BINUS UNIVERSITY ONLINE

People
Innovation
Excellence

Algorithm and Programming

Kelas: COMP6112036 PQCA - LAB

Diusulkan oleh Group 4 :

DADAN HAMDANI (2802621405)

DHEA FIKY FATCHATUR RIZKY (2802621393)

HERMAWAN (2802622111)

IAN WINANTO (2802612741)

NICHOLAS CHANDRA (2802617061)

Binus University

Semester Ganjil 2024/2025 Periode 1

People
Innovation
Excellence

1. Link Git : https://github.com/dhearizky12/01.-Tugas-Praktikum-LAB-2
2. Link Demo Program: Click here

BINUS UNIVERSITY ONLINE

People Innovation Excellence

PROYEK PRAKTIKUM
(Kelompok)

Soal Case

Sebuah toko buku menjual berbagai macam jenis buku. Pemilik toko buku tersebut menginginkan sebuah aplikasi yang mampu mencatat semua transaksi yang ada sehingga memudahkan dalam me-monitoring usahanya.

Berikut aturan dalam membuat aplikasinya:
Pertama kali program dijalankan, program akan membaca file "databuku.txt". Pada data tersebut terdapat data buku antara lain: *kode buku, nama buku, jenis buku, dan harga*.

Program memiliki 6 pilihan, yaitu:

a. Input
b. View History
c. View Buku
d. Delete History
e. Delete Buku
f. Exit

Keterangan Pilihan tampilan

1) Jika user memilih menu Input data (tekan tombol '1'), maka program akan:
   - Meminta inputan buku (*nama buku, jenis buku, dan harga)*
2) Jika user memilih menu *View History* (tekan tombol '2'), maka program akan menampilkan data history penjualan yang pernah dilakukan.
3) Jika user memilih menu *View Data* (tekan tombol '3'), maka program akan menampilkan seluruh data buku.
4) Jika user memilih menu *Delete History* (tekan tombol '4'), maka program akan menampilkan list data history penjualan.
   - Meminta inputan index. Validasikan input minimal 1 dan maksimal sebanyak jumlah data

- Hapus data sesuai dengan index yang diinput. Contoh: Jika user memilih index 1, maka hapus data yang pertama. Lalu Tampilkan pesan “Data Successfully delete..”.

5) Jika user memilih menu *Delete Buku* (tekan tombol ‘5’), maka program akan menampilkan list data buku.
   - Meminta inputan index. Validasikan input minimal 1 dan maksimal sebanyak jumlah data
   - Hapus data sesuai dengan index yang diinput. Contoh: Jika user memilih index 1, maka hapus data yang pertama. Lalu Tampilkan pesan “Data Successfully delete..”.
6) Jika user memilih menu *Exit* (tekan tombol ‘6’), maka program akan **menulis data** tersebut ke dalam file ”databuku.txt” dan program selesai dijalankan.

People
Innovation
Excellence

PENGUMPULAN TUGAS:

3. Link Git : https://github.com/dhearizky12/01.-Tugas-Praktikum-LAB-2
4. Link Demo Program: Click here
5. Berikan catatan/komentar dari masing-masing code yang dibuat.
   a. Header dan Definisi Konstanta

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#define MAXC 50
#define TYPE 10
#define CREATED_TIME 20
#define INITIAL_BUFFER_SIZE 256

#define MAX_BOOK_CODE 20
#define MAX_BOOK_NAME 32
#define MAX_BOOK_TYPE 10
#define MAX_BOOK_PRICE 10
#define MAX_RECORD_TYPE 10
#define MAX_BUYER 20
#define MAX_SALE_DATE 20
#define TEMPORARY_FILE_NAME "./tempdatabuku.txt"

#ifdef _WIN32
    #define FILE_PATH "../databuku.txt"  // Windows
    #define CLEAR_SCREEN "cls"
#else
    #define FILE_PATH "./databuku.txt"  // macOS/Linux
    #define CLEAR_SCREEN "clear"
#endif
```

Penjelasan:

- Beberapa library standar C di-include, seperti stdio.h, string.h, stdlib.h, dan time.h, untuk mendukung operasi input/output, manipulasi string, alokasi memori, dan penanganan waktu.
- Beberapa konstanta didefinisikan untuk membatasi ukuran string

dan buffer, seperti MAX_BOOK_NAME, MAX_BOOK_TYPE, dan INITIAL_BUFFER_SIZE.
- TEMPORARY_FILE_NAME digunakan untuk menyimpan nama file sementara saat melakukan operasi penghapusan atau pembaruan data.
- FILE_PATH dan CLEAR_SCREEN didefinisikan secara kondisional berdasarkan sistem operasi (_WIN32 untuk Windows dan lainnya untuk macOS/Linux).

b. Struktur Data Book

```
typedef struct
{
    char bookCode[MAX_BOOK_CODE]; // AUTO GENERATED SINCE IT IS A
UNIQUE
    char bookName[MAX_BOOK_NAME];
    char bookType[MAX_BOOK_TYPE];
    unsigned int price; // price should not be minus.
    char createdTime[CREATED_TIME];
    char recordType[TYPE]; // Store data type: "buku" or
"penjualan"
    union{
        struct{
            char buyer[MAX_BUYER];
            char saleDate[CREATED_TIME];
        }dataPenjualan;
    } additionalData; //berisi data misal data buku, data
penjualan
    int isDeleted; //flag untuk menandai apakah entri terhapus
} Book;
```

Penjelasan:

- Struktur Book digunakan untuk merepresentasikan data buku dan penjualan.
- bookCode: Kode buku yang dihasilkan secara otomatis dan unik.
- bookName: Nama buku dengan batasan panjang MAX_BOOK_NAME.
- bookType: Jenis buku dengan batasan panjang MAX_BOOK_TYPE.
- price: Harga buku yang tidak boleh negatif (tipe unsigned int).

- createdTime: Waktu pembuatan record dengan format string.
- recordType: Menyimpan tipe record, yaitu "buku" atau "penjualan".
- additionalData: Union yang menyimpan data tambahan. Untuk "penjualan", terdapat buyer (pembeli) dan saleDate (tanggal penjualan).
- isDeleted: Flag untuk menandai apakah record telah dihapus (1 untuk dihapus, 0 untuk tidak dihapus).

c. Fungsi allocateBuffer

```
char *allocateBuffer(size_t size)
{
    char *buffer = (char *)malloc(size);
    if (buffer == NULL)
    {
        fprintf(stderr, "Error: Failed to allocate memory.\n");
        exit(EXIT_FAILURE);
    }
    return buffer;
}
```

Penjelasan:

- Fungsi ini mengalokasikan memori untuk buffer dengan ukuran yang ditentukan (size).
- Jika alokasi gagal, program menampilkan pesan error dan keluar dengan status EXIT_FAILURE.
- Fungsi ini berguna untuk menghindari penulisan kode alokasi memori yang berulang.

d. Fungsi reallocateBuffer

```
char *reallocateBuffer(char *buffer, size_t newSize)
{
    char *newBuffer = (char *)realloc(buffer, newSize);
    if (newBuffer == NULL)
    {
        fprintf(stderr, "Error: Failed to reallocate memory.\n");
        free(buffer);
```

```
        exit(EXIT_FAILURE);
    }
    return newBuffer;
}
```

Penjelasan:

- Fungsi ini mengubah ukuran buffer yang sudah dialokasikan sebelumnya menggunakan realloc.
- Jika realokasi gagal, program menampilkan pesan error, membebaskan memori buffer lama, dan keluar dengan status EXIT_FAILURE.
- Fungsi ini berguna untuk menangani kasus di mana ukuran buffer perlu diperbesar (misalnya, saat input melebihi ukuran buffer awal).

e. Fungsi createTXTIfNotExists

```
void createTXTIfNotExists(const char *filename)
{
    FILE *file = fopen(filename, "r");
    if (file == NULL)
    {
        file = fopen(filename, "w");
        if (file == NULL)
        {
            fprintf(stderr, "Error creating file.\n");
            exit(1);
        }
        fprintf(file,
"bookCode,bookName,bookType,bookPrice,createdTime,recordType,buye
r,saleDate,isDeleted\n");
        fclose(file);
    }
    else
    {
        fclose(file);
    }
}
```

Penjelasan:

- Fungsi ini memeriksa apakah file dengan nama yang diberikan ada. Jika tidak, fungsi akan membuat file baru dan menulis header ke dalamnya. Ini memastikan bahwa file selalu memiliki format yang konsisten.

f. Fungsi createBookCode

```
void createBookCode(Book *book)
{
    time_t currentTime = time(NULL);
    struct tm tm = *localtime(&currentTime);
    int todayYear = tm.tm_year + 1900;
    int currentMonth = tm.tm_mon + 1;
    int currentDay = tm.tm_mday;
    int currentHour = tm.tm_hour;
    int currentMinute = tm.tm_min;
    int currentSec = tm.tm_sec;
    sprintf(book->bookCode, "%04d%02d%02d_%02d%02d%02d",
todayYear, currentMonth, currentDay, currentHour, currentMinute,
currentSec);
}
```

Penjelasan:

- Fungsi ini menghasilkan kode buku unik berdasarkan waktu saat ini. Kode buku dihasilkan dalam format YYYYMMDD_HHMMSS, yang memastikan bahwa setiap kode buku unik.

g. Fungsi createdTime

```
void createdTime(Book *book, const char *desiredRecordType)
{
    time_t currentTime = time(NULL);
    struct tm *tm = localtime(&currentTime);

    if ( strcmp(desiredRecordType, "buku") == 0 )
    {
        strftime(book->createdTime, sizeof(book->createdTime),
"%Y-%m-%d %H:%M:%S", tm);
```

```
    }
    else if ( strcmp (desiredRecordType, "penjualan") == 0 )
    {
        strftime(book->additionalData.dataPenjualan.saleDate,
sizeof(book->additionalData.dataPenjualan.saleDate), "%Y-%m-%d
%H:%M:%S", tm);
    }
}
```

Penjelasan:

- Fungsi ini mengatur waktu pembuatan (createdTime) atau waktu penjualan (saleDate) berdasarkan jenis record. Waktu diambil dari sistem dan diformat sebagai string.

h. Fungsi removeTrailingNewLine

```
void removeTrailingNewLine(char *str)
{
    size_t len = strlen(str);
    if(len > 0 && str[len-1] == '\n')
    {
        str[len-1] = '\0';
    }
}
```

Penjelasan:

- Fungsi ini menghapus newline (\n) dari akhir string yang dihasilkan oleh fgets. Ini berguna untuk memastikan bahwa input pengguna tidak mengandung newline yang tidak diinginkan.

i. Fungsi clearInputBuffer

```
void clearInputBuffer()
{
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}
```

Penjelasan:

- Fungsi ini membersihkan buffer input dengan membaca dan

membuang karakter sampai menemukan newline atau EOF. Ini berguna untuk menghindari masalah input yang tersisa di buffer setelah menggunakan scanf.

j. Fungsi insertBook

```
int insertBook(const char *filename, Book *book)
{
    FILE *file = fopen(filename, "a");
    if (file == NULL)
    {
        fprintf(stderr, "Error opening file.\n");
        return 1;
    }
```

Penjelasan:

- Fungsi ini bertujuan untuk menambahkan data buku ke file.
- File dibuka dalam mode append ("a"). Jika gagal, program menampilkan pesan error dan mengembalikan nilai 1.

k. Fungsi Alokasi Buffer Dinamis untuk Nama Buku

```
// Alokasi buffer dinamis untuk nama buku
    char *bookName = allocateBuffer(INITIAL_BUFFER_SIZE);
    size_t bookNameSize = INITIAL_BUFFER_SIZE;

    printf("Insert book name (max %d characters): ",
MAX_BOOK_NAME);
    if (fgets(bookName, bookNameSize, stdin) == NULL)
    {
        fprintf(stderr, "Error reading book name.\n");
        free(bookName);
        return 1;
    }
    removeTrailingNewLine(bookName);

    // Realokasi jika input melebihi buffer awal
    while (strlen(bookName) >= bookNameSize - 1)
    {
```

```
        bookNameSize *= 2;
        bookName = reallocateBuffer(bookName, bookNameSize);
    }
    strncpy(book->bookName, bookName, MAX_BOOK_NAME);
    free(bookName); // Bebaskan memori setelah digunakan
```

Penjelasan:

- Buffer dinamis dialokasikan untuk menyimpan nama buku menggunakan fungsi allocateBuffer.
- Input nama buku dibaca menggunakan fgets untuk menghindari masalah buffer overflow.
- Jika input melebihi ukuran buffer awal, buffer direalokasi menggunakan reallocateBuffer.
- Newline dihapus dari input menggunakan removeTrailingNewLine.
- Nama buku disalin ke struktur book menggunakan strncpy.
- Buffer dinamis dibebaskan setelah digunakan untuk menghindari memory leak.

I. Fungsi Alokasi Buffer Dinamis untuk Jenis Buku

```
// Alokasi buffer dinamis untuk jenis buku
    char *bookType = allocateBuffer(INITIAL_BUFFER_SIZE);
    size_t bookTypeSize = INITIAL_BUFFER_SIZE;

    printf("Insert book type (max %d characters): ",
MAX_BOOK_TYPE);
    if (fgets(bookType, bookTypeSize, stdin) == NULL) {
        fprintf(stderr, "Error reading book type.\n");
        free(bookType);
        return 1;
    }
    removeTrailingNewLine(bookType);

    // Realokasi jika input melebihi buffer awal
    while (strlen(bookType) >= bookTypeSize - 1)
    {
        bookTypeSize *= 2;
        bookType = reallocateBuffer(bookType, bookTypeSize);
    }
```

```
    strncpy(book->bookType, bookType, MAX_BOOK_TYPE);
    free(bookType); // Bebaskan memori setelah digunakan
```

Penjelasan:

- Logika yang sama digunakan untuk jenis buku seperti pada nama buku.
- Buffer dinamis dialokasikan, input dibaca, newline dihapus, dan buffer direalokasi jika diperlukan.
- Jenis buku disalin ke struktur book dan buffer dibebaskan setelah digunakan.

m. Fungsi Meminta Input Harga Buku

```
char *priceInput = allocateBuffer(INITIAL_BUFFER_SIZE);
    int price = 0;

    while (1)
    {
        printf("Insert book price (Please use numeric value in
IDR): ");
        if (fgets(priceInput, INITIAL_BUFFER_SIZE, stdin) ==
NULL)
        {
            puts("(User canceled input)");
            free(priceInput);
            return -1; // Handle manual EOF or input error
        }

        removeTrailingNewLine(priceInput);

        // Convert input to an integer and validate
        char *end;
        price = strtol(priceInput, &end, 10);

        if (*end != '\0' || priceInput == end)
        {
            fprintf(stderr, "Please enter a valid price.\n");
            // stderr argument is used to display the error
message on the screen because it can procesing dynamic value
```

```
        }
        else if (price <= 0)
        {
            printf("Price should be above 0\n");
        }
        else
        {
            book->price = price;
            break;
        }
    }
    free(priceInput); // Bebaskan memori setelah digunakan
```

Penjelasan:

- Buffer dinamis dialokasikan untuk menyimpan input harga buku.
- Input harga dibaca menggunakan fgets dan newline dihapus.
- Input diubah menjadi integer menggunakan strtol dan divalidasi.
- Jika input tidak valid (bukan angka atau kurang dari atau sama dengan 0), program meminta input lagi.
- Jika input valid, harga disimpan ke struktur book dan loop dihentikan.
- Buffer dinamis dibebaskan setelah digunakan.

n. Fungsi Mengatur Tipe Record dan Status Penghapusan

```
    //set record type to "buku"
    strcpy(book->recordType, "buku"); //strcpy untuk mengcopy
string "buku" ke dalam recordType


    //set isDeleted to 0 (not deleted)
    book->isDeleted = 0;
```

Penjelasan:

- Tipe record diatur menjadi "buku" menggunakan strcpy.
- Status penghapusan (isDeleted) diatur ke 0 (tidak dihapus).

o. Fungsi Membuat Kode Buku dan Waktu Pembuatan

```
    //autogenerate book code
    createBookCode(book);
```

```
    //auto set created time data
    createdTime(book, "buku");
```

Penjelasan:

- Kode buku dibuat secara otomatis menggunakan fungsi createBookCode.
- Waktu pembuatan record diatur menggunakan fungsi createdTime.

p. Fungsi Menulis Data Buku ke File

```
   fprintf(file, "%s,%s,%s,%d,%s,%s,,,%d\n", book->bookCode,
book->bookName, book->bookType, book->price, book->createdTime,
book->recordType, book->isDeleted);
    fclose(file);

    return 0;
}
```

Penjelasan:

- Data buku ditulis ke file dalam format CSV menggunakan fprintf.
- File ditutup setelah operasi selesai.
- Fungsi mengembalikan 0 jika operasi berhasil.

q. Fungsi setConsoleFontColor dan resetConsoleFontColor

```
void setConsoleFontColor(int colour)
{
    printf("\033[38;5;%dm", colour);
}

void resetConsoleFontColor()
{
    printf("\033[0m");
}
```

Penjelasan: Fungsi ini mengubah warna teks di konsol menggunakan kode ANSI escape sequence. resetConsoleFontColor mengembalikan warna teks ke default.

r. Fungsi displayData

```
void displayData(const char *filename, const char
*desiredRecordType)
{
    FILE *file = fopen(filename, "r");

    if (file == NULL)
    {
        fprintf(stderr, "Error opening file.\n");
        return;
    }
```

Penjelasan:

- Fungsi ini bertujuan untuk menampilkan data dari file berdasarkan tipe record yang diinginkan (desiredRecordType).
- File dibuka dalam mode read ("r"). Jika gagal, program menampilkan pesan error dan keluar dari fungsi.

s. Fungsi Inisialisasi Variabel dan Membaca Header

```
char line[INITIAL_BUFFER_SIZE];
    int isEmpty = 1;

    // Read and display the header
    if (fgets(line, sizeof(line), file) != NULL)
    {
        if ( strcmp(desiredRecordType, "penjualan") == 0 )
        {
            printf("\n------- --------------------
---------------------------------- ------------ -----------
-------------------- ---------- --------------------
--------------------\n");
            printf("%-7s %-20s %-35s %-13s %-11s %-20s %-11s
%-20s
%-20s\n","index","bookCode","bookName","bookType","bookPrice","cr
eatedTime","recordType","buyer","saleDate\n");
            printf("------- --------------------
```

```
---------------------------------- ------------ -----------
------------------- ---------- --------------------
--------------------\n");
        }
        else if( strcmp(desiredRecordType, "buku") == 0 )
        {
            printf("\n------- --------------------
---------------------------------- ------------ -----------
-------------------- ----------\n");
            printf("%-7s %-20s %-35s %-13s %-11s %-20s
%-11s\n","index","bookCode","bookName","bookType","bookPrice","cr
eatedTime","recordType\n");
            printf("------- --------------------
---------------------------------- ------------ -----------
-------------------- ----------\n");
        }
    }
```

Penjelasan:

- Variabel isEmpty digunakan untuk menandai apakah ada data yang sesuai dengan tipe record yang diinginkan.
- Header file dibaca menggunakan fgets dan disimpan di variabel line.
- Jika tipe record adalah "penjualan", program menampilkan header untuk data penjualan.
- Jika tipe record adalah "buku", program menampilkan header untuk data buku.

t. Fungsi Iterasi Melalui Data dan Menampilkan Data yang Sesuai

```
//membaca dan memproses setiap baris data
    int index = 1;
    while (fgets(line, sizeof(line), file))
    {
        char *bookCode = strtok(line, ","); //strtok untuk
memisahkan string berdasarkan delimiter. parameter line bertujuan
untuk memisahkan string line berdasarkan delimiter ","
        char *bookName = strtok(NULL, ","); // parameter NULL
pada strtok bertujuan untuk melanjutkan pemisahan string dari
posisi terakhir pemisahan
```

```
        char *bookType = strtok(NULL, ","); // strtok(NULL, ",")
berarti melanjutkan pemisahan string dari posisi terakhir
pemisahan
        char *bookPrice = strtok(NULL, ",");
        char *createdTime = strtok(NULL, ",");
        char *recordType = strtok(NULL, ",");
        char *buyer = strtok(NULL, ",");
        char *saleDate = strtok(NULL, ",");
        char *isDeleted = strtok(NULL, ",");

        int isDeletedInt = atoi(isDeleted); //atoi untuk mengubah
string menjadi integer
```

Penjelasan:

- Setiap baris file dibaca menggunakan fgets dan diproses menggunakan strtok untuk memisahkan nilai-nilai yang dipisahkan oleh koma.
- strtok digunakan untuk memecah string berdasarkan delimiter (koma dalam hal ini).
- atoi digunakan untuk mengubah string isDeleted menjadi integer.

u. Fungsi Menampilkan Data yang Sesuai dengan Tipe Record

```
//memeriksa kondisi isDeleted == 0 dan recordType ==
desiredRecordType
        //strcmp untuk membandingkan dua string
        if ( isDeletedInt == 0 && strcmp(recordType,
desiredRecordType) == 0 && strcmp("penjualan", desiredRecordType)
== 0 )
        {
            printf("%-7d %-20s %-35s %-13s %-11s %-20s %-11s
%-20s %-20s\n", index, bookCode, bookName, bookType, bookPrice,
createdTime, recordType, buyer, saleDate);
            isEmpty = 0;
            index++;
        }
        else if ( isDeleted == 0 && strcmp(recordType,
desiredRecordType) == 0 && strcmp("buku", desiredRecordType) ==
```

```
0)
        {
            printf("%-7d %-20s %-35s %-13s %-11s %-20s
%-11s\n",index, bookCode, bookName, bookType, bookPrice,
createdTime, recordType );
            isEmpty = 0;
            index++;
        }
    }
```

Penjelasan:

- Jika isDeleted == 0 dan recordType sesuai dengan desiredRecordType, data tersebut ditampilkan.
- Untuk tipe record "penjualan", semua kolom ditampilkan, termasuk buyer dan saleDate.
- Untuk tipe record "buku", hanya kolom yang relevan dengan data buku yang ditampilkan.
- Variabel isEmpty diatur ke 0 jika ada data yang ditampilkan, dan index bertambah untuk menandai nomor urut data.

v. Fungsi Menampilkan Pesan Jika Tidak Ada Data

```
  if (isEmpty)
    {
        setConsoleFontColor(160);
        printf(" ------------------------------------------ No
data available.
-----------------------------------------------\n");
        resetConsoleFontColor();
    }

    fclose(file);
}
```

Penjelasan:

- Jika tidak ada data yang sesuai dengan tipe record yang diinginkan (isEmpty == 1), program menampilkan pesan "No data available" dengan warna konsol yang sesuai.
- File ditutup setelah operasi selesai.

w. Fungsi insertSaleBook

```
int insertSaleBook(const char *filename, Book *book)
{
    FILE *file;
    // Menampilkan daftar buku
    displayData(filename, "buku");
```

Penjelasan:

- Fungsi ini bertujuan untuk menambahkan data penjualan buku ke file.
- Pertama, program menampilkan daftar buku yang tersedia menggunakan fungsi displayData dengan parameter "buku".

x. Fungsi Loop untuk Memilih Buku yang Akan Dijual

```
while (1)
    {
        file = fopen(filename, "r");
        if (file == NULL)
        {
            fprintf(stderr, "Error opening file.\n");
            return 1;
        }
```

Penjelasan:

- Program membuka file dalam mode read ("r"). Jika gagal, program menampilkan pesan error dan mengembalikan nilai 1.
- Loop while (1) digunakan untuk memastikan pengguna dapat mencoba lagi jika kode buku yang dimasukkan tidak valid.

y. Fungsi Meminta Input Kode Buku

```
    // Meminta kode buku dari user
    char selectedBookCode[MAX_BOOK_CODE];
    printf("Masukkan kode buku yang akan dijual: ");
    fgets(selectedBookCode, sizeof(selectedBookCode), stdin);
    removeTrailingNewLine(selectedBookCode);
```

Penjelasan:

- Program meminta pengguna untuk memasukkan kode buku yang akan dijual.
- Input dibaca menggunakan fgets untuk menghindari masalah

buffer overflow.
- Fungsi removeTrailingNewLine digunakan untuk menghapus newline (\n) dari input.

z. Fungsi Mencari Buku yang Sesuai dengan Kode yang Dimasukkan

```
    int found = 0;
    Book tempBook;

    char line[INITIAL_BUFFER_SIZE];
    fgets(line, sizeof(line), file); // Copy header to temp

    while (fgets(line, sizeof(line), file))
    {
        sscanf(line, "%[^,],%[^,],%[^,],%u,%[^,],%[^,],,,%d",
               tempBook.bookCode, tempBook.bookName, tempBook.bookType,
               &tempBook.price, tempBook.createdTime, tempBook.recordType,
&tempBook.isDeleted);

        if (strcmp(tempBook.bookCode, selectedBookCode) == 0 &&
strcmp(tempBook.recordType, "buku") == 0 && tempBook.isDeleted == 0)
        {
            found = 1;
            *book = tempBook;
        }
    }
    fclose(file);
```

Penjelasan:
- Program membaca setiap baris file dan mem-parsenya ke dalam struktur tempBook.
- Jika kode buku yang dimasukkan cocok dengan data di file, dan record tersebut adalah "buku" serta belum dihapus (isDeleted == 0), maka buku tersebut disimpan ke variabel book.
- Variabel found diatur ke 1 jika buku ditemukan.

aa. Fungsi Penanganan Jika Buku Tidak Ditemukan

```
if (!found)
        {
```

```
            setConsoleFontColor(160);
            fprintf(stderr, "Kode buku tidak ditemukan. Silakan
coba lagi.\n");
            resetConsoleFontColor();
        }
        else
        {
            break;
        }
    }
```

Penjelasan:

- Jika buku tidak ditemukan, program menampilkan pesan error dengan warna konsol yang sesuai.
- Jika buku ditemukan, program keluar dari loop.

bb. Fungsi Meminta Input Nama Pembeli

```
// Meminta nama pembeli
   printf("Masukkan nama pembeli: ");
   fgets(book->additionalData.dataPenjualan.buyer,
sizeof(book->additionalData.dataPenjualan.buyer), stdin);
   removeTrailingNewLine(book->additionalData.dataPenjualan.buyer);
```

Penjelasan:

- Program meminta pengguna untuk memasukkan nama pembeli.
- Input dibaca menggunakan fgets dan newline dihapus menggunakan removeTrailingNewLine.

cc. Fungsi Mengatur Data Penjualan

```
strcpy(book->recordType, "penjualan");
   createdTime(book, "penjualan");
   book->isDeleted = 0;
```

Penjelasan:

- Tipe record diatur menjadi "penjualan".
- Fungsi createdTime dipanggil untuk mengatur waktu pembuatan record.
- isDeleted diatur ke 0 karena record ini baru dan belum dihapus.

dd. Fungsi Menulis Data Penjualan ke File

```
// Menulis data ke file
    file = fopen(filename, "a");
    if (file == NULL)
    {
        fprintf(stderr, "Error opening file.\n");
        return 1;
    }
    fprintf(file, "%s,%s,%s,%u,%s,%s,%s,%s,%d\n",
            book->bookCode, book->bookName, book->bookType,
book->price, book->createdTime, book->recordType,
book->additionalData.dataPenjualan.buyer,
book->additionalData.dataPenjualan.saleDate, book->isDeleted);
    fclose(file);

    return 0;
}
```

Penjelasan:

- File dibuka dalam mode append ("a") untuk menambahkan data penjualan ke akhir file.
- Jika gagal membuka file, program menampilkan pesan error dan mengembalikan nilai 1.
- Data penjualan ditulis ke file dalam format CSV.
- File ditutup setelah operasi selesai.

ee. Fungsi deleteData

```
int deleteData(const char *filename, int deletedIndex, const char
*desiredRecordType) {
    FILE *file;
    FILE *tempFile;
    while (1)
    {
        file = fopen(filename, "r");
        if (file == NULL)
        {
            fprintf(stderr, "Error opening file.\n");
            return 1;
```

```
        }
```

Penjelasan:

- Fungsi ini bertujuan untuk menghapus data dari file berdasarkan indeks dan tipe record (desiredRecordType).
- File dibuka dalam mode read ("r"). Jika gagal, program menampilkan pesan error dan mengembalikan nilai 1.
- Loop while (1) digunakan untuk memastikan operasi penghapusan dapat diulang jika diperlukan.

ff. Fungsi Inisialisasi Variabel dan Membaca Header

```
       int found = 0;
        Book tempBook;

        char line[INITIAL_BUFFER_SIZE];
        fgets(line, sizeof(line), file); // Copy header to temp

        int indexPenjualan = 0;
        int indexBuku = 0;
```

Penjelasan:

- Variabel found digunakan untuk menandai apakah indeks yang ingin dihapus ditemukan.
- tempBook digunakan untuk menyimpan data sementara dari setiap baris file.
- Header file dibaca menggunakan fgets dan disimpan di variabel line.
- indexPenjualan dan indexBuku digunakan untuk melacak indeks saat iterasi melalui data.

gg. Fungsi Membuat File Temporary

```
    tempFile = fopen(TEMPORARY_FILE_NAME, "w"); // mempersiapkan
file temporary untuk menyimpan data terupdate setelah di delete
        if (tempFile == NULL)
        {
            fprintf(stderr, "Error opening tempFile.\n");
            return 1;
        }
        fprintf(tempFile,
"bookCode,bookName,bookType,bookPrice,createdTime,recordType,buye
r,saleDate,isDeleted\n"); // mempersiapkan header file di
```

```
temporary file
```

Penjelasan:

- File temporary dibuat untuk menyimpan data yang sudah diupdate setelah penghapusan.
- Jika gagal membuka file temporary, program menampilkan pesan error dan mengembalikan nilai 1.
- Header file ditulis ke file temporary untuk memastikan struktur file tetap konsisten.

hh. Fungsi Iterasi Melalui Data dan Penghapusan

```
    while (fgets(line, sizeof(line), file))
        {
            sscanf(line,
"%[^,],%[^,],%[^,],%u,%[^,],%[^,],%[^,],%[^,],%d",
                    tempBook.bookCode, tempBook.bookName,
tempBook.bookType,
                    &tempBook.price, tempBook.createdTime,
tempBook.recordType, tempBook.additionalData.dataPenjualan.buyer,
tempBook.additionalData.dataPenjualan.saleDate,
&tempBook.isDeleted);
            if (strcmp(desiredRecordType, "penjualan") == 0 &&
tempBook.isDeleted == 0 && strcmp( tempBook.recordType,
desiredRecordType ) == 0) { // memastikan hanya melakukan looping
terhadap record penjualan dan belum pernah di delete
                indexPenjualan++;
                if (indexPenjualan == deletedIndex) { //
melakukan pengecekan apakah index yg ingin di delete sudah sesuai
                    found = 1;
                    // fprintf(tempFile,
"%s,%s,%s,%u,%s,%s,%s,%s,%d\n", tempBook.bookCode,
tempBook.bookName, tempBook.bookType, tempBook.price,
tempBook.createdTime, tempBook.recordType,
tempBook.additionalData.dataPenjualan.buyer,
tempBook.additionalData.dataPenjualan.saleDate, 1); // menyimpan
baris yg akan didelete dengan isDeleted 1
                    continue;
                }
            }
            else if (strcmp(desiredRecordType, "buku") == 0 &&
tempBook.isDeleted == 0 && strcmp( tempBook.recordType,
desiredRecordType ) == 0) { // memastikan hanya melakukan looping
```

```
terhadap record penjualan dan belum pernah di delete
                indexBuku++;
                if (indexBuku == deletedIndex) { // melakukan
pengecekan apakah index yg ingin di delete sudah sesuai
                    found = 1;
                    // fprintf(tempFile,
"%s,%s,%s,%u,%s,%s,,,%d\n", tempBook.bookCode, tempBook.bookName,
tempBook.bookType, tempBook.price, tempBook.createdTime,
tempBook.recordType, 1); // menyimpan baris yg akan didelete
dengan isDeleted 1
                    continue;
                }
            }
```

Penjelasan:

- Setiap baris file dibaca menggunakan fgets dan di-parse menggunakan sscanf untuk memisahkan nilai-nilai yang dipisahkan oleh koma.
- Jika tipe record adalah "penjualan" dan belum dihapus (isDeleted == 0), program memeriksa apakah indeks yang ingin dihapus sesuai dengan deletedIndex.
- Jika tipe record adalah "buku" dan belum dihapus, program melakukan pengecekan yang sama.
- Jika indeks yang ingin dihapus ditemukan, variabel found diatur ke 1, dan baris tersebut dilewati (tidak ditulis ke file temporary).

ii. Fungsi Menulis Data ke File Temporary

```
if (strcmp( tempBook.recordType, "buku" ) == 0)
            {
                fprintf(tempFile, "%s,%s,%s,%u,%s,%s,,,%d\n",
                    tempBook.bookCode, tempBook.bookName,
tempBook.bookType, tempBook.price, tempBook.createdTime,
tempBook.recordType, tempBook.isDeleted);

            }
            else
            {
                fprintf(tempFile, "%s,%s,%s,%u,%s,%s,%s,%s,%d\n",
                    tempBook.bookCode, tempBook.bookName,
tempBook.bookType, tempBook.price, tempBook.createdTime,
```

```
tempBook.recordType,
strlen(tempBook.additionalData.dataPenjualan.buyer) > 0 ?
tempBook.additionalData.dataPenjualan.buyer : "",
strlen(tempBook.additionalData.dataPenjualan.saleDate) > 0 ?
tempBook.additionalData.dataPenjualan.saleDate : "",
tempBook.isDeleted);
            }
        }
        fclose(file);
        fclose(tempFile);
```

Penjelasan:

- Data yang tidak dihapus ditulis kembali ke file temporary.
- Jika tipe record adalah "buku", data ditulis tanpa kolom buyer dan saleDate.
- Jika tipe record adalah "penjualan", semua kolom ditulis, termasuk buyer dan saleDate.
- File utama dan file temporary ditutup setelah operasi selesai.

jj. Fungsi Overwrite File Utama

```
remove(filename); // Delete file data buku
        rename(TEMPORARY_FILE_NAME, filename); // Ubah nama file
temporary menjadi databuku

        if (!found)
        {
            setConsoleFontColor(160);
            fprintf(stderr, "Indeks tidak ditemukan. Silakan coba
lagi.\n");
            resetConsoleFontColor();
            return 1;
        }
        else
        {
            break;
        }
    }
    return 0;
}
```

Penjelasan:

- File utama dihapus menggunakan remove, dan file temporary diubah namanya menjadi file utama menggunakan rename.
- Jika indeks yang ingin dihapus tidak ditemukan, program menampilkan pesan error dan mengembalikan nilai 1.
- Jika penghapusan berhasil, program keluar dari loop dan mengembalikan nilai 0.

kk. Fungsi deleteDataMenu

```
void deleteDataMenu(const char *filename, const char
*desiredRecordType)
{
    displayData(filename, desiredRecordType);


printf("\n=========================================================
==\n");
    printf("Silahkan input indeks yg ingin di delete: ");
    int deletedIndex;
    scanf("%d", &deletedIndex);
    int resultDelete = deleteData(filename, deletedIndex,
desiredRecordType);
    if (resultDelete == 0) {
        setConsoleFontColor(177);
        printf("Data Penjualan indeks ke %d berhasil dihapus",
deletedIndex);
        resetConsoleFontColor();
    }
}
```

Penjelasan:

- Fungsi ini menampilkan data menggunakan displayData dan meminta pengguna untuk memasukkan indeks yang ingin dihapus.
- Fungsi deleteData dipanggil untuk melakukan penghapusan.
- Jika penghapusan berhasil, program menampilkan pesan sukses dengan warna konsol yang sesuai.

ll. Fungsi main

```
int main()
```

```
{
    const char *fileName = FILE_PATH;  // Menggunakan konstanta
FILE_PATH untuk menyimpan path file
    createTXTIfNotExists(fileName);  // Membuat file jika belum
ada
    int choice;
    Book book;

    setConsoleFontColor(106);

printf("\n============================================================
========================\n");
    printf("****************** Welcome to the BINUS Group 4 Book
store ******************\n");

printf("============================================================
======================\n");
    resetConsoleFontColor();
```

Penjelasan:

- Program dimulai dengan menginisialisasi path file menggunakan konstanta FILE_PATH.
- Fungsi createTXTIfNotExists dipanggil untuk memastikan file data sudah ada sebelum program berjalan.
- Pesan selamat datang ditampilkan dengan warna konsol yang diatur menggunakan setConsoleFontColor dan direset setelahnya.

mm. Fungsi Loop Utama (Menu)

```
while (1)
    {

printf("\n============================================================
========================\n");
        printf("***************************** Please Select MENU
***************************\n");

printf("============================================================
======================\n");
        printf("1. Insert Book Data.");
```

```
        setConsoleFontColor(154);
        printf("Pada menu ini, user dapat menginput data
buku\n");
        resetConsoleFontColor();

        printf("2. View History Penjualan.");
        setConsoleFontColor(154);
        printf("Pada menu ini akan ditampilkan data history
penjualan\n");
        resetConsoleFontColor();

        printf("3. View Book Data.");
        setConsoleFontColor(154);
        printf("Pada menu ini akan ditampilkan seluruh data
buku\n");
        resetConsoleFontColor();

        printf("4. Delete History Penjualan.");
        setConsoleFontColor(154);
        printf("Pada menu ini user dapat mendelete data
penjualan\n");
        resetConsoleFontColor();

        printf("5. Delete Book.");
        setConsoleFontColor(154);
        printf("Pada menu ini sistem akan menampilkan data buku
untuk di delete\n");
        resetConsoleFontColor();

        printf("6. Insert Data Penjualan.");
        setConsoleFontColor(154);
        printf("Pada menu ini user dapat menginput data
penjualan\n");
        resetConsoleFontColor();

        printf("7. Exit\n");
        printf("Choose an option: ");
        if (scanf("%d", &choice) != 1)
        {
            fprintf(stderr, "Invalid input. Please input numeric
value\n");
            return 1;
        }
```

- Penjelasan:
- Program menggunakan loop while (1) untuk menampilkan menu secara terus-menerus hingga pengguna memilih untuk keluar.
- Setiap opsi menu memiliki deskripsi yang jelas dan ditampilkan dengan warna konsol yang berbeda untuk meningkatkan keterbacaan.
- Input pengguna (choice) divalidasi untuk memastikan bahwa input adalah angka. Jika tidak, program akan menampilkan pesan error dan keluar.

nn. Fungsi Membersihkan Input Buffer

```
while (getchar() != '\n' && getchar() != EOF);
```

Penjelasan:

- Input buffer dibersihkan setelah scanf untuk menghindari masalah jika ada karakter yang tersisa di buffer (misalnya, newline atau karakter tidak valid).
- Ini adalah praktik yang baik untuk memastikan input berikutnya tidak terpengaruh oleh sisa karakter di buffer.

oo. Fungsi Switch Case untuk Menangani Pilihan Menu

```
switch (choice)
        {
        case 1:
            if (insertBook(fileName, &book) == 0)
            {
                setConsoleFontColor(106);
                printf(" ------ Book inserted successfully.
------ \n");
                resetConsoleFontColor();
            }
            break;
        case 2:
            displayData(fileName, "penjualan");
            break;
        case 3:
            displayData(fileName, "buku");
            break;
```

```
        case 4:
            deleteDataMenu(fileName, "penjualan");
            break;
        case 5:
            deleteDataMenu(fileName, "buku");
            break;
        case 6:
            if (insertSaleBook(fileName, &book) == 0)
            {
                setConsoleFontColor(106);
                printf(" ------ Data Penjualan Buku inserted
successfully. ------ \n");
                resetConsoleFontColor();
            }
            break;
        case 7:
            exit(0);
        default:
            setConsoleFontColor(160);
            printf("Invalid choice. Please try again.\n");
            resetConsoleFontColor();
        }
    }
    return 0;
}
```

Penjelasan:

- Case 1 (Insert Book Data):
  Memanggil fungsi insertBook untuk menambahkan data buku ke file. Jika berhasil, menampilkan pesan sukses dengan warna konsol yang sesuai.
- Case 2 (View History Penjualan):
  Memanggil fungsi displayData dengan parameter "penjualan" untuk menampilkan data penjualan.
- Case 3 (View Book Data):
  Memanggil fungsi displayData dengan parameter "buku" untuk menampilkan data buku.
- Case 4 (Delete History Penjualan):
  Memanggil fungsi deleteDataMenu dengan parameter "penjualan" untuk menghapus data penjualan.
- Case 5 (Delete Book):
  Memanggil fungsi deleteDataMenu dengan parameter "buku" untuk menghapus data buku.
- Case 6 (Insert Data Penjualan):

Memanggil fungsi insertSaleBook untuk menambahkan data penjualan ke file.
Jika berhasil, menampilkan pesan sukses dengan warna konsol yang sesuai.

- Case 7 (Exit):
  Menghentikan program dengan memanggil exit(0).
- Default:
  Menangani input yang tidak valid dengan menampilkan pesan error dan meminta pengguna untuk mencoba lagi.