山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验题目：有序链表操作 | | | |
| 实验学时：4 | | 实验日期：2021年11月2日星期二 | |
| 实验目的：  1、 掌握有序链表的基本操作：插入、删除、查找。  2、 掌握链表遍历器的使用方法。 | | | |
| 硬件环境：  lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz | | | |
| 软件环境：  CLion 2020.3.3 x64 | | | |
| 实验步骤与内容：  1、 输入n个不为零的整数作为节点元素值，遇到0代表输入结束（不创建元素值为0的节点），创建有序链表。输出整个链表。  2、 输入一个整数，将该数插入到有有序链表相应位置。输出整个链表。  3、 输入一个整数，在链表中进行搜索，输出其在链表中的第一个出现的位置。如果不存在输出0。  4、 再一次输入一个整数，在链表中进行搜索，输出其在链表中的第一个出现的位置。如果不存在输出0。  5、 再一次输入n个不为零的整数作为节点元素值，遇到0代表输入结束（不创建元素值为0的节点），创建一个新的有序链表。输出整个链表。  6、 使用链表遍历器实现上面两个有序链表的合并，输出合并后的有序链表。  7、 提示：注意单节点链表的测试。 | | | |
| 结论分析与体会：  对于链表的插入排序算法，总结如下几点：  1.考虑链表是否为空，如果为空则直接在头节点插入数据  2.如果不为空：  设置p节点与pre节点，当p指向的元素比要插入的元素小而且p的后面还有元素时，p后移，pre指向前一个；  该循环停下，可能是p指向的元素变大于了插入元素，此时直接插入  还有可能是指向了后面的元素是空值，这时有如下几种情况：  此事的情况有，一开始链表只有一个元素的时候、有多个元素的时候；  如果p指向的元素小，不过有几个元素，一定是向后面插入。  如果p指向的元素大于插入元素，  只有一个元素时，利用p==头节点判断，如果p是头节点，则让元素插入到头节点；如果p不是头节点，则该元素是倒数第二大元素，此时按照正常pre方法插入。  如果是p指向的元素小于插入元素停下时，用pre方法判断。但pre方法不适用于头节点的情况，因此要先判断是不是头节点，是头节点直接变头节点，不是的话再用pre方法。  void insert(int num){  if (firstNode==NULL){ #include <iostream>  using namespace std;  struct node{  int ele;  node \* next;  node(int t\_ele,node\* t\_next){  ele = t\_ele;  next = t\_next;  }  };  class chain{  public:  node \* firstNode;  int listSize;  chain():firstNode(NULL),listSize(0){};  bool isEmpty(){return listSize==0;}  void insert(int num){  if (firstNode==NULL){  firstNode = new node(num,NULL);  }else{  node \* p = firstNode;  node \* pre = p;  while (p->ele<num && p->next!=NULL){  pre = p;  p = p->next;  }  if (p->next==NULL){  if (p->ele<num)p->next = new node(num,NULL);  else if (p==firstNode){  node \* temp = new node(num,p);  firstNode = temp;  } else{  node \* temp = new node(num,p);  pre->next = temp;  }  } else if (p==firstNode){  node \* temp = new node(num,p);  firstNode = temp;  } else if (p->next!=NULL){  node \* temp = new node(num,p);  pre->next = temp;  }  }  listSize++;  }  int search(int num){  int result = 0;  node \* p = firstNode;  while (p->ele!=num){  if (p->next==NULL)return 0;  p = p->next;  result++;  }  return result+1;  }  void givRes(){  node \* p = firstNode;  while (p->next != NULL){  cout << p->ele<<",";  p = p->next;  }  cout << p->ele<<endl;  }  };  chain together(chain chain1,chain chain2){  chain result;  node \*p1 = chain1.firstNode;  node \*p2 = chain2.firstNode;  while (p1!=NULL){  result.insert(p1->ele);  p1 = p1->next;  }  while (p2!=NULL){  result.insert(p2->ele);  p2 = p2->next;  }  return result;  }  int main() {  int enter;  chain theChain;  cout << "Input1"<<endl;  cin >> enter;  while (enter!=0){  theChain.insert(enter);  cin >> enter;  }  cout << "Output1"<<endl;  theChain.givRes();  cout << "Input2" << endl;  cin >> enter;  theChain.insert(enter);  cout << "Output2"<<endl;  theChain.givRes();  cout << "Input3"<<endl;  cin >> enter;  cout << "Output3"<<endl;  cout << theChain.search(enter)<<endl;  cout << "Input4" << endl;  cin >> enter;  cout << "Output4"<<endl;  cout << theChain.search(enter)<<endl;  chain theChain1;  cout << "Input5"<<endl;  cin >> enter;  while (enter!=0){  theChain1.insert(enter);  cin >> enter;  }  cout << "Output5"<<endl;  theChain1.givRes();  chain togChain = together(theChain,theChain1);  togChain.givRes();  cout << "End0"<<endl;  return 0;  }  firstNode = new node(num,NULL);  }else{  node \* p = firstNode;  node \* pre = p;  while (p->ele<num && p->next!=NULL){  pre = p;  p = p->next; #include <iostream>  using namespace std;  struct node{  int ele;  node \* next;  node(int t\_ele,node\* t\_next){  ele = t\_ele;  next = t\_next;  }  };  class chain{  public:  node \* firstNode;  int listSize;  chain():firstNode(NULL),listSize(0){};  bool isEmpty(){return listSize==0;}  void insert(int num){  if (firstNode==NULL){  firstNode = new node(num,NULL);  }else{  node \* p = firstNode;  node \* pre = p;  while (p->ele<num && p->next!=NULL){  pre = p;  p = p->next;  }  if (p->next==NULL){  if (p->ele<num)p->next = new node(num,NULL);  else if (p==firstNode){  node \* temp = new node(num,p);  firstNode = temp;  } else{  node \* temp = new node(num,p);  pre->next = temp;  }  } else if (p==firstNode){  node \* temp = new node(num,p);  firstNode = temp;  } else if (p->next!=NULL){  node \* temp = new node(num,p);  pre->next = temp;  }  }  listSize++;  }  int search(int num){  int result = 0;  node \* p = firstNode;  while (p->ele!=num){  if (p->next==NULL)return 0;  p = p->next;  result++;  }  return result+1;  }  void givRes(){  node \* p = firstNode;  while (p->next != NULL){  cout << p->ele<<",";  p = p->next;  }  cout << p->ele<<endl;  }  };  chain together(chain chain1,chain chain2){  chain result;  node \*p1 = chain1.firstNode;  node \*p2 = chain2.firstNode;  while (p1!=NULL){  result.insert(p1->ele);  p1 = p1->next;  }  while (p2!=NULL){  result.insert(p2->ele);  p2 = p2->next;  }  return result;  }  int main() {  int enter;  chain theChain;  cout << "Input1"<<endl;  cin >> enter;  while (enter!=0){  theChain.insert(enter);  cin >> enter;  }  cout << "Output1"<<endl;  theChain.givRes();  cout << "Input2" << endl;  cin >> enter;  theChain.insert(enter);  cout << "Output2"<<endl;  theChain.givRes();  cout << "Input3"<<endl;  cin >> enter;  cout << "Output3"<<endl;  cout << theChain.search(enter)<<endl;  cout << "Input4" << endl;  cin >> enter;  cout << "Output4"<<endl;  cout << theChain.search(enter)<<endl;  chain theChain1;  cout << "Input5"<<endl;  cin >> enter;  while (enter!=0){  theChain1.insert(enter);  cin >> enter;  }  cout << "Output5"<<endl;  theChain1.givRes();  chain togChain = together(theChain,theChain1);  togChain.givRes();  cout << "End0"<<endl;  return 0;  }  }  if (p->next==NULL){  if (p->ele<num)p->next = new node(num,NULL);  else if (p==firstNode){  node \* temp = new node(num,p);  firstNode = temp;  } else{  node \* temp = new node(num,p);  pre->next = temp;  }   } else if (p==firstNode){  node \* temp = new node(num,p);  firstNode = temp;  } else if (p->next!=NULL){  node \* temp = new node(num,p);  pre->next = temp;  }   }  listSize++; } | | | |
|  | | | |

**本次实验主要代码：**

#include <iostream>  
using namespace std;  
  
struct node{  
 int ele;  
 node \* next;  
  
 node(int t\_ele,node\* t\_next){  
 ele = t\_ele;  
 next = t\_next;  
 } #include <iostream>

using namespace std;

struct node{

int ele;

node \* next;

node(int t\_ele,node\* t\_next){

ele = t\_ele;

next = t\_next;

}

};

class chain{

public:

node \* firstNode;

int listSize;

chain():firstNode(NULL),listSize(0){};

bool isEmpty(){return listSize==0;}

void insert(int num){

if (firstNode==NULL){

firstNode = new node(num,NULL);

}else{

node \* p = firstNode;

node \* pre = p;

while (p->ele<num && p->next!=NULL){

pre = p;

p = p->next;

}

if (p->next==NULL){

if (p->ele<num)p->next = new node(num,NULL);

else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else{

node \* temp = new node(num,p);

pre->next = temp;

}

} else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else if (p->next!=NULL){

node \* temp = new node(num,p);

pre->next = temp;

}

}

listSize++;

}

int search(int num){

int result = 0;

node \* p = firstNode;

while (p->ele!=num){

if (p->next==NULL)return 0;

p = p->next;

result++;

}

return result+1;

}

void givRes(){

node \* p = firstNode;

while (p->next != NULL){

cout << p->ele<<",";

p = p->next;

}

cout << p->ele<<endl;

}

};

chain together(chain chain1,chain chain2){

chain result;

node \*p1 = chain1.firstNode;

node \*p2 = chain2.firstNode;

while (p1!=NULL){

result.insert(p1->ele);

p1 = p1->next;

}

while (p2!=NULL){

result.insert(p2->ele);

p2 = p2->next;

}

return result;

}

int main() {

int enter;

chain theChain;

cout << "Input1"<<endl;

cin >> enter;

while (enter!=0){

theChain.insert(enter);

cin >> enter;

}

cout << "Output1"<<endl;

theChain.givRes();

cout << "Input2" << endl;

cin >> enter;

theChain.insert(enter);

cout << "Output2"<<endl;

theChain.givRes();

cout << "Input3"<<endl;

cin >> enter;

cout << "Output3"<<endl;

cout << theChain.search(enter)<<endl;

cout << "Input4" << endl;

cin >> enter;

cout << "Output4"<<endl;

cout << theChain.search(enter)<<endl;

chain theChain1;

cout << "Input5"<<endl;

cin >> enter;

while (enter!=0){

theChain1.insert(enter);

cin >> enter;

}

cout << "Output5"<<endl;

theChain1.givRes();

chain togChain = together(theChain,theChain1);

togChain.givRes();

cout << "End0"<<endl;

return 0;

}  
};  
class chain{  
public:  
 node \* firstNode;  
 int listSize;  
 chain():firstNode(NULL),listSize(0){};  
 bool isEmpty(){return listSize==0;}  
  
 void insert(int num){  
 if (firstNode==NULL){  
 firstNode = new node(num,NULL);  
 }else{  
 node \* p = firstNode;  
 node \* pre = p;  
 while (p->ele<num && p->next!=NULL){  
 pre = p;  
 p = p->next;  
 }  
 if (p->next==NULL){  
 if (p->ele<num)p->next = new node(num,NULL);  
 else if (p==firstNode){  
 node \* temp = new node(num,p); #include <iostream>

using namespace std;

struct node{

int ele;

node \* next;

node(int t\_ele,node\* t\_next){

ele = t\_ele;

next = t\_next;

}

};

class chain{

public:

node \* firstNode;

int listSize;

chain():firstNode(NULL),listSize(0){};

bool isEmpty(){return listSize==0;}

void insert(int num){

if (firstNode==NULL){

firstNode = new node(num,NULL);

}else{

node \* p = firstNode;

node \* pre = p;

while (p->ele<num && p->next!=NULL){

pre = p;

p = p->next;

}

if (p->next==NULL){

if (p->ele<num)p->next = new node(num,NULL);

else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else{

node \* temp = new node(num,p);

pre->next = temp;

}

} else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else if (p->next!=NULL){

node \* temp = new node(num,p);

pre->next = temp;

}

}

listSize++;

}

int search(int num){

int result = 0;

node \* p = firstNode;

while (p->ele!=num){

if (p->next==NULL)return 0;

p = p->next;

result++;

}

return result+1;

}

void givRes(){

node \* p = firstNode;

while (p->next != NULL){

cout << p->ele<<",";

p = p->next;

}

cout << p->ele<<endl;

}

};

chain together(chain chain1,chain chain2){

chain result;

node \*p1 = chain1.firstNode;

node \*p2 = chain2.firstNode;

while (p1!=NULL){

result.insert(p1->ele);

p1 = p1->next;

}

while (p2!=NULL){

result.insert(p2->ele);

p2 = p2->next;

}

return result;

}

int main() {

int enter;

chain theChain;

cout << "Input1"<<endl;

cin >> enter;

while (enter!=0){

theChain.insert(enter);

cin >> enter;

}

cout << "Output1"<<endl;

theChain.givRes();

cout << "Input2" << endl;

cin >> enter;

theChain.insert(enter);

cout << "Output2"<<endl;

theChain.givRes();

cout << "Input3"<<endl;

cin >> enter;

cout << "Output3"<<endl;

cout << theChain.search(enter)<<endl;

cout << "Input4" << endl;

cin >> enter;

cout << "Output4"<<endl;

cout << theChain.search(enter)<<endl;

chain theChain1;

cout << "Input5"<<endl;

cin >> enter;

while (enter!=0){

theChain1.insert(enter);

cin >> enter;

}

cout << "Output5"<<endl;

theChain1.givRes();

chain togChain = together(theChain,theChain1);

togChain.givRes();

cout << "End0"<<endl;

return 0;

}  
 firstNode = temp;  
 } else{  
 node \* temp = new node(num,p);  
 pre->next = temp;  
 }  
  
 } else if (p==firstNode){  
 node \* temp = new node(num,p);  
 firstNode = temp;  
 } else if (p->next!=NULL){  
 node \* temp = new node(num,p);  
 pre->next = temp;  
 }  
  
 }  
 listSize++;  
 }  
  
  
 int search(int num){  
 int result = 0;  
 node \* p = firstNode;  
 while (p->ele!=num){ #include <iostream>

using namespace std;

struct node{

int ele;

node \* next;

node(int t\_ele,node\* t\_next){

ele = t\_ele;

next = t\_next;

}

};

class chain{

public:

node \* firstNode;

int listSize;

chain():firstNode(NULL),listSize(0){};

bool isEmpty(){return listSize==0;}

void insert(int num){

if (firstNode==NULL){

firstNode = new node(num,NULL);

}else{

node \* p = firstNode;

node \* pre = p;

while (p->ele<num && p->next!=NULL){

pre = p;

p = p->next;

}

if (p->next==NULL){

if (p->ele<num)p->next = new node(num,NULL);

else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else{

node \* temp = new node(num,p);

pre->next = temp;

}

} else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else if (p->next!=NULL){

node \* temp = new node(num,p);

pre->next = temp;

}

}

listSize++;

}

int search(int num){

int result = 0;

node \* p = firstNode;

while (p->ele!=num){

if (p->next==NULL)return 0;

p = p->next;

result++;

}

return result+1;

}

void givRes(){

node \* p = firstNode;

while (p->next != NULL){

cout << p->ele<<",";

p = p->next;

}

cout << p->ele<<endl;

}

};

chain together(chain chain1,chain chain2){

chain result;

node \*p1 = chain1.firstNode;

node \*p2 = chain2.firstNode;

while (p1!=NULL){

result.insert(p1->ele);

p1 = p1->next;

}

while (p2!=NULL){

result.insert(p2->ele);

p2 = p2->next;

}

return result;

}

int main() {

int enter;

chain theChain;

cout << "Input1"<<endl;

cin >> enter;

while (enter!=0){

theChain.insert(enter);

cin >> enter;

}

cout << "Output1"<<endl;

theChain.givRes();

cout << "Input2" << endl;

cin >> enter;

theChain.insert(enter);

cout << "Output2"<<endl;

theChain.givRes();

cout << "Input3"<<endl;

cin >> enter;

cout << "Output3"<<endl;

cout << theChain.search(enter)<<endl;

cout << "Input4" << endl;

cin >> enter;

cout << "Output4"<<endl;

cout << theChain.search(enter)<<endl;

chain theChain1;

cout << "Input5"<<endl;

cin >> enter;

while (enter!=0){

theChain1.insert(enter);

cin >> enter;

}

cout << "Output5"<<endl;

theChain1.givRes();

chain togChain = together(theChain,theChain1);

togChain.givRes();

cout << "End0"<<endl;

return 0;

}  
 if (p->next==NULL)return 0;  
 p = p->next;  
 result++;  
 }  
 return result+1;  
 }  
  
  
#include <iostream>

using namespace std;

struct node{

int ele;

node \* next;

node(int t\_ele,node\* t\_next){

ele = t\_ele;

next = t\_next;

}

};

class chain{

public:

node \* firstNode;

int listSize;

chain():firstNode(NULL),listSize(0){};

bool isEmpty(){return listSize==0;}

void insert(int num){

if (firstNode==NULL){

firstNode = new node(num,NULL);

}else{

node \* p = firstNode;

node \* pre = p;

while (p->ele<num && p->next!=NULL){

pre = p;

p = p->next;

}

if (p->next==NULL){

if (p->ele<num)p->next = new node(num,NULL);

else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else{

node \* temp = new node(num,p);

pre->next = temp;

}

} else if (p==firstNode){

node \* temp = new node(num,p);

firstNode = temp;

} else if (p->next!=NULL){

node \* temp = new node(num,p);

pre->next = temp;

}

}

listSize++;

}

int search(int num){

int result = 0;

node \* p = firstNode;

while (p->ele!=num){

if (p->next==NULL)return 0;

p = p->next;

result++;

}

return result+1;

}

void givRes(){

node \* p = firstNode;

while (p->next != NULL){

cout << p->ele<<",";

p = p->next;

}

cout << p->ele<<endl;

}

};

chain together(chain chain1,chain chain2){

chain result;

node \*p1 = chain1.firstNode;

node \*p2 = chain2.firstNode;

while (p1!=NULL){

result.insert(p1->ele);

p1 = p1->next;

}

while (p2!=NULL){

result.insert(p2->ele);

p2 = p2->next;

}

return result;

}

int main() {

int enter;

chain theChain;

cout << "Input1"<<endl;

cin >> enter;

while (enter!=0){

theChain.insert(enter);

cin >> enter;

}

cout << "Output1"<<endl;

theChain.givRes();

cout << "Input2" << endl;

cin >> enter;

theChain.insert(enter);

cout << "Output2"<<endl;

theChain.givRes();

cout << "Input3"<<endl;

cin >> enter;

cout << "Output3"<<endl;

cout << theChain.search(enter)<<endl;

cout << "Input4" << endl;

cin >> enter;

cout << "Output4"<<endl;

cout << theChain.search(enter)<<endl;

chain theChain1;

cout << "Input5"<<endl;

cin >> enter;

while (enter!=0){

theChain1.insert(enter);

cin >> enter;

}

cout << "Output5"<<endl;

theChain1.givRes();

chain togChain = together(theChain,theChain1);

togChain.givRes();

cout << "End0"<<endl;

return 0;

}  
 void givRes(){  
 node \* p = firstNode;  
 while (p->next != NULL){  
 cout << p->ele<<",";  
 p = p->next;  
 }  
 cout << p->ele<<endl;  
 }  
};  
  
chain together(chain chain1,chain chain2){  
 chain result;  
 node \*p1 = chain1.firstNode;  
 node \*p2 = chain2.firstNode;  
 while (p1!=NULL){  
 result.insert(p1->ele);  
 p1 = p1->next;  
 }  
 while (p2!=NULL){  
 result.insert(p2->ele);  
 p2 = p2->next;  
 }  
 return result;  
}  
  
  
  
int main() {  
 int enter;  
 chain theChain;  
 cout << "Input1"<<endl;  
 cin >> enter;  
 while (enter!=0){  
 theChain.insert(enter);  
 cin >> enter;  
 }  
 cout << "Output1"<<endl;  
 theChain.givRes();  
 cout << "Input2" << endl;  
 cin >> enter;  
 theChain.insert(enter);  
 cout << "Output2"<<endl;  
 theChain.givRes();  
 cout << "Input3"<<endl;  
 cin >> enter;  
 cout << "Output3"<<endl;  
 cout << theChain.search(enter)<<endl;  
 cout << "Input4" << endl;  
 cin >> enter;  
 cout << "Output4"<<endl;  
 cout << theChain.search(enter)<<endl;  
  
 chain theChain1;  
 cout << "Input5"<<endl;  
 cin >> enter;  
 while (enter!=0){  
 theChain1.insert(enter);  
 cin >> enter;  
 }  
 cout << "Output5"<<endl;  
 theChain1.givRes();  
 chain togChain = together(theChain,theChain1);  
 togChain.givRes();  
 cout << "End0"<<endl;  
  
  
  
  
  
  
  
  
 return 0;  
}