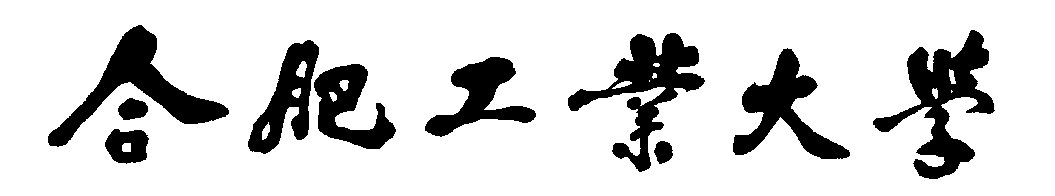
****

计算机与信息学院

数据结构实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术21-1 |
| 学生姓名及学号 | 杨程锦 2021214710 |
| 课程教学班号 |  |
| 任 课 教 师 | 胡学刚 |
| 实验指导教师 | 李培培 |
| 实验地点 | C201 |
| 2021 ~2022 学年第 二 学期 | |

说 明

实验报告是关于实验教学内容、过程及效果的记录和总结，因此，应注意以下事项和要求：

1．每个实验单元在4页的篇幅内完成一份报告。“实验单元”指按照实验指导书规定的实验内容。若篇幅不够，可另附纸。

2、各实验的**预习部分的内容是进入实验室做实验的必要条件**，请按要求做好预习。

3．实验报告要求：**书写工整规范，语言表达清楚，数据和程序真实。理论联系实际，认真分析实验中出现的问题与现象，总结经验。**

4．参加实验的每位同学应独立完成实验报告的撰写，其中程序或相关的设计图纸也可以采用打印等方式粘贴到报告中。严禁抄袭或拷贝，否则，一经查实，按作弊论取，并取消理论课考试资格。

5．实验报告作为评定实验成绩的依据。

实验序号及名称：实验 二 **循环链表、双链表及链表**

实验时间∶ 年 月 日

|  |
| --- |
| 预习内容 |
| 一、实验目的和要求∶  (1)理解单循环链表及双循环链表的特点。  (2)掌握这两种结构的算法设计。  (3)运用链表存储数据设计有关算法。  (4)理解头结点、头指针概念以及设置头结点的优点 |
| 二、实验任务∶  **<1>设计算法依次访问无头结点的单循环链表的各结点。**  实验测试数据基本要求：  第一组数据：链表元素为 （1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60）  第二组数据：链表元素为 （10, 20, 30, 40, 50, 60, 70, 80, 90, 100）  **<2>设计算法以判断一个带头结点的单循环链表是否满足这样的条件：**  **其中每个结点的元素值与其序号的差的绝对值不大于３。**  若成立, 返回TRUE, 否则返回FALSE。  实验测试数据基本要求：  第一组数据：链表元素为  （1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 18）  第二组数据：链表元素为  （1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 20, 18）  **<3>利用递增有序的单循环链表表示集合, 分别求两个链表表示的集合的交、并集所构成的链表。**  实验测试数据基本要求：  第一组  第一个链表元素为 （1, 3, 6, 10, 15, 16, 17, 18, 19, 20）  第二个链表元素为 （1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 20, 30）  第二组  第一个链表元素为 （1, 3, 6, 10, 15, 16, 17, 18, 19, 20）  第二个链表元素为 （2, 4, 5, 7, 8, 9, 12, 22）  第三组  第一个链表元素为 （）  第二个链表元素为 （1, 2, 3, 4, 5, 6, 7, 8, 9, 10）  **<4>设计算法以构造带头结点的双循环链表。**  实验测试数据基本要求：  第一组数据：链表元素为 （1, 2, 3, 4, 5, 6, 7, 8, 9, 10）  第二组数据：链表元素为 （10, 30, 40, 55, 60, 70, 88, 99, 100）  **<5>编写算法以判断一个带头结点的双循环链表是否是对称的, 若成立, 返回TRUE, 否则返回FALSE。**  实验测试数据基本要求：  第一组数据：链表元素为 （1, 2, 3, 4, 5, 4, 3, 2, 1）  第二组数据：链表元素为 （1, 2, 3, 4, 5, 5, 4, 3, 2, 1）  第三组数据：链表元素为 （1, 2, 3, 4, 5, 6, 3, 2, 1）  第四组数据：链表元素为 （1, 2, 3, 4, 5, 5, 6, 4, 3, 2, 1） |
| 三、实验准备方案，包括以下内容：  （硬件类实验：实验原理、实验线路、设计方案等）  （软件类实验：所采用的核心方法、框架或流程图及程序清单）  任务一：设计算法依次访问无头结点的单循环链表的各结点    任务二：设计算法以判断一个带头结点的单循环链表是否满足其中每个结点的元素值与其序号的差的绝对值不大于３  遍历链表，寻找满足条件的结点    任务三：**利用递增有序的单循环链表表示集合, 分别求两个链表表示的集合的交、并集所构成的链表**  交集：遍历A、B两个链表，将相同的元素插入链表C中。    并集：  构造链表C为A的深拷贝，遍历B和C，将B，C中不同的数插入C中    任务四；**设计算法以构造带头结点的双循环链表**  在node结点加入前驱指针pre，插入删除等操作需处理前驱指针，可以直接用一个指针遍历链表，利用前驱完成插入删除操作。    **任务五：编写算法以判断一个带头结点的双循环链表是否是对称的, 若成立, 返回TRUE, 否则返回FALSE。**  设两个指针，一个从head往next遍历，一个从head->pre往pre遍历，如果两个指针指向的结点data不同，直接输出“F”，否则遍历结束后输出“N” |

|  |
| --- |
| 实验内容 |
| 一、实验用仪器、设备：  Legion R9000p，cLion |
| 二、实验内容与步骤（过程及数据记录）：  **任务一：设计算法依次访问无头结点的单循环链表的各结点**  Error\_code noheadlist::element\_delete(int i) {  node\* q = head;  node\* p;  for (int j = 0; j < i; j++)  q = q->next;  p = q->next;  q->next = p->next;  delete p;  count--;  return 1;  }  实验结果见实验二，正常  任务二：**设计算法以判断一个带头结点的单循环链表是否满足其中每个结点的元素值与其序号的差的绝对值不大于３**  #include <iostream>  using namespace std;  typedef int elementype;  typedef int Error\_code;  typedef struct node {  node \*next;  elementype data;  };  class noheadlist {  public:  noheadlist();  ~noheadlist();  Error\_code insert(int i,elementype x);  Error\_code element\_delete(int i);  Error\_code get\_element(int i, elementype &x);  Error\_code length();  private:  int count;  node \*head;  };  noheadlist::noheadlist() {  count = 0;  }  Error\_code noheadlist::length() {  return count;  }  Error\_code noheadlist::insert(int i, elementype x) {  node\* p;  p = new node;  p->data = x;  if (count == 0) {  p->next = p;  head = p;  }  else  {  node\* q=head;  for (int j = 1; j < i; j++)  q = q->next;  p->next = q->next;  q->next = p;  }  count++;  return 1;  }  Error\_code noheadlist::element\_delete(int i) {  node\* q = head;  node\* p;  for (int j = 0; j < i; j++)  q = q->next;  p = q->next;  q->next = p->next;  delete p;  count--;  return 1;  }  Error\_code noheadlist::get\_element(int i, elementype& x) {  node\* q = head;  for (int j = 0;j < i;j++)  q = q->next;  x = q->data;  return 1;  }  noheadlist::~noheadlist() {  while (count > 0)  element\_delete(length());  }  int main()  {  noheadlist(List);  for (int i = 0; i < 15; i++) {  List.insert(i, i);  }  int judge = 0;  for (int i = 0; i < List.length(); i++) {  elementype x;  List.get\_element(i, x);  cout << x << " ";  if (!(x - i<3 && x - i>-3))  judge = 1;  }  cout << endl;  if (judge == 0)  cout << "TRUE";  else if (judge == 1)  cout << "FLASE";  }  实验结果正常  任务三：**利用递增有序的单循环链表表示集合, 分别求两个链表表示的集合的交、并集所构成的链表**  #include <iostream>  typedef int elemenType;  typedef int Error\_code;  const int maxlen = 10;  using namespace std;  class List {  public:  List();  int Length() const;  Error\_code Get\_element(const int i, elemenType& x) const;  int Locate(const elemenType x) const;  Error\_code Insert(const int i, const elemenType x);  Error\_code Delete\_element(const int i);  private:  int data[maxlen];  int count;  };  List::List() { count = 0; }  int List::Length()const { return count; }  Error\_code List::Get\_element(const int i, elemenType& x) const {  if (i<0 || i>count)  return 0;  x = data[i];  return 1;  }  int List::Locate(const elemenType x) const {  for (int i = 0; i < Length(); i++)  if (data[i] == x)  return (i + 1);  return 0;  }  Error\_code List::Insert(const int i, const elemenType x)  {  if (count == maxlen)  return 0;  if (i < 0 || i > Length() + 1)  return 0;  for (int j = count; j >= i; j--)  data[j] = data[j - 1];  data[i] = x;  count++;  return 1;  }  Error\_code List::Delete\_element(const int i)  {  if (Length() == 0) return 0;  if (i < 0 || i > Length()) return 0;  for (int j = i + 1; j <= Length(); j++)  data[j - 2] = data[j - 1];  count--;  return 1;  }  void U(List A, List B, List& C) {  for (int i = 0; i < A.Length(); i++) {  int x = 0;  A.Get\_element(i, x);  C.Insert(i, x);  }  int x = 0, y = 0; bool judge = 0;  for (int i = 0; i < B.Length(); i++) {  judge = 0;  for (int j = 0; j < A.Length(); j++) {  A.Get\_element(j, x);  B.Get\_element(i, y);  if (x == y)  judge = 1;  }  if (judge == 0)  C.Insert(C.Length(), y);  }  };  void n(List A, List B, List& C) {  int x = 0, y = 0;  for (int i = 0; i < A.Length(); i++)  for (int j = 0; j < B.Length(); j++) {  A.Get\_element(i, x);  B.Get\_element(j, y);  if (x == y)  C.Insert(C.Length(), y);  }  }  void different(List A, List B, List& C) {  int x = 0, y = 0; bool judge = 0;  for (int i = 0; i < B.Length(); i++) {  judge = 0;  for (int j = 0; j < A.Length(); j++) {  A.Get\_element(i, x);  B.Get\_element(j, y);  if (x == y)  judge = 1;  }  if (judge == 0)  C.Insert(C.Length(), x);  }  for (int i = 0; i < A.Length(); i++) {  judge = 0;  for (int j = 0; j < B.Length(); j++) {  A.Get\_element(j, x);  B.Get\_element(i, y);  if (x == y)  judge = 1;  }  if (judge == 0)  C.Insert(C.Length(), y);  }  };  int main()  {  List(A);  A.Insert(0, 1);  A.Insert(1, 2);  A.Insert(2, 3);  A.Insert(3, 4);  for (int i = 0; i < A.Length(); i++) {  int x;  A.Get\_element(i, x);  cout << x << " ";  }  cout << endl;  List(B);  B.Insert(0, 3);  B.Insert(1, 4);  B.Insert(2, 5);  B.Insert(3, 6);  for (int i = 0; i < B.Length(); i++) {  int x;  B.Get\_element(i, x);  cout << x << " ";  }  cout << endl;  List(C1);  List(C2);  List(C3);  //U，C， different  U(A, B, C1);  n(A, B, C2);  different(A, B, C3);  for (int i = 0; i < C1.Length(); i++) {  int x;  C1.Get\_element(i, x);  cout << x << " ";  }  cout << endl;  for (int i = 0; i < C2.Length(); i++) {  int x;  C2.Get\_element(i, x);  cout << x << " ";  }  cout << endl;  for (int i = 0; i < C3.Length(); i++) {  int x;  C3.Get\_element(i, x);  cout << x << " ";  }  cout << endl;  A.~List();  B.~List();  // C.~List();  }  实验结果正常  任务四；**设计算法以构造带头结点的双循环链表**  #include<iostream>  typedef int elementype;  typedef int Error\_code;  using namespace std;  typedef struct dunode {  elementype data;  dunode\* prior, \* next;  };  class duList {  public:  duList();  Error\_code Length();  Error\_code Insert(int i,elementype x);  Error\_code element\_delete(int i);  Error\_code Get\_element(int i,elementype &x);  Error\_code change();  ~duList();  private:  int count;  dunode \*head;  };  duList::duList() {  head = new dunode;  head->prior = head;  head->next = head;  count = 0;  }  Error\_code duList::Length() { return count; }  Error\_code duList::Insert(int i, elementype x) {  dunode\* p = head, \* q;  q = new dunode;  for (int k = 0; i > k; k++) {  p = p->next;  }  p->next = q;  q->data = x;  q->prior = p;  p = q;  count++;  return 1;  }  Error\_code duList::element\_delete(int i) {  dunode\* p;  p = head->next;  for (int k = 0; k < i; k++)  p = p->next;  p->prior->next = p->next;  p->next->prior = p->prior;  count--;  delete p;  return 1;  }  Error\_code duList::Get\_element(int i, elementype& x) {  dunode\* s= head->next;  for (int k = 0; k < i; k++)  s = s->next;  x = s->data;  return 1;  }  Error\_code duList::change() {  dunode\* p=head->next, \* q;  for (int i = 0; i < Length();i++) {  q = head->next->next;  p->next = q->next;  q->next->prior = p;  p->prior = q;  q->next=p;  head->next = q;  q->prior = head;  }    return 1;  }  duList::~duList(){  while (Length() > 0)  element\_delete(Length());  delete head;  }  **结果见任务五，实验结果正常**  **任务五：编写算法以判断一个带头结点的双循环链表是否是对称的, 若成立, 返回TRUE, 否则返回FALSE。**  #include <iostream>  #include"../duList.h"  using namespace std;  int main() {  duList(list);  int x;  for (int i = 0; i < 3; i++) {  cin >> x;  list.Insert(i, x);  }  int\* a = new int[list.Length()];  for (int i = 0; i < 3; i++) {  list.Get\_element(i, x);  cout << x<<" ";  a[i] = x;  }  int judge = 0;  for (int i = 0; i < list.Length() / 2; i++) {  if (a[i] != a[list.Length() - 1])  judge = 1;  }  if (judge == 0)  cout << "TRUE";  else if (judge == 1)  cout << "FLASE";    list.~duList();  }  实验结果正常 |

|  |
| --- |
|  |
| 三、实验结果分析、思考题解答∶  实验结果正常，成功构造了无头结点的单循环链表，以及双循环链表，并完成了插入删除查找的操作，完成了，判断每个结点的元素值与其序号的差的绝对值不大于３**，**两个链表表示的集合的交、并集所构成的链表。判断一个带头结点的双循环链表是否是对称。 |
| 四、感想、体会、建议∶  这个实验基本上没什么阻碍，大部分数据结构都在作业中构造好了，基本上没什么感想，就是删除操作仍然忘了头结点问题。 |
| 实验成绩∶  指导教师签名：  年 月 日 |