Foundation Programing

Action on Weight

Contents

[Design 2](#_Toc437481424)

[Screen Design 2](#_Toc437481425)

[Program 1 (Nurses) 2](#_Toc437481426)

[Program 2 (Consultant) 5](#_Toc437481427)

[JSP Diagram (Nurses) 7](#_Toc437481428)

[JSP Diagram (Consultant) 8](#_Toc437481429)

[Testing 9](#_Toc437481430)

[Testing Plan (Nurses) 9](#_Toc437481431)

[Testing Plan (Consultant) 10](#_Toc437481432)

[Test Log (Nurses) 10](#_Toc437481433)

[Test Log (Consultant) 15](#_Toc437481434)

[Evaluation 16](#_Toc437481435)

[Improvements 18](#_Toc437481436)

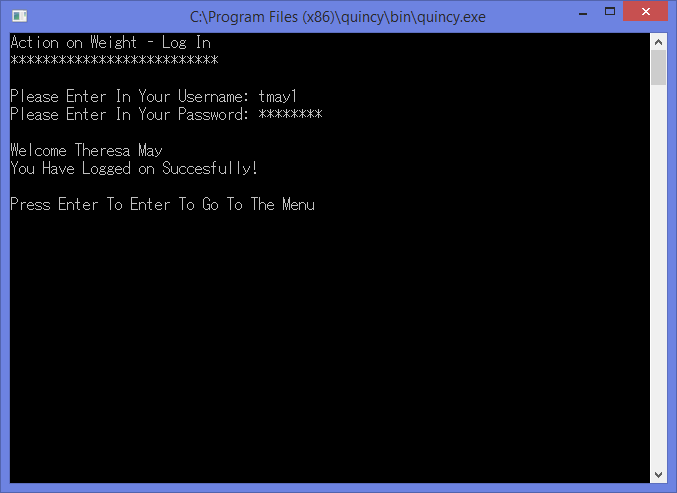
[Conclusion 19](#_Toc437481437)

[References 19](#_Toc437481438)

[C Programming Code 20](#_Toc437481439)

[Program 1 (Nurse) 20](#_Toc437481440)

[Program 2 (Consultant) 41](#_Toc437481441)



# Design

## Screen Design

### Program 1 (Nurses)

Action on Weight – Log In

Please Enter In Your Username:

Please Enter In Your Password:

The user will first be greeted by a log in screen where there will be required to enter in their username and password.

Welcome “Full Name”

You have successfully logged on!

If you successfully log on then you will be greeted with the message stating so.

Incorrect Login

Press Enter To Try Again!

However if you are unsuccessful at logging in you will be greeting with this message stating so.

Action on Weight – Menu | “Full Name”

1. Enter Patient Details

2. Change Password

3. Add A New User

4. Remove A User

5. Log Off

Please Select An Option (1, 2, 3, 4 or 5):

Once a user has successfully logged in they will then be brought to the menu where they can select a few options.

If the user picks the first option they will then be taken to a page where they can enter in the various details of the patient, once they have done that they will then have the option to enter in details for another patient or go back to the menu.

Error: Only integer values should be entered

OR

Error: You need to enter in 6 digits for the DoB (dd/mm/yy)

Action on Weight – Enter Patient Details | “Full Name”

1. Enter The Patient's First Name:

2. Enter The Patient's Surename:

3. Enter The Patient's Date of Birth (e.g. 010196):

4. Enter The Patient's Height (In Centimetres):

5. Enter The Patient's Waist Measurement (In Centimetres):

6. Enter The Patient's Waist Measurement (In Centimetres):

7. Enter In Any Patient Comments:

Would You Like To Enter In Details For Another Patient?

1. Yes

2. No (Go Back To The Menu)

Please Select An Option (1 or 2):

However if the user enters in an invalid data input for some of the options they will then be greeted with an error message and asked to enter in a correct input value.

Action on Weight – Change Password | “Full Name”

Username: “Username”

Enter LogIn Password:

Enter New Password:

Verify Password:

You have successfully changed your password “Full Name”

If the user chooses the 2nd option they will be taken to this screen where they can enter their login password, a new password and confirm their new password.

However if the user does not enter in a password in the correct format or they enter in passwords that don’t match they will be greeted by this error message.

Action on Weight – Add New User | “Full Name”

Please Enter In The Users' Full Name:

Please Enter In The Users' Username:

Please Enter In The Users' Password:

You are about to add the user with the following details:

Full Name:

UserID:

Password:

Are You Sure?

1. Yes

2. No (Back To Menu)

Please select an option (1 or 2):

“Full Name” has now been added!

Would You Like To Add Another User?

1. Yes

2. No (Back To Menu)

Please select an option (1 or 2):

Error: Password must be exactly 8 characters long and start with a letter

OR

Error: Passwords don't match

If the user selects the 3rd option they will be taken to a screen where they can add another user to the text file, the user will also be presented with options to double check that they want to add the user and to ask if they would like to add another user or return to the main menu.

If the user chooses the 4th option they will then be taken to a screen where they can select a use that exists within the text file that they wish to remove, they are then asked to confirm that they wish to remove that user and if so a message saying they have successfully done so will be displayed.

You have selected a user outside of the range

Action on Weight – Remove User | “Full Name”

List Of Currently Registered Members:

1. A

2. B

3. C

Please select the user you would like to remove:

You are about to remove “Full Name”

Are You Sure?

1. Yes

2. No (Back To Menu)

Please select an option (1 or 2):

“Full Name” has successfully been removed

Remaining Registered Members:

1. A

2. C

However if you choose a number that is outside of the range of the available members you will be greeted with this error message.

### Program 2 (Consultant)

Action on Weight – Read Patient File | Consultant

Please Enter In The Patient's File Name (Surname+Date of Birth):

File: “Filename”

Upon opening the program the consult will then be asked to enter the patient file name.

Once a file has been found successfully it will then display the contents of the file on the screen, the consultant will then have the option of going back to enter in a new patient file name.

Action on Weight – Patient Details | Consultant

Full Name:

Date Of Birth: / /

Height: centimetres

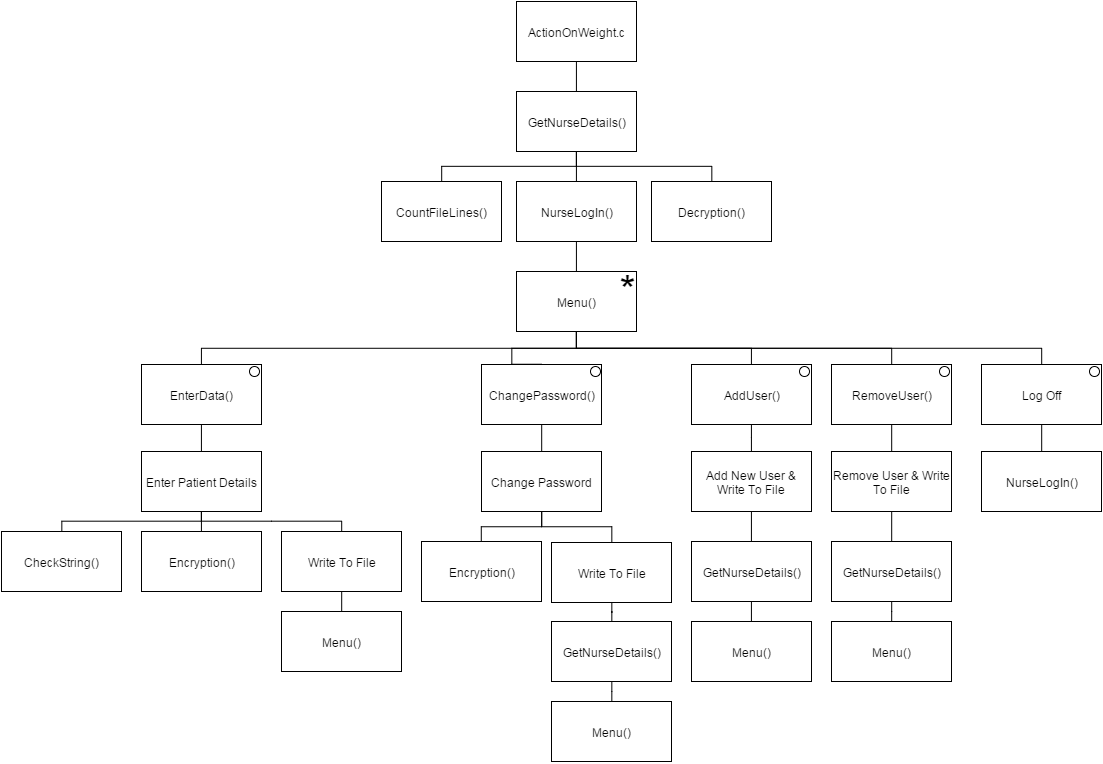
Weight: kilograms

Waist: centimetres

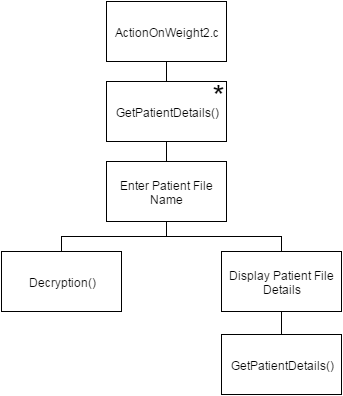
Comments:

Press Enter To Select A New Patient File

## JSP Diagram (Nurses)



## JSP Diagram (Consultant)



# Testing

## Testing Plan (Nurses)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Input Data | Expected Output | Comments |
| 1 | tmay1 | Username: tmay1 | simple valid data |
| 2 | password | Password: \*\*\*\*\*\*\*\* | simple valid data |
| 3 | x | Error | Invalid data type |
| 4 | 1 | Display first option in menu | simple valid data |
| 5 | 2 | Display second option in menu | simple valid data |
| 6 | 3 | Display third option in menu | simple valid data |
| 7 | 4 | Display forth option in menu | simple valid data |
| 8 | 5 | Display fifth option in menu | simple valid data |
| 9 | passw | Error – Password too short | Lower boundary |
| 10 | Pasword1234 | Error – Password too long | Upper boundary |
| 11 | 1assword | Error – Password to start with letter | Invalid data type |
| 12 | 1st January 2000 | Error – Date of birth should be in dd/mm/yy format. | Invalid data type |
| 13 | 1.4 metres | Error – Invalid input | Invalid data type |
| 14 | 12 inches | Error – Invalid input | Invalid data type |
| 15 | 55 stones | Error – Invalid input | Invalid data type |
| 16 | Jack  Robertson  060801 | Roberston060801.pat file | simple valid data |
| 17 | Jake Cake  jcake  password | Data to be added to text file | simple valid data |
| 18 | 0 | Error – value outside of range | Lower boundary |
| 19 | 8 | Error – value outside of range | Upper boundary |
| 20 | password | tewwasvh | simple valid data |
| 21 | tewwasvh | password | simple valid data |
| 22 | 3 | Delete’s 3rd user from the system | simple valid data |
| 23 | “patient details” | Writes patient details to text file | simple valid data |
| 24 | Check if patient files are encrypted | Patient files are encrypted | simple valid data |
| 25 | Check to see if newly updated password is encrypted | Passwords are encrypted | simple valid data |

## Testing Plan (Consultant)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Input Data | Expected Output | Comments |
| 1 | Somarib040696 | Patient File: Somarib040696.pat | simple valid data |
| 2 | “read .pat file” | Displays patient details | simple valid data |

## Test Log (Nurses)

|  |  |  |
| --- | --- | --- |
| Step | Actual Output | Comments |
| 1 & 2 |  | OK, username is correctly displayed and the password field prints asterisk correctly. |
| 3 |  | OK, the program will not continue if it receives an invalid input for an option. |
| 4 |  | OK, the first function loads when the user chooses the first option |
| 5 |  | OK, the second function loads when the user chooses the second option |
| 6 |  | OK, the third function loads when the user chooses the third option |
| 7 |  | OK, the forth function loads when the user chooses the forth option |
| 8 |  | OK, the fifth function loads when the user chooses the fifth option |
| 9 |  | OK, the password cannot be shorter than 8 characters. |
| 10 |  | OK, the password cannot be longer than 8 characters. |
| 11 |  | OK, password must start with a letter |
| 12 |  | OK, DoB must be entered in the dd/mm/yy format |
| 13 |  | Failed, user can enter in incorrect data |
| 14 |  | Failed, user can enter in incorrect data |
| 15 |  | Failed, user can enter in incorrect data |
| 16 |  | OK, patient file is successfully generated |
| 17 |  | OK, new users are able to be successfully added |
| 18 |  | OK, numbers outside rage aren’t accepted |
| 19 |  | OK, numbers outside rage aren’t accepted |
| 20 & 21 |  | OK, passwords are properly written to the text file and in turn are successfully decrypted when read into the program. |
| 22 |  | OK, users can be deleted from the text file |
| 23 |  | OK, patient files are being stored correctly |
| 24 |  | OK, details in the patient text files are all encrypted |
| 25 |  | OK, passwords are successfully saved and updated in the text file.s |

## Test Log (Consultant)

|  |  |  |
| --- | --- | --- |
| Step | Actual Output | Comments |
| 1 |  | OK |
| 2 |  | OK |

# Evaluation

This programming assignment was yet again quite a step up in difficulty compared to the second one as such it challenged me a lot and required me to learn about and use an even wider range of skill. While it was much shorter than the second one hitting the higher mark points was quite complicated to do. Nonetheless I still enjoyed working on this last assignment for Foundation Programming.

List of C library functions used:

* IF/ELSE/ELSE IF
* WHILE/FOR/DO
* SWITCH CASE
* ARRAYS
* STRUCTURES
* POINTERS
* PRINTF
* FPRINTF
* FOPEN/FCLOSE
* FGETS
* SCANF
* STRCPY
* STRCAT
* STRLEN
* STRCHR
* ISALPHA
* ISPRINT
* GETCH

I first started with the CountFileLines() function, this is the backbone behind almost every loop in the program as it calculates how many lines they are in the text file this is then used to determine how many times the loops need to repeat. This is quite a crucial feature as due to the use of the delete and add new user functions the numbers of lines in the text file will change and so having a just a static number in the loops would cause errors. The function makes use of a WHILE which keeps reading the line until the end of the line then an IF statement is used to increment a variable by 1, at the end of the loop the variable will now store the integer value of how many lines they are in the text file. Once that is done this function can then be called wherever it is needed and the variable can then be used in the various FOR loops in the program.

I then went onto making the encryption and decryption functions. They operate by receiving a string in a variable from the global structure, calculating the length of the string using STRLEN so the loop knows how many times to repeat. Within this loop each character of the string is then converted into an integer which displays its ASCII value which is then incriminated/decreased by 4 values for just the alphanumeric values, special characters are left alone. To get the program to wrap round back to A/0 the program will see if the character is an ‘w’ or higher and instead of adding 4 it will subtract/add 22 this will bring it all the way back to A in terms of encryption or back to ‘w’ in terms of decryption for example. The same principles apply to numbers as well just with a subtract/add 6 to the ASCII values. Once the loop has been completed the now encrypted/decrypted string is then copied back into the global structure using STRCPY which can then be called in another function and STRCPY’d again back into the desired variable. This was an efficient way to programming this as instead of having these functions copy pasted everywhere the variable can just be passed into the function to be processed and the resulted can just be copied back.

I next started with the GetNurseDetails() function that would open a text file, read and then store the required data from the file into some arrays. I used FOPEN to open the nurse login details text file and FGETS to read and store the lines into arrays, since FGETS adds a new lines character ‘\n’ to the end of the string it gets I then used an IF statement with STRCHR and POINTERS to locate this new line character in the string that was being read and replace it with a null character. The passwords where then decrypted after that, once that was done the items were then stored into arrays.

After that the NurseLogIn() function is then used. This function asks for the user id and password and checks it against the user id and password stored in the arrays from the GetNurseDetails() function, if there is a correct match then the user will be able to successfully log into the system. A GETCH() and ISPRINT is used to display asterisks in the password field instead of the actual letters, this is done by hiding the actual input characters from the keyboard from being displayed and printing asterisk in its place. This makes the program more secure as over lookers won’t be able to see the actual password being entered on the screen. The passwords are then re-encrypted and stored back in their arrays.

Following a successful login the user will then be directed to the Menu() function, here they will get given 5 option, to change the password, delete a user, add a user, log off and add patient details. A WHILE loop is used here for validation to make sure that the system only proceeds with a valid input and a SWITCH CASE is then used to take the user to the required function for each option.

With the ChangePassword() function it takes a copied version of the logged in user’s full name, user id and password (before it’s re-encrypted) from the NurseLogIn() function using STRCPY into a variable in the global structure. Within the loop the ‘i’ variable is counted so that it can be used to determine which place the user in the array is, this is so that you can accurate update the correct password in the array. For extra security the user is asked to re-enter in their password again which will then be checked against the copied version of the password to make sure there is a match. If the user is successful then they will then be able to enter in a new password, validation is then used here to make sure that the entered password meets the requirements of the specification. ISALPHA is used to check the first character of the entered password to make sure it starts with a letter, in conjunction with making sure the password has to be exactly 8 characters long in a WHILE loop. Unless these requirements are met the program will not continue. Once that was done the user is then asked to verify their password by entering it in again, another WHILE loop is then used to make sure that the 2 password match, this reduces the chances of the user entering in a mistake in their password. After that has been successful the password is then re-encrypted and then the text file is opened using FOPEN and everything is then re-printed to the text file using FPRINTF, the GetNurseDetails() function is the called which will re-read the text file and the updated password will then be stored in the array of the program.

Next is the RemoveUser() function, this works by listing all the current users in the text file, the user is then asked to pick a user they wish to remove, validation is then used to ask the user if they are sure they want to remove this user. If the select yes the position of where the user in gets filled by fill by the item after it using STRCPY and in turn every item moves up one place in the array once this is done the selected user’s details will no longer be in the array, this array is then printed to the text file and so now the text file will not include the removed user’s details anymore. The GetNurseDetails() function is then called which makes the system recheck the txt file and update the array with the newly removed user.

The log off option simply takes the user back to the NurseLogIn() function, this adds extra security to the system as if the nurse needs to leave the machine unattended for a while they can log off so no other unauthorised user can access the system during that time.

The AddUser() function adds a new user to the text file, I first ask for the user to input the new user’s full name, userid and password, the password field is likewise subjected to the same kind of validation as the ChangePassword() function which ensures that the entered password meets the specification. The newly entered details are then displayed and the user is asked if they wish to proceed with the addition. If they wish to proceed then FOPEN is used to open the text file but this time in ‘a’ append mode, this means that the printed data will just get added to the end of the file. Once the file had been opened in append mode I used FPRINTF to append the new data to the existing file. The GetNurseDetails() function is then called which makes the system recheck the txt file and update the array with the newly added user.

Lastly with the EnterData() function the nurses can now enter in the various details of the patients, since the data needs to be encrypted every entered value is then passed to the encryption function which then encrypts the data and passes it back to its desired variable to be stored. All inputs that could be validated where validated for example with the data of birth input was passed to a new function called CheckString() this function makes use of ISALPHA in a loop that checks to make sure that every character in the string is an integer, it’s used together with a WHILE loop to ensure that it is also only 6 characters long. This is to enforce the DDMMYY date format as it is used to combine with the surname variable using STRCAT to automatically generate the patient file names. The same function and method was then applied to the other input fields such as the waist, height and weight measurements. Once all the data has been successfully entered and encrypted and the patient file name generated I used FOPEN to create a file with the generated file name and then print all the encrypted inputted data to it. When that was successfully done the user is then asked if they would like to enter in more data or go back to the main menu where they can select other options.

The 2nd program was much more simple in comparison, FOPEN is used to open the file and FGETS is used to read the file and pass the data to the decrypt function which then passes back the now decrypted data using STRCPY which is then stored in arrays. The data in the arrays is then printed for the consultant to view.

## Improvements

Nonetheless there are quite a few improvements that could be made, for starters a forget password option could be implemented where if a user can’t remember their password they can have some option on how to get around that which could be done via email or locally like answering a few security questions which would allow you to change/reset your password.

Another improvement could be to allow the nurses to update the patient files, this would be quite useful since the data that is entered into the patient files such as height, weight and waist will not remain the same for the patients forever. So allowing their files to be read and updated would be beneficial.

## Conclusion

I learned even more skills with this assignment, while it wasn’t as long as the 2nd programming assignment hitting all of the higher marks criteria did make the program quite a bit more complicated. However the task has made me feel even more confident in the c programming language. Overall I found that adjusting the values in memory first and then writing to file to be the easier option of adjusting text files.

# References

Counting the number of lines in a text file - http://www.sanfoundry.com/c-program-number-lines-text-file/

Deleting items in an array - http://www.programmingsimplified.com/c/source-code/c-program-delete-element-from-array

Remove newline character from fgets() http://stackoverflow.com/questions/2693776/removing-trailing-newline-character-from-fgets-input

# C Programming Code

## Program 1 (Nurse)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ActionOnWeight.c

Jonathan Somarib

A login program for entered patient files

December 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <conio.h>

#include <ctype.h>

/\*Global Structure\*/

struct data

{

char buffer[256];

char nursename[255];

char nursepassword[255];

char nursefullname[100][20];

char username[100][20];

char password[100][20];

char encryptdata[255];

char firstname[255];

char surname[255];

char dob[255];

char height[255];

char waist[255];

char weight[255];

char comment[255];

char newuserfullname[255];

char newusername[255];

char newuserpassword[255];

int count\_lines;

char chr;

char loggedinuser[255];

char loggedinusername[255];

char loggedinpassword[255];

int check;

char test[255];

int results;

};

/\*Function Prototypes\*/

struct data NurseLogIn(struct data action);

struct data decryption(struct data action);

struct data encryption(struct data action);

struct data EnterData(struct data action);

struct data GetNurseDetails(struct data action);

struct data AddNewUser(struct data action);

struct data CountFileLines(struct data action);

struct data Menu(struct data action);

struct data RemoveUser(struct data action);

struct data ChangePassword(struct data action);

struct data CheckString(struct data action);

int main()

{

struct data action =

{

0

};

action = GetNurseDetails(action);

action = NurseLogIn(action);

return 0;

}

/\*Menu Function that that gives the user 5 options to choose from

a switch case is then used to direct the user to their desired operation\*/

struct data Menu(struct data action)

{

int option;

int valid1;

system("cls");

printf("Action on Weight - Menu | %s\n", action.loggedinuser);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("1. Enter Patient Details\n");

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

printf("3. Add A New User\n");

printf("4. Remove A User\n");

printf("5. Log Off\n\n");

printf("Please select an option (1, 2, 3, 4 or 5): ");

valid1 = scanf("%d", &option);

while (option < 1 || option > 5 || valid1 != 1)

{

printf("Please select an option (1, 2, 3, 4 or 5): ");

scanf("%d", &option);

fflush(stdin);

}

switch (option)

{

case 1:

action = EnterData(action);

break;

case 2:

action = ChangePassword(action);

break;

case 3:

action = AddNewUser(action);

break;

case 4:

action = RemoveUser(action);

break;

case 5:

printf("You have sucessfully logged out!");

getch();

action = NurseLogIn(action);

break;

}

return action;

}

/\*This function reads the text file and counts how many lines

there are in the text file this is then stored in a varible in the

global structure so that then infomation can be called into other functions\*/

struct data CountFileLines(struct data action)

{

FILE \*fp3;

action.count\_lines = 0;

/\*Looks for and pens "username.txt" in read mode\*/

fp3 = fopen("username.txt", "r");

action.chr = getc(fp3);

while (action.chr != EOF)

{

if (action.chr == '\n')

{

action.count\_lines = action.count\_lines + 1;

}

action.chr = getc(fp3);

}

fclose(fp3);

return action;

}

/\*This is the function which is used to remove a user from the text file\*/

struct data RemoveUser(struct data action)

{

FILE \*fp4;

int position, c, n, k, valid1, option;

action = CountFileLines(action);

n = action.count\_lines / 3;

system("cls");

printf("Action on Weight - Remove User Account | %s\n", action.loggedinuser);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("List Of Currently Registered Members\n\n");

/\*FOR loop that lists all the users currently in the text file\*/

for ( c = 0 ; c < n ; c++ )

{

k = c + 1;

printf("%d: %s\n", k, action.nursefullname[c]);

}

printf("\nPlease select the user you would like to remove: ");

valid1 = scanf("%d", &position);

/\*While loop for data validation\*/

while (position >= n + 1 || valid1 != 1)

{

printf("You have selected a user outside of the range");

printf("\nPlease select the user you would like to remove: ");

scanf("%d", &position);

fflush(stdin);

}

printf("\nYou are about to remove: %s", action.nursefullname[position - 1]);

printf("\nAre you sure?");

printf("1. Yes\n");

printf("2. No (Go Back To Menu)\n");

printf("Please select an option (1 or 2): ");

valid1 = scanf("%d", &option);

fflush(stdin);

/\*While loop for data validation\*/

while (option < 1 || option > 2 || valid1 != 1)

{

printf("Please select an option (1 or 2): ");

scanf("%d", &option);

fflush(stdin);

}

/\*A switch case that runs different parts of the code depending on the

users option\*/

switch (option)

{

case 1:

/\*FOR loop that replaces the selected user's details with the next item in the array

this effectively removes the selected user from the array\*/

for ( c = position - 1 ; c < n - 1 ; c++ )

{

strcpy(action.nursefullname[c], action.nursefullname[c+1]);

strcpy(action.username[c], action.username[c+1]);

strcpy(action.password[c], action.password[c+1]);

}

printf("\n%s has sucessfully been removed", action.nursefullname[position - 1]);

printf("\nRemaining Registered Members: \n\n");

fp4 = fopen("username.txt", "w");

/\*This for loop then prints the new array with the removed user to the text file

and then displays the remaining users within the program\*/

for ( c = 0 ; c < n - 1 ; c++ )

{

k = c + 1;

printf("%d: %s\n", k, action.nursefullname[c]);

fprintf(fp4, "%s\n", action.nursefullname[c]);

fprintf(fp4, "%s\n", action.username[c]);

fprintf(fp4, "%s\n", action.password[c]);

}

fclose(fp4);

getch();

action = GetNurseDetails(action);

action = Menu(action);

break;

case 2:

action = Menu(action);

break;

}

return action;

}

/\*Add new user function is used to add new users to the text file\*/

struct data AddNewUser(struct data action)

{

FILE \*fp2;

int valid1;

int option;

int len;

int test;

system("cls");

fp2 = fopen("username.txt", "a");

printf("Action on Weight - Add New User Account | %s\n", action.loggedinuser);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Please Enter In The Users' Full Name: ");

scanf(" %[^\n]", &action.newuserfullname);

printf("Please Enter In The Users' Username: ");

scanf(" %[^\n]", &action.newusername);

printf("Please Enter In The Users' Password: ");

scanf(" %[^\n]", &action.newuserpassword);

len = strlen(action.newuserpassword);

/\*ISALPHA is used to check if the first character of the string is a number

or not\*/

test = isalpha(action.newuserpassword[0]);

while (len > 8 || len < 8 || test == 0)

{

printf("\nPassword must be exactly 8 characters long and start with a letter\n");

printf("Enter New Password: ");

scanf("%s", &action.newuserpassword);

fflush(stdin);

len = strlen(action.newuserpassword);

test = isalpha(action.newuserpassword[0]);

}

printf("\n\nYou are about to add the user with the following details: ");

printf("\nFull Name: %s", action.newuserfullname);

printf("\nUserID: %s", action.newusername);

printf("\nPassword: %s", action.newuserpassword);

printf("\n\nAre you sure?\n");

printf("1. Yes\n");

printf("2. No (Go Back To Menu)\n");

printf("Please select an option (1 or 2): ");

valid1 = scanf("%d", &option);

fflush(stdin);

while (option < 1 || option > 2 || valid1 != 1)

{

printf("Please select an option (1 or 2): ");

scanf("%d", &option);

fflush(stdin);

}

switch (option)

{

case 1:

/\*Copies the password string into a varible in the global structure

which is then passed into the encrypt function and then copied back

in the gobal varible and finally back to the original varible\*/

strcpy(action.encryptdata, action.newuserpassword);

action = encryption(action);

strcpy(action.newuserpassword, action.encryptdata);

fprintf(fp2, "%s\n", action.newuserfullname);

fprintf(fp2, "%s\n", action.newusername);

fprintf(fp2, "%s\n", action.newuserpassword);

fclose(fp2);

printf("\n%s has now been added\n\n", &action.newuserfullname);

printf("Would You Like To Add Another User?\n");

printf("1. Yes\n");

printf("2. No (Go Back To Menu)\n");

printf("Please select an option (1 or 2): ");

valid1 = scanf("%d", &option);

fflush(stdin);

while (option < 1 || option > 2 || valid1 != 1)

{

printf("Please select an option (1 or 2): ");

valid1 = scanf("%d", &option);

fflush(stdin);

}

switch (option)

{

case 1:

action = AddNewUser(action);

break;

case 2:

action = GetNurseDetails(action);

action = Menu(action);

break;

}

break;

case 2:

action = Menu(action);

break;

}

return action;

}

/\*Change password function,this takes the password from the text file

stores it in an array then the required password is then automatically chosen

depending on who logs into the system and is then replaced with the newly entered password

and finally printed to the text file\*/

struct data ChangePassword(struct data action)

{

FILE \*fp5;

char password[255];

char password1[255];

char password2[255];

int len;

int lines;

int i;

int test;

action = CountFileLines(action);

lines = action.count\_lines / 3;

system("cls");

printf("Action on Weight - Change Password | %s\n", action.loggedinuser);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Username: %s\n", action.loggedinusername);

printf("Enter LogIn Password: ");

scanf("%s", &password);

fflush(stdin);

/\*Compares the entered LogIn password agaisnt the one stored int he array

to make sure they match\*/

if ((strcmp(action.loggedinpassword, password) == 0))

{

printf("Enter New Password: ");

scanf("%s", &password1);

fflush(stdin);

len = strlen(password1);

test = isalpha(password1[0]);

/\*Ensures that the newly entered password meets the requirements of

the specification\*/

while (len > 8 || len < 8 || test == 0)

{

printf("Password must be exactly 8 characters long and start with a letter\n");

printf("Enter New Password: ");

scanf("%s", &password1);

fflush(stdin);

len = strlen(password1);

test = isalpha(password1[0]);

}

printf("Verify New Password: ");

scanf("%s", &password2);

fflush(stdin);

/\*Ensures that the new password and the verify password field

are they same\*/

if ((strcmp(password1, password2) == 0))

{

/\*Password is encryped and then written to the text file\*/

strcpy(action.encryptdata, password1);

action = encryption(action);

strcpy(password1, action.encryptdata);

strcpy(action.password[action.check], password1);

fp5 = fopen("username.txt", "w");

for (i = 0; i < lines; i++)

{

fprintf(fp5, "%s\n", action.nursefullname[i]);

fprintf(fp5, "%s\n", action.username[i]);

fprintf(fp5, "%s\n", action.password[i]);

}

fclose(fp5);

printf("\nYou have sucessfully changed your password %s\n", action.loggedinuser);

getch();

action = GetNurseDetails(action);

action = Menu(action);

}

else

{

printf("Passwords don't match");

getch();

action = ChangePassword(action);

}

}

else

{

printf("Incorrect Password!");

getch();

action = ChangePassword(action);

}

return action;

}

/\* GetNurseDetails function looks for the "username.txt" where all the

nurse log in details are saved, reads them and then stores them in arrays\*/

struct data GetNurseDetails(struct data action)

{

FILE \*fp;

int i;

char \*posi;

char \*posi2;

char \*posi3;

int lines;

action = CountFileLines(action);

lines = action.count\_lines / 3;

fp = fopen("username.txt", "r");

for (i = 0; i < lines; i++)

{

fgets(action.nursefullname[i], 20, fp);

/\*This searches the read string from the file for a new line character

and replaces it with a null character since fgets adds a trailing new line character

at the end of every string it reads and stores\*/

if ((posi3 = strchr(action.nursefullname[i], '\n')) != NULL)

{

\*posi3 = '\0';

};

fgets(action.username[i], 20, fp);

if ((posi = strchr(action.username[i], '\n')) != NULL)

{

\*posi = '\0';

};

fgets(action.password[i], 20, fp);

/\*The password is decypted as soon as it's read before it is finally

stored in an array\*/

strcpy(action.encryptdata, action.password[i]);

action = decryption(action);

strcpy(action.password[i], action.encryptdata);

if ((posi2 = strchr(action.password[i], '\n')) != NULL)

{

\*posi2 = '\0';

};

}

fclose(fp);

return action;

}

/\*The NurseLogIn function uses the data that was read and stored

from the GetNurseDetails function and checks it agaist the inputted username

and password to see if there is a correct match, if so the user will then

be able to access the system\*/

struct data NurseLogIn(struct data action)

{

char c;

int i;

int pos = 0;

int check = 0;

char \*posi;

char \*posi2;

int lines;

action = CountFileLines(action);

lines = action.count\_lines / 3;

system("cls");

printf("Action on Weight - Log In\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Please Enter In Your Username: ");

scanf(" %s", &action.nursename);

printf("Please Enter In Your Password: ");

/\*Prints asterisk instead of the actual password in the

password field\*/

do

{

c = getch();

if ( isprint(c) )

{

action.buffer[pos++] = c;

printf("%c", '\*');

}

else if ( c == 8 && pos )

{

action.buffer[pos--] = '\0';

printf("%s", "\b \b");

}

}

while ( c != 13 );

for (i = 0; i < lines; i++)

{

if ((strcmp(action.nursename, action.username[i]) == 0) && (strcmp(action.buffer, action.password[i]) == 0))

{

/\*Copies the logged in users detail into a new global varible

which is then later used in the Change Password function\*/

action.check = i;

strcpy(action.loggedinuser, action.nursefullname[i]);

strcpy(action.loggedinusername, action.username[i]);

strcpy(action.loggedinpassword, action.buffer);

for (i = 0; i < lines; i++)

{

/\*Decrypted passwords are then re-encrypted and stored back in the arrays\*/

strcpy(action.encryptdata, action.password[i]);

action = encryption(action);

strcpy(action.password[i], action.encryptdata);

}

printf("\n\nWelcome %s\n", action.loggedinuser);

printf("You Have Logged on Succesfully!\n\n");

printf("Press Enter To Enter To Go To The Menu");

getch();

action = Menu(action);

}

}

printf("\n\nIncorrect login!\n\n");

printf("Press Enter To Try Again");

getch();

action = NurseLogIn(action);

return action;

}

/\*Enter Data function is used for entering the patient details\*/

struct data EnterData(struct data action)

{

FILE \*fp1;

int i;

int valid1;

char filename[255];

char name[255];

char birth[255];

int option;

int len;

system("cls");

printf("Action on Weight - Enter Patient Details | %s\n", action.loggedinuser);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Enter The Patient's First Name: ");

scanf(" %s", &action.firstname);

fflush(stdin);

/\*Copies the chosen string into a varible in the global structure

which is then passed into the encrypt function and then copied back

in the gobal varible and finally back to the original varible\*/

strcpy(action.encryptdata, action.firstname);

action = encryption(action);

strcpy(action.firstname, action.encryptdata);

printf("Enter The Patient's Surename: ");

scanf(" %s", &action.surname);

fflush(stdin);

strcpy(action.encryptdata, action.surname);

strcpy(name, action.surname);

action = encryption(action);

strcpy(action.surname, action.encryptdata);

printf("Enter The Patient's Date of Birth (e.g. 010196): ");

scanf(" %s", &action.dob);

fflush(stdin);

len = strlen(action.dob);

/\*Copies the chosen string into a varible in the global structure

which is then passed into the CheckFileLines function which then returns

a value that is used in for the WHILE loop validation\*/

strcpy(action.test, action.dob);

action = CheckString(action);

while (len < 6 || len > 6 || action.results != 0)

{

printf("\nError: You need to enter in 6 digits for the DoB (dd/mm/yy)");

printf("\nEnter The Patient's Date of Birth (e.g. 010196): ");

scanf(" %s", &action.dob);

fflush(stdin);

len = strlen(action.dob);

strcpy(action.test, action.dob);

action = CheckString(action);

}

strcpy(action.encryptdata, action.dob);

strcpy(birth, action.dob);

action = encryption(action);

strcpy(action.dob, action.encryptdata);

printf("Enter The Patient's Height (In Centimetres): ");

scanf(" %s", &action.height);

fflush(stdin);

strcpy(action.test, action.height);

action = CheckString(action);

while (action.results != 0)

{

printf("\nError: Only integer values should be entered");

printf("\nEnter The Patient's Height (In Centimetres): ");

scanf(" %s", &action.height);

fflush(stdin);

strcpy(action.test, action.height);

action = CheckString(action);

}

strcpy(action.encryptdata, action.height);

action = encryption(action);

strcpy(action.height, action.encryptdata);

printf("Enter The Patient's Waist Measurement (In Centimetres): ");

scanf(" %s", &action.waist);

fflush(stdin);

strcpy(action.test, action.waist);

action = CheckString(action);

while (action.results != 0)

{

printf("\nError: Only integer values should be entered");

printf("Enter The Patient's Waist Measurement (In Centimetres): ");

scanf(" %s", action.waist);

fflush(stdin);

strcpy(action.test, action.waist);

action = CheckString(action);

}

strcpy(action.encryptdata, action.waist);

action = encryption(action);

strcpy(action.waist, action.encryptdata);

printf("Enter The Patient's Weight (In Kilograms): ");

scanf(" %s", &action.weight);

fflush(stdin);

strcpy(action.test, action.weight);

action = CheckString(action);

while (action.results != 0)

{

printf("\nError: Only integer values should be entered");

printf("Enter The Patient's Weight (In Kilograms): ");

scanf(" %s", action.weight);

fflush(stdin);

strcpy(action.test, action.weight);

action = CheckString(action);

}

strcpy(action.encryptdata, action.weight);

action = encryption(action);

strcpy(action.weight, action.encryptdata);

printf("Enter In Any Patient Comments: ");

scanf(" %[^\n]", &action.comment);

fflush(stdin);

strcpy(action.encryptdata, action.comment);

action = encryption(action);

strcpy(action.comment, action.encryptdata);

/\*The patient file names are automatically generated by the program

by adding their surname to their date of birth followed by the ".pat" extention\*/

strcpy(filename, name);

strcat(filename, birth);

strcat(filename, ".pat");

fp1 = fopen(filename, "w");

fprintf(fp1, "%s\n", action.firstname);

fprintf(fp1, "%s\n", action.surname);

fprintf(fp1, "%s\n", action.dob);

fprintf(fp1, "%s\n", action.height);

fprintf(fp1, "%s\n", action.waist);

fprintf(fp1, "%s\n", action.weight);

fprintf(fp1, "%s", action.comment);

fclose(fp1);

system("cls");

printf("%s has now been created\n\n", filename);

printf("Would You Like To Enter In Details For Another Patient?\n");

printf("1. Yes\n");

printf("2. No (Go Back To Menu)\n");

printf("Please select an option (1 or 2): ");

valid1 = scanf("%d", &option);

fflush(stdin);

while (option < 1 || option > 2 || valid1 != 1)

{

printf("Please select an option (1 or 2): ");

scanf("%d", &option);

fflush(stdin);

}

switch (option)

{

case 1:

action = EnterData(action);

break;

case 2:

action = Menu(action);

break;

}

return action;

}

/\*The CheckString function is used to check if a string contains

any integer or alpha characters which the function will then output a 1 or a 0

which is then used by the WHILE loops in the EnterData function for validation\*/

struct data CheckString(struct data action)

{

int checking;

int counting = 0;

int len;

int i;

action.results = 0;

len = strlen(action.test);

for (i = 0; i < len; i++)

{

/\*Checks every character in a string to see if there are any alpha characters\*/

checking = isalpha(action.test[i]);

if (checking != 0)

{

counting = counting + 1;

}

}

if (counting == len)

{

action.results = 1;

}

else

{

action.results = 0;

}

return action;

}

/\*The encryption function is used to encypt the "encryptdata" varible that is stored in

the global structure, whatever string is stored there when the encyption function is called

will then be encrypted and the encryped string then returned back to the same "encryptdata" varible in

the structure which can then be called into any function where necessary\*/

struct data encryption(struct data action)

{

int len = 0;

int i = 0;

int encrypting[255] = {0};

char encrypted[255] = {0};

len = strlen(action.encryptdata);

/\*The loops works by coverting the string character by character into an int type varible

from a char type varible, this will then display the ASCII code value of each character

which is then be increased by 4 this new ASCII value will then be converted back character by

character into a char type varible which will then store the newly encrypted characters.

Special charcters are left alone and so they are not encrypted\*/

for (i = 0 ; i < len ; i++)

{

encrypting[i] = action.encryptdata[i];

/\*IF statements are used for conditioning of the various ASCII values\*/

if ((encrypting[i] < 48) && (encrypting[i] > 31))

{

encrypted[i] = encrypting[i];

}

if ((encrypting[i] < 123) && (encrypting[i] > 96))

{

if (encrypting[i] < 119)

{

encrypted[i] = encrypting[i] + 4;

}

else

{

/\*To get the encryption to loop round the A-Z selection

when it reached a vaue that would take it out of the A-Z range such as w

it would then subtract instead of adding 4 which would take it back round

to the start of the alphabet again\*/

encrypted[i] = encrypting[i] - 22;

}

}

else

{

if ((encrypting[i] < 58) && (encrypting[i] > 47))

{

if (encrypting[i] < 54)

{

encrypted[i] = encrypting[i] + 4;

}

else

{

/\*To get the encryption to loop round the 0-9 selection

when it reached a vaue that would take it out of the 0-9 range such as 6

it would then subtract instead of adding 4 which would take it back round

to the start of the number range again\*/

encrypted[i] = encrypting[i] - 6;

}

}

else

{

if ((encrypting[i] < 91) && (encrypting[i] > 64))

{

if (encrypting[i] < 87)

{

encrypted[i] = encrypting[i] + 4;

}

else

{

encrypted[i] = encrypting[i] - 22;

}

}

}

}

}

strcpy(action.encryptdata, encrypted);

return action;

}

/\*The decryption function is used to decrypt the "decryptdata" varible that is stored in

the global structure, whatever string is stored there when the encyption function is called

will then be encrypted and the encryped string then returned back to the same "decryptdata" varible in

the structure which can then be called into any function where necessary\*/

struct data decryption(struct data action)

{

int len = 0;

int i = 0;

int encrypting[255] = {0};

char encrypted[255] = {0};

len = strlen(action.encryptdata);

/\*The loops works by coverting the string character by character into an int type varible

from a char type varible, this will then display the ASCII code value of each character

which is then be decrease by 4 this new ASCII value will then be converted back character by

character into a char type varible which will then store the newly encrypted characters.

Special charcters are left alone and so they are not encrypted\*/

for (i = 0 ; i < len ; i++)

{

encrypting[i] = action.encryptdata[i];

if ((encrypting[i] < 48) && (encrypting[i] > 31))

{

encrypted[i] = encrypting[i];

}

if ((encrypting[i] < 123) && (encrypting[i] > 96))

{

if (encrypting[i] >= 101)

{

encrypted[i] = encrypting[i] - 4;

}

else

{

encrypted[i] = encrypting[i] + 22;

}

}

else

{

if ((encrypting[i] < 58) && (encrypting[i] > 47))

{

if (encrypting[i] >= 52)

{

encrypted[i] = encrypting[i] - 4;

}

else

{

encrypted[i] = encrypting[i] + 6;

}

}

else

{

if ((encrypting[i] < 91) && (encrypting[i] > 64))

{

if (encrypting[i] >= 69)

{

encrypted[i] = encrypting[i] - 4;

}

else

{

encrypted[i] = encrypting[i] + 22;

}

}

}

}

}

strcpy(action.encryptdata, encrypted);

return action;

}

## Program 2 (Consultant)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ActionOnWeight2.c

Jonathan Somarib

A reading program for reading patient files

December 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <conio.h>

struct data

{

char encryptdata[255];

char firstname[255];

char surname[255];

char dob[255];

char height[255];

char waist[255];

char weight[255];

char comment[255];

};

struct data decryption(struct data action);

struct data GetPatientDetails(struct data action);

int main()

{

struct data action =

{

0

};

action = GetPatientDetails(action);

}

/\*The GetPatientDetails function opens and reads the chosen text file

decrypts the data and then displays it\*/

struct data GetPatientDetails(struct data action)

{

FILE \*fp;

int i;

char \*posi1;

char \*posi2;

char \*posi3;

char \*posi4;

char \*posi5;

char \*posi6;

char \*posi7;

char username[255];

char filename[255];

printf("Action on Weight - Read Patient File | Consultant\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

/\*Asks for the consultant to enter in the name of the file they wish to read\*/

printf("Please Enter In The Patient's File Name (Surname+Date of Birth): ");

scanf("%s", &username);

fflush(stdin);

/\*The program will then automatically add the ".pat" extention to the varible\*/

strcpy(filename, username);

strcat(filename, ".pat");

fp = fopen(filename, "r");

for (i = 0; i < 1; i++)

{

fgets(action.firstname, 20, fp);

/\*This searches the read string from the file for a new line character

and replaces it with a null character since fgets adds a trailing new line character

at the end of every string it reads and stores\*/

if ((posi1 = strchr(action.firstname, '\n')) != NULL)

{

\*posi1 = '\0';

};

/\*The lines are then decrypted as they are read and stored into arrays\*/

strcpy(action.encryptdata, action.firstname);

action = decryption(action);

strcpy(action.firstname, action.encryptdata);

fgets(action.surname, 20, fp);

if ((posi2 = strchr(action.surname, '\n')) != NULL)

{

\*posi2 = '\0';

};

strcpy(action.encryptdata, action.surname);

action = decryption(action);

strcpy(action.surname, action.encryptdata);

fgets(action.dob, 20, fp);

if ((posi3 = strchr(action.dob, '\n')) != NULL)

{

\*posi3 = '\0';

};

strcpy(action.encryptdata, action.dob);

action = decryption(action);

strcpy(action.dob, action.encryptdata);

fgets(action.height, 20, fp);

if ((posi4 = strchr(action.height, '\n')) != NULL)

{

\*posi4 = '\0';

};

strcpy(action.encryptdata, action.height);

action = decryption(action);

strcpy(action.height, action.encryptdata);

fgets(action.waist, 20, fp);

if ((posi5 = strchr(action.waist, '\n')) != NULL)

{

\*posi5 = '\0';

};

strcpy(action.encryptdata, action.waist);

action = decryption(action);

strcpy(action.waist, action.encryptdata);

fgets(action.weight, 20, fp);

if ((posi6 = strchr(action.weight, '\n')) != NULL)

{

\*posi6 = '\0';

};

strcpy(action.encryptdata, action.weight);

action = decryption(action);

strcpy(action.weight, action.encryptdata);

fgets(action.comment, 255, fp);

if ((posi7 = strchr(action.comment, '\n')) != NULL)

{

\*posi7 = '\0';

};

strcpy(action.encryptdata, action.comment);

action = decryption(action);

strcpy(action.comment, action.encryptdata);

}

fclose(fp);

/\*Once all the lines have been read an decrypted they are then displayed nicely for the

consultant to view\*/

printf("File Name: %s", filename);

getch();

system("cls");

printf("Action on Weight - Patient Details\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Name: %s %s\n", action.firstname, action.surname);

printf("Date of Birth: %c%c/%c%c/%c%c \n", action.dob[0], action.dob[1], action.dob[2], action.dob[3], action.dob[4], action.dob[5]);

printf("Height: %s centimetres\n", action.height);

printf("Waist: %s centimetres\n", action.waist);

printf("Weight: %s kilograms\n", action.weight);

printf("Comment: %s", action.comment);

printf("\n\nPress Enter To Select A New Patient File");

getch();

system("cls");

action = GetPatientDetails(action);

return action;

}

/\*The decryption function is used to decrypt the "decryptdata" varible that is stored in

the global structure, whatever string is stored there when the encyption function is called

will then be encrypted and the encryped string then returned back to the same "decryptdata" varible in

the structure which can then be called into any function where necessary\*/

struct data decryption(struct data action)

{

int len = 0;

int i = 0;

int encrypting[255] = {0};

char encrypted[255] = {0};

len = strlen(action.encryptdata);

/\*The loops works by coverting the string character by character into an int type varible

from a char type varible, this will then display the ASCII code value of each character

which is then be decrease by 4 this new ASCII value will then be converted back character by

character into a char type varible which will then store the newly encrypted characters.

Special charcters are left alone and so they are not encrypted\*/

for (i = 0 ; i < len ; i++)

{

encrypting[i] = action.encryptdata[i];

if ((encrypting[i] < 48) && (encrypting[i] > 31))

{

encrypted[i] = encrypting[i];

}

if ((encrypting[i] < 123) && (encrypting[i] > 96))

{

if (encrypting[i] >= 101)

{

encrypted[i] = encrypting[i] - 4;

}

else

{

encrypted[i] = encrypting[i] + 22;

}

}

else

{

if ((encrypting[i] < 58) && (encrypting[i] > 47))

{

if (encrypting[i] >= 52)

{

encrypted[i] = encrypting[i] - 4;

}

else

{

encrypted[i] = encrypting[i] + 6;

}

}

else

{

if ((encrypting[i] < 91) && (encrypting[i] > 64))

{

if (encrypting[i] >= 69)

{

encrypted[i] = encrypting[i] - 4;

}

else

{

encrypted[i] = encrypting[i] + 22;

}

}

}

}

}

strcpy(action.encryptdata, encrypted);

return action;

}