

**COMPUTER SCIENCE DEPARTMENT**

# Total Marks:\_\_\_\_\_\_\_\_\_

**Obtained Marks:**

Fundamentals of Programming

**Final Project**

# Submitted To: Mr. Sultan Ahmed

**Student Names: Khalid abbas , Furqan Maqbool**

# Reg. Numbers: 2312236 , 2312286



**COMPUTER SCIENCE DEPARTMENT**

**Personal Assistant**

**Introduction:**

This program is developed as part of our Fop project and aims to provide a collection of useful tools for various tasks. The program includes features such as a calculator, finance management, weather analysis, library management, note-taking, contact management. Each component serves a specific purpose and offers functionality to assist users in their daily activities.

**Features:**

* Calculator: Allows the user to perform basic arithmetic operations (addition, subtraction, multiplication, and division) on two numbers.
* Finance Manager: Helps manage personal finances by allowing the user to add income, subtract expenses, calculate Zakat (a form of alms-giving in Islam).
* Weather App: Collects temperature data for a specified number of days, calculates the average, maximum, and minimum temperatures, and displays the results.
* Book App: Uses a struct to store book details (title, author, and book ID) and provides functionality to display and enter book information.
* Note-Taking App: Allows the user to add and view notes. The notes are stored in a text file ("notes.txt").
* Contacts Manager: Uses a struct to store contact details (name, address, email, and phone number) and provides functionality to display and enter contact information.

**Objectives:**

* To provide an easy-to-use interface for users to perform calculations and manage their finances.
* To help users stay organized by providing a note-taking feature that allows them to store important information in one place.
* To offer personalized reminders and alerts to users based on their interactions with the program, helping them stay on top of their tasks and responsibilities.
* To accurately calculate taxes and zakat based on user input, providing valuable information to users to make informed financial decisions.
* To help users manage their contacts efficiently
* To aware users about weather, time, and other updates at the same place.

**What we used:**

* String
* Int
* Char
* float
* double
* bool
* functions
* struct
* if/else + switch
* arrays
* file handling
* Loops (while loop, for Loop)

**Source Code:**

#include <iostream>

#include <string>

#include <iomanip>

#include <fstream>

using namespace std;

double addNum(double a, double b)

{

return a + b;

}

double subNum(double a, double b)

{

return a - b;

}

double mulNum(double a, double b)

{

return a \* b;

}

double diviNum(double a, double b)

{

return a / b;

}

void calculator()

{

double op1, op2;

char option;

double result = 0; // Initialize result to avoid using an uninitialized value

cout << "What Todo\n (+,-,\*,/)" << endl;

cin >> option;

cout << "Enter Numbers" << endl;

cin >> op1 >> op2;

switch (option)

{

case '+':

result = addNum(op1, op2);

break;

case '-':

result = subNum(op1, op2);

break;

case '\*':

result = mulNum(op1, op2);

break;

case '/':

if (op2 != 0) // Check for division by zero

result = diviNum(op1, op2);

else

cout << "Cannot divide by zero" << endl;

break;

default:

cout << "Invalid operator" << endl;

}

cout << "Result = " << result << endl;

}

double calculateTotal(double salary, double amount)

{

return salary + amount;

}

double remainingBudget(double salary, double amount)

{

return salary - amount;

}

double zakatAmount(double salary)

{

double zakat = (2.5 / 100) \* salary;

return zakat;

}

void Finance()

{

double salary;

double total = 0;

int financeChoice;

bool opFinance = true;

cout << " You Chose Finance\n\n\n" << endl;

cout << "Enter your Salary" << endl;

cin >> salary;

while (opFinance)

{

cout << " Do You Want to: " << endl;

cout << " 1. Add Income +" << endl;

cout << " 2. Remove Expenses -" << endl;

cout << " 3. Zakat" << endl;

cout << " 0. To Exit Finances" << endl;

cout << " Enter Your Choices 0-3" << endl;

cin >> financeChoice;

cout << "\n" << endl;

if (financeChoice == 1)

{

double addFinance;

cout << "How much more money can you get this month?" << endl;

cin >> addFinance;

total = calculateTotal(salary, addFinance);

cout << "Your Total for this month is: " << total << " Whoo!\n\n\n\n" << endl;

}

else if (financeChoice == 2)

{

double Expenses;

cout << "Enter Total Expenses This Month\n";

cin >> Expenses;

total = remainingBudget(salary, Expenses);

cout << "Your Total Remaining Budget This Month Is: " << total << endl;

}

else if (financeChoice == 3)

{

double Zakat = zakatAmount(salary);

cout << "\nThe Amount Of Zakat You Have To Pay From Your Salary Is: " << Zakat << endl;

}

else if (financeChoice == 0)

{

opFinance = false;

}

else

{

cout << "Invalid Choice" << endl;

}

}

}

const int NUM\_DAYS = 7;

void getWeatherData(int temperatures[], int numDays)

{

cout << "Enter the temperatures for the next " << numDays << " days:" << endl;

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

{

cout << "Day " << (i + 1) << ": ";

cin >> temperatures[i];

}

}

double calculateAverage(int temperatures[], int numDays)

{

int sum = 0;

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

{

sum += temperatures[i];

}

return static\_cast<double>(sum) / numDays; // Use static\_cast for type casting

}

int findMaxTemperature(int temperatures[], int numDays)

{

int maxTemp = temperatures[0];

for (int i = 1; i < numDays; i++)

{

if (temperatures[i] > maxTemp)

{

maxTemp = temperatures[i];

}

}

return maxTemp;

}

int findMinTemperature(int temperatures[], int numDays)

{

int minTemp = temperatures[0];

for (int i = 1; i < numDays; i++)

{

if (temperatures[i] < minTemp)

{

minTemp = temperatures[i];

}

}

return minTemp;

}

void weatherApp()

{

int temperatures[NUM\_DAYS];

getWeatherData(temperatures, NUM\_DAYS);

double average = calculateAverage(temperatures, NUM\_DAYS);

int maxTemp = findMaxTemperature(temperatures, NUM\_DAYS);

int minTemp = findMinTemperature(temperatures, NUM\_DAYS);

cout << "Average temperature: " << average << endl;

cout << "Maximum temperature: " << maxTemp << endl;

cout << "Minimum temperature: " << minTemp << endl;

}

struct Books

{

char title[50];

char author[50];

int bookID;

};

void displayBooks(struct Books b1, struct Books b2)

{

cout << "Book 1: " << endl;

cout << "Title: " << b1.title << endl;

cout << "Author: " << b1.author << endl;

cout << "Book ID: " << b1.bookID << endl;

cout << endl;

cout << "Book 2: " << endl;

cout << "Title: " << b2.title << endl;

cout << "Author: " << b2.author << endl;

cout << "Book ID: " << b2.bookID << endl;

cout << endl;

}

void bookApp()

{

struct Books b1, b2;

// Book 1 specifications

cout << "Enter the book name: ";

cin.ignore(); // Ignore newline character from previous input

cin.getline(b1.title, 50);

cout << "Enter the author: ";

cin.getline(b1.author, 50);

cout << "Enter the book id: ";

cin >> b1.bookID;

// Book 2 specifications

cout << "Enter the book name: ";

cin.ignore(); // Ignore newline character from previous input

cin.getline(b2.title, 50);

cout << "Enter the author name: ";

cin.getline(b2.author, 50);

cout << "Enter the book id: ";

cin >> b2.bookID;

// Display Book 1 and Book 2

displayBooks(b1, b2);

}

void addNote()

{

ofstream file("notes.txt", ios::app);

if (!file)

{

cout << "Unable to open file for writing." << endl;

return;

}

string note;

cout << "Enter your note (press Enter to finish): ";

cin.ignore(); // Ignore newline character from previous input

getline(cin, note);

file << note << endl;

file.close();

cout << "Note added successfully." << endl;

cout << endl;

}

void viewNotes()

{

ifstream file("notes.txt");

if (!file)

{

cout << "No notes found." << endl;

return;

}

string line;

int noteCount = 0;

while (getline(file, line))

{

cout << line << endl;

noteCount++;

}

file.close();

cout << "Total notes: " << noteCount << endl;

}

void noteApp()

{

int choice;

while (true)

{

cout << " \*NOTE TAKING\* " << endl;

cout << "1. Add Note" << endl;

cout << "2. View Notes" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice)

{

case 1:

cin.ignore(); // Ignore newline character from previous input

addNote();

break;

case 2:

viewNotes();

break;

case 3:

cout << "Exiting the program." << endl;

return; // Exit the function instead of setting a flag

default:

cout << "Invalid choice. Try again." << endl;

break;

}

cout << endl;

}

}

struct Contacts

{

char address[50];

char email[50];

char name[50];

int phoneNumber;

};

void displayContacts(struct Contacts contact1, struct Contacts contact2)

{

cout << "Contact 1: " << endl;

cout << "Name: " << contact1.name << endl;

cout << "Address: " << contact1.address << endl;

cout << "Phone Number: " << contact1.phoneNumber << endl;

cout << "Email: " << contact1.email << endl;

cout << endl;

cout << "Contact 2: " << endl;

cout << "Name: " << contact2.name << endl;

cout << "Address: " << contact2.address << endl;

cout << "Phone Number: " << contact2.phoneNumber << endl;

cout << "Email: " << contact2.email << endl;

cout << endl;

}

void getContacts(struct Contacts &contact)

{

cout << "Enter the name: ";

cin.ignore(); // Ignore newline character from previous input

cin.getline(contact.name, 50);

cout << "Enter the address: ";

cin.getline(contact.address, 50);

cout << "Enter the phone number: ";

cin >> contact.phoneNumber;

cout << "Enter the email: ";

cin.ignore(); // Ignore newline character from previous input

cin.getline(contact.email, 50);

cout << endl;

}

void contactApp()

{

struct Contacts contact1, contact2;

// Getting contact details

cout << "Enter details for Contact 1:" << endl;

getContacts(contact1);

cout << "Enter details for Contact 2:" << endl;

getContacts(contact2);

// Display contact1 and contact2

displayContacts(contact1, contact2);

}

int main()

{

string name;

int choice;

cout << "Enter Name Please" << endl;

cin >> name;

while (true)

{

cout << "Hello " << name << " How Can I Help You Today" << endl;

cout << "\n\n\t\t\t1. Calculator\t\t\t\t\t\t2. Finance\n\n\t\t\t3. Weather\t\t\t\t\t\t4. Book Keeping\n\n\t\t\t5. Note App\t\t\t\t\t\t6. Contact App\n\n\t\t\t0. Exit\t\t\t\t\t\t";

cin >> choice;

switch (choice)

{

case 1:

calculator();

break;

case 2:

Finance();

break;

case 3:

weatherApp();

break;

case 4:

bookApp();

break;

case 5:

noteApp();

break;

case 6:

contactApp();

break;

case 0:

cout << "Exiting the program." << endl;

return 0; // Exit the program

default:

cout << "Invalid choice. Try again." << endl;

break;

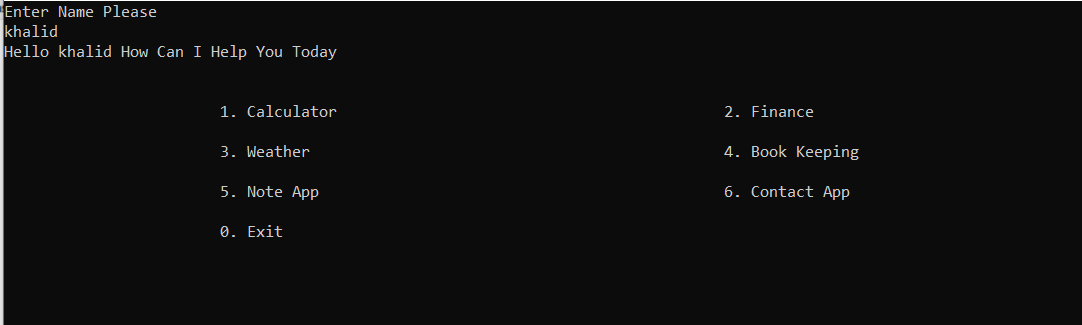
}

}

return 0;

}

**Welcome Screen:**



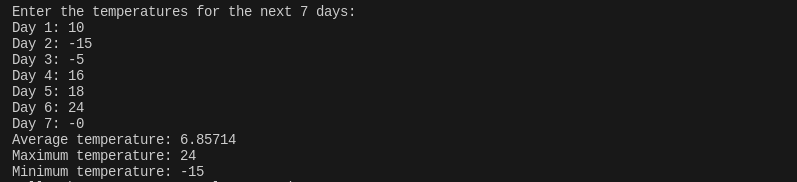
1. **Calculator:**

****

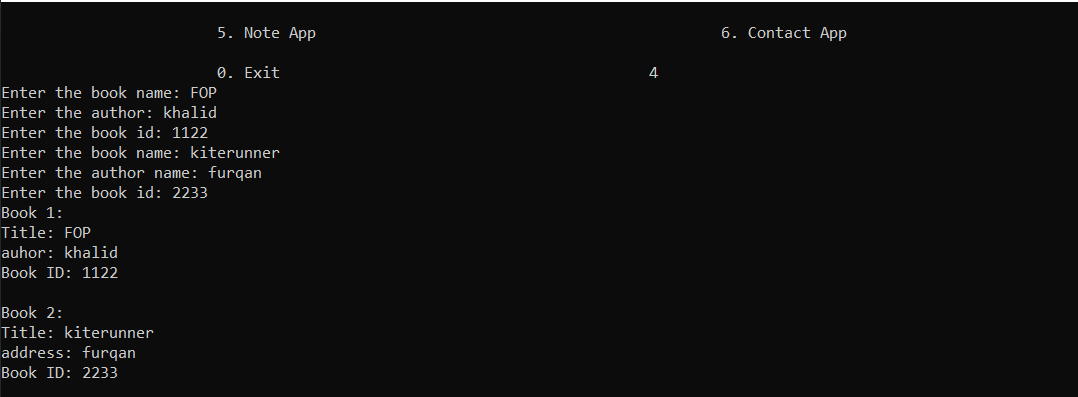
1. **Finance(Zakat Finder/Salary Calculator)**

****

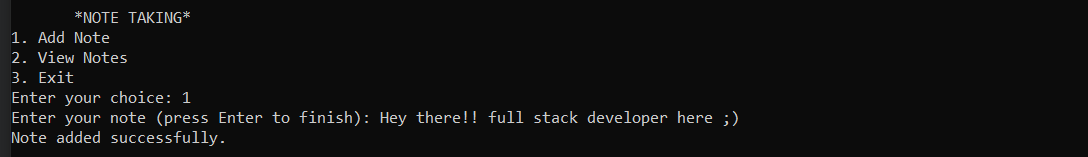
1. **Weather**

****

1. **Bookkeeping:**



1. **Note App:**



1. **Contact App:**



**Conclusion:**

Personal Assistant is a multifunctional utility suite, encompassing a range of applications from basic arithmetic operations to financial management, weather data analysis, bookkeeping, note-taking, and contact management. The modular design contributes to code clarity and maintainability, allowing users to interact with distinct features seamlessly.

While the program exhibits robust functionality, there are areas for improvement. Enhancements in error handling, such as more user-friendly messages for invalid inputs, could enhance the overall user experience. Consistent use of camelCase for function names and thorough validation of user inputs are additional aspects to consider for code refinement.

The program's versatility, coupled with its ability to handle various tasks within a single application, demonstrates thoughtful design. With continued attention to user interaction, error management, and code organization, this C++ program has the potential to evolve into a well-rounded and user-friendly utility suite.