# 实验报告

姓名：熊雷 学号：2019202814

1. 需求分析

给定大规模数据集，包含美国股市和中国股市的每日行情数据，编写 C++程序完成要求的数据处理、分析和可视化等任务。

1.**计算涨跌额和涨跌幅：**

计算每只股票每日的涨跌额和涨跌幅。计算公式如下：

涨跌额 = 今日收盘价 – 昨日收盘价 涨跌幅 = （涨跌额 / 昨日收盘价）\* 100%

将计算出的涨跌额和涨跌幅添加到原来的数据集中，即新增两列，存成文件。

**2.**

**外排序：**

将每日股票行情数据按照涨跌幅从大到小排列，日期由近及远。即对数据集按照日期和涨跌幅两列进行降序排列，先排日期，最靠近现在的日期排在最前；再对同一日内的所有股票行情按照涨跌幅的降序排列。

**3.**

**计算夏普比率：**

计算每只股票每年内涨跌幅的平均值和标准差，这里视为股票的年平均收益和标准差，用年平均收益除以标准差计算出该股票该年的夏普比率。

**4.**

**输出某月单日涨跌幅最大的 k 条股票交易数据：**

对某一个指定的单月份的数据（已划分好的单月份数据的外部存储），进行外部排序。

排序的关键字为涨跌幅 的绝对值，按降序排序。指定单月份后（例如 2020 年 06 月），访问读取并排序单月份数据（如 2020-06.csv），单日涨跌幅最大的 k 条股票交易数据。严格限制内存大小（最多使用 10M 内存空间），通过检查 Windows 任务管理器的监控内存占用并测试。

**5.**

**可视化：**任意选择某时间段（365 天）内，夏普比例最高的 10 只股票，使用 QT 和外部库画出他们在指定期间（1 年）的日 K 线图。

6. 基本思路：

功能1：使用c++文件读写就能完成，主要注意的是字符串的赋值复制等操作；

功能2：排序使用快排，自定义比较器；有一种读写文件特别方便的方式就是将文件按照二进制来读写；然后为了使外排序的复用率更高，使用一个status参数来决定每次使用的比较函数；

功能3：使用两个参数来记录上一次的时间和股票名，当时间过了一年之后就计算一次夏普率然后写入，股票类型变化时也需要写入；

功能4：在分割文件之前，利用外排序2号比较器生成一个按照涨跌幅降序得到的外排序文件，然后再把该文件分割为每个月份，这样根据输入参数年、月和k就可以直接读文件进行输出。

功能5：利用外部库实现。 （具体实现结果见实验报告四、实验结果以及视频）

代码中包含const、static、类的继承、复制构造等

1. 概要设计
2. 抽象数据类型设计
3. 股票类

class Base\_stock

{

public:

Base\_stock() {}

Base\_stock(string dt,string sy,float op,float hi,float lo,float cl,float vo,float zdv,float zdr) {

}

string getDate(){return datetime;}

string getSym(){return symbol;}

float getOpen(){return open;}

float getHigh(){return high;}

float getLow(){return low;}

float getClose();

float getVolume(){return volume;}

float getzdv(){ return zdval; }

float getzdr() {return zdrate;}

void setzdval(float);

void setzdrate(float);

private:

string datetime;//date

string symbol;//股票代码

float open;//开盘价

float high;//最高价

float low;//最低价

float close;//收盘价

float volume;//成交量

float zdval;//涨跌额

float zdrate;//涨跌率

};

class Zh\_stock:public virtual Base\_stock{

public:

Zh\_stock(){}

void computeZD1(Zh\_stock);//涨跌额

void computeZD2(Zh\_stock);//涨跌率

private:

float share;//流动股本

float turnover;//换手率=成交量/流动股本

};

class Us\_stock:public Base\_stock

{

public:

Us\_stock() {}

Us\_stock(string dt,string sy,float op,float hi,float lo,float cl,float vo,float zdv,float zdr,float adj)

:Base\_stock(dt,sy,op,hi,lo,cl,vo,zdv,zdr)

{

this->adjclose=adj;

}

float getAdj(){return adjclose;}

private:

float adjclose;//复权后的收盘价

};

float Base\_stock::getClose(){

return close;

}

void Base\_stock::setzdval(float a){

zdval=a;

}

void Base\_stock::setzdrate(float a){

zdrate=a;

}

1. 大根堆节点

struct MaxheapNode

{

Us\_stock element;

int index;

};

1. 大根堆类（败者树）

class Maxheap

{

private:

MaxheapNode \*val;

int capacity;//最小堆的最大容纳量

int n\_size;//当前堆中元素数

int status;//选择比较函数1.2两种

public:

Maxheap(MaxheapNode a[],int size,int s) {

val=a;n\_size=size;status=s;

int i=(n\_size-1)/2;

while(i>=0) {

heapify(i);

i--;

}

}

void heapify(int);//使得堆满足最小最大化

int parent(int i){return (i-1)/2;}

int leftson(int i){return 2\*i+1;}

int rightson(int i) {return 2\*i+2;}

MaxheapNode getMax();//返回根

void replaceroot(MaxheapNode x);

};

1. 程序主要模块
2. 主函数
3. 文件读写
4. 外排序
5. 夏普率计算
6. 每月单日k大涨跌率
7. 可视化
8. 详细设计

Maxheap.h

#ifndef MAXHEAP\_H

#include"stock.h"

#define MAXHEAP\_H

#endif // MAXHEAP\_H

struct MaxheapNode

{

Us\_stock element;

int index;

};

Us\_stock Minstock=Us\_stock("0/0/0","aaa",-1,-1,-1,-1,-1,-1,-1,-1);

bool islagerThan1(Us\_stock a,Us\_stock b)//1为a>b

{

string dt1=a.getDate();string dt2=b.getDate();

//if(dt1.empty()==1 || dt2.empty()==1) cout<<1<<endl;

//string y1=NULL;string m1=NULL;string d1=NULL;string y2=NULL;string m2=NULL;string d2=NULL;

int tmp1=dt1.find\_first\_of('/');int tmp2=dt1.find\_last\_of('/');

int y1=atoi(dt1.substr(0,tmp1).c\_str());int m1=atoi(dt1.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d1=atoi(dt1.substr(tmp2+1,dt1.length()-tmp2).c\_str());

tmp1=dt2.find\_first\_of('/');tmp2=dt2.find\_last\_of('/');

int y2=atoi(dt2.substr(0,tmp1).c\_str());int m2=atoi(dt2.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d2=atoi(dt2.substr(tmp2+1,dt2.length()-tmp2).c\_str());

//compare

if(y1>y2) {

return true;

}

else{

if(y1==y2){

if(m1>m2) return true;

else{

if(m1==m2){

if(d1>d2) return true;

else{

if(d1==d2)

return a.getzdr()>=b.getzdr()?true:false;

}

}

}

}

}

return false;

}

bool islagerThan2(Us\_stock a,Us\_stock b)//1为a>b

{

string dt1=a.getDate();string dt2=b.getDate();

//if(dt1.empty()==1 || dt2.empty()==1) cout<<1<<endl;

//string y1=NULL;string m1=NULL;string d1=NULL;string y2=NULL;string m2=NULL;string d2=NULL;

int tmp1=dt1.find\_first\_of('/');int tmp2=dt1.find\_last\_of('/');

int y1=atoi(dt1.substr(0,tmp1).c\_str());int m1=atoi(dt1.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d1=atoi(dt1.substr(tmp2+1,dt1.length()-tmp2).c\_str());

tmp1=dt2.find\_first\_of('/');tmp2=dt2.find\_last\_of('/');

int y2=atoi(dt2.substr(0,tmp1).c\_str());int m2=atoi(dt2.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d2=atoi(dt2.substr(tmp2+1,dt2.length()-tmp2).c\_str());

//compare

if(y1>y2) {

return true;

}

else{

if(y1==y2){

if(m1>m2) return true;

else{

if(m1==m2){

if(d1>d2) return true;

else{

if(d1==d2)

return abs(a.getzdr())>=abs(b.getzdr())?true:false;

}

}

}

}

}

return false;

}

bool islagerThan3(Us\_stock a,Us\_stock b)//1为a>b

{

string dt1=a.getDate();string dt2=b.getDate();

//if(dt1.empty()==1 || dt2.empty()==1) cout<<1<<endl;

//string y1=NULL;string m1=NULL;string d1=NULL;string y2=NULL;string m2=NULL;string d2=NULL;

int tmp1=dt1.find\_first\_of('/');int tmp2=dt1.find\_last\_of('/');

int y1=atoi(dt1.substr(0,tmp1).c\_str());int m1=atoi(dt1.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d1=atoi(dt1.substr(tmp2+1,dt1.length()-tmp2).c\_str());

tmp1=dt2.find\_first\_of('/');tmp2=dt2.find\_last\_of('/');

int y2=atoi(dt2.substr(0,tmp1).c\_str());int m2=atoi(dt2.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d2=atoi(dt2.substr(tmp2+1,dt2.length()-tmp2).c\_str());

//compare

return a.getzdr()>b.getzdr()?true:false;

}

void swap(MaxheapNode \*a,MaxheapNode \*b){

MaxheapNode temp;

temp=\*a;

\*a=\*b;

\*b=temp;

}

class Maxheap

{

private:

MaxheapNode \*val;

int capacity;//最小堆的最大容纳量

int n\_size;//当前堆中元素数

int status;//选择比较函数1.2两种

public:

Maxheap(MaxheapNode a[],int size,int s) {

val=a;n\_size=size;status=s;

int i=(n\_size-1)/2;

while(i>=0) {

heapify(i);

i--;

}

}

void heapify(int);//使得堆满足最小最大化

int parent(int i){return (i-1)/2;}

int leftson(int i){return 2\*i+1;}

int rightson(int i) {return 2\*i+2;}

MaxheapNode getMax();//返回根

void replaceroot(MaxheapNode x);

};

void Maxheap::heapify(int i){

int l=leftson(i);

int r=rightson(i);

int maxnum=i;

if(status==1){

if(l<n\_size && islagerThan1(val[l].element,val[maxnum].element)==true) maxnum=l; //与左儿子相比

if(r<n\_size && islagerThan1(val[r].element,val[maxnum].element)==true) maxnum=r; //与右儿子相比

}

if(status==2){

if(l<n\_size && islagerThan2(val[l].element,val[maxnum].element)==true) maxnum=l; //与左儿子相比

if(r<n\_size && islagerThan2(val[r].element,val[maxnum].element)==true) maxnum=r; //与右儿子相比

}

if(status==3){

if(l<n\_size && islagerThan3(val[l].element,val[maxnum].element)==true) maxnum=l; //与左儿子相比

if(r<n\_size && islagerThan3(val[r].element,val[maxnum].element)==true) maxnum=r; //与右儿子相比

}

if(maxnum!=i) // 如果根节点小，则下沉

{

swap(&val[maxnum],&val[i]);

heapify(maxnum);

}

}

MaxheapNode Maxheap::getMax(){

return val[0];

}

void Maxheap::replaceroot(MaxheapNode x){

val[0]=x;

heapify(0);

}

Stock.h

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<Qstring>

#include<cmath>

using namespace std;

class Base\_stock

{

public:

Base\_stock() {}

Base\_stock(string dt,string sy,float op,float hi,float lo,float cl,float vo,float zdv,float zdr) {

this->datetime=dt; this->symbol=sy; this->open=op; this->high=hi; this->low=lo;

this->close=cl; this->volume=vo; this->zdval=zdv; this->zdrate=zdr;

}

string getDate(){return datetime;}

string getSym(){return symbol;}

float getOpen(){return open;}

float getHigh(){return high;}

float getLow(){return low;}

float getClose();

float getVolume(){return volume;}

float getzdv(){ return zdval; }

float getzdr() {return zdrate;}

void setzdval(float);

void setzdrate(float);

private:

string datetime;//date

string symbol;//股票代码

float open;//开盘价

float high;//最高价

float low;//最低价

float close;//收盘价

float volume;//成交量

float zdval;//涨跌额

float zdrate;//涨跌率

};

class Zh\_stock:public virtual Base\_stock{

public:

Zh\_stock(){}

void computeZD1(Zh\_stock);//涨跌额

void computeZD2(Zh\_stock);//涨跌率

private:

float share;//流动股本

float turnover;//换手率=成交量/流动股本

};

class Us\_stock:public Base\_stock

{

public:

Us\_stock() {}

Us\_stock(string dt,string sy,float op,float hi,float lo,float cl,float vo,float zdv,float zdr,float adj)

:Base\_stock(dt,sy,op,hi,lo,cl,vo,zdv,zdr)

{

this->adjclose=adj;

}

float getAdj(){return adjclose;}

private:

float adjclose;//复权后的收盘价

};

float Base\_stock::getClose(){

return close;

}

void Base\_stock::setzdval(float a){

zdval=a;

}

void Base\_stock::setzdrate(float a){

zdrate=a;

}

Sharpe.cpp

#include<iostream>

#include<cstdio>

#include<cstring>

#include<fstream>

#include<string>

#include<cstdlib>

#include<cmath>

using namespace std;

float zdrmean[370];

bool thisyear=true;

bool thisstock=true;

char \*lastsy=NULL;char \*lastyear=NULL;int cnt1=0;float xiapu=0.0;

float cal1(float a[],int size){

if(size==1) return 0;

float mean=0.0,std=0.0;

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

mean+=a[i];

}

mean/=size;

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

{

std+=(a[i]-mean)\*(a[i]-mean);

}

std/=size;

std=sqrt(std);

if(std==0) return 0;

return mean/std;

}

int main()

{

char\* dt;char\* sy;float op;float hi;float lo;float cl;float vo;float zdv=0;float zdr=0;float adj;

FILE \*fp=fopen("C:/Users/86158/Desktop/����2����ҵ/data/test\_us\_stock\_daily1.csv","r");

FILE \*fp1=fopen("C:/Users/86158/Desktop/����2����ҵ/data/test\_us\_stock\_dailyxiapu.csv","w") ;

char buf[128];

string temp;

string temp1;

string temp2;

string temp3;

int i=0;

while(fgets(buf,128,fp))

{

dt=NULL;op=0.0;hi=0.0;lo=0.0;cl=0.0;vo=0.0;

sy=NULL;zdv=0.0;zdr=0.0;adj=0.0;

int cnt=0;//������

char \*substr=strtok(buf,",");

while(substr)

{

switch(cnt)

{

case 0:{

temp=substr;

dt=(char \*)temp.c\_str();

break;

}

case 1:{

temp1=substr;

sy=(char \*)temp1.c\_str();

break;

}

case 2:op=strtof(substr,NULL);

case 3:hi=strtof(substr,NULL);

case 4:lo=strtof(substr,NULL);

case 5:cl=strtof(substr,NULL);

case 7:vo=strtof(substr,NULL);

case 6:adj=strtof(substr,NULL);

case 8:zdv=strtof(substr,NULL);

case 9:zdr=strtof(substr,NULL);

}

cnt++;

substr=strtok(NULL,",");

}

char \*thisyear=NULL;

if(i==0)

{

temp2=sy;

lastsy=(char \*)temp2.c\_str();

lastyear=strtok(dt,"/");

zdrmean[cnt1++]=zdr;

}

else

{

if(strcmp(lastsy,sy)==0)

{

thisyear=strtok(dt,"/");

if(strcmp(thisyear,lastyear)==0)

{

zdrmean[cnt1++]=zdr/100;

}

else

{ //���ݲ�ͬ

xiapu=cal1(zdrmean,cnt1);

cnt1=0;

cout<<lastyear<<sy<<xiapu<<endl;

fprintf(fp1,"%s,%s,%f,\n",lastyear,sy,xiapu);

memset(zdrmean,false,sizeof(zdrmean));

temp3=thisyear; //����lastyear

lastyear=(char \*)temp3.c\_str();

}

}

else

{

thisyear=strtok(dt,"/");

xiapu=cal1(zdrmean,cnt1);

cout<<lastyear<<lastsy<<xiapu<<endl;

fprintf(fp1,"%s,%s,%f,\n",lastyear,lastsy,xiapu);

memset(zdrmean,false,sizeof(zdrmean));

temp3=thisyear; //����lastyear

lastyear=(char \*)temp3.c\_str();

cnt1=1;

temp2=sy; //����lastsy

lastsy=(char \*)temp2.c\_str();

}

}

i++;

}

}

Extenalsort：

bool cmp1(Us\_stock a,Us\_stock b)

{

string dt1=a.getDate();string dt2=b.getDate();

//if(dt1.empty()==1 || dt2.empty()==1) cout<<1<<endl;

//string y1=NULL;string m1=NULL;string d1=NULL;string y2=NULL;string m2=NULL;string d2=NULL;

int tmp1=dt1.find\_first\_of('/');int tmp2=dt1.find\_last\_of('/');

int y1=atoi(dt1.substr(0,tmp1).c\_str());int m1=atoi(dt1.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d1=atoi(dt1.substr(tmp2+1,dt1.length()-tmp2).c\_str());

tmp1=dt2.find\_first\_of('/');tmp2=dt2.find\_last\_of('/');

int y2=atoi(dt2.substr(0,tmp1).c\_str());int m2=atoi(dt2.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d2=atoi(dt2.substr(tmp2+1,dt2.length()-tmp2).c\_str());

//compare

if(y1>y2) {

return true;

}

else{

if(y1==y2){

if(m1>m2) return true;

else{

if(m1==m2){

if(d1>d2) return true;

else{

if(d1==d2)

return a.getzdr()>=b.getzdr()?true:false;

}

}

}

}

}

return false;

}

bool cmp2(Us\_stock a,Us\_stock b)

{

string dt1=a.getDate();string dt2=b.getDate();

//if(dt1.empty()==1 || dt2.empty()==1) cout<<1<<endl;

//string y1=NULL;string m1=NULL;string d1=NULL;string y2=NULL;string m2=NULL;string d2=NULL;

int tmp1=dt1.find\_first\_of('/');int tmp2=dt1.find\_last\_of('/');

int y1=atoi(dt1.substr(0,tmp1).c\_str());int m1=atoi(dt1.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d1=atoi(dt1.substr(tmp2+1,dt1.length()-tmp2).c\_str());

tmp1=dt2.find\_first\_of('/');tmp2=dt2.find\_last\_of('/');

int y2=atoi(dt2.substr(0,tmp1).c\_str());int m2=atoi(dt2.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d2=atoi(dt2.substr(tmp2+1,dt2.length()-tmp2).c\_str());

//compare

if(y1>y2) {

return true;

}

else{

if(y1==y2){

if(m1>m2) return true;

else{

if(m1==m2){

if(d1>d2) return true;

else{

if(d1==d2)

return abs(a.getzdr())>=abs(b.getzdr())?true:false;

}

}

}

}

}

return false;

}

bool cmp3(Us\_stock a,Us\_stock b)

{

string dt1=a.getDate();string dt2=b.getDate();

int tmp1=dt1.find\_first\_of('/');int tmp2=dt1.find\_last\_of('/');

int y1=atoi(dt1.substr(0,tmp1).c\_str());int m1=atoi(dt1.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d1=atoi(dt1.substr(tmp2+1,dt1.length()-tmp2).c\_str());

tmp1=dt2.find\_first\_of('/');tmp2=dt2.find\_last\_of('/');

int y2=atoi(dt2.substr(0,tmp1).c\_str());int m2=atoi(dt2.substr(tmp1+1,tmp2-tmp1-1).c\_str());int d2=atoi(dt2.substr(tmp2+1,dt2.length()-tmp2).c\_str());

//compare

return a.getzdr()>b.getzdr()?true:false;

}

FILE\* fileopen(char \*filename,char \*mode)

{

FILE\* fp = fopen(filename, mode);

if (fp == NULL) {

perror("Error while opening the file.\n");

exit(EXIT\_FAILURE);

}

return fp;

}

void fun1(Us\_stock &a,char \*buff)

{

string dt;string sy;float op=0.0;float hi=0.0;

float lo=0.0;float cl=0.0;float vo=0.0;float zdv=0.0;

float zdr=0.0;float adj=0.0;

int cnt=0;

char \*substr=strtok(buff,",");

while(substr){

switch(cnt)

{

case 0:{dt=substr;break;}

case 1:{

sy=substr;

break;

}

case 2:{

op=strtof(substr,NULL);

break;

}

case 3:{

hi=strtof(substr,NULL);

break;

}

case 4:{

lo=strtof(substr,NULL);

break;

}

case 5:{

cl=strtof(substr,NULL);

break;

}

case 7:{

vo=strtof(substr,NULL);

break;

}

case 6:{

adj=strtof(substr,NULL);

break;

}

case 8:{

zdv=strtof(substr,NULL);

break;

}

case 9:{

zdr=strtof(substr,NULL);

break;

}

}

cnt++;

substr=strtok(NULL,",");

}

Us\_stock b(dt,sy,op,hi,lo,cl,vo,zdv,zdr,adj);

a=b;

}

void fun2(Us\_stock &a,char \*buff){

string dt;string sy;float op=0.0;float hi=0.0;

float lo=0.0;float cl=0.0;float vo=0.0;float zdv=0.0;

float zdr=0.0;float adj=0.0;

int cnt=0;

char \*substr=strtok(buff,",");

while(substr){

switch(cnt)

{

case 0:{

dt=substr;break;

}

case 1:{

sy=substr;

break;

}

case 2:{

zdr=strtof(substr,NULL);

break;

}

}

cnt++;

substr=strtok(NULL,",");

}

Us\_stock b(dt,sy,op,hi,lo,cl,vo,zdv,zdr,adj);

a=b;

}

void mergeFiles(char \*outfile,const int runsize,const int runnums,int status)//

{

FILE \*in[runnums];

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

{

char filename[10];

snprintf(filename,sizeof (filename),"%d",i);

in[i]=fileopen(filename,"r");

}

FILE \*out=fileopen(outfile,"w");

MaxheapNode \*harr=new MaxheapNode[runnums]; //结构体模板

int i=0;

char buf[200];

for(i=0;i<runnums;i++)

{

if(fgets(buf,200,in[i])==NULL) break;

fun1(harr[i].element,buf);

harr[i].index=i;

}

Maxheap hp(harr,i,status);

cout<<hp.getMax().element.getzdr()<<endl;

int cnt=0;//jishuqi

while(cnt!=i){

MaxheapNode root=hp.getMax();

string dt=root.element.getDate();

string sy=root.element.getSym();

const char \*dt1=dt.c\_str(); const char \*sy1=sy.c\_str();

fprintf(out,"%s,%s,%f%,\n",dt1,sy1,root.element.getzdr());

char \*flag=fgets(buf,200,in[root.index]);

fun1(root.element,buf);

if(flag==NULL)

{root.element=Minstock;cnt++;}

hp.replaceroot(root);

}

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

fclose(in[i]);

fclose(out);

}

void creatruns(char \*inputfile,int runsize,int runnums,int status)

{

FILE \*in=fileopen(inputfile,"r");

FILE \*out[runnums];//

char filename[10];

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

{

snprintf(filename,sizeof (filename),"%d",i);

out[i]=fileopen(filename,"w");

}

bool more\_input=true;

Us\_stock \*arr=new Us\_stock[runsize];

int next\_output\_file=0;

char buf[200];

while(more\_input)

{

int i;

for ( i=0;i<runsize;i++) {

if(!fgets(buf,200,in)) //read till EOF

{

more\_input=false;

break;

}

//根据文件名进行两个数据集的构造.

//if us\_stock

//cout<<buf<<endl;

if(status==1 || status==2) fun1(arr[i],buf);

if(status==3) fun2(arr[i],buf);

//cout<<arr[i].getDate()<<endl;

//else

//fun2(&arr[i],buf);

}

//mergesort(arr,0,runsize-1);

if(status==1)

sort(arr,arr+i,cmp1);

if(status==2)

sort(arr,arr+i,cmp2);

if(status==3)

sort(arr,arr+i,cmp3);

//if(i==291) cout<<arr[i-1].getDate()<<endl;

//cout<<next\_output\_file<<more\_input<<endl;

//适应两个数据集

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

//fwrite((char \*)&arr[i],sizeof(arr[i]),1,out[next\_output\_file]); //二进制形式写入

const char \*dt=arr[j].getDate().c\_str();const char \*sy=arr[j].getSym().c\_str();

fprintf(out[next\_output\_file],"%s,%s,%f,%f,%f,%f,%f,%f,%f,%f\n",dt,sy,arr[j].getOpen(),arr[j].getHigh(),arr[j].getLow(),arr[j].getClose(),arr[j].getAdj(),arr[j].getVolume(),arr[j].getzdv(),arr[j].getzdr());

dt=NULL;sy=NULL;

}

next\_output\_file++;

}

//delete arr;

for(int i=0;i<next\_output\_file;i++)

{

fclose(out[i]);

}

fclose(in);

}

void externalsort(char \*openfile,char \*outfile,int runsize,int runnums,int status)

{

creatruns(openfile,runsize,runnums,status);

mergeFiles(outfile,runsize,runnums,status);

}

Topk：

void strsplit(string a, int &y,int &m){

int tmp1=a.find\_first\_of('/');int tmp2=a.find\_last\_of('/');

y=atoi(a.substr(0,tmp1).c\_str());m=atoi(a.substr(tmp1+1,tmp2-tmp1-1).c\_str());

}

bool strcmp1(char\* a, char\* b){

if(strlen(a)!=strlen(b)) return false;

for(int i=0;i<strlen(a);i++)

{

if(a[i]!=b[i]) {

return false;

}

}

return true;

}

void creatruns1(char \*inputfile) //

{

FILE \*in=fileopen(inputfile,"r");

char buf[200];

char \*substr=NULL;int tot=0;char lastfina[20];bool flag=false;

char \*dt;char \*sy;float zdr;

while(fgets(buf,200,in))

{

int cnt=0;

substr=strtok(buf,",");

while(substr)

{

switch(cnt){

case 0:{

dt=substr;

break;

}

case 1:{

sy=substr;

break;

}

case 2:{

zdr=atof(substr);

break;

}

}

cnt++;

substr=strtok(NULL,",");

}

string tmp=dt; char filename[20];

int year;int month;

strsplit(tmp,year,month);

snprintf(filename,sizeof(filename),"%d-%d.csv",year,month);

if(!flag) {

flag=true;

int i;

for(i=0;i<strlen(filename);i++)

lastfina[i]=filename[i];

lastfina[i]='\0';

}

if(strcmp1(lastfina,filename)==false) {

tot++;

int i;

for(i=0;i<strlen(filename);i++) lastfina[i]=filename[i];

lastfina[i]='\0';

}

FILE \*out=fileopen(filename,"a");

fprintf(out,"%s,%s,%f%,\n",dt,sy,zdr);

fclose(out);

}

}

void topkexternalsort(char \*openfile,char \*outfile){

//externalsort("C:/Users/86158/Desktop/程设2大作业/data/test\_us\_stock\_daily1.csv",

// "C:/Users/86158/Desktop/程设2大作业/data/test\_us\_ex\_sort2.csv",1000,216,2);

//creatruns1(openfile);

int y,m,k;

cin>>y>>m>>k;

char filename[20];char outfilename1[20];char outfilename2[20];

snprintf(filename,sizeof(filename),"%d-%d.csv",y,m);

//cout<<filename;

snprintf(outfilename1,sizeof(outfilename1),"%d-%dtop%d1.csv",y,m,k);

snprintf(outfilename2,sizeof(outfilename2),"%d-%dtop%d2.csv",y,m,k);

externalsort(filename,outfilename1,100,10,3);

FILE \*in=fopen(outfilename1,"r");

FILE \*out=fopen(outfilename2,"w");

char buf[100];

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

{

fgets(buf,100,in);

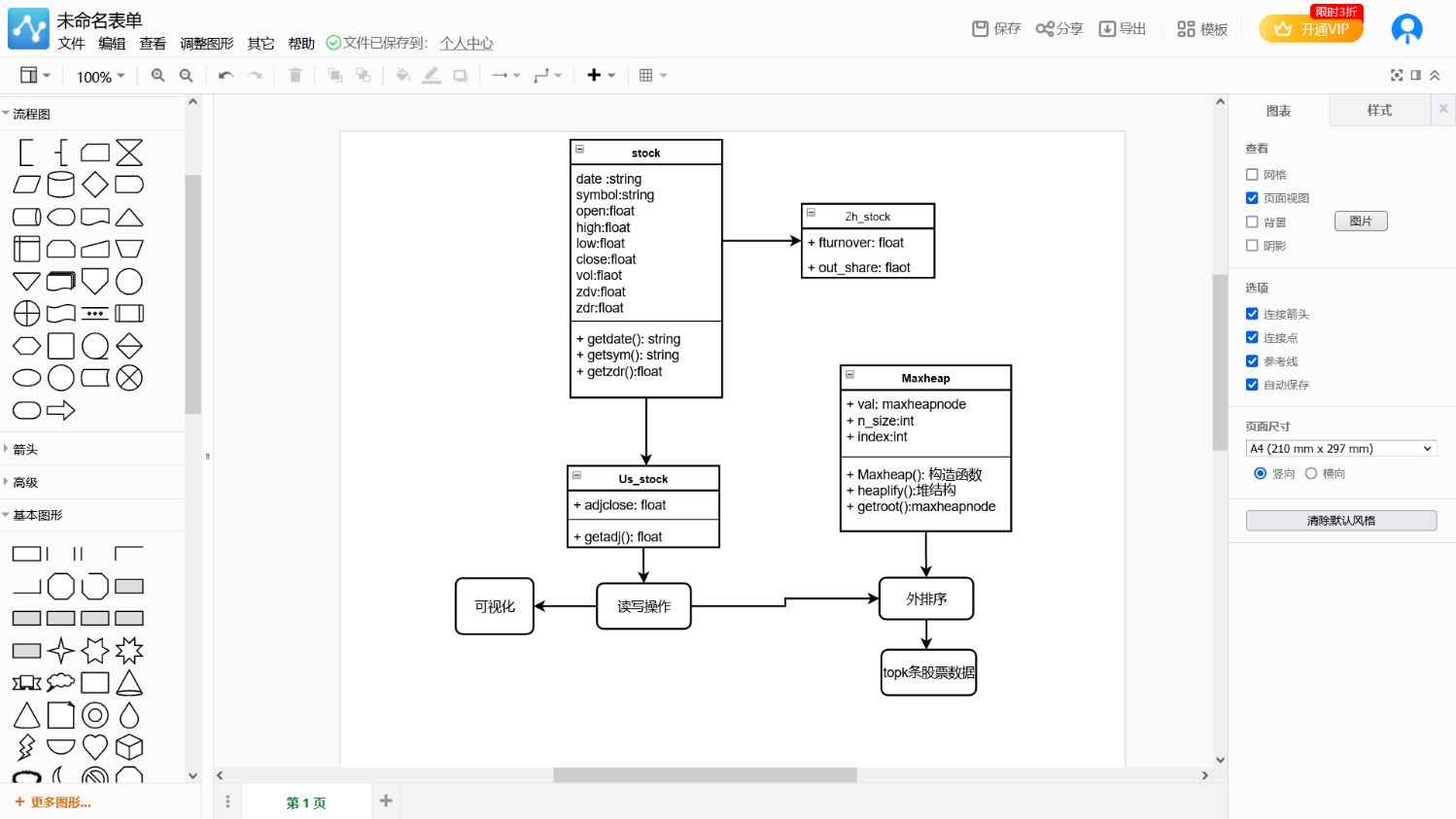
//fputs(buf,out);

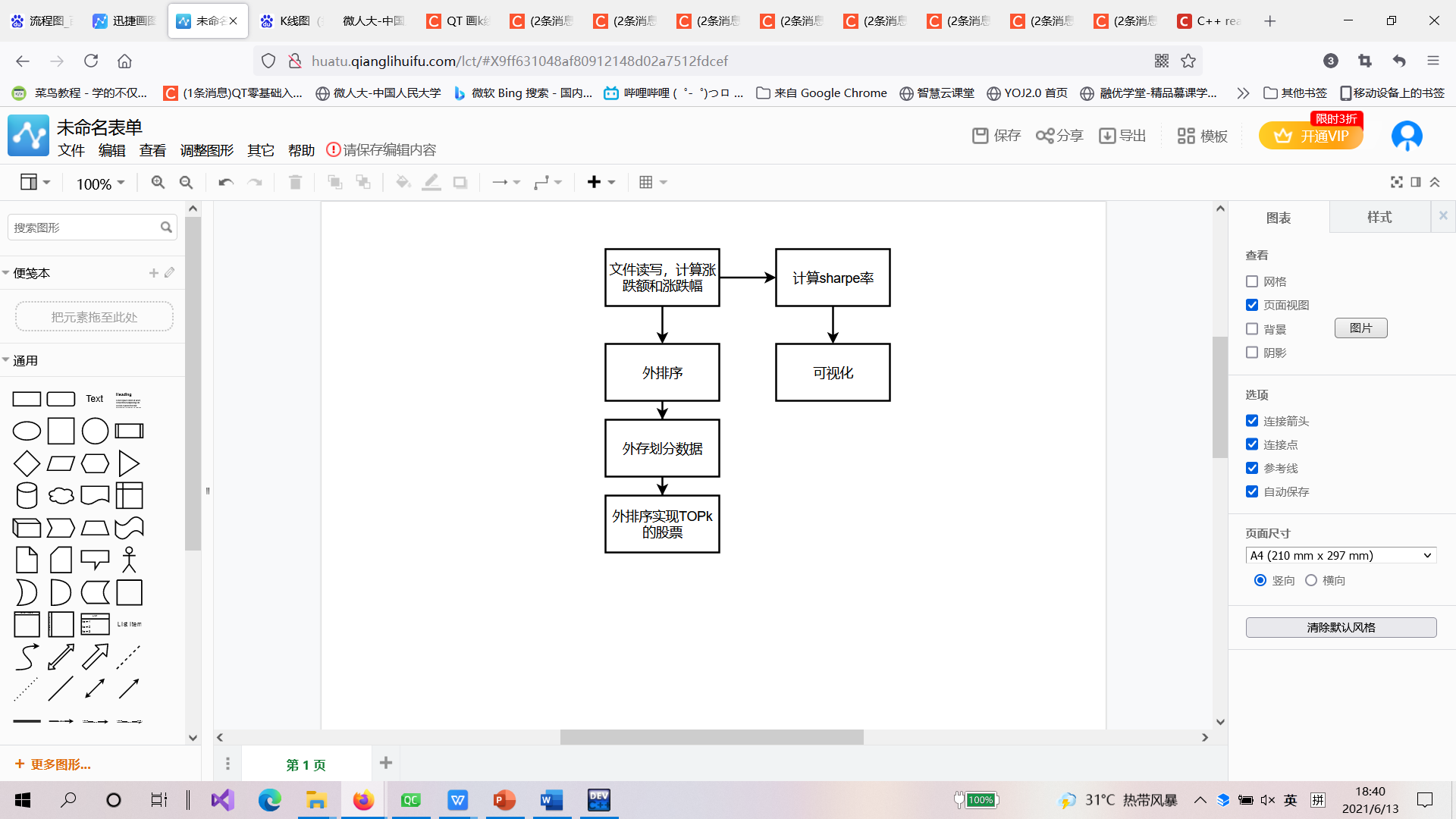
fprintf(out,"%s",buf);

}

}

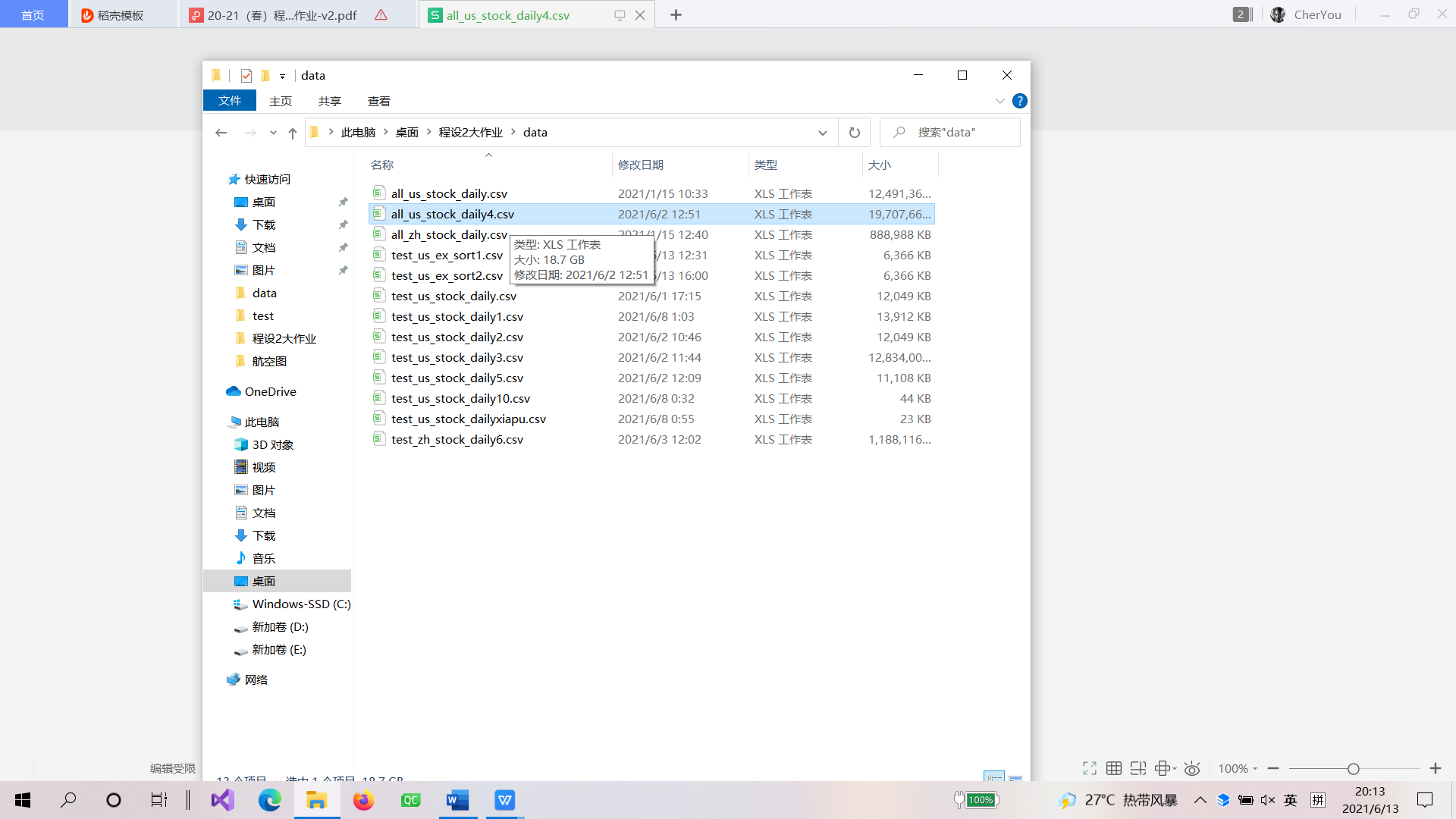
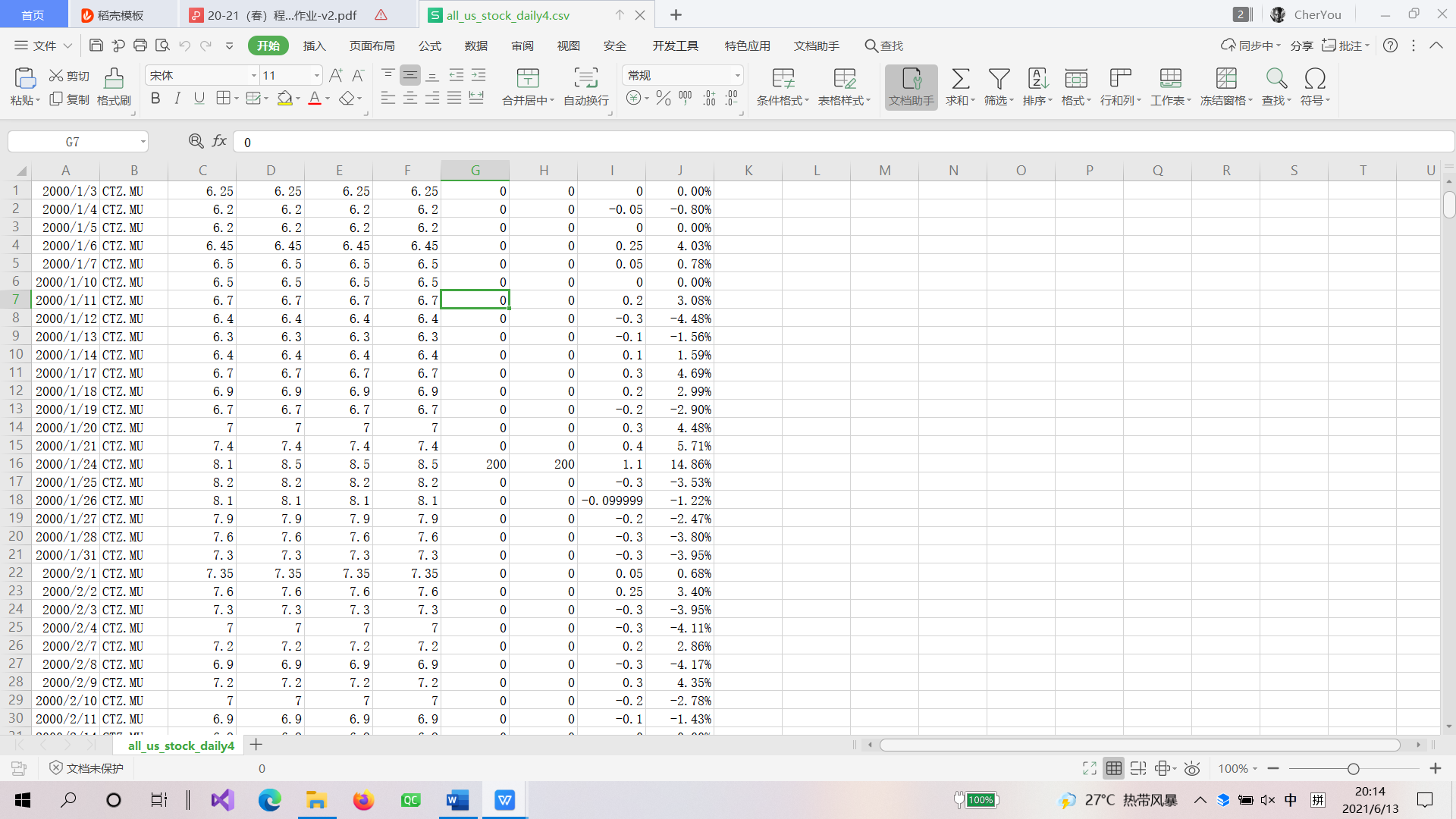
可视化：

1. 调试分析
2. UML类图：
3. 系统流程图：

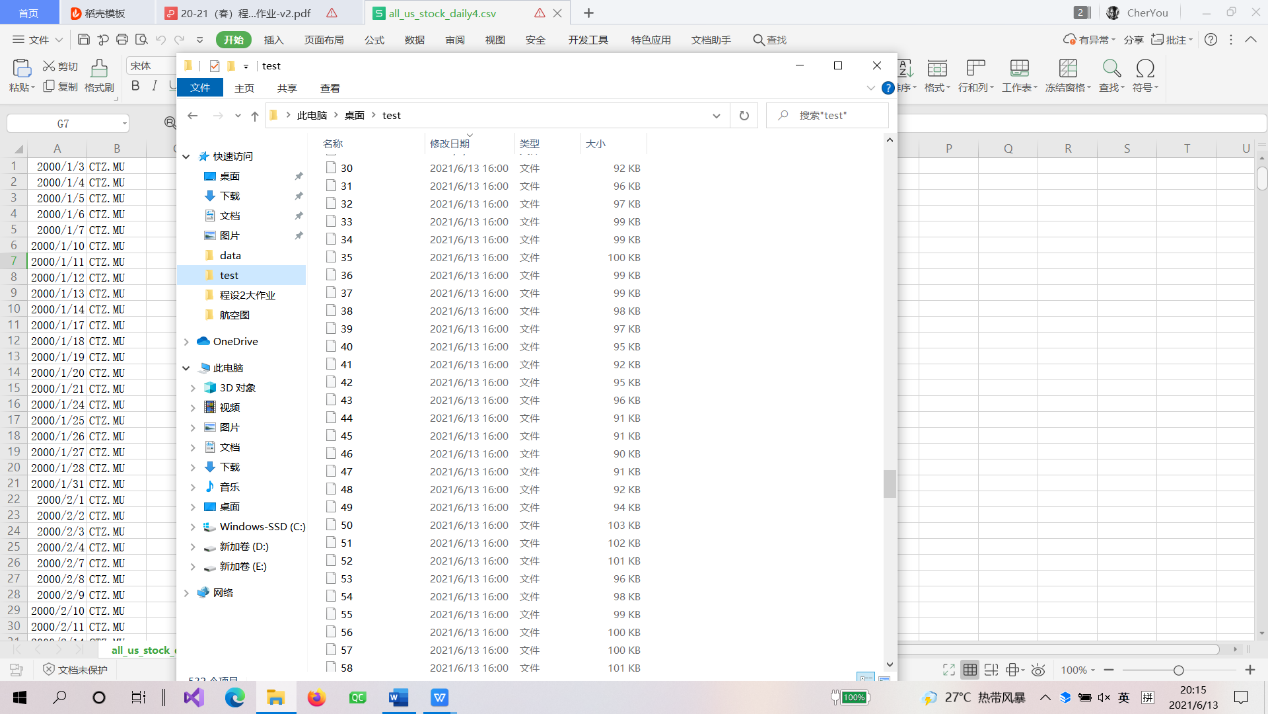
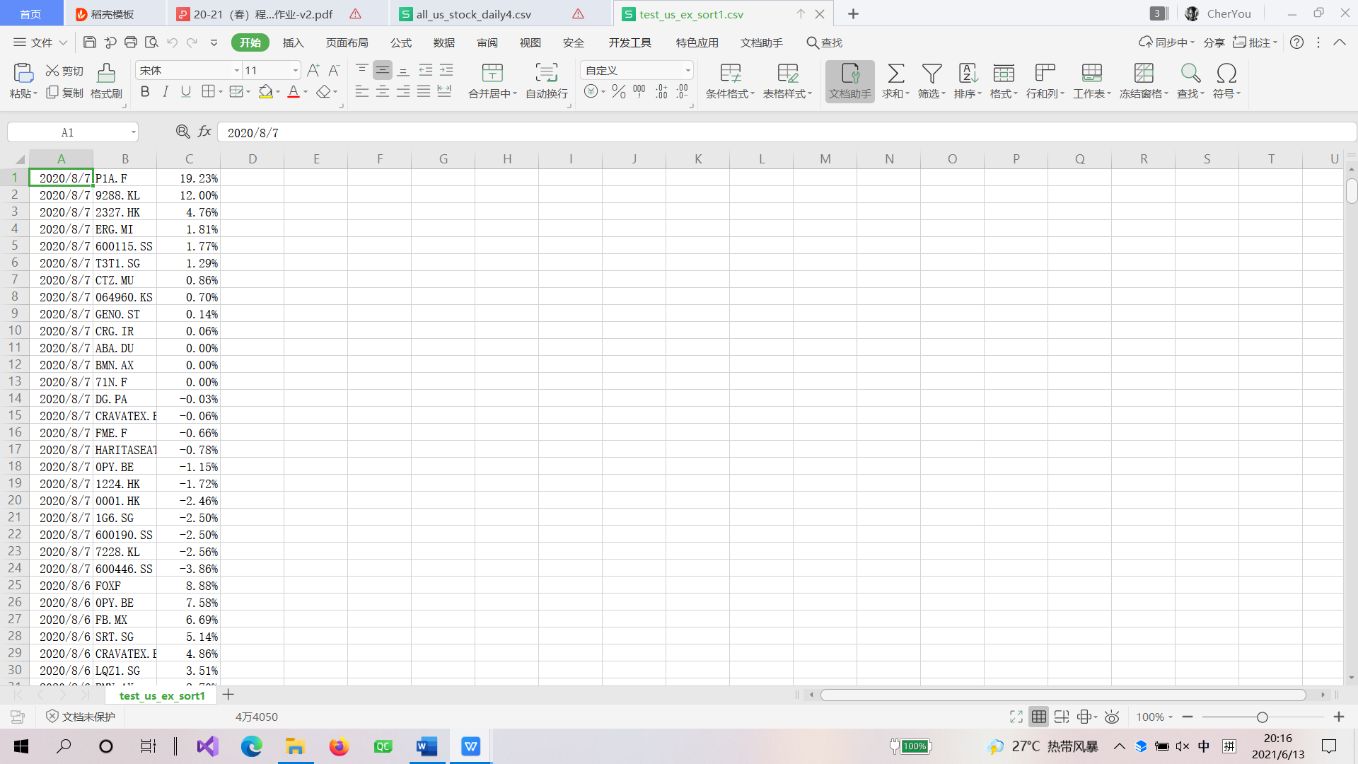


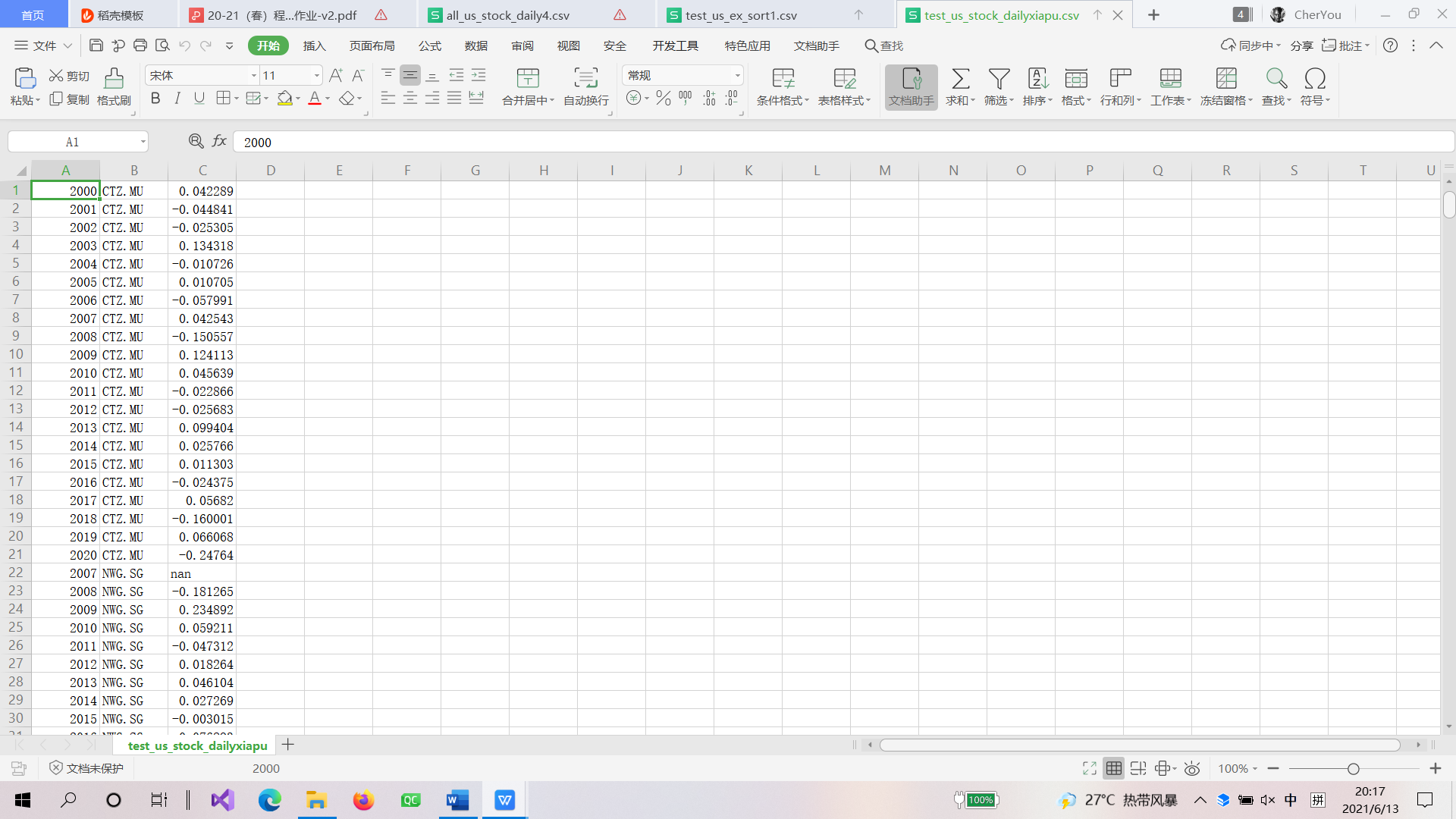
1. 实验结果：

功能1 实现涨跌额涨跌幅的写入：

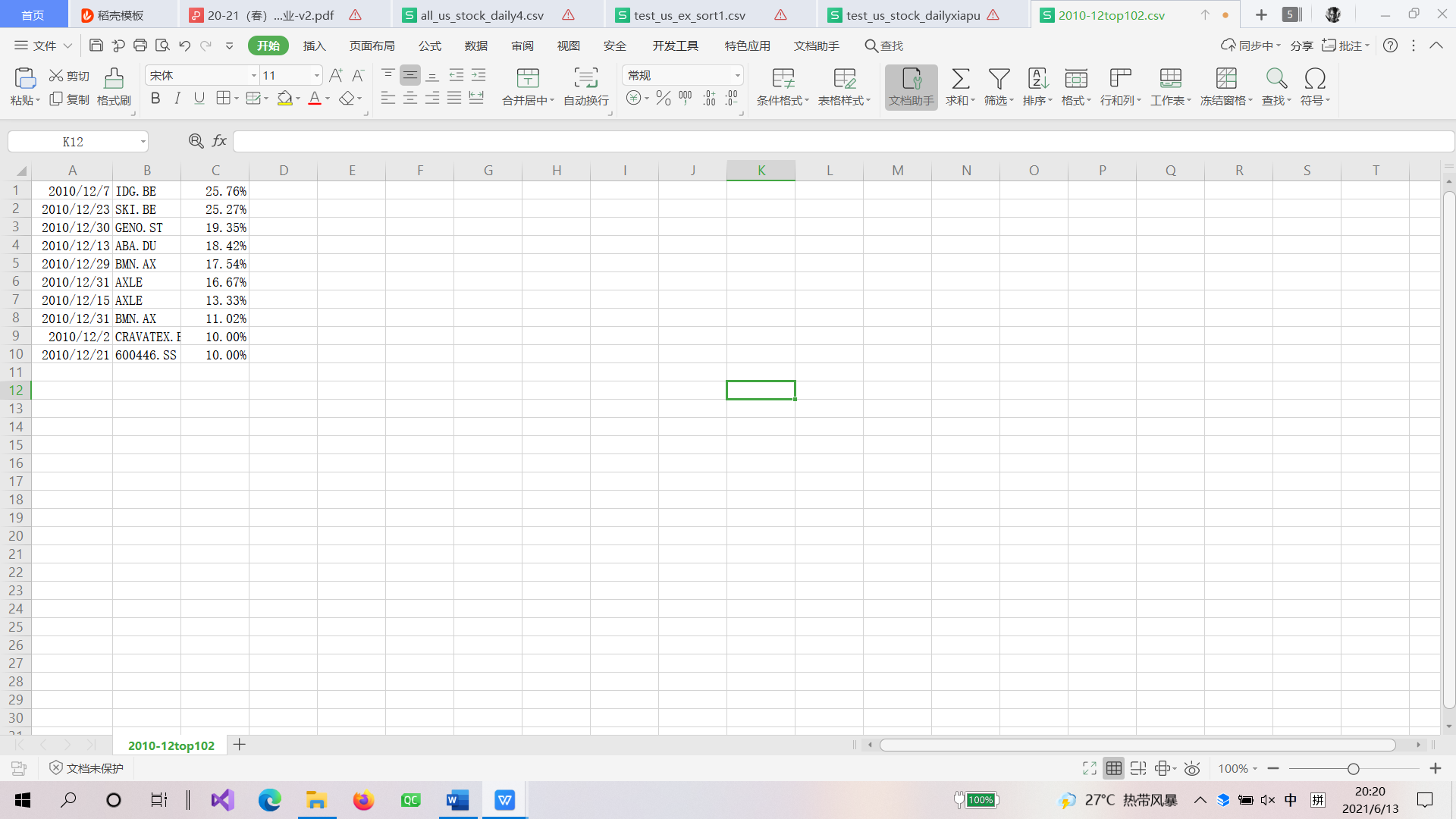
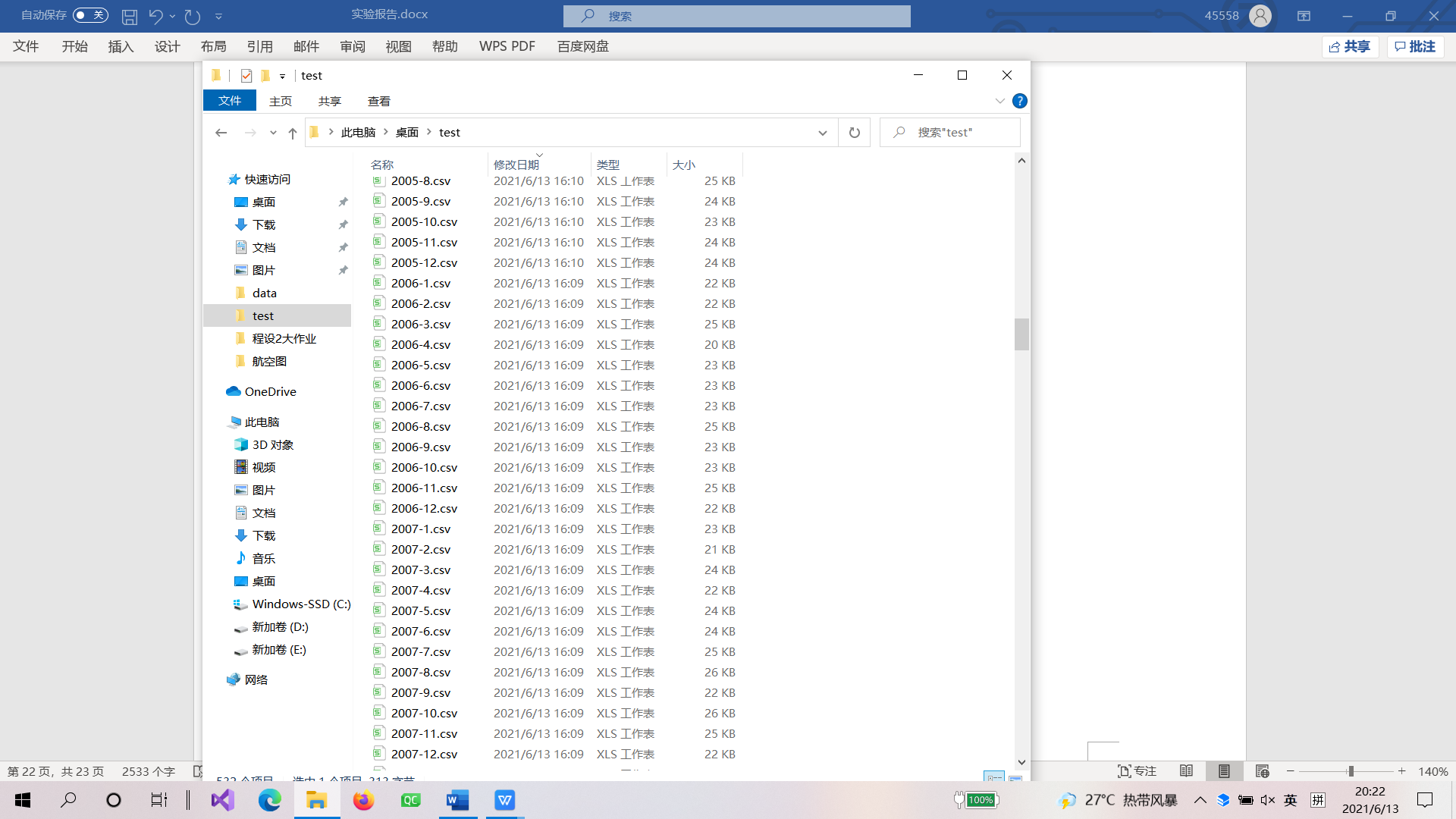
写入后文件大小位18.9G

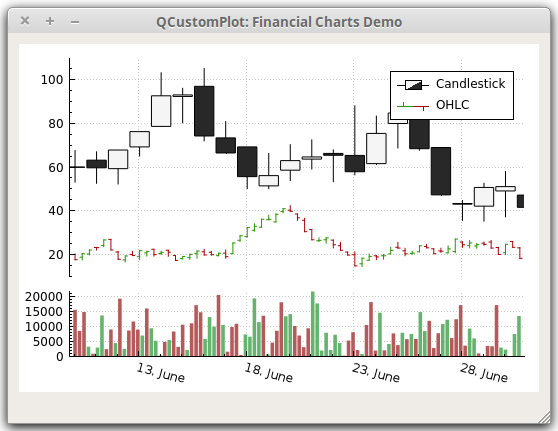
功能二实现外排序：

左图为外排序文件，右图为外排序结果：

功能3 ：计算sharpe率：

功能4 ：按时间分割数据并寻找单月涨跌额最大的TOP k条股票

2010 12 top 10 ：

功能5 可视化：

1. 拓展功能：无
2. 附录

两个工程文件：

Stock111.pro

Stock2.pro