**ENGLISH COURSE**

LAPORAN STUDI KASUS  
MATA KULIAH COMP6360004 – ALGORITHM AND PROGRAMMING  
KELAS BD20



Oleh :  
2602160750 – BRYAN ORVILLE AUDRIC  
2602163336 – DAVIN NAYAKA PANDYA  
2602152982 – JONATHAN EDMUND WILLIAM

Semester Ganjil 2022/2023

MALANG

1. **Gambaran Umum**

Studi kasus yang kami pilih adalah English Course. Dalam kasus ini kami membuat program berupa sistem penilaian siswa kursus secara sederhana yang dapat menampung data nilai kursus setiap siswanya. Data nilai kursus siswa tersebut diolah terlebih dahulu, lalu disimpan pada sebuah file bernama dataSiswa.txt.

Sistem yang kami buat terdiri dari kemampuan sebagai berikut :

1. Mengelola data siswa yang terbagi menjadi 3, yaitu membuat data baru, menambahkan data, dan menghapus data.
2. Mencari data berdasarkan kriteria yang diinginkan user.
3. Menyortir data sesuai ketentuan yang diinginkan user.

Dari 3 kemampuan pokok diatas dibuatlah 7 fitur dalam program kami sebagai berikut :

1. Create/Add data
2. Append data
3. Display data
4. Delete data
5. Update data
6. Sorting data
7. Search data
8. Exit

Program akan membaca, dan mengedit file yang bernama dataSiswa.txt. Berikut rincian cara kerja setiap fiturnya :

1. Membuat data file baru di file dataSiswa.txt
2. Menambahkan data baru pada file dataSiswa.txt tanpa menghapus data yang sudah ada
3. Membaca dataSiswa.txt, kemudian menampilkan data yang terdapat dalam dataSiswa.txt
4. Membaca dataSiswa.txt, Lalu menghapus data yang dipilih oleh user
5. Membaca dataSiswa.txt, Setelah itu, memilih data yang ingin diubah user. Kemudian mengubah sub-data sesuai keinginan user pada data yang telah dipilih sebelumnya.
6. Membaca dataSiswa.txt, melakukan sortir data sesuai keinginan user.
7. Membaca dataSiswa.txt, melakukan pencarian terhadap data yang ingin dicari oleh user.
8. Keluar program
9. **HASIL**

**Fitur-Fitur Aplikasi**

Dalam program ini tampilan awalya berupa menu utama dengan opsi yang dapat user pilih sesuai dengan apa yang user inginkan dengan menginput pilihan pada tempat yang sudah disediakan. Tampilan menu utama terlampir sebagai berikut :

Text

Description automatically generated

Program ini terdiri dari 7 opsi yakni :

1. Fitur Create/Add Data (Fitur Utama)

2. Fitur Append Data (Fitur Utama)

3. Fitur Display Data (Fitur Utama)

4. Fitur Delete Data (Fitur Utama)

5. Fitur Update Data (Fitur Tambahan)

6. Fitur Sorting Data (Fitur Utama)

7. Fitur Search Data (Fitur Utama)

1. Fitur Create/Add Data

Apabila user ingin membuat data baru, maka user bisa memilih opsi Create/Add Data pada nomor 1, Lalu akan muncul tampilan sebagai berikut :

Text

Description automatically generated

Disini user dapat menginput total berapa kali ia ingin memasukkan data siswa. Berikut contoh bila user ingin memasukkan 2 data siswa

Text

Description automatically generated

Setelah ini baru mulai proses input data. User dapat menginput ID, Nama siswa, dan nilai-nilai siswa satu per satu. Berikut tampilan yang akan muncul

Text

Description automatically generated

pada fase ini user dapat memasukkan data ID siswa. Setelah memasukkan ID dan menekan enter, User dapat menginput nama Siswa. Tampilannya sebagai berikut :

Text

Description automatically generated

Text

Description automatically generated

Kemudian user dapat memasukkan jenis course siswa tersebut, tampilan akan berubah sebagai berikut :

Text

Description automatically generated

Text

Description automatically generated

Kemudian user dapat memasukkan Nilai-nilai siswa tersebut, tampilan akan berubah sebagai berikut :

Pertama, memasukkan quiz score

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Kedua, memasukkan Assignment score

Text

Description automatically generated

Text

Description automatically generated

Ketiga, memasukkan Exam score

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Setelah menginput exam score, maka program akan mengulang atau looping untuk menginput data siswa lain yang ditandai dengan munculnya tampilan berikut :

Text

Description automatically generated

Berikut adalah contoh input data siswa kedua kedua

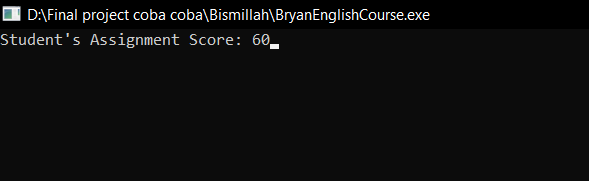
Shape

Description automatically generated with medium confidence

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Lalu, setelah user selesai menginput semua data dan menekan enter. Maka akan muncul tampilan seperti dibawah ini

Text

Description automatically generated

Pada fase ini user dapat memenekan tombol apapun untuk kembali ke menu utama

Berikut adalah isi data pada file dataSiswa.txt

Text

Description automatically generated

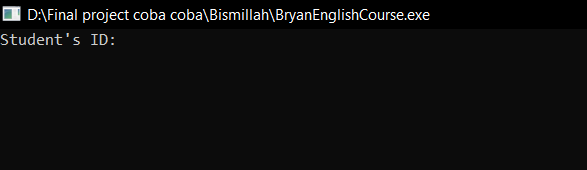
1. Fitur Append Data

Fitur ini dapat digunakan untuk menambah data siswa baru tanpa menghapus data yang sudah diinput sebelumnya. Tampilan tidak jauh berbeda dengan yang dimiliki oleh fitur *Create/Add Data.* berikut tampilan program yang muncul secara berurutan beserta contoh inputnya :

Text

Description automatically generated

Pertama user memasukkan ID siswa tambahan



Text

Description automatically generated

Lalu user memasukkan nama siswa tersebut

Text

Description automatically generated

Text

Description automatically generated

Setelah itu, User dapat memasukkan jenis Course yang diikuti siswa

Text

Description automatically generated

Text

Description automatically generated

Kemudian User dapat menginput nilai-nilai siswa tersebut

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Lalu Program Append Selesai dan User kembali ke menu Utama

Text

Description automatically generated

Berikut adalah screenshot file dataSiswa.txt setelah ditambahkan data baru menggunakan fitur append

Graphical user interface, text, application

Description automatically generated

1. Fitur Display

Fitur ini akan menampilkan data inputan yang tersimpan dalam file dataSiswa.txt. Data yang muncul dalam program kami bentuk berupa tabel agar terlihat lebih rapi dan mudah dibaca oleh user.

Berikut adalah data dalam file dataSiswa.txt :

Graphical user interface, text, application

Description automatically generated

Kemudian dibawah ini adalah tampilan di dalam program :

Graphical user interface

Description automatically generated with low confidence

Setelah ini, user dapat menekan tombol atau key apapun untuk kembali ke menu utama.

1. Fitur Delete Data

Fitur ini memungkinkan user untuk menghapus data siswa yang diinginkan. Dalam fitur ini terdapat mekanisme untuk menghapus data siswa dimana user perlu memasukkan ID dari siswa yang ingin dihapus, Kemudian akan berjalan program untuk menghapus data siswa dengan ID yang dipilih. Kemudian program fitur delete akan selesai berjalan dan untuk melihat hasilnya user dapat kembali menggunakan fitur Display.

Berikut adalah tampilan program secara berurutan :

Text

Description automatically generated

Kemudian akan muncul tampilan seperti terlampir dibawah

A screenshot of a computer screen

Description automatically generated with medium confidence

Kemudian user dapat memilih ID siswa mana yang ingin di hapus dan menginputnya

A picture containing text, screenshot, monitor

Description automatically generated

Lalu, user hanya perlu menekan tombol enter dan program fitur delete akan selesai berjalan

Text

Description automatically generated

Berikut adalah perubahan isi dalam file dataSiswa.txt :

Graphical user interface, text, application

Description automatically generated

Dan berikut terlampir tampilan ketika user ingin melihat perubahan data melalui program menggunakan fitur display :

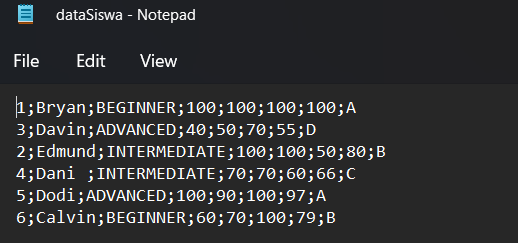
Graphical user interface

Description automatically generated with medium confidence

1. Fitur Update Data

Fitur ini terbagi menjadi beberapa sub-fitur dimana user dapat mengubah data yang diinginkan secara terperinci. User dapat memilih data mana yang ingin diubah dengan menginput ID siswa, lalu keluar menu yang memberikan opsi lagi yang berisi pilihan untuk mengubah isi data mulai dari nama, tiap-tiap nilai, dan course yang diikuti. Lalu apabila ada perubahan dibagian nilai, maka akan ada perubahan pada bagian final score dan grade. Semua hasil perubahan data bisa dilihat menggunakan fitur display di menu utama

Berikut isi data sebagai contoh dalam file dataSiswa.txt :

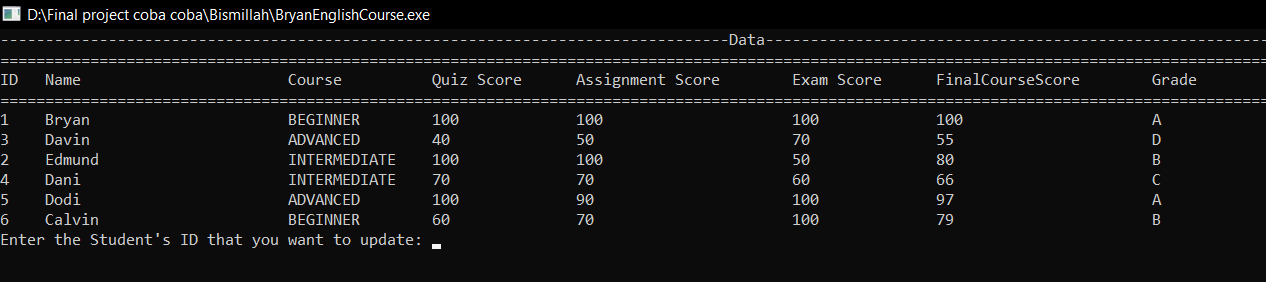


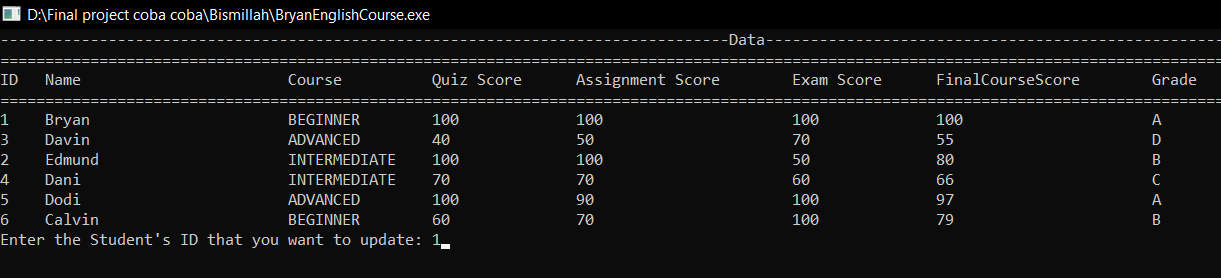
Berikut tampilan ketika memilih opsi update data

Text

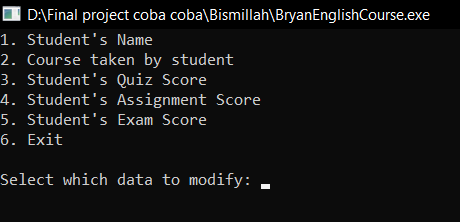
Description automatically generated

Berikut adalah tampilan dalam fitur update data





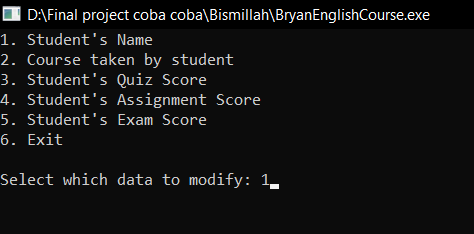
Misalkan user menginput [1] maka akan terpilih data dengan ID satu dan program akan berlanjut dengan menampilkan opsi-opsi bagian yang ingin diubah sebagai berikut



Berikut screenshot dan penjelasan setiap opsi :

1. Student’s name

Dengan opsi ini, user bisa mengubah nama siswa dalam data tersebut



Graphical user interface, text

Description automatically generated

Misalkan dalam contoh ini nama Bryan di ganti dengan Audric

Graphical user interface, text

Description automatically generated

Selanjutnya program akan kembali ke tampilan menu opsi bagian-bagian yang ingin di ubah.

1. Course taken by student

Dengan opsi ini, user bisa mengubah jenis course yang diikuti siswa dalam data tersebut

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Misalkan dalam contoh ini Jenis Coursenya diubah menjadi Advanced

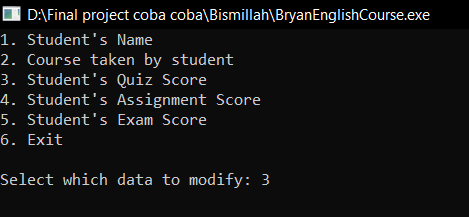
Text

Description automatically generated

Selanjutnya program akan kembali ke tampilan menu opsi bagian-bagian yang ingin di ubah.

1. Student’s Quiz Score

Dengan opsi ini, user bisa mengubah nilai Quiz siswa dalam data tersebut



Text

Description automatically generated

Misalkan dalam contoh ini nilai Quiz diubah menjadi 90

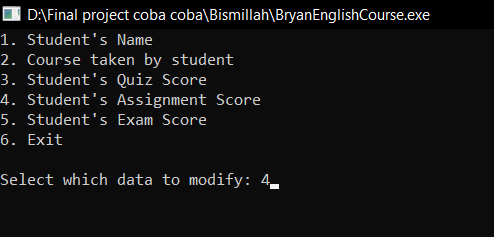
Graphical user interface, text

Description automatically generated

Selanjutnya program akan kembali ke tampilan menu opsi bagian-bagian yang ingin di ubah.

1. Student’s Assignment Score

Dengan opsi ini, user bisa mengubah nilai Assignment siswa dalam data tersebut



Graphical user interface, text

Description automatically generated

Misalkan dalam contoh ini nilai Assignment diubah menjadi 70

Text

Description automatically generated

Selanjutnya program akan kembali ke tampilan menu opsi bagian-bagian yang ingin di ubah.

1. Student’s Exam Score

Dengan opsi ini, user bisa mengubah nilai Exam siswa dalam data tersebut

Text

Description automatically generated

Text

Description automatically generated

Misalkan dalam contoh ini nilai Exam diubah menjadi 80

Text

Description automatically generated

Selanjutnya program akan kembali ke tampilan menu opsi bagian-bagian yang ingin di ubah.

1. Exit

Keluar dari program

Text

Description automatically generated

Text

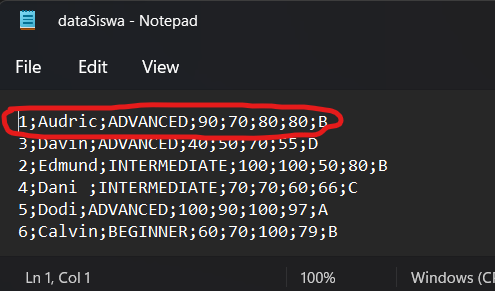
Description automatically generated

Setelah Exit, Tampilan akan kembali ke menu Utama di awal, untuk melihat hasil perubahan data dapat melalui fitur display, hasilnya sebagai berikut :

A screenshot of a computer

Description automatically generated with medium confidence

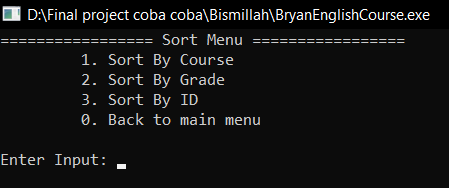
Data berhasil diubah, berikut terlampir screenshot isi dalam file dataSiswa.txt :



1. Fitur Sorting Data

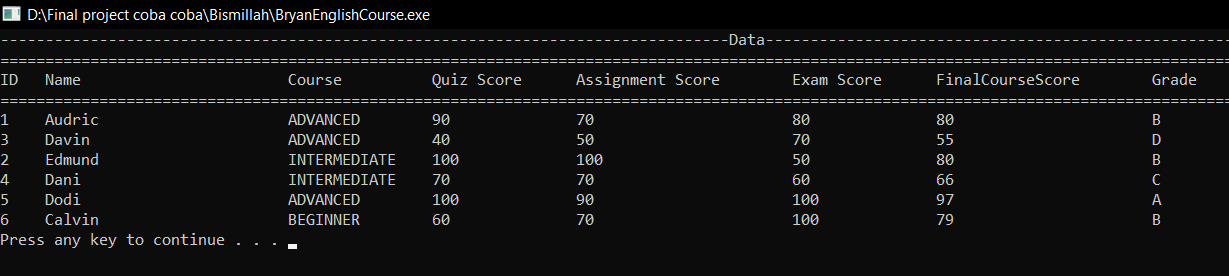
Text

Description automatically generated



Fitur ini digunakan untuk menyortir data berdasarkan kriteria sort yang diinginkan user, terdapat pilihan kriteria sorting, sort berdasarkan jenis Course, Grade, dan ID setiap siswa. Hasil dari sort bisa dilihat melalui fitur display.

Berikut adalah screenshot data awal :



Berikut terlampir screenshot hasil setiap sort :

1. Sort by Course

Text

Description automatically generated

A screenshot of a computer screen

Description automatically generated with medium confidence

1. Sort by Grade

Text

Description automatically generated

Text

Description automatically generated

1. Sort by ID

Text

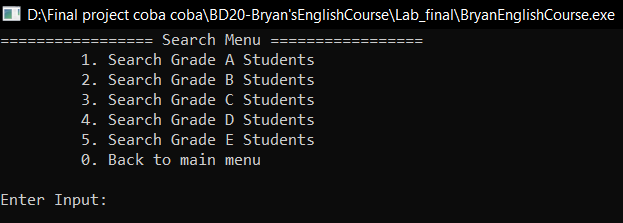
Description automatically generated

A screenshot of a computer screen

Description automatically generated with medium confidence

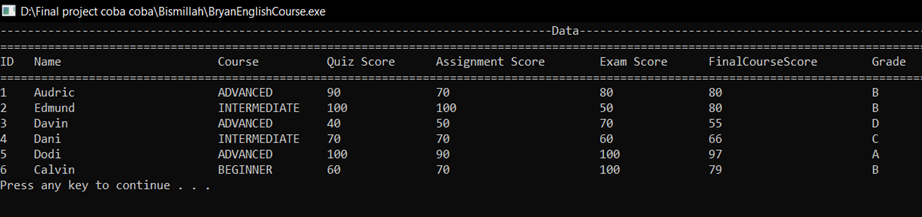
1. Fitur Search Data

Fitur ini menawarkan pencarian data berdasarkan grade yang didapat siswa



Setiap opsi hanya akan menampilkan data siswa sesuai grade yang diminta,

Sebagai contoh, berikut isi data yang terdapat dalam file :



Berikut tampilan apabila user memilih opsi 1. Search Grade A Students :

Graphical user interface, text

Description automatically generated

Berikut tampilan apabila user memilih opsi 2. Search Grade B Students:

A screenshot of a computer screen

Description automatically generated with medium confidence

Berikut tampilan apabila user memilih opsi 3. Search Grade C Students:

A screenshot of a computer

Description automatically generated with medium confidence

Berikut tampilan apabila user memilih opsi 4. Search Grade D Students:

Graphical user interface, text

Description automatically generated

Berikut tampilan apabila user memilih opsi 5. Search Grade E Students:

Graphical user interface, text

Description automatically generated

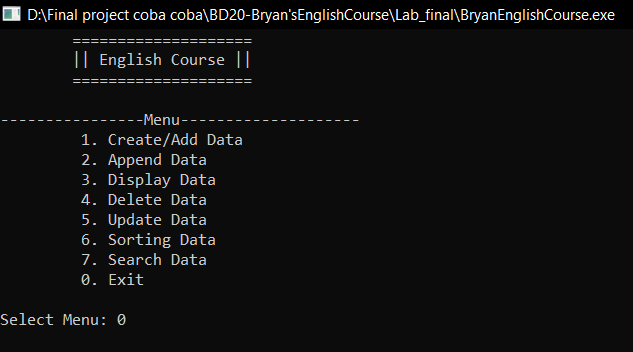
Pada gambar tampilan diatas, yang muncul hanya tabelnya saja karena tidak ada data siswa dengan grade E.

Kemudian User akan kembali ke menu utama apabila memilih opsi 0. Back to Menu

Text

Description automatically generated

Dan terakhir, berikut adalah tampilan apabila user memilih opsi 0.Exit pada menu utama



Graphical user interface

Description automatically generated with medium confidence

Dan akhirnya keseluruhan program telah selesai berjalan.

**Penjelasan Singkat Fungsi Dalam Program**

* createData(), adalah fungsi untuk membuat data.
* appendData(), adalah fungsi untuk menambahkan data baru.
* displayData() adalah fungsi untuk menampilkan data.
* deleteData(), adalah fungsi untuk menghapus data sesuai keinginan user.
* updateData(), adalah fungsi untuk mengubah data yang sudah ada.
* sortByCourse(), adalah fungsi untuk menyortir data berdasarkan jenis course yang diikuti masing-masing siswa.
* sortByGrade(), adalah fungsi untuk menyortir data berdasarkan grade setiap siswa.
* sortByID(), adalah fungsi untuk menyortir data berdasarkan ID setiap siswa.
* searchGradeA(), adalah fungsi untuk mencari data dengan grade A.
* searchGradeB(), adalah fungsi untuk mencari data dengan grade B.
* searchGradeC(), adalah fungsi untuk mencari data dengan grade C.
* searchGradeD(), adalah fungsi untuk mencari data dengan grade D.
* searchGradeE(), adalah fungsi untuk mencari data dengan grade E.
* sortMenu(), adalah fungsi untuk menampilkan menu sorting.
* searchMenu(), adalah fungsi untuk menampilkan menu search (pencarian).
* main(), adalah fungsi untuk menampilkan menu utama.

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

struct student{

int id;

char name[101];

char courseLevel[20];

int quizScore;

int assignmentScore;

int examScore;

int finalCourseScore;

char grade;

}studentData[21];

// Entry

void createData(){

// Declaring & opening file

FILE \*filePointer;

filePointer = fopen("dataSiswa.txt", "w");

// Check for file

if(filePointer == NULL){

printf("File not found\n");

}

// Initializing "amount" (number of data that is going in)

int amount;

printf("How much student's data you want to enter: ");

scanf("%d", &amount);

// Input Validation

// Denying negative as integer inputs

while(amount <= 0){

printf("ID cannot be negative or 0\n");

printf("Student's ID: ");

fflush(stdin);

scanf("%d", &amount);

}

// Loop for data input

for(int index = 0; index < amount; index++){

// ID Input

printf("Student's ID: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].id) != 1){

printf("ID can only be numbers\n");

printf("Student's ID: ");

fflush(stdin);

}

// Denying negative as integer inputs

while(studentData[index].id <= 0){

printf("ID cannot be negative or 0\n");

printf("Student's ID: ");

fflush(stdin);

scanf("%d", &studentData[index].id);

}

// Making sure that multiple id doesn't exist

if(index > 0){

for(int i = 1; i <= index; i++){

while(studentData[i].id == studentData[i-1].id){

printf("ID already existed\n");

printf("Student's ID: ");

fflush(stdin);

scanf("%d", &studentData[index].id);

}

}

}

system("cls");

// Student Name Input

printf("Student's name: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].name);

system("cls");

// Course Taken By Student Input

printf("Course taken by student: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].courseLevel);

getchar();

// Input Validation

// Making input to be uppercase

for(int i = 0; studentData[index].courseLevel[i] != '\0'; i++){

if(studentData[index].courseLevel[i] >= 'a' && studentData[index].courseLevel[i] <= 'z'){

studentData[index].courseLevel[i] = studentData[index].courseLevel[i] -32;

}

}

// Denying courses that doesn't exist

while(strcmp(studentData[index].courseLevel, "BEGINNER") != 0 &&

strcmp(studentData[index].courseLevel, "INTERMEDIATE") != 0 &&

strcmp(studentData[index].courseLevel, "ADVANCED") != 0){

printf("Course doesn't exist\n");

printf("Student's Course: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].courseLevel);

for(int i = 0; studentData[index].courseLevel[i] != '\0'; i++){

if(studentData[index].courseLevel[i] >= 'a' && studentData[index].courseLevel[i] <= 'z'){

studentData[index].courseLevel[i] = studentData[index].courseLevel[i] -32;

}

}

}

system("cls");

// Quiz Score Input

printf("Student's Quiz Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].quizScore) != 1){

printf("Quiz Score can only be numbers\n");

printf("Student's Quiz Score: ");

fflush(stdin);

}

// Denying negative score and score that is more than 100 as input

while(studentData[index].quizScore < 0 || (studentData[index].quizScore) > 100){

printf("Score cannot be negative or over 100\n");

printf("Student's Quiz Score: ");

fflush(stdin);

scanf("%d", &studentData[index].quizScore);

}

system("cls");

// Assignment Score Input

printf("Student's Assignment Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].assignmentScore) != 1){

printf("Assignment Score can only be numbers\n");

printf("Student's Assignment Score: ");

fflush(stdin);

}

// Denying negative score and score that is more than 100 as input

while(studentData[index].assignmentScore < 0 || studentData[index].assignmentScore > 100){

printf("Score cannot be negative or over 100\n");

printf("Student's Assignment Score: ");

fflush(stdin);

scanf("%d", &studentData[index].assignmentScore);

}

system("cls");

// Exam Score Input

printf("Student's Exam Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].examScore) != 1){

printf("Assignment Score can only be numbers\n");

printf("Student's Assignment Score: ");

fflush(stdin);

}

// Denying negative score and score that is more than 100 as input

while(studentData[index].examScore < 0 || studentData[index].examScore > 100){

printf("Score cannot be negative or over 100\n");

printf("Student's Exam Score: ");

fflush(stdin);

scanf("%d", &studentData[index].examScore);

}

system("cls");

// Input Final Course score Based On Calculation

studentData[index].finalCourseScore = ((0.3) \* (studentData[index].quizScore)) + ((0.3) \* (studentData[index].assignmentScore)) + ((0.4) \* (studentData[index].examScore));

// Input Student Grade Based On Grading Scale

if((studentData[index].finalCourseScore) >= 85 && (studentData[index].finalCourseScore) <= 100){

studentData[index].grade = 'A';

}

else if((studentData[index].finalCourseScore) >= 70 && (studentData[index].finalCourseScore) <= 84){

studentData[index].grade = 'B';

}

else if((studentData[index].finalCourseScore) >= 60 && (studentData[index].finalCourseScore) <= 69){

studentData[index].grade = 'C';

}

else if((studentData[index].finalCourseScore) >= 50 && (studentData[index].finalCourseScore) <= 59){

studentData[index].grade = 'D';

}

else if((studentData[index].finalCourseScore) >= 0 && (studentData[index].finalCourseScore) <= 49){

studentData[index].grade = 'E';

}

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

fclose(filePointer);

printf("---------------------Data Created Succesfully---------------------\n");

}

void appendData(){

int targetID, line = 0, index = 0, found = 0;

// Declaring & opening file

FILE \*filePointer;

filePointer = fopen("./dataSiswa.txt", "a+");

// Check for file

if(filePointer == NULL){

printf("File not found\n");

}

// ID Input

printf("Student's ID: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &targetID) != 1){

printf("ID can only be numbers\n");

printf("Student's ID: ");

fflush(stdin);

}

studentData[index].id = targetID;

// Denying negative as integer inputs

while(targetID <= 0){

printf("ID cannot be negative or 0\n");

printf("Student's ID: ");

fflush(stdin);

scanf("%d", &targetID);

}

studentData[index].id = targetID;

// Making sure that multiple id doesn't exist in data.txt

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore,

&studentData[index].grade) != EOF){

if(targetID == studentData[index].id){

found = 1;

break;

}

}

rewind(filePointer);

while(found == 1){

printf("Student's ID already exist\n");

printf("Student's ID: ");

fflush(stdin);

scanf("%d", &targetID);

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore,

&studentData[index].grade) != EOF){

while(targetID == studentData[index].id){

printf("Student's ID already exist\n");

printf("Student's ID: ");

fflush(stdin);

scanf("%d", &targetID);

break;

}

found = 0;

}

}

studentData[index].id = targetID;

system("cls");

// Student Name Input

printf("Student's name: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].name);

system("cls");

// Course Taken By Student Input

printf("Course taken by student: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].courseLevel);

getchar();

// Input Validation

// Making input to be uppercase

for(int i = 0; studentData[index].courseLevel[i] != '\0'; i++){

if(studentData[index].courseLevel[i] >= 'a' && studentData[index].courseLevel[i] <= 'z'){

studentData[index].courseLevel[i] = studentData[index].courseLevel[i] -32;

}

}

// Denying courses that doesn't exist

while(strcmp(studentData[index].courseLevel, "BEGINNER") != 0 &&

strcmp(studentData[index].courseLevel, "INTERMEDIATE") != 0 &&

strcmp(studentData[index].courseLevel, "ADVANCED") != 0){

printf("Course doesn't exist\n");

printf("Student's Course: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].courseLevel);

for(int i = 0; studentData[index].courseLevel[i] != '\0'; i++){

if(studentData[index].courseLevel[i] >= 'a' && studentData[index].courseLevel[i] <= 'z'){

studentData[index].courseLevel[i] = studentData[index].courseLevel[i] -32;

}

}

}

system("cls");

// Quiz Score Input

printf("Student's Quiz Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].quizScore) != 1){

printf("Quiz Score can only be numbers\n");

printf("Student's Quiz Score: ");

fflush(stdin);

}

// Denying negative score and score that is more than 100 as input

while(studentData[index].quizScore < 0 || (studentData[index].quizScore) > 100){

printf("Score cannot be negative or over 100\n");

printf("Student's Quiz Score: ");

fflush(stdin);

scanf("%d", &studentData[index].quizScore);

}

system("cls");

// Assignment Score Input

printf("Student's Assignment Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].assignmentScore) != 1){

printf("Assignment Score can only be numbers\n");

printf("Student's Assignment Score: ");

fflush(stdin);

}

// Denying negative score and score that is more than 100 as input

while(studentData[index].assignmentScore < 0 || studentData[index].assignmentScore > 100){

printf("Score cannot be negative or over 100\n");

printf("Student's Assignment Score: ");

fflush(stdin);

scanf("%d", &studentData[index].assignmentScore);

}

system("cls");

// Exam Score Input

printf("Exam Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].examScore) != 1){

printf("Assignment Score can only be numbers\n");

printf("Student's Assignment Score: ");

fflush(stdin);

}

// Denying negative score and score that is more than 100 as input

while(studentData[index].examScore < 0 || studentData[index].examScore > 100){

printf("Score cannot be negative or over 100\n");

printf("Student's Exam Score: ");

fflush(stdin);

scanf("%d", &studentData[index].examScore);

}

system("cls");

// Input Final Course score Based On Calculation

studentData[index].finalCourseScore = ((0.3) \* (studentData[index].quizScore)) + ((0.3) \* (studentData[index].assignmentScore)) + ((0.4) \* (studentData[index].examScore));

// Input Student Grade Based On Grading Scale

if((studentData[index].finalCourseScore) >= 85 && (studentData[index].finalCourseScore) <= 100){

studentData[index].grade = 'A';

}

else if((studentData[index].finalCourseScore) >= 70 && (studentData[index].finalCourseScore) <= 84){

studentData[index].grade = 'B';

}

else if((studentData[index].finalCourseScore) >= 60 && (studentData[index].finalCourseScore) <= 69){

studentData[index].grade = 'C';

}

else if((studentData[index].finalCourseScore) >= 50 && (studentData[index].finalCourseScore) <= 59){

studentData[index].grade = 'D';

}

else if((studentData[index].finalCourseScore) >= 0 && (studentData[index].finalCourseScore) <= 49){

studentData[index].grade = 'E';

}

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

fclose(filePointer);

printf("---------------------Data Added Succesfully---------------------\n");

}

void displayData(){

// Declaring & opening file

FILE \*filePointer;

filePointer = fopen("dataSiswa.txt", "r");

// Check for file

if(filePointer == NULL){

printf("File not found\n");

}

// Menu

printf("---------------------------------------------------------------------------------Data-----------------------------------------------------------------------------------\n");

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

printf("ID Name\t\t\tCourse\t\tQuiz Score\tAssignment Score\tExam Score\tFinalCourseScore\tGrade\n");

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

// Display Algorithm

for(int index = 0; (fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF); index++){

printf("%-5d%-20s\t%s\t%d\t\t%d\t\t\t%d\t\t%d\t\t\t%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

fclose(filePointer);

}

void deleteData(){

// Declaring variables

int targetID, line = 0, found = 0, index = 0;

// Declaring & opening file

FILE \*filePointer;

FILE \*tempfilePointer;

filePointer = fopen("dataSiswa.txt", "r");

tempfilePointer = fopen("temp.txt", "w");

// Display function so user can see what's happening

displayData();

// Asking for target id

printf("Enter the ID that you want to delete: ");

scanf("%d", &targetID);

getchar();

// Searching Algorithm (skipping the targeted line)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore,

&studentData[index].grade) != EOF){

line++;

if(studentData[index].id == targetID){

found = 1;

break;

}

}

rewind(filePointer);

if(found){

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore,

&studentData[index].grade) != EOF){

line--;

if (line == 0) continue;

fprintf(tempfilePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

fclose(filePointer);

fclose(tempfilePointer);

remove("dataSiswa.txt");

rename("temp.txt", "dataSiswa.txt");

system("cls");

printf("---------------------Data Deleted Succesfully---------------------\n");

}

else{

fclose(filePointer);

fclose(tempfilePointer);

remove("dataSiswa.txt");

rename("temp.txt", "dataSiswa.txt");

system("cls");

printf("---------------------Data not found---------------------\n");

}

}

void updateData(){

// Declaring variables

int targetID, line = 0, found = 0, index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*ftemp;

filePointer = fopen("dataSiswa.txt", "r");

ftemp = fopen("temp.txt", "w");

// Display function so user can see what's happening

displayData();

// Asking for target id

printf("Enter the Student's ID that you want to update: ");

scanf("%d", &targetID);

getchar();

// Searching Algorithm (Also asking for inputs to modify the data)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].id == targetID){

found = 1;

fflush(stdin);

int userChoice = 0;

do{

system("cls");

printf("1. Student's Name\n");

printf("2. Course taken by student\n");

printf("3. Student's Quiz Score\n");

printf("4. Student's Assignment Score\n");

printf("5. Student's Exam Score\n");

printf("6. Exit\n\n");

printf("Select which data to modify: ");

fflush(stdin);

scanf("%d", &userChoice);

getchar();

switch (userChoice){

case 1:{

// Asking for new name

system("cls");

printf("New Student's Name: ");

fflush(stdin);

scanf("%[^\n]", studentData[index].name);

system("pause");

break;

}

case 2:{

// Asking for new course data

system("cls");

printf("New Student's Course: ");

fflush(stdin);

scanf("%[^\n]", studentData[index].courseLevel);

// Input Validation

// Making inputted data to be uppercase

for(int i = 0; studentData[index].courseLevel[i] != '\0'; i++){

if(studentData[index].courseLevel[i] >= 'a' && studentData[index].courseLevel[i] <= 'z'){

studentData[index].courseLevel[i] = studentData[index].courseLevel[i] -32;

}

}

// Making sure inputted course exist

while(strcmp(studentData[index].courseLevel, "BEGINNER") != 0 &&

strcmp(studentData[index].courseLevel, "INTERMEDIATE") != 0 &&

strcmp(studentData[index].courseLevel, "ADVANCED") != 0 ){

printf("Course doesn't exist\n");

printf("New Course: ");

fflush(stdin);

scanf("%[^\n]s", studentData[index].courseLevel);

for(int i = 0; studentData[index].courseLevel[i] != '\0'; i++){

if(studentData[index].courseLevel[i] >= 'a' && studentData[index].courseLevel[i] <= 'z'){

studentData[index].courseLevel[i] = studentData[index].courseLevel[i] -32;

}

}

}

system("pause");

break;

}

case 3:{

// Input & Validation

system("cls");

printf("New Quiz Score: ");

fflush(stdin);

while(scanf("%d", &studentData[index].quizScore) != 1){

printf("Quiz Score can only be numbers\n");

printf("New Quiz Score: ");

fflush(stdin);

}

while(studentData[index].quizScore < 0 || (studentData[index].quizScore) > 100){

printf("Score cannot be negative or over 100\n");

printf("New Quiz Score: ");

fflush(stdin);

scanf("%d", &studentData[index].quizScore);

}

}

// Input Final Course Score Based On Calculation

studentData[index].finalCourseScore = ((0.3) \* (studentData[index].quizScore)) + ((0.3) \* (studentData[index].assignmentScore)) + ((0.4) \* (studentData[index].examScore));

// Input Student Grade Based On Grading Scale

if((studentData[index].finalCourseScore) >= 85 && (studentData[index].finalCourseScore) <= 100){

studentData[index].grade = 'A';

}

else if((studentData[index].finalCourseScore) >= 70 && (studentData[index].finalCourseScore) <= 84){

studentData[index].grade = 'B';

}

else if((studentData[index].finalCourseScore) >= 60 && (studentData[index].finalCourseScore) <= 69){

studentData[index].grade = 'C';

}

else if((studentData[index].finalCourseScore) >= 50 && (studentData[index].finalCourseScore) <= 59){

studentData[index].grade = 'D';

}

else if((studentData[index].finalCourseScore) >= 0 && (studentData[index].finalCourseScore) <= 49){

studentData[index].grade = 'E';

}

system("pause");

break;

case 4:

system("cls");

printf("New Assignment Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].assignmentScore) != 1){

printf("Assignment Score can only be numbers\n");

printf("New Assignment Score: ");

fflush(stdin);

}

// Denying negative integers and integers that is over 100

while(studentData[index].assignmentScore < 0 || studentData[index].assignmentScore > 100){

printf("Score cannot be negative or over 100\n");

printf("New Assignment Score: ");

fflush(stdin);

scanf("%d", &studentData[index].assignmentScore);

}

// Input Final Course Score Based On Calculation

studentData[index].finalCourseScore = ((0.3) \* (studentData[index].quizScore)) + ((0.3) \* (studentData[index].assignmentScore)) + ((0.4) \* (studentData[index].examScore));

// Input Student Grade If met the condition

if((studentData[index].finalCourseScore) >= 85 && (studentData[index].finalCourseScore) <= 100){

studentData[index].grade = 'A';

}

else if((studentData[index].finalCourseScore) >= 70 && (studentData[index].finalCourseScore) <= 84){

studentData[index].grade = 'B';

}

else if((studentData[index].finalCourseScore) >= 60 && (studentData[index].finalCourseScore) <= 69){

studentData[index].grade = 'C';

}

else if((studentData[index].finalCourseScore) >= 50 && (studentData[index].finalCourseScore) <= 59){

studentData[index].grade = 'D';

}

else if((studentData[index].finalCourseScore) >= 0 && (studentData[index].finalCourseScore) <= 49){

studentData[index].grade = 'E';

}

system("pause");

break;

case 5:

system("cls");

printf("New Exam Score: ");

fflush(stdin);

// Input Validation

// Denying string as inputs

while(scanf("%d", &studentData[index].examScore) != 1){

printf("Exam Score can only be numbers\n");

printf("New Exam Score: ");

fflush(stdin);

}

// Denying negative integers and integers that is over 100

while(studentData[index].examScore < 0 || studentData[index].examScore > 100){

printf("Score cannot be negative or over 100\n");

printf("New Exam Score: ");

fflush(stdin);

scanf("%d", &studentData[index].examScore);

}

// Input Final Course Score Based On Calculation

studentData[index].finalCourseScore = ((0.3) \* (studentData[index].quizScore)) + ((0.3) \* (studentData[index].assignmentScore)) + ((0.4) \* (studentData[index].examScore));

// Input Student Grade Based On Grading Scale

if((studentData[index].finalCourseScore) >= 85 && (studentData[index].finalCourseScore) <= 100){

studentData[index].grade = 'A';

}

else if((studentData[index].finalCourseScore) >= 70 && (studentData[index].finalCourseScore) <= 84){

studentData[index].grade = 'B';

}

else if((studentData[index].finalCourseScore) >= 60 && (studentData[index].finalCourseScore) <= 69){

studentData[index].grade = 'C';

}

else if((studentData[index].finalCourseScore) >= 50 && (studentData[index].finalCourseScore) <= 59){

studentData[index].grade = 'D';

}

else if((studentData[index].finalCourseScore) >= 0 && (studentData[index].finalCourseScore) <= 49){

studentData[index].grade = 'E';

}

system("pause");

break;

case 6:

system("cls");

printf("------------------------------Data Updated-----------------------------\n");

break;

default:

printf("Error");

break;

}

}while(userChoice != 6);

}

fprintf(ftemp,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

// Closing file for switching mode

fclose(filePointer);

fclose(ftemp);

if(found){

filePointer = fopen("dataSiswa.txt", "w");

ftemp = fopen("temp.txt", "r");

while(fscanf(ftemp,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

fclose(filePointer);

fclose(ftemp);

}

remove ("temp.txt");

}

// Sort

void sortByCourse(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

FILE \*tempFilePointer2;

FILE \*tempFilePointer3;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

tempFilePointer2 = fopen("temp2.txt", "w");

tempFilePointer3 = fopen("temp3.txt", "w");

// Sorting Algorithm

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(strcmp(studentData[index].courseLevel,"BEGINNER") == 0){

fprintf(tempFilePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

else if(strcmp(studentData[index].courseLevel,"INTERMEDIATE") == 0){

fprintf(tempFilePointer2,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

else if(strcmp(studentData[index].courseLevel,"ADVANCED") == 0){

fprintf(tempFilePointer3,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

// Closing file for switching data

fclose(filePointer);

fclose(tempFilePointer);

fclose(tempFilePointer2);

fclose(tempFilePointer3);

// Opening file with different mode

filePointer = fopen("CourseSorteddataSiswa.txt","a");

tempFilePointer = fopen("temp.txt", "r");

tempFilePointer2 = fopen("temp2.txt", "r");

tempFilePointer3 = fopen("temp3.txt", "r");

// Merging Algorithm

while(fscanf(tempFilePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

while(fscanf(tempFilePointer2,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

while(fscanf(tempFilePointer3,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

// Closing File

fclose(filePointer);

fclose(tempFilePointer);

fclose(tempFilePointer2);

fclose(tempFilePointer3);

// Remove & rename as a part of "merging" algorithm

remove("temp.txt");

remove("temp2.txt");

remove("temp3.txt");

remove("dataSiswa.txt");

rename("CourseSorteddataSiswa.txt", "dataSiswa.txt");

}

void sortByGrade(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

FILE \*tempFilePointer2;

FILE \*tempFilePointer3;

FILE \*tempFilePointer4;

FILE \*tempFilePointer5;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

tempFilePointer2 = fopen("temp2.txt", "w");

tempFilePointer3 = fopen("temp3.txt", "w");

tempFilePointer4 = fopen("temp4.txt", "w");

tempFilePointer5 = fopen("temp5.txt", "w");

// Divide & Conquer Algorithm to seperate data

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].grade == 'A'){

fprintf(tempFilePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

else if(studentData[index].grade == 'B'){

fprintf(tempFilePointer2,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

else if(studentData[index].grade == 'C'){

fprintf(tempFilePointer3,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

else if(studentData[index].grade == 'D'){

fprintf(tempFilePointer4,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

else if(studentData[index].grade == 'E'){

fprintf(tempFilePointer5,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

// Closing file to switch mode

fclose(filePointer);

fclose(tempFilePointer);

fclose(tempFilePointer2);

fclose(tempFilePointer3);

fclose(tempFilePointer4);

fclose(tempFilePointer5);

// Switching file mode

filePointer = fopen("GradeSorteddataSiswa.txt","a");

tempFilePointer = fopen("temp.txt", "r");

tempFilePointer2 = fopen("temp2.txt", "r");

tempFilePointer3 = fopen("temp3.txt", "r");

tempFilePointer4 = fopen("temp4.txt", "r");

tempFilePointer5 = fopen("temp5.txt", "r");

// Merge Data that have been sorted out

while(fscanf(tempFilePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

while(fscanf(tempFilePointer2,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

while(fscanf(tempFilePointer3,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

while(fscanf(tempFilePointer4,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

while(fscanf(tempFilePointer5,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

fprintf(filePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

// Closing File

fclose(filePointer);

fclose(tempFilePointer);

fclose(tempFilePointer2);

fclose(tempFilePointer3);

fclose(tempFilePointer4);

fclose(tempFilePointer5);

// Remove old data & Rename temp File

remove("temp.txt");

remove("temp2.txt");

remove("temp3.txt");

remove("temp4.txt");

remove("temp5.txt");

remove("dataSiswa.txt");

rename("GradeSorteddataSiswa.txt", "dataSiswa.txt");

}

void sortByID(){

int arraySize;

// Opening & Declaring File

FILE \*filePointer;

FILE \*tempFilePointer;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

// Searching array size

for(int index = 0; (fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF); index++){

arraySize = index;

}

arraySize++;

// Bubble Sort Algorithm

int i, j;

struct student temp;

for (i = 1; i < arraySize - 1; i++){

for (j = 0; j < arraySize - i; j++){

if (studentData[j+1].id < studentData[j].id){

temp = studentData[j];

studentData[j] = studentData[j+1];

studentData[j+1] = temp;

}

}

}

// Printing sorted data back into file

for(int index = 0; index < arraySize; index++){

fprintf(tempFilePointer,"%d;%s;%s;%d;%d;%d;%d;%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

// Closing File, remove old data, and rename temp file

fclose(filePointer);

fclose(tempFilePointer);

remove("dataSiswa.txt");

rename("temp.txt", "dataSiswa.txt");

}

// Search

void searchGradeA(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

// Menu

printf("---------------------------------------------------------------------------------Data-----------------------------------------------------------------------------------\n");

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

printf("ID Name\t\t\tCourse\t\tQuiz Score\tAssignment Score\tExam Score\tFinalCourseScore\tGrade\n");

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

// Searching Algorithm (Print only selected data)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].grade == 'A'){

printf("%-5d%-20s\t%s\t%d\t\t%d\t\t\t%d\t\t%d\t\t\t%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

fclose(filePointer);

fclose(tempFilePointer);

remove("temp.txt");

}

void searchGradeB(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

// Menu

printf("---------------------------------------------------------------------------------Data-----------------------------------------------------------------------------------\n");

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

printf("ID Name\t\t\tCourse\t\tQuiz Score\tAssignment Score\tExam Score\tFinalCourseScore\tGrade\n");

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

// Searching Algorithm (Print only selected data)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].grade == 'B'){

printf("%-5d%-20s\t%s\t%d\t\t%d\t\t\t%d\t\t%d\t\t\t%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

fclose(filePointer);

fclose(tempFilePointer);

remove("temp.txt");

}

void searchGradeC(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

// Menu

printf("---------------------------------------------------------------------------------Data-----------------------------------------------------------------------------------\n");

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

printf("ID Name\t\t\tCourse\t\tQuiz Score\tAssignment Score\tExam Score\tFinalCourseScore\tGrade\n");

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

// Searching Algorithm (Print only selected data)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].grade == 'C'){

printf("%-5d%-20s\t%s\t%d\t\t%d\t\t\t%d\t\t%d\t\t\t%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

fclose(filePointer);

fclose(tempFilePointer);

remove("temp.txt");

}

void searchGradeD(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

// Menu

printf("---------------------------------------------------------------------------------Data-----------------------------------------------------------------------------------\n");

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

printf("ID Name\t\t\tCourse\t\tQuiz Score\tAssignment Score\tExam Score\tFinalCourseScore\tGrade\n");

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

// Searching Algorithm (Print only selected data)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].grade == 'D'){

printf("%-5d%-20s\t%s\t%d\t\t%d\t\t\t%d\t\t%d\t\t\t%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

fclose(filePointer);

fclose(tempFilePointer);

remove("temp.txt");

}

void searchGradeE(){

int index = 0;

// Declaring & Opening File

FILE \*filePointer;

FILE \*tempFilePointer;

filePointer = fopen("dataSiswa.txt","r");

tempFilePointer = fopen("temp.txt", "w");

// Menu

printf("---------------------------------------------------------------------------------Data-----------------------------------------------------------------------------------\n");

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

printf("ID Name\t\t\tCourse\t\tQuiz Score\tAssignment Score\tExam Score\tFinalCourseScore\tGrade\n");

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

// Searching Algorithm (Print only selected data)

while(fscanf(filePointer,"%d;%[^\';'];%[^\';'];%d;%d;%d;%d;%c\r\n",

&studentData[index].id, studentData[index].name, studentData[index].courseLevel, &studentData[index].quizScore,

&studentData[index].assignmentScore, &studentData[index].examScore, &studentData[index].finalCourseScore, &studentData[index].grade) != EOF){

if(studentData[index].grade == 'E'){

printf("%-5d%-20s\t%s\t%d\t\t%d\t\t\t%d\t\t%d\t\t\t%c\n",

studentData[index].id, studentData[index].name, studentData[index].courseLevel, studentData[index].quizScore,

studentData[index].assignmentScore, studentData[index].examScore, studentData[index].finalCourseScore, studentData[index].grade);

}

index++;

}

fclose(filePointer);

fclose(tempFilePointer);

remove("temp.txt");

}

// Menu

void sortMenu(){

int userChoice = -1;

do{

printf("================= Sort Menu =================\n");

printf("\t 1. Sort By Course\n");

printf("\t 2. Sort By Grade\n");

printf("\t 3. Sort By ID\n");

printf("\t 0. Back to main menu\n\n");

printf("Enter Input: ");

scanf("%d", &userChoice);

getchar();

switch (userChoice){

case 1:{

system("cls");

sortByCourse();

printf("---------------------Data Sorted Succesfully---------------------\n");

system("pause");

system("cls");

break;

}

case 2:{

system("cls");

sortByGrade();

printf("---------------------Data Sorted Succesfully---------------------\n");

system("pause");

system("cls");

break;

}

case 3:{

system("cls");

sortByID();

printf("---------------------Data Sorted Succesfully---------------------\n");

system("pause");

system("cls");

break;

}

case 0:{

system("cls");

printf("---------------------Data Sorted Succesfully---------------------\n");

system("pause");

system("cls");

return;

}

default:{

printf("Input error, please try again\n");

break;

}

}

}while(userChoice != 0);

}

void searchMenu(){

int userChoice = -1;

do{

printf("================= Search Menu =================\n");

printf("\t 1. Search Grade A Students\n");

printf("\t 2. Search Grade B Students\n");

printf("\t 3. Search Grade C Students\n");

printf("\t 4. Search Grade D Students\n");

printf("\t 5. Search Grade E Students\n");

printf("\t 0. Back to main menu\n\n");

printf("Enter Input: ");

scanf("%d", &userChoice);

getchar();

switch (userChoice){

case 1:{

system("cls");

searchGradeA();

system("pause");

system("cls");

break;

}

case 2:{

system("cls");

searchGradeB();

system("pause");

system("cls");

break;

}

case 3:{

system("cls");

searchGradeC();

system("pause");

system("cls");

break;

}

case 4:{

system("cls");

searchGradeD();

system("pause");

system("cls");

break;

}

case 5:{

system("cls");

searchGradeE();

system("pause");

system("cls");

break;

}

case 0:{

system("cls");

printf("---------------------Going Back To Main Menu---------------------\n");

system("pause");

system("cls");

return;

}

default:{

printf("Input error, please try again\n");

break;

}

}

}while(userChoice != 0);

}

int main(){

int menu;

printf(" \t====================\n");

printf("\t|| English Course ||\n");

printf(" \t====================\n\n");

do{

printf("----------------Menu--------------------\n");

printf("\t 1. Create/Add Data\n");

printf("\t 2. Append Data\n");

printf("\t 3. Display Data\n");

printf("\t 4. Delete Data\n");

printf("\t 5. Update Data\n");

printf("\t 6. Sorting Data\n");

printf("\t 7. Search Data\n");

printf("\t 0. Exit\n");

printf("\nSelect Menu: ");

scanf("%d", &menu);

getchar();

switch(menu){

case 1:{

system("cls");

createData();

system("pause");

system("cls");

break;

}

case 2:{

system("cls");

appendData();

system("pause");

system("cls");

break;

}

case 3:{

system("cls");

displayData();

system("pause");

system("cls");

break;

}

case 4:{

system("cls");

deleteData();

system("pause");

system("cls");

break;

}

case 5:{

system("cls");

updateData();

system("pause");

system("cls");

break;

}

case 6:{

system("cls");

sortMenu();

system("cls");

break;

}

case 7:{

system("cls");

searchMenu();

system("cls");

break;

}

case 0:{

system("cls");

printf("---------------------GOOD BYE!!!---------------------\n");

system("pause");

system("cls");

break;

}

}

}while(menu!=0)

return 0;

}

1. **Pembagian Tugas**

* Bryan Orville A. (ketua) :
* Design Program secara rinci, beserta algoritma
* Debugging dan Problem Solving
* Jonathan Edmund W. :
* Coding utama
* Debugging
* Davin Nayaka P. :
* Design garis besar program
* Debugging
* Laporan