template<class K,class E>

class pair

{

K first;

E second;

};

template<class K,class E>

class pairNode

{

public:

pair<const K,E> element;

pairNode<K,E>\* next;

friend ostream& operator<<(ostream& out,const pairNode<K,E> Pair)

{

cout<<"Key:"<<element.first<<"\tValue:"<<element.second<<endl;

}

};

template<class K,class E>

class sortedArraylist

{

public:

sortedArraylist(int intialCapacity=10)

{

if(intialCapacity<0)

{

cerr<<"初始化非法！"<<endl;

exit(1);

}

element=new pairNode<K,E> [intialCapacity];

arrayLength=intialCapacity;

}

~sortedArraylist()

{

delete [] element;

}

bool empty() const;

{

if(dSize==0) return true;

else return false;

}

int size() const;

{

return dSize;

}

int find(const K& theKey) const;

{//返回匹配数对在数组中的索引，使用二分查找

int mid=dSize/2;

int left=0;

int right=dSize-1;

while(1)

{

if(element[mid].first==theKey) return mid;

if(element[mid].first>theKey)

{

mid=(mid-1+left)/2;

}

if(element[mid].first<theKey)

{

mid=(mid+1+right)/2;

}

if(mid==left||mid==right&&element[mid].first!=theKey)

break;

}

return -1;

}

void erase(const K& theKey)

{//删除第一个关键字为theKey的元素

int index=find(theKey);

if(index==-1) return;

else

{

for(int i=index;i<dSize;i++)

{

element[i]=element[i+1];

}

element[dSize-1].~pairNode<K,E>(); //把最后的元素析构

dSize--;

}

}

void insert(const pair<const K,E>& thePair)

{//向字典中保证有序地插入thePair,如果有重复关键字则覆盖

int cursor;

for(cursor=0;cursor<dSize;cursor++)

{

if(thePair.first==element[cursor].first)

{//重复

element[cursor].second=thePair.second;

}

if(thePair.first>element[cursor].first)

{//无匹配数对，插入新空间

if(dSize==arrayLength)

{//数组空间已满

pairNode<K,E> \*p=new T[2\*arrayLength];

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

{

p[i]=element[i];

}

delete [] element;

element=p;

arrayLength\*=2;

}

for(int i=dSize;i>cursor-1;i--) //元素后移

{ element[i+1]=element[i]; }

element[cursor]=thePair; //赋值

dSize++;

}

}

}

void output() const

{

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

{

cout<<element[i];

}

}

private:

pairNode<K,E>\* element; //存放数对的数组指针

int dSize; //字典中数对的数量

int arrayLength; //字典容量

};

void hashTable<K,E>:: erase(const K& theKey)

{//删除关键字为theKey的数对，如果没有就什么也不做

//需要把它后面的某些元素一并向前移动，（直到某个key%D==索引或者遇到空）

int b=search(theKey);

if(table[b]==NULL||table[b]->first!=theKey)

{//即没有匹配，什么也不做

return ;

}

int pos; //这个位置之前的元素都要前移一位

for(pos=b;pos<divisor+b;pos++)

{//下面是找到一个合适的终止移动位置

if( (table[pos]->first)%divisor == pos || table[pos]==NULL)

break;

}

//如果pos在(b,divisor)内

if(pos>=b)

{

for(int j=b;j<pos;j++)

table[j]=table[j+1];

}

else{//如果pos在(0,b)内

for(int j=b;j<pos;j++)

table[j]=table[j+1];

table[divisor]=table[0];

for(int j=0;j<pos;j++)

table[j]=table[j+1];

}

table[(pos-1)%divisor]=NULL;

}