**数据结构上机实习报告**

题目：1.3 集合的并、交和差运算的程序

班级：1403013 学号：14030130061 姓名：刘豪 完成日期：2015.12.04

**一、需求分析**

1．集合的元素限制为小写字符[‘a’..’z’]。

2．演示程序以客户和计算机的对话方式执行

**二、概要设计**

为了实现上述程序功能，我写了一个集合的类。

基本操作：

Set()

初始化一个空集合对象;

Set(int \_sz)

通过输入元素个数随机生成一个约瑟夫环;

~Set()

析构函数删除对象;

del()

模拟约瑟夫环的节点删除;

Set& operator = (const Set &A)

重载等于符号方便深拷贝;

bool find(const int &val)

集合的查找函数;

void sort()

集合的排序;

void quick\_sort(Node \*L,Node \*R)

集合的快排操作;

void add(int val)

向集合中添加元素;

void print()

打印集合;

Set inter(Set &A)

求交集;

Set unio(Set &A)

求并集;

Set comple(Set &A)

求差集;

**三、详细设计**

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<ctime>

using namespace std;

class Set{

struct Node{

int val;

Node \*nex;

};

private:

int \_sz;

Node \*head;

public:

Set();

Set(int \_sz);

~Set();

Set& operator = (const Set &A);

bool find(const int &val);

void sort();

void quick\_sort(Node \*L,Node \*R);

Set inter(Set &A);

Set unio(Set &A);

Set comple(Set &A);

void add(int val);

void print();

};

Set::Set(){

head=NULL;

\_sz=0;

}

Set& Set::operator = (const Set &A){

Node \*tA=A.head;

while(tA!=NULL){

(\*this).add(tA->val);

tA=tA->nex;

}

return \*this;

}

bool Set::find(const int &val){

Node \*tmp=head;

while(tmp!=NULL){

if((tmp->val)==val)

break;

tmp=tmp->nex;

}

return tmp!=NULL;

}

void Set::add(int val){

\_sz++;

Node \*pnew=new Node;

pnew->val=val;

pnew->nex=NULL;

if(head==NULL){

head=pnew;

}

else{

Node \*tmp=head;

while((tmp->nex)!=NULL){

tmp=tmp->nex;

}

tmp->nex=pnew;

}

}

Set::Set(int sz){

\_sz=0;

head=NULL;

while(\_sz<sz){

int tmp=rand()%26;

while((\*this).find(tmp)){

tmp=rand()%26;

}

(\*this).add(tmp);

}

(\*this).sort();

}

Set::~Set(){

Node \*tmp;

while(\_sz--){

tmp=head;

head=head->nex;

delete tmp;

}

}

void Set::sort(){

quick\_sort(head,NULL);

}

void Set::quick\_sort(Node \*L,Node \*R){

if(L==R)

return;

Node \*p=L;

Node \*q=L->nex;

int key=L->val;

while(q!=R){

if((q->val)<key){

p=p->nex;

swap(p->val,q->val);

}

q=q->nex;

}

swap(p->val,L->val);

if(L!=R){

quick\_sort(L,p);

quick\_sort(p->nex,R);

}

}

Set Set::inter(Set &A){

A.sort();

(\*this).sort();

Set tmp;

Node \*tA=A.head;

Node \*tB=this->head;

while(tA!=NULL&&tB!=NULL){

if((tA->val)==tB->val){

tmp.add(tA->val);

tA=tA->nex;

tB=tB->nex;

}

else if((tA->val)<tB->val){

tmp.add(tA->val);

tA=tA->nex;

}

else{

tmp.add(tB->val);

tB=tB->nex;

}

}

while(tA!=NULL){

tmp.add(tA->val);

tA=tA->nex;

}

while(tB!=NULL){

tmp.add(tB->val);

tB=tB->nex;

}

return tmp;

}

Set Set::unio(Set &A){

A.sort();

(\*this).sort();

Set tmp;

Node \*tA=A.head;

Node \*tB=this->head;

while(tA!=NULL&&tB!=NULL){

if((tA->val)==tB->val){

tmp.add(tA->val);

tA=tA->nex;

tB=tB->nex;

}

else if((tA->val)<tB->val)

tA=tA->nex;

else

tB=tB->nex;

}

return tmp;

}

Set Set::comple(Set &A){

A.sort();

(\*this).sort();

Set tmp;

Node \*tA=A.head;

Node \*tB=this->head;

while(tB!=NULL){

if(!A.find(tB->val))

tmp.add(tB->val);

tB=tB->nex;

}

return tmp;

}

void Set::print(){

Node \*tmp=head;

while(tmp!=NULL){

printf("%c ",(tmp->val)+'a');

tmp=tmp->nex;

}

printf("\n");

}

int main(){

srand((unsigned)time(0));

int t1,t2;

printf("请分别输入两个集合的元素个数:\n");

scanf("%d %d",&t1,&t2);

Set a(t1);

Set b(t2);

printf("集合A：");

a.print();

printf("集合B: ");

b.print();

Set c=a.inter(b);

printf("交集: ");

c.print();

Set d=a.unio(b);

printf("并集：");

d.print();

Set e=a.comple(b);

printf("差集：");

e.print();

return 0;

}

**四、调试分析**

1.可通过输入元素个数随机生成集合。

2.通过快速排序对集合进行排序。

3.通过生成数字来代替字母。

**五、用户手册**

本程序的运行环境为WIN10，执行文件为:集合操作.exe

**六、测试结果**

