Report for project part b
LP104

1509853G-I011-0202

Wang, Jingqing

The whole project part b was completed under visual studio 2008. For part b, it totally contains three files.

1.CompileFile.h

2.Compilefile.cpp

3. main.cpp

The file1 is a header file contents the class CompileFile. File2 contents the function of the class CompileFile. File3 is the main function of the project.

Before the compile, please put the measles.txt in the same folder with [project name].vcproj、[project name].aps、CompileFile.h、Compilefile.cpp and main.cpp files. If you want to change the file path, please add the absolute path of measles.txt into source code.

A、Class Function

In this class, it contains 11 functions.

1. void get_the_file_name();

This function is to asked user to input the name of file. After user input the file's name, it will add the relative path to the file's name. Then,it will call the open_file() function to open the file.

2. void get_the_years();

This function use ofstream to creat or write the result into the file named result.txt. In the middle, it use seekg function to locate the pointer of ifstream. Then, it use string array to store every line of the file measles.txt. At the same time, it will check the specific substring in every element ofstring array to select the line that meet the user search condition.

3. void showreport();

This function is to show the result of the searching.

4. void income_level();

This function is to asked user to input the income level he want to search. It will show that which number corresponds to which level. After it gets the input from user, it will use ifstream to open resule.txt to

search the condition that meets user's input. Then, using ofstream to output the resule to another file named finalresult.txt.

At last, it will call process_file() function.

5. void process_file(ifstream&fin);

This process_file function is to process the data that selected by the user search condition input. It will extract the substring that stand for percent vaccinated and store in the file named percent.txt.

At the same time, it transform the char to the int as well as store in the int array, and caculate the sum ,average of the percent.

After all above, it call Find_max_min() function.

6. void creat_result_file();

This function is to creat the result_file for the search years part.

7. void Find_max_min();

This function is to find the maximum and the minimum of percent in the finalresult.txt. It use the int array to compara the number of percent, then out put the whole line of that number.

8. Ifstream & open_file();

This function is used to open the file. It use the ifstream as reference,

and it will return the ifstream fin after open the file that user input.

9. bool checkFile();

This function is to check the file that user input if exist and correct. In the main function, it was used to control the loop. If the file doesn't exist, it will asked user to input again until user input the correct file name. The return type is boolean.

10. bool checkYear();

This function is to check if the user input the correct years' search condition. Whatever user inputted, the get_the_years() function will search in the file measles.txt, but if the user input incorrect years, the resule file will be empty. This checkYear function control the loop in the main.

It returns boolean type.

11. void linenumber ();

This function is to calculate the amount of the line in the file measles.txt. Then redefine the Dynamic tring array.

B、Whole idea

The whole idea of project is using ifstream and ofstream as private stream in the class. I use these two stream as a file container to transport the file between the function in the class. In the stream, I mainly use string array and seekg() to process the file.

C、 Main function

1. Firstly, declare a CompileFile type named newfile.

2. Using get_file_name ,checkFile and a do...while loop to let user input the file name, and open the file which was put in Debug folder.

3. Creat a middle transit file named result.txt. Then, use function linenumber() to calculate the line of opening file.

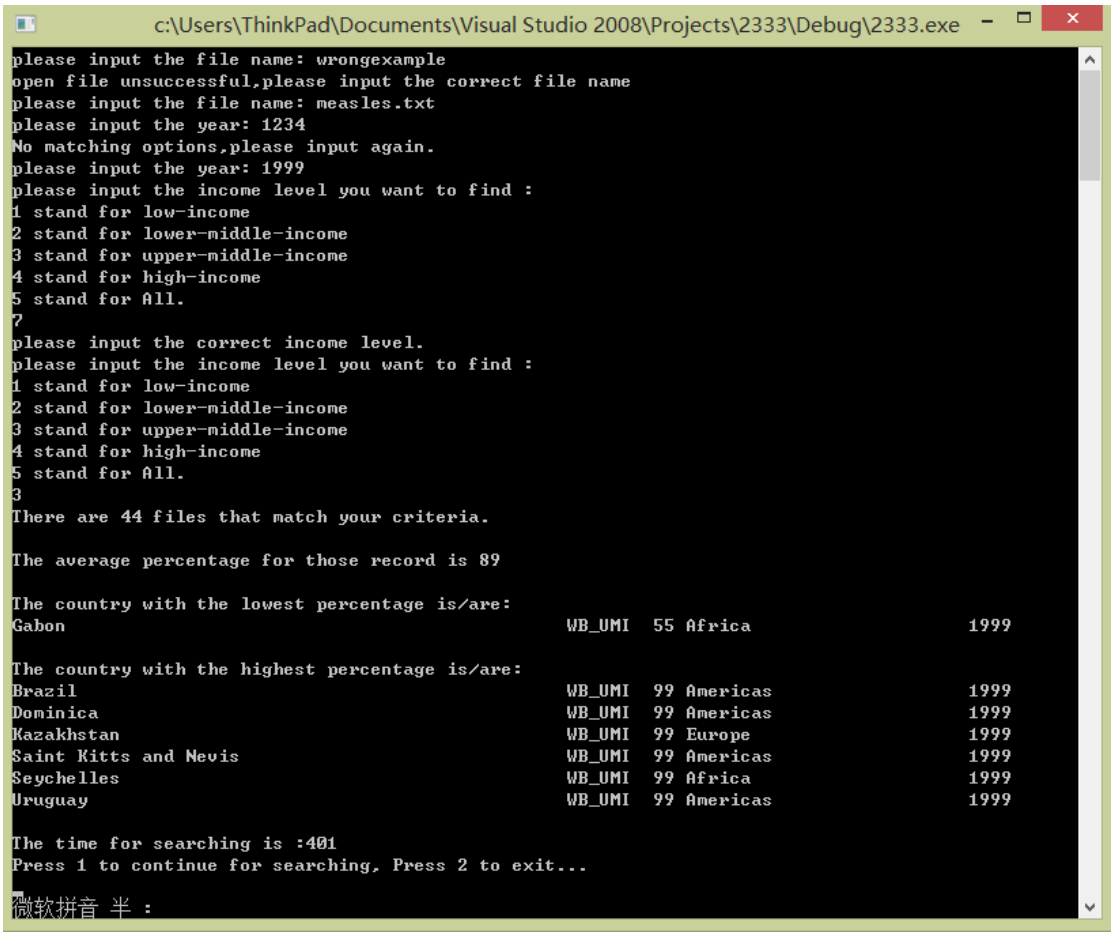
4. Use function get_the_year ,checkYear and a do...while loop to let user input the years that he wants to search.

5. Using income_level function to let user input the income level he want to search.

6. Using showreport function to show the result of the search.

7. The exit controller.

D、Simple output.



E、Source code

1. CompileFile.h

```
#include<iostream>

#include<cstdlib>

#include<string>

#include<fstream>

#include <sstream>

#include<time.h>

using namespace std;


class CompileFile{


public:

    CompileFile(){}


    void get_the_file_name();

    void get_the_years();

    void showreport();

    void income_level();

    void process_file(ifstream&fin);

    void creat_result_file();

    void Find_max_min();
```

```
    void linenumber ();
    ifstream& open_file();
    bool checkFile();
    bool checkYear();

private :
    string strname;
    string stryears;
    string *str1;
    string incomeLevel;
    string *maxC;
    string *minC;
    int i,number_of_result,n,m,k,sum,min,max,linenumbers;
    int *Percent_Vaccinated;
    double average,usetime;
    ifstream pin;
    ofstream pout;
    ifstream fin;
    ofstream fout;
    ifstream qin;
    };
```

2. Compilefile.cpp

```
#include"CompileFile.h"


void CompileFile::get_the_file_name()
{
    usetime =0;
    string newname;
    cout<<"please input the file name: ";
    cin>>newname;


     clock_t start_time=clock();
    {
    strname="./ "+newname;
     open_file();
     }
    clock_t end_time=clock();
    usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_S
EC*1000;
}


ifstream & CompileFile::open_file()
{
```

```
    clock_t start_time=clock();
    {
    fin.open(strname.c_str(),ios::in|ios::out);
     }
    clock_t end_time=clock();

usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_SEC*
1000;
    return fin;
}

bool CompileFile::checkFile()
{

    if(!fin.is_open())
    {
        cout<<"open file unsuccessful,please input the correct file
name"<<endl;
          fin.close();
          return false;
    }
    else
```

```
    {return true;}
}

void CompileFile::creat_result_file()
{
    clock_t start_time=clock();
    {
    fout.open("./result.txt");
    fout.close();
      }
    clock_t end_time=clock();

usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_SEC*
1000;
}

void CompileFile::get_the_years()
{


    cout<<"please input the year: ";
    cin>>stryears;
```

```
  clock_t start_time=clock();
{
      fout.open("./result.txt",ios::in|ios::out);
if(!fout.is_open())
{
      cout<<"Open fail.";
      exit(1);
}
      n=stryears.length();

fin.seekg(0,ios::beg);

for(i=0;fin.peek() != EOF;i++)
{
      getline(fin,str1[i]);
      if(str1[i].substr(88,n) == stryears)
      {
      fout<<str1[i]<<endl;}
      else if(stryears==""||stryears=="all"||stryears=="ALL")
      {
      fout<<str1[i]<<endl;}
}
```

```
    fin.close();
    fout.flush();
    fout.close();
     }
    clock_t end_time=clock();

usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_SEC*
1000;
}

bool CompileFile::checkYear()
{
    pin.open("./result.txt",ios::in|ios::out);
    if(!pin.is_open())
    {
        cout<<"Open fail.";
    }
    if(pin.get()==EOF)
    {
    cout<<"No matching options,please input again."<<endl;
    fin.open(strname.c_str(),ios::in|ios::out);
```

```
        pin.clear();
        pin.close();
        return false;
        }
        else return true;

}

void CompileFile::income_level()
{

         int p;
        number_of_result=0;
        incomeLevel="0"   ;
        while(incomeLevel=="0")
        {
             cout<<"please input the income level you want to find : "
                  <<endl<<"1 stand for low-income"
                  <<endl<<"2 stand for lower-middle-income"
                  <<endl<<"3 stand for upper-middle-income"
                  <<endl<<"4 stand for high-income"
                  <<endl<<"5 stand for All."
```

```
            <<endl;


    cin>>p;
    switch(p)
    {
    case 1:
        {incomeLevel="WB_LI";break;}
    case 2:
        {incomeLevel="WB_LMI";break;}
    case 3:
        {incomeLevel="WB_UMI";break;}
    case 4:
        {incomeLevel="WB_HI";break;}
    case 5:
            {incomeLevel="ALL";break;}
     default:
        cout<<"please input the correct income level."<<endl;
    }
    }
clock_t start_time=clock();
    {
     pin.close();
```

```
pin.open("./result.txt");
if(!pin.is_open())
{
	cout<<"Open fail.";
	exit(1);
}

fout.open("./finalresult.txt");
if(!fout.is_open())
{
	cout<<"Open fail.";
	exit(1);
}

  n=incomeLevel.length();
pin.seekg(0,ios::beg);
for(i=0;pin.peek() != EOF;i++)
{
	getline(pin,str1[i]);
	if(str1[i].substr(51,n) == incomeLevel)
	{
	fout<<str1[i]<<endl;
```

```
            number_of_result++;
            }
            if(incomeLevel=="ALL")
            {
            fout<<str1[i]<<endl;
            number_of_result++;
            }
        }
        fout.flush();
        fout.close();
        fin.open("./finalresult.txt");
        process_file(fin);

        }
        clock_t end_time=clock();

usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_SEC*
1000;
}

void CompileFile::process_file(ifstream&fin)
{
```

```
    sum=0;
    Percent_Vaccinated=new int[number_of_result];
    fout.open("./percent.txt");
    if(!fout.is_open())
    {
        cout<<"Open fail.";
        exit(1);
    }
    fin.seekg(0,ios::beg);
    for(i=0;fin.peek() != EOF;i++)
    {
        getline(fin,str1[i]);

        fout<<str1[i].substr(59,2)<<endl;
        Percent_Vaccinated[i]=atoi(str1[i].substr(59,3).c_str());
        sum+=Percent_Vaccinated[i];
    }
    fin.close();
    average=sum/i;
    Find_max_min();

}
```

```
void CompileFile::Find_max_min()
{
    m=k=0;
     min=Percent_Vaccinated[0];
    max=Percent_Vaccinated[0];

    for(i=1;i<number_of_result;i++)
    {
        if(max<Percent_Vaccinated[i])
        {max=Percent_Vaccinated[i];
        }

        if(min>Percent_Vaccinated[i])
        {min=Percent_Vaccinated[i];
        }
    }

    stringstream ss;

     string maxS,minS;
     ss<<max;
```

```
  ss>>maxS;


stringstream ssr;
ssr<<min;
ssr>>minS;


fin.open("./finalresult.txt");
if(!fin.is_open())
{
      cout<<"Open fail.";
      exit(1);
}
fin.seekg(0,ios::beg);


for(i=0;fin.peek() != EOF;i++)
{
      getline(fin,str1[i]);
      if(str1[i].substr(59,maxS.length()) == maxS)
            {maxC[m]=str1[i];m++;}
      if(str1[i].substr(59,minS.length()) == minS)
      {minC[k]=str1[i];k++;}
      if(str1[i].substr(59,minS.length()) == "
```

```
"&&str1[i].substr(60,minS.length()) == minS)
            {minC[k]=str1[i];k++;}
      }
}

void CompileFile::linenumber ()
{
      clock_t start_time=clock();
      {
      char next;
      int q=0;

      qin.open(strname.c_str(),ios::in|ios::out);
        if(!qin.is_open())
      {
            cout<<"Open fail.";
            exit(1);
      }

      qin.get(next);
      q++;
      for(i=0;qin.peek() != EOF;i++)
```

```
{
if(next=='\n')
{q++;}

        qin.get(next);
}
linenumbers=q;
str1=new string[linenumbers];
  maxC=new string[linenumbers];
  minC=new string[linenumbers];

qin.clear ();
qin.close();

}
clock_t end_time=clock();

usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_SEC*
1000;
}

void CompileFile::showreport()
```

```
{
    clock_t start_time=clock();
    {
    cout<<"There are "<<number_of_result<<" files that match your
criteria. "<<endl<<endl;
    cout<<"The average percentage for those record is
"<<average<<endl;
    cout<<endl<<"The country with the lowest percentage is/are:
"<<endl;
    for(i=0;i<k;i++){ cout<<minC[i]<<endl;}
    cout<<endl<<"The country with the highest percentage is/are:
"<<endl;
    for(i=0;i<m;i++) {cout<<maxC[i]<<endl;}
     }
    clock_t end_time=clock();

usetime+=static_cast<double>(end_time-start_time)/CLOCKS_PER_SEC*
1000;
    cout<<endl<<"The time for searching is :"<<usetime<<endl;
       }
```

3. main.cpp

```
#include"CompileFile.h"

using namespace std;


int main()

{

    while(1)

    {

    CompileFile newfile;

    do {

        newfile.get_the_file_name();


    }while(newfile.checkFile()==false);


    newfile.creat_result_file();

    newfile.linenumber();


    do{

        newfile.get_the_years();


    }

    while(newfile.checkYear()==false);
```

```
        newfile.income_level();
        newfile.showreport();

        int u;
          cout<<"Press 1 to continue for searching, Press 2 to exit..."<<endl;
              cin>>u;
        if(u==1)continue;
        else if (u==2)break;
        }

        return 0;
}
```