Documentation for Interest program

This project is created for the purpose of re-factoring the interest project int he previous semester by separating the program's process using functions and re-factoring that code again by implementing as user defined data type(in our case using **struct**).

The reason behind the re-factoring of the code is to make the code more readable and to remove redundancy form our previous programs by using modular programming and hopefully making the program less complicated than the previous one.

Modular programming is an approach to programs that are written in multiple functions, and these functions do a specific task in a program. This makes it easier to read and understand the code. Besides it uses the technique of breaking down a problem into small problems, which is the core principle of software engineering. We tried to embrace our code with functions that will do some tasks. Therefore, we're going through our code and try to describe all about the code we wrote.

Basic Components of The Program

As usual we started writing our program by including the **iostream** pre-processor which enables the flow of data in the program as an input and output. Besides we inserted **using namespace std** in our program which enables us to access some standard identifiers like **cin, cout** and **endl.** Other main component of the main function, in which it display the program.

Used Functions

In this project we used three different functions:

- An **inpuT** function to input interest rate and the amount of loan.
- A **dataCalc** function to calculate the the payment, interest, reaming loan and rate of payment.
- Finally an **outpuT** function, as it's name suggests it prints out the table for our program.

The working's of the code

After including header file and using the standard name space we started with declaring and initializing global variables, secondly function declaration or writing the function prototype of the function. This is good practice of writing a function code, besides we will call some of the functions in another function so this will avoid compilation error. Here is how we declared our function in our program:

```
float inteRate,overInterst=0,paymenT,interseT,loaN,originaLoan,annualRate;
int i=0;

void inpuT();
void dataCalc();
void outpuT();
```

The three function that we used are void function, meaning they don't have a return type. The functions are also parameter less as the use case of global variables to minimize the usage of passing by value(it create an issue where there are unnecessary amount of variables declared), and passing by reference was deemed not needed as the program didn't need to edit the value of the variables.

The **inpuT** function is a function that is supposed to ask for input from the used and store them to the global variable's. The function includes the storing of the loan in the variable **loaN** and the storing of the interest rate in the **inteRate** variable.

The initialization of the function looks like this:

```
void inpuT()
{
    cout<<"Enter the the original loan:"<<endl<<">";
    cin>>loaN;
    originaLoan=loaN;
    cout<<"Enter the interst rate of the loan(the rate should be in decimal):"<<endl<<">";
    cin>>inteRate;
    system("clear");
    cout<<setw(20)<<"Loan(in Birr)"<<setw(20)<<"Payment(in Birr)"<<setw(20)<<"Interst(in Birr)"<<setw(20)<<"Annual Rate(in %)"<<endl<<endl;
}
```

The second function is the **dataCalc** function, this function is used to do the mathematical intensive part of the program that includes:

- Calculating the interest,
- Calculating the payment,

- Calculating the annual rate,
- The payment over the interest,
- Finally the new(remaining) loan.

After calculating the function stores the data to there variable; interest in **interseT**, payment in **paymenT**, annual rate in **annualRate**, the payment over the interest in **overInterst**, the remaining loan in **loaN**.

The initialization of the function looks like this:

```
void dataCalc()
{
    paymenT=originaLoan/20;
    interseT=(inteRate*loaN)/12;
    annualRate=(loaN/originaLoan)*100;
    overInterst=paymenT-interseT;
    loaN-=overInterst;
}
```

The last function is the **outpuT** function that handles the presentation of the data. After some discussion we understood that there was no reason to delicately store the data of each cycle of the program so we opted to displaying the data every time one cycle of the program ends and calling again the **dataCalc** function until the loan is below zero.

We used a recursive function in the program to handle the execution of the function, and using if, else if and else statements for conditions of execution. The conditions being:

- If the loan is greater than zero and less than the previous loan(original loan), print out the loan, payment, interest and annual rate then call the **dataCalac** function and call it's self.
- If the loan is greater than the previous loan(original loan) then print the first line of the data and exit printing the "loan can't be payed". This case allows to avoid cases where the loan can't be payed and the program runs infinitely.
- If the loan in less than zero it prints out a summary of the program and exits.
    - The summary includes the the rate of interest and the cycles needed for the loan to be paid.

The initialization of the function looks like this:

```
void outpuT()
{
    if(loaN<originaLoan&&loaN>=0)
    {
        i++;
        cout<<setw(20)<<loaN<<setw(20)<<paymenT<<setw(20)<<interseT<<setw(20)<<annualRate<<"%"<<endl;
        dataCalc();
        outpuT();
    }
    else if(loaN>=originaLoan)
    {
        cout<<setw(20)<<loaN<<setw(20)<<paymenT<<setw(20)<<interseT<<setw(20)<<annualRate<<"%"<<endl;
        cout<<"The loan can't be payed with this interset rate."<<endl;
    }
    else
    {
        cout<<endl<<"Summary:"<<endl<<"The interst rate is: "<<inteRate*100<<"%"
            <<endl<<"Loan payment due(in years): "<<i;
    }
}
```

Final output

Case the interest rate is 0.2 and the loan is 4000:

```
      Loan(in Birr)    Payment(in Birr)    Interst(in Birr)   Annual Rate(in %)

            3866.67                 200             66.6667                100%
            3731.11                 200             64.4445             96.6667%
             3593.3                 200             62.1852             93.2778%
            3453.18                 200             59.8883             89.8324%
            3310.74                 200             57.5531             86.3296%
            3165.92                 200              55.179             82.7684%
            3018.68                 200             52.7653             79.1479%
            2868.99                 200             50.3114             75.4671%
            2716.81                 200             47.8166             71.7248%
            2562.09                 200             45.2802             67.9202%
            2404.79                 200             42.7015             64.0523%
            2244.87                 200             40.0799             60.1198%
            2082.29                 200             37.4145             56.1218%
            1916.99                 200             34.7048             52.0572%
            1748.94                 200             31.9499             47.9248%
            1578.09                 200              29.149             43.7235%
            1404.39                 200             26.3015             39.4522%
             1227.8                 200             23.4065             35.1098%
            1048.26                 200             20.4633             30.6949%
            865.732                 200              17.471             26.2065%
            680.161                 200             14.4289             21.6433%
            491.497                 200              11.336              17.004%
            299.689                 200             8.19162             12.2874%
            104.683                 200             4.99481             7.49222%

Summary:
The interst rate is: 20%
Loan payment due(in years): 24%
```

Version2 of the re-factoring

This part includes the usage of a user defined data type(**struct** in our case) to store different types of data type globally.

In this version of the program the only significant change from the previous implementation is the declaration of a **struct datA** and initialization of it using a **tyP** variable. The rest of the program includes the rewriting of the data's to complement the usage of the data type **datA**.

The data type **datA** included two data types:

- Float
- Int

The **struct datA** looks like this:

```
struct datA
{
    float inteRate,overInterst=0,paymenT,interseT,loaN,originaLoan,annualRate;
    int i=0;
};
```

The new implementation of the code looks like this:

For the **inpuT** function:

```
void inpuT()
{
    cout<<"Enter the the original loan:"<<endl<<">";
    cin>>tyP.loaN;
    tyP.originaLoan=tyP.loaN;
    cout<<"Enter the interst rate of the loan(the rate should be in decimal):"<<endl<<">";
    cin>>tyP.inteRate;
    system("clear");
    cout<<setw(20)<<"Loan(in Birr)"<<setw(20)<<"Payment(in Birr)"<<setw(20)<<"Interst(in Birr)"<<setw(20)<<"Annual Rate(in %)"<<endl<<endl;
}
```

For the **dataCalc** function:

```
void dataCalc()
{
    tyP.paymenT=tyP.originaLoan/20;
    tyP.interseT=(tyP.inteRate*tyP.loaN)/12;
    tyP.annualRate=(tyP.loaN/tyP.originaLoan)*100;
    tyP.overInterst=tyP.paymenT-tyP.interseT;
    tyP.loaN-=tyP.overInterst;
}
```

For the **outPut** function;

```
void outpuT()
{
    if(tyP.loaN<tyP.originaLoan&&tyP.loaN>=0)
    {
        tyP.i++;
        cout<<setw(20)<<tyP.loaN<<setw(20)<<tyP.paymenT<<setw(20)<<tyP.interseT<<setw(20)<<tyP.annualRate<<"%"<<endl;
        dataCalc();
        outpuT();
    }
    else if(tyP.loaN>=tyP.originaLoan)
    {
        cout<<setw(20)<<tyP.loaN<<setw(20)<<tyP.paymenT<<setw(20)<<tyP.interseT<<setw(20)<<tyP.annualRate<<"%"<<endl;
        cout<<"The loan can't be payed with this interset rate."<<endl;
    }
    else
    {
        cout<<endl<<"Summary:"<<endl<<"The interst rate is: "<<tyP.inteRate*100<<"%"
            <<endl<<"Loan payment due(in years): "<<tyP.i;
    }
}
```