

1.2 Rumusan Masalah

Adapun rumusan masalah dari laporan ini adalah “bagaimana membuat program text editor menggunakan Pemrograman Berorientasi Objek (OOP) C++ yang handal, efisien, dan memiliki berbagai fitur?”

1.3 Tujuan

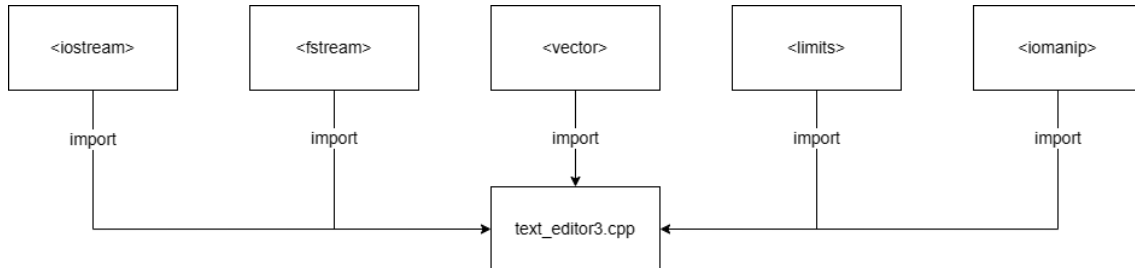
Adapun tujuan dari penulisan laporan ini adalah untuk mengetahui bagaimana pembuatan program text editor menggunakan OOP C++ yang menghasilkan program yang handal, efisien, dan memiliki berbagai fitur.

BAB II

PEMBAHASAN

2.1 Arsitektur Program

Berikut adalah beberapa diagram arsitektur program text editor:



Program text editor ini memiliki 5 library yaitu :

1. <iostream>

Library iostream adalah library standar C++ yang berisi fungsi-fungsi untuk operasi input-output. Fungsi-fungsi penting dalam library ini yang digunakan dalam program text editor antara lain:

- ``cout`` dan ``<<`` : Digunakan untuk menampilkan output ke layar.
- ``cin`` dan ``>>`` : Digunakan untuk menerima input pengguna melalui keyboard.
- ``endl`` : Digunakan untuk mencetak baris baru

2. <fstream>

Library fstream menyediakan fungsi-fungsi yang diperlukan untuk operasi pada file, seperti membuka, menutup, membaca, dan menulis file. Beberapa fungsi penting dari library ini yang digunakan dalam program text editor adalah:

- ``ifstream`` : Objek untuk membaca dari file.
- ``ofstream`` : Objek untuk menulis ke dalam file.
- ``fstream`` : Objek untuk membaca dan menulis file.
- ``open()`` : Digunakan untuk membuka file.
- ``close()`` : Digunakan untuk menutup file.

3. <vector>

Library vector menyediakan struktur data vektor (dynamic array) yang dapat digunakan untuk menyimpan dan mengelola kumpulan data. Beberapa fungsi dan fitur penting dari library ini yang digunakan dalam program text editor adalah:

- a. ``vector`` : Struktur data vektor yang dapat diperluas secara dinamis.
- b. ``push_back()`` : Menambahkan elemen baru di bagian belakang vektor.
- c. ``size()`` : Mengembalikan jumlah elemen dalam vektor.
- d. ``clear()`` : Menghapus semua elemen dalam vektor.
- e. ``erase()`` : Menghapus elemen pada posisi tertentu dalam vektor.

4. `<limits>`

Library `limits` berisi berbagai konstanta dan fungsi untuk memperoleh batasan-batasan nilai dalam bahasa C++. Beberapa contoh penggunaan library ini yang digunakan dalam program text editor adalah:

- a. ``numeric_limits<T>::max()`` : Mengembalikan nilai maksimum yang dapat dinyatakan oleh tipe data T.
- b. ``numeric_limits<T>::min()`` : Mengembalikan nilai minimum yang dapat dinyatakan oleh tipe data T.

5. `<iomanip>`

Library `iomanip` menyediakan fungsi-fungsi untuk melakukan manipulasi format pada output, seperti pengaturan lebar kolom, presisi desimal, dan lainnya. Salah satu fungsi penting dalam library ini yang digunakan dalam program text editor adalah:

- a. ``setw()`` : Mengatur lebar kolom pada output.

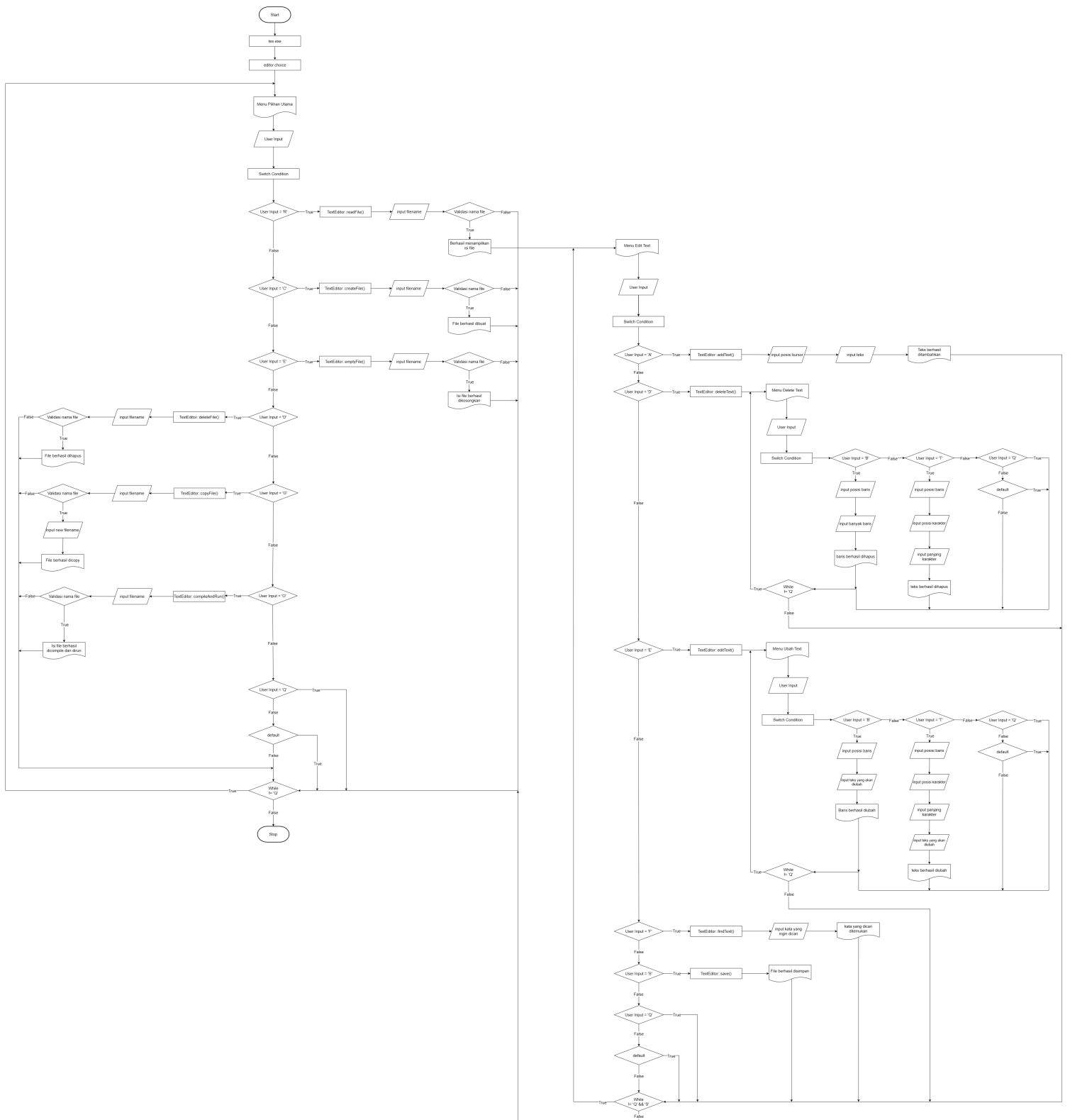
TextEditor
- fileName : string - lines : vector<string> - cursorPosition : int
+ TextEditor() + createFile() : void + emptyFile() : void + deleteFile() : void + copyFile() : void + compileAndRun() : void + readFile() : void + choice() : void + addText() : void + deleteText() : void + editText() : void + findText() : void + save() : void + edit () : void + ~TextEditor()

Dalam arsitektur program ini juga terdapat diagram kelas yang menggambarkan struktur class dalam teks editor. Berikut penjelasan dari diagram kelas ini :

- Kelas **`TextEditor`** merupakan kelas utama yang memiliki atribut atau variable dan metode atau fungsi.
- Atribut atau variabel **`fileName`** bertipe data string untuk menampung nama file yang diinput.
- Atribut atau variabel **`lines`** bertipe vector string untuk menampung teks yang dibaca atau diubah.
- Atribut atau variabel **`cursorPosition`** bertipe data integer untuk menampung angka yang menunjuk pada baris dalam teks.
- Metode atau fungsi **`TextEditor()`** merupakan constructor untuk menginisialisasi objek **`TextEditor`**.
- Metode atau fungsi **`createFile()`** digunakan untuk membuat file.
- Metode atau fungsi **`emptyFile()`** digunakan untuk mengosongkan isi file.
- Metode atau fungsi **`deleteFile()`** digunakan untuk menghapus file.
- Metode atau fungsi **`copyFile()`** digunakan untuk menyalin file.
- Metode atau fungsi **`compileAndRun()`** digunakan untuk mengkompilasi dan menjalankan file.
- Metode atau fungsi **`readFile()`** digunakan untuk membaca file.
- Metode atau fungsi **`choice()`** digunakan untuk menampilkan menu pilihan utama.
- Metode atau fungsi **`addText()`** digunakan untuk menambah teks pada file.
- Metode atau fungsi **`deleteText()`** digunakan untuk menghapus teks pada file.
- Metode atau fungsi **`findText()`** digunakan untuk mencari teks pada file.
- Metode atau fungsi **`save()`** digunakan untuk menyimpan teks yang telah diedit pada file.
- Metode atau fungsi **`edit()`** digunakan untuk menampilkan menu pilihan edit.
- Metode atau fungsi **`~TextEditor`** merupakan destructor untuk membersihkan sumber daya yang digunakan oleh objek **`TextEditor`**.

2.2 Flowchart

Berikut adalah flowchart dari program text editor



2.3 Desain

1. Tampilan menu pilihan utama dan hasil

a. Tampilan menu pilihan utama

Berikut adalah tampilan menu pilihan utama:

```
~~~~~  
| TEKS EDITOR |  
~~~~~  
  
-----  
# menu pilihan utama #  
-----  
  
R - baca file  
C - buat file  
E - kosongkan file  
D - hapus file  
G - salin file  
O - kompilasi dan jalankan file  
Q - exit  
Pilih perintah: █
```

b. Tampilan hasil dari setiap menu pilihan utama

- Baca File

Berikut adalah tampilan hasil dari menu R (baca file) saat membaca file 'hello.cpp':

```
~~~~~  
| TEKS EDITOR |  
~~~~~  
  
-----  
# menu pilihan utama #  
-----  
  
R - baca file  
C - buat file  
E - kosongkan file  
D - hapus file  
G - salin file  
O - kompilasi dan jalankan file  
Q - exit  
Pilih perintah: R  
Masukkan nama file : hello.cpp█
```

```

1 #include <iostream>
2
3 using namespace std;
4
5 class HelloWorld
6 {
7     public:
8         void helo()
9         {
10             cout << "Hello, world!" << endl;
11         }
12 };
13
14 int main()
15 {
16     HelloWorld h1;
17
18     h1.helo();
19
20     return 0;
21 }

```

Edit Teks

A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: █

- Buat File

Berikut adalah tampilan hasil dari menu C (buat file):

```

~~~~~
| TEKS EDITOR |
~~~~~

-----

# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: C
Masukkan nama file berekstensi (contoh : uwau.txt): hello.txt
File berhasil dibuat.
Tekan Enter untuk melanjutkan...█

```

- Berikut adalah tampilan hasil dari menu E (kosongkan file):

```
~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit

Pilih perintah: E
Masukkan nama file: hello.txt
File berhasil dikosongkan.
Tekan Enter untuk melanjutkan...
```

- Berikut adalah tampilan hasil dari menu D (hapus file):

```
~~~~~  
| TEKS EDITOR |  
~~~~~  
  
-----  
# menu pilihan utama #  
-----  
  
R - baca file  
C - buat file  
E - kosongkan file  
D - hapus file  
G - salin file  
O - kompilasi dan jalankan file  
Q - exit  
  
Pilih perintah: D  
Masukkan nama file: hello.txt  
File berhasil dihapus.  
Tekan Enter untuk melanjutkan... 
```

- Salin file

Berikut adalah tampilan hasil dari menu G (salin file):

```
~~~~~
| TEKS EDITOR |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: G
Masukkan nama file yang ingin disalin: hello.cpp
Masukkan nama file baru: hello1.cpp
File berhasil disalin.
Tekan Enter untuk melanjutkan...█
```

- Kompilasi dan jalankan file

Berikut adalah tampilan hasil dari menu O (kompilasi dan jalankan file)

saat menjalankan file hello.cpp:

```
~~~~~
| TEKS EDITOR |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: O
Masukkan nama file dengan ekstensi .cpp (contoh: test.cpp): hello.cpp
Program berhasil dikompilasi.
Hello, world!
Tekan Enter untuk melanjutkan...█
```

- Exit

Berikut adalah tampilan hasil dari menu Q (exit):

```
~~~~~
| TEKS EDITOR |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: Q
Tekan Enter untuk melanjutkan...
```

2. Tampilan menu edit teks dan hasil

a. Tampilan menu edit teks

Menu edit teks keluar saat mengakses menu baca file pada menu pilihan utama.

Berikut adalah tampilan menu edit teks pada file 'pbo.txt':

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek
2 Saya mahasiswa TI Universitas Sumatera Utara

-----
# Edit Teks #
-----

A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: 
```

b. Tampilan hasil dari setiap menu edit teks

- Tambah teks

Berikut adalah tampilan hasil dari menu A (tambah teks):

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek
2 Saya mahasiswa TI Universitas Sumatera Utara

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: A
Masukkan posisi kursor(pilih baris): 1
Teks baru(ketik xx untuk menyelesaikan):
pada semester 2
xx
Tekan Enter untuk melanjutkan...
```

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2
2 Saya mahasiswa TI Universitas Sumatera Utara

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah:
```

- Hapus teks

Berikut adalah tampilan hasil dari menu D (hapus teks):

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2
2 Saya mahasiswa TI Universitas Sumatera Utara

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah:
```

Pada menu hapus teks terdapat menu pilihan. Berikut adalah tampilan hasil dari setiap menu:

➤ Hapus baris

Berikut adalah tampilan hasil dari menu B (hapus baris):

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2
2 Saya mahasiswa TI Universitas Sumatera Utara

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: B
Masukkan posisi kursor (pilih baris) : 2
Masukkan banyak baris yang akan dihapus : 1
Baris telah dihapus
Tekan Enter untuk melanjutkan...

1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: 
```

➤ Hapus teks

Berikut adalah tampilan hasil dari menu T (hapus teks):

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: T
Masukkan posisi kursor (pilih baris) : 1
Masukkan posisi kursor (pilih character) : 5
Masukkan panjang (character) yang akan dihapus : 6
Tekan Enter untuk melanjutkan...

1 Saya mempelajari Pemrograman Berorientasi Objek pada semester 2

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: 
```

- Ubah teks

Berikut adalah tampilan hasil dari menu E (ubah teks):

```
1 Saya mempelajari Pemrograman Berorientasi Objek pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: 
```

Pada menu ubah teks terdapat menu pilihan. Berikut adalah tampilan hasil dari setiap menu:

➤ Ubah baris

Berikut adalah tampilan hasil dari menu B (ubah baris):

```
1 Saya mempelajari Pemrograman Berorientasi Objek pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: B
Masukkan posisi kursor (pilih baris) : 1
Teks baru : Saya mempelajari PBO pada semester 2
Baris berhasil diubah
Tekan Enter untuk melanjutkan...

1 Saya mempelajari PBO pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah:
```

➤ Ubah teks

Berikut adalah tampilan hasil dari menu T (ubah teks):

```
1 Saya mempelajari PBO pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: T
Masukkan posisi kursor (pilih baris) : 1
Masukkan posisi kursor (pilih character) : 18
Masukkan panjang (character) yang akan diubah : 3
Teks baru : PBO (Pemrograman Berorientasi Objek)
Tekan Enter untuk melanjutkan...

1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah:
```


- Cari Teks

Berikut adalah tampilan hasil dari menu F (cari teks):

```

1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: F
Masukkan teks yang ingin ditemukan: pada
kata "pada" ditemukan pada kalimat:
Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2 (baris ke 1)
Tekan Enter untuk melanjutkan...

```

- Simpan dan keluar

Berikut adalah tampilan hasil dari menu S (simpan dan keluar):

```

1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: S
File berhasil disimpan.
Tekan Enter untuk melanjutkan...

```

- Keluar tanpa menyimpan

Berikut adalah tampilan hasil dari menu Q (keluar tanpa menyimpan):

```

1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: Q
Keluar tanpa menyimpan.
Tekan Enter untuk melanjutkan...

```

2.4 Hasil Implementasi

1. Menu pilihan utama

Code :

```
void TextEditor::choice() {
    char command;
    do {
        system("cls");
        cout << "\033[32m";
        cout << "~~~~~" << endl;
        cout << "| TEKS EDITOR |" << endl;
        cout << "~~~~~" << endl;
        cout << endl;
        cout << "\033[35m";
        cout << "-----" << endl;
        cout << "# menu pilihan utama #" << endl;
        cout << "-----" << endl;
        cout << "\033[0m";
        cout << "R - baca file" << endl;
        cout << "C - buat file" << endl;
        cout << "E - kosongkan file" << endl;
        cout << "D - hapus file" << endl;
        cout << "G - salin file" << endl;
        cout << "O - kompilasi dan jalankan file" << endl;
        cout << "Q - exit" << endl;
        cout << "Pilih perintah: ";
        cin >> command;
        switch (command) {
            case 'R':
                readFile();
                break;
            case 'C':
                createFile();
                break;
            case 'E':
                emptyFile();
                break;
            case 'D':
                deleteFile();
                break;
            case 'G':
                copyFile();
                break;
            case 'O':
                compileAndRun();
                break;
            case 'Q':
                break;
            default:
                cout << "Perintah tidak valid." << endl;
                break;
        }
        cout << "Tekan Enter untuk melanjutkan...";
        cin.ignore();
        cin.get();
    } while (command != 'Q');
}
```

Penjelasan:

- Pertama, menginisialisasi variabel `command` bertipe char.
- Selanjutnya melakukan perulangan `do while` yang diawali dengan membersihkan layar system menggunakan fungsi `system('cls')`.
- Set warna `\033[32m` dari tampilan output 'Teks Editor' dan menampilkannya.
- Set warna `\033[32m` dari tampilan 'menu pilihan utama' dan menampilkannya.
- Tidak lupa untuk mengembalikan warna teks ke semula dengan kode warna `\033[0m` dan menampilkan seluruh fitur dari Teks Editor seperti yang terlihat pada tampilan dibawah, dilanjutkan meminta inputan command dari pengguna.
- Kemudian menggunakan pengkondisian `switch case` untuk menyesuaikan command yang diinput dengan pemanggilan fungsi yang akan dijalankan, seperti command `'R'` untuk `readFile()`, command `'C'` untuk `createFile()`, command `'E'` untuk `emptyFile()`, command `'D'` untuk `deleteFile()`, command `'G'` untuk `copyFile()`, command `'O'` untuk `compileAndRun()`, command `'Q'` untuk `exit`.
- Perulangan `dowhile` akan berhenti jika pengguna menginput karakter `'Q'`.

Tampilan pada terminal :

```
~~~~~
| TEKS EDITOR |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: █
```

2. Fitur baca file

Code :

```
void TextEditor::readFile() {
    cout << "Masukkan nama file : ";
    cin >> fileName;
    ifstream file(fileName);
    if (file.is_open()) {
        string line;
        while (getline(file, line)) {
            lines.push_back(line); //code utama untuk membaca file yaitu push_back() yang merupakan fungsi khusus pada vector
        }
        file.close();
        edit();
    }
    else {
        cout << "File tidak ditemukan." << endl;
    }
    lines.clear();
}
```

Penjelasan :

- Pertama pengguna diminta untuk memasukkan nama file yang sudah ada.
- Selanjutnya membuat objek `ifstream file(filename)` yang berfungsi untuk membaca file sesuai dengan nama file yang diinput pengguna.
- Kemudian, jika file berhasil dibuka, maka program mengambil setiap baris dalam file lalu ditampung di variabel `line` bertipe data string. baris yg sudah diambil ditampilkan di terminal menggunakan fungsi `push_back()` yang merupakan fungsi khusus pada vector. Untuk baris selanjutnya dilakukan hal yang sama seperti baris pertama, baris pertama yang ditampung di variabel `line` digantikan oleh baris selanjutnya, lalu ditampilkan di terminal menggunakan fungsi `push_back()`.
- Selanjutnya, file ditutup menggunakan fungsi `file_close()`. Kemudian memanggil fungsi `edit()` untuk memperbolehkan pengguna melakukan pengeditan pada teks yang telah dibaca.
- Jika file yang dibuka tidak ada, maka akan mengeluarkan output “file tidak ditemukan”.
- Terakhir, memanggil fungsi `lines.clear()` untuk menghapus tampilan teks sebelumnya yang ada di terminal ketika kembali ke menu pilihan.

Tampilan hasil pada terminal :

```
~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: R
Masukkan nama file : hello.cpp

1 #include <iostream>
2
3 using namespace std;
4
5 class HelloWorld
6 {
7     public:
8         void helo()
9         {
10             cout << "Hello, world!" << endl;
11         }
12 };
13
14 int main()
15 {
16     HelloWorld h1;
17
18     h1.helo();
19
20     return 0;
21 }

-----
# Edit Teks #
-----

A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: 
```

3. Fitur buat file

Code :

```
void TextEditor::createFile() {
    cout << "Masukkan nama file berekstensi (contoh : uwau.txt): ";
    cin >> fileName;
    ofstream file(fileName);           //file dibuat menggunakan objek ofstream bernama file sesuai inputan
    if (file) {
        cout << "File berhasil dibuat." << endl;
        file.close();
    } else {
        cout << "File gagal dibuat." << endl;
    }
}
```

Penjelasan :

- Pertama, program akan meminta user untuk memasukkan nama file yang ingin dibuat
- kemudian ditampung menggunakan objek ofstream yang bernama `file`.
- Jika `file` berhasil dibuat maka program akan dijalankan
- kemudian file akan ditutup menggunakan fungsi `file.close()`
- Dan apabila gagal maka program gagal untuk dijalankan

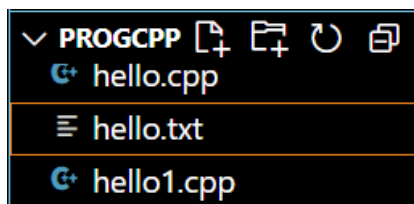
Tampilan pada terminal :

```
~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: C
Masukkan nama file berekstensi (contoh : uwau.txt): hello.txt
File berhasil dibuat.
Tekan Enter untuk melanjutkan...
```

Tampilan hasil :



4. Fitur kosongkan file

Code :

```
void TextEditor::emptyFile() {
    cout << "Masukkan nama file: ";
    cin >> fileName;
    ifstream inputFile(fileName);

    if (inputFile.is_open()) {                                     //last update : pengecekan file tersedia
        inputFile.close();

        ofstream outputFile(fileName, ios::trunc); //code utama untuk mengosongkan teks pada file menggunakan objek
                                                    ofstream dengan mode ios::trunc

        if (outputFile.is_open()) {
            outputFile.close();
            cout << "File berhasil dikosongkan.\n";
        } else {
            cout << "Gagal membuka file untuk pengosongan.\n";
        }
    } else {
        cout << "Gagal membuka file.\n";
    }
}
```

Penjelasan :

- Pertama, program akan meminta user untuk memasukkan nama file yang ingin dikosongkan isi filenya yang kemudian ditampung menggunakan objek ifstream yang bernama **`inputFile`**.
- Kemudian selanjutnya jika file berhasil dibuka dengan menggunakan fungsi **`outputFile.is_open()`** , kemudian file akan ditutup dengan menggunakan **`inputFile.close()`**
- kemudian program akan mengkosongkan file dengan menggunakan objek ofstream yang bernama **`outputFile`** dengan metode **`ios::trunc`** .
- Jika file yang telah dibuat menggunakan ofstream berhasil dibuka dengan fungsi **`outputFile.is_open()`** maka program akan menutup file tersebut dengan fungsi **`outputFile.close()`** , jika tidak berhasil dibuka maka program gagal dijalankan.
- Dan apabila file yang kita input tidak bisa dibuka menggunakan fungsi **`outputFile.is_open()`** maka program gagal dijalankan.

```

~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit

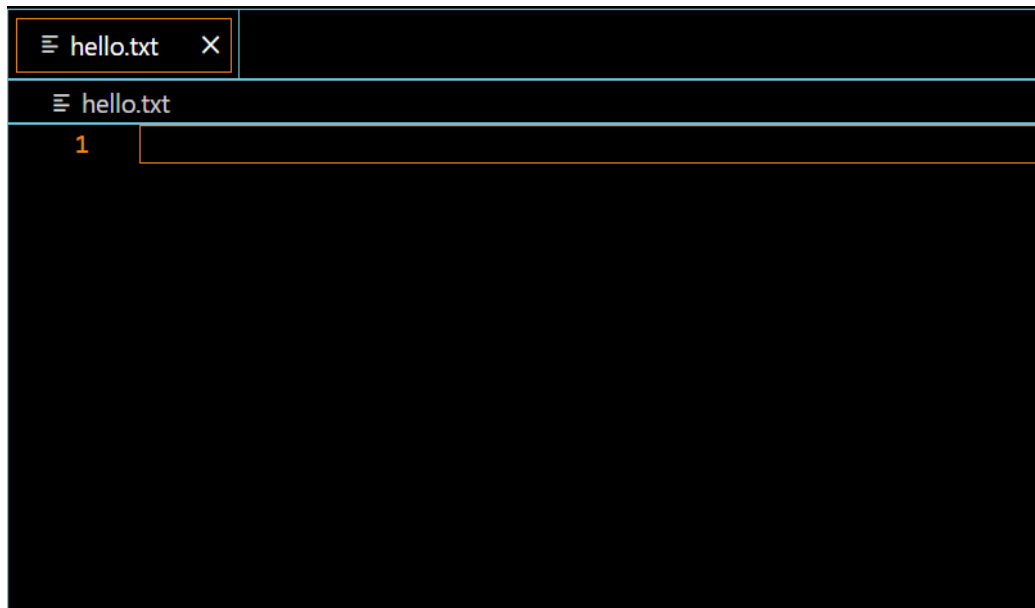
Pilih perintah: E
Masukkan nama file: hello.txt
File berhasil dikosongkan.
Tekan Enter untuk melanjutkan...

```

File sebelum dikosongkan :

```
hello.txt x
hello.txt
1 Hello world !
2 welcome to object oriented programming
```


File sesudah dikosongkan :



5. Fitur hapus file

Code :

```
void TextEditor::deleteFile() {
    cout << "Masukkan nama file: ";
    cin >> fileName;
    if (remove(fileName.c_str()) == 0) {    //fungsi remove menghapus file, jika menghasilkan 0 maka fungsi tidak
                                            menghasilkan pesan kesalahan dan file berhasil dihapus
        cout << "File berhasil dihapus." << endl;
    } else {
        cout << "File gagal dihapus." << endl;
    }
}
```

Penjelasan :

- Pertama, program akan meminta user menginput sebuah file yang akan dihapus filenya yang ditampung menggunakan variable **`filename`**.
- kemudian jika file ada dan fungsi **`remove(filename.c_Str())`** bernilai 0 maka program akan menghapus file tersebut dan jika bernilai 1 maka program tidak berhasil dijalankan.

Tampilan pada terminal :

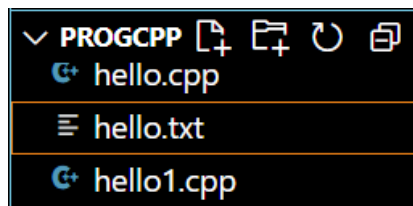
```
~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

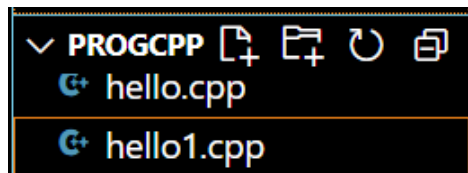
R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: D
Masukkan nama file: hello.txt
File berhasil dihapus.
Tekan Enter untuk melanjutkan...
```

Tampilan hasil :

Sebelum file 'hello.txt' dihapus :



Sesudah file 'hello.txt' dihapus :



6. Fitur salin file

Code :

```
void TextEditor::copyFile() {
    string newFileName;
    cout << "Masukkan nama file yang ingin disalin: ";
    cin >> fileName;
    cout << "Masukkan nama file baru: ";
    cin >> newFileName;

    ifstream sourceFile(fileName);
    ofstream destFile(newFileName);

    if (sourceFile && destFile) {
        destFile << sourceFile.rdbuf(); //fungsi rdbuf() sebagai fungsi utama digunakan untuk menyalin file
        cout << "File berhasil disalin." << endl;
    } else {
        cout << "Gagal menyalin file." << endl;
    }

    sourceFile.close();
    destFile.close();
}
```

Penjelasan :

- Pertama, pengguna diminta untuk memasukkan nama file yang ingin disalin.
- Lalu, pengguna juga diminta untuk memasukkan nama file baru yang akan jadi salinan file yang telah dimasukkan sebelumnya.
- Nama file baru ditampung di variabel ``newFileName`` bertipe string yang sudah dideklarasikan sebelumnya.
- Kemudian, membuat objek ``ifstream sourceFile(filename)`` yang berfungsi untuk membuka lalu membaca isi file sumber yang ingin disalin sesuai dengan nama file yang dimasukkan pengguna.
- Selanjutnya membuat objek ``ofstream destFile(newFileName)`` yang berfungsi untuk membuka lalu menulis isi file tujuan yang sesuai dengan isi file yang ingin disalin pada file baru dengan nama file yang telah dimasukkan pengguna.
- Jika file sumber (file yang ingin disalin) dan file tujuan (file hasil salinan) bisa dibuka, maka dengan menggunakan fungsi ``rdbuf()`` file sumber (``sourceFile``) dibaca, dan dimasukkan ke buffer (memori sementara) lalu dimasukkan file tujuan (``destFile``).
- Dengan kata lain, fungsi ini berguna untuk menyalin isi file sumber ke file tujuan. Jika salah satu atau kedua file, baik file sumber atau file tujuan tidak bisa dibuka, maka akan mengeluarkan output “Gagal menyalin file”.
- Kemudian, file sumber ditutup setelah selesai disalin dan file tujuan juga ditutup setelah selesai ditulis.

Tampilan pada terminal :

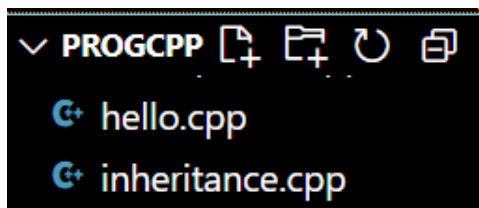
```
~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

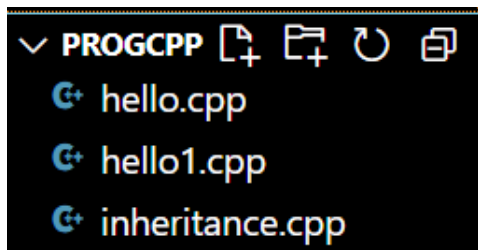
R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: G
Masukkan nama file yang ingin disalin: hello.cpp
Masukkan nama file baru: hello1.cpp
File berhasil disalin.
Tekan Enter untuk melanjutkan...█
```

Tampilan hasil :

Sebelum file hello.cpp disalin:



Sesudah file hello.cpp disalin ke file hello1.cpp:



Isi file hello.cpp dan hello1.cpp :

```
hello.cpp X
G+ hello.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  class HelloWorld
6  {
7      public:
8          void helo()
9          {
10             cout << "Hello, world!" << endl;
11         }
12     };
13
14     int main()
15     {
16         HelloWorld h1;
17
18         h1.helo();
19
20         return 0;
21     }
```

```
hello1.cpp X
G+ hello1.cpp
1  #include <iostream>
2
3  using namespace std;
4
5  class HelloWorld
6  {
7      public:
8          void helo()
9          {
10             cout << "Hello, world!" << endl;
11         }
12     };
13
14     int main()
15     {
16         HelloWorld h1;
17
18         h1.helo();
19
20         return 0;
21     }
```

7. Fitur kompilasi dan jalankan file

Code :

```
void TextEditor::compileAndRun() {
    cout << "Masukkan nama file dengan ekstensi .cpp (contoh: test.cpp): ";
    cin >> fileName;

    string compileCommand = "g++ " + fileName + " -o temp";
    string runCommand = "temp.exe";

    int compileResult = system(compileCommand.c_str()); //code utama untuk mengkompilasi file yaitu
                                                         system(compileCommand.c_str()) untuk menjalankan command pada system terminal
    if (compileResult == 0) {
        cout << "Program berhasil dikompilasi." << "\033[33m" << endl;
        int runResult = system(runCommand.c_str()); //code utama untuk run file yaitu system(runCommand.c_str()) untuk
                                                         menjalankan command pada system terminal
        cout << "\033[0m";
        if (runResult != 0) {
            cout << "Program tidak dapat dijalankan." << endl;
        }
    } else {
        cout << "Program gagal dikompilasi." << endl;
    }
}
```

Penjelasan :

- Pesan "Masukkan nama file dengan ekstensi (contoh: test.cpp): " ditampilkan di layar.
- Pengguna diminta untuk memasukkan nama file dengan ekstensi yang sesuai menggunakan `cin >> fileName`. Contoh: "test.cpp".
- Sebuah string `compileCommand` dibuat dengan menggabungkan teks "g++ ", nama file, dan " -o temp". Ini akan menjadi perintah untuk mengompilasi file menggunakan `g++` dan menghasilkan file biner dengan nama "temp". Sebuah string `runCommand` dibuat dengan teks "temp.exe". Ini akan menjadi perintah untuk menjalankan file biner yang dihasilkan.
- Fungsi `system()` digunakan untuk menjalankan perintah kompilasi `compileCommand` dengan menggunakan `system(compileCommand.c_str())`. Nilai yang dikembalikan dari `system()` menunjukkan hasil dari eksekusi perintah tersebut.
- Jika nilai `compileResult` sama dengan 0, itu berarti kompilasi berhasil. Pesan "Program berhasil dikompilasi." ditampilkan di layar. Fungsi `system()` kemudian digunakan untuk menjalankan perintah eksekusi `runCommand` dengan menggunakan `system(runCommand.c_str())`.
- Nilai yang dikembalikan dari `system()` menunjukkan hasil dari eksekusi perintah tersebut. Jika nilai `runResult` tidak sama dengan 0, itu berarti program tidak dapat dijalankan. Pesan "Program tidak dapat dijalankan." ditampilkan di layar. Jika nilai `compileResult` tidak sama dengan 0, itu

berarti kompilasi gagal. Pesan "Program gagal dikompilasi." ditampilkan di layar.

- Dengan demikian, fungsi ``compileAndRun()`` bertanggung jawab untuk mengompilasi file yang diberikan menggunakan ``g++`` dan menjalankan program jika kompilasi berhasil, memberikan umpan balik yang sesuai ke pengguna dalam setiap langkahnya.

Tampilan hasil pada terminal :

```
~~~~~
|  TEKS EDITOR  |
~~~~~

-----
# menu pilihan utama #
-----

R - baca file
C - buat file
E - kosongkan file
D - hapus file
G - salin file
O - kompilasi dan jalankan file
Q - exit
Pilih perintah: O
Masukkan nama file dengan ekstensi .cpp (contoh: test.cpp): hello.cpp
Program berhasil dikompilasi.
Hello, world!
Tekan Enter untuk melanjutkan...|
```

8. Menu edit

Code :

```
void TextEditor::edit() {
    char command;
    do {
        int k = 1;
        system("cls");
        for (const string& line : lines) {
            cout << "\033[34m" << setw(3) << k << " " << "\033[37m" << line << endl;
            k++;
        }
        cout << endl;
        cout << "\033[35m";
        cout << "-----" << endl;
        cout << "# Edit Teks #" << endl;
        cout << "-----" << endl;
        cout << "\033[0m";
        cout << "A - tambah teks" << endl;
        cout << "D - hapus teks" << endl;
        cout << "E - ubah teks" << endl;
        cout << "F - cari teks" << endl;
        cout << "S - simpan dan keluar" << endl;
        cout << "Q - keluar tanpa menyimpan" << endl;
        cout << "Pilih perintah: ";
        cin >> command;
        switch (command) {
            case 'A':
                addText();
                break;
            case 'D':
                deleteText();
                break;
            case 'E':
                editText();
                break;
            case 'F':
                findText();
                break;
            case 'S':
                save();
                break;
            case 'Q':
                cout << "Keluar tanpa menyimpan." << endl;
                break;
            default:
                cout << "Perintah tidak valid." << endl;
                break;
        }
        cout << "Tekan Enter untuk melanjutkan...";
        cin.ignore();
        cin.get();
    } while (command != 'Q' && command != 'S');
}
```

Penjelasan :

- Pertama, menginisialisasi variabel `command` bertipe char.
- Selanjutnya melakukan perulangan do while yang diawali dengan membersihkan layar system menggunakan fungsi `system('cls')`.

- Set warna `\033[35m`` dari tampilan 'Edit Teks' dan menampilkannya.
- Tidak lupa untuk mengembalikan warna teks ke semula dengan kode warna `\033[0m`` dan menampilkan seluruh fitur dari menu Edit Teks seperti yang terlihat pada tampilan dibawah, dilanjutkan meminta inputan command dari pengguna.
- Kemudian menggunakan pengkondisian switch case untuk menyesuaikan command yang diinput dengan pemanggilan fungsi yang akan dijalankan, seperti command 'A' untuk ``addText()``, command 'D' untuk ``deleteText()``, command 'E' untuk ``editText()``, command 'F' untuk ``findText()``, command 'S' untuk ``save()``, command 'Q' untuk ``exit``.
- Perulangan do while akan berhenti jika pengguna menginput karakter 'S' dan 'Q'.

Tampilan pada terminal (muncul setelah menu baca file dipilih) :

```

1 Saya ingin mempelajari Pemrograman Berorientasi Objek
2 Saya mahasiswa TI Universitas Sumatera Utara

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: █

```

9. Fitur tambah teks

Code :

```

//fungsi addText menambahkan teks pada posisi kursor yang ditentukan oleh pengguna ke dalam lines.
void TextEditor::addText() {
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    string newText;

    cout << "Masukkan posisi kursor(pilih baris): ";
    cin >> cursorPosition;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Teks baru(ketik xx untuk menyelesaikan): \n";

    while (true) {
        int i = cursorPosition - 1;
        getline(cin, newText);

        if (newText == "xx") {
            lines.erase(lines.begin() + i);
            break;
        }

        lines.insert(lines.begin() + cursorPosition, "");
        lines[i].append(newText);
        cursorPosition++;
    }
}

```

Penjelasan :

- `cin.ignore(numeric_limits<streamsize>::max(), '\n')` Mengabaikan atau menghapus karakter yang tersisa di stream input setelah membaca input sebelumnya.
- Pendeklarasian variabel `string newText` untuk menyimpan teks baru yang dimasukkan oleh pengguna.
- Meminta inputan posisi kursor (memilih baris) oleh pengguna dan ditampilkan dalam variabel `cursorPosition`.
- Meminta inputan teks baru yang akan ditambahkan oleh pengguna menggunakan perulangan `while()` dan fungsi `getline()` untuk mengambil setiap line dari inputan, lalu terdapat base case `if (newText == "xx")` untuk menghentikan inputan teks oleh pengguna, dilanjutkan dengan menambahkan baris baru menggunakan fungsi `insert()` dan menambahkan teks baru pada baris yang telah ditambahkan menggunakan fungsi `append()`. Setelahnya `cursorPosition` akan di increment untuk melanjutkan perulangan pada baris selanjutnya.

Tampilan hasil pada terminal :

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek
2 Saya mahasiswa TI Universitas Sumatera Utara

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: A
Masukkan posisi kursor(pilih baris): 1
Teks baru(ketik xx untuk menyelesaikan):
    pada semester 2
xx
Tekan Enter untuk melanjutkan...
```

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2
2 Saya mahasiswa TI Universitas Sumatera Utara
```

```
-----
# Edit Teks #
-----
```

```
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: █
```

10. Fitur Hapus teks

Code :

```
void TextEditor::deleteText() {
    int i, j;
    int cursor;
    int length;
    char command;
    do {
        int k = 1;
        system("cls");
        for (const string& line : lines) {
            cout << "\033[34m" << setw(3) << k << " " << "\033[37m" << line << endl;
            k++;
        }
        cout << endl;
        cout << "\033[35m";
        cout << "Hapus Teks : " << endl;
        cout << "\033[0m";
        cout << "B - hapus baris" << endl;
        cout << "T - hapus teks" << endl;
        cout << "Q - kembali ke menu edit" << endl;
        cout << "Pilih perintah: ";
        cin >> command;
        switch (command) {
            case 'B': //case untuk menghapus setiap baris
                cout << "Masukkan posisi kursor (pilih baris) : ";
                cin >> cursorPosition;
                cout << "Masukkan banyak baris yang akan dihapus : ";
                cin >> length;
                i = cursorPosition - 1;
                if (i >= 0 && i + length <= lines[i].size()) {
                    lines.erase(lines.begin() + i, lines.begin() + i + length); //fungsi erase() untuk menghapus baris
                    cout << "Baris telah dihapus" << endl;
                } else {
                    cout << "banyak baris melebihi batas." << endl;
                }
                break;
            case 'T': //case untuk menghapus teks pada baris
                cout << "Masukkan posisi kursor (pilih baris) : ";
                cin >> cursorPosition;
                cout << "Masukkan posisi kursor (pilih character) : ";
                cin >> cursor;
                cout << "Masukkan panjang (character) yang akan dihapus : ";
                cin >> length;
                i = cursorPosition - 1;
                j = cursor - 1;
                if (i >= 0 && i <= lines.size() && j >= 0 && j < lines[i].size()) {
                    lines[i].erase(lines[i].begin() + j, lines[i].begin() + j + length); //fungsi erase() untuk menghapus baris
                } else {
                    cout << "Baris tidak ditemukan" << endl;
                }
                break;
            case 'Q':
                break;
            default:
                cout << "Perintah tidak valid." << endl;
                break;
        }
        cout << "Tekan Enter untuk melanjutkan...";
        cin.ignore();
        cin.get();
    } while (command != 'Q');
}
```

Penjelasan :

- Pendeklarasian variabel `int i, j, cursor, length` untuk digunakan pada perulangan pada fungsi, `char command` untuk menampung inputan command pengguna.
- Perulangan `do while` untuk menampilkan pilihan fitur antara `B` untuk hapus baris dan `T` untuk hapus teks.
- Perulangan `for (const string& line : lines)` digunakan untuk menampilkan lagi isi file yang telah dibaca sebelumnya.
- Menampilkan menu hapus teks dengan warna yang ditentukan pada code program dan meminta inputan pengguna untuk fitur yang dipilih dan akan dilanjutkan dengan pengkondisian `switch case`.
- Case `B` meminta inputan pengguna berupa posisi kursor (pilih baris) dan banyak baris yang akan dihapus, selanjutnya menggunakan pengkondisian `if()` jika baris ditemukan pada file dan menghapus baris yang telah dipilih sebelumnya menggunakan fungsi `erase()`, namun jika kondisi `else()` yang terpenuhi pesan kesalahan akan ditampilkan.
- Case `T` meminta inputan pengguna berupa posisi kursor (pilih baris), posisi kursor (pilih karakter) dan panjang karakter yang akan dihapus, selanjutnya menggunakan pengkondisian `if()` jika baris ditemukan pada file dan menggunakan lagi pengkondisian `if()` jika ditemukan karakter yang dipilih dan menghapus karakter tersebut menggunakan fungsi `erase()`, namun jika kondisi `else()` yang terpenuhi pesan kesalahan akan ditampilkan.

Tampilan pada terminal (menu hapus teks) :

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2
2 Saya mahasiswa TI Universitas Sumatera Utara

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: █
```

Tampilan hasil pada terminal (hapus baris) :

```
1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: █

1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2
2 Saya mahasiswa TI Universitas Sumatera Utara

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: B
Masukkan posisi kursor (pilih baris) : 2
Masukkan banyak baris yang akan dihapus : 1
Baris telah dihapus
Tekan Enter untuk melanjutkan...█
```

Tampilan hasil pada terminal (hapus teks) :

```
1 Saya mempelajari Pemrograman Berorientasi Objek pada semester 2

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: █

1 Saya ingin mempelajari Pemrograman Berorientasi Objek pada semester 2

Hapus Teks :
B - hapus baris
T - hapus teks
Q - kembali ke menu edit
Pilih perintah: T
Masukkan posisi kursor (pilih baris) : 1
Masukkan posisi kursor (pilih character) : 5
Masukkan panjang (character) yang akan dihapus : 6
Tekan Enter untuk melanjutkan...█
```

11. Fitur ubah teks

Code :

```
//fungsi editText mengubah teks yang dipilih oleh pengguna dalam lines.
void TextEditor::editText() {
    cin.ignore();
    string newText;
    int i, j, cursor, length;
    char command;
    do {
        int k = 1;
        system("cls");
        for (const string& line : lines) {
            cout << "\033[34m" << setw(3) << k << " " << "\033[37m" << line << endl;
            k++;
        }
        cout << endl;
        cout << "\033[35m";
        cout << "Ubah Teks : " << endl;
        cout << "\033[0m";
        cout << "B - ubah baris" << endl;
        cout << "T - ubah teks" << endl;
        cout << "Q - kembali ke menu edit" << endl;
        cout << "Pilih perintah: ";
        cin >> command;
        switch (command) {
            case 'B':
                cout << "Masukkan posisi kursor (pilih baris) : ";
                cin >> cursorPosition;
                cin.ignore();
                cout << "Teks baru : ";
                getline(cin, newText);
                i = cursorPosition - 1;
                if (i >= 0 && i < lines[i].size()) {
                    lines.erase(lines.begin() + i);
                    lines.insert(lines.begin() + i, "");
                    lines[i].append(newText);
                    cout << "Baris berhasil diubah" << endl;
                } else {
                    cout << "Baris tidak ditemukan." << endl;
                }
                break;
            case 'T':
                cout << "Masukkan posisi kursor (pilih baris) : ";
                cin >> cursorPosition;
                cout << "Masukkan posisi kursor (pilih character) : ";
                cin >> cursor;
                cout << "Masukkan panjang (character) yang akan diubah : ";
                cin >> length;
                cin.ignore();
                cout << "Teks baru : ";
                getline(cin, newText);
                i = cursorPosition - 1;
                j = cursor - 1;
                if (i >= 0 && i < lines.size()){
                    if(j >= 0 && j < lines[i].size()){
                        lines[i].erase(lines[i].begin() + j, lines[i].begin() + j + length);
                        lines[i].insert(j, newText);
                    } else {
                        cout << "posisi kursor dalam baris (character) tidak ditemukan";
                    }
                } else {
                    cout << "Baris tidak ditemukan" << endl;
                }
                break;
            case 'Q':
                break;
            default:
                cout << "Perintah tidak valid." << endl;
                break;
        }
        cout << "Tekan Enter untuk melanjutkan...";
        cin.ignore();
        cin.get();
    } while (command != 'Q');
}
```

Penjelasan :

- Pendeklarasian variabel ``string newText`` untuk menyimpan teks baru, ``int i, j, cursor, length`` untuk digunakan pada perulangan pada fungsi, ``char command`` untuk menampung inputan command pengguna.
- Perulangan do while untuk menampilkan pilihan fitur antara ``B`` untuk ubah baris dan ``T`` untuk ubah teks.
- Perulangan ``for (const string& line : lines)`` digunakan untuk menampilkan lagi isi file yang telah dibaca sebelumnya.
- Menampilkan menu ubah teks dengan warna yang ditentukan pada code program dan meminta inputan pengguna untuk fitur yang dipilih dan akan dilanjutkan dengan pengkondisian switch case.
- Case ``B`` meminta inputan pengguna berupa posisi kursor (pilih baris) dan teks baru yang akan diubah, selanjutnya menggunakan pengkondisian ``if()`` jika baris ditemukan pada file dan menghapus baris yang telah dipilih sebelumnya menggunakan fungsi ``erase()``, lalu memasukkan baris baru pada baris yang dihapus menggunakan fungsi ``insert()`` dilanjutkan dengan menambahkan teks baru pada baris yang telah diinsert menggunakan fungsi ``append()``, namun jika kondisi ``else()`` yang terpenuhi pesan kesalahan akan ditampilkan.
- Case ``T`` meminta inputan pengguna berupa posisi kursor (pilih baris), posisi kursor (pilih karakter) dan teks baru yang akan diubah, selanjutnya menggunakan pengkondisian ``if()`` jika baris ditemukan pada file dan menggunakan lagi pengkondisian ``if()`` jika ditemukan karakter yang dipilih dan menghapus karakter tersebut menggunakan fungsi ``erase()``, lalu memasukkan teks baru pada karakter yang dihapus menggunakan fungsi ``insert()``, namun jika kondisi ``else()`` yang terpenuhi pesan kesalahan akan ditampilkan.

Tampilan pada terminal (menu ubah teks):

```
1 Saya mempelajari Pemrograman Berorientasi Objek pada semester 2
Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: █
```

Tampilan hasil pada terminal (ubah baris) :

```
1 Saya mempelajari Pemrograman Berorientasi Objek pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: B
Masukkan posisi kursor (pilih baris) : 1
Teks baru : Saya mempelajari PBO pada semester 2
Baris berhasil diubah
Tekan Enter untuk melanjutkan...█
```

```
1 Saya mempelajari PBO pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: █
```

Tampilan hasil pada terminal (ubah teks) :

```
1 Saya mempelajari PBO pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: T
Masukkan posisi kursor (pilih baris) : 1
Masukkan posisi kursor (pilih character) : 18
Masukkan panjang (character) yang akan diubah : 3
Teks baru : PBO (Pemrograman Berorientasi Objek)
Tekan Enter untuk melanjutkan...█
```

```
1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

Ubah Teks :
B - ubah baris
T - ubah teks
Q - kembali ke menu edit
Pilih perintah: █
```


12. Fitur cari teks

Code :

```
//fungsi findText mencari teks yang dimasukkan oleh pengguna dalam lines.
void TextEditor::findText() {
    string findStr, replaceStr;
    cout << "Masukkan teks yang ingin ditemukan: ";
    cin.ignore();
    getline(cin, findStr);

    for (char& c : findStr) {
        c = std::tolower(c);
    }

    for (size_t i = 0; i < lines.size(); i++) {
        size_t pos = lines[i].find(findStr);
        if(true){
            cout << "kata \"" << "\033[33m" << findStr << "\033[0m" << "\" ditemukan pada kalimat: " << endl;
            break;
        }
    }

    for (size_t i = 0; i < lines.size(); i++) {
        string lowercaseLine = lines[i];
        for (char& c : lowercaseLine) {
            c = std::tolower(c);
        }

        size_t pos = lowercaseLine.find(findStr);
        while (pos != string::npos) {
            cout << "\033[33m" << lines[i] << "\033[0m" << " (baris ke " << i + 1 << ")" << endl;
            pos = lowercaseLine.find(findStr, pos + findStr.length());
        }
    }
}
```

Penjelasan :

- ``string findStr, replaceStr`` Mendeklarasikan dua variabel bertipe string yang akan digunakan untuk menyimpan teks yang akan dicari dan teks pengganti.
- Selanjutnya meminta inputan pengguna untuk memasukkan teks yang akan dicari dan ditampung oleh ``string findStr`` menggunakan fungsi ``getline()`` untuk membaca seluruh baris teks.
- Menggunakan perulangan ``for (char& c : findStr)`` untuk mengiterasi melalui setiap karakter dalam findStr yang akan mengubah setiap karakter menjadi huruf kecil menggunakan fungsi ``tolower()`` guna menghindari case-insensitive
- Menggunakan perulangan ``for (size_t i = 0; i < lines.size(); i++)`` untuk mengiterasi setiap elemen dalam vektor lines, selanjutnya mencari kemunculan pertama dari findStr dalam ``string lines[i]`` menggunakan fungsi ``find()`` mengembalikan posisi kemunculan pertama jika ditemukan atau ``string::npos`` jika tidak ditemukan. ``if(true)`` akan menampilkan kata yang diinputkan dengan warna kuning atau ``\033[33m``.

- Menggunakan perulangan `for (size_t i = 0; i < lines.size(); i++)` lagi, selanjutnya menginisialisasi `string lowercaseLine` untuk membuat salinan string `lines[i]` dan `string lowercaseLine` diubah setiap karakter menjadi huruf kecil guna pencarian case-insensitive.
- Selanjutnya menggunakan fungsi `find()` lagi dan perulangan `while()` untuk mencari setiap kemunculan dari teks yang diinputkan pengguna.

Tampilan hasil pada terminal :

```
1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: F
Masukkan teks yang ingin ditemukan: pada
kata "pada" ditemukan pada kalimat:
Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2 (baris ke 1)
Tekan Enter untuk melanjutkan...
```

13. Fitur simpan dan keluar

Code :

```
void TextEditor::save() {
    ofstream file(fileName);
    if (file.is_open()) {
        for (const string& line : lines) {
            file << line << endl; //code utama untuk menyimpan file yaitu memasukkan seluruh perubahan pada lines ke file
        }
        file.close();
        cout << "File berhasil disimpan." << endl;
    }
    else {
        cout << "File tidak dapat disimpan." << endl;
    }
}
/* fungsi save menyimpan perubahan yang dilakukan pada lines ke file yang sedang aktif */
```

Penjelasan :

- Pertama, membuka file dengan membuat objek `ofstream file(fileName)` yang berguna untuk membuka dan menulis isi file yang sudah dilakukan berbagai pengeditan dengan beberapa fitur.
- Jika file berhasil dibuka, maka dilakukan loop sepanjang vektor `lines` yang berisi baris yang telah diedit yang disimpan di `string& line` baris per baris.
- Lalu, dari variabel `line` dimasukkan ke variabel `file` yang dibuka baris perbaris yang berarti file yang sudah diedit berhasil diperbarui.

- Kemudian file ditutup dan mengeluarkan output “File berhasil disimpan”. Jika file tidak dapat dibuka maka, akan mengeluarkan output “File tidak dapat disimpan”.

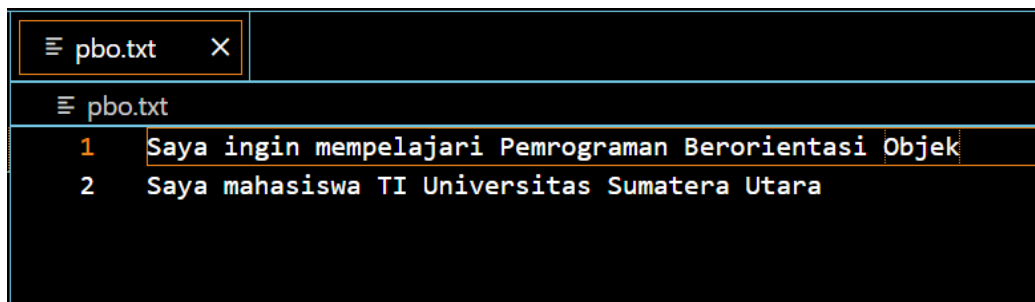
Tampilan pada terminal :

```
1 Saya mempelajari PBO (Pemrograman Berorientasi Objek) pada semester 2

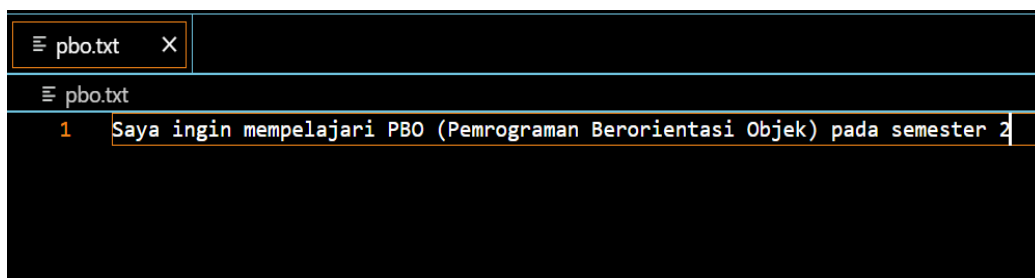
-----
# Edit Teks #
-----
A - tambah teks
D - hapus teks
E - ubah teks
F - cari teks
S - simpan dan keluar
Q - keluar tanpa menyimpan
Pilih perintah: S
File berhasil disimpan.
Tekan Enter untuk melanjutkan...
```

Tampilan hasil :

Sebelum dilakukan simpan dan keluar:



Sesudah dilakukan simpan dan keluar :



BAB III

PENUTUP

3.1 Kesimpulan

Dalam pembuatan program text editor menggunakan OOP C++ yang memiliki berbagai fitur ada beberapa hal yang diperlukan yaitu arsitektur program yang memuat library yang digunakan oleh program seperti <iostream>, <fstream>, <vector>, <limits>, dan <iomanip> yang memiliki beberapa fungsi yang mendukung pembuatan program text editor ini. Selain itu, dalam pembuatan program ini harus memperhatikan bagaimana alur programnya berjalan, untuk itu diperlukannya flowchart yang memuat alur, langkah dan proses berjalannya program. Selanjutnya, diperlukan juga desain atau tampilan program saat dijalankan oleh pengguna baik dari tampilan menu sampai output yang dikeluarkan oleh setiap fitur yang dipilih oleh pengguna. Kemudian, setiap code program diimplementasikan sesuai dengan fungsi dari setiap fitur yang ada. Jika beberapa hal tersebut dilakukan, maka dapat membuat program text editor dengan baik.

3.2 Saran

Program text editor menggunakan OOP C++ ini masih masih perlu dilakukan penyempurnaan dengan penambahan beberapa fitur atau penyempurnaan beberapa fitur. Penulis menyarankan kepada para pembaca untuk memahami Pemrograman Berbasis Objek (OOP) agar dapat mengimplementasikannya dalam pembuatan berbagai macam program.

DAFTAR PUSTAKA

Best Text Editor Software. (2023, Juni 12). Retrieved from G2.com:

<https://www.g2.com/categories/text-editor>

Nugraha, M. (2021). *Dasar Pemrograman Dengan C++ Materi Paling Dasar Untuk Menjadi Programmer Berbagai Platform*. Yogyakarta: Deepublish.

Oualline, S. (2003). *Practical C++ Programming*. California: O'Really Media, Inc.