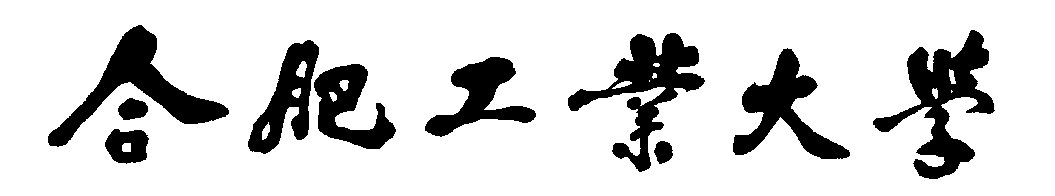
****

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 四 线性表

实验时间∶ 年 月 日

|  |
| --- |
| 预习内容 |
| 一、实验目的和要求∶  （1）理解数组与广义表两种扩展线性结构的特点；  （2）根据实际问题，熟练掌握基于多维数组与广义表的  存储结构及有关算法的设计。 |
| 二、实验任务∶  **<1> 设计算法实现稀疏矩阵的运算应用。**  **要求：1）设计稀疏矩阵类；2）给出测试用例，实现n组稀疏矩阵的相加与相乘。**  **<2>问题描述：本科生导师制：每个老师指导n个学生，老师带的研究生也可直接负责本科生。数据元素具有如下形式： ⑴ (老师, ((研究生1,(本科生1, …, 本科生m)), … )) ⑵ (老师, (本科生1, …, 本科生m))**  **设计算法，实现上述任务的插入、删除、查询、输出。**  **要求：**  **1）老师、研究生与本科生个数、相关信息自定义；**  **2）设计广义表类；**  **3）给出测试用例。** |
| 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：**稀疏矩阵的运算应用**  用一个三元组储存结构体，结构体的属性包括行数、列数、非零元data，  矩阵的加法：遍历两个三元组，行列数相等的结构体data相加，不相等的结构体创建新的结构体，储存在三元组中。  矩阵的乘法：    任务二：本科导师制  用结构体储存信息（指针，姓名，职位），向下插入研究生（本科生），向右插入其他老师（研究生/本科生），遍历输出所有的结点。 |

|  |
| --- |
| 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：  #include <iostream>  using namespace std;  #define MaxSize 100//矩阵中非零元素最多个数  #define N 4 //稀疏矩阵行数  #define M 4 //稀疏矩阵列数  typedef int elem;  typedef struct  {  int r; //行数  int c; //列数  elem d; //数据元素  }TupNode; //三元组定义  typedef struct  {  int row; //行数值  int cols; //列数值  int nums; //非零元素个数  TupNode data[MaxSize]; //存放数组  }TSMatrix; //三元组顺序表定义  //生成稀疏矩阵A三元组  void CreatMat(TSMatrix& t, elem A[N][M])  {  int i, j;  t.row = N; //总行数  t.cols = M; //总列数  t.nums = 0; //总非零元素数，初始为零  for (i = 0; i < N; i++)//遍历数组A  {  for (j = 0; j < M; j++)  {  if (A[i][j] != 0) //判断是否为非零元素  {  t.data[t.nums].r = i; //记录非零元素所在的行数  t.data[t.nums].c = j; //记录非零元素所在的列数  t.data[t.nums].d = A[i][j]; //记录非零元素值  t.nums++; //非零元素个数加一  }  }  }  }  //输出三元组  void DipMat(TSMatrix t)  {  int i;  if (t.nums <= 0)//判断是否有非零元素  return;  //printf("\t%d\t%d\t%d\n", t.row, t.cols, t.nums);  cout << t.row << " " << t.cols << " " << t.nums << endl;  //printf("-------------------------------------\n");  cout << "-------------------------------------" << endl;  for (i = 0; i < t.nums; i++)  //printf("\t%d\t%d\t%d\n", t.data[i].r+1, t.data[i].c+1, t.data[i].d);  cout << t.data[i].r + 1 << " " << t.data[i].c + 1 << " " << t.data[i].d << endl;  }  //两个稀疏矩阵相加后对应的三元组  void MatAdd(TSMatrix a, TSMatrix b, TSMatrix& c)  {  if (a.row != b.row || a.cols != b.cols)//判断是否符合矩阵相加条件，即两矩阵行数和列数分别相等  {  //printf("矩阵相加操作失败\n");  cout << "矩阵相加操作失败" << endl;  return;  }  c.row = a.row; //总行数赋值  c.cols = a.cols; //总列数赋值  int i = 0, j = 0, k = 0;  while (i < a.nums || j < b.nums) //遍历两个三元组  {  if (a.data[i].r < b.data[j].r)//比较非零数值所在的行数大小，将较小的行数的非零数值放进c的三元组  {  c.data[k].r = a.data[i].r;  c.data[k].c = a.data[i].c;  c.data[k].d = a.data[i].d;  i++;  k++;  }  else if (a.data[i].r == b.data[j].r)//比较非零数值所在的行数大小，相等时  {  if (a.data[i].c < b.data[j].c)//比较非零数值所在的列数大小，将较小的列数非零数值放进c的三元组  {  c.data[k].r = a.data[i].r;  c.data[k].c = a.data[i].c;  c.data[k].d = a.data[i].d;  i++;  k++;  }  else if (a.data[i].c == b.data[j].c)//比较非零数值所在的列数大小，将两个非零元素相加的值放进c的三元组  {  c.data[k].r = b.data[j].r;  c.data[k].c = b.data[j].c;  c.data[k].d = a.data[i].d + b.data[j].d;  i++;  j++;  k++;  }  else //比较非零数值所在的列数大小，将较小的列数非零数值放进c的三元组  {  c.data[k].r = b.data[j].r;  c.data[k].c = b.data[j].c;  c.data[k].d = b.data[j].d;  j++;  k++;  }  }  else //比较非零数值所在的行数大小，将较小的行数数非零数值放进c的三元组  {  c.data[k].r = b.data[j].r;  c.data[k].c = b.data[j].c;  c.data[k].d = b.data[j].d;  j++;  k++;  }  }  c.nums = k;//非零元素个数  }  //返回三元组 t 表示的 A[i][j]值  int getvalue(TSMatrix t, int i, int j)  {  for (int k = 0; k < t.nums; k++)  {  if (t.data[k].r == i && t.data[k].c == j)  return t.data[k].d;  }  return 0;  }  //两个稀疏矩阵相乘后对应的三元组  void MatMul(TSMatrix a, TSMatrix b, TSMatrix& c)  {  if (a.cols != b.row) //判断是否满足两矩阵相乘的条件，即第一个矩阵的列数与第二矩阵的行数相等  {  cout << "矩阵相加操作失败" << endl;  return;  }  int i, j, k = 0, s;  c.row = a.row;  c.cols = b.cols;  for (i = 0; i < a.row; i++) //控制行数  {  for (j = 0; j < b.cols; j++) //控制列数  {  s = 0;  for (int m = 0; m < b.cols; m++)  {  s = s + getvalue(a, m, i) \* getvalue(b, j, m);//第一个矩阵行的每个元素与第二个矩阵列的每个元素相乘，并将结果相加  }  if (s != 0) //如果数据元素不为0，则将其放进c三元组中  {  c.data[k].r = i;  c.data[k].c = j;  c.data[k].d = s;  k++;  }  }  }  c.nums = k; //总非零元素个数  }  int main()  {  TSMatrix t1, t2, c; //t1, t2为三元组  elem a[4][4] = {  {0,0,0,0},  {0,0,0,0},  {0,0,0,0},  {0,0,0,1}  };  CreatMat(t1, a); //生成稀疏矩阵A三元组  cout << "a的三元组：" << endl;  DipMat(t1); //输出三元组  elem b[4][4] = {  {3,0,0,0},  {0,0,0,0},  {0,0,0,0},  {0,0,0,1} };  CreatMat(t2, b); //生成稀疏矩阵B三元组  cout << "b的三元组：" << endl;  DipMat(t2); //输出三元组  cout << "c = a + b" << endl << "c的三元组:" << endl;  MatAdd(t1, t2, c); //计算两个稀疏矩阵相加后对应的三元组  DipMat(c); //输出三元组  cout << "c = a \* b" << endl;  cout << "c的三元组:" << endl;  MatMul(t1, t2, c); //矩阵相乘  DipMat(c); //输出三元组  return 0;  }    #include <iostream>  #include<string.h>  using namespace std;  typedef struct GLNode  {  char name[100]; /\*教师或学生的姓名\*/  string type; /\*结点类型：0-教师，1-研究生，2-本科生\*/  struct { struct GLNode\* hp, \* tp; };  /\*hp指向同级的下一结点，tp指向下级的首结点\*/  }GList;  class LS  {  private:  struct Node//建立结点  {  string name;  string type;  Node\* right, \*down;//每个节点有向右和向下的指针  Node() :right(NULL), down(NULL) {}  Node(string name1, string a)  {  name = name1;  type = a;  right = NULL;  down = NULL;  }  };  Node\* head;  Node\* cur;  public:  int number0 = 0, number1 = 0, number2 = 0;//导师人数、研究生人数、学生人数  int nameflag = 0;//名字标记  LS()//构造函数  {  Node\* p = new Node();//建立头节点  head = p;  }  void dfs(Node\* cur)//dfs 遍历列表，得出三类人的各人数  {  if (cur->type == "老师")  {  number0++;  cout << cur->name<< " " << cur->type << endl;  }  if (cur->type == "研究生")  {  number1++;  cout << cur->name<< " " << cur->type << endl;  }  if (cur->type == "本科生")  {  number2++;  cout << cur->name << " " << " " << cur->type << endl;  }  if (cur->right)dfs(cur->right);  if (cur->down)dfs(cur->down);  }  Node\* findname(Node\* cur, string name)//查询姓名,返回类型  {  if (cur->name == name) {  cout << "found" << endl;  cout << cur->name << " " << cur->type << endl;  nameflag = 1;  return cur;  }  if (cur->right)findname(cur->right, name);  if (cur->down)findname(cur->down, name);  if (nameflag == 0)return 0;  }  void all\_function()  {  cout << "please input the frist's name prof type" << endl;  string Name;  string Type;  cin >> Name >> Type;  Node\* p = new Node(Name,Type);//输入第一个节点  head->down = p;  cur = p;  while (1)  {  cout << "please input:" << endl;  cout << "1-show now node" << endl;//打印当前节点信息  cout << "2-insert rigth" << endl;//在当前节点往右插入  cout << "3-insert down" << endl;//在当前节点往下插入  cout << "4-return the frist" << endl;//返回头节点  cout << "5-moveright" << endl;//当前节点往右移动  cout << "6-movedown" << endl;//当前节点往下移动  cout << "7-delete now Node" << endl;//删除当前节点  cout << "8-all number" << endl;//统计所有结点  cout << "9-find name" << endl;//查询姓名  cout << "10-print all student" << endl;//打印当前节点的所有学生  cout << "0-break" << endl;//结束循环  int n;  cin >> n;  if (n == 0)break;  string Name, Prof;  string Type;  switch (n)  {  case 1://打印当前节点信息  {  cout << "now Node is:" << endl;  cout << cur->name << " " << cur->type << endl;  break;  }  case 2://在当前节点往右插入  {  cout << "now Node is:" << endl;  cout << cur->name << " " << cur->type << endl;  cout << "please input right name" << endl;//插入  cin >> Name >> Type;  Node\* p = new Node(Name,Type);  cur->right = p;  cur = cur->right;  cur = head->down;  break;  }  case 3://在当前节点往下插入  {  cout << "now Node is:" << endl;  cout << cur->name << " " << cur->type << endl;  cout << "please input numbers" << endl;//插入几个  int insertdown;  cin >> insertdown;  for (int i = 0; i < insertdown; i++)  {  cout << "please input Node name, prof, type" << endl;  cin >> Name >> Type;  Node\* p = new Node(Name, Type);  cur->down = p;  cur = cur->down;  }  cur = head->down;  break;  }  case 4://返回头节点  {  cur = head->down;  break;  }  case 5://当前节点往右移动  {  cout << "now Node is:" << endl;  cout << cur->name << " "<< " " << cur->type << endl;  cout << "after movedown Node is:" << endl;  cur = cur->right;  cout << cur->name << " " << cur->type << endl;  break;  }  case 6://当前节点往下移动  {  cout << "now Node is:" << endl;  cout << cur->name << " " << cur->type << endl;  cout << "after movedown Node is:" << endl;  cur = cur->down;  cout << cur->name << " " << cur->type << endl;  break;  }  case 7://删除当前节点  {  cout << "the Node will be delete is:" << endl;  cout << cur->name << " " << cur->type << endl;  int flag1 = 0, flag2 = 0;  Node\* p1 = NULL;  Node\* p2 = NULL;  if (cur->right != NULL)  {  p1 = cur->right;  flag1 = 1;  }  if (cur->down != NULL)  {  p2 = cur->down;  flag2 = 1;  }  if (flag1 && flag2)//右边和下边都有元素的话，把右边的节点赋值给当前要删除的节点，并且把下面的元素连接到当前节点  {  cur->name = p1->name;  cur->type = p1->type;  delete p1;  cur->down = p2;  }  else if (flag1)//只有右边有元素，把右边的节点赋值给当前要删除的节点  {  cur->name = p1->name;  cur->type = p1->type;  delete p1;  }  else if (flag2)//只有下边有元素，把下边的节点赋值给当前要删除的节点  {  cur->name = p2->name;  cur->type = p2->type;  delete p2;  }  break;  }  case 8://统计所有结点  { number0 = 0; number1 = 0; number2 = 0;  dfs(cur);  cout << "teacher master undergraduate:"<< number0 << " " << number1 << " " << number2 << endl;  break;  }  case 9://查询姓名（dfs）  {  string name;  cout << "please chaxun name" << endl;  cin >> name;  if (!findname(cur, name))  {  cout << "can not find" << endl;  }  nameflag = 0;  break;  }  case 10://打印当前节点的所有学生  {  dfs(cur);  break;  }  }  }  }  };  int main()  {  LS dusk;  dusk.all\_function();  } |
|  |
| 三、实验结果分析、思考题解答∶  实验结果正常，成功构造了稀疏矩阵，实现了稀疏矩阵的乘法加法，  并且成功实现了本科导师制。 |
| 四、感想、体会、建议∶  稀疏矩阵的乘法消耗了一些时间，和正常矩阵有一些不同，本科导师制想了很多储存结构，最后感觉链式结构最为简洁，容易经行其他操作。 |
| 实验成绩∶  指导教师签名：  年 月 日 |