

**VALORANT TRACKER**

LAPORAN STUDI KASUS TUGAS AOL

MATA KULIAH COMP6360004 – ALGORITHM AND PROGRAMMING KELAS LB20

Oleh :

2602160750 – Bryan Orville Audric

Semester Ganjil 2022/2023

MALANG

Daftar Isi

Bab 1 Pendahuluan………………………………………………………………………………………………...3

Bab 2 Usulan Desain Program…………………………………………………………………………………….4-14

Bab 3 Program dan Penjelasannya………………………………………………………………………………..15-40

Bab 4 Tampilan Hasil……………………………………………………………………………………………..41-54

Bab 5 Kesimpulan…………………………………………………………………………………………………55

BAB 1

Pendahuluan

Ide Project – Valorant Tracker

Valorant tracker adalah sebuah program yang didesain untuk menyimpan sekaligus menglist beberapa data player valorant. Valorant tracker dibuat agar player bisa menyimpan sekaligus melihat beberapa statistik *ingame* yang mereka miliki. Pembuatan program Valorant Tracker menggunakan bahasa pemrograman c. Berikut adalah beberapa unsur yang digunakan saat membuat program Valorant Tracker:

File Processing

File Processing digunakan untuk menambah, membaca, sekaligus mengubah data yang telah diinput player saat membuka program

Media Penyimpanan

Media penyimpanan yang digunakan untuk menyimpan data yang telah diinput oleh player adalah file txt yang berjudul “Account.txt” dan file txt yang berjudul “Player.txt”. “Account.txt” digunakan untuk menyimpan data akun user sedangkan “Player.txt” digunakan untuk menyimpan *ingame* data dari player.

Tipe Data

“Account.txt” menyimpan beberapa data seperti username, password, ingame name serta tagline. Semua data tersebut disimpan dengan tipe data string. “Player.txt” menyimpan beberapa data seperti ingame name, tagline, dan rank dengan tipe data string sedangkan data seperti jumlah kill, jumlah death, perbandingan antara jumlah kill dan jumlah death (K/D ratio), jumlah menang, jumlah kalah, serta perbandingan antara jumlah menang dan jumlah kalah (winrate) disimpan dengan tipe data float.

Struct

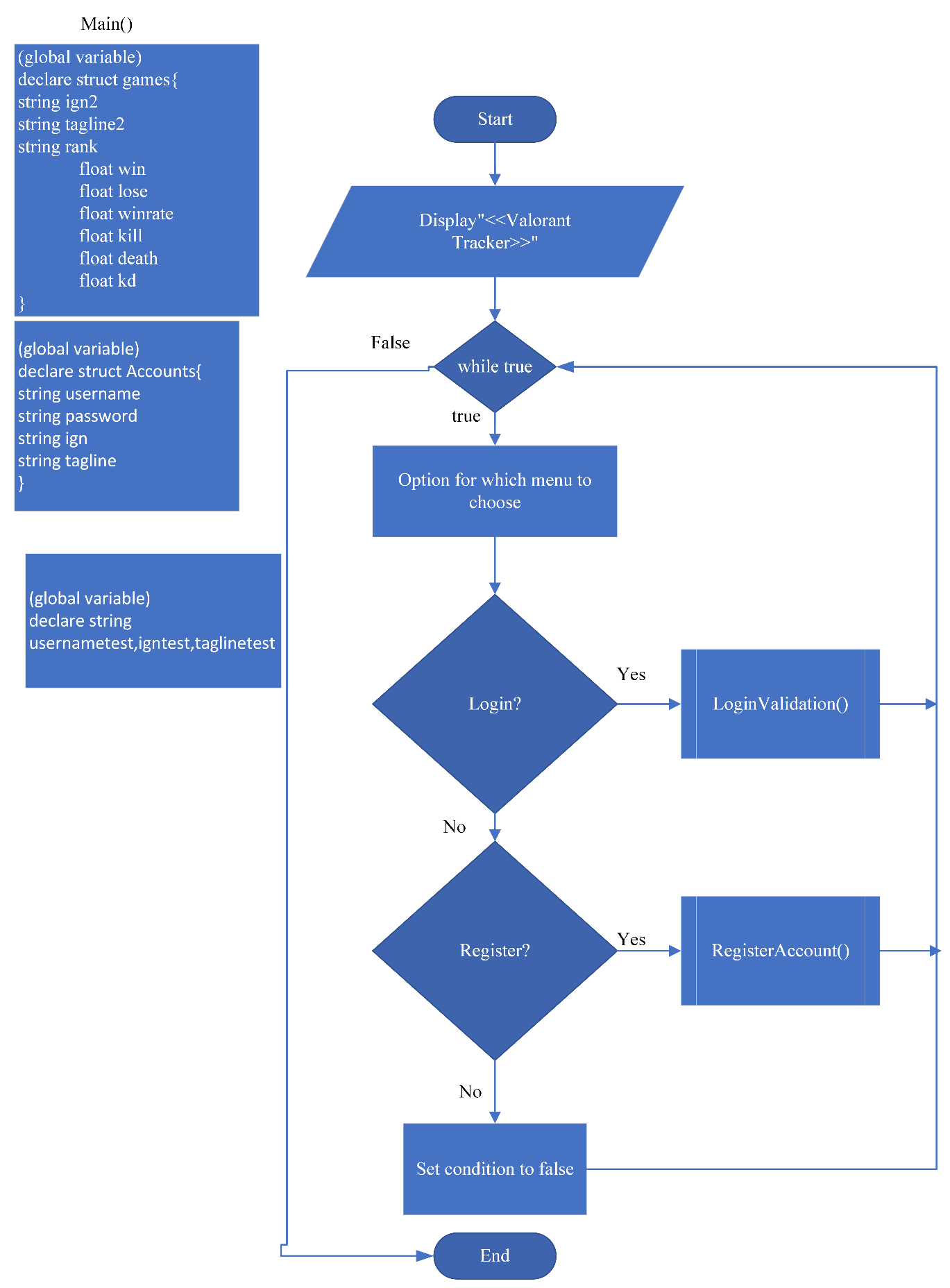
Struct digunakan untuk menyimpan data yang telah dibaca dari file “Account.txt” dan “Player.txt”. Ada dua struct yang digunakan yaitu struct untuk menyimpan data dari file “Account.txt” yang berisi username, password, ingame name serta tagline dan struct untuk menyimpan data dari file “Player.txt” yang berisi ingame name, tagline, jumlah kill, jumlah death, perbandingan antara jumlah kill dan jumlah death(K/D ratio), jumlah menang, jumlah kalah serta perbandingan antara jumlah menang dan kalah (winrate).

Selection

Program Valorant Tracker memiliki beberapa menu dan fitur yang bisa digunakan oleh user. Selection dapat membantu user untuk menseleksi/memilih menu yang ingin digunakan.

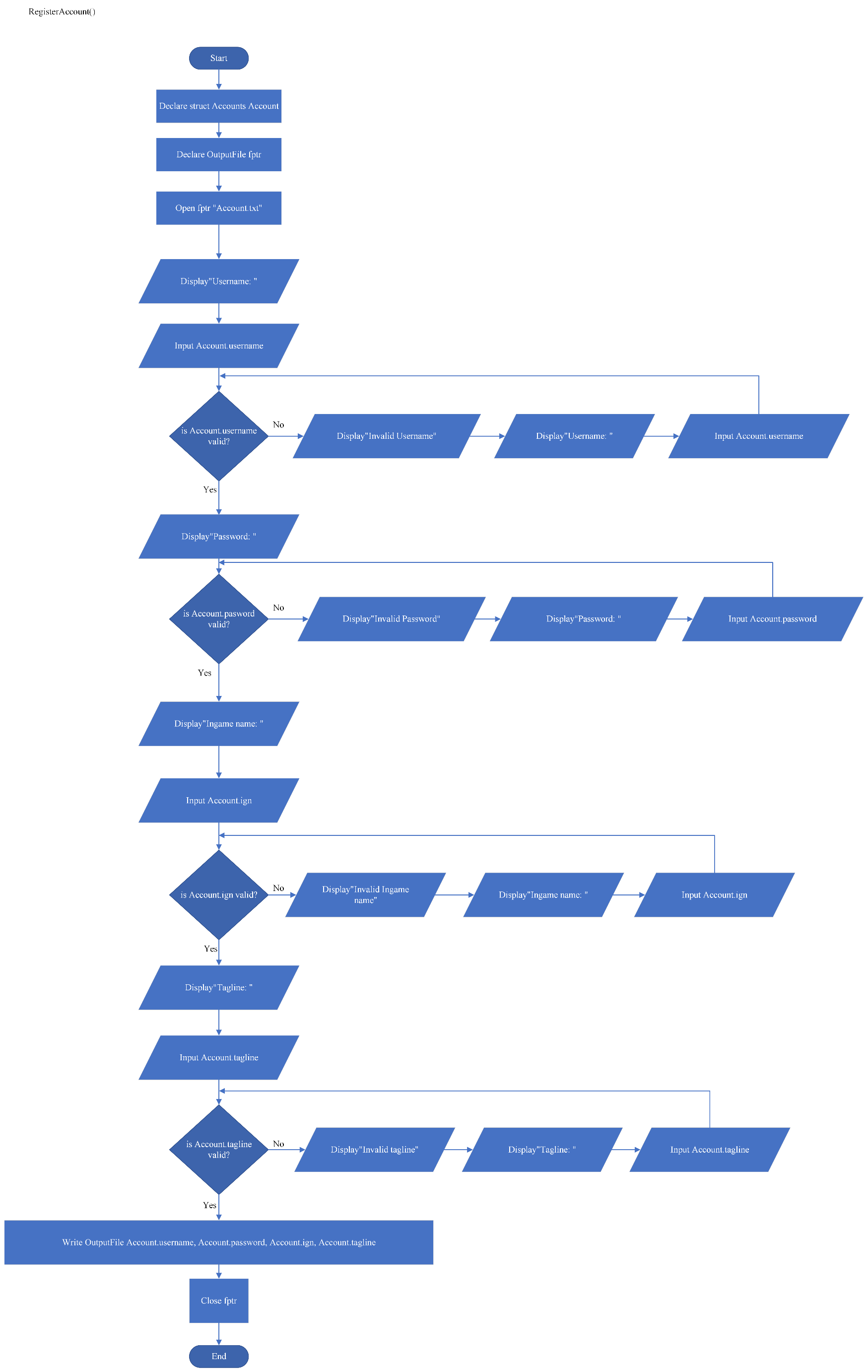
Bab 2

Usulan Desain Program



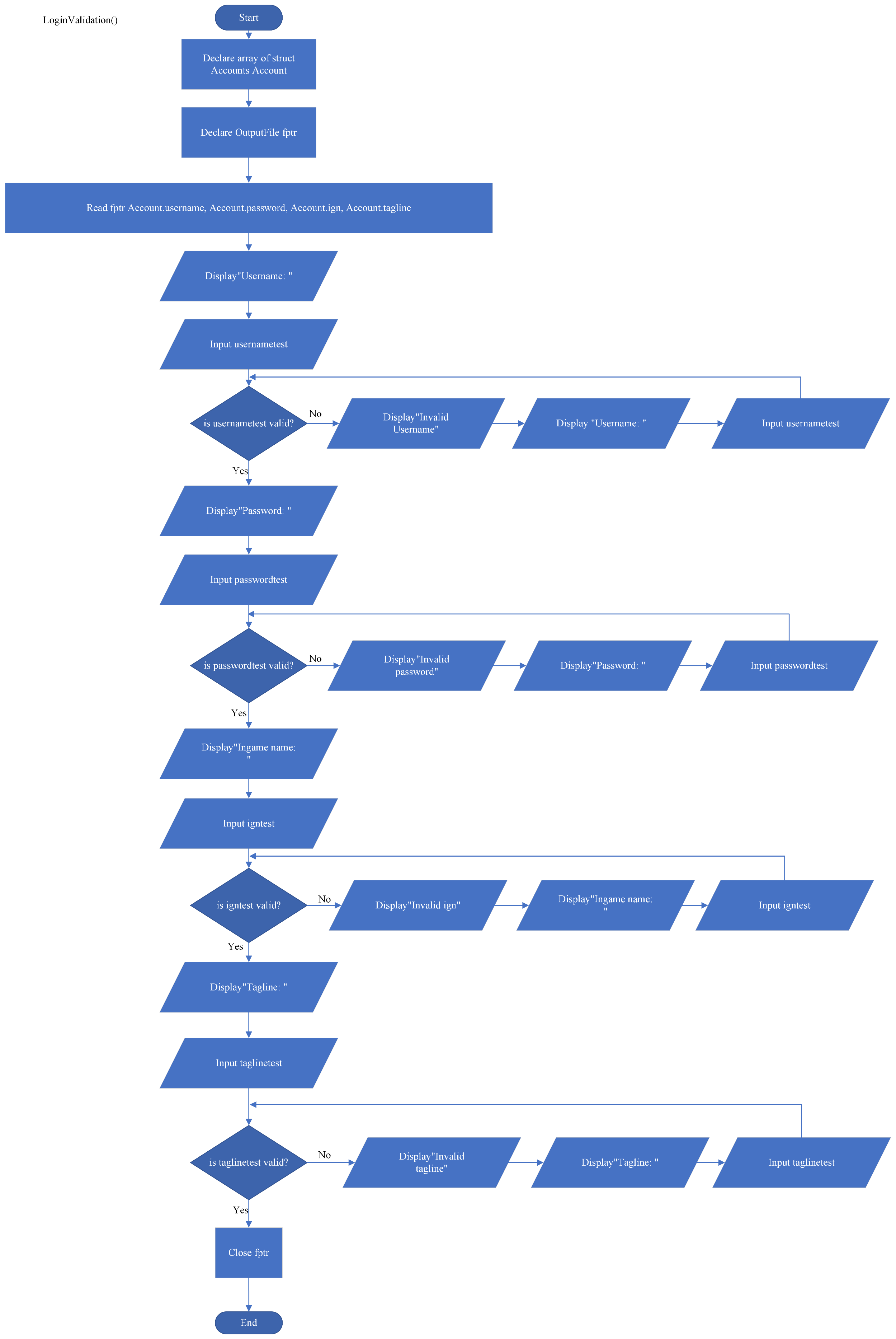
Penjelasan:

User akan melewati tampilan menu pertama yang terdiri opsi login dan opsi register. Bagi user yang pertama kali menggunakan program diharuskan untuk melakukan register akun agar akun bisa tertulis di file txt yang berjudul “Account.txt”. User yang sudah pernah melakukan register akun bisa langsung memilih opsi login



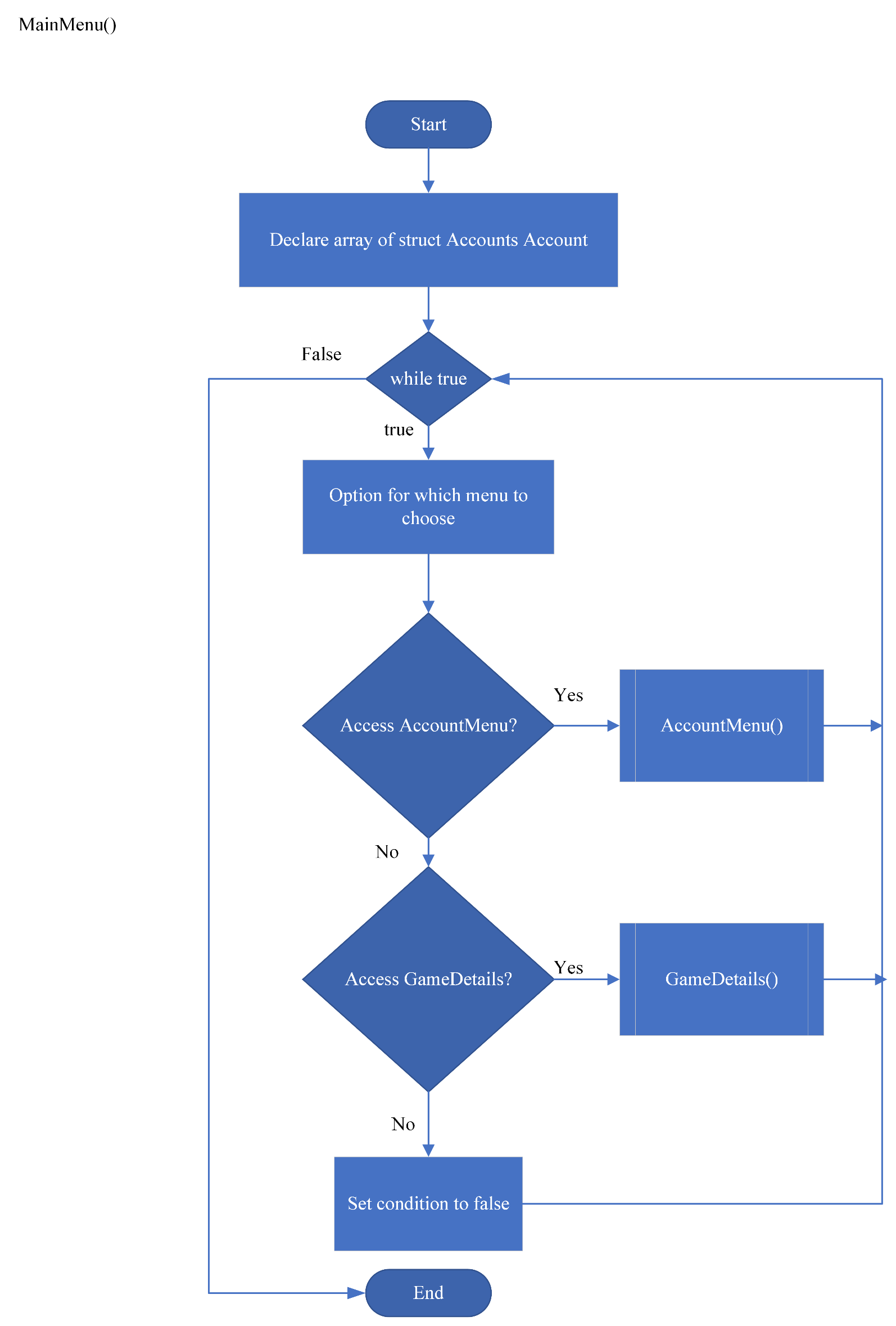
Penjelasan:

User yang belum memiliki akun akan memilih opsi register. Saat opsi register terpilih, function yang bernama “RegisterAccount” akan terpanggil. Di function tersebut, user akan diminta untuk memasukan username, password, ingame name dan tagline. Masing-masing data yang telah diinput user akan divalidasi terlebih dahulu, kemudian data yang sudah tervalidasi akan ditulis kedalam file txt yang berjudul “Account.txt”.



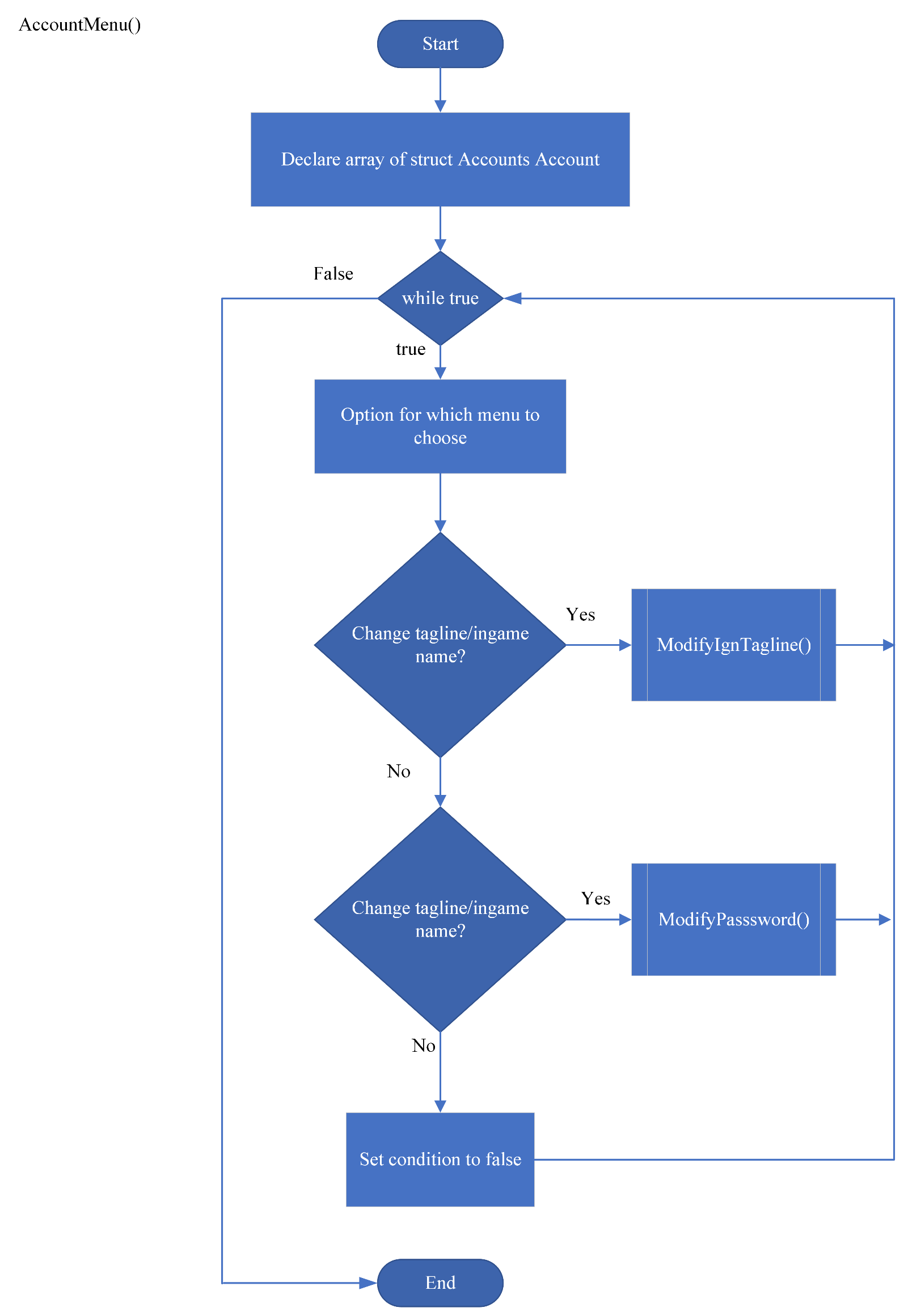
Penjelasan:

User yang sudah memiliki akun akan memilih opsi login. Saat opsi login terpilih, function yang bernama “LoginValidation” akan terpanggil. Di function ini, user akan diminta untuk menginput username, password, ingame name serta tagline yang sudah dibuat/terdaftar. Seluruh input yang telah user masukan akan divalidasi. Setelah semua input sukses tervalidasi, function yang bernama “MainMenu” akan terpanggil. User akan lanjut ke menu utama dari program



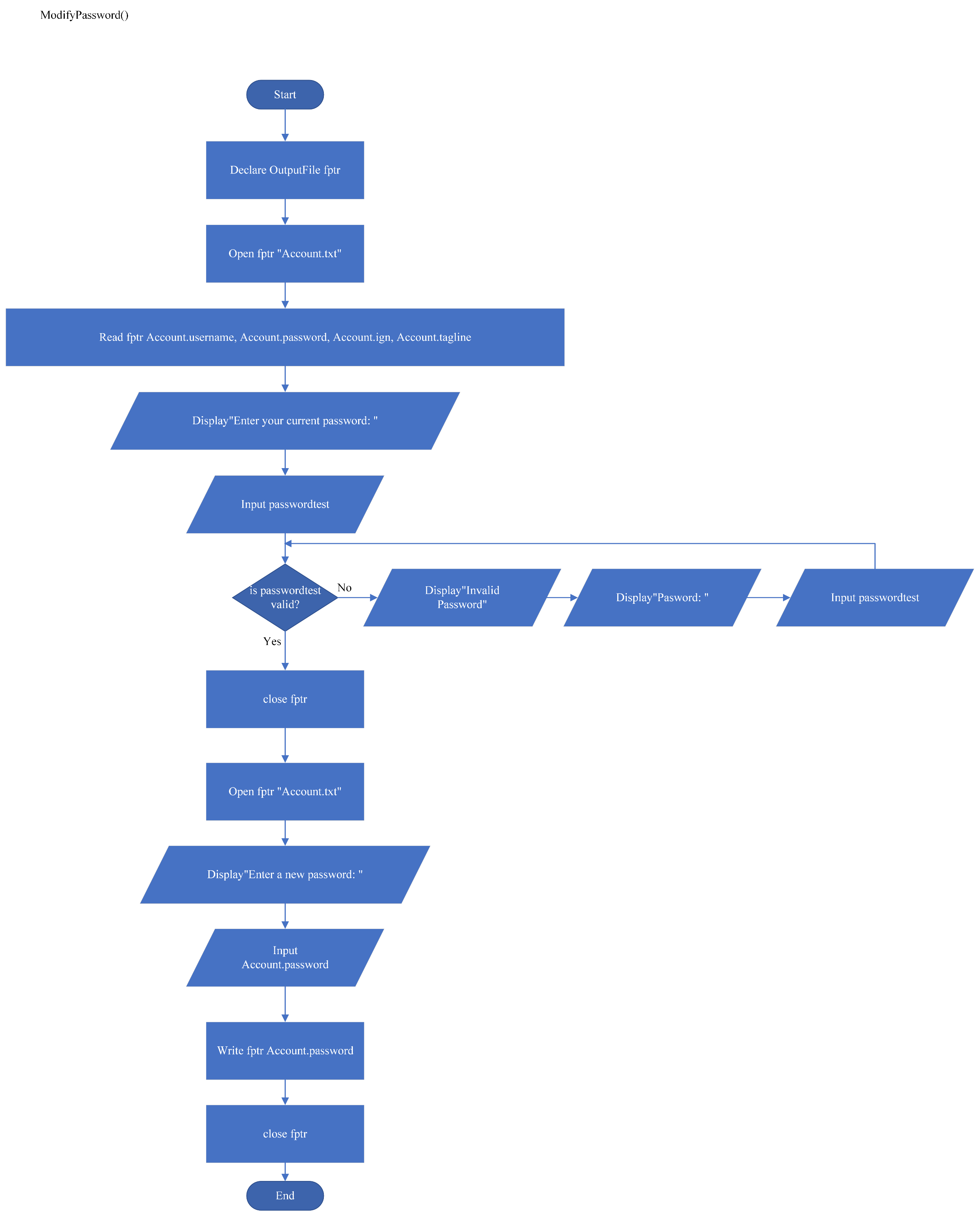
Penjelasan:

Saat function “MainMenu” terpanggil, user akan diarahkan ke menu utama dari program. Di menu utama, user memiliki dua opsi yaitu mengakses menu untuk mengakses akun user atau “AccountMenu” dan menu untuk mengakses game details user atau “GameDetails”



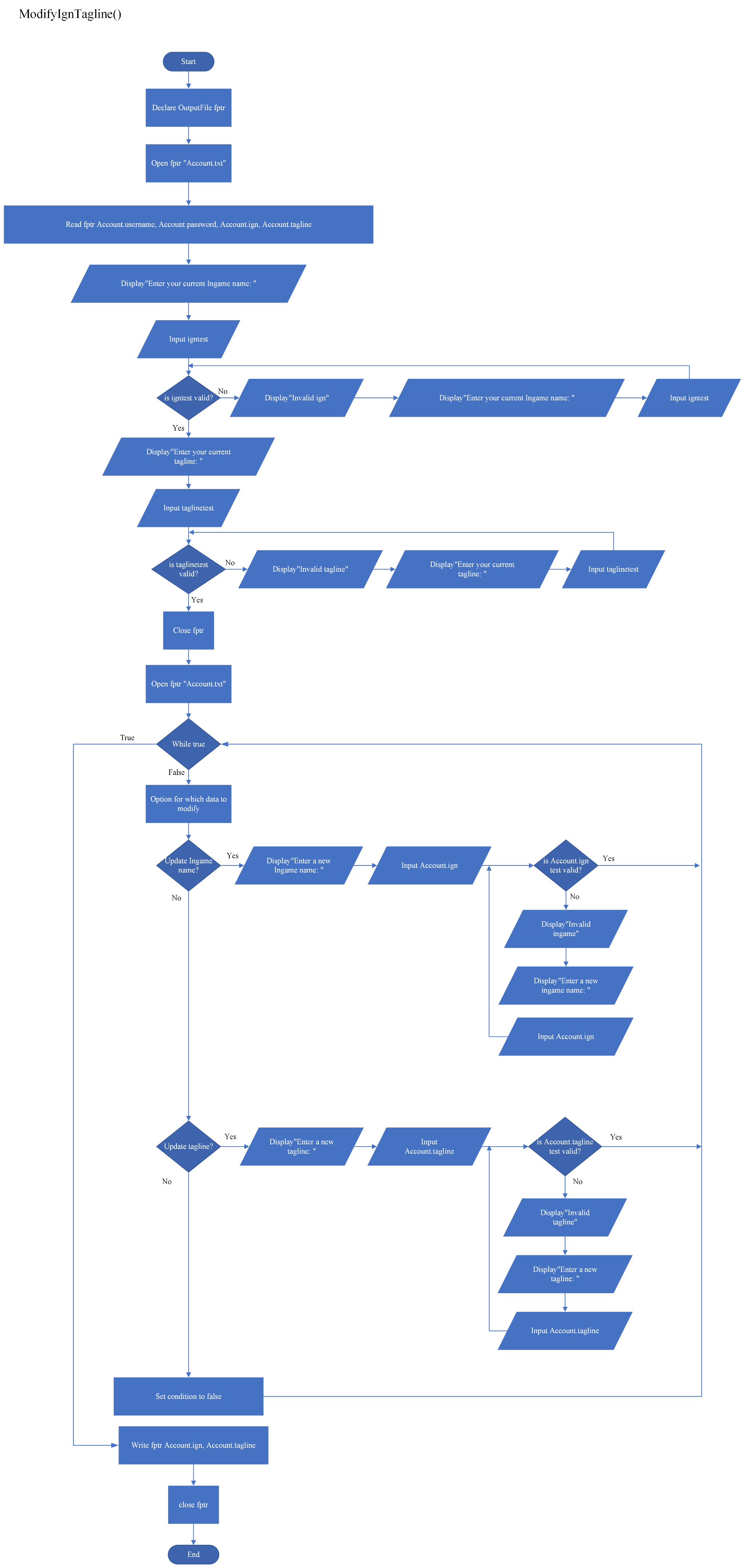
Penjelasan:

Saat user memilih menu untuk mengakses akun user, function “AccountMenu” akan terpanggil. Di function ini terdapat beberapa opsi seperti opsi untuk mengganti password dan opsi untuk mengganti tagline dan ign



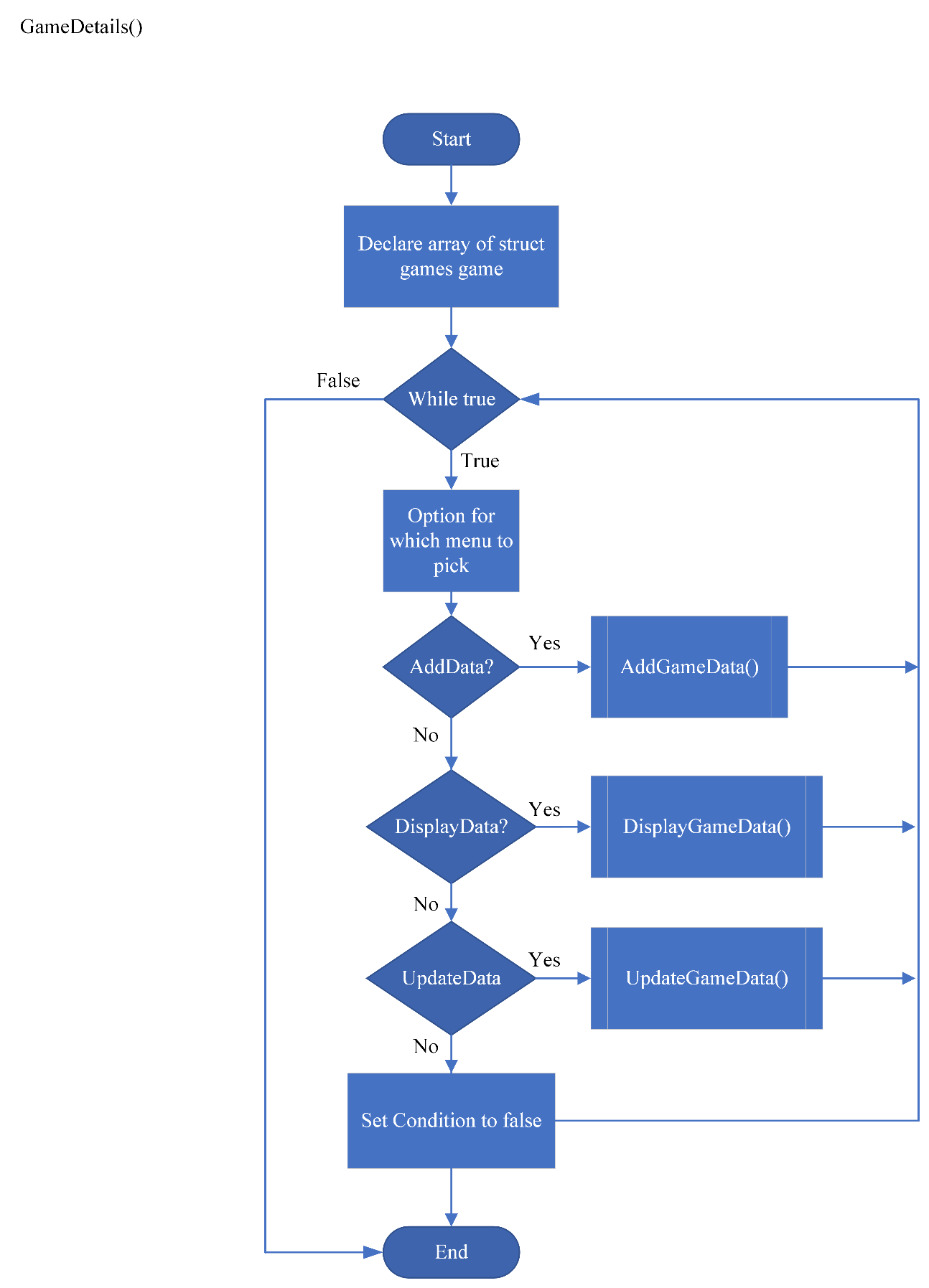
Penjelasan:

Saat user memilih opsi untuk mengganti password akun, function “ModifyPasword()” akan terpanggil. Di function ini, user akan diminta untuk memasukan password sebelumnya kemudian terdapat input validasi terhadap password yang barusan diinput. Setelah sukses tervalidasi, user akan diminta untuk memasukan password baru dan password tersebut akan ditulis ke dalam file txt “Account.txt”



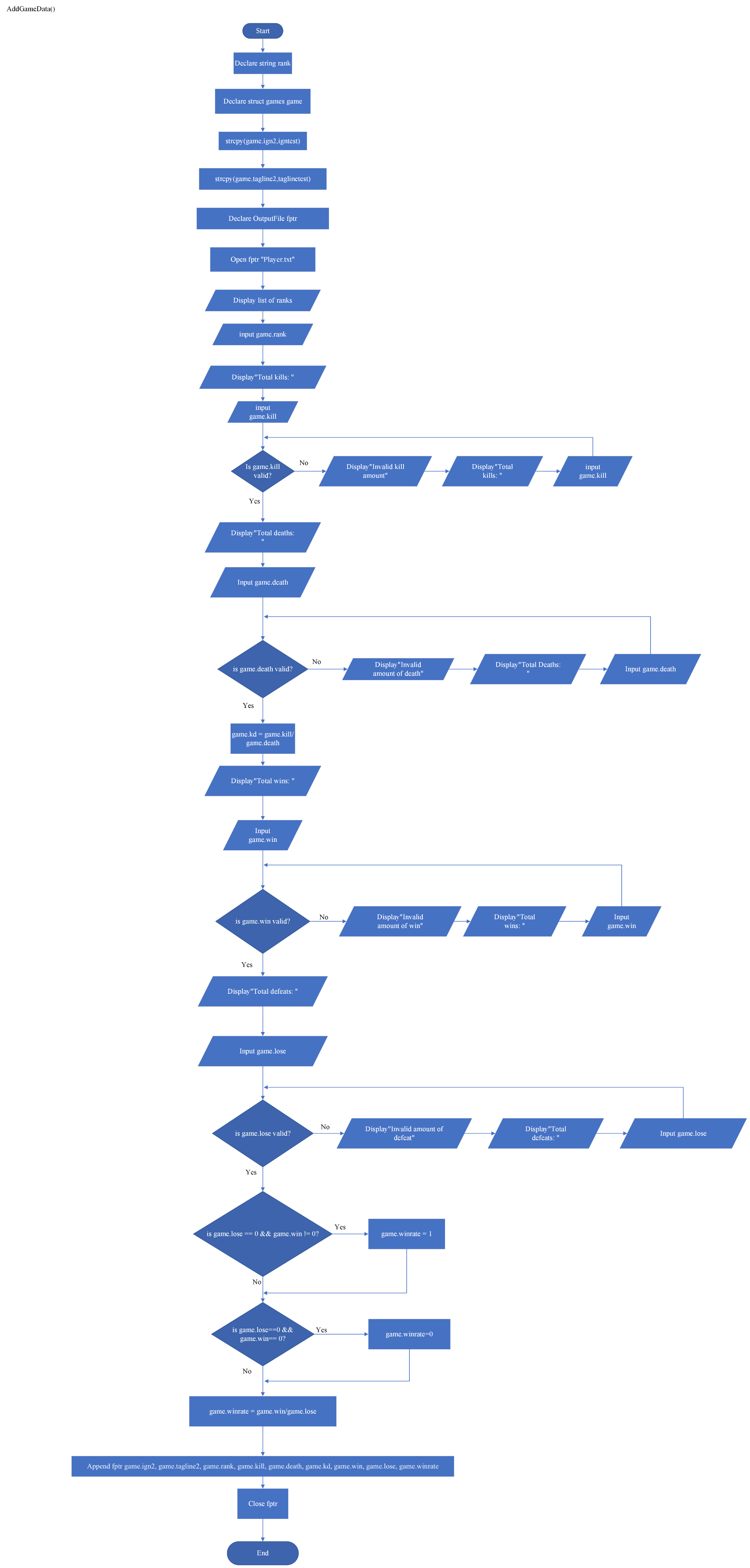
Penjelasan:

Saat user memilih menu untuk mengganti tagline/ingame name, function “ModifyIgnTagline” akan terpanggil. Di function ini, user akan disuruh untuk memasukan ingame name serta tagline sebelumnya, kemudian inputan tersebut akan divalidasi. Setelah selesai divalidasi, user akan ditampilkan dua opsi yaitu opsi untuk mengganti ingame name dan opsi untuk mengganti tagline. Kedua opsi akan meminta user untuk memasukan ingame name/tagline baru kemudian ingame name/tagline yang telah dimasukan akan ditulis ke dalam file txt “Account.txt”



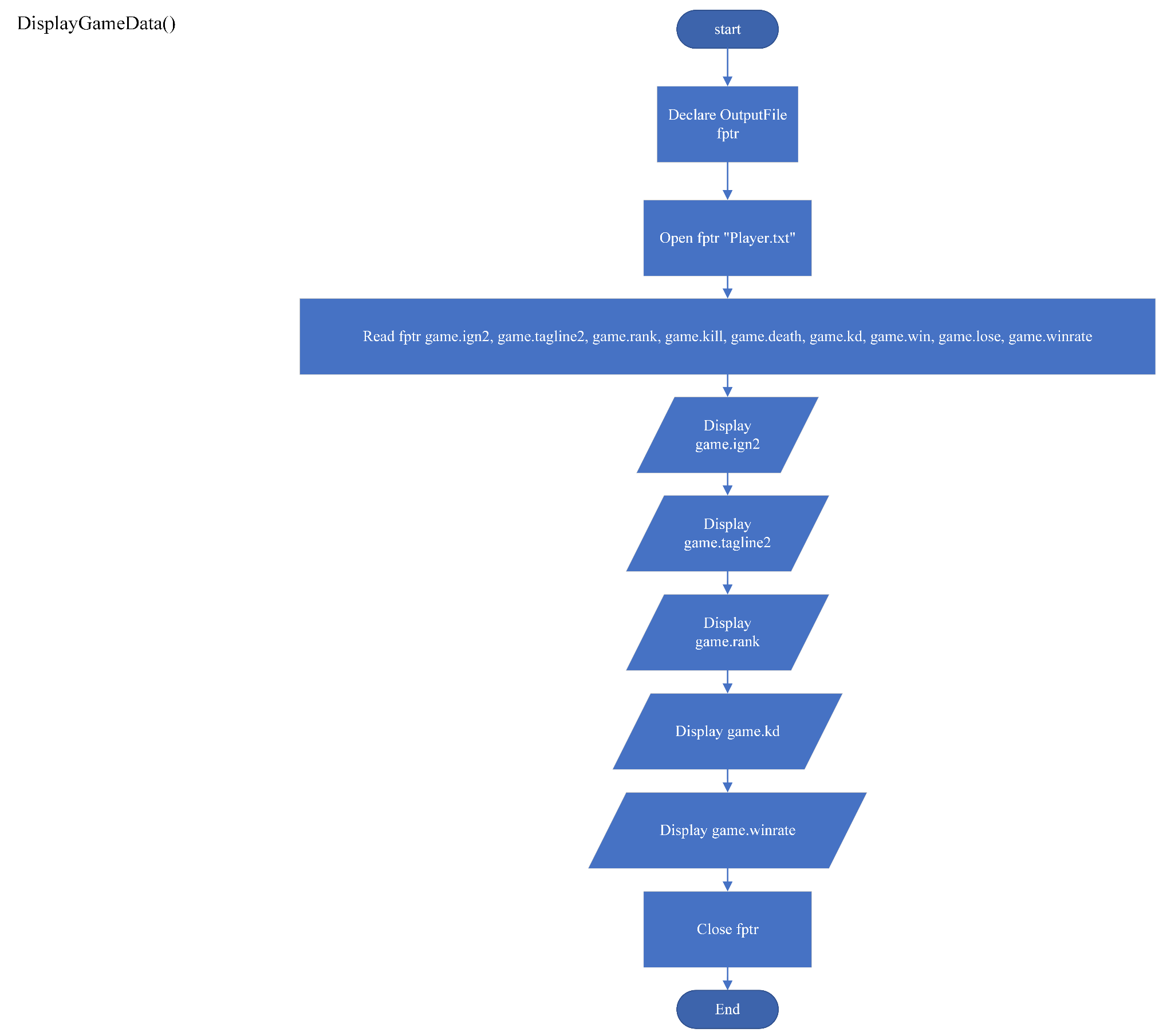
Penjelasan:

Saat user berada di menu utama program dan memilih menu untuk mengakses game details user, function “GameDetails” akan terpanggil. Disini user akan ditampilka 3 opsi yaitu Menambahkan data, menampilkan data dan mengubah data.



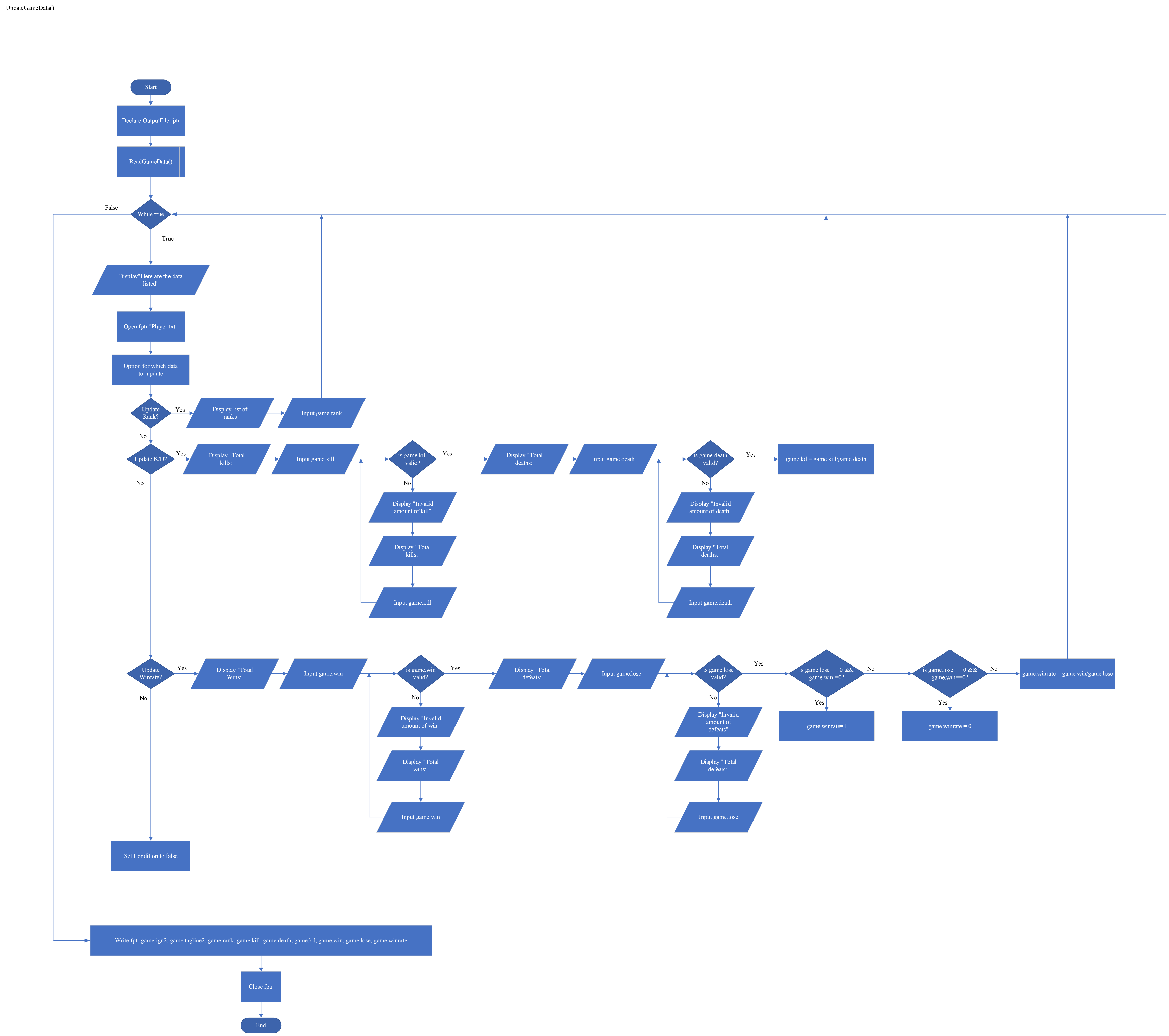
Penjelasan:

Saat user memilih menu “AddData”, function “AddGameData” akan terpanggil. Di function ini, user akan diminta untuk memasukan beberapa data seperti rank, jumlah kill, jumlah death, jumlah win, dan jumlah lose. Semua data kecuali rank akan divalidasi terlebih dahulu. Data seperti jumlah kill dan jumlah death akan diolah menjadi K/D(kill death ratio) sedangkan data seperti jumlah win dan jumlah lose akan diolah menjadi winrate. Setelah itu, data yang sukses tervalidasi dan data yang sudah diolah akan ditulis di file txt “Player.txt”



Penjelasan:

Saat user memilih opsi “DisplayData”, function “DisplayGameData” akan terpanggil. Di function ini, data akan dibaca dari file txt”Player.txt” kemudian akan ditampilkan kepada user



Penjelasan:

Saat user memilih opsi “UpdateData”, function “UpdateGameData” akan terpanggil. Di function ini user akan ditampilkan data yang sebelumnya telah terinput kemudian user akan diarahkan ke menu yang memiliki tiga opsi, yaitu opsi untuk mengubah rank, opsi untuk mengubah kd(kill/death ratio), dan opsi untuk mengubah winrate. Untuk opsi mengubah kd(kill/death ratio), user diharuskan untuk memasukkan jumlah kill dan jumlah death, kemudian data tersebut akan divalidasi setelah itu data yang telah tervalidasi akan diolah menjadi kd(kill/death ratio). Untuk opsi mengubah winrate, user diharuskan untuk memasukkan jumlah win dan jumlah death, kemudian data tersebut akan divalidasi kemudian data yang telah tervalidasi akan diolah menjadi winrate. Data yang telah terubah akan ditulis ke dalam file txt “Player.txt”

Bab 3

Program dan Penjelasannya

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//Declaring struct for Player.txt's data

typedef struct games{

char ign2[17];

char tagline2[17];

char rank[15];

float win;

float lose;

float winrate;

float kill;

float death;

float kd;

}games;

//Declaring struct for Account.txt's data

typedef struct Accounts{

char username[20];

char password[20];

char ign[17];

char tagline[8];

}Accounts;

//Declaring string variable to store user's username when login

char usernametest[20];

//Declaring string variable to store user's ingame name when login

char igntest[17];

//Declaring string variable to store user's tagline when login

char taglinetest[8];

//Declaring array of string to store valorant's rank

char rank1[3][10]={"IRON 1", "IRON 2", "IRON 3"};

char rank2[3][20]={"BRONZE 1", "BRONZE 2", "BRONZE 3"};

char rank3[3][10]={"SILVER 1", "SILVER 2", "SILVER 3"};

char rank4[3][10]={"GOLD 1", "GOLD 2", "GOLD 3"};

char rank5[3][10]={"PLATINUM 1", "PLATINUM 2", "PLATINUM 3"};

char rank6[3][10]={"DIAMOND 1", "DIAMOND 2", "DIAMOND 3"};

char rank7[3][12]={"ASCENDANT 1", "ASCENDANT 2", "ASCENDANT 3"};

char rank8[3][12]={"IMMORTAL 1", "IMMORTAL 2", "IMMORTAL 3"};

char rank9[]="RADIANT";

//Function to add and write player's data into "Player.txt"

void AddGameData(){

//Declaring outputfile

FILE\*fptr;

//Declaring integer's variable for switch case

int menu,ranknum;

//Declaring struct to store data

games game;

//Copying data from user's login information to struct

strcpy(game.ign2,igntest);

strcpy(game.tagline2,taglinetest);

//Opening file to add data

fptr=fopen("Player.txt","a");

//switch case for user to select their rank

printf("<<Rank Selection>>\n");

printf("1. IRON\n");

printf("2. BRONZE\n");

printf("3. SILVER\n");

printf("4. GOLD\n");

printf("5. PLATINUM\n");

printf("6. DIAMOND\n");

printf("7. ASCENDANT\n");

printf("8. IMMORTAL\n");

printf("9. RADIANT\n");

printf("\nRank: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

printf("1. IRON 1\n");

printf("2. IRON 2\n");

printf("3. IRON 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank1[0]);

break;

case 2:

strcpy(game.rank,rank1[1]);

break;

case 3:

strcpy(game.rank,rank1[2]);

break;

}

break;

case 2:

system("cls");

printf("1. BRONZE 1\n");

printf("2. BRONZE 2\n");

printf("3. BRONZE 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank2[0]);

break;

case 2:

strcpy(game.rank,rank2[1]);

break;

case 3:

strcpy(game.rank,rank2[2]);

break;

}

break;

case 3:

system("cls");

printf("1. SILVER 1\n");

printf("2. SILVER 2\n");

printf("3. SILVER 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank3[0]);

break;

case 2:

strcpy(game.rank,rank3[1]);

break;

case 3:

strcpy(game.rank,rank3[2]);

break;

}

break;

case 4:

system("cls");

printf("1. GOLD 1\n");

printf("2. GOLD 2\n");

printf("3. GOLD 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank4[0]);

break;

case 2:

strcpy(game.rank,rank4[1]);

break;

case 3:

strcpy(game.rank,rank4[2]);

break;

}

break;

case 5:

system("cls");

printf("1. PLATINUM 1\n");

printf("2. PLATINUM 2\n");

printf("3. PLATINUM 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank5[0]);

break;

case 2:

strcpy(game.rank,rank5[1]);

break;

case 3:

strcpy(game.rank,rank5[2]);

break;

}

break;

case 6:

system("cls");

printf("1. DIAMOND 1\n");

printf("2. DIAMOND 2\n");

printf("3. DIAMOND 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank6[0]);

break;

case 2:

strcpy(game.rank,rank6[1]);

break;

case 3:

strcpy(game.rank,rank6[2]);

break;

}

break;

case 7:

system("cls");

printf("1. ASCENDANT 1\n");

printf("2. ASCENDANT 2\n");

printf("3. ASCENDANT 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank7[0]);

break;

case 2:

strcpy(game.rank,rank7[1]);

break;

case 3:

strcpy(game.rank,rank7[2]);

break;

}

break;

case 8:

system("cls");

printf("1. IMMORTAL 1\n");

printf("2. IMMORTAL 2\n");

printf("3. IMMORTAL 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game.rank,rank8[0]);

break;

case 2:

strcpy(game.rank,rank8[1]);

break;

case 3:

strcpy(game.rank,rank8[2]);

break;

}

break;

case 9:

strcpy(game.rank,rank9);

break;

}

system("cls");

//User's input for kills

printf("Total kills: ");

// Number of kills validation

while(scanf("%f", &game.kill) != 1){

system("cls");

printf("Number of kills can only be numbers\n");

printf("Total kills: ");

fflush(stdin);

}

while(game.kill < 0){

system("cls");

printf("Kills cannot be negative\n");

printf("Total kills: ");

fflush(stdin);

scanf("%f", &game.kill);

}

system("cls");

//User's input for number of deaths

printf("Total deaths: ");

// Number of deaths validation

while(scanf("%f", &game.death) != 1){

system("cls");

printf("Number of deaths can only be numbers\n");

printf("Total deaths: ");

fflush(stdin);

}

while(game.death < 0 || game.death==0){

system("cls");

printf("Number of deaths can neither be negative nor zero\n");

printf("Total deaths: ");

fflush(stdin);

scanf("%f", &game.death);

}

//calculating kd(kill/death ratio)

game.kd= game.kill/game.death;

system("cls");

//User's input for number of wins

printf("Total wins: ");

// Number of win validation

while(scanf("%f", &game.win) != 1){

system("cls");

printf("Number of wins can only be numbers\n");

printf("Total wins: ");

fflush(stdin);

}

while(game.win < 0){

system("cls");

printf("Number of wins cannot be negative\n");

printf("Total wins: ");

fflush(stdin);

scanf("%f", &game.win);

}

system("cls");

//User's input for number of loses/defeats

printf("Total defeats: ");

// Number of loses validation

while(scanf("%f", &game.lose) != 1){

printf("Number of loses can only be numbers\n");

printf("Total defeats: ");

fflush(stdin);

}

while(game.lose < 0){

system("cls");

printf("Number of loses cannnot be negative\n");

printf("Total defeats: ");

fflush(stdin);

scanf("%f", &game.lose);

}

//Calculating game's winrate

if((game.lose)==0 && (game.win)!=0){

game.winrate=1;

}

else if((game.win && game.lose)==0){

game.winrate= 0;

}

else{

game.winrate = (game.win)/(game.win + game.lose);

}

//writing all the data into "Player.txt" file

fprintf(fptr,"\n%s;%s;%s;%.0f;%.0f;%.2f;%.0f;%.0f;%.2f\n",game.ign2,game.tagline2,game.rank,game.kill,game.death,game.kd,game.win,game.lose,game.winrate);

//Closing file

fclose(fptr);

system("cls");

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

}

//Function to read data from "Player.txt" and displaying it

void DisplayGameData(games game[100],int \*x){//passing struct and pointer variable as parameter to pass its data to this function

//declaring outputfile

FILE\*fptr;

//transfering pointer variable's data into new variable

int limit=\*x;

//Opening file to read data

fptr=fopen("Player.txt","r");

//Reading data from file and displaying it

printf("Ingame name\t\t\t tagline\t\t Rank\t\t K/D\t\t Winrate\n");

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

while(fscanf(fptr,"\n%[^';'];%[^';'];%[^';'];%f;%f;%f;%f;%f;%f\n",&game[limit].ign2,&game[limit].tagline2,&game[limit].rank,

&game[limit].kill,&game[limit].death,&game[limit].kd,&game[limit].win,&game[limit].lose,&game[limit].winrate)!=EOF){

printf("%-20s\t\t %-20s\t %-10s\t %-10.2f\t %-10.2f\n",game[limit].ign2,game[limit].tagline2,game[limit].rank,game[limit].kd,game[limit].winrate);

limit++;

}

printf("\n");

//transfering variable's data into pointer variable

\*x=limit;

//closing file

fclose(fptr);

}

//Function to change player's data and writing into "Player.txt" file

void UpdateGameData(games game[100],int \*x){//passing struct and pointer variable as parameter to pass its data to this function

//declaring outputfile

FILE\*fptr;

//declaring variable to help selecting in switch case as well as to store array's index number

int indign,option;

//Calling displaydata's function to read and display data

DisplayGameData(game,x);

//transfering pointer variable's data into new variable

int limit=\*x;

printf("\nHere are the data listed\n");

system("pause");

//storing index's number based on tagline and ingame name that were inputted earlier when login

for(int i=0;i<limit;i++){

if((strcmp(game[i].ign2,igntest)==0)&&(strcmp(game[i].tagline2,taglinetest)==0)){

indign=i;

break;

}

}

//declaring variable to help selecting in switch case as well as to store array's index number

int menu,menu2,ranknum;

//Opening file to write data

fptr=fopen("Player.txt","w");

//option for the user to choose which data to update

do{

system("cls");

printf("<<Update Data>>\n");

printf("1. Rank\n");

printf("2. K/D\n");

printf("3. Winrate\n");

printf("4. Exit\n");

printf("\nOption: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

printf("<<Rank Selection>>\n");

printf("1. IRON\n");

printf("2. BRONZE\n");

printf("3. SILVER\n");

printf("4. GOLD\n");

printf("5. PLATINUM\n");

printf("6. DIAMOND\n");

printf("7. ASCENDANT\n");

printf("8. IMMORTAL\n");

printf("9. RADIANT\n");

printf("\nRank: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

printf("1. IRON 1\n");

printf("2. IRON 2\n");

printf("3. IRON 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank1[0]);

break;

case 2:

strcpy(game[indign].rank,rank1[1]);

break;

case 3:

strcpy(game[indign].rank,rank1[2]);

break;

}

break;

case 2:

system("cls");

printf("1. BRONZE 1\n");

printf("2. BRONZE 2\n");

printf("3. BRONZE 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank2[0]);

break;

case 2:

strcpy(game[indign].rank,rank2[1]);

break;

case 3:

strcpy(game[indign].rank,rank2[2]);

break;

}

break;

case 3:

system("cls");

printf("1. SILVER 1\n");

printf("2. SILVER 2\n");

printf("3. SILVER 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank3[0]);

break;

case 2:

strcpy(game[indign].rank,rank3[1]);

break;

case 3:

strcpy(game[indign].rank,rank3[2]);

break;

}

break;

case 4:

system("cls");

printf("1. GOLD 1\n");

printf("2. GOLD 2\n");

printf("3. GOLD 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank4[0]);

break;

case 2:

strcpy(game[indign].rank,rank4[1]);

break;

case 3:

strcpy(game[indign].rank,rank4[2]);

break;

}

break;

case 5:

system("cls");

printf("1. PLATINUM 1\n");

printf("2. PLATINUM 2\n");

printf("3. PLATINUM 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank5[0]);

break;

case 2:

strcpy(game[indign].rank,rank5[1]);

break;

case 3:

strcpy(game[indign].rank,rank5[2]);

break;

}

break;

case 6:

system("cls");

printf("1. DIAMOND 1\n");

printf("2. DIAMOND 2\n");

printf("3. DIAMOND 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank6[0]);

break;

case 2:

strcpy(game[indign].rank,rank6[1]);

break;

case 3:

strcpy(game[indign].rank,rank6[2]);

break;

}

break;

case 7:

system("cls");

printf("1. ASCENDANT 1\n");

printf("2. ASCENDANT 2\n");

printf("3. ASCENDANT 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank7[0]);

break;

case 2:

strcpy(game[indign].rank,rank7[1]);

break;

case 3:

strcpy(game[indign].rank,rank7[2]);

break;

}

break;

case 8:

system("cls");

printf("1. IMMORTAL 1\n");

printf("2. IMMORTAL 2\n");

printf("3. IMMORTAL 3\n");

printf("\nOption: ");

scanf("%d",&ranknum);

switch(ranknum){

case 1:

strcpy(game[indign].rank,rank8[0]);

break;

case 2:

strcpy(game[indign].rank,rank8[1]);

break;

case 3:

strcpy(game[indign].rank,rank8[2]);

break;

}

break;

case 9:

strcpy(game[indign].rank,rank9);

break;

}

break;

case 2:

system("cls");

//User's input for amount of kills

printf("Total Kills: ");

fflush(stdin);

// Number of kills validation

while(scanf("%f", &game[indign].kill) != 1){

system("cls");

printf("Number of kills can only be numbers\n");

printf("Total kills: ");

fflush(stdin);

}

while(game[indign].kill < 0){

system("cls");

printf("Kills cannot be negative\n");

printf("Total kills: ");

fflush(stdin);

scanf("%f", &game[indign].kill);

getchar();

}

system("cls");

//User's input for amount of deaths

printf("Total Deaths: ");

fflush(stdin);

// Number of deaths validation

while(scanf("%f", &game[indign].death) != 1){

system("cls");

printf("Number of deaths can only be numbers\n");

printf("Total deaths: ");

fflush(stdin);

}

while(game[indign].death < 0|| game[indign].death==0){

system("cls");

printf("Number of deaths can neither be negative nor zero\n");

printf("Total deaths: ");

fflush(stdin);

scanf("%f", &game[indign].death);

getchar();

}

//Calculating kd(kill/death ratio)

game[indign].kd= game[indign].kill/game[indign].death;

system("pause");

break;

case 3:

system("cls");

//User's input for amount of win

printf("Total win: ");

fflush(stdin);

// Number of wins validation

while(scanf("%f", &game[indign].win) != 1){

system("cls");

printf("Number of wins can only be numbers\n");

printf("Total wins: ");

fflush(stdin);

}

while(game[indign].win < 0){

system("cls");

printf("Number of wins cannot be negative\n");

printf("Total wins: ");

fflush(stdin);

scanf("%f", &game[indign].win);

getchar();

}

system("cls");

//User's input for amount of loses/defeats

printf("Total defeats: ");

fflush(stdin);

// Number of loses validation

while(scanf("%f", &game[indign].lose) != 1){

system("cls");

printf("Number of defeats can only be numbers\n");

printf("Total defeats: ");

fflush(stdin);

}

while(game[indign].lose < 0){

system("cls");

printf("Number of defeats cannot be negative\n");

printf("Total defeats: ");

fflush(stdin);

scanf("%f", &game[indign].lose);

getchar();

}

//Calculating game's winrate

if((game[indign].lose)==0 && (game[indign].win)!=0){

game[indign].winrate=1;

}

else if((game[indign].win && game[indign].lose)==0){

game[indign].winrate= 0;

}

else{

game[indign].winrate = game[indign].win/(game[indign].win + game[indign].lose);

}

system("pause");

break;

case 4:

//Writing changed data into "Player.txt"

for (int i = 0; i < limit; i++)

{

if(i == limit-1){

fprintf(fptr,"%s;%s;%s;%.0f;%.0f;%.2f;%.0f;%.0f;%.2f",game[i].ign2,game[i].tagline2,game[i].rank,game[i].kill,game[i].death,game[i].kd,game[i].win,game[i].lose,game[i].winrate);

}else{

fprintf(fptr,"%s;%s;%s;%.0f;%.0f;%.2f;%.0f;%.0f;%.2f\n",game[i].ign2,game[i].tagline2,game[i].rank,game[i].kill,game[i].death,game[i].kd,game[i].win,game[i].lose,game[i].winrate);

}

}

}

}while(menu!=4);

//closing file

fclose(fptr);

system("cls");

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

}

//Gamedetails' menu for the user to choose which feature that the user wants to use

void GameDetails(){

//declaring integers variable to help in selecting options in switch case

int menu;

//declaring array of struct

games\* game= malloc(100 \* sizeof(\*game));

//Switch cases to cater options for the user to pick

do{

int limit=0;

printf("<<Game Details>>\n");

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

printf("2. Read Data\n");

printf("3. Update Data\n");

printf("4. Exit\n");

printf("\nMenu: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

AddGameData();

system("pause");

system("cls");

break;

case 2:

system("cls");

DisplayGameData(game,&limit);

system("pause");

system("cls");

break;

case 3:

system("cls");

UpdateGameData(game,&limit);

system("pause");

system("cls");

break;

}

}while(menu!=4);

//freeing the memory of array of struct

free(game);

}

//Function to add user account's data as well as a function to write in "Account.txt" file

void RegisterAccount(){

//declaring struct to store user's information

Accounts Account;

//declaring outputfile

FILE\*fptr;

//Opening file to add data into "Account.txt"file

fptr=fopen("Account.txt","a");

//Asking for user's username input

printf("Username: ");

fflush(stdin);

scanf("%s",Account.username);

getchar();

//Validating user's username

while(strlen(Account.username)>20){

system("cls");

printf("Username cannot be longer than 16 characters\n");

printf("\nUsername: ");

fflush(stdin);

scanf("%s",Account.username);

getchar();

}

//Asking for user's password input

printf("Password: ");

fflush(stdin);

scanf("%s",Account.password);

getchar();

//validating user's password

while(strlen(Account.password)>20){

system("cls");

printf("Username cannot be longer than 16 characters\n");

printf("\nUsername: ");

fflush(stdin);

scanf("%s",Account.password);

getchar();

}

//Asking for user's ingame name input

printf("Ingame name: ");

fflush(stdin);

scanf("%s",Account.ign);

getchar();

//valdating user's ingame name

while(strlen(Account.ign)>17){

system("cls");

printf("Username cannot be longer than 16 characters\n");

printf("\nIngame name: ");

fflush(stdin);

scanf("%s",Account.ign);

getchar();

}

//Asking for user's tagline input

printf("Tagline: ");

fflush(stdin);

scanf("%s",Account.tagline);

getchar();

//validating user's tagline

while(strlen(Account.tagline)>5){

system("cls");

printf("Username cannot be longer than 16 characters\n");

printf("\nTagline: ");

fflush(stdin);

scanf("%s",Account.tagline);

getchar();

}

system("cls");

//writing data inputted into "Account.txt" file

fprintf(fptr,"\n%s;%s;%s;%s\n",Account.username,Account.password,Account.ign,Account.tagline);

printf("--------------------Account Added---------------------------------\n");

//closing file

fclose(fptr);

}

//function to validate user's login

void LoginValidation(){

//declaring array of struct to store data read from Account.txt file

Accounts\* Account= malloc(100 \* sizeof(\*Account));

//declaring outputfile

FILE\*fptr;

//Declaring variable that helps in validating input

int found,i=0;

//Declaring variable to store user's password login input

char passwordtest[20];

//Opening file to read data

fptr=fopen("Account.txt","r");

//Validating incase file doesnot exist

if (fptr == NULL)

{

printf("Account doesnt exist\n");

return;

}

//Prompt for user to input username

printf("Username: ");

fflush(stdin);

scanf("%s",usernametest);

getchar();

//Username Validation

for(i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].username,usernametest)==0){

found=0;

break;

}

found=1;

}

rewind(fptr);

for(i=0;found == 1;){

printf("Invalid Username\n");

printf("Username: ");

fflush(stdin);

scanf("%s", usernametest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].username,usernametest)==0){

found=0;

break;

}

}

rewind(fptr);

}

//Prompt for user to input password

system("cls");

printf("Password: ");

fflush(stdin);

scanf("%s",passwordtest);

getchar();

//Password Validation

for(i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].password,passwordtest)==0){

found=0;

break;

}

found=1;

}

rewind(fptr);

for(i=0;found == 1;){

printf("Invalid Password\n");

printf("Password: ");

fflush(stdin);

scanf("%s", passwordtest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].password,passwordtest)==0){

found=0;

break;

}

}

rewind(fptr);

}

//Prompt for user to input ingame name

system("cls");

printf("ingame name: ");

fflush(stdin);

scanf("%s",igntest);

getchar();

//ingame name Validation

for(i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].ign,igntest)==0){

found=0;

break;

}

found=1;

}

rewind(fptr);

for(i=0;found == 1;){

printf("Invalid ingame name\n");

printf("Ingame name: ");

fflush(stdin);

scanf("%s", igntest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].ign,igntest)==0){

found=0;

break;

}

}

rewind(fptr);

}

//Prompt for user to input tagline

system("cls");

printf("Tagline: ");

fflush(stdin);

scanf("%s",taglinetest);

getchar();

//tagline Validation

for(i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].tagline,taglinetest)==0){

found=0;

break;

}

found=1;

}

rewind(fptr);

for(i=0;found == 1;){

printf("Invalid tagline\n");

printf("Tagline: ");

fflush(stdin);

scanf("%s", taglinetest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].tagline,taglinetest)==0){

found=0;

break;

}

}

rewind(fptr);

}

//freeing the memory used in array of struct

free(Account);

system("cls");

printf("----------------Login Success----------------\n");

//closing file

fclose(fptr);

}

//Function to change user's password

void ModifyPassword(Accounts Account[100]){//Passing array of struct as a parameter to this function

//declaring outputfile

FILE\*fptr;

//Declaring integers variable to help validate as well as to store's array's index number

int indubah;

int found=0,limit;

//Declaring string variable to store user's password

char passwordtest[20];

//Opening file to read data

fptr=fopen("Account.txt","r");

//Count the amount of line

for(int i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

limit++;

}

rewind(fptr);

//Input Current Password

printf("Enter your current password: ");

fflush(stdin);

scanf("%s",passwordtest);

getchar();

//Password Validation

for(int i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].password,passwordtest)==0){

indubah=i;

found=0;

break;

}

found=1;

}

rewind(fptr);

for(int i=0;found == 1;){

printf("Invalid Password\n");

printf("Enter your current password: ");

fflush(stdin);

scanf("%s", passwordtest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].password,passwordtest)==0){\

indubah=i;

found=0;

break;

}

}

rewind(fptr);

}

//closing file

fclose(fptr);

//Input New Password

fptr=fopen("Account.txt","w");

system("cls");

printf("Enter a new password: ");

fflush(stdin);

scanf("%s",Account[indubah].password);

getchar();

//Rewriting Old Password in "Account.txt" file

for (int i = 0; i < limit; i++){

if(i == limit-1){

fprintf(fptr, "%s;%s;%s;%s", Account[i].username, Account[i].password, Account[i].ign, Account[i].tagline);

}else{

fprintf(fptr, "%s;%s;%s;%s\n", Account[i].username, Account[i].password, Account[i].ign, Account[i].tagline);

}

}

system("cls");

printf("--------------------Password changed------------------------\n");

//closing file

fclose(fptr);

}

//Function to change user's ingamename and tagline

void ModifyIgnTagline(Accounts Account[100]){//Passing array of struct as a parameter to this function

//declaring output file

FILE\*fptr;

//declaring integer variable to help validate data as well as to store array's index number

int indubah;

int found=0,limit;

//Opening file to read data from "Account.txt"

fptr=fopen("Account.txt","r");

//Count the amount of line

for(int i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

limit++;

}

rewind(fptr);

//Input Current Ingame name

printf("Enter your current ingame name: ");

fflush(stdin);

scanf("%s",igntest);

getchar();

//Ingame name Validation

for(int i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].ign,igntest)==0){

indubah=i;

found=0;

break;

}

found=1;

}

rewind(fptr);

for(int i=0;found == 1;){

printf("Invalid ingame name\n");

printf("Enter your current ingame name: ");

fflush(stdin);

scanf("%s", igntest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].ign,igntest)==0){\

indubah=i;

found=0;

break;

}

}

rewind(fptr);

}

system("cls");

//Input Current tagline

printf("Enter your current tagline: ");

fflush(stdin);

scanf("%s",taglinetest);

getchar();

//Tagline validation

for(int i=0;fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF;){

if(strcmp(Account[i].tagline,taglinetest)==0){

indubah=i;

found=0;

break;

}

found=1;

}

rewind(fptr);

for(int i=0;found == 1;){

printf("Invalid tagline\n");

printf("Enter your current tagline: ");

fflush(stdin);

scanf("%s", taglinetest);

while(fscanf(fptr,"\n%[^\';'];%[^\';'];%[^\';'];%s\n", &Account[i].username,&Account[i].password,&Account[i].ign,&Account[i].tagline)!=EOF){

if(strcmp(Account[i].tagline,taglinetest)==0){

indubah=i;

found=0;

break;

}

}

rewind(fptr);

}

//closing file

fclose(fptr);

//opening file to write data into "Account.txt"

fptr=fopen("Account.txt","w");

system("cls");

//declaring integer variable to help user selection options in switch case

int menu;

//switch case for user to pick which data to change

do{

printf("Which one do you wish to change?\n");

printf("1. Ingame name\n");

printf("2. Tagline\n");

printf("3. Exit\n");

printf("\nOption: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

//Input new ingame name

printf("Enter a new ingame name: ");

fflush(stdin);

scanf("%s",Account[indubah].ign);

getchar();

//validating ingame name

while(strlen(Account[indubah].ign)>17){

system("cls");

printf("Username cannot be longer than 16 characters\n");

printf("\nIngame name: ");

fflush(stdin);

scanf("%s",Account[indubah].ign);

getchar();

}

system("pause");

system("cls");

break;

case 2:

system("cls");

//Input new tagline

printf("Enter a new tagline: ");

fflush(stdin);

scanf("%s",Account[indubah].tagline);

getchar();

//validating tagline

while(strlen(Account[indubah].tagline)>5){

system("cls");

printf("Username cannot be longer than 16 characters\n");

printf("\nTagline: ");

fflush(stdin);

scanf("%s",Account[indubah].tagline);

getchar();

}

system("pause");

system("cls");

break;

case 3:

//Rewriting tagline/ingame name into the file("Account.txt")

for (int i = 0; i < limit; i++){

if(i == limit-1){

fprintf(fptr, "%s;%s;%s;%s", Account[i].username, Account[i].password, Account[i].ign, Account[i].tagline);

}

else{

fprintf(fptr, "%s;%s;%s;%s\n", Account[i].username, Account[i].password, Account[i].ign, Account[i].tagline);

}

}

}

}while(menu!=3);

system("cls");

printf("--------------------Details changed------------------------\n");

//closing file

fclose(fptr);

}

//Function for Account's menu

void AccountMenu(){

//declaring integer variable to help user pick which menu to use in switch case

int menu,limit;

//declaring array of struct

Accounts\* Account= malloc(100 \* sizeof(\*Account));

//switch case for user to pick which menu to use

do{

int limit=0;

printf("<<Account Menu>>\n");

printf("1. Change tagline/ingame name\n");

printf("2. Change Password\n");

printf("3. Exit\n");

printf("\nMenu: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

ModifyIgnTagline(Account);

system("pause");

system("cls");

break;

case 2:

system("cls");

ModifyPassword(Account);

system("pause");

system("cls");

break;

}

}while(menu!=3);

//freeing array of struct's memory

free(Account);

}

//Function for main menu that can direct to other menu

void MainMenu(){

//declaring integer variable to help user pick which menu to use in switch case

int menu,limit;

//declaring array of struct

Accounts\* Account= malloc(100 \* sizeof(\*Account));

//switch case for user to pick which menu to use

do{

int limit=0;

printf("Hi %s\n",usernametest);

printf("\n<<Main Menu>>\n");

printf("1. Account\n");

printf("2. Game Details\n");

printf("3. Exit\n");

printf("\nMenu: ");

scanf("%d",&menu);

switch(menu){

case 1:

system("cls");

AccountMenu();

system("pause");

system("cls");

break;

case 2:

system("cls");

GameDetails(Account);

system("pause");

system("cls");

break;

}

}while(menu!=3);

//freeing array of struct's memory

free(Account);

}

//Main function that either force user to login or register/create an account

int main(){

//declaring integer variable to help user pick which menu to use in switch case

int menu;

//switch case for user to pick which menu to use

do{

system("cls");

printf("<<Valorant Tracker>>\n");

printf("1.Login\n");

printf("2.Register\n");

printf("3.Exit\n");

printf("\nMenu: ");

scanf("%d",&menu);

switch (menu){

case 1:

system("cls");

LoginValidation();

system("pause");

system("cls");

MainMenu();

break;

case 2:

system("cls");

RegisterAccount();

system("pause");

system("cls");

break;

}

}while(menu!=3);

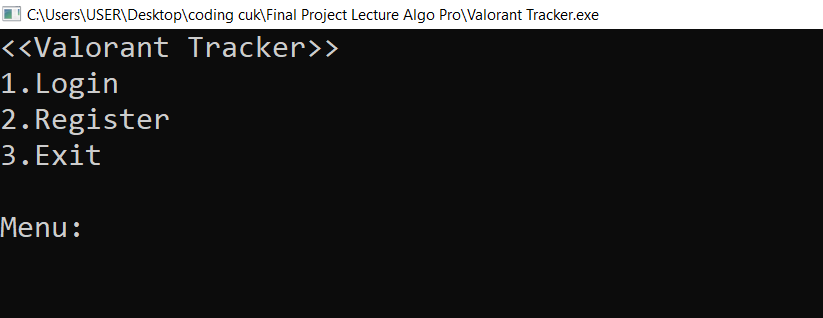
system("cls");

printf("----------------------------------------Program Closed-----------------------------\n");

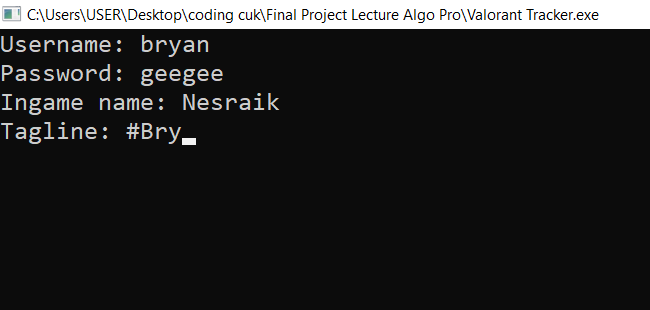
}

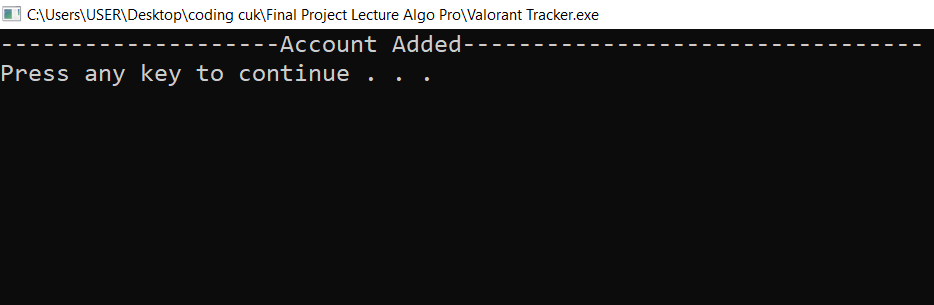
Bab 4

Tampilan Hasil

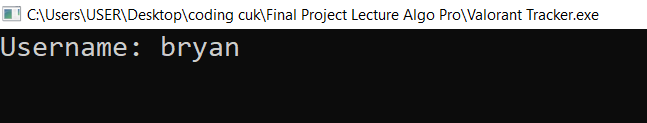
Menu Pertama

Saat user memilih register, user diminta untuk memasukkan username, password, ingame name dan tagline

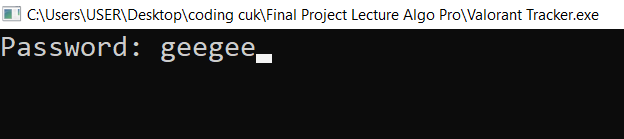


Saat dienter akan muncul tampilan ini

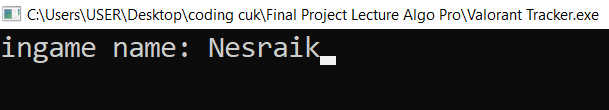
Saat user memilih login, username akan diminta



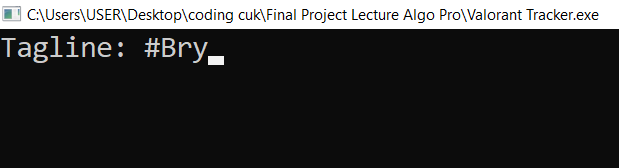
Password akan diminta

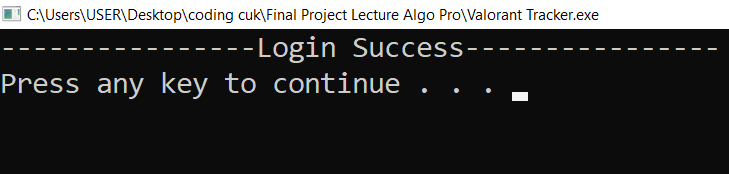


Ingame name akan diminta

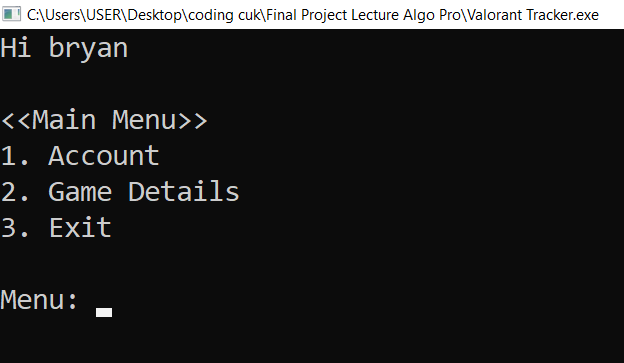


Tagline akan diminta

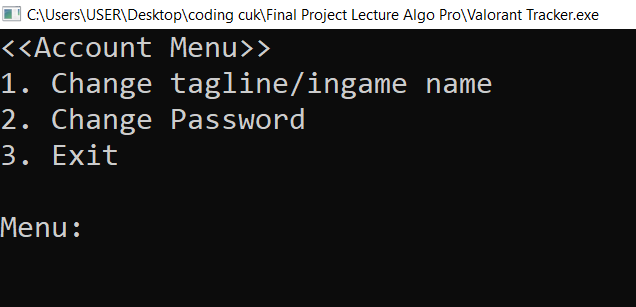


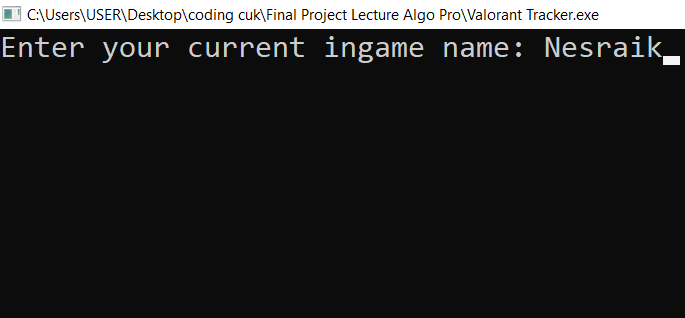
Setelah sukses tervalidasi, akan muncul tampilan seperti berikut

Menu utama program

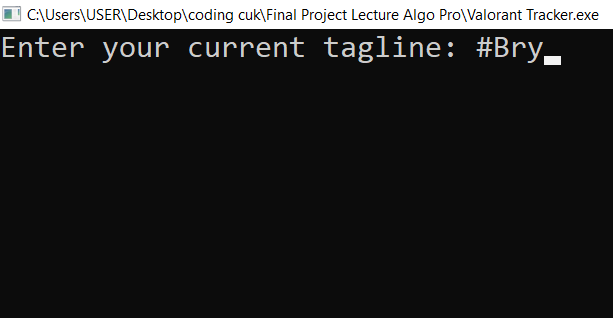


Saat memilih menu account

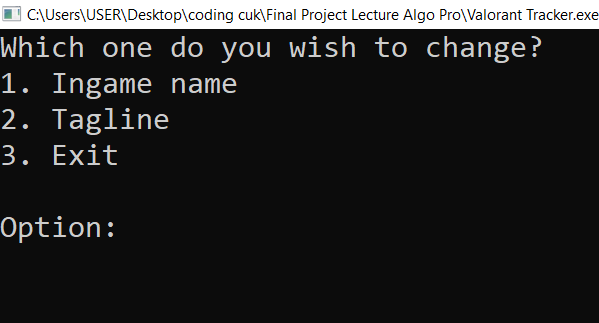


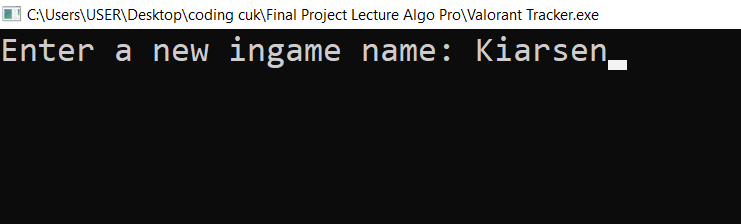
Saat memilih change tagline/ingame name, ingame name yang telah terdaftar akan diminta

Tagline yang telah terdaftar akan diminta



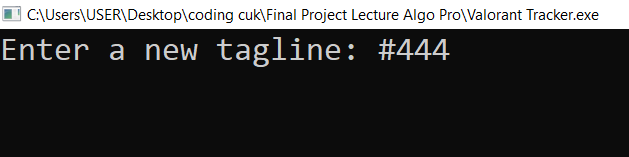
Setelah inputan sukses tervalidasi, tampilan ini akan muncul



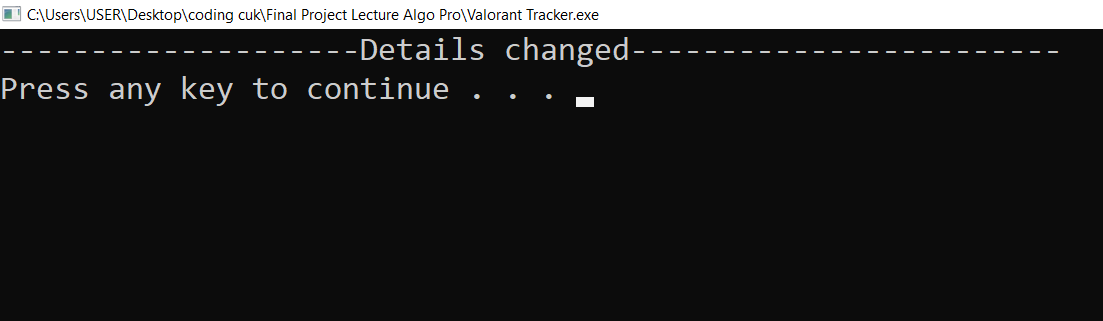
Saat memilih opsi ingame name akan muncul tampilan seperti ini. User akan diminta untuk memasukan ingame name baru

Saat memilih opsi tagline akan muncul seperti ini, user akan diminta untuk memasukkan tagline baru

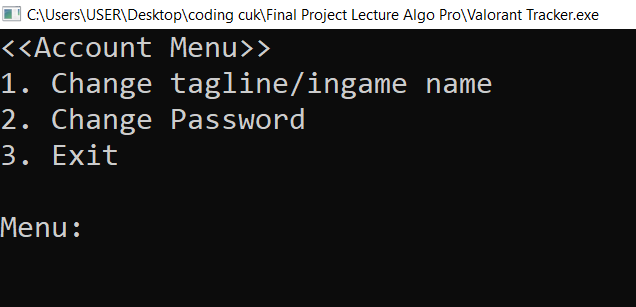
Jika user memilih tagline, user akan diminta untuk memasukan tagline baru



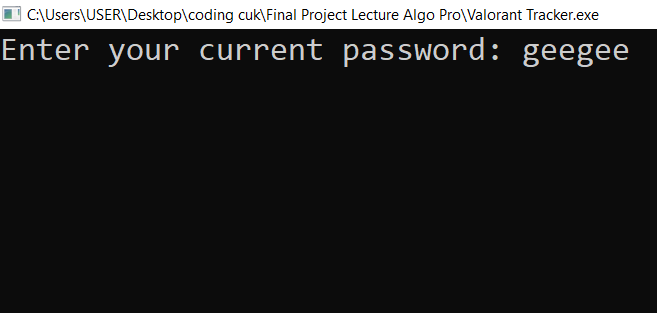
Saat dienter akan kembali ke menu sebelumnya dan ketika opsi exit terpilih akan muncul seperti ini



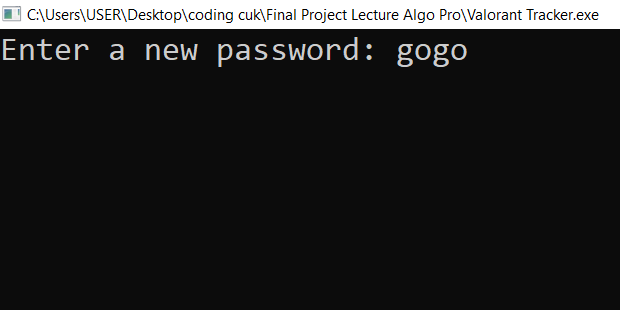
Setelah dienter akan muncul menu sebelumnya

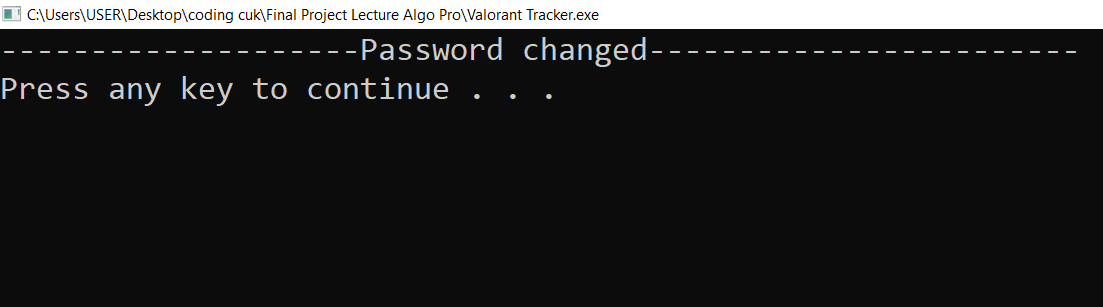


Saat opsi change password terpilih, user akan diminta untuk memasukkan password yang telah terdaftar

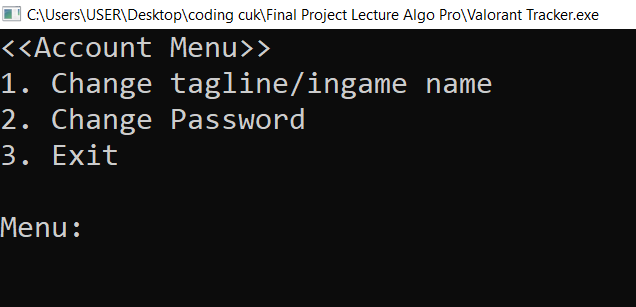


User akan diminta untuk memasukan password baru

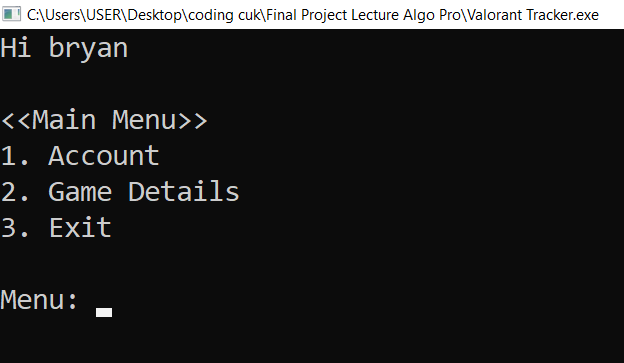


Setelah dienter akan muncul tampilan seperti ini

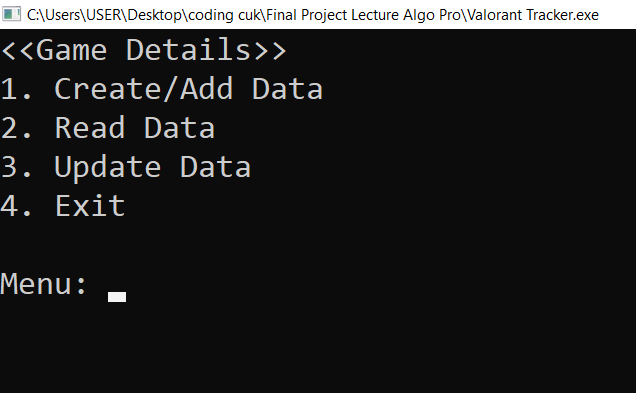
Saat user dienter, user akan kembali ke menu sebelumnya



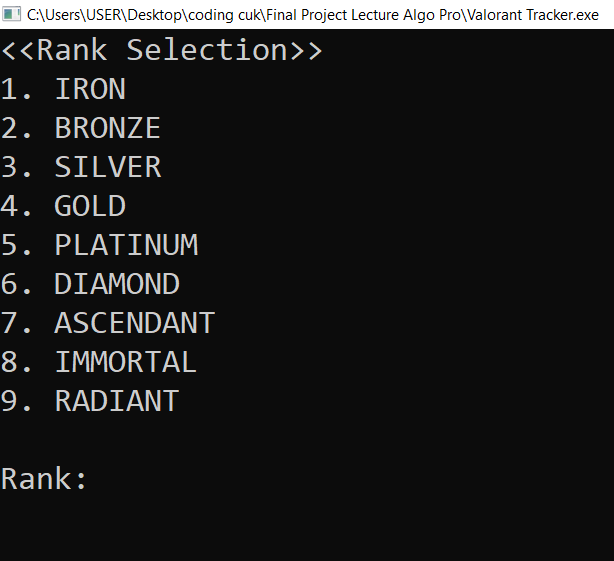
Saat menu exit terpilih, user akan kembali ke menu utama

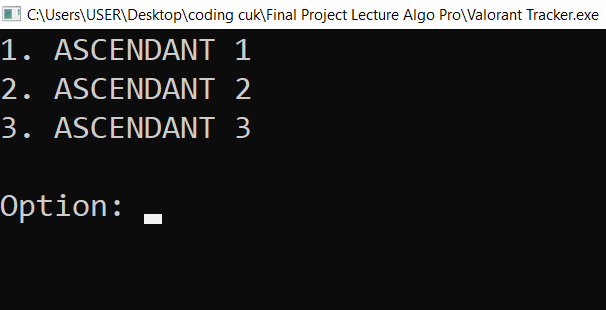


Saat user memilih opsi game details akan muncul tampilan menu game details



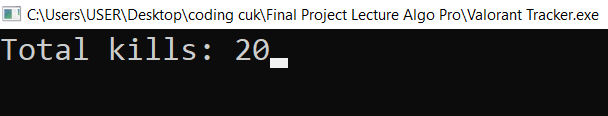
Saat user memilih opsi create/add data akan muncul tampilan seperti ini



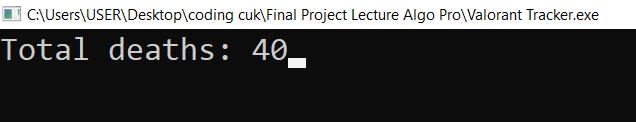
Saat user memilih opsi 7 akan muncul tampilan seperti ini

User akan diminta untuk memasukkan jumlah kill

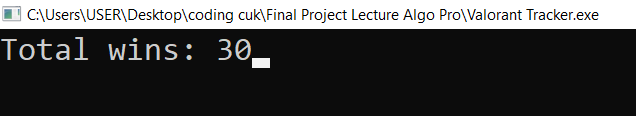
Saat user memilih opsi 3, maka akan muncul tampilan seperti ini. User diminta untuk memasukkan jumlah kills



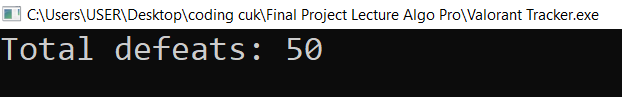
User akan diminta untuk memasukkan jumlah death



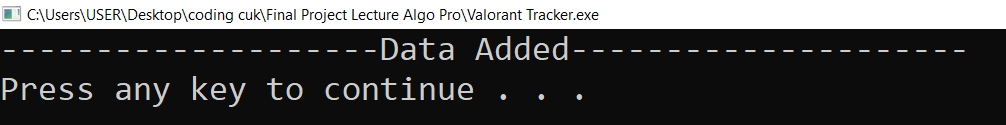
User akan diminta untuk memasukkan jumlah win

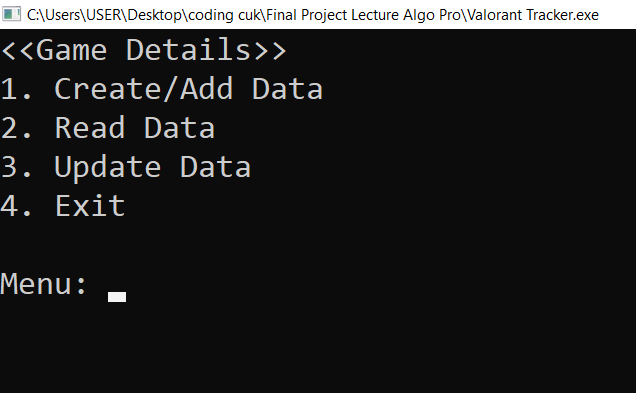


User akan diminta untuk memasukkan jumlah defeats

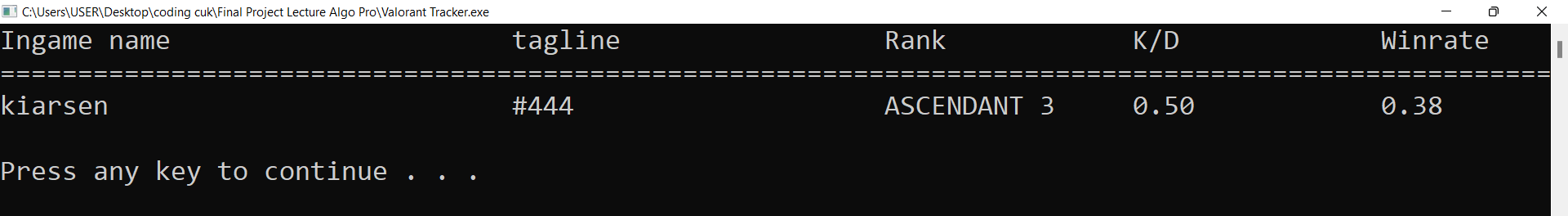


Setelah dienter akan muncul tampilan seperti ini

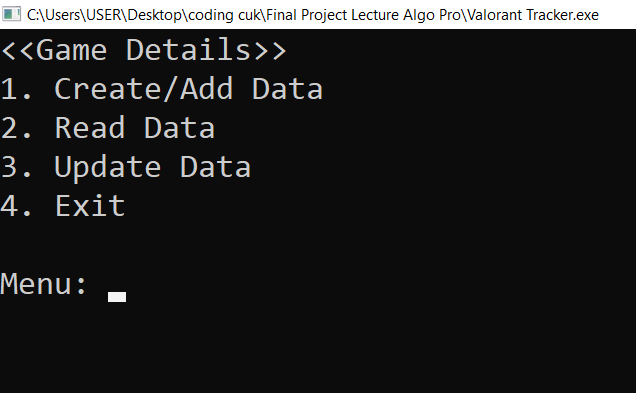


Setelah dienter akan kembali ke menu sebelumnya

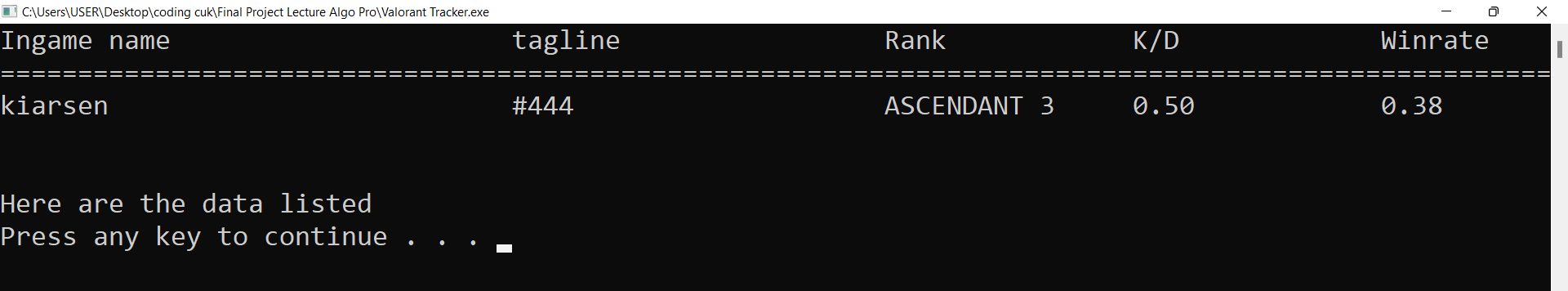
Jika opsi Read Data terpilih akan muncul tampilan seperti ini



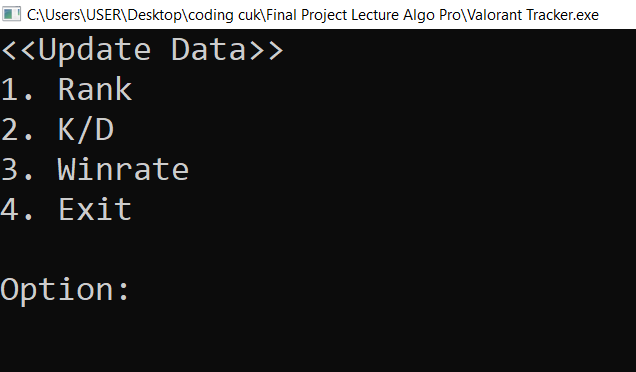
Jika dienter akan kembali ke menu sebelumnya



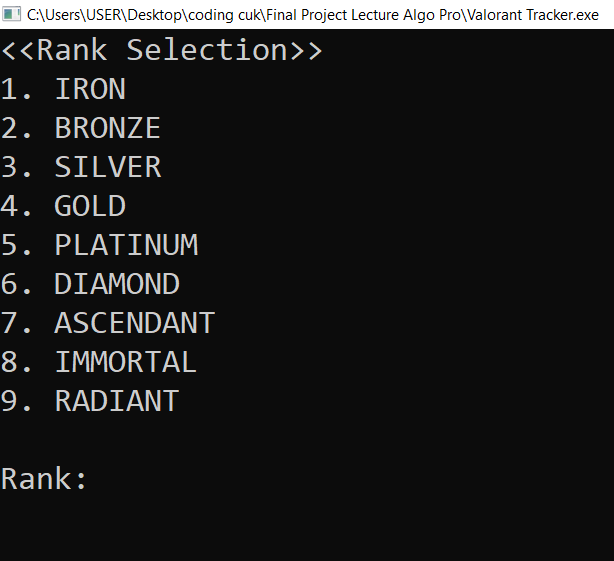
Jika opsi Update Data terpilih akan muncul tampilan seperti ini



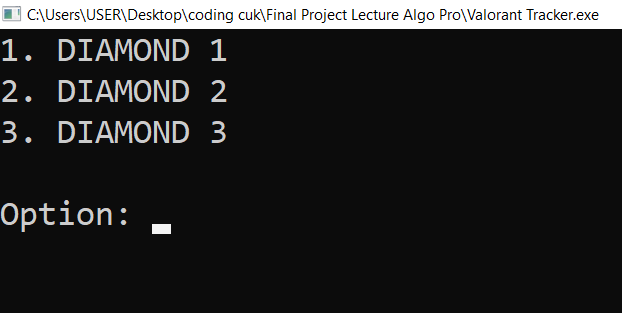
Jika dienter akan muncul tampilan seperti ini



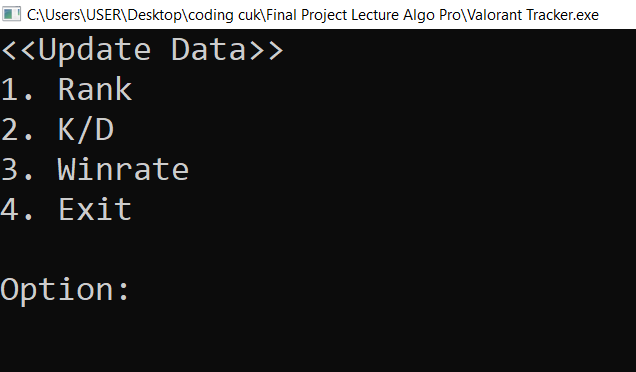
Jika opsi rank terpilih, maka akan muncul tampilan sperti ini



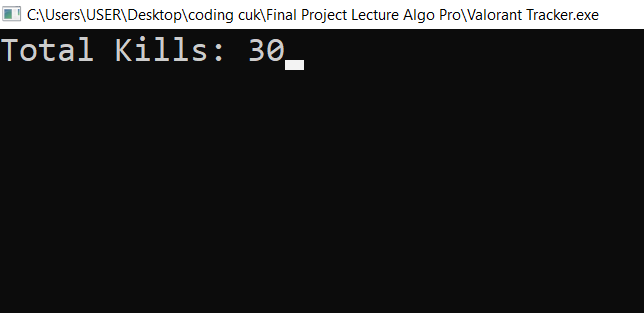
Jika opsi diamond terpilih, maka akan muncul tampilan seperti ini



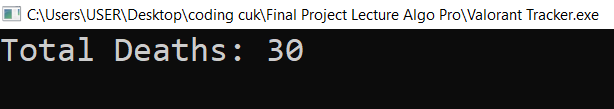
Jika opsi diamond 3 terpilih, maka akan kembali ke menu sebelumnya



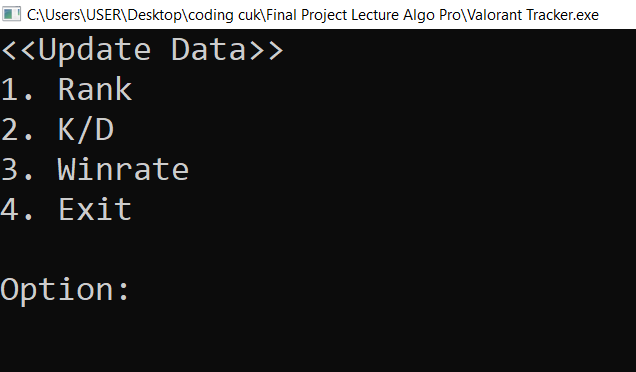
Jika opsi K/D terpilih maka akan muncul tampilan seperti ini. User akan diminta untuk memasukkan jumlah kills



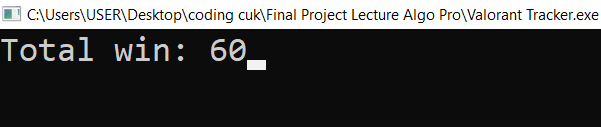
Saat dienter, maka tampilan ini akan muncul. User diminta untuk memasukkan jumlah deaths



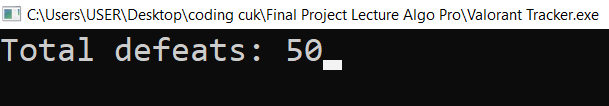
Saat dienter, user akan kembali ke menu sebelumnya



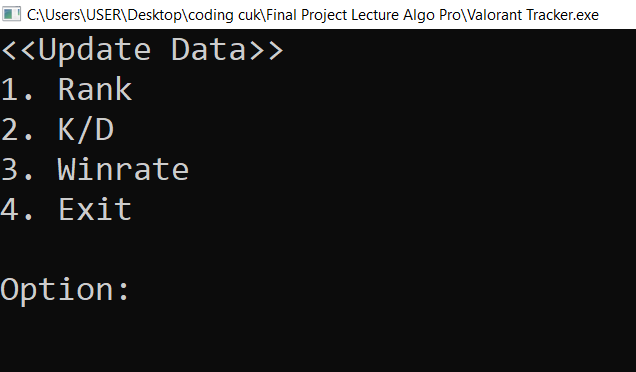
Saat opsi 3 terpilih maka tampilan seperti ini akan muncul. User akan diminta untuk memasukkan jumlah win



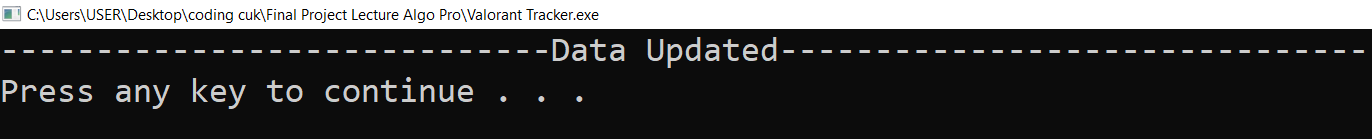
Saat dienter, tampilan seperti ini akan muncul. User akan diminta untuk memasukkan jumlah defeats



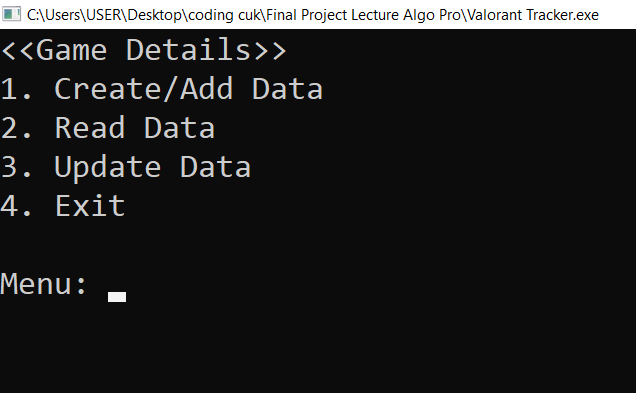
Setelah dienter, user akan kembali ke menu sebelumnya.



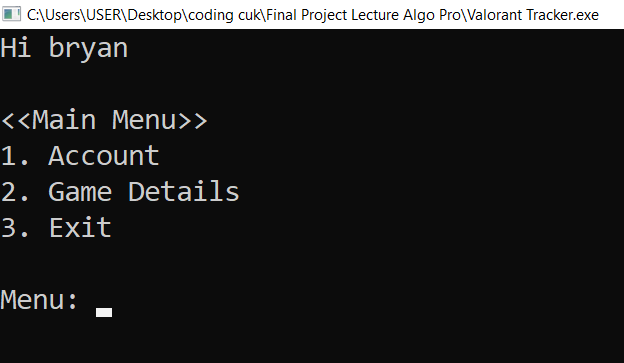
Saat opsi exit terpilih, akan muncul tampilan seperti ini



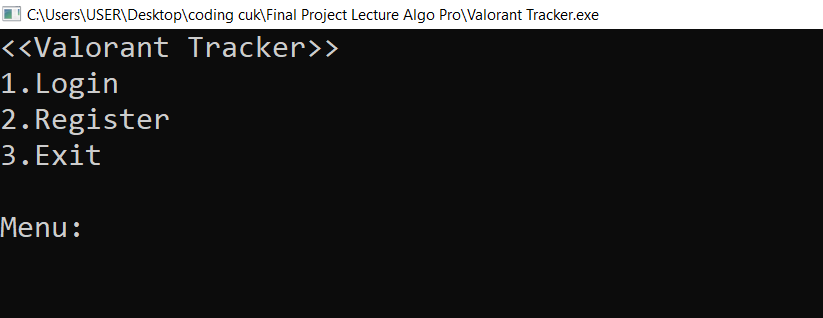
Saat dienter, user akan kembali ke menu GameDetails

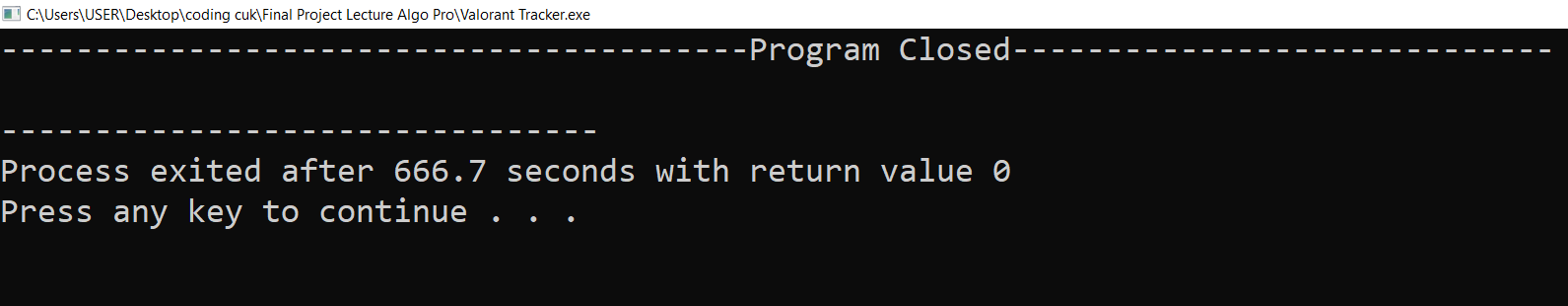


Saat user memilih exit, user akan kembali ke menu utama program



Saat user memilih exit, user akan kembali ke menu paling pertama



Saat user memilih opsi exit, tampilan seperti ini akan muncul

Bab 5

Kesimpulan

Valorant Tracker adalah program yang didesain untuk menampung, menampilkan serta mengupdate data yang telah user masukan. Valorant tracker dapat digunakan oleh player Valorant yang ingin melihat statistik game yang telah mereka main. Beberapa fitur pemrograman c seperti file processing, struct, serta selection digunakan untuk membantu pembuatan fitur program Valorant Tracker. Kedepannya, fitur program Valorant Tracker bisa terus ditambah agar penggunaannya bisa lebih fleksibel dan pemanfaatany