电子科技大学信息与软件工程学院

实 验 报 告

学　　号 XXXXXX

姓　　名

（实验）　课程名称　数据结构与算法

理论教师　陈安龙

实验教师　陈安龙

电子科技大学教务处制表

电 子 科 技 大 学

实　验　报　告（1）

学生姓名：XXXXX　学 号：XXXXX　指导教师：陈安龙

实验地点：清水河科技实验大楼　**实验时间：2018/5/23**

一、实验室名称：学校实验中心软件实验室

二、实验项目名称：编程实现线性表的合并

三、实验学时：4

四、实验原理：

在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据元素之间的逻辑关系是由指针域来确定的。链式存储方式即可以用于表示线性结构，也可用于表示非线性结构。一般来说，在线性表的链式存储结构中，各数据结点的存储符号是不连续的，并且各结点在存储空间中的位置关系与逻辑关系也不一致。对于线性链表，可以从头指针开始，沿各结点的指针扫描到链表中的所有结点。

线性表的链接存储中，为了方便在表头插入和删除结点的操作，经常在表头结点（存储第一个元素的结点）的前面增加一个结点，称之为头结点或表头附加结点。这样原来的表头指针由指向第一个元素的结点改为指向头结点，头结点的数据域为空，头结点的指针域指向第一个元素的结点。

五、实验目的：

本实验通过定义单向链表的数据结构，设计创建链表、插入结点、遍历结点等基本算法，使学生掌握线性链表的基本特征和算法，并能熟练编写 C 程序，培养理论联系实际和自主学习的能力，提高程序设计水平。

六、实验内容：

使用数据结构 typedef struct node {

```
        Elemtype    data;
        struct    node    *next;
} ListNode, *ListPtr;
typedef struct stuInfo {
    int      stuID;
    char    stuName[10];          /*学生姓名*/
    int       Age    /*年龄*/
 } ElemType
```

实现带头结点的单向链表的创建、删除链表、插入结点等操作，可每个学生的学号互不相同，学号不同而姓名相同则为不同的学生，每个学生的学号在合并后的链表中不重复，如果出现重复，则删除年龄较小结点。最后打印输出合并后的链表元素，验证结果的正确性。

（1）设计学生信息结点的数据结构；

（2）用 C 语言实现创建升序链表的函数，每个结点的学号不同，按照学号升序排列；

（3）用 C 语言实现结点的插入的函数，插入后仍然为升序；

（4）编程实现两个单向链表合并，合并后仍然升序；

（5）编程实现合并后链表逆序排列的算法；

（6）打印输出合并后的链表元素。

七、实验器材（设备、元器件）：

PC 机一台，装有 C 语言集成开发环境。

八、数据结构与程序：

```
#include <algorithm>
#include <cstring>
#include <fstream>
#include <iostream>
using namespace std;

typedef struct {
  char Name[20];
  int Age;
  char StuID[20];
} StuInfo;

typedef struct StuNode {
  StuInfo Data;
  StuNode *next;

  ~StuNode() {
    if (this->next) delete this->next;
  }
} StuNode;

bool InitList(ifstream *, StuNode *);
void InsertNode(StuNode *, StuNode *);
StuNode *GetMergedList(StuNode *, StuNode *&);
void ReverseList(StuNode *);

bool InitList(ifstream *infile, StuNode *list) {
  if (!infile->is_open()) return false;

  char buffer[50];
  infile->seekg(0, ios::beg);
  infile->getline(buffer, 100);

  while (infile->getline(buffer, 100)) {
    StuNode *TempNode = new StuNode;
    TempNode->next = NULL;

    char *pin, *pout;

    pin = buffer, pout = TempNode->Data.Name;
    while (*pin != ',') *pout++ = *pin++;
    *pout = 0, ++pin;
```

```
    pout = TempNode->Data.StuID;
    while (*pin != ',') *pout++ = *pin++;
    *pout = 0, ++pin;

    int TempValue = 0;
    while (*pin != ',' && *pin != 0) TempValue = TempValue * 10 + *pin++
- '0';
    TempNode->Data.Age = TempValue;

    InsertNode(TempNode, list);
  }

  infile->close();
  return true;
}

void InsertNode(StuNode *stu, StuNode *list) {
  char ThisID[20];
  strcpy(ThisID, stu->Data.StuID);

  StuNode *pnode;
  for (pnode = list;
       pnode->next != NULL && (strcmp(pnode->next->Data.StuID, ThisID) <
0);
       pnode = pnode->next)
    ;

  stu->next = pnode->next;
  pnode->next = stu;
}

StuNode *GetMergedList(StuNode *list1, StuNode *&list2) {
  StuNode *p1 = list1, *p2 = list2->next;
  while (p1->next != NULL && p2 != NULL) {
    int dis = strcmp(p1->next->Data.StuID, p2->Data.StuID);
    if (dis < 0) {
      p1 = p1->next;
    } else if (dis == 0) {
      p2->Data.Age = max(p1->next->Data.Age, p2->Data.Age);
      StuNode *temp = p2;
      p2 = p2->next;
      temp->next = p1->next->next;
      p1->next->next = NULL;
```

```
      delete p1->next;
      p1->next = temp;
    } else {
      StuNode *temp = p2;
      p2 = p2->next;
      temp->next = p1->next;
      p1->next = temp;
    }
  }

  if (p2 != NULL) p1->next = p2;
  list2->next = NULL;
  delete list2;
  list2 = NULL;
  return list1;
}

void ReverseList(StuNode *list) {
  StuNode *TempHead = new StuNode;
  TempHead->next = NULL;

  for (StuNode *p = list->next; p != NULL;) {
    StuNode *temp = p;
    p = p->next;
    temp->next = TempHead->next;
    TempHead->next = temp;
  }

  list->next = TempHead->next;
  delete TempHead;
}

int main(void) {
  //------------------task 1---------------------------
  ifstream *csv1 = new ifstream;
  csv1->open("./Exp1_StuInfo1.csv", ios::in);
  StuNode *list1 = new StuNode;
  list1->next = NULL;

  if (InitList(csv1, list1)) {
    cout << "\nTask1: Create linked list in the increasing order of
Student "
            "ID\n\n";
    for (StuNode *p = list1->next; p != NULL; p = p->next)
```

```
    cout << p->Data.Name << " " << p->Data.Age << " " << p->Data.StuID
         << endl;
  }

  //--------------------task 2----------------------
  cout
      << "\nTask 2: Insert the info of students (enter \"#\" as
student's Name "
         "to exit)\n\n";
  while (1) {
    StuNode *InputStu = new StuNode;
    InputStu->next = NULL;
    cout << "Input the student's name: " << endl;
    cin >> InputStu->Data.Name;
    if (InputStu->Data.Name[0] == '#') {
      delete InputStu;
      break;
    }

    cout << "Input the student's age: " << endl;
    cin >> InputStu->Data.Age;

    cout << "Input the student's Student ID: " << endl;
    cin >> InputStu->Data.StuID;

    InsertNode(InputStu, list1);
  }

  //-------------------task 3-------------------------
  ifstream *csv2 = new ifstream;
  csv2->open("Exp1_StuInfo2.csv", ios::in);
  StuNode *list2 = new StuNode;
  list2->next = NULL;
  InitList(csv2, list2);
  list1 = GetMergedList(list1, list2);

  cout << "\nTask 3: Get merged list\n\n";
  for (StuNode *p = list1->next; p != NULL; p = p->next)
    cout << p->Data.Name << " " << p->Data.Age << " " << p->Data.StuID
<< endl;

  //---------------task 4--------------------------------
  cout << "\nTask 4: Reverse the merged linked list\n\n";
  ReverseList(list1);
```

```
  for (StuNode *p = list1->next; p != NULL; p = p->next)
    cout << p->Data.Name << " " << p->Data.Age << " " << p->Data.StuID
<< endl;

  delete list1;  //递归释放所有结点（见析构函数）

  system("Pause");
  return 0;
}
```

九、程序运行结果：

	A	B	C
1	StuName	StuID	Age
2	A	20170004	18
3	B	20170202	19
4	C	20170101	18
5	D	20170003	18
6	E	20170055	19
7	F	20170020	18
8	F	20170087	20
9	H	20170008	17
10	I	20170009	18
11	J	20170010	18
12			

	A	B	C
1	StuName	StuID	Age
2	A	20170004	18
3	b	20170022	19
4	c	20170021	18
5	d	20170023	18
6	e	20170025	19
7	f	20170040	18
8	g	20170027	20
9	h	20170028	17
10	i	20170029	18
11	j	20170030	18
12			

学生数据源文件: Exp1_StuInfo1.csv | Exp1_StuInfo2.csv

```
PS C:\Users\mx_028\Dropbox\Programming\C++
\DSCourse\Exper> ./Exp_1.exe

Task1: Create linked list in the increasin
g order of Student ID

D 18 20170003
A 18 20170004
H 17 20170008
I 18 20170009
J 18 20170010
F 18 20170020
E 19 20170055
F 20 20170087
C 18 20170101
B 19 20170202

Task 2: Insert the info o students (enter
"#" as student's Name to exit)

Input the student's name:
Jack
Input the student's age:
19
Input the student's Student ID:
20170089
Input the student's name:
#
```

```
Task 3: Get merged list

D 18 20170003
A 18 20170004
H 17 20170008
I 18 20170009
J 18 20170010
F 18 20170020
c 18 20170021
b 19 20170022
d 18 20170023
e 19 20170025
g 20 20170027
h 17 20170028
i 18 20170029
j 18 20170030
f 18 20170040
E 19 20170055
F 20 20170087
Jack 19 20170089
C 18 20170101
B 19 20170202
```

```
Task 4: Reverse the merged linked list

B 19 20170202
C 18 20170101
Jack 19 20170089
F 20 20170087
E 19 20170055
f 18 20170040
j 18 20170030
i 18 20170029
h 17 20170028
g 20 20170027
e 19 20170025
d 18 20170023
b 19 20170022
c 18 20170021
F 18 20170020
J 18 20170010
I 18 20170009
H 17 20170008
A 18 20170004
D 18 20170003
Press any key to continue . . .
```

（2）读取文件，创建升序链表 |（4）升序链表合并 |（5）链表逆序并打印
（3）插入学生信息

十、实验结论：

1. 任务 1：设计学生信息结点的数据结构。

结论：通过将学生的姓名、学号（考虑到学号的位数，改为使用字符串存储）、年龄打包在节点结构 StuNode 中，实现了实验目标。

2. 任务 2：实现创建升序链表的函数，每个结点的学号不同，按照学号升序排列。

结论：在函数 InitList()中，使用 C++的文件流按行读取与程序同目录下的.csv 文件，从而构造临时结构；然后通过调用下述的 InsertNode()函数插入到合适的位置。通过边读取，边插入的流程，实现了实验目标。

3. 任务 3：实现结点的插入的函数，插入后仍然为升序。

结论：函数 InsertNode()通过以下几个步骤实现了插入：首先，读取由外部传入的临时结构体（从 main()中调用时由用户输入，在 InitList()中调用时从文件中读取）；然后从链表首节点开始依次比较学号的字典序，找到合适的插入位置；然后修改指针域插入节点。实现了实验目标。

4. 任务 4：编程实现合并后链表逆序排列的算法。

结论：

合并：使用二路归并算法，实现了两个链表的合并（List2 并入 List1 中，然后释放 List2 的 List2 的头节点并置头节点指针为 NULL）。

逆序：首先建立一个临时的头节点指针，然后使用头插法依次移动原链表的首节点到新的链表中；然后将新链表的所有节点整体移动到原链表中并删除临时头节点。

将以上操作封装至 GetMergeList()和 ReverseList()中，实现了实验目的。

5. 任务 5：打印输出合并后的链表元素。

结论：依次输出所有节点的数据，实现了实验目的。

十一、总结及心得体会：

1. 链表对于存储需要变换顺序的结构十分高效，但在设计针对链表的算法时要特别注意越界判定：不能访问 NULL 指针、不能访问未分配内存的节点指针。在设计程序的过程前期我稍不注意就容易因这两个原因出现段错误。
2. 操作链表中节点的增删、移动时，要尤其注意指针域改变的顺序，尽可能先对孤立节点的指针域赋值（增箭头），后改变链中节点的指针域（删箭头）。
3. 使用 C++语言中的 new、delete 关键字并设计合适的递归析构函数可以使链表的创建与销毁极其方便。
4. 使用逗号分隔的.csv 文件作为测试集，实际上操作与纯文本文件无异，但可以从程序外部使用 Excel 方便地查看与更改数据。所以，.csv 文件可以作为理想的顺序表外部存储容器。