

Laporan Program Alpenlify

Dokumen Tugas Besar MK Struktur Data dan Algoritma (Praktik)



Disusun oleh

Kelompok 2, Kelas 1-A

Andi Putra Wijaya	241511001
Gilang Aditya Sumarna	241511012
Raffi Fauzi Hermawan	241511025

**Jurusan Teknik Komputer dan Informatika
Program Studi D3 Teknik Informatika
Politeknik Negeri Bandung**

2025

TABEL REVISI

No	Tanggal	Keterangan	PIC
1			
2			
3			
4			

Daftar Isi

BAB 1 SPESIFIKASI PROGRAM	4
1.1 Definisi Program	4
1.2 Identifikasi Proses	5
1.3 Identifikasi Data	5
BAB 2 PERANCANGAN PROGRAM	10
2.1 Chart	10
2.1.1 Structured chart 1	10
2.1.2 Structured chart 2	10
2.1.3 Structured chart 3	11
2.1.4 Structured chart 4	12
2.1.4 Structured chart 5	13
2.2 Tabel Prosedur	13
2.2.1 Tree	13
2.2.2 Player.h	15
2.2.3 UI	17
2.2.4 Time	18
2.3 Algoritma	20
2.4 Perancangan Tampilan (Output)	32
BAB 3 HASIL AKHIR PROGRAM	36
3.1. Pembahasan Hasil Implementasi	36
BAB 4 KESIMPULAN	43
DAFTAR PUSTAKA	44
DAFTAR KONTRIBUSI ANGGOTA KELOMPOK	45

BAB 1 SPESIFIKASI PROGRAM

1.1 Definisi Program

Program music player adalah aplikasi perangkat lunak yang berfungsi untuk membaca, mendekode, dan memainkan file audio digital. Program ini mengimplementasikan berbagai komponen pemrosesan sinyal digital dan antarmuka pengguna untuk memungkinkan interaksi yang efisien dalam pemutaran audio.

Alpenlify adalah salah satu aplikasi pemutar musik yang dibuat untuk memenuhi tugas besar mata kuliah Struktur Data dan Algoritma. Alpenlify memiliki fitur sebagai berikut:

1. Memutar musik.
2. Membuat Playing *queue*.
3. *Seek* waktu musik.
4. *Skip* musik di *playing queue*.
5. *Rewind* musik yang sedang berjalan.
6. *Search* musik secara spesifik.
7. Lihat list musik di direktori musik pengguna.
8. Cek *queue* musik.

Untuk memastikan pengguna dapat menjalankan program dengan benar dan memanfaatkan seluruh fitur yang tersedia, berikut ini adalah aturan-aturan penggunaan yang harus diperhatikan selama program dijalankan.

1. Pengguna dapat melihat seluruh daftar musik yang tersedia di dalam folder musik lokal. Seluruh file musik yang terdapat dalam folder tersebut akan ditampilkan secara otomatis saat program dijalankan.
2. Pengguna dapat memilih musik dari daftar dan memasukkannya ke dalam antrian pemutaran. Musik yang berada di urutan pertama antrian akan langsung dimainkan secara otomatis.
3. Apabila terdapat musik dalam antrian, pengguna dapat memilih untuk mengulang lagu yang sedang diputar tanpa menghapusnya dari antrian.
4. Pengguna dapat melewati lagu yang sedang diputar dan langsung melanjutkan ke lagu berikutnya dalam antrian, jika ada.
5. Pengguna dapat memindahkan posisi waktu pemutaran lagu dengan memasukkan nilai waktu dalam satuan waktu, yakni detik saja, menit dan detik, atau jam, menit dan detik.

Selain aturan penggunaan, terdapat beberapa batasan yang perlu diketahui oleh pengguna. Batasan-batasan ini menggambarkan keterbatasan fitur dalam program dan ruang lingkup fungsionalitas yang disediakan.

1. Program hanya dapat mengakses file musik yang berada di dalam folder lokal pengguna. Musik dari sumber eksternal atau online tidak dapat digunakan.
2. Pengguna hanya dapat menambahkan satu lagu ke dalam antrian pada satu waktu. Tidak tersedia fitur untuk menambahkan banyak lagu secara bersamaan.
3. Program hanya akan menjalankan musik dengan ekstensi **.mp3**.
4. Musik atau direktori yang berada di direktori musik tidak boleh memiliki nama yang sama dengan file lain di direktori.
5. Input dari user untuk *search* musik maksimal **1024** karakter.

6. Logging (print dengan prefix INFO) mungkin akan tertimpa karena race condition, Hal ini tidak di handle di aplikasi kami.

1.2 Identifikasi Proses

Program ini memiliki penggunaan utama sebagai berikut.

1. Membuka program.
2. Muncul menu dengan pilihan sebagai berikut
 - a) Lihat Musik

Fitur ini akan menampilkan musik dalam bentuk hirarki direktori yang berada di device pengguna yang berada di folder Music. Contoh tampilan outputnya seperti di bawah ini.

```
Music/  
  Rock/  
    KingSlayer.mp3  
  
  Dangdut/  
    CintaSatuMalam.mp3  
    KeretaMalam.mp3  
    Bergadang.mp3
```

- b) Tambahkan music ke queue.

Fitur ini akan menambahkan lagu yang dipilih ke dalam queue music. berikut adalah contoh proses pemilihan lagu untuk dimasukan ke Queue menggunakan contoh direktori sebelumnya.

```
Rock/  
Dangdut/  
  
> Dangdut  
  
CintaSatuMalam.mp3  
KeretaMalam.mp3  
Bergadang.mp3  
  
> Bergadang.mp3
```

- c) Skip

Memungkinkan musik yang sedang berjalan dilewati dan memulai lagu berikutnya yang di dalam queue jika ada.

- d) Rewind

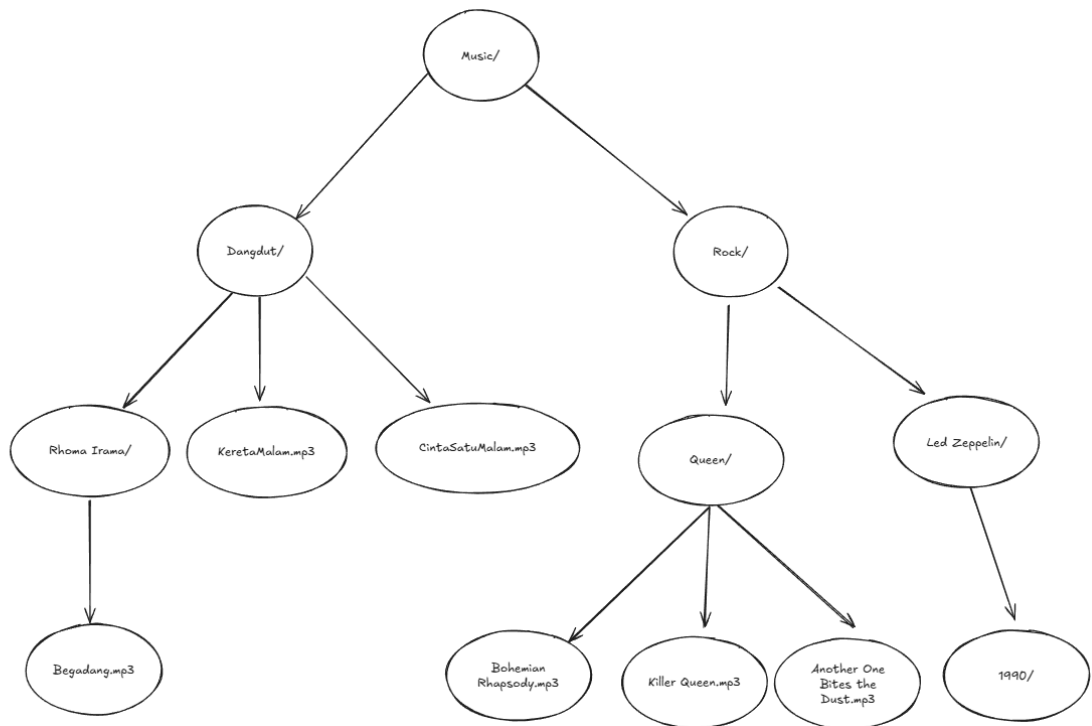
Mengulang musik yang sedang berjalan ke awal.

- e) Seek

Memungkinkan user untuk mencari detik lagu secara spesifik. Contoh dibawah ini akan memindahkan musik ke detik ke-120.

```
Pindah waktu ke detik berapa?  
  
> 120
```

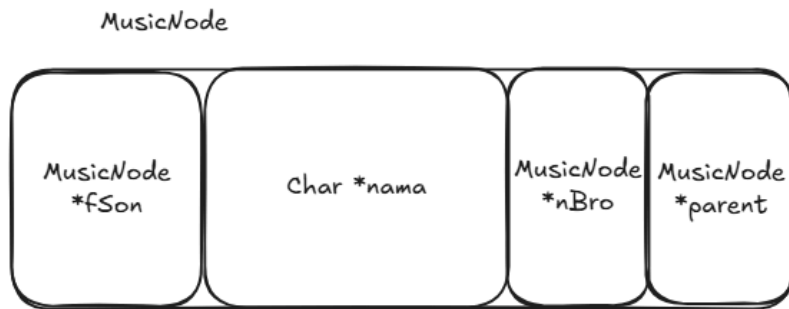
1.3 Identifikasi Data



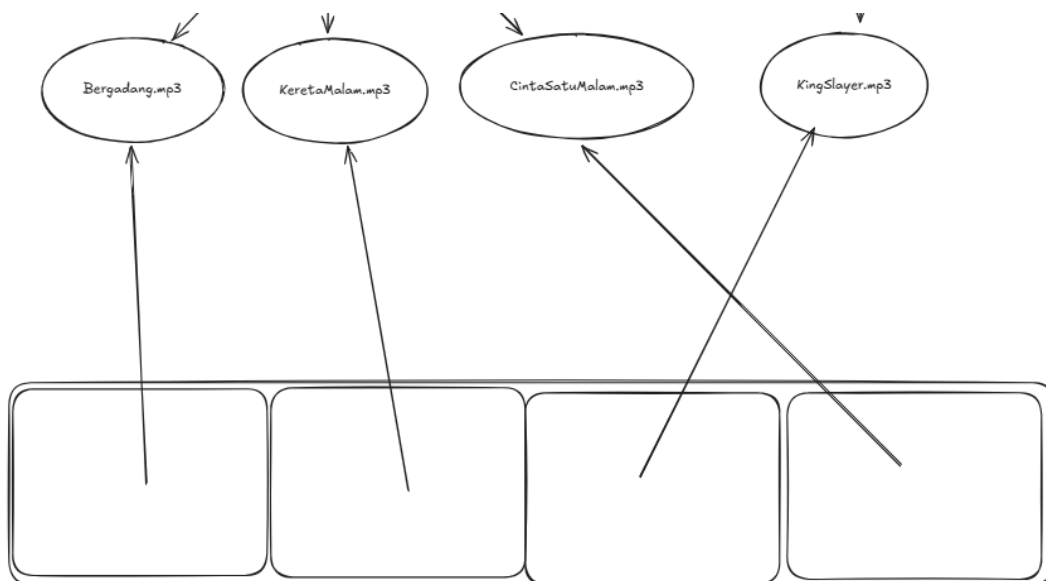
Saat program dijalankan akan dibuat sebuah tree berdasarkan direktori. Gambar di atas adalah contoh visualisasi dari direktori tersebut.

Untuk masing - masing node, diberikan struktur data seperti gambar di bawah ini.

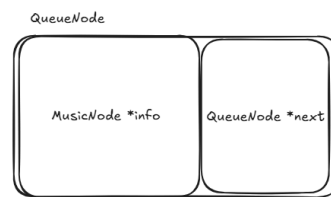
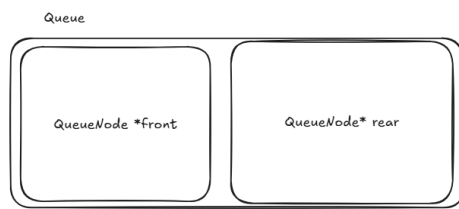
```
type Record MusicNode :  
  name ^char  
  parent ^MusicNode  
  nbrother ^MusicNode  
  fson ^MusicNode
```



Untuk Queue musik akan digunakan sebuah queue yang infonya berisi pointer menuju node musik yang ada di dalam tree



Setiap queue, berisi pointer yang menunjuk ke bagian depan dan belakang queue. Untuk queue-nya sendiri akan diimplementasikan menggunakan linked list dengan struktur info yang merupakan pointer ke musik node dan next



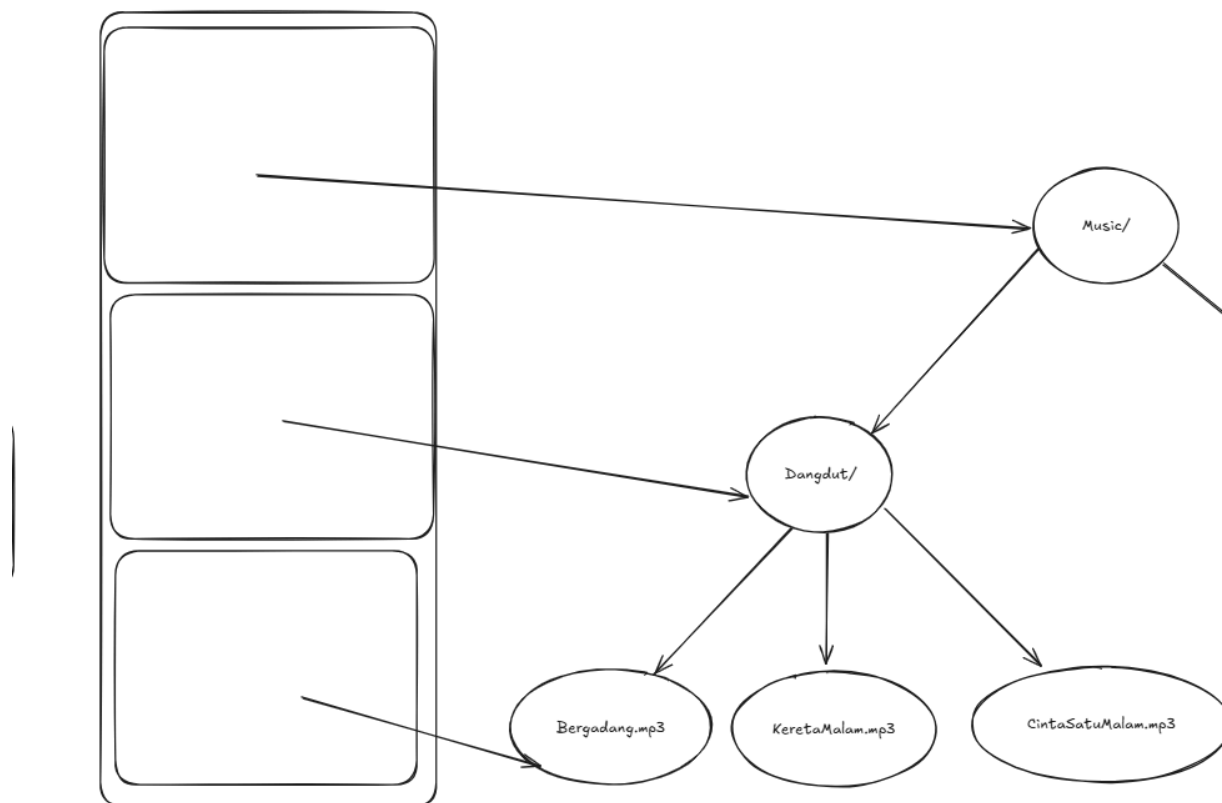
```

type Record Queue :
  front ^QueueNode
end ^QueueNode
  
```

```

type Record Node:
  info ^MusicNode
  next ^Node
  
```

Untuk mendapatkan *full path* dari musik sedangkan queue hanya memiliki pointer ke musik node yang kemungkinan merupakan *leaf* perlu digunakan stack untuk mendapatkan *full path* dari musik.



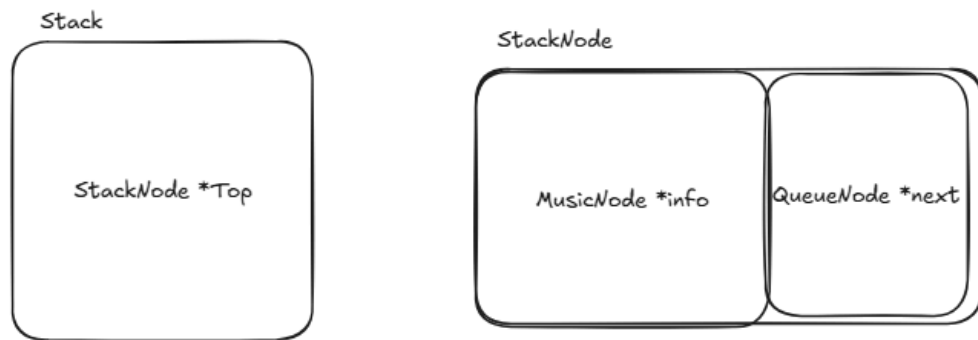
Setiap stack, berisi pointer yang menunjuk ke bagian atas stack. Untuk stack-nya sendiri akan diimplementasikan menggunakan linked list dengan struktur info yang merupakan pointer ke musik node dan next

```

type Record Stack :
  front ^QueueNode
end ^QueueNode
  
```



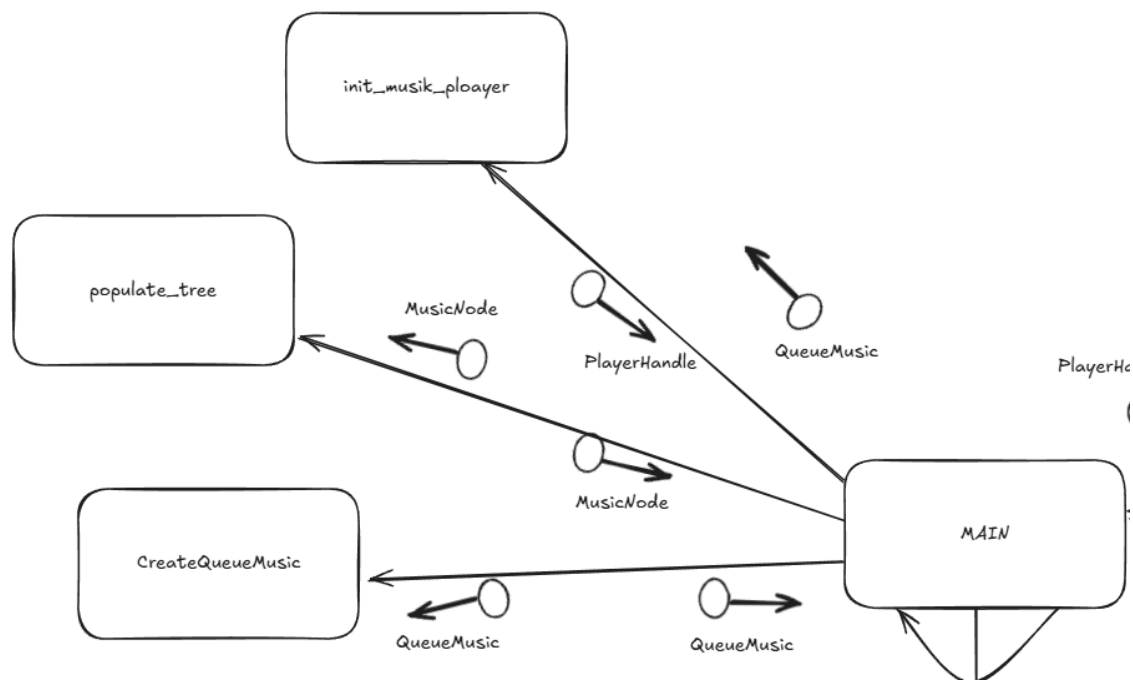
```
type Record Node:
  info ^MusicNode
  next ^Node
```



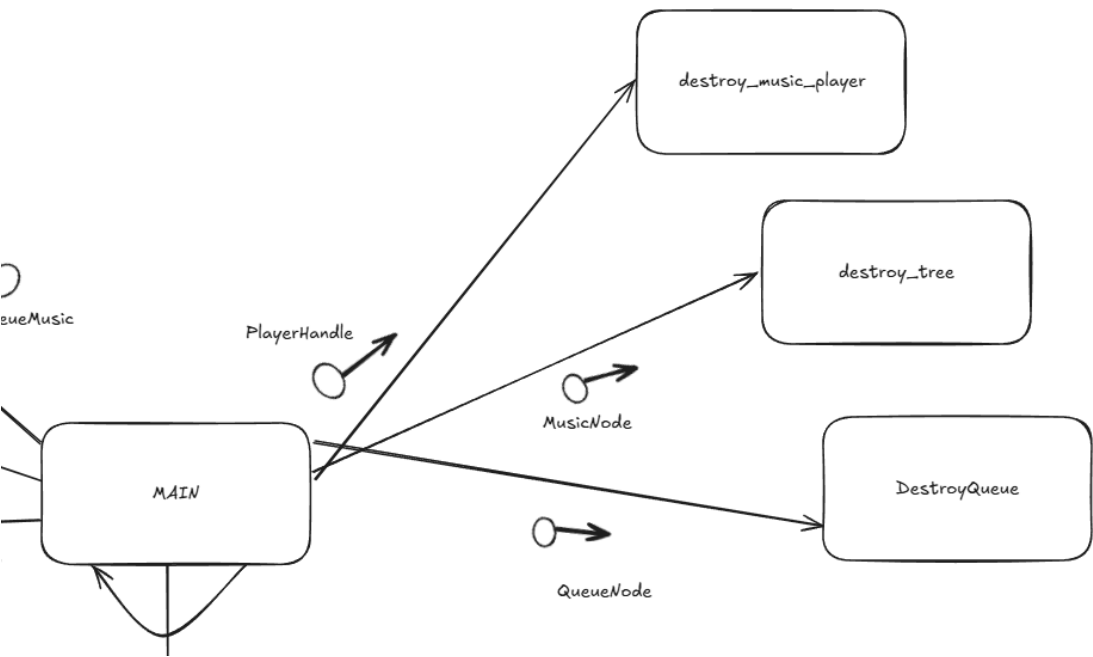
BAB 2 PERANCANGAN PROGRAM

2.1 Chart

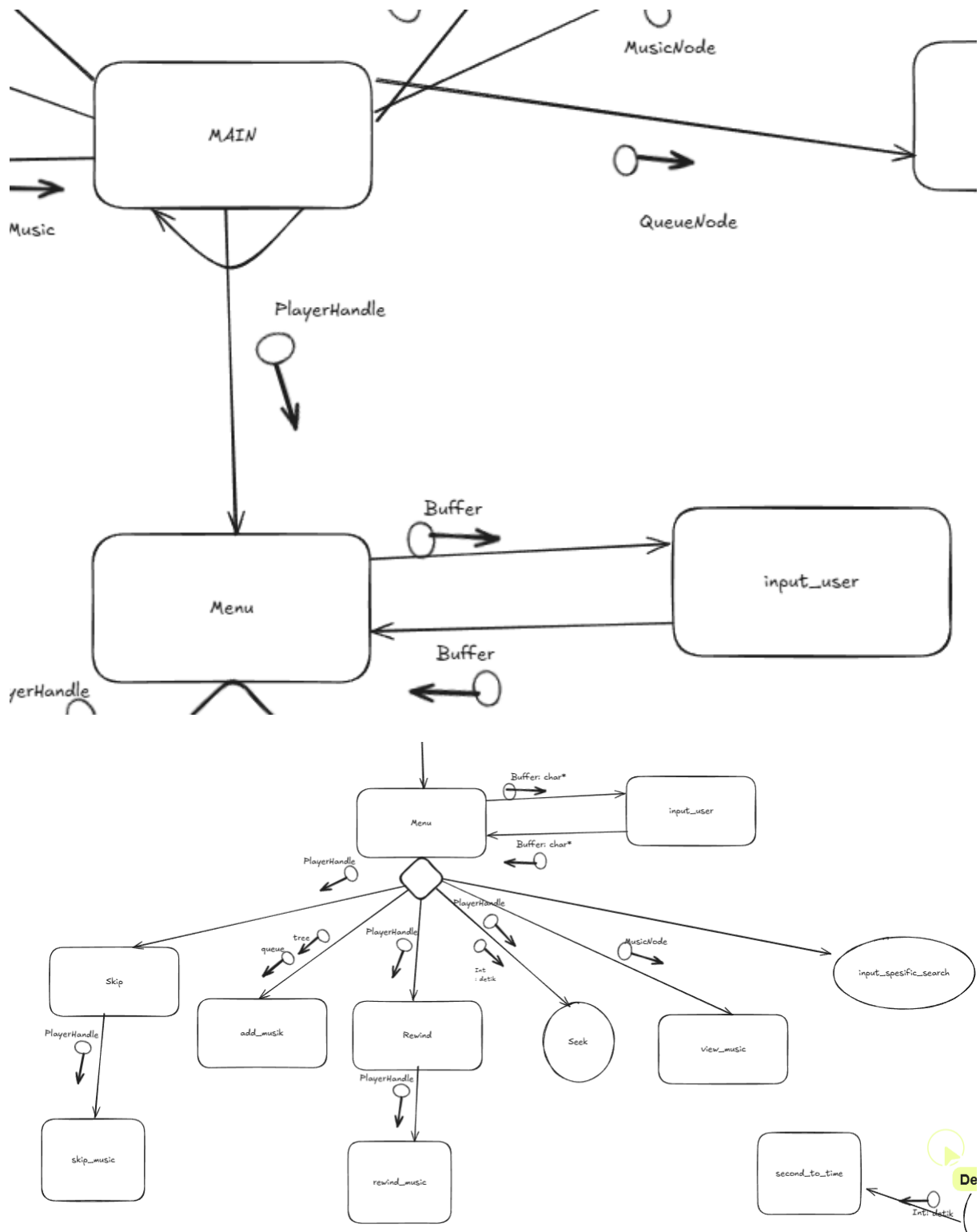
2.1.1 Structured chart 1



2.1.2 Structured chart 2

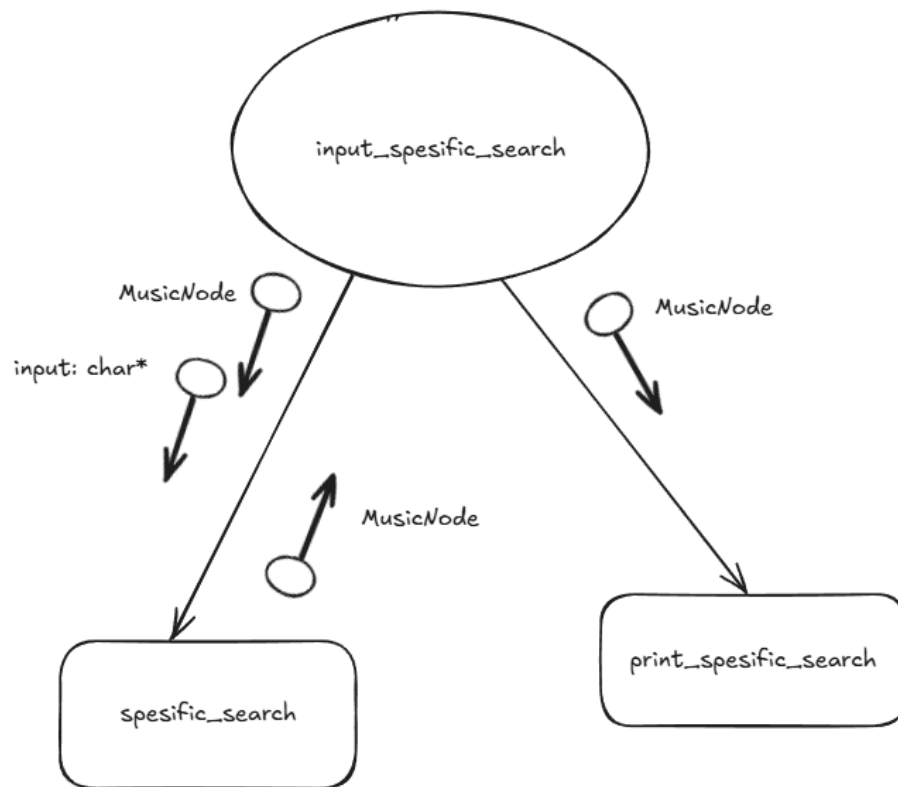


2.1.3 Structured chart 3

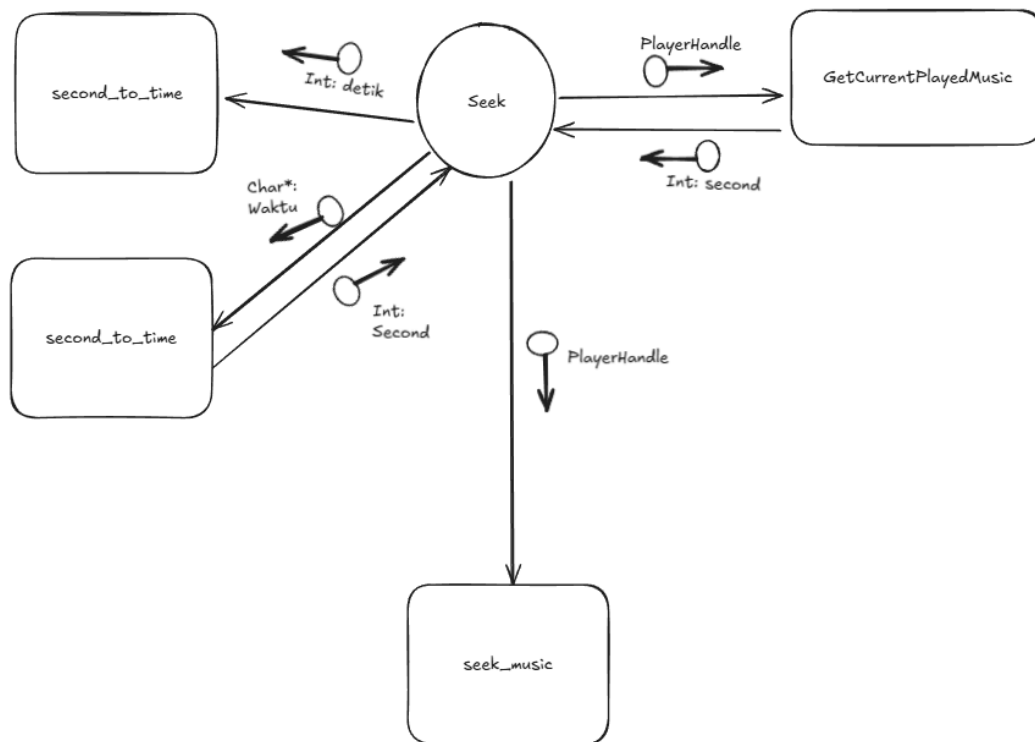


2.1.4 Structured chart 4

to move canvas, hold mouse wheel or spacebar while dragging, or use the hand tool



2.1.4 Structured chart 5



2.2 Tabel Prosedur

2.2.1 Tree

No	Nama Modul	Deskripsi	Jenis	Parameter	Kamus Data (lokal)
1	end_with_mp3 Pembuat: Gilang Aditya	Mengecek apakah nama file diakhiri dengan ekstensi ".mp3".	Function	filename : const char * IS: Diketahui nama file FS: Mengembalikan nilai true jika ada file dengan ekstensi ".mp3"	exit : const char *
2	read_dir_music Pembuat: Gilang Aditya	Membaca isi direktori secara rekursif dan membangun sub-tree untuk node root yang diberikan, menambahkan di-	Procedure	base_path : const char * root : MusicNode* IS: Direktori Musik ada di user FS: Tree ada dengan isi direktori musik user	find_data : WIN32_FIND_D hFind : HANDLE search_path : char[1024] name : const char * full_path : char[1024] child : MusicNode*

		rektori dan file .mp3 sebagai children.			
3	populate_tree Pembuat: Gilang Aditya	Menginisialisasi tree musik utama dan mengisinya dengan data dari direktori musik default pengguna.	Procedure	root : MusicNode** IS: Tree belum ada FS: Tree ada dengan isi direktori musik user	
4	destroy_tree Pembuat: Gilang Aditya	Menghapus semua node dalam tree musik secara rekursif untuk membebaskan memori.	Procedure	root : MusicNode* IS: Tree ada FS: Tree Dihapus (dealokasi)	
5	print_children Pembuat: Gilang Aditya	Mencetak nama-nama dari semua child langsung dari sebuah node parent.	Procedure	parent : MusicNode* IS: Tree sudah terbuat FS: children dari sebuah parent ditampilkan	child : MusicNode*
6	print_tree Pembuat: Gilang Aditya	Mencetak keseluruhan struktur tree musik dengan indentasi (pre-order, fson kemudian nbrother) untuk menunjukkan hierarki.	Procedure	root : MusicNode* height : int IS: Tree sudah terbuat FS: Tree ditampilkan dengan bentuk pre order tapi dari kanan	i : int
7	search_node Pembuat: Gilang Aditya	Mencari sebuah node dalam tree berdasarkan namanya secara rekur-	Function	root : MusicTree target : char* IS: Tree sudah terbuat FS: Node yang dicari dikembalikan	found_in_fson : MusicTree

		sif (pre-order traversal).			
8	input_specific_search Pembuat: Gilang Aditya	Menginput nama node yang ingin dicari	Procedure	root : MusicTree IS: Nama file yang ingin dicari belum diketahui FS: User Mengetik file yang ingin dicari (bisa ketemu bisa tidak)	input : char[1024]
9	specific_search Pembuat: Gilang Aditya	Mencari sebuah node musik spesifik dalam tree berdasarkan namanya secara rekursif (serupa dengan search_node).	Function	root : MusicTree target : char* IS: Tree Musik ada FS: Mengembalikan node music yang dicari	found : MusicNode*
10	print_specific_search Pembuat: Gilang Aditya	Menampilkan path lengkap dari root direktori musik hingga node yang dicari, beserta nama node tersebut dengan indentasi.	Procedure	node : MusicTree IS: Node musik yang dicari ada FS: Menampilkan music yang dicari dan direktorinya	stack : Stack current : MusicNode * full_path : char[2048] mn : MusicNode * level : int temp : MusicNode* i : int
11	add_children Pembuat: Gilang Aditya	Menambahkan sebuah node baru sebagai child dari node parent yang diberikan dalam tree.	Procedure	root : MusicTree info : char* IS: Root diketahui FS: children bertambah (music atau direktori)	new_child : MusicNode* temp : MusicNode*

2.2.2 Player.h

No	Nama Modul	Deskripsi	Jenis	Parameter	Kamus Data (lokal)
1	init_music_player	Menginisialisasi music player dan memulai	Procedure	handle : PlayerHandle * music_queue : QueueMusic * IS: Music player belum diinisialisasi	r : ma_result

	Pembuat: Andi Putra	thread pemutar. 		FS: Music player sudah diinisialisasi	
2	destroy_music_player Pembuat: Andi Putra	Menghentikan engine musik dan menutup thread handle.	Procedure	handle : PlayerHandle * IS: Music player sudah terinisialisasi dan ada isinya FS: Music player hilang, gone, ter-deinisialisasi	
3	get_currently_playing_music_length Pembuat: Andi Putra	Mendapatkan panjang musik yang sedang diputar dalam detik.	Function	handle : PlayerHandle * IS: Music player sudah terinisialisasi FS: mengembalikan panjang dari musik dalam detik. Jika error, mengembalikan -1	outLen : float res : ma_result
4	music_thread Pembuat: Andi Putra	Thread worker yang memproses antrian musik, memuat, memainkan, dan mengontrol musik (rewind, seek, skip).	Function	lpParam : LPVOID IS: Thread terbuat FS: -	handle : PlayerHandle * length : ma_uint64 result : ma_result stack : Stack cursor : MusicTree music_path : char * music_path_len : size_t path_buffer : char * path_buffer_len : size_t path_len : size_t is_loaded : bool
5	rewind_music Pembuat: Andi Putra	Mengirim perintah untuk memutar ulang musik saat ini dari awal.	Procedure	handle : PlayerHandle * IS: Thread terbuat FS: merewind musik dari awal	
6	skip_music Pembuat: Andi Putra	Mengirim perintah untuk melompati musik yang sedang diputar.	Procedure	handle : PlayerHandle * IS: Thread terbuat FS: menskip musik dari awal	
7	seek_music Pembuat: Andi Putra	Mengirim perintah untuk memindahkan posisi pemutar.	Procedure	handle : PlayerHandle * secs : int IS: Thread terbuat FS: seek musik ke detik yang ditentukan	

		taran musik ke detik tertentu.			
--	--	--------------------------------	--	--	--

2.2.3 UI

No	Nama Modul	Deskripsi	Jenis	Parameter	Kamus Data (lokal)
1	menu Pembuat: Raf	Menampilkan antarmuka menu utama kepada pengguna, memungkinkan interaksi dengan berbagai fitur pemutar musik seperti melihat musik, menambah ke antrian, skip, replay, seek, memeriksa antrian, dan mencari musik.	Procedure	tree : MusicNode queue : QueueMusic * handle : PlayerHandle * IS: Terminal kosong FS: Terminal menampilkan pilihan menu dan menanyakan pilihan user	x : int
2	view_music Pembuat: Raf	Member-sihkan layar terminal dan menampilkan seluruh struktur tree musik yang tersedia.	Procedure	tree : MusicNode IS: Tree tidak kosong FS: Tree ditampilkan ke layar	
3	add_music Pembuat: Raf	Memandu pengguna melalui tree musik untuk memilih dan menambahkan file musik (.mp3) ke dalam antrian pemutaran.	Procedure	tree : MusicNode queue : QueueMusic * IS: Queue mungkin kosong FS: Musik dalam queue bertambah	x : char[1024] temp1 : MusicTree temp2 : MusicTree len : int

4	skip Pembuat: Raf	Memanggil fungsi untuk melompati musik yang sedang diputar dan melanjutkan ke musik berikutnya dalam antrian (jika ada).	Procedure	handle : PlayerHandle * IS: Musik dijalankan FS: Musik dihentikan dan lanjut ke music berikutnya di queue jika ada	
5	replay Pembuat: Raf	Memanggil fungsi untuk memutar ulang musik yang sedang berjalan dari awal.	Procedure	handle : PlayerHandle * IS: Musik berjalan FS: Musik diulang dari awal	
6	seek Pembuat: Raff	Memungkinkan pengguna untuk memindahkan posisi pemutaran musik yang sedang berjalan ke durasi (detik) tertentu yang diinput oleh pengguna.	Procedure	handle : PlayerHandle * IS: Musik berjalan FS: Musik berjalan di detik yang ditentukan	second : int detik : int waktu : char[10] i : int

2.2.4 Time

No	Nama Modul	Deskripsi	Jenis	Parameter	Kamus Data (lokal)
1	hour_to_second Pembuat: Raff	Mengonversi nilai jam menjadi total detik dengan mengalikannya dengan 3600.	Function	hour : int IS: Detik diketahui FS: Jam diketahui	
2	minute_to_second	Mengonversi nilai menit	Function	minute : int	

	Pembuat: Raff	menjadi total detik dengan mengalikannya dengan 60.		IS: Menit diketahui FS: Jam diketahui	
3	second_to_hour Pembuat: Raff	Mengonversi total detik (yang diterima melalui pointer) menjadi jam. Nilai yang ditunjuk oleh pointer detik diperbarui menjadi sisa detik setelah konversi. Mengembalikan jumlah jam.	Function	second : int * IS: Menit diketahui FS: Jam diketahui	hour : int
4	second_to_minute Pembuat: Raff	Mengonversi total detik (yang diterima melalui pointer) menjadi menit. Nilai yang ditunjuk oleh pointer detik diperbarui menjadi sisa detik setelah konversi. Mengembalikan jumlah menit.	Function	second : int * IS: Detik diketahui FS: Menit diketahui	minute : int
5	second_to_time Pembuat: Raff	Mengonversi total detik menjadi format waktu (JJ:MM:DD, MM:DD, atau DD) dan mencetaknya ke konsol.	Procedure	second : int IS: Detik diketahui FS: Waktu dengan format jam:menit:detik diketahui	hour : int minute : int

		Menggunakan fungsi second_to_hour dan second_to_minute untuk konversi.			
6	time_to_second Pembuat: Raff	Mengonversi string waktu (dalam format JJ:MM:DD, MM:DD, atau DD) menjadi total detik. Memparsing string untuk jam, menit, dan detik berdasarkan jumlah delimiter ':'. Function		time : char[10] IS: Waktu dengan format jam:menit:detik atau menit:detik atau detik diketahui FS: Detik diketahui	len : int second : int minute : int hour : int totalsec : int colonCount : int i : int

2.3 Algoritma

```

FUNCTION main() -> integer
// kamus
Tree: MusicTree
queue: QueueMusic
handle: PlayerHandle
// algoritma
Tree <- NIL
CreateQueueMusic(address(queue))
populate_tree(address(Tree))
init_music_player(address(handle), address(queue))
print_tree(Tree, 0)

menu(Tree^, address(queue), address(handle))
destroy_tree(Tree)
destroy_queue(address(queue))
destroy_music_player(address(handle))

RETURN 0

ENDFUNCTION

PROCEDURE init_music_player(handle: ^PlayerHandle, music_queue: ^QueueMusic)
// kamus
r: ma_result
// algoritma
handle.music_queue <- music_queue
handle.is_loaded <- false
r <- ma_engine_init(NIL, address(handle.engine))

```

```

IF r != MA_SUCCESS THEN
WRITE "Failed to start engine"
exit(1)
ENDIF
handle.thread_handle <- CreateThread(NIL, 0, music_thread, handle, 0, NIL)
ENDPROCEDURE

FUNCTION music_thread(lpParam: ^void) -> DWORD
// kamus
handle: ^PlayerHandle
length: ma_uint64
result: ma_result
stack: Stack
cursor: MusicTree
music_path: ^char
music_path_len: integer
path_buffer: ^char
path_buffer_len: integer
path_len: integer

// algoritma
handle <- lpParam AS ^PlayerHandle
path_buffer_len <- 0
path_len <- 0
CreateEmpty(address(stack))
handle.is_loaded <- false
music_path <- get_music_folder_path()
music_path_len <- strlen(music_path)

WHILE (true) DO
    IF (NOT is_Empty(handle.music_queue)) AND ((NOT handle.is_loaded) OR (NOT
ma_sound_is_playing(address(handle.sound)))) THEN
        IF handle.is_loaded THEN
            ma_sound_stop(address(handle.sound))
            ma_sound_uninit(address(handle.sound))
        ENDIF
        deQueueMusic(handle.music_queue, address(cursor))
        WRITE "\nINFO: Memutar musik %s\n", cursor.name

        path_buffer_len <- 0
        path_len <- 0
        WHILE cursor.parent != NIL DO
            Push(address(stack), cursor)
            path_buffer_len <- path_buffer_len + strlen(cursor.name) + 1
            cursor <- cursor.parent
        ENDWHILE
        path_buffer_len <- path_buffer_len + music_path_len + 1

        path_buffer <- malloc(path_buffer_len)
        strcpy(path_buffer, music_path)
        path_len <- path_len + music_path_len
        path_buffer[path_len] <- '/'
        path_len <- path_len + 1

        WHILE NOT IsEmpty(stack) DO
            Pop(address(stack), address(cursor))

```

```

        strcpy(address(path_buffer[path_len]), cursor.name)
        path_len <- path_len + strlen(cursor.name)
        path_buffer[path_len] <- '/'
        path_len <- path_len + 1
    ENDWHILE
    path_buffer[path_len - 1] <- '\0'

    result <- ma_sound_init_from_file(address(handle.engine), path_buffer, 0, NIL,
NIL, address(handle.sound))
    IF result != MA_SUCCESS THEN
        RETURN -1
    ENDIF

    ma_sound_start(address(handle.sound))
    ma_sound_get_length_in_pcm_frames(address(handle.sound), address(length))
    ma_sound_get_length_in_seconds(address(handle.sound),
address(handle._current_music_time_in_secs))
    handle.is_loaded <- true
    ENDIF

    IF handle._command != NONE THEN
        SWITCH handle._command
            CASE NONE:
                // Should be unreachable, do nothing
                BREAK
            CASE REWIND:
                ma_sound_seek_to_pcm_frame(address(handle.sound), 0)
                WRITE "\nINFO: Musik di rewind\n"
                BREAK
            CASE SEEK:
                ma_sound_seek_to_second(address(handle.sound), handle._command_args[0])
                WRITE "\nINFO: Musik di seek ke: "
                second_to_time(handle._command_args[0])
                BREAK
            CASE SKIP:
                ma_sound_seek_to_pcm_frame(address(handle.sound), length)
                ma_sound_stop(address(handle.sound))
                WRITE "\nINFO: Musik di skip\n"
                BREAK
            DEFAULT:
                ASSERT(false)
        ENDSWITCH
        handle._command <- NONE
    ENDIF

    Sleep(16)
ENDWHILE

ENDFUNCTION

FUNCTION get_currently_player_music_length(handle: ^PlayerHandle) -> integer
// kamus
// algoritma
IF NOT handle.is_loaded THEN
RETURN -1

```

```

ENDIF
IF NOT ma_sound_is_playing(address(handle.sound)) THEN
RETURN -1
ENDIF

RETURN floor(handle._current_music_time_in_secs)

ENDFUNCTION

PROCEDURE destroy_music_player(handle: ^PlayerHandle)
// kamus
// algoritma
ma_engine_uninit(address(handle.engine))
CloseHandle(handle.thread_handle)
ENDPROCEDURE

PROCEDURE rewind_music(handle: ^PlayerHandle)
// kamus
// algoritma
handle._command <- REWIND
ENDPROCEDURE

PROCEDURE skip_music(handle: ^PlayerHandle)
// kamus
// algoritma
handle._command <- SKIP
ENDPROCEDURE

PROCEDURE seek_music(handle: ^PlayerHandle, secs: integer)
// kamus
// algoritma
handle._command <- SEEK
handle._command_args[0] <- secs
ENDPROCEDURE

FUNCTION end_with_mp3(filename: ^char) -> boolean
// kamus
exit_str: ^char
// algoritma
exit_str <- strrchr(filename, '.')
RETURN exit_str != NIL AND _stricmp(exit_str, ".mp3") = 0
ENDFUNCTION

PROCEDURE read_dir_music(base_path: ^char, root: ^MusicNode)
// kamus
find_data: WIN32_FIND_DATA
hFind: HANDLE
search_path: char[1024]
name: ^char
full_path: char[1024]
child: ^MusicNode
// algoritma
sprintf(search_path, 1024, "%s\\*", base_path)

hFind <- FindFirstFile(search_path, address(find_data))

```

```

IF hFind = INVALID_HANDLE_VALUE THEN
    WRITE "Gagal membuka direktori: %s\n", base_path
    RETURN
ENDIF

LOOP // Emulates do-while loop
    name <- find_data.cFileName

    IF NOT (strcmp(name, ".") = 0 OR strcmp(name, "..") = 0) THEN
        snprintf(full_path, 1024, "%s\\%s", base_path, name)

        IF (find_data.dwFileAttributes (BITWISEAND) FILE_ATTRIBUTE_DIRECTORY) THEN
            add_children(root, strdup(name))

            child <- root.fson
            WHILE child.nbrother != NIL DO
                child <- child.nbrother
            ENDWHILE
            read_dir_music(full_path, child)
        ELSEIF end_with_mp3(name) THEN
            add_children(root, strdup(name))
        ENDIF
    ENDIF
UNTIL NOT FindNextFile(hFind, address(find_data))
ENDLOOP

FindClose(hFind)

ENDPROCEDURE

PROCEDURE populate_tree(root_ptr: ^^MusicNode)
// kamus
// algoritma
root_ptr^ <- malloc(sizeof MusicNode)
root_ptr^.fson <- NIL
root_ptr^.nbrother <- NIL
root_ptr^.parent <- NIL
root_ptr^.name <- strdup("")
read_dir_music(get_music_folder_path(), root_ptr^)
ENDPROCEDURE

PROCEDURE print_tree(root: ^MusicNode, height: integer)
// kamus
i: integer
// algoritma
IF root = NIL THEN
    RETURN
ELSE
    FOR i FROM 0 TO height - 1 DO
        WRITE " "
    ENDFOR
    WRITE root.name, NEWLINE
    print_tree(root.fson, height + 2)
    print_tree(root.nbrother, height)

```



```

ENDIF
ENDPROCEDURE

FUNCTION search_node(root: ^MusicNode, target: ^char) -> ^MusicNode
// kamus
found_in_fson: ^MusicNode
// algoritma
IF root = NIL THEN
RETURN NIL
ENDIF

IF strcmp(root.name, target) = 0 THEN
RETURN root
ENDIF

found_in_fson <- search_node(root.fson, target)
IF found_in_fson != NIL THEN
RETURN found_in_fson
ENDIF

RETURN search_node(root.nbrother, target)

ENDFUNCTION

PROCEDURE input_specific_search(root: ^MusicNode)
// kamus
input_str: char[1024]
found: ^MusicNode
idx: integer
// algoritma
WRITE "Masukkan Nama file: "
READ input_str
idx <- strcspn(input_str, "\n")
input_str[idx] <- '\0'
found <- specific_search(root, input_str)
IF found != NIL THEN
print_specific_search(found)
ELSE
WRITE "Musik/Direktori tidak ditemukan"
ENDIF
ENDPROCEDURE

FUNCTION specific_search(root: ^MusicNode, target: ^char) -> ^MusicNode
// kamus
found: ^MusicNode
// algoritma
IF root = NIL THEN
RETURN NIL
ENDIF

IF strcmp(root.name, target) = 0 THEN
RETURN root
ENDIF

found <- specific_search(root.fson, target)

```

```

IF found != NIL THEN
    RETURN found
ENDIF

RETURN specific_search(root.nbrother, target)

ENDFUNCTION

PROCEDURE print_specific_search(node: ^MusicNode)
// kamus
stack: Stack
current: ^MusicNode
full_path: char[2048]
mn: ^MusicNode
level: integer
temp: ^MusicNode
i: integer
// algoritma
WRITE "Musik/Direktori yang kamu cari yaitu: %s\n", node.name
WRITE "Musik/Direktori tersebut berada di:\n"
CreateEmpty(address(stack))

current <- node

WHILE current != NIL DO
    Push(address(stack), current)
    current <- current.parent
ENDWHILE

snprintf(full_path, 2048, "%s", get_music_folder_path())

WRITE "Music"
WHILE NOT IsEmpty(stack) DO
    Pop(address(stack), address(mn))

    level <- 0
    temp <- mn
    WHILE temp != NIL DO
        level <- level + 1
        temp <- temp.parent
    ENDWHILE
    FOR i FROM 0 TO (level * 2) - 1 DO
        WRITE " "
    ENDFOR
    WRITE mn.name, NEWLINE
ENDWHILE
WRITE NEWLINE

ENDPROCEDURE

PROCEDURE destroy_tree(root: ^MusicNode)
// kamus
// algoritma
IF root = NIL THEN

```

```

RETURN
ELSE
destroy_tree(root.fson)
destroy_tree(root.nbrother)
free(root)
ENDIF
ENDPROCEDURE

PROCEDURE print_children(parent: ^MusicNode)
// kamus
child: ^MusicNode
// algoritma
IF parent != NIL THEN
child <- parent.fson
IF child != NIL THEN
WRITE child.name, NEWLINE
WHILE child.nbrother != NIL DO
child <- child.nbrother
WRITE child.name, NEWLINE
ENDWHILE
ELSE
WRITE "Parent tidak memiliki anak\n"
ENDIF
ENDIF
ENDPROCEDURE

PROCEDURE add_children(root: ^MusicNode, ingfo: ^char)
// kamus
new_child: ^MusicNode
temp: ^MusicNode
// algoritma
IF root != NIL THEN
new_child <- malloc(sizeof MusicNode)
new_child.name <- ingfo
new_child.fson <- NIL
new_child.nbrother <- NIL
new_child.parent <- root
IF root.fson = NIL THEN
root.fson <- new_child
ELSE
temp <- root.fson
WHILE temp.nbrother != NIL DO
temp <- temp.nbrother
ENDWHILE
temp.nbrother <- new_child
ENDIF
ENDIF
ENDPROCEDURE

PROCEDURE menu(tree: MusicNode, queue: ^QueueMusic, handle: ^PlayerHandle)
// kamus
x: integer
// algoritma
WHILE (true) DO
WRITE "1. Lihat Musik\n2. Tambahkan musik ke queue\n3. Skip\n4. Replay\n5. Seek\n6.
Check Queue\n7. Search Musik\n8. Keluar\nMasukkan pilihan: "

```

```

READ x
// getch() is typically for consuming newline, omitted in pseudocode
SWITCH x
CASE 1:
view_music(tree)
BREAK
CASE 2:
add_music(tree, queue)
BREAK
CASE 3:
skip(handle)
BREAK
CASE 4:
replay(handle)
BREAK
CASE 5:
seek(handle)
BREAK
CASE 6:
PrintQueueMusic(queue^)
getch()
BREAK
CASE 7:
input_specific_search(address(tree))
getch()
BREAK
CASE 8:
return
DEFAULT:
BREAK
ENDSWITCH
system("cls")
ENDWHILE
ENDPROCEDURE

```

```

PROCEDURE view_music(tree: MusicNode)
// kamus
// algoritma
system("cls")
WRITE "Music"
print_tree(address(tree), 0)
getch()
ENDPROCEDURE

```

```

PROCEDURE add_music(tree: MusicNode, queue: ^QueueMusic)
// kamus
x: char[1024]
temp1, temp2: MusicTree
len: integer
// algoritma
temp1 <- address(tree)
temp2 <- temp1
WHILE (true) DO
print_children(temp1)
WRITE "Masukkan nama musik atau folder: "
READ x

```

```

len <- strlen(x)
x[len - 1] <- '\0'
temp1 <- search_node(temp1, x)
// Example was temp1 = search_node(temp1, x) -> temp1 <- search_node(temp1, x)
// The example's final search_node(address(tree),x) had a semicolon.
// This implies standalone calls get semicolons.
IF temp1 = NIL THEN
WRITE "Tidak ada file/direktori tersebut\n"
temp1 <- temp2
getch()
continue
ENDIF

IF len >= 5 THEN
    IF strcmp(address(x[len - 5]), ".mp3") = 0 THEN
        EnQueueMusic(queue, temp1)
        WRITE "Lagu %s berhasil dimasukkan\n", temp1->name
        getch()
        return
    ELSE
        temp2 <- temp1
    ENDIF
ENDIF
system("cls")
ENDWHILE
search_node(address(tree), x)

```

ENDPROCEDURE

```

PROCEDURE skip(handle: ^PlayerHandle)
// kamus
// algoritma
skip_music(handle)
ENDPROCEDURE

```

```

PROCEDURE replay(handle: ^PlayerHandle)
// kamus
// algoritma
rewind_music(handle)
ENDPROCEDURE

```

```

PROCEDURE seek(handle: ^PlayerHandle)
// kamus
second: integer
detik: integer
waktu: char[10]
i: integer
// algoritma
second <- get_currently_player_music_length(handle)
detik <- -1

```

```

IF second = -1 THEN
    WRITE "\nTidak ada musik yang sedang dijalankan\n"
    getch()
    return

```

ENDIF

loop:

WHILE (detik < 0) OR (detik > second) DO

system("cls")

WRITE "Panjang lagu saat ini: "

second_to_time(second)

WRITE "Contoh waktu seek:\n12 (detik saja)\n2:52 (menit:detik)\n1:53:23 (jam:menit:detik)\n"

WRITE "Masukkan detik ke berapa: "

READ waktu

i <- 0

WHILE waktu[i] != '\0' DO

IF (NOT isdigit(waktu[i])) AND (waktu[i] != ':') THEN

WRITE "Hanya masukkan angka atau ':'\n"

getch()

GOTO loop

ENDIF

i <- i + 1

ENDWHILE

detik <- time_to_second(waktu)

IF detik > second THEN

WRITE "Waktu yang diinputkan melebihi waktu lagu\n"

getch()

continue

ENDIF

ENDWHILE

seek_music(handle, detik)

ENDPROCEDURE

FUNCTION hour_to_second(hour: integer) -> integer

// kamus

// algoritma

RETURN hour * 3600

ENDFUNCTION

FUNCTION minute_to_second(minute: integer) -> integer

// kamus

// algoritma

RETURN minute * 60

ENDFUNCTION

FUNCTION second_to_hour(second: ^integer) -> integer

// kamus

hour: integer

// algoritma

hour <- 0

hour <- second^ / 3600

second^ <- second^ MOD 3600

RETURN hour

ENDFUNCTION

FUNCTION second_to_minute(second: ^integer) -> integer

// kamus

```

minute: integer
// algoritma
minute <- 0
minute <- second^ / 60
second^ <- second^ MOD 60
RETURN minute
ENDFUNCTION

PROCEDURE second_to_time(second_param: integer)
// kamus
hour: integer
minute: integer
second_local: integer
// algoritma
second_local <- second_param
hour <- second_to_hour(address(second_local))
minute <- second_to_minute(address(second_local))
IF hour != 0 THEN
WRITE "%02d:%02d:%02d\n", hour, minute, second_local
ELSEIF minute != 0 THEN
WRITE "%02d:%02d\n", minute, second_local
ELSE
WRITE "%02d\n", second_local
ENDIF
ENDPROCEDURE

FUNCTION time_to_second(time: char[10]) -> integer
// kamus
len: integer
second: integer
minute: integer
hour: integer
totalsec: integer
colonCount: integer
i: integer
// algoritma
len <- strlen(time)
totalsec <- 0
colonCount <- 0
FOR i FROM 0 TO len - 1 DO
IF time[i] = ':' THEN
colonCount <- colonCount + 1
ENDIF
ENDFOR

IF colonCount = 2 THEN
sscanf(time, "%d:%d:%d", address(hour), address(minute), address(second))
totalsec <- totalsec + hour_to_second(hour)
totalsec <- totalsec + minute_to_second(minute)
totalsec <- totalsec + second
ELSEIF colonCount = 1 THEN
sscanf(time, "%d:%d", address(minute), address(second))
totalsec <- totalsec + minute_to_second(minute)
totalsec <- totalsec + second
ELSE
sscanf(time, "%d", address(second))

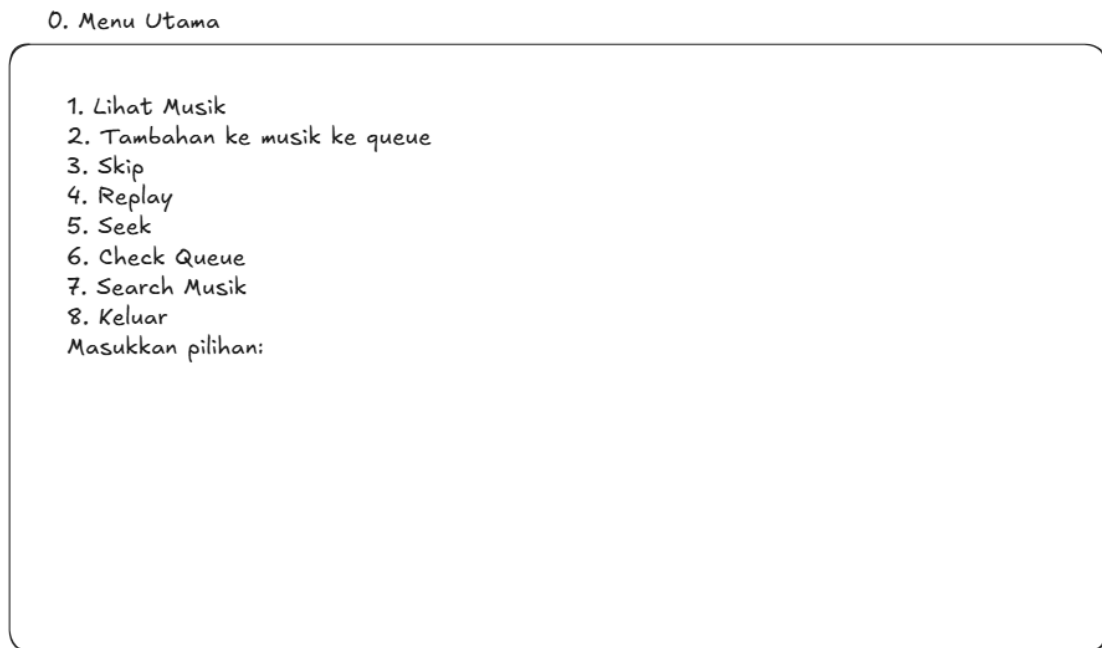
```

```
        totalsec <- totalsec + second
ENDIF

RETURN totalsec

ENDFUNCTION
```

2.4 Perancangan Tampilan (Output)



Tampilan diatas adalah tampilan yang akan muncul saat awal masuk program

1. Lihat Musik

```
Music/  
  Rock/  
    KingSlayer/mp.3  
  
Dangdut/  
  CintaSatuMalam.mp3  
  KeretaMalam.mp3  
  Bergadang.mp3
```

Tampilan diatas adalah tampilan yang akan muncul pada memasuki menu “Lihat Musik”

2. Masukkan Music ke Queue

```
Rock/  
Dangdut/  
  
> Dangdut  
  
CintaSatuMalam.mp3  
KeretaMalam.mp3  
Bergadang.mp3  
  
> Bergadang.mp3
```

Tampilan diatas adalah tampilan yang akan muncul pada memasuki menu “Tambahkan musik ke queue”. Tampilan ini akan muncul sampai pengguna menginputkan sebuah file.

5. Seek

Panjang lagu saat ini: 03:41
Contoh waktu seek:
12 (detik saja)
2:52 (menit:detik)
1:53:23(jam:menit:detik)
Masukkan detik ke berapa:
> 02:00

Tampilan diatas adalah tampilan yang akan muncul saat pengguna memasuki menu "Seek".

6. Check Queue

1. Lihat Musik
2. Tambahan ke musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan:
> 6

KingSlayer.mp3 -> CintaSatuMalam.mp3 -> KeretaMalam.mp3 ->

Tampilan diatas adalah tampilan yang akan muncul jika ingin melihat musik pada antrian.

7. Search Music

1. Lihat Musik
2. Tambahan ke musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar

Masukkan pilihan:

> 7

Masukkan Nama file:

> KingSlayer.mp3

Musik/direktori yang kamu cari yaitu : KingSlayer.mp3

Musik/direktori Tersebut berada di:

Music/

Rock/

KingSlayer.mp3

Tampilan diatas adalah tampilan yang akan muncul ketika fila yang dicari ditemukan dalam direktori user menggunakan fitur *search*

BAB 3 HASIL AKHIR PROGRAM

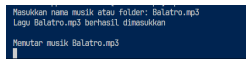
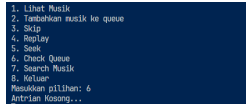
3.1. Pembahasan Hasil Implementasi

1. Fitur Memutar Musik

- Memutar lagu dari direktori lokal pengguna

```
Masukkan nama musik atau folder: Balatro.mp3
Lagu Balatro.mp3 berhasil dimasukkan

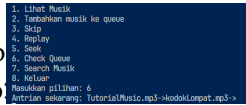
Memutar musik Balatro.mp3
█
```

No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	queue = [balatro.mp3]	balatro.mp3 mulai diputar	balatro.mp3 mulai diputar		PASS
2	queue = []	tidak ada lagu yang diputar	tidak ada lagu yang diputar		PASS

2. Fitur Membuat Playing Queue

- Menambahkan lagu ke antrian untuk diputar

```
1. Lihat Musik
2. Tambahkan musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan: 6
Antrian sekarang: TutorialMusic.mp3->kodokLompat.mp3->
```

No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	input = Bergadang.mp3 TutorialMusic.mp3 kodokLompat.mp3	Bergadang.mp3, TutorialMusic.mp3, kodokLompat.mp3 ditambahkan ke queue	Bergadang.mp3, TutorialMusic.mp3, kodokLompat.mp3 ditambahkan ke queue		PASS

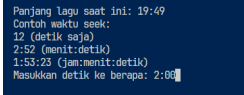
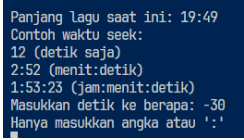
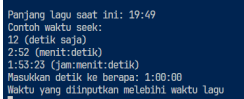
3. Fitur Seek Waktu Musik

- Melompat ke waktu tertentu dalam lagu

```

Panjang lagu saat ini: 19:49
Contoh waktu seek:
12 (detik saja)
2:52 (menit:detik)
1:53:23 (jam:menit:detik)
Masukkan detik ke berapa: 2:00

```

No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	input = 2:00	waktu pemutaran lagu berpindah ke detik 120	waktu pemutaran lagu berpindah ke detik 120		PASS
2	input = -30	hanya masukan angka atau ':'	hanya masukan angka atau ':'		PASS
3	input = 1:00:00 untuk lagu 1:00	Waktu yang diinputkan melebihi waktu lagu	Waktu yang diinputkan melebihi waktu lagu		PASS

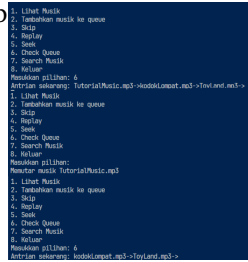
4. Fitur Skip Musik

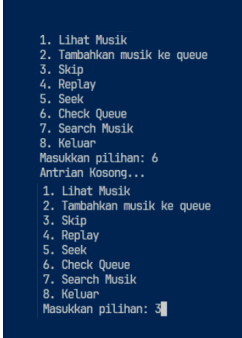
- Melompati lagu yang sedang diputar

```

1. Lihat Musik
2. Tambahkan musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan: 6
Antrian sekarang: TutorialMusic.mp3->kodokLompat.mp3->ToyLand.mp3->
1. Lihat Musik
2. Tambahkan musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan:
Memutar musik TutorialMusic.mp3
1. Lihat Musik
2. Tambahkan musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan: 6
Antrian sekarang: kodokLompat.mp3->ToyLand.mp3->

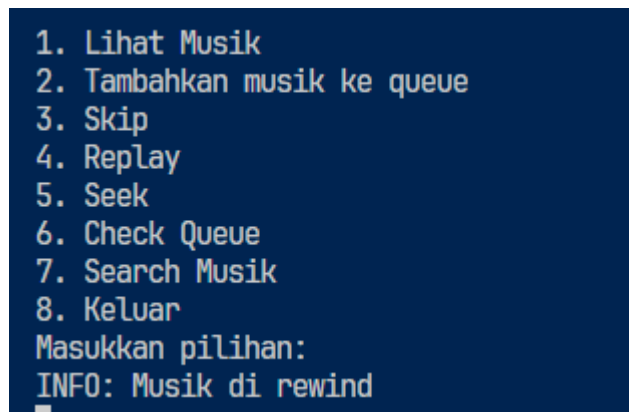
```

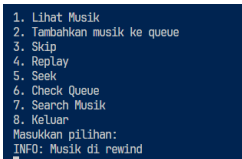
No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	queue = [TutorialMusic.mp3, kodokLompat.mp3, ToyLand.mp3], sekarang = Balatro.mp3	TutorialMusic.mp3 mulai diputar	TutorialMusic.mp3 mulai diputar		PASS

2	queue = [], sekarang = WordTutorialMusik.mp3	queue kosong, musik berhenti	queue kosong, musik berhenti		PASS
---	--	------------------------------	------------------------------	---	------

5. Fitur Rewind Musik

- Mengulang lagu dari awal



No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	lagu = Bergadang.mp3, waktu = 2:30	lagu kembali ke menit 0:00	lagu kembali ke menit 0:00		PASS

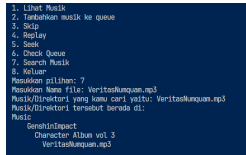
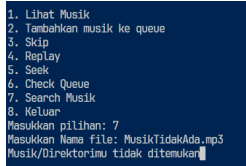
6. Fitur Search Musik

- Mencari lagu di direktori musik

```

1. Lihat Musik
2. Tambahkan musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan: 7
Masukkan Nama file: VeritasNumquam.mp3
Musik/Direktori yang kamu cari yaitu: VeritasNumquam.mp3
Musik/Direktori tersebut berada di:
Music
  GenshinImpact
    Character Album vol 3
      VeritasNumquam.mp3

```

No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	KeretaMalam	lagu ditemukan dalam folder Dangdut	lagu ditemukan dalam folder Dangdut		PASS
2	MusikTidakAda.mp3	hasil tidak ditemukan	hasil tidak ditemukan		PASS

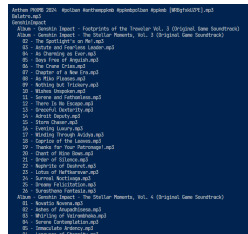
7. Fitur Lihat List Musik

- Menampilkan daftar musik dalam folder lokal


```

Anthem PKKMB 2024 #polban #antheppkmb #ppkmbpolban #ppkmb [WRBgfxkUZPE].mp3
Balatro.mp3
GenshinImpact
Album - Genshin Impact - Footprints of the Traveler Vol. 3 (Original Game Soundtrack)
Album - Genshin Impact - The Stellar Moments, Vol. 3 (Original Game Soundtrack)
02 - The Spotlight's on Me!.mp3
03 - Astute and Fearless Leader.mp3
04 - As Charming as Ever.mp3
05 - Days Free of Anguish.mp3
06 - The Crane Cries.mp3
07 - Chapter of a New Era.mp3
08 - As Miko Pleases.mp3
09 - Nothing but Trickery.mp3
10 - Wishes Unspoken.mp3
11 - Serene and Fathomless.mp3
12 - There Is No Escape.mp3
13 - Graceful Dexterity.mp3
14 - Adroit Deputy.mp3
15 - Storm Chaser.mp3
16 - Evening Luxury.mp3
17 - Winding Through Avidya.mp3
18 - Caprice of the Leaves.mp3
19 - Thanks for Your Patronage!.mp3
20 - Chant of Nine Bows.mp3
21 - Order of Silence.mp3
22 - Nephrite of Deshret.mp3
23 - Lotus of Haftkarsvar.mp3
24 - Surreal Noctivaga.mp3
25 - Dreamy Felicitation.mp3
26 - Surasthana Fantasia.mp3
Album - Genshin Impact - The Stellar Moments, Vol. 4 (Original Game Soundtrack)
01 - Novatio Novena.mp3
02 - Ashes of Anupadhisesa.mp3
03 - Whirling of Vairambhaka.mp3
04 - Serene Contemplation.mp3
05 - Immaculate Ardency.mp3
06 - Language of Eternity.mp3

```

No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1		struktur direktori lagu ditampilkan dengan indentasi	struktur direktori lagu ditampilkan dengan indentasi		PASS

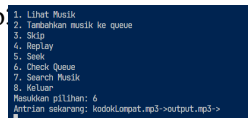
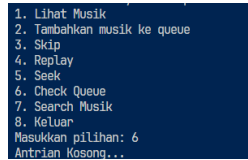
8. Fitur Cek Queue Musik

- Menampilkan isi antrian lagu saat ini

```

1. Lihat Musik
2. Tambahkan musik ke queue
3. Skip
4. Replay
5. Seek
6. Check Queue
7. Search Musik
8. Keluar
Masukkan pilihan: 6
Antrian sekarang: kodokLompat.mp3->output.mp3->

```

No	Data Input	Hasil yang diharapkan	Hasil Keluaran Program	Screen Capture	Hasil Pengujian
1	queue = [kodokLompat.mp3, output.mp3]	kodokLompat.mp3, output.mp3 di- tampilkan	kodokLompat.mp3 output.mp3 di- tampilkan		PASS
2	queue = []	pesan queue kosong	pesan queue kosong		PASS

BAB 4 KESIMPULAN

Program music player bernama **Alpenlify** telah berhasil dikembangkan sebagai bentuk implementasi dari konsep-konsep Struktur Data dan Algoritma. Program ini telah mampu menjalankan seluruh fitur utama yang telah dirancang sebagaimana dijelaskan pada BAB I, meskipun dengan beberapa batasan teknis yang disepakati sejak awal pengembangan.

Secara umum, seluruh fitur fungsional berhasil diimplementasikan dengan baik, antara lain:

- Pemutaran musik dari direktori lokal pengguna.
- Penambahan lagu ke dalam playing queue dan pemutaran otomatis dari urutan pertama.
- Kemampuan untuk melakukan seek ke waktu tertentu dalam lagu menggunakan input waktu dalam berbagai format (detik, menit:detik, atau jam:menit:detik).
- Fitur skip untuk melompati lagu ke antrian berikutnya.
- Fitur rewind lagu yang sedang diputar tanpa menghapus lagu dari antrian.
- Tampilan daftar lagu dari folder musik secara keseluruhan.
- Pemeriksaan dan penampilan antrian lagu secara real-time.

Namun, pencapaian tersebut tetap berada dalam batasan-batasan yang telah ditetapkan, seperti:

- Program hanya dapat membaca file musik dari folder lokal, dan tidak mendukung sumber eksternal atau daring.
- Penambahan lagu ke antrian dilakukan satu per satu, tidak dapat memasukkan lebih dari 2 musik ke antrian secara langsung.
- Program hanya mendukung file dengan ekstensi .mp3.
- Nama file musik di direktori tidak boleh duplikat untuk menghindari konflik identifikasi file.
- Panjang input nama file dari pengguna untuk *search* dibatasi maksimal 1024 karakter.
- Logging (print dengan prefix INFO) mungkin akan tertimpa karena race condition.

Dengan demikian, **Alpenlify** telah berhasil memenuhi seluruh spesifikasi dasar yang telah dirancang pada tahap awal pengembangan dan berfungsi sesuai dengan ekspektasi, dengan tetap memperhatikan ruang lingkup batasan yang telah ditetapkan. Seluruh fitur utama yang direncanakan telah diimplementasikan secara fungsional dan berjalan dengan baik.

Meskipun demikian, program ini masih memiliki beberapa keterbatasan teknis, antara lain hanya dapat membaca file musik lokal dengan ekstensi .mp3, pembatasan panjang input dari pengguna, proses penambahan lagu ke dalam antrian yang hanya dapat dilakukan satu per satu, serta larangan penggunaan nama file atau direktori musik yang sama untuk menghindari konflik.

DAFTAR PUSTAKA

Berisi daftar referensi yang dijadikan acuan dalam pembuatan program ini.

<https://stackoverflow.com/questions/1981459/using-threads-in-c-on-windows-simple-example>

<https://learn.microsoft.com/en-us/windows/win32/procthread/creating-threads?redirectedfrom=MSDN>

<https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread>

<https://github.com/mackron/miniaudio/tree/master>

<https://www.geeksforgeeks.org/strchr-in-c/>

<https://www.geeksforgeeks.org/how-to-read-data-using-sscanf-in-c/>

<https://www.ibm.com/docs/en/i/7.1.0?topic=functions-stricmp-compare-strings-without-case-sensitivity>

DAFTAR KONTRIBUSI ANGGOTA KELOMPOK

Berisi daftar keterangan kontribusi setiap anggota kelompok pada pengerjaan pembuatan program ini.

No	: 241511001
Nama	: Andi Putra Wijaya
Kontribusi	: a) Laporan. b) Integrasi aplikasi dengan library miniaudio. c) Membuat template laporan menggunakan typst. d) Flow menjalankan musik. e) Visualisasi

No	: 241511012
Nama	: Gilang Aditya Sumarna
Kontribusi	: a) Laporan b) Pembuatan Tree Musik c) Membuat fitur search. d) Visualisasi.

No	: 241511025
Nama	: Raffi Fauzi Hermawan
Kontribusi	: a) Laporan. b) Membuat tampilan dan kode bersama ADT bernama ui. c) Membuat modul untuk konversi waktu. d) Visualisasi.