

Car Insurance Advisor

Dev Fernandez, FY B. Sc.IT, A014, 45207220002

Aryan Gawde, FY B. Sc.IT, A018, 45207220012

Abstract/Synopsis:

This project is an implementation of a Car Insurance Calculator. It prompts the user to input details such as age, car value, number of accidents, number of tickets and gender for a given number of users, and calculates the insurance cost for each user based on the given input. The program is organized using an object-oriented approach, where a class named CarInsurance is defined to encapsulate the input details and insurance cost calculation logic. The constructor of this class takes in the input details, and the calculateInsuranceCost() method calculates the insurance cost based on these input details.

A vector of CarInsurance objects is used to store the insurance details of each user.

The program first prompts the user to input the number of users and then, for each user, the user inputs the necessary details, and a CarInsurance object is created using the input details and added to the vector. The program then calculates the insurance cost for each user using the

calculateInsuranceCost() method and outputs the result.

Keywords:

class: used to define a new class type

private: used to specify the access level of class members as private, meaning they can only be accessed within the class itself

public: used to specify the access level of class members as public, meaning they can be accessed by any code that uses the class

namespace: used to define a namespace, which is a way of grouping related code together to avoid naming conflicts

using: used to bring a namespace or specific name from a namespace into the current scope

int: used to declare an integer variable

float: used to declare a floating-point variable

string: used to declare a string variable

vector: used to declare a vector container that can hold multiple objects of the same type

cin: used to read input from the user via the console

cout: used to output data to the console

if: used to execute code if a certain condition is true

else: used to execute code if a certain condition is false

return: used to return a value from a function

main(): the entry point of the program, where execution begins

new: used to dynamically allocate memory for a new object

this: used to refer to the current object being operated on in a member function

push_back(): used to add an element to the end of a vector container.

Objective and Scope:

Objective: To calculate the car insurance cost for multiple users based on various factors such as age, car value, number of accidents, number of tickets, and gender. The program prompts the user to input these factors for each user and creates a CarInsurance object for each user. The

program then calculates the insurance cost for each user using the formula mentioned in the calculateInsuranceCost() method of the CarInsurance class. Finally, the program outputs the insurance cost for each user.

Scope: It's limited to calculating the car insurance cost based on the input factors and does not cover other aspects such as generating insurance policies, managing user accounts, or handling payment processing. The program is designed to be a simple prototype to calculate car insurance cost and can be further extended to meet specific requirements.

Detailed working of the project:

The program is designed to calculate the cost of car insurance for a given set of users. It uses a class called CarInsurance to represent each user and calculate their insurance cost based on various factors, such as age, car value, number of accidents, number of tickets, and gender.

The program begins by declaring the necessary header files and defining the CarInsurance class with its constructor and insurance cost calculation function. The

class contains private member variables to store the user's age, car value, number of accidents, number of tickets, gender, and insurance cost.

In the `main()` function, the program asks the user to enter the number of users for which they want to calculate the insurance cost. It then uses a for loop to iterate through each user, asking them to enter their age, car value, number of accidents, number of tickets, and gender. For each user, the program creates a `CarInsurance` object and adds it to a vector called `users`.

Finally, the program uses another for loop to iterate through each user in the `users` vector and calculates their insurance cost using the `calculateInsuranceCost()` function. It then outputs the insurance cost for each user to the console.

Here is a brief summary of the **program's flow**:

- i. Declare necessary header files and define `CarInsurance` class with its constructor and insurance cost calculation function.
- ii. Declare variables to store user input, including `numUsers`, `age`, `carValue`, `numAccidents`, `numTickets`, and `gender`.
- iii. Create a vector called `users` to store each user's `CarInsurance` object.

- iv. Ask the user to enter the number of users for which they want to calculate the insurance cost.
- v. Use a for loop to iterate through each user, asking them to enter their age, car value, number of accidents, number of tickets, and gender. For each user, create a `CarInsurance` object and add it to the `users` vector.
- vi. Use another for loop to iterate through each user in the `users` vector and calculate their insurance cost using the `calculateInsuranceCost()` function.
- vii. Output the insurance cost for each user to the console.

Use and Purpose:

Purpose: To calculate the insurance cost for a group of users based on their age, car value, number of accidents, number of tickets, and gender. The program takes user input for each user and creates a `CarInsurance` object for each user using the input values. Then, the program calculates the insurance cost for each user using the `calculateInsuranceCost()` function of the `CarInsurance` class and outputs it to the user.

Use: It uses a vector of `CarInsurance` objects to store the `CarInsurance` objects

for each user. The user input values are passed as parameters to the CarInsurance constructor to create a CarInsurance object for each user. The calculateInsuranceCost() function is used to calculate the insurance cost for each user based on their input values.

The program uses if statements to check the user's age, number of accidents, number of tickets, and gender to calculate the insurance cost. If the user is under 25 years old, the insurance cost is calculated as 10% of the car value, otherwise, it is calculated as 7% of the car value. If the user has had any accidents or tickets, the insurance cost is increased by a certain percentage based on the number of accidents or tickets. Finally, if the user is male, the insurance cost is increased by 5% of the insurance cost.

Merits and Demerits :

Merits:

- i. It allows for user input to determine the insurance cost for each user based on their age, car value, number of accidents, number of tickets, and gender.
- ii. It uses a vector to store multiple instances of the CarInsurance class,

allowing for calculation of insurance cost for multiple users.

- iii. The CarInsurance class encapsulates all the necessary data and methods required for insurance calculation, providing a clean and organized code structure.
- iv. It includes input validation to ensure that the user inputs the correct data type and format for each variable.
- v. It calculates the insurance cost based on a set of predefined rules and criteria, making it easy to understand and modify if needed.

Demerits:

- i. It does not include any error handling mechanisms for unexpected errors or exceptions that may occur during runtime.
- ii. It assumes that the input provided by the user is always correct and does not include any additional error-checking or validation beyond data type and format checking.
- iii. It calculates the insurance cost using a fixed set of rules and criteria, which may not accurately reflect the true cost of insurance for each user.

- iv. It does not include any options for modifying or updating user information after it has been entered, requiring the user to re-enter all information if a mistake is made.

Future Enhancements:

Add more factors for determining

insurance cost: Currently, the program only takes into account age, car value, number of accidents, number of tickets, and gender. However, there are many other factors that can affect insurance cost, such as driving experience, location, type of car, and more. Adding more factors could make the program more accurate and useful.

Improve the accuracy of the

calculations: The current calculations for insurance cost are based on simple formulas, but insurance companies use more complex algorithms to determine rates. Implementing more accurate calculations could improve the reliability of the program's results.

Add a GUI: Instead of using the console for user input and output, a graphical user interface (GUI) could be added to make the program more user-friendly.

Store user data: Currently, the program only calculates insurance cost for each user once and then discards the data. It would be useful to store the data for each user in a database so that the program can be used to track insurance costs over time.

Expand to other types of insurance:

While this program is focused on car insurance, similar programs could be developed for other types of insurance, such as health insurance or home insurance. This would require modifying the calculations and adding new input fields, but the basic structure of the program could remain the same.

References:

Books:

OOP with C++ by E Balagurusamy

Cash After A Crash by Shawn DeVries

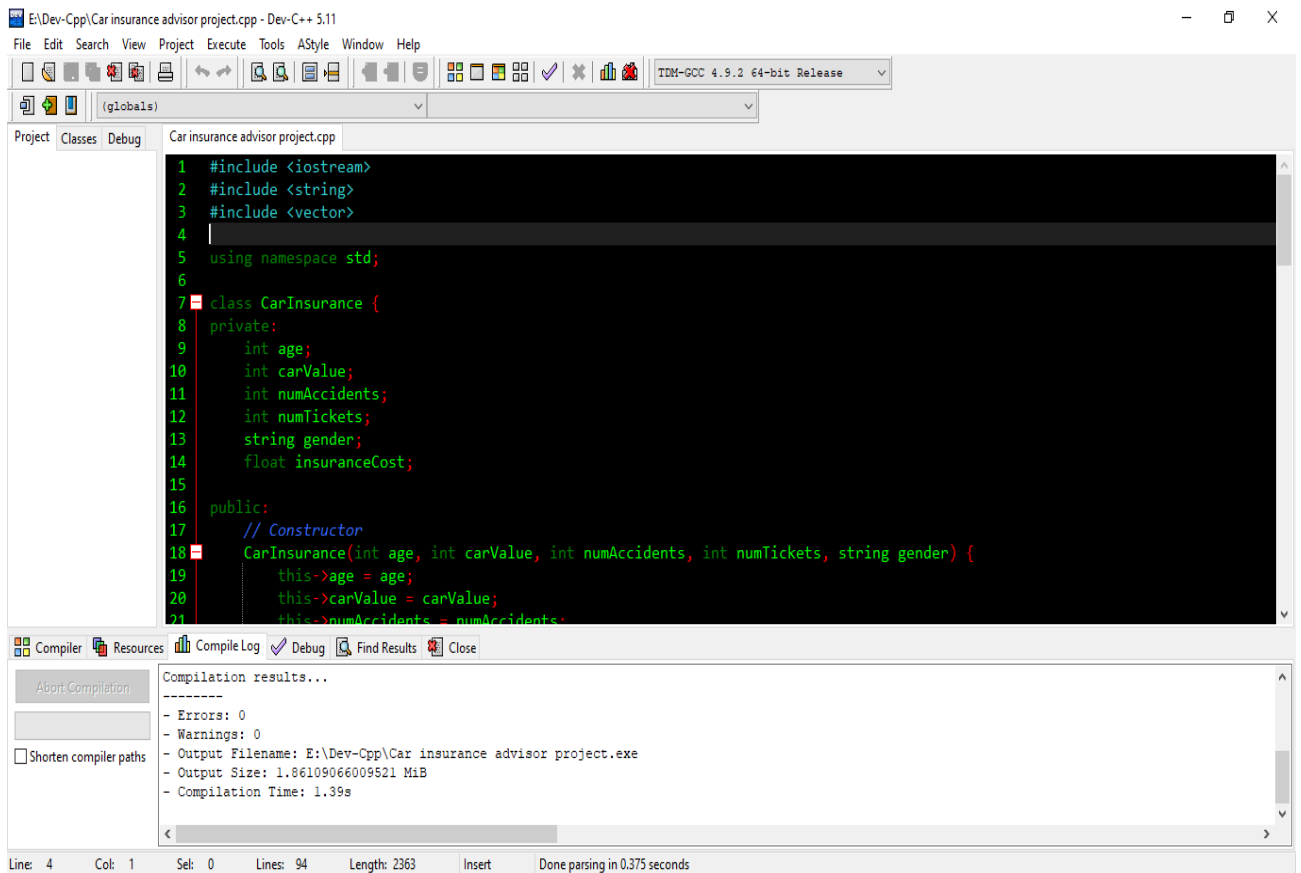
Web references:

YouTube: <https://www.youtube.com/>

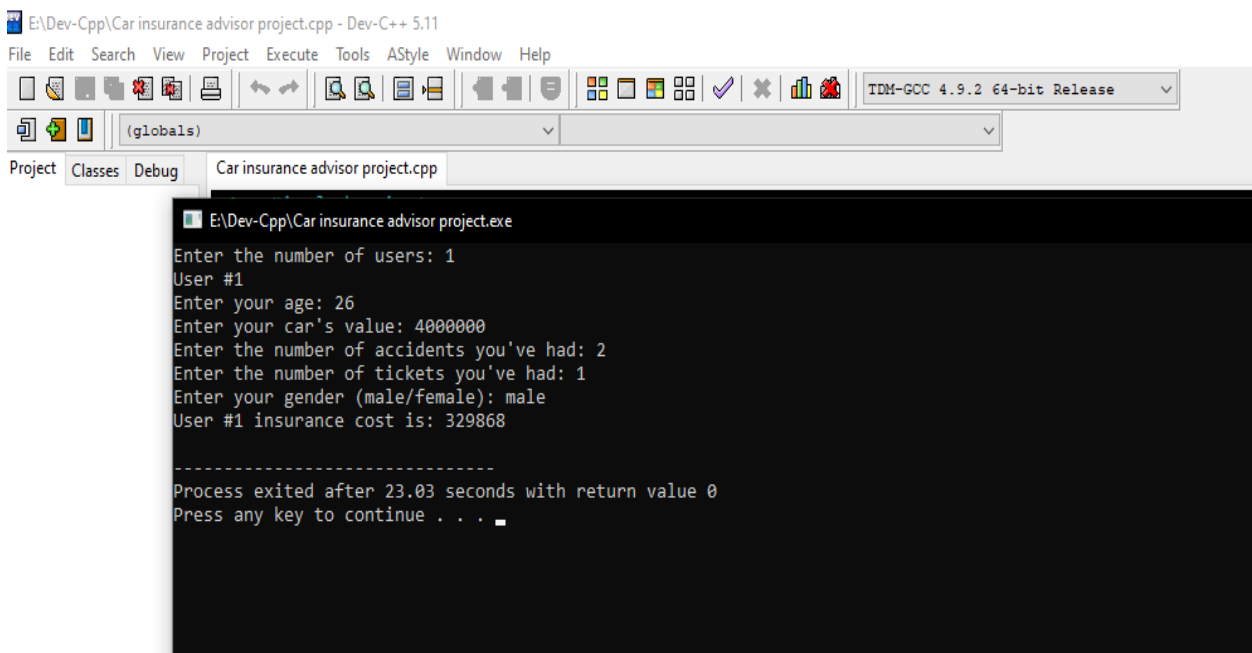
Wikipedia: <https://www.wikipedia.org/>

Screen Shots:

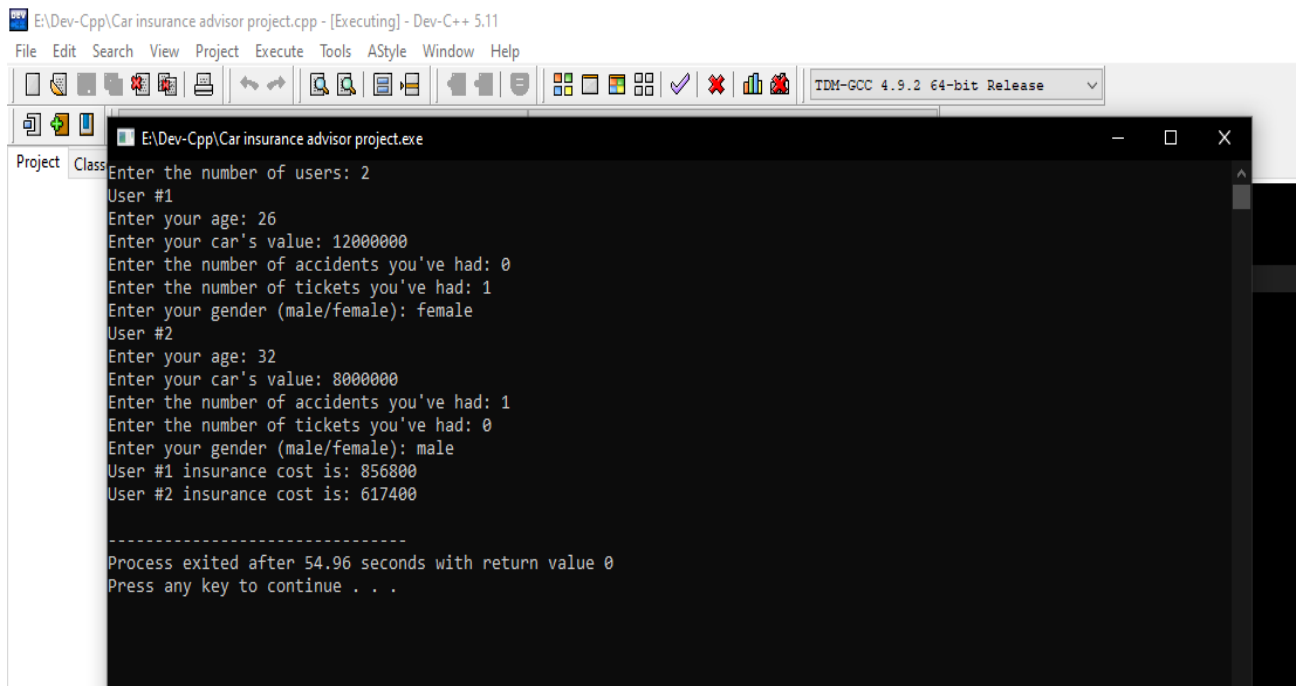
This program has no errors after being compiled:



Output for 1 user:



Output for 2 users:

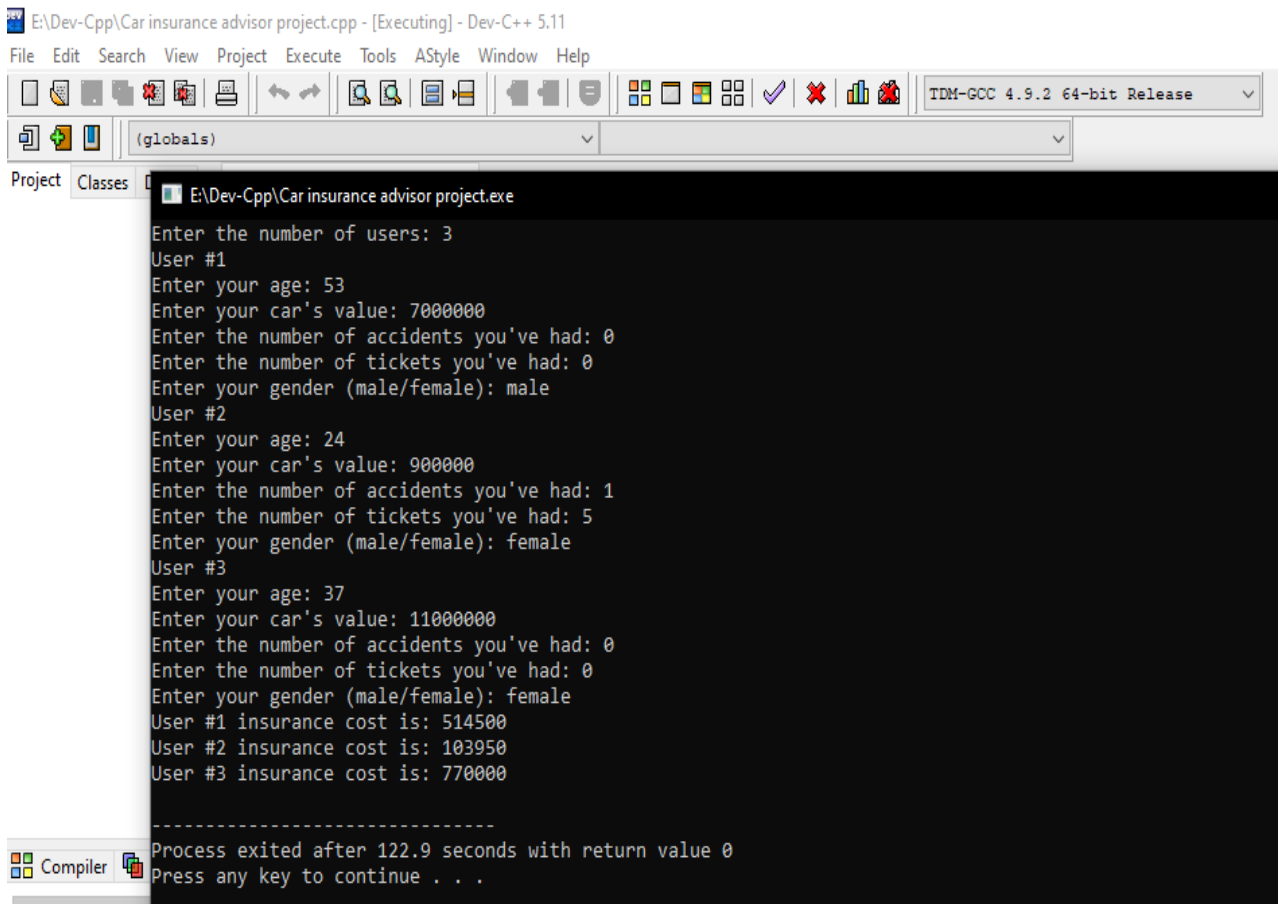


The screenshot shows the Dev-C++ IDE with the file 'E:\Dev-Cpp\Car insurance advisor project.cpp' open and executed. The console window displays the following text:

```
E:\Dev-Cpp\Car insurance advisor project.exe
Enter the number of users: 2
User #1
Enter your age: 26
Enter your car's value: 12000000
Enter the number of accidents you've had: 0
Enter the number of tickets you've had: 1
Enter your gender (male/female): female
User #2
Enter your age: 32
Enter your car's value: 8000000
Enter the number of accidents you've had: 1
Enter the number of tickets you've had: 0
Enter your gender (male/female): male
User #1 insurance cost is: 856800
User #2 insurance cost is: 617400

-----
Process exited after 54.96 seconds with return value 0
Press any key to continue . . .
```

Output for 3 users:



The screenshot shows the Dev-C++ IDE with the file 'E:\Dev-Cpp\Car insurance advisor project.cpp' open and executed. The console window displays the following text:

```
E:\Dev-Cpp\Car insurance advisor project.exe
Enter the number of users: 3
User #1
Enter your age: 53
Enter your car's value: 7000000
Enter the number of accidents you've had: 0
Enter the number of tickets you've had: 0
Enter your gender (male/female): male
User #2
Enter your age: 24
Enter your car's value: 900000
Enter the number of accidents you've had: 1
Enter the number of tickets you've had: 5
Enter your gender (male/female): female
User #3
Enter your age: 37
Enter your car's value: 11000000
Enter the number of accidents you've had: 0
Enter the number of tickets you've had: 0
Enter your gender (male/female): female
User #1 insurance cost is: 514500
User #2 insurance cost is: 103950
User #3 insurance cost is: 770000

-----
Process exited after 122.9 seconds with return value 0
Press any key to continue . . .
```

Source code:

```
#include <iostream>

#include <string>

#include <vector>

using namespace std;

class CarInsurance {

private:

    int age;

    int carValue;

    int numAccidents;

    int numTickets;

    string gender;

    float insuranceCost;

public:

    // Constructor

    CarInsurance(int age, int carValue, int numAccidents, int numTickets, string gender) {

        this->age = age;

        this->carValue = carValue;

        this->numAccidents = numAccidents;

        this->numTickets = numTickets;

        this->gender = gender;

    }

    // Calculate insurance cost

    float calculateInsuranceCost() {
```



```
    if (age < 25) {  
        insuranceCost = carValue * 0.10;  
    } else {  
        insuranceCost = carValue * 0.07;  
    }  
  
    if (numAccidents > 0) {  
        insuranceCost += (numAccidents * 0.05) * insuranceCost;  
    }  
  
    if (numTickets > 0) {  
        insuranceCost += (numTickets * 0.02) * insuranceCost;  
    }  
  
    if (gender == "male") {  
        insuranceCost += 0.05 * insuranceCost;  
    }  
  
    return insuranceCost;  
}  
};  
  
int main() {  
    int numUsers;  
  
    int age;  
  
    int carValue;  
  
    int numAccidents;  
  
    int numTickets;  
  
    string gender;
```

```

vector<CarInsurance> users;

// Get number of users

cout << "Enter the number of users: ";

cin >> numUsers;

// Get user input for each user

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

    cout << "User #" << i+1 << endl;

    cout << "Enter your age: ";

    cin >> age;

    cout << "Enter your car's value: ";

    cin >> carValue;

    cout << "Enter the number of accidents you've had: ";

    cin >> numAccidents;

    cout << "Enter the number of tickets you've had: ";

    cin >> numTickets;

    cout << "Enter your gender (male/female): ";

    cin >> gender;

    // Create a CarInsurance object and add it to the vector

    CarInsurance insurance(age, carValue, numAccidents, numTickets, gender);

    users.push_back(insurance);

}

// Calculate insurance cost for each user and output it to the user

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

    cout << "User #" << i+1 << " insurance cost is: " << users[i].calculateInsuranceCost()
<< endl; } return 0; }

```

