**PASSWORD MANAGER**

LAPORAN PROYEK AKHIR  
MATA KULIAH COMP6362004 – DATA STRUCTURES  
KELAS BE20



Oleh :  
2602160750 – BRYAN ORVILLE AUDRIC  
2602060722 – NICO HIMAWAN  
2602163336 – DAVIN NAYAKA PANDYA

Semester Genap 2022/2023

MALANG

**LEMBAR PERSETUJUAN PROYEK AKHIR**

**PASSWORD MANAGER**

MATA KULIAH COMP6362004 – DATA STRUCTURES

KELAS BE20

Semester Genap 2022/2023

Laporan akhir proyek ini adalah benar karya kami :

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **BRYAN ORVILLE AUDRIC** | **NICO HIMAWAN** | **DAVIN NAYAKA PANDYA** |
| **2602160750** | **2602060722** | **2602163336** |
|  |  |  |

**Malang, 10 – Juni 2023**

**(MIRZA RAMADHANI, S.kom, M.kom)  
D6261**

1. **LATAR BELAKANG**

Tim kami berencana untuk mengembangkan sebuah program password manager sederhana yang bertujuan untuk membantu pengguna mengelola dan mengamankan kata sandi mereka. Program ini kami buat menggunakan bahasa C dengan gabungan teknik Hash dan AVL.

Alasan kami memilih topik ini karena sistem password manager merupakan program yang unik, serta dalam penerapannya bisa menggunakan metode Data Structure AVL dan Hash yang sesuai dengan materi pembelajaran di kelas Data Structures.

Program Password Manager yang kami kembangkan adalah sebuah sistem yang memungkinkan pengguna untuk menyimpan, mengelola, dan mengamankan kata sandi mereka dengan efisien. Dalam program ini pengguna dapat membuat dan menyimpan kata sandi untuk berbagai akun. Selain itu, juga terdapat fitur pencarian yang mempermudah pengguna untuk menemukan dan mengakses kata sandi dengan cepat.

1. **TINJAUAN PUSTAKA**

4 Komponen utama program kami :

1. *AVL Tree*

*AVL Tree* (Pohon AVL) dibuat oleh Georgy Adelson-Velsky dan Evgenii Landis. *AVL Tree* adalah *Binary Search Tree* (BST) self-balancing di mana perbedaan antara ketinggian subpohon kiri dan kanan dimana untuk setiap *node* tidak boleh lebih besar dari satu. Faktor penyeimbang dari sebuah *node* adalah perbedaan antara ketinggian subpohon kiri dan kanan untuk *node* tersebut[1].

1. *Hashing*

*Hashing* adalah teknik untuk dengan cepat mengidentifikasi nilai tertentu dalam sebuah array. Metode ini menghasilkan kode hash yang unik untuk setiap elemen *array*, dan kemudian menyimpan kode hash di lokasi elemen aslinya. Ini memungkinkan pencarian cepat saat mencari nilai tertentu menggunakan kode hash yang telah dihasilkan[2].

1. *Array*

*Array* adalah metode untuk mengumpulkan data dari beberapa item dengan tipe yang sama dalam memori / penyimpanan dengan alokasi yang berurutan[3].

1. *Linked List*

*Linked list* adalah jenis *Data Structure* linier yang terdiri dari banyak node yang terhubung. Setiap node menyimpan data dan alamat node berikutnya di lokasi ini[4].

1. **DEFINISI PROGRAM**

* Pseudocode dan flowchart : terlampir pada file terpisah
* Overview
* Deskripsi Program :

Program dibuat seperti password manager pada umumnya. User dapat menyimpan password untuk beberapa akun pada beberapa aplikasi.

* Layout Design :

1. Tampilan depan

* Tampilan memiliki tata letak yang sederhana
* Merupakan tampilan yang muncul pada saat pertama kali user mengakses program.
* Terdapat judul aplikasi di bagian pojok atas dan dibawahnya terdapat beberapa opsi untuk user Create account (membuat akun), login, Delete Account (menghapus akun), dan Exit. Berikut adalah design dari setiap opsi yang ada di menu depan :

1. Create Account :

* Akan muncul kolom bagi user untuk mengisi data-data yang dibutuhkan untuk membuat akun seperti username, password, dan pertanyaan darurat.
* Masing-masing pertanyaan akan muncul satu persatu setelah user menekan “enter”.

1. Login :

* Akan muncul kolom untuk mengisi username, dan password yang telah dibuat oleh user.
* Setelah user memasukkan data akun dengan benar, user akan masuk ke dalam menu berikutnya yang bernama Tampilan Utama.

1. Delete Account :

* Akan muncul kolom untuk memasukkan username dan password akun yang akan dihapus.

1. Tampilan Utama

* Merupakan tampilan pada menu utama yang akan muncul setelah user dapat login menggunakan akun yang user buat.
* Terdapat sapaan dengan nama user di bagian atas, dan dibawahnya terdapat beberapa opsi, yakni Log Out, Change my Password Manager password, add, View, Delete. Berikut adalah design layout dari setiap opsi opsi yang ada di menu utama :

1. Log Out :

* User akan menjalankan opsi ini ketika memasukkan input “0”
* Program akan keluar dari tampilan menu utama, kembali ke menu depan

1. Change my Password Manager password :

* User akan menjalankan opsi ini ketika memasukkan input “1”
* Akan muncul kolom untuk mengisi password baru

1. Add :

* User akan menjalankan opsi ini ketika memasukkan input “2”
* Akan muncul pilihan “Back” untuk kembali ke menu utama, “Add new data” untuk membuat data password pada aplikasi atau situs baru, dan “Add New Account to Existing Data” untuk menambahkan password lain pada aplikasi atau situs yang sudah ditambahkan sebelumnya.
* Pada menu opsi “Add new data” tampilannya sederhana dengan hanya menampilkan kolom untuk mengisi site/app dan username. Setelah user input username akan muncul password yang *random generated.*
* Pada menu opsi “Add New Account to Existing Data” tampilannya tidak jauh berbeda dengan pada opsi “Add new data”. Akan muncul password yang sebelumnya sudah ada pada aplikasi yang diinginkan, lalu dibawahnya akan muncul kolom untuk menginput akun password baru.

1. View :

* User akan menjalankan opsi ini ketika memasukkan input “3”
* Akan muncul pilihan “Back” untuk kembali ke menu utama, “ View all data” untuk menampilkan semua data, dan “View Spesific Data” untuk menampilkan data dari aplikasi yang diinginkan secara spesifik.
* Username dan password akan ditampilkan dalam model tabel dengan urutan sejajar kebawah

1. Delete :

* User akan menjalankan opsi ini ketika memasukkan input “4”
* Akan muncul opsi “Back” untuk kembali ke menu utama, “Delete account from a data” untuk menghapus salah satu password pada satu akun aplikasi, dan “Delete an existing data” untuk menghapus keseluruhan data dalam akun aplikasi yang diinginkan.
* Fitur Program :

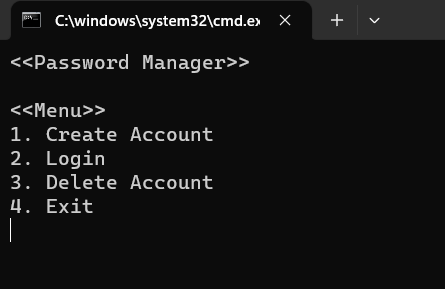
1. Menu depan :

* Create Account (membuat akun password manager)
* Login (masuk akun password manager)
* Delete Account (menghapus akun password manager)

1. Menu utama :

* Change Password Manager password (mengganti password akun Password Manager)
* Add (menambahkan data password akun aplikasi)
* View (menampilkan data password akun aplikasi)
* Delete (menghapus data password akun aplikasi)
* Detail Programs :
* AVL : Right rotation, Left Rotation, Checking balance factor,
* Hash : Hashing, chaining,
* Srand() untuk menciptakan seed dan rand syntax untuk menciptakan string acak

1. **HASIL**
2. **Screenshot tampilan**
3. Menu Depan :



Tampilan awal pada menu depan

* Create Account :

A screenshot of a computer screen

Description automatically generated with medium confidence

Ketika awal masuk fitur create account akan muncul kolom untuk mengisi username

A screenshot of a computer

Description automatically generated

Setelah selesai mengisi username, akan muncul kolom untuk mengisi password

A picture containing text, screenshot, font

Description automatically generated

Lalu setelah password, akan muncul opsi untuk memilih pertanyaan darurat

A screenshot of a computer

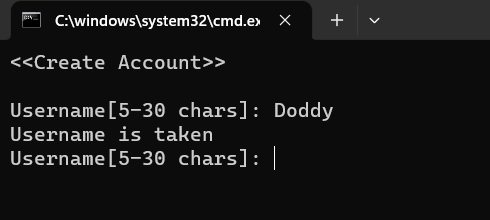
Description automatically generated with medium confidence

Setelah memilih, akan muncul kolom untuk menjawab pertanyaan tersebut

A screenshot of a computer

Description automatically generated with medium confidence

Create Account selesai



Diatas adalah contoh input validasi ketika membuat akun dengan username yang sama

* Login :

A screenshot of a computer

Description automatically generated with medium confidence

Diatas adalah contoh input validasi ketika memasukkan username yang salah ketika login

A screenshot of a computer program

Description automatically generated with medium confidence

Jika username dan password benar, maka akan muncul tampilan login success

A screenshot of a computer

Description automatically generated

Setelah berhasil login, maka akan berlanjut ke menu utama

* Delete Account :

A screenshot of a computer

Description automatically generated

Pertama akan muncul tampilan untuk memasukkan username

A screenshot of a computer

Description automatically generated with medium confidence

Setelah mengisi username, akan muncul kolom untuk mengisi password

A screenshot of a computer

Description automatically generated

Lalu muncul verifikasi apakah user benar-benar ingin menghapus akun

A screenshot of a computer program

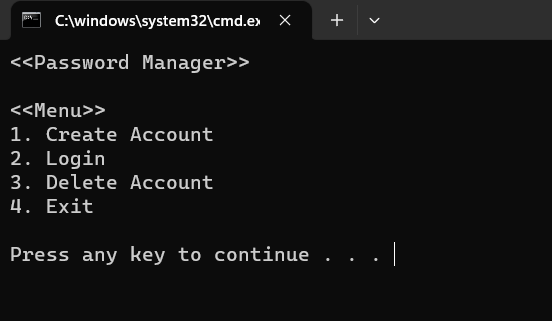
Description automatically generated with medium confidence

Bila user input “2”, maka user akan langsung kembali ke tampilan menu depan  
A screenshot of a computer program

Description automatically generated with medium confidence

Bila user input “1”, maka akun akan benar-benar dihapus

* Exit :



Diatas adalah tampilan ketika user memilih Exit untuk keluar dari program

1. Menu Utama

A screenshot of a computer

Description automatically generated

Diatas adalah tampilan menu utama

* Log Out :

A screenshot of a computer

Description automatically generated with medium confidence

Ini adalah tampilan ketika memilih opsi log out, lalu user akan kembali ke manu depan

* Change my Password Manager password :

A screenshot of a computer screen

Description automatically generated with medium confidence

Ketika memilih opsi ini, user akan diminta untuk input password baru

A screenshot of a computer screen

Description automatically generated with medium confidence

Setelah mengisi maka password dari akun Password Manager user akan berganti

* Add :

A screenshot of a computer screen

Description automatically generated with medium confidence

Ketika user memilih menu add, akan muncul tampilan dengan memberikan 2 opsi seperti yang dilampirkan diatas.

* Add new data

A black screen with white text

Description automatically generated with low confidence

Pertama akan muncul kolom untuk mengisi site/app yang akan digunakan

A screenshot of a computer

Description automatically generated with medium confidence

Lalu akan muncul kolom untuk mengisi usernam

A screenshot of a computer

Description automatically generated with medium confidence

Setelah user mengisi username, maka akan muncul password yang tercipta secara acak

* Add new Account to existing data

A screenshot of a computer

Description automatically generated with medium confidence

Pertama akan muncul kolom untuk mengisi jenis site/app yang ingin ditambahkan password lainnya

A screenshot of a computer

Description automatically generated

Lalu akan muncul password akun yang sebelumnya sudah ada, dan kolom untuk mengisi username password akun baru

A screenshot of a computer

Description automatically generated

Lalu user bisa mengisi username akun baru

A screenshot of a computer

Description automatically generated  
Setelah itu akan muncul password yang tercipta secara acak, dan data akun baru selesai dibuat.

* View :

A screenshot of a computer

Description automatically generated with medium confidence

Diatas adalah tampilan ketika user memilih menu view, akan muncul 2 opsi seperti pada gambar yang terlampir diatas.

* View all data

A screenshot of a computer

Description automatically generated

Diatas adalah tampilan ketika user memilih opsi “view all data” dimana akan muncul semua data akun yang sebelumnya sudah dibuat.

* View specific data

A screenshot of a computer

Description automatically generated with medium confidence

Awalnya user akan diminta untuk menginput nama jenis site/app

A screenshot of a computer

Description automatically generated with medium confidence

Diatas adalah tampilan bila user salah dalam menginput nama site/app

A screen shot of a computer

Description automatically generated with medium confidence

Kemudian ketika user menginput nama site/app dengan benar

A screenshot of a computer

Description automatically generated

Maka akan muncul tampilan seperti diatas dan muncul 3 opsi lanjutan dan 1 opsi untuk kembali ke tampilan menu utama

A screenshot of a computer

Description automatically generated

Jika user memilih opsi “1” maka akan menjalankan menu add new account

A screenshot of a computer

Description automatically generated

Diatas terlampir tampilan ketika user berhasil menambah akun baru

A screenshot of a computer

Description automatically generated

Diatas adalah tampilan dengan penambahan akun baru oleh user

A screenshot of a computer

Description automatically generated

Kemudian bila user menekan angka “2” maka akan langsung menjalankan menu opsi nomer 2 dimana user akan diminta untuk memilih data mana yang ingin dihapus

A screenshot of a computer

Description automatically generated

Sebagai contoh user ingin menghapus data nomer 1

A screenshot of a computer

Description automatically generated

Lalu akan muncul pertanyaan verifikasi

A screenshot of a computer

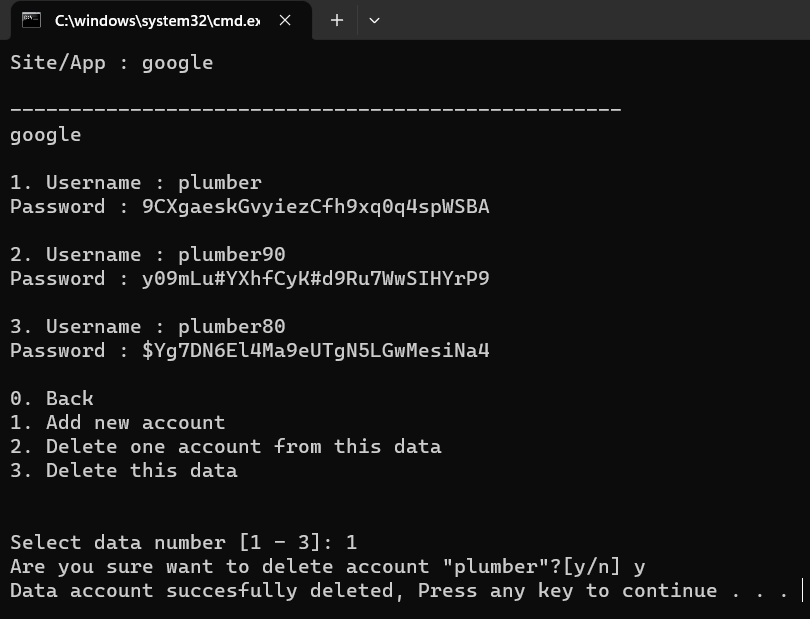
Description automatically generated

Diatas adalah contoh ketika user menjawab n (tidak). Maka penghapusan data akan dibatalkan

A screenshot of a computer

Description automatically generated

Kemudian, kali ini user meng-input “y”



Maka data akun akan dihapus

A screenshot of a computer

Description automatically generated

Dan ini adalah tampilan ketika user memilih opsi “3” untuk menghapus data, maka keseluruhan data “google” beserta akun didalamnya akan dihapus

A screenshot of a computer

Description automatically generated

Bukti dapat dihapus ketika user kembali ke menu “view” dan memilih “view all data”.

* Delete :

A screenshot of a computer

Description automatically generated

Ini adalah tampilan menu delete

* Delete Account from a data :

A screenshot of a computer

Description automatically generated with medium confidence

Pertama user akan diminta untuk memasukkan nama site/app

A black screen with white text

Description automatically generated with low confidence

User dapat mengisinya pada kolom yang tersedia

A screenshot of a computer

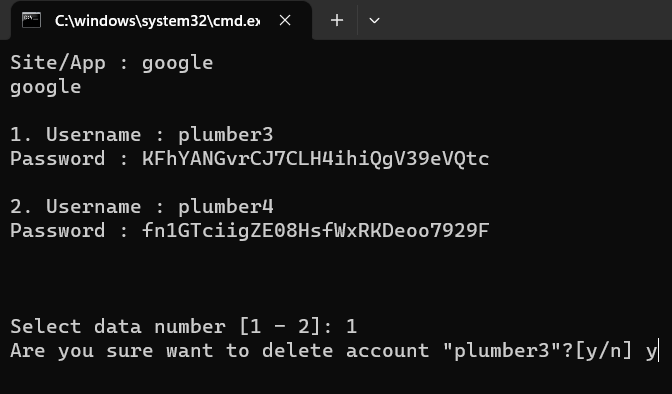
Description automatically generated

Setelah user mengisi nama site/app dengan benar, maka akan muncul akun-akun yang ada dalam data site/app tersebut. Sebagai contoh user memilih akun nomer 1 untuk dihapus

A screenshot of a computer

Description automatically generated

Kemudian akan muncul pertanyaan verifikasi, diatas adalah contoh ketika user meng-input “n”(tidak) sehingga akun batal dihapus



Lalu, diatas adalah ketika user meng-input “y” (iya)

A screenshot of a computer

Description automatically generated

Berikut adalah tampilan lanjutannya, dimana akun berhasil dihapus

A screenshot of a computer

Description automatically generated

Buktinya dapat dilihat kembali melalui opsi view di menu depan

* Delete an existing data :

A black screen with white text

Description automatically generated with low confidence

Pertama user akan diminta untuk mengisi nama site/app

A screenshot of a computer

Description automatically generated

Kemudian akan muncul akun yang terdaftar atas nama site/app tersebut dan muncul pertanyaan verifikasi

A screenshot of a computer

Description automatically generated with medium confidence

Kemudian bila ingin menghapus user dapat mengisi “y” (yes)

A screenshot of a computer

Description automatically generated

Akun berhasil dihapus

A screenshot of a computer

Description automatically generated

Diatas adalah bukti akun dihapus dapat dilihat dengan mengakses opsi view.

**B.** **Source Code :**

**Mainfile.c :**

#include"Menus.c"

#include <ctype.h>

struct node{

int NodeHeight;

char Username[30];

char Password[30];

struct node \*left;

struct node \*right;

//Nambah didalam ini diganti ama nama structnya

struct treeNode \*root[10];

char SecurityAnswer[30];

char SecurityQuestion[300];

int key;

};

char UsernameCurrent[30];

int HashKey;

int Height(struct node \*NodeInput){

if(NodeInput==NULL){

return 0;

}else{

return NodeInput->NodeHeight;

}

}

int Max(int num1,int num2){

if(num1>num2){

return num1;

}else{

return num2;

}

}

int CheckBalance(struct node \*NodeInput){

if(NodeInput==NULL){

return 0;

}else{

return Height(NodeInput->left)-Height(NodeInput->right);

}

}

struct node \*RightRotate(struct node \*OldCurrent){

struct node \*NewCurrent = OldCurrent->left;

struct node \*RightNewCurrent=NewCurrent->right;

NewCurrent->right=OldCurrent;

OldCurrent->left=RightNewCurrent;

OldCurrent->NodeHeight=1 + Max(Height(OldCurrent->left),Height(OldCurrent->right));

NewCurrent->NodeHeight=1 + Max(Height(NewCurrent->left),Height(NewCurrent->right));

return NewCurrent;

}

struct node \*LeftRotate(struct node \*OldCurrent){

struct node \*NewCurrent=OldCurrent->right;

struct node \*LeftNewCurrent = NewCurrent->left;

NewCurrent->left=OldCurrent;

OldCurrent->right=LeftNewCurrent;

OldCurrent->NodeHeight= 1 + Max(Height(OldCurrent->left),Height(OldCurrent->right));

NewCurrent->NodeHeight= 1 + Max(Height(NewCurrent->left),Height(NewCurrent->right));

return NewCurrent;

}

struct node \*MinVal (struct node \*root){

struct node \*Current = root;

while(Current->left!=NULL){

Current=Current->left;

}

return Current;

}

struct node \*insert(struct node \*root,char \*UsernameInput, char \*PasswordInput,int keyInput, char \*SecurityQuestionIn, char \*AnswerIn){

if(root==NULL){

struct node \*Newnode = (struct node\*)malloc(sizeof(struct node));

int i;

for(i=0;i<10;i++) Newnode->root[i] = NULL;

strcpy(Newnode->Username,UsernameInput);

strcpy(Newnode->Password,PasswordInput);

strcpy(Newnode->SecurityQuestion,SecurityQuestionIn);

strcpy(Newnode->SecurityAnswer,AnswerIn);

Newnode->key=keyInput;;

Newnode->NodeHeight=1;

Newnode->right=NULL;

Newnode->left=NULL;

return Newnode;

}

if(keyInput<root->key){

root->left=insert(root->left,UsernameInput,PasswordInput,keyInput,SecurityQuestionIn,AnswerIn);

}else if(keyInput>root->key){

root->right=insert(root->left,UsernameInput,PasswordInput,keyInput,SecurityQuestionIn,AnswerIn);

}

root->NodeHeight= 1 + Max(Height(root->left),Height(root->right));

int i;

for(i=0;i<10;i++) root->root[i] = NULL;

int BalanceFactor = CheckBalance(root);

if(BalanceFactor>1 && keyInput<root->left->key){

return RightRotate(root);

}

if(BalanceFactor>1 && keyInput>root->left->key){

root->left=LeftRotate(root->left);

return RightRotate(root);

}

if(BalanceFactor<-1 && keyInput>root->right->key){

return LeftRotate(root);

}

if(BalanceFactor<-1 && keyInput<root->right->key){

root->right=RightRotate(root->right);

return LeftRotate(root);

}

return root;

}

struct node \*DeleteNode(struct node \*root, int key){

if(root==NULL){

return root;

}

if(key<root->key){

root->left=DeleteNode(root->left,key);

}else if(key>root->key){

root->right=DeleteNode(root->right,key);

}else{

if(root->left==NULL){

struct node \*temp = root->right;

free(root);

return temp;

}else if(root->right==NULL){

struct node \*temp = root->left;

free(root);

return temp;

}else{

struct node \*temp = MinVal(root->right);

root->key=temp->key;

root->right=DeleteNode(root->right,temp->key);

}

}

root->NodeHeight= 1 + Max(Height(root->left),Height(root->right));

int BalanceFactor = CheckBalance(root);

if(BalanceFactor>1 && CheckBalance(root->left)>=0){

return RightRotate(root);

}

if(BalanceFactor>1 && CheckBalance(root->left)<0){

root->left=LeftRotate(root->left);

return RightRotate(root);

}

if(BalanceFactor<-1 && CheckBalance(root->right)<=0){

return LeftRotate(root);

}

if(BalanceFactor<-1 && CheckBalance(root->right)>0){

root->right=RightRotate(root->right);

return LeftRotate(root);

}

return root;

}

struct node \*Search(struct node \*root,int KeyString){

if(root==NULL){

return root;

}

if(KeyString==root->key){

return root;

}else if(KeyString<root->key){

return Search(root->left,KeyString);

}else if(KeyString>root->key){

return Search(root->right,KeyString);

}

else return NULL;

}

void CreateAccount(struct node \*\*root){

char Usernametest[30],Passwordtest[30];

int length,UsernameSum;

struct node \*Data;

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

do{

printf("Username[5-30 chars]: ");

scanf("%s",Usernametest);

getchar();

fflush(stdin);

length = strlen(Usernametest);

UsernameSum=0;

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

if(i<2){

UsernameSum+=Usernametest[i]%10;

}else{

UsernameSum+=Usernametest[i];

}

}

Data = Search((\*root),UsernameSum);

if(Data!=NULL){

printf("Username is taken\n");

}

}while(Data!=NULL || length < 5 || length >30);

do{

printf("Password[5-30 chars]: ");

scanf("%s",Passwordtest);

getchar();

fflush(stdin);

length = strlen(Passwordtest);

}while(length<5 || length>30);

char SecurityQuestion[300],Answer[30];

int menu;

do{

printf("\n<<Choose one of security question below>>\n");

printf("1. What is the middle name of your youngest child?\n");

printf("2. What was the name of your first stuffed animal?\n");

printf("3. In what city or town did your mother and father meet?\n");

printf("4. What was the first exam you failed?\n");

printf(">> ");

scanf("%d",&menu);

fflush(stdin);

switch(menu){

case 1:

system("cls");

printf("1. What is the middle name of your youngest child?\n");

strcpy(SecurityQuestion,"What is the middle name of your youngest child?");

break;

case 2:

system("cls");

printf("2. What was the name of your first stuffed animal?\n");

strcpy(SecurityQuestion,"What was the name of your first stuffed animal?");

break;

case 3:

system("cls");

printf("3. In what city or town did your mother and father meet?\n");

strcpy(SecurityQuestion,"In what city or town did your mother and father meet?");

break;

case 4:

system("cls");

printf("4. What was the first exam you failed?\n");

strcpy(SecurityQuestion,"What was the first exam you failed?");

break;

}

}while(menu<1 || menu > 4);

printf("Answer: ");

fflush(stdin);

scanf("%[^\n]",Answer);

getchar();

length = strlen(Answer);

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

if(Answer[i]<'a'){

Answer[i]= Answer[i]+32;

}

}

\*root = insert((\*root),Usernametest,Passwordtest,UsernameSum,SecurityQuestion,Answer);

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

}

void ChangePass(struct node \*root){

int length,Sum=0;

printf("<<Change Password>>\n\n");

length = strlen(UsernameCurrent);

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

if(i<2){

Sum+=UsernameCurrent[i]%10;

}else{

Sum+=UsernameCurrent[i];

}

}

struct node \*Data = Search(root,Sum);

char Passwordtest[30];

do{

printf("New Password[5-30 chars]: ");

scanf("%s",Passwordtest);

getchar();

fflush(stdin);

length = strlen(Passwordtest);

}while(length<5 || length>30);

strcpy(Data->Password,Passwordtest);

printf("<<Password Changed!!>>\n\n");

}

void DeleteAccount(struct node \*\*root){

int length;

char UsernameCheck[30];

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

printf("Username: ");

scanf("%s",UsernameCheck);

getchar();

fflush(stdin);

length=strlen(UsernameCheck);

int UsernameSum=0;

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

if(i<2){

UsernameSum+=UsernameCheck[i]%10;

}else{

UsernameSum+=UsernameCheck[i];

}

}

struct node \*Data = Search(\*root,UsernameSum);

if(Data==NULL){

printf("Account Doesnot Exist\n");

return;

}else{

char PasswordCheck[30];

printf("Password: ");

scanf("%s",PasswordCheck);

fflush(stdin);

if(strcmp(PasswordCheck,Data->Password)!=0){

printf("Invalid Password\n");

}else{

int menu;

do{

printf("Are you sure you want to delete your current account?\n");

printf("1. Yes\n");

printf("2. No\n");

printf(">> ");

scanf("%d",&menu);

fflush(stdin);

switch(menu){

case 1:

(\*root) = DeleteNode((\*root),UsernameSum);

printf("Delete Success\n");

break;

case 2:

printf("Going back to menu\n");

break;

}

}while(menu<1 || menu >2);

}

}

}

void mainMenu(struct node \*node){

char temp = 'a';

int i;

strcpy(session.name, node->Username);

for(i = 0; i < 10; i++){

session.hashRoot[i] = node->root[i];

}

while(temp != '0'){

printf("----------------------------\n Hello, %s\n----------------------------\n", session.name);

printf("0. Log Out\n1. Change my Password Manager password\n2. Add\n3. View\n4. Delete\n");

temp = getch();

switch(temp){

case '0':{

system("cls");

for(i=0;i<10;i++) node->root[i] = session.hashRoot[i];

session\_destroy();

printf("\nLogged Out, ");

return;

}

case '1':{

system("cls");

ChangePass(node);

break;

}

case '2':{

system("cls");

add();

break;

}

case '3':{

system("cls");

view();

break;

}

case '4':{

system("cls");

delete();

break;

}

}

system("cls");

}

}

void LoginAccount(struct node \*root){

if(root==NULL){

printf("No account registered\n");

return;

}

char UsernameCheck[30],PasswordCheck[30];

printf("<<Login>>\n\n");

printf("Username: ");

scanf("%s",UsernameCheck);

getchar();

fflush(stdin);

int length=strlen(UsernameCheck);

int UsernameSum=0;

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

if(i<2){

UsernameSum+=UsernameCheck[i]%10;

}else{

UsernameSum+=UsernameCheck[i];

}

}

struct node \*Data = Search(root,UsernameSum);

if(Data==NULL){

printf("Account Doesnot Exist\n");

return;

}else{

strcpy(UsernameCurrent,UsernameCheck);

int limit=3,limit2=3;

do{

printf("Password: ");

scanf("%s",PasswordCheck);

getchar();

fflush(stdin);

if(strcmp(PasswordCheck,Data->Password)!=0){

limit--;

printf("Invalid Password\n");

}else{

break;

}

}while(limit>0);

if(limit==0){

system("cls");

int menu = 0,length2;

char AnswerCheck[30],NewPassword[30];

while(menu != 1 && menu != 2){

printf("Forgot Password?\n");

printf("1. Yes\n");

printf("2. No\n");

printf(">> ");

scanf("%d",&menu);

switch(menu){

case 1:

do{

system("cls");

printf("<<Answer this question>>[Remaining Attempt: %d]\n",limit2);

printf("%s\n",Data->SecurityQuestion);

printf(">> ");

scanf("%s",AnswerCheck);

fflush(stdin);

length2 = strlen(AnswerCheck);

for(int i=0;i<length2;i++){

if(AnswerCheck[i]<'a'){

AnswerCheck[i]= AnswerCheck[i]+32;

}

}

if(strcmp(Data->SecurityAnswer,AnswerCheck)!=0){

limit2--;

continue;

}else{

printf("Enter a new Password\n");

scanf("%s",NewPassword);

strcpy(Data->Password,NewPassword);

getchar();

fflush(stdin);

return;

}

}while(limit2>0);

system("cls");

printf("<<Attempt limit is reached>>\n");

printf("<<Your account is now blocked>>\n");

printf("<<Please contact our customer service>>\n");

printf("CS (Davin): 082233445566\n");

return;

case 2:

return;

}

}//Endwhile

}

}

printf("\n\n<<Login Success>>\n\n");

system("pause");

system("cls");

mainMenu(Data);

}

int main(){

struct node \*root=NULL;

char menu;

session\_destroy();

do{

printf("<<Password Manager>>\n\n");

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

printf("1. Create Account\n");

printf("2. Login\n");

printf("3. Delete Account\n");

printf("4. Exit\n");

menu = getch();

switch(menu){

case '1':

system("cls");

CreateAccount(&root);

system("pause");

break;

case '2':

system("cls");

LoginAccount(root);

system("pause");

break;

case '3':

system("cls");

DeleteAccount(&root);

system("pause");

break;

case '4':{

return 0;

}

}

system("cls");

}while(menu!='4');

}

**Menus.c :**

#include"CRUD\_Functions.c"

#include<conio.h>

void viewExtra(treeNode \*node){

char temp = 'a';

if(node == NULL){

printf("\n Error\n\n");

return;

}

while(1){

int amount = readDataAccount(node);

printf("0. Back\n1. Add new account\n2. Delete one account from this data\n3. Delete this data\n");

temp = getch();

switch(temp){

case '0':{

return;

}

case '1':{

accountCell \*temp, \*newData = (accountCell\*) malloc(sizeof(accountCell));

printf("\nInput a new account data\n\n");

do{

printf("Username : "); scanf(" %s", newData->username);

fflush(stdin);

for(temp = node->head; temp != NULL && newData->username != temp->username; temp = temp->next);

if(temp != NULL) printf("\n Username must be unique\n\n");

} while(temp != NULL);

generate\_password(newData->password);

printf("Password (Auto Generated): %s\n\n", newData->password);

newData->next = NULL;

newData->prev = NULL;

createDataAccount(node, newData);

printf("Data account succesfully created, ");

system("pause");

break;

}

case '2':{

int temp = 0, i;

char temp2;

printf("\n\n");

while(temp <= 0 || temp > amount){

printf("Select data number [1 - %d]: ", amount);

scanf(" %d",&temp);

}

accountCell \*cell = node->head;

for(i=0;i<temp-1;i++) cell = cell->next;

while(1){

printf("Are you sure want to delete account \"%s\"?[y/n] ", cell->username); scanf(" %c", &temp2);

if(tolower(temp2) == 'n'){

printf("\nDelete canceled, ");

system("pause");

return;

}

else if(tolower(temp2 == 'y')) break;

}

deleteDataAccount(cell, node);

printf("Data account succesfully deleted, ");

system("pause");

break;

}

case '3':{

printf("\nAre you sure you want to delete this data?[y/n]");

while(1){

temp = getch();

if(tolower(temp) == 'y') break;

else if(tolower(temp) == 'n') return;

}

deleteData(node);

printf("\n\nData succesfully deleted, ");

system("pause");

return;

break;

}

}

system("cls");

}

}

void add(){

char temp = 'a';

while(1){

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

printf("0. Back\n1. Add New Data\n2. Add New Account to existing data\n");

temp = getch();

switch(temp){

case '0':{

return;

}

case '1':{

system("cls");

treeNode \*newData = (treeNode\*) malloc(sizeof(treeNode)), \*node = session.root;

printf("Site/App : "); scanf(" %[^\n]", newData->name);

session\_change\_root(newData->name[0]);

node = traverse(newData->name, session.root);

if(node != NULL){

printf("\nSame site/app name has been found for ");

viewExtra(node);

return;

}

newData->prev = NULL;

newData->left = NULL;

newData->right = NULL;

accountCell \*newAcc = (accountCell\*) malloc(sizeof(accountCell));

newData->head = newAcc;

printf("Username : "); scanf(" %[^\n]", newAcc->username);

generate\_password(newAcc->password);

printf("Password (Auto Generated): %s\n\n", newAcc->password);

newAcc->next = NULL;

newAcc->prev = NULL;

createData(newData);

printf("Data succesfully created, ");

system("pause");

return;

break;

}

case '2':{

system("cls");

char str[30];

printf("Site/App : "); scanf(" %[^\n]", str);

session\_change\_root(str[0]);

treeNode \*node = traverse(str, session.root);

if(node == NULL){

printf("\n ERROR\n");

system("pause");

return;

}

readDataAccount(node);

accountCell \*temp, \*newData = (accountCell\*) malloc(sizeof(accountCell));;

printf("\nInput a new account data\n\n");

do{

printf("Username : "); scanf(" %s", newData->username);

fflush(stdin);

for(temp = node->head; temp != NULL && newData->username != temp->username; temp = temp->next);

if(temp != NULL) printf("\n Account name must be unique\n\n");

} while(temp != NULL);

generate\_password(newData->password);

printf("Password (Auto Generated): %s\n\n", newData->password);

newData->next = NULL;

newData->prev = NULL;

createDataAccount(node, newData);

printf("Data account succesfully created, ");

system("pause");

return;

break;

}

}

system("cls");

}

}

void view(){

char temp = 'a';

int i = dataCheck();

if(i == 0){

printf("\nThere is no data, ");

system("pause");

return;

}

while(1){

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

printf("0. Back\n1. View all data\n2. View specific data\n");

temp = getch();

switch(temp){

case '0':{

return;

}

case '1':{

system("cls");

int i;

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

for(i=0;i<10;i++) if(session.hashRoot[i] != NULL) readData(session.hashRoot[i]);

system("pause");

break;

}

case '2':{

system("cls");

char str[30];

printf("Site/App : "); scanf(" %[^\n]", str);

session\_change\_root(str[0]);

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

viewExtra(traverse(str, session.root));

system("pause");

break;

}

}

system("cls");

}

}

void delete(){

char temp = 'a';

int i = dataCheck();

if(i == 0){

printf("\nThere is no data, ");

system("pause");

return;

}

while(1){

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

printf("0. Back\n1. Delete account from a data\n2. Delete an existing data\n");

temp = getch();

switch(temp){

case '0':{

return;

}

case '1':{

system("cls");

char str[30], temp2;

printf("Site/App : "); scanf(" %[^\n]", str);

session\_change\_root(str[0]);

treeNode \*node = traverse(str, session.root);

int amount = readDataAccount(node);

int temp = 0, i;

printf("\n\n");

while(temp <= 0 || temp > amount){

printf("Select data number [1 - %d]: ", amount);

scanf(" %d", &temp);

}

accountCell \*cell = node->head;

for(i=0;i<temp-1;i++) cell = cell->next;

while(1){

printf("Are you sure want to delete account \"%s\"?[y/n] ", cell->username); scanf(" %c", &temp2);

fflush(stdin);

if(tolower(temp2) == 'n'){

printf("\nDelete canceled, ");

system("pause");

return;

}

else if(tolower(temp2 == 'y')) break;

}

deleteDataAccount(cell, node);

printf("Data account succesfully deleted, ");

system("pause");

return;

break;

}

case '2':{

system("cls");

char str[30], temp;

printf("Site/App : "); scanf(" %[^\n]", str);

session\_change\_root(str[0]);

treeNode \*node = traverse(str, session.root);

readDataAccount(node);

printf("\nAre you sure you want to delete this data?[y/n] ");

while(1){

scanf(" %c", &temp);

fflush(stdin);

if(tolower(temp) == 'y') break;

else if(tolower(temp) == 'n') return;

}

deleteData(node);

printf("\n\nData succesfully deleted, ");

system("pause");

return;

break;

}

}

system("cls");

}

}

**CRUD\_Functions.c :**

#include"Base\_AVL.c"

//------- CRD for tree -------

void createData(treeNode \*newNode){

if(session.root == NULL){

session.root = newNode;

newNode->prev = NULL;

session.hashRoot[newNode->name[0] % 10] = session.root;

}

else{

treeNode \*node = session.root;

while(1){

if(strcmp(newNode->name, node->name) < 0){

if(node->left != NULL) node = node->left;

else {

node->left = newNode;

newNode->prev = node;

break;

}

}

else if(strcmp(newNode->name, node->name) > 0){

if(node->right != NULL) node = node->right;

else {

node->right = newNode;

newNode->prev = node;

break;

}

}

else{

printf("\a\nError\n\n");

break;

}

}

AVL\_check(session.root);

}

}

void readData(treeNode \*node){ // In Order

if (node->left != NULL) readData(node->left);

int i = 1;

accountCell \*cell;

printf("%s\n\n", node->name);

for(cell = node->head; cell != NULL; cell = cell->next){

printf("%d. Username : %s\nPassword : %s\n\n", i, cell->username, cell->password);

i++;

}

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

if (node->right != NULL) readData(node->right);

}

void freeList(treeNode \*node){

accountCell \*cell = node->head, \*temp = node->head;

while(cell != NULL){

cell = temp->next;

free(temp);

temp = cell;

}

}

void deleteData(treeNode \*node){

treeNode \*parent = node->prev, \*child;

char hashValue = node->name[0];

if (node->left == NULL){

if(node->right == NULL){

if(node->prev == NULL)session.root = NULL;

else if(node == parent->left) parent->left = NULL;

else parent->right = NULL;

freeList(node);

free(node);

}

else{

child = node->right;

if(node == session.root) session.root = child;

else if(node == parent->left)parent->left = child;

else parent->right = child;

child->prev = parent;

freeList(node);

free(node);

}

}

else if(node->right == NULL){

child = node->left;

if(node == session.root) session.root = child;

else if(node == parent->left) parent->left = child;

else parent->right = child;

child->prev = parent;

freeList(node);

free(node);

}

else{ // Both left sub tree and right sub of node tree exist

child = node->left;

for(; child->right != NULL; child = child->right);

strcpy(node->name, child->name);

node->head = child->head;

deleteData(child);

}

session.hashRoot[(int) hashValue % 10] = session.root;

AVL\_check(session.root);

}

//------- CRUD for data accounts -------

void createDataAccount(treeNode \*node, accountCell \*newCell){

accountCell \*cell = node->head;

if(cell == NULL) node->head = newCell;

else{

for(; cell->next != NULL; cell = cell->next);

cell->next = newCell;

newCell->prev = cell;

}

}

int readDataAccount(treeNode \*node){

int i = 1;

accountCell \*cell;

printf("%s\n\n", node->name);

for(cell = node->head; cell != NULL; cell = cell->next){

printf("%d. Username : %s\nPassword : %s\n\n", i, cell->username, cell->password);

i++;

}

return i-1;

}

void deleteDataAccount(accountCell \*cell, treeNode \*node){

accountCell \*before = cell->prev, \*after = cell->next;

if(before == NULL) node->head = after;

else before->next = after;

if(after != NULL) after->prev = before;

free(cell);

}

**Base\_AVL.c :**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#include<time.h>

//------- Structures, Globals, and Base Functions -------

typedef struct accountCell{

struct accountCell \*next, \*prev;

char username[30];

char password[30];

}accountCell;

typedef struct treeNode{

struct treeNode \*left, \*right, \*prev;

struct accountCell \*head;

char name[30];

int level, AVL;

}treeNode;

//node -> for tree

//cell -> for linked list

struct Session{

treeNode \*hashRoot[10];

treeNode \*root;

char name[30];

};

struct Session session; // Global variable related to log in

void session\_destroy(){

int i;

for(i=0;i<10;i++) session.hashRoot[i] = NULL;

session.root = NULL;

strcpy(session.name, "\0");

}

void session\_change\_root(char str){

session.root = session.hashRoot[str % 10];

}

treeNode \*traverse(char str[], treeNode \*node){

if(node == NULL) return NULL;

else if(strcmp(str, node->name) < 0) traverse(str, node->left);

else if(strcmp(str, node->name) > 0) traverse(str, node->right);

else return node;

return NULL;

}

void generate\_password(char str[30]){

int i, temp;

srand(time(NULL));

for(i = 0; i < 29; i++){

temp = rand() % 64;

if(temp < 26) str[i] = temp + 65;

else if(temp < 52) str[i] = temp + 71;

else if(temp < 62) str[i] = temp - 4;

else str[i] = temp - 27;

}

str[29] = '\0';

}

int dataCheck(){

int i, j = 0;

for(i=0;i<10;i++) if(session.hashRoot[i] != NULL)j++;

return j;

}

//------- AVL Functions -------

void AVL\_L\_rotate(treeNode \*node){

treeNode \*parent = node->prev;

node->prev = parent->prev;

parent->prev = node;

parent->left = node->right;

node->right = parent;

treeNode \*temp = node->prev;

if(temp == NULL) session.root = node;

else if(temp->left == parent) temp->left = node;

else temp->right = node;

temp = parent->left;

if(temp!=NULL)temp->prev = parent;

session.hashRoot[node->name[0] % 10] = session.root;

}

// // -----Troubleshooting Rotation Helper-----

// printf("\n\nr %s\n", session.root->name);

// printf("\nn %s\n", node->name);

// if(node->prev != NULL) printf("n-P %s\n", node->prev->name);

// if(node->left != NULL) printf("n-L %s\n", node->left->name);

// if(node->right != NULL) printf("n-R %s\n", node->right->name);

// printf("\np %s\n", parent->name);

// if(parent->prev != NULL) printf("p-P %s\n", parent->prev->name);

// if(parent->left != NULL) printf("p-L %s\n", parent->left->name);

// if(parent->right != NULL) printf("p-R %s\n", parent->right->name);

void AVL\_R\_rotate(treeNode \*node){

treeNode \*parent = node->prev;

node->prev = parent->prev;

parent->prev = node;

parent->right = node->left;

node->left = parent;

treeNode \*temp = node->prev;

if(temp == NULL) session.root = node;

else if(temp->left == parent) temp->left = node;

else temp->right = node;

temp = parent->right;

if(temp!=NULL)temp->prev = parent;

session.hashRoot[node->name[0] % 10] = session.root;

}

void AVL\_violation(treeNode \*node, int type);

void AVL\_check(treeNode \*node){

if(node == NULL) return;

else if(node->left == NULL && node->right == NULL){

node->level = 1;

node->AVL = 0;

}

else{

treeNode \*left = node->left, \*right = node->right;

if(left != NULL) AVL\_check(left);

if(right != NULL) AVL\_check(right);

if(left == NULL){

node->level = right->level+1;

node->AVL = - right->level;

}

else if(right == NULL){

node->level = left->level+1;

node->AVL = left->level;

}

else {

if(left->level > right->level) node->level = left->level+1;

else node->level = right->level +1;

node->AVL = left->level - right->level;

}

if(node->AVL > 1) AVL\_violation(node, 0);

else if(node->AVL < -1) AVL\_violation(node, 1);

}

}

void AVL\_violation(treeNode \*node, int type){

treeNode \*child;

if(type == 0){

child = node->left;

if(child->AVL > 0){ //LL

AVL\_L\_rotate(child);

}

else{ //LR

AVL\_R\_rotate(child->right);

AVL\_L\_rotate(child->prev);

}

}

else{

child = node->right;

if(child->AVL > 0){ //RL

AVL\_L\_rotate(child->left);

AVL\_R\_rotate(child->prev);

}

else{ //RR

AVL\_R\_rotate(child);

}

}

AVL\_check(node);

}

1. **DAFTAR PUSTAKA**

[1] GeeksforGeeks. (2023a, March 22). *AVL Tree Data Structure*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-avl-tree/

[2] *Hashing in data structure: What, types, and functions*. KnowledgeHut. (n.d.). https://www.knowledgehut.com/blog/programming/hashing-in-data-structure

[3] GeeksforGeeks. (2023b, May 9). *Array Data Structure*. GeeksforGeeks. https://www.geeksforgeeks.org/array-data-structure/

[4] GeeksforGeeks. (2023b, May 8). *Linked list data structure*. GeeksforGeeks. https://www.geeksforgeeks.org/data-structures/linked-list/

1. **LAINNYA**

Jobdesk :

|  |  |
| --- | --- |
| Bryan Orville Audric | : Program Code, Flowchart, Pseudocode |
| Nico Himawan | : Program Code, Flowchart, Pseudocode |
| Davin Nayaka Pandya | : Design Fitur, Laporan, debugging |

Nilai oleh Bryan :

|  |  |
| --- | --- |
| Bryan Orville Audric | 94 |
| Nico Himawan | 89 |
| Davin Nayaka Pandya | 84 |

Nilai oleh Nico :

|  |  |
| --- | --- |
| Bryan Orville Audric | 90 |
| Nico Himawan | 95 |
| Davin Nayaka Pandya | 85 |

Nilai oleh Davin :

|  |  |
| --- | --- |
| Bryan Orville Audric | 95 |
| Nico Himawan | 90 |
| Davin Nayaka Pandya | 85 |

LEMBAR PENILAIAN

**PASSWORD MANAGER**

MATA KULIAH COMP6362004 – DATA STRUCTURES   
KELAS BE20

Semester Genap 2022/2023

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NAMA  MAHASIWA** | **NILAI** | | | | **NILAI** | | | | **NILAI** | | | | **TOTAL KREDIT** | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |  |
| 2602160750 – BRYAN ORVILLE AUDRIC |  |  |  |  | 20% | 30% | 30% | 20% |  |  |  |  |  |
| 2602060722 – NICO HIMAWAN |  |  |  |  | 20% | 30% | 30% | 20% |  |  |  |  |  |
| 2602163336 – DAVIN NAYAKA PANDYA |  |  |  |  | 20% | 30% | 30% | 20% |  |  |  |  |  |
| **TOTAL** | | | | | | | | | | | | |  | |

**KETERANGAN :**

**• Skala Penilaian : 0 sd 100   
• Komponen**1 : Laporan   
2 : Produk   
3 : Pengetahuan dan Solusi   
4 : Presentasi

**Malang,……-……………. ………**

**(MIRZA RAMADHANI, S.kom, M.kom)  
D6261**