

Project Report of CSE 8351 Computer Arithmetic

Error Estimation of High Precision Reciprocal of Decimal Numbers Obtained from a Prescaled LUT

Zizhen Chen

Advisor: Professor Matula

SMU ID: 37366827

Email: dragonz@live.cn

1. Project Purpose

We investigate the utilization of additive prescaling for reducing the input range for determining a higher precision lookup table (LUT) value employing moderate table size. This method is specifically used on application of division.

This report is for the project of estimating errors between high precision reciprocal of decimal numbers obtained from a prescaled LUT and its real value.

I use C++ Language to build a program via Visual Studio 2010 Compiler on Windows 7 platform to accomplish this project.

2. Estimation sum results via calculus

The format of the 9,000,000 numbers is formulated below.

$$y = d_0.d_1d_2d_3d_4d_5d_6 \quad (1 \leq d_0 \leq 9).$$

So the range is from 1 to 9.999999.

Because we want to calculate sum of the reciprocals of such 9,000,000 numbers, we can use definite integral method to estimate the value of sum like below.

$$Sum = \sum_{1}^{9.999999} \frac{1}{y}$$

Since from the concept of definite integral $\frac{1}{y}$ from 1 to 9.999999, we can get

$$\int_1^{9.999999} \frac{1}{y} dy = \lim_{\Delta y \rightarrow 0} \sum_{i=1}^{9.999999} \frac{1}{y_i} \times \Delta y = \sum_{i=1}^{9.999999} \frac{1}{y_i} \times \Delta y + o(y)$$

Here, $o(y)$ is Higher Order Infinitesimal of y , so $o(y)$ is a really tiny value which we can omit.

$$\text{Then, } \sum_{i=1}^{9.999999} \frac{1}{y_i} \times \Delta y + o(y) \approx \sum_{i=1}^{9.999999} \frac{1}{y_i} \times \Delta y$$

Therefore, we can get

$$\int_1^{9.999999} \frac{1}{y} dy \approx \sum_{i=1}^{9.999999} \frac{1}{y_i} \times \Delta y$$

$$\text{Since } \int_1^{9.999999} \frac{1}{y} dy \\ = \ln(9.999999) - \ln 1 = \ln(9.999999) \approx 2.302584993,$$

$$\sum_{i=1}^{9.999999} \frac{1}{y_i} \times \Delta y \approx 2.302584993$$

Here, $\Delta y = 0.0000001$ (Interval between two neighboring y_i)

$$\text{Therefore, } \text{Sum} = \sum_{i=1}^{9.999999} \frac{1}{y_i} \approx \frac{2.302584993}{\Delta y} = \frac{2.302584993}{0.0000001} = 2302584.993$$

In conclusion, the sum of reciprocals of such 9 million numbers is estimated as 2302584.993.

3. Project Process with Specification of Programming code

The whole program is going to compute 9,000,000 decimal 32-bit format normalized values. The format of the 9,000,000 numbers is formulated below.

$$y = d_0.d_1d_2d_3d_4d_5d_6 \quad (1 \leq d_0 \leq 9).$$

Related code of comparing these 9,000,000 decimal numbers:

```
struct
{
    unsigned index;
    double
        oriNum, revOriNum, resFactor, renNum, row, numHat, numHat5, c, revNumHatA, revNumA, revNumB, error;
}store[9000000];
```

Specification:

Here I applied 9,000,000 units of memory space to store a struct which includes index and all kinds of computing factor of each number. Below is the detail specification of variables in the struct.

index: Index of 9,000,000 numbers.

oriNum: Original value of the number. (y)

revOriNum: Reciprocal value of the original number. ($\frac{1}{y}$)

resFactor: Rescale Factor. (r)

renNum: Renormalized value of the original number. ($y' = y \times r$)

row: Prescale Factor. (ρ)

numHat: This stores the prescaled value of the renormalized number. ($y^\wedge = y' \times \rho$)

numHat5: This is the truncated value of y^\wedge with accuracy of 5 decimal digits. (y_5^\wedge is truncated value, not rounded one of y^\wedge .)

c: A constant value which is defined by the formula: $c = \frac{1}{y_5^\wedge} - (2 - y_5^\wedge)$. (In real use, the value of c is got from a look up table)

revNumHatA: This stores the value of reciprocal prescaled renormalized number.

$$(\frac{1}{y_a^\wedge} = 2 - y_5^\wedge + c)$$

revNumA: This stores the postscaled value of *revNumHatA* above. ($\frac{1}{y_a} = \rho \times \frac{1}{y_a^*}$)
revNumB: Renormalize back the *revNumA* above, then store the value in this variable. ($\frac{1}{y_b} = r \times \frac{1}{y_a}$) <‘r’ is the rescale factor>
error: This stores the final error results between $\frac{1}{y}$ and $\frac{1}{y_b}$. ($|\frac{1}{y} - \frac{1}{y_b}|$)

Then let’s begin the addressing process.

Step One: Renormalize numbers to range $[0.8 \ll y \ll 1.28]$

Renormalize the original decimal numbers to range $[0.8 \ll y \ll 1.28]$ according to the table below.

Interval	Rescale Factor	Renormalized Range
[1, 1.6)	0.8	[0.8, 1.28)
[1.6, 2)	0.5	[0.8, 1.0)
[2, 3.2)	0.4	[0.8, 1.28)
[3.2, 4)	0.25	[0.8, 1.0)
[4, 6.4)	0.2	[0.8, 1.28)
[6.4, 10)	0.125	[0.8, 1.25)

(And we should remember to renormalize back at the end of the process.)

The according programming code is listed below.

```

if(store[index].oriNum<1.6)
    store[index].resFactor=0.8;
else if(store[index].oriNum<2)
    store[index].resFactor=0.5;
else if(store[index].oriNum<3.2)
    store[index].resFactor=0.4;
else if(store[index].oriNum<4)
    store[index].resFactor=0.25;
else if(store[index].oriNum<6.4)
    store[index].resFactor=0.2;
else
    store[index].resFactor=0.125;

store[index].renNum=store[index].oriNum*store[index].resFactor;

```

Specification:

Make a judgement of what range the original number is located, then find the rescale factor according to the table showed above. This is implemented by ‘if...else’ statements in the programming code above. Finally, multiply the rescale factor and the original number to get the renormalized number.

Step Two: Determine prescaled values of renormalized numbers ($y^{\wedge} = y' \times \rho$)

First, calculate the according prescale factor of each renormalized number.

Since the range of renormalized numbers is $[0.8 \ll y \ll 1.28]$, we equally divided the range into 48 parts. Each part has an interval of 0.01. Each according prescale factor is the reciprocal value of the midpoint of each interval (The value has accuracy of two decimal digits). For example, if the renormalized number locates in the range of interval $[0.80, 0.81)$, then the prescale factor can be determine like this:

$$\rho = \frac{1}{0.805} \approx 1.24 \text{ (two decimal digits accuracy).}$$

Therefore, the related programming code calculating the prescale factor is listed below.

```
store[index].row=floor(10000/(floor(100*store[index].renNum)+0.5)+0.5)/100;
```

Specification:

The 'floor' function in C++ programming language is used to truncate decimal numbers into its integer part like 'floor(1.977)=1'.

Second, compute the prescaled values of renormalized numbers ($y^\wedge = y' \times \rho$). Since we already got the prescale factor, it is easy to calculate and the according code is listed below.

```
store[index].numHat=store[index].row*store[index].renNum;
```

Step Three: Find c for 1000 values as follows from table.

$$c = \frac{1}{y_5^\wedge} - (2 - y_5^\wedge) \text{ (} y_5^\wedge \text{ is truncated value of } y^\wedge \text{ with accuracy of 5 decimal digits)}$$

The according programming code is listed below.

```
store[index].numHat5=floor(store[index].numHat*100000)/100000;
store[index].c=1/store[index].numHat5-(2-store[index].numHat5);
```

Step Four: Let $\frac{1}{y_a^\wedge} = 2 - y_5^\wedge + c$

The $\frac{1}{y_a^\wedge}$ is just the prescaled value of reciprocal decimal numbers obtained from LUT. The according programming code is listed below.

```
store[index].revNumHatA=(2-store[index].numHat5)+store[index].c;
```

Step Five: Postscale

Since the former values we calculated up to now is prescaled ($y^\wedge = y' \times \rho$), we should postscale it ($y_a = \frac{y_a^\wedge}{\rho}$). Then $\frac{1}{y_a} = \frac{1}{\frac{y_a^\wedge}{\rho}} = \rho \times \frac{1}{y_a^\wedge}$, so we only need to multiply prescale factor ρ to $\frac{1}{y_a^\wedge}$, the according programming code is listed below.

```
store[index].revNumA=store[index].row*store[index].revNumHatA;
```


Step Six: Renormalize back the reciprocal value.

Just like step five, since we have renormalized original numbers y to range $[0.8 \ll y \ll 1.28]$ ($y' = y \times r$), we should renormalize it back to original value range ($y_b = \frac{y_a}{r}$). Then $\frac{1}{y_b} = \frac{1}{\frac{y_a}{r}} = \frac{1}{y_a} \times r$, so we only need to multiply rescale factor r to $\frac{1}{y_a}$, the according programming code is listed below.

```
store[index].revNumB=store[index].revNumA*store[index].resFactor;
```

Step Seven: Compute Errors

This step is just to do the final thing which is the main purpose of the project, computer errors between the real value of reciprocal decimal numbers and the high precision value of reciprocal decimal numbers obtained from LUT. $Error = \left| \frac{1}{y} - \frac{1}{y_b} \right|$. The according programming code is listed below.

```
store[index].error=abs(store[index].revOriNum-store[index].revNumB);
```

Step Eight: Compute Sum error area

This step is to add up all the reciprocal of original numbers and add up all the approximated reciprocal of original numbers, then calculate the difference of these two sums. The related code is listed below:

```
for(unsigned i=0;i<9000000;i++)
{
    oriSum+=store[i].revOriNum;
    aprSum+=store[i].revNumB;
}
cout<<"Sum of Original Reciprocal Numbers:    "<<oriSum<<endl;
cout<<"Sum of Approximated Reciprocal Numbers: "<<aprSum<<endl;
cout<<"Difference between Sums:                "<<abs(oriSum-
aprSum)<<endl;
```

4. Other specification of the project program

a. I stores all the results into csv files. CSV stands for Comma-separated values, so CSV file is the file stores Comma-separated values which means it consists of any number of records, separated by line breaks of some kind; each record consists of fields, separated by some other character or string, most commonly a literal comma or tab. Usually, all records have an identical sequence of fields.

CSV is a common, relatively simple file format that is widely supported by consumer, business, and scientific applications and it is the main reason I chose this format to store results. CSV file

can easily open via Microsoft Excel and each field in CSV file occupies one column in Excel.

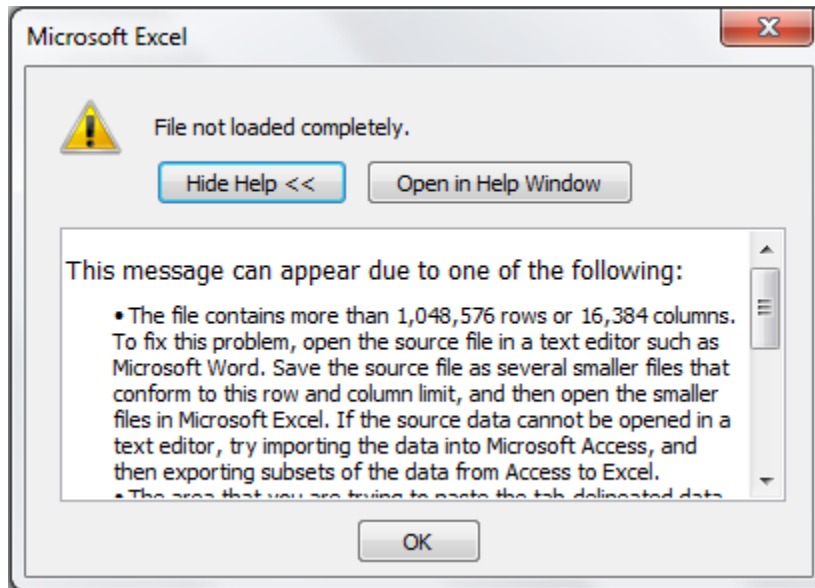
- b. Since the program is used to compute errors of 9,000,000 numbers, the volume of the results is really huge.

At first, I tried to store all the results into one single file and the file has 925 Megabytes storage!

Below is the screenshot of this single file.



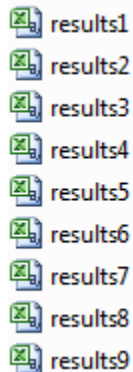
However, there is a problem with Microsoft Excel, that is Excel only can show 1,048,576 rows of the file (this is the maximum rows Excel can show). Since we have 9,000,000 numbers to calculate which means the file has 9,000,000 rows stored in and if I opened the single file anyway, I will get such pop-up tip.



Therefore, finally, I decided to store the whole results in 9 separate CSV files. That's why in the programming code, it has a loop which will run 9 iterations. The code is listed below.

```
for(unsigned i=1;i<=9;i++)
```

And the 9 results files is showed below.



c. Functions of Program

There are three main functions of this project program:

1) Start Running

This start calculating the 9,000,000 numbers and when it's done, there will be 9 CSV files appeared at the same place where the program locates.

2) Open CSV File

This will open one of the 9 CSV result files according to your choice. It will open the CSV file via the default software which can deal with CSV Files on Windows platform. On Windows platforms

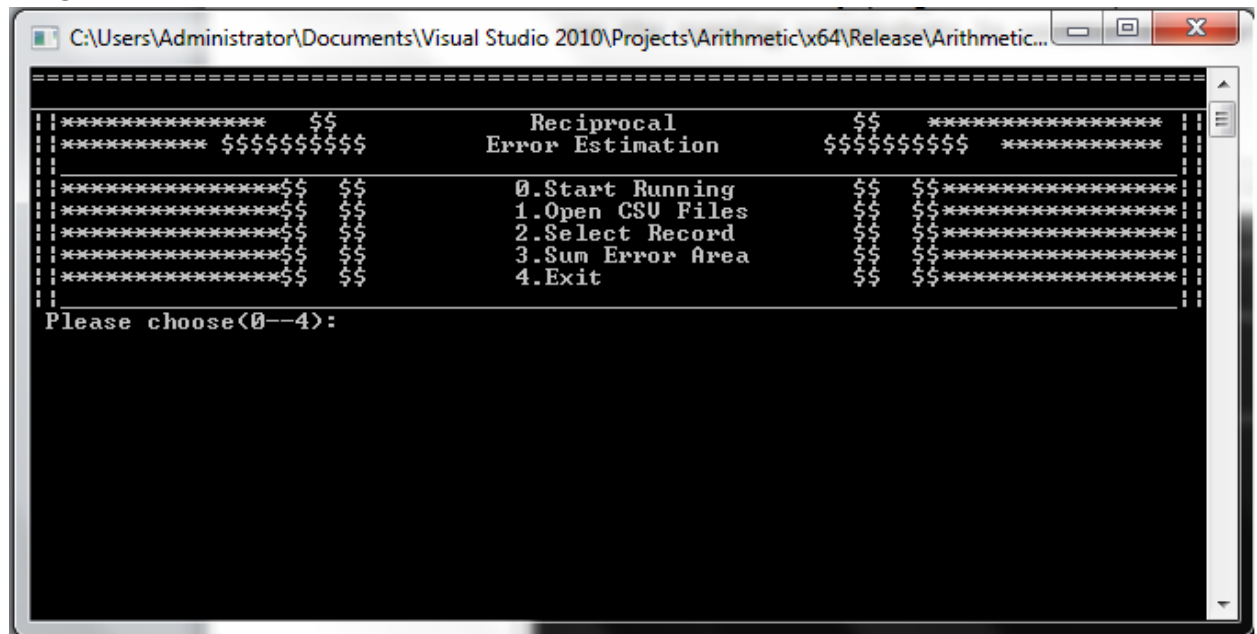
with Microsoft Office installed, the default software dealing with CSV files is Microsoft Excel.

3) Select Record

This is a function which can search a single number out of the 9,000,000 ones to view the whole record of each related factors of that number in my program itself. (For more details, please see the running screenshots in section d of part 4).

5. Screenshots of program running

a. Program Menu



```
=====
|*****  $$          Reciprocal  $$ *****|
|*****  $$$$$$      Error Estimation  $$$$$$ *****|
|-----|
|*****  $$  0.Start Running  $$ *****|
|*****  $$  1.Open CSU Files  $$ *****|
|*****  $$  2.Select Record  $$ *****|
|*****  $$  3.Sum Error Area  $$ *****|
|*****  $$  4.Exit           $$ *****|
|-----|
Please choose(0--4):
```

b. Start Running and Running Complete


```
C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Arithmetic\x64\Release\Arithmetic...

=====
|*****  $$          Reciprocal          $$  *****|
|*****  $$$$$$$$      Error Estimation  $$$$$$$$  *****|
|*****  $$$  $$$      0.Start Running    $$$  $$$*****|
|*****  $$$  $$$      1.Open CSU Files   $$$  $$$*****|
|*****  $$$  $$$      2.Select Record    $$$  $$$*****|
|*****  $$$  $$$      3.Sum Error Area    $$$  $$$*****|
|*****  $$$  $$$      4.Exit              $$$  $$$*****|
|*****  $$$  $$$*****|
|Please choose(0--4): 0|
|-----|
|Long time to wait...|
|-----|
```

```
C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Arithmetic\x64\Release\Arithmetic...

=====
|*****  $$          Reciprocal          $$  *****|
|*****  $$$$$$$$      Error Estimation  $$$$$$$$  *****|
|*****  $$$  $$$      0.Start Running    $$$  $$$*****|
|*****  $$$  $$$      1.Open CSU Files   $$$  $$$*****|
|*****  $$$  $$$      2.Select Record    $$$  $$$*****|
|*****  $$$  $$$      3.Sum Error Area    $$$  $$$*****|
|*****  $$$  $$$      4.Exit              $$$  $$$*****|
|*****  $$$  $$$*****|
|Please choose(0--4): 0|
|-----|
|Long time to wait...|
|Mission complete! Press any key to go back menu...|
|-----|
```

c. Open CSV Files


```

***** $$$$$$$$$$ Error Estimation $$$$$$$$$$ *****
*****$ $ 0.Start Running $ $*****
*****$ $ 1.Open CSU Files $ $*****
*****$ $ 2.Select Record $ $*****
*****$ $ 3.Sum Error Area $ $*****
*****$ $ 4.Exit $ $*****
Please choose(0--4): 2
-----
Please input the number you want to search:1.13242
Index=132421
Y=1.13242
1/Y=0.883064587
Rescale Factor=0.8
Renormalized Y=0.905936
Row=1.1
y^=0.9965296
y^(5)=0.99652
C=1.21526914e-005
1/(y^a)=1.00349215
1/(Ya)=1.10384137
1/(Yb)=0.883073094
Errors=8.50702448e-006
Mission complete! Press any key to go back menu...

```

e. Sum error Area

```

C:\Users\Administrator\Documents\Visual Studio 2010\Projects\Arithmetic\x64\Release\Arithmetic...
*****$ $ Reciprocal $ $*****
*****$ $ Error Estimation $ $*****
*****$ $ 0.Start Running $ $*****
*****$ $ 1.Open CSU Files $ $*****
*****$ $ 2.Select Record $ $*****
*****$ $ 3.Sum Error Area $ $*****
*****$ $ 4.Exit $ $*****
Please choose(0--4): 3
-----
Sum of Original Reciprocal Numbers: 2302585.54
Sum of Approximated Reciprocal Numbers: 2302597.03
Difference between Sums: 11.4885386
Mission complete! Press any key to go back menu...

```

6. Attachments of Whole C++ Code of Project Program

(This code only can be fully compatible on Visual Studio C++ platform)

```

// Arithmetic.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"

using std::string;
using std::stringstream;
using std::cout;
using std::cin;

```



```

using std::endl;
using std::ofstream;
using std::setprecision;

unsigned choice;
struct
{
    unsigned index;
    double
oriNum, revOriNum, resFactor, renNum, row, numHat, numHat5, c, revNumHatA, revNumA, revNumB, error;
}store[9000000];

int _tmain(int argc, _TCHAR* argv[])
{
    void print1(), print2();
    unsigned index=0;
    string fileNamePart1="results", fileNamePart3=".csv", fileName;
    stringstream ss;
    ofstream output;
    double oriSum=0, aprSum=0;
    do
    {
        print1();
        switch(choice)
        {
            case 0:
                cout<<"Long time to wait..."<<endl;
                for(unsigned i=1; i<=9; i++)
                {
                    output.close();
                    output.clear();
                    ss<<i;
                    fileName=fileNamePart1+ss.str()+fileNamePart3;
                    output.open(fileName);
                    ss.str("");
                    output<<"Index,Y,1/Y,Rescale Factor,Renormalized
Y,Row,y^,y^(5),C,1/(y^a),1/(Ya),1/(Yb),Errors"<<endl;
                    for(unsigned j=0; j<=999999; j++)
                    {
                        store[index].index=index+1;
                        store[index].oriNum=(i*1000000+j)/1000000.0;
                        store[index].revOriNum=1/store[index].oriNum;
                        if(store[index].oriNum<1.6)
                            store[index].resFactor=0.8;
                        else if(store[index].oriNum<2)
                            store[index].resFactor=0.5;
                        else if(store[index].oriNum<3.2)
                            store[index].resFactor=0.4;
                        else if(store[index].oriNum<4)
                            store[index].resFactor=0.25;
                        else if(store[index].oriNum<6.4)
                            store[index].resFactor=0.2;
                        else
                            store[index].resFactor=0.125;

                        store[index].renNum=store[index].oriNum*store[index].resFactor;

                        store[index].row=floor(10000/(floor(100*store[index].renNum)+0.5)+0.5)/100;

```



```

        store[index].numHat=store[index].row*store[index].renNum;

        store[index].numHat5=floor(store[index].numHat*100000)/100000;
        store[index].c=1/store[index].numHat5-(2-
store[index].numHat5);
        store[index].revNumHatA=(2-
store[index].numHat5)+store[index].c;

        store[index].revNumA=store[index].row*store[index].revNumHatA;

        store[index].revNumB=store[index].revNumA*store[index].resFactor;
        store[index].error=abs(store[index].revOriNum-
store[index].revNumB);

        output<<setprecision(9)<<store[index].index<<','<<store[index].oriNum<<','<
<<store[index].revOriNum<<','<<store[index].resFactor<<','<
<<store[index].renNum<<','<<store[index].row<<','<<store[index].numHat
<<','<<store[index].numHat5<<','<<store[index].c<<','<<store[index].revNumHatA
<<','<<store[index].revNumA<<','<<store[index].revNumB<<','<<store[index].error<<e
endl;

        index++;
    }
}
output.close();
print2();
break;
case 1:
    cout<<"There are 9 files with results, please choose(1--9):";
    cin>>choice;
    ss<<choice;
    fileName=fileNamePart1+ss.str()+fileNamePart3;
    ShellExecute(NULL,"open",fileName.c_str(),NULL,NULL,SW_SHOWNORMAL);
    ss.str("");
    print2();
    break;
case 2:
    double number;
    cout<<"Please input the number you want to search:";
    cin>>number;
    choice=number*1000000-1000000;
    cout<<"Index=";
    cout<<store[choice].index<<endl;
    cout<<"Y=";
    cout<<setprecision(9)<<store[choice].oriNum<<endl;
    cout<<"1/Y=";
    cout<<setprecision(9)<<store[choice].revOriNum<<endl;
    cout<<"Rescale Factor=";
    cout<<setprecision(9)<<store[choice].resFactor<<endl;
    cout<<"Renormalized Y=";
    cout<<setprecision(9)<<store[choice].renNum<<endl;
    cout<<"Row=";
    cout<<setprecision(9)<<store[choice].row<<endl;
    cout<<"y^=";

```



```

        cout<<setprecision(9)<<store[choice].numHat<<endl;
        cout<<"y^(5)=";
        cout<<setprecision(9)<<store[choice].numHat5<<endl;
        cout<<"C=";
        cout<<setprecision(9)<<store[choice].c<<endl;
        cout<<"1/(y^a)=";
        cout<<setprecision(9)<<store[choice].revNumHatA<<endl;
        cout<<"1/(Ya)=";
        cout<<setprecision(9)<<store[choice].revNumA<<endl;
        cout<<"1/(Yb)=";
        cout<<setprecision(9)<<store[choice].revNumB<<endl;
        cout<<"Errors=";
        cout<<setprecision(9)<<store[choice].error<<endl;
        print2();
        break;
    case 3:
        if(oriSum==0)
            for(unsigned i=0;i<9000000;i++)
            {
                oriSum+=store[i].revOriNum;
                aprSum+=store[i].revNumB;
            }
        cout<<setprecision(9)<<"Sum of Original Reciprocal Numbers:
"oriSum<<endl;
        cout<<setprecision(9)<<"Sum of Approximated Reciprocal Numbers:
"aprSum<<endl;
        cout<<setprecision(9)<<"Difference between Sums:
"<<abs(oriSum-aprSum)<<endl;
        print2();
        break;
    default:
        if(choice!=4)
        {
            cout<<"Wrong choice, press any key to go back menu...";
            cin.sync();
            getchar();
        }
    }
    while(choice!=4);
    return 0;
}

void print1()
{
    system("cls");
    cout<<"=====
=====";
    cout<<"
";
    cout<<"| *****    $$              Reciprocal              $$    *****
|";
    cout<<"| *****  $$$$$$$$$$      Error Estimation      $$$$$$$$$$  *****
|";

    cout<<"| _____|";
    cout<<"| *****$$  $$              0.Start Running              $$
$$$$*****|";

```



```

        cout<<"||*****$$  $$          1.Open CSV Files      $$
$$*****||";
        cout<<"||*****$$  $$          2.Select Record      $$
$$*****||";
        cout<<"||*****$$  $$          3.Sum Error Area      $$
$$*****||";
        cout<<"||*****$$  $$          4.Exit                $$
$$*****||";

cout<<"||_____||";
        cout<<" Please choose(0--4): ";
        cin>>choice;
        cout<<"-----
-----";
}

void print2()
{
        cout<<"Mission complete! Press any key to go back menu...";
        cin.sync();
        getchar();
}

```