Tugas Kelompok ke-2

Week 8

Anggota Kelompok 6

NIM	Nama
2802548713	**MUHAMAD TOHA**
2802527532	**CINDI TIA ARDANESWARI. H**
2802550541	**SUGENG WAHYUDI**
2802567126	**USAMA WARDANA**

Tugas 1: Analisis Kebutuhan Program untuk Manajemen File

Deskripsi Soal:

Anda diminta untuk merancang kebutuhan program yang mengelola database siswa menggunakan file. Database ini harus bisa melakukan operasi menambah, menghapus, dan mencari data siswa. Setiap entri data siswa harus berisi nama, umur, dan nomor registrasi.

Instruksi Pengerjaan:

1. Gambarkan pseudo code atau flowchart yang mendetail untuk setiap operasi (menambah, menghapus, mencari data siswa).
2. Jelaskan struktur file yang akan digunakan, termasuk format setiap entri data dalam file tersebut.
3. Berikan analisis mengenai bagaimana program Anda akan menangani memori selama operasi file.

- **Pseudo Code/Flowchart**: Harus jelas menunjukkan langkah-langkah untuk input data, proses, dan output hasil.
- **Struktur File**: Deskripsikan format file (misalnya, file teks dengan setiap entri memiliki format tertentu atau file biner). Jelaskan pemisahan data (misal, delimiter seperti koma atau newline).

- **Memory Management**: Analisis bagaimana program akan mengalokasi dan membebaskan memori selama operasi file, misalnya saat membaca atau menulis data.

Tugas 2: Analisis Kesalahan Program untuk Alokasi Memori

Deskripsi Soal

Diberikan cuplikan kode yang bertujuan untuk membuat daftar nilai mahasiswa dinamis tetapi mengandung kesalahan pengelolaan memori. Anda harus menganalisis dan memperbaiki kesalahan tersebut.

Cuplikan Kode:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *grades;
    int numStudents, i;

    printf("Enter number of students: ");
    scanf("%d", &numStudents);

    grades = malloc(numStudents * sizeof(int));  // Memory allocation
    for(i = 0; i < numStudents; i++) {
        printf("Enter grade for student %d: ", i + 1);
        scanf("%d", &grades[i]);
    }

    // Memory leak, no free call
    return 0;
}
```

Instruksi Pengerjaan:

1. Identifikasi dan jelaskan kesalahan dalam pengelolaan memori pada kode tersebut.
2. Berikan kode yang sudah diperbaiki.
3. Gambarkan flowchart yang menunjukkan proses yang benar.

Jawaban:

Tugas 1: Analisis Kebutuhan Program untuk Manajemen File

source code lengkap program kami sertakan dalam file studentmgmt.c dan pseudocode.txt yang merupakan kesatuan dari jawaban ini.

1. **Gambar Pseudo Code atau flowchart**

- **Main Function**

```
MAIN FUNCTION
    DECLARE INT navmenu = 0
    WHILE navmenu != 5 DO
        CALL clrscr
        PRINT program header
        IF msg_notif is not NULL THEN
            PRINT msg_notif
            FREE msg_notif
        ENDIF

        PRINT menu options
        READ navmenu
        SWITCH navmenu DO
            CASE 1:
                CALL addStudent
            CASE 2:
                CALL displayStudent
            CASE 3:
                CALL searchStudent
            CASE 4:
                CALL deleteStudent
            DEFAULT:
                PRINT "Pilihan tidak valid."
        END SWITCH
    END WHILE
    RETURN status
END PROGRAM
```

- **Fungsi addStudent untuk menambahkan siswa**

```
FUNCTION addStudent
    DECLARE Students newStudent
    DECLARE CHAR inputlagi
    DECLARE SIZE_T counter = 0
    OPEN file "db.csv" for appending
    IF file is open THEN
        DO
            ALLOCATE memory for newStudent
            IF memory allocation fails THEN
                PRINT error message
                SET status to FAILED
            ENDIF

            CALL clrscr
            PRINT prompt for registration number
            READ input into newStudent[counter].nomor_registrasi

            PRINT prompt for student name
            READ input into newStudent[counter].nama

            PRINT prompt for student age
            READ input into newStudent[counter].umur

            PRINT prompt for adding more students
            READ input into inputlagi
            INCREMENT counter
        WHILE inputlagi is 'y' or 'Y'

        FOR i from 0 to counter-1 DO
            WRITE newStudent[i] to file
        END FOR

        FREE memory allocated for newStudent
        CLOSE file
```

```
    ELSE
        PRINT error message
        SET status to FAILED
    ENDIF
END FUNCTION
```

- **Fungsi displayStudent untuk menampilkan data siswa**

```
FUNCTION displayStudent
    DECLARE Students savedStudent
    DECLARE SIZE_T counter = 0
    DECLARE CHAR bufferStudents[256]
    OPEN file "db.csv" for reading
    IF file is open THEN
        SET records to countRecords(file)
        ALLOCATE memory for savedStudent

        IF memory allocation fails THEN
            PRINT error message
            SET status to FAILED
        ENDIF

        REWIND file
        PRINT header for student data
        WHILE READ line from file into bufferStudents DO
            PARSE bufferStudents into savedStudent[counter]
            INCREMENT counter
        END WHILE

        PRINT total records
        FOR i from 0 to counter-1 DO
            PRINT savedStudent[i] details
        END FOR
```

```
        FREE memory allocated for savedStudent
        PRINT prompt to return to main menu
        CLOSE file
    ELSE
        PRINT error message
        SET status to FAILED
    ENDIF
END FUNCTION
```

- **Fungsi searchStudent untuk mencari siswa berdasarkan nomor registrasi**

```
FUNCTION searchStudent
    DECLARE CHAR input_nomor_registrasi[12]
    DECLARE CHAR buffer[256]
    DECLARE SIZE_T counter = 0
    DECLARE Students savedStudent
    OPEN file "db.csv" for reading
    IF file is open THEN
        PRINT prompt for registration number
        READ input_nomor_registrasi

        SET records to countRecords(file)
        ALLOCATE memory for savedStudent

        IF memory allocation fails THEN
            PRINT error message
            SET status to FAILED
        ENDIF

        REWIND file
        WHILE READ line from file into buffer DO
            PARSE buffer into savedStudent[counter]
            INCREMENT counter
        END WHILE
```

```
        FOR i from 0 to records-1 DO
            IF input_nomor_registrasi matches savedStudent[i].nomor_registrasi THEN
                PRINT savedStudent[i] details
            ENDIF
        END FOR

        FREE memory allocated for savedStudent
        CLOSE file
        PRINT prompt to return to main menu
    ELSE
        PRINT error message
        SET status to FAILED
    ENDIF
END FUNCTION
```

- **Fungsi deleteStudent untuk menghapus data mahasiswa berdasarkan nomor registrasi**

```
FUNCTION deleteStudent
    DECLARE CHAR input_nomor_registrasi[11]
    DECLARE CHAR buffer[256]
    DECLARE SIZE_T counter = 0
    DECLARE SIZE_T found = 0
    DECLARE Students savedStudent
    DECLARE Students notDeletedStudent
    OPEN file "db.csv" for reading and writing
    IF file is open THEN
        PRINT prompt for registration number
        READ input_nomor_registrasi

        SET records to countRecords(file)
        ALLOCATE memory for savedStudent and notDeletedStudent

        IF memory allocation fails THEN
            PRINT error message
            SET status to FAILED
        ENDIF

        REWIND file
        WHILE READ line from file into buffer DO
            PARSE buffer into savedStudent[counter]
            INCREMENT counter
        END WHILE

        CLOSE file
```

```
        DECLARE SIZE_T counternotdeleted = 0
        FOR i from 0 to records-1 DO
            IF input_nomor_registrasi does not match savedStudent[i].nomor_registrasi
THEN
                COPY savedStudent[i] to notDeletedStudent[counternotdeleted]
                INCREMENT counternotdeleted
            ELSE
                PRINT savedStudent[i] details
                INCREMENT found
            ENDIF
        END FOR

        IF found == 0 THEN
            PRINT "Data tidak ditemukan"
        ELSE
            OPEN file "db.csv" for writing
            IF file is open THEN
                FOR i from 0 to counternotdeleted-1 DO
                    WRITE notDeletedStudent[i] to file
                END FOR
            ELSE
                PRINT error message
                SET status to FAILED
            ENDIF
        ENDIF

        FREE memory allocated for savedStudent and notDeletedStudent
        CLOSE file
        PRINT prompt to return to main menu
    ELSE
        PRINT error message
        SET status to FAILED
    ENDIF
END FUNCTION
```

2. **Struktur file yang digunakan**

   a. **Format file**

   - Type File = .csv yaitu file teks biasa yang menyimpan data dalam format tabel. Dalam file CSV, setiap baris mewakili satu *record* atau satu set informasi data, dan setiap kolom menandakan *attribute* atau bidang dalam baris tersebut. Kesederhanaannya membuatnya kompatibel dengan berbagai aplikasi dan program.
   - Nama File = db.csv

   b. **Format entry data**

   Terdapat 3 entry data yaitu :

   Terdapat 3 entry data yaitu :

   - nomor_registrasi: variabel yang merupakan character string yang berasal dari koleksi tipe data sejenis yaitu char dengan diakhiri dengan null terminantor '\0' yang memiliki ukuran maksimal 11 karakter yang digunakan untuk menyimpan Nomor Registrasi.
   - Nama: variabel yang merupakan character string yang berasal dari koleksi tipe data sejenis yaitu char yang diakhiri dengan null character/terminator. Variabel nama memiliki total panjang maksimal 70 karakter yang berisi nama mahasiswa.
   - Umur: tipe data integer yang menunjukan umur siswa.

   c. **Pemisahan data**

   Dalam data yang disimpan pada file .csv terdapat beberapa delimiter atau separator yang sering digunakan seperti comma, tab, dan semicolon. Namun, untuk menghindari error yang sering terjadi seperti nama siswa yang memiliki karakter spesial seperti koma maka kami menggunakan delimiter dalam bentuk pipe (|).

   d. **Baris Baru**

   Setiap entry data akan berada pada baris baru.

e. Pembacaan data

- Program membaca input menggunakan dua metode yaitu fgets dan scanf. fgets digunakan karena memiliki keunggulan lebih aman dan menghindari buffer overflow karena kita menuliskan maksimal ukuran yang dibaca sedangkan scanf dgunakan karena lebih praktis dan tidak merekam spasi atau newline karakter sehingga ketika terdapat spasi atau newline otomatis tidak akan terekam. fgets hanya digunakan untuk input dalam bentuk string sedangkan scanf dapat digunakan sesuai format specifier. Selain itu. terdapat juga getchar(). untuk menggunakan residu newline yang ada pada buffer.
- Data kemudian dilakukan operasi sesuai dengan kebutuhan yang terdapat dua jenis operasi yaitu **sscanf** dan **fprintf**. Setelah data setiap line dibaca dengan fgets atau scanf, sscanf digunakan untuk membaca stream dalam buffer sesuai dengan format yang ada dalam .csv yaitu "%10[^|]|%69[^|]|%d". format tersebut memililki arti 10 karakter pertama atau semua karakter sebelum pipe (|) pertama, 69 karakter berikutnya atau semuanya sebelum pipe (|) kedua dan yang terakhir adalah integer. Sedangkan fprintf digunakan untuk melakukan formating pada data dalam program atau buffer sebelum disimpan ke dalam file yaitu dengan mengurutkan dan menambahkan delimiter sebagai berikut "%s|%s|%d\n" yang memiliki arti string diikuti pipe(|) kemudian string lagi diikuti pipe(|) dan terakhir adalah integer untuk menyimpan secara berururan nomor registrasi, nama , dan umur.

3. Memory Management

Program ini menggunakan alokasi memori dinamis dengan fungsi calloc() dan realloc() untuk mengelola struktur data yang akan diolah.

Alokasi memori ini diperlukan karena jumlah data mahasiswa tidak diketahui sebelumnya, sehingga memori dialokasikan saat diperlukan:

a. Alokasi Memori Dinamis

Pada fungsi addStudent, memori dialokasikan secara dinamis untuk menyimpan data mahasiswa yang baru dengan menggunakan fungsi realloc. Memori bertambah setiap kali pengguna menambahkan entri baru. Setiap kali pengguna menambahkan data mahasiswa baru, program mengalokasikan blok memori baru untuk menampung data tersebut. Setelah semua data mahasiswa ditambahkan, memori ini dibebaskan dengan free.

Pada fungsi displayStudent, searchStudent, dan deleteStudent, memori dialokasikan secara dinamis menggunakan calloc untuk menyimpan data mahasiswa yang dibaca dari file. Alokasi memori ini dilakukan berdasarkan jumlah total catatan yang dihitung dengan fungsi countRecords. Setelah operasi selesai (misalnya, menampilkan atau menghapus data), memori yang dialokasikan dibebaskan dengan free.

Pada fungsi setMsgNotif, memori dialokasikan untuk menyimpan pesan notifikasi menggunakan calloc. Memori ini kemudian dibebaskan pada proses berikutnya dalam main ketika pesan baru akan ditampilkan, menggunakan free(msg_notif).

b. Manajemen Memori Selama Operasi File

Program ini bekerja dengan membaca dan menulis data ke file CSV (db.csv), di mana data mahasiswa disimpan. Berikut beberapa aspek manajemen memori saat program mengakses file

- **Membaca File**

Dalam fungsi seperti displayStudent, searchStudent, dan deleteStudent, file dibuka dengan mode baca (r). Program membaca file baris per baris menggunakan fgets, dan setiap baris disimpan sementara dalam buffer. Setelah semua data dimasukkan ke dalam struktur Students, buffer ini tidak perlu dikelola lebih lanjut, karena hanya digunakan untuk parsing sementara.

- **Menulis ke File**

Dalam fungsi seperti addStudent dan deleteStudent, file dibuka dalam mode tambah (a) atau tulis (w+). Data baru ditambahkan atau file ditulis ulang dengan data yang diperbarui. Memori yang dialokasikan untuk menyimpan data sementara akan dibebaskan dengan free setelah operasi selesai.

Program membaca seluruh file untuk menghitung jumlah baris (countRecords) atau catatan. Meskipun buffer sementara digunakan dalam proses ini, buffer ini tidak membutuhkan manajemen memori eksplisit karena penggunaannya sangat terbatas (dalam satu blok fungsi).

c. Pembebasan Memori

Setelah memori dialokasikan dan digunakan, penting untuk membebaskan memori yang tidak lagi diperlukan. Setelah operasi penulisan data selesai, program membebaskan memori yang dialokasikan untuk newStudent dengan memanggil free(newStudent). Hal ini menghindari kebocoran memori yang bisa terjadi jika memori tidak dibebaskan.

Begitu program selesai menampilkan data mahasiswa atau setelah pencarian mahasiswa selesai, memori untuk savedStudent juga dibebaskan menggunakan free(savedStudent). Ini penting untuk menjaga efisiensi penggunaan memori selama eksekusi program.

Tugas 2: Analisis Kesalahan Program untuk Alokasi Memori

1. Identifikasi dan Penjelasan Kesalahan dalam Pengelolaan Memori

Pada kode yang diberikan, terdapat kesalahan dalam pengelolaan memori yang disebut **memory leak**. Berikut adalah detail kesalahan tersebut:

- **Penggunaan malloc tanpa free**

  Kode menggunakan fungsi malloc untuk mengalokasikan memori untuk array grades, tetapi tidak ada pemanggilan free untuk membebaskan memori yang telah dialokasikan setelah penggunaannya. Hal ini menyebabkan memori yang tidak digunakan tetap teralokasi, yang dapat mengakibatkan peningkatan penggunaan memori dan menurunnya kinerja program seiring waktu.

2. Kode yang Sudah Diperbaiki

Berikut adalah kode yang sudah diperbaiki dengan menambahkan pemanggilan free untuk membebaskan memori:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *grades;
    int numStudents, i;

    printf("Enter number of students: ");
    scanf("%d", &numStudents);

    grades = malloc(numStudents * sizeof(int));  // Memory allocation

    if (grades == NULL) {
        // Check if malloc failed
        printf("Memory allocation failed!\n");
```

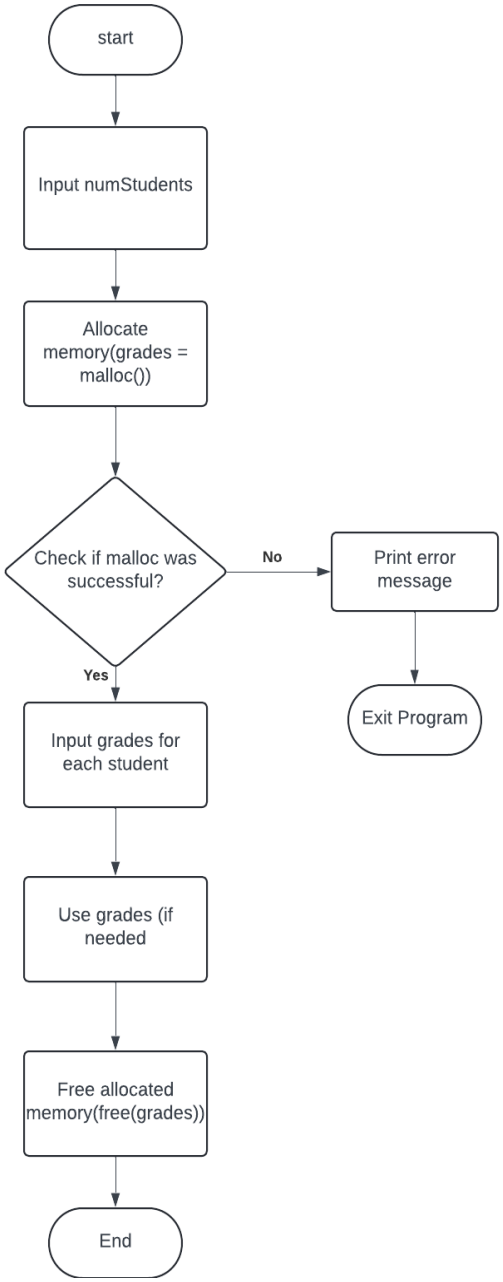
```
            return 1;
        }

        for(i = 0; i < numStudents; i++) {
            printf("Enter grade for student %d: ", i + 1);
            scanf("%d", &grades[i]);
        }

        // Free the allocated memory
        free(grades);

        return 0;
}
```

Dalam kode yang diperbaiki:

- **Pemeriksaan Kesuksesan Alokasi**
  Dilakukan penambahan pemeriksaan untuk memastikan bahwa malloc berhasil mengalokasikan memori. Jika tidak, program akan menampilkan pesan kesalahan dan keluar dengan kode kesalahan.
- **Pemanggilan free**
  Dilakukan penambahan free(grades); untuk membebaskan memori yang telah dialokasikan setelah penggunaannya selesai.

3. Gambaran Flowchart Proses yang Benar

Berikut adalah gambaran flowchart untuk menunjukkan proses pengelolaan memori yang benar dalam program:



Flowchart ini menggambarkan langkah-langkah yang diambil dalam program, termasuk pengelolaan memori yang benar dengan memeriksa hasil alokasi dan membebaskan memori setelah digunakan. Ini adalah praktik yang baik dalam pemrograman untuk menghindari memory leak.

BINUS UNIVERSITY ONLINE

People Innovation Excellence

Sumber referensi

ChatGPT pada link https://chatgpt.com/share/67107bd6-6974-800c-a877-e3b450b9368b diakses tanggal 17 Oktober 2024

GeeksforGeeks - Dynamic Memory Allocation in C using malloc(), calloc(), free() and realloc()

https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/ diakses tanggal 17 Oktober 2024

Cplusplus.com - C Library: malloc and free https://cplusplus.com/reference/cstdlib/free/#google_vignette diakses tanggal 17 Oktober 2024

Dynamic Allocation in C - https://www.scaler.com/topics/c/dynamic-memory-allocation-in-c/ diakses tanggal 19 Oktober 2024