# The first report

These days, I tried to take a way how to make up my codes programming ago. We are supposed to make a class to Integrate these pieces of codes.

I found most of previous codes are allowed to use in the class, I only need change their parameters, return value or type.

First, I created a class named mySet and it has two data in privated, name object name and element set elements. They are both of string type.

Then, I opened previous cpp documents, replace their parameter type with mySet. And replace the return type with string which is their element string.

About what I had finish a set class it has 4 constructors a copy constructor and a destructor but the destructor has not any operation now, also 10 set operations and I use some const and private to make sure some security of data, vector is used as well.

There are some of code and They are distributed in different files :

Class:

*class mySet {*

*private:*

*char name;//Set name no define*

*string element;//储存元素字符*

*vector<int> ele;*

*string pick\_ele(const string &s);*

*public:*

*mySet();*

*mySet(const string s); //s={a,b,c}*

*mySet(const int a[],int n);//Initialize object by Array n is lenth of array*

*mySet(mySet &set);*

*~mySet();*

*string getEle();*

*void setEle(const string s);//带逗号或者纯元素进行initalize*

*void myPrint();*

*int myFind\_ele(const string &s);//return the first element index of s*

*int getEleNum();*

*bool Belong();//输入一个数判断是否属于set*

*string sumSet(mySet set);//合集*

*string subSet(mySet set);//subSet子集*

*bool isEmpty();*

*string DiffSet(mySet set);//差集*

*int subNum();//子集个数*

*bool isSubSet(mySet set); //是否为mySet的子集*

*bool isRealSub(mySet set);//是否为mySet的真子集*

*string oppsDiffSet(mySet set);//对称差集*

*bool isEqual(mySet set); //判断是否相等*

*};*

## Union

*string mySet::sumSet(mySet set)*

*{*

*int sumTemp[max\_len];*

*string sum;//add result*

*memset(sumTemp,0,sizeof(sumTemp));*

*for(int i=0; i<max(getEleNum(),set.getEleNum()); i++) {*

*sumTemp[element[i]-'0']++;*

*sumTemp[set.element[i]-'0']++;//利用桶排序思想除去重复*

*//cout<<i<<" ";*

*}*

*for(int i=0; i<max\_len; i++) {*

*if(sumTemp[i]!=0){*

*sum+=i+'0';*

*}*

*//给sum赋值*

*}*

*//cout<<sum.size()<<endl;*

*return sum;//return result*

*}*

## Belong

bool mySet::Belong()

{

char ch;

cout<<"enter a char :";

cin>>ch;

if(element.find(ch)!=string::npos) {

cout<<ch<<" is belong the set"<<endl;

return true;

}

cout<<"ch is not belong the set"<<endl;

return false;

}

## Sub

string mySet:: subSet(mySet set)//交集 返回交集set

{

int aTemp[max\_len],bTemp[max\_len];

string sub;

memset(aTemp,0,sizeof(aTemp));

memset(bTemp,0,sizeof(bTemp));//initialize to 0

for(int i=0;i<max(getEleNum(),set.getEleNum());i++)

{

aTemp[element[i]-'0']++;//upper one by one if across

bTemp[set.element[i]-'0']++;

}

for(int i=0;i<max\_len;i++){

if(aTemp[i]!=0&&bTemp[i]!=0)

sub+=i+'0';

}

return sub;

}

## isEmpty

bool mySet::isEmpty()

{

return element.size();

}

## DiffSet

string mySet::DiffSet(mySet set)

{

int aTemp[max\_len],bTemp[max\_len];

memset(aTemp,0,sizeof(aTemp));

memset(bTemp,0,sizeof(bTemp));//initalize value with 0

string diff;

for(int i=0; i<max(getEleNum(),set.getEleNum()); i++)

{

aTemp[element[i]-'0']++;

bTemp[set.element[i]-'0']++;//利用桶排序思想除去重复

//cout<<i<<" ";

}

for(int i=0;i<'z';i++){

if(aTemp[i]!=0&&bTemp[i]==0)

diff+=i+'0';

}

//cout<<diff.size()<<endl;

return diff;

}

## Subset num

int mySet::subNum()

{//子集个数

return 1<<element.size();

}

## isSubset

bool mySet::isSubSet(mySet set)

{

int aTemp[max\_len],bTemp[max\_len];

memset(aTemp,0,sizeof(aTemp));

memset(bTemp,0,sizeof(bTemp));

for(int i=0;i<max((\*this).getEleNum(),set.getEleNum());i++)

{//先转换

aTemp[element[i]-'0']++;

if(set.element[i]>0&&set.element[i]<106)

bTemp[set.element[i]-'0']++;

}

for(int i=0;i<max(getEleNum(),set.getEleNum());i++){

if(aTemp[i]==0&&bTemp[i]!=0){

// cout<<"is not subSet"<<endl;

return false;

}

}

// cout<<"is subSet"<<endl;

return true;

}

## isRealSubset

bool mySet::isRealSub(mySet set)

{

if(isSubSet(set)&&set.getEleNum()<element.size())

return true;

return false;

}

## oppsite diffSet

string mySet::oppsDiffSet(mySet set)

{

mySet atemp((\*this).DiffSet(set));

mySet btemp(set.DiffSet(\*this)) ;

return atemp.sumSet(btemp);

}

## equal

bool mySet::isEqual(mySet set)

{

if(getEleNum()==set.getEleNum()&&isSubSet(set))

return true;

return false;

}