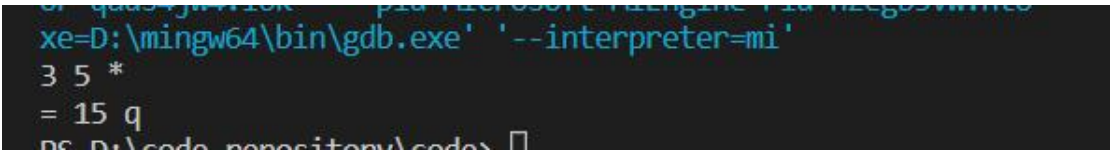


# 计算机学院 高级语言程序设计 课程实验报告

实验题目：线性群体类和算法		学号：202200130048
日期：2023. 4. 28	班级： 6	姓名： 陈静雯
Email：1205037094@qq. com		
<p>实验步骤与内容：</p> <ol style="list-style-type: none"><li>1. 实践第 9 章 PPT，例 9-8, 9-9，栈实现计算器。</li><li>2. 实现第 9 章实验 9，实验任务（1），利用课本上的例 9-7 作为应用程序。</li><li>3. 实现第 9 章实验 9，实验任务（2），利用本实验给出的 link.h 文件，实现例 9-6 的链表类。完成 lab9_2. cpp 问 link.h 的实现代码中有 bug 吗？如何修改？</li><li>4. 实现第 9 章实验 9，实验任务（3），用链表实现队列，完成 lab9_3. cpp。我们在用链表实现队列的时候，除了保存链表的头节点之外，还需要保存最后的尾节点来实现更快的 push 和 pop 操作。那队列头对应链表头还是链表尾更好？请按照你的想法用链表实现一个队列</li><li>6. 练习习题 9-4，利用 DNode. h，编写使用双向链表的程序，其中实现左插入、右插入，删除，显示等链表的功能。</li><li>7. 练习习题 9-10，直接插入排序。</li><li>8. 练习习题 9-12，选择排序。</li><li>9. 练习习题 9-14，起泡排序。</li><li>10. 练习习题 9-19，折半查找。</li></ol>		
<p>结论分析与体会：</p> <ol style="list-style-type: none"><li>1. </li><li>2. <pre>#include &lt;iostream&gt; using namespace std; #ifndef NODE_H #define NODE_H //类模板的定义 template &lt;class T&gt; class Node { private:     Node&lt;T&gt; *next;    //指向后继结点的指针 public:     T data;    //数据域</pre></li></ol>		

```

Node() {} ; //比课本上多一个默认构造函数
Node (const T &data, Node<T> *next = 0); //构造函数
void insertAfter (Node<T> *p); //在本结点之后插入一个同类结点 p
Node<T> *deleteAfter (); //删除本结点的后继结点，并返回其地址
Node<T> *nextNode (); //获取后继结点的地址
const Node<T> *nextNode () const; //获取后继结点的地址
};
//类的实现部分
//构造函数，初始化数据和指针成员
template <class T>
Node<T>::Node(const T& data, Node<T> *next/* = 0 */): data(data),
next(next) { }

//返回后继结点的指针
template <class T>
Node<T> *Node<T>::nextNode () {
    return next;
}
//返回后继结点的指针
template <class T>
const Node<T> *Node<T>::nextNode () const {
    return next;
}
//在当前结点之后插入一个结点 p
template <class T>
void Node<T>::insertAfter (Node<T> *p) {
    p->next = next; //p 结点指针域指向当前结点的后继结点
    next = p; //当前结点的指针域指向 p
}
//删除当前结点的后继结点，并返回其地址
template <class T> Node<T> *Node<T>::deleteAfter () {
    Node<T> *tempPtr = next; //将欲删除的结点地址存储到 tempPtr 中
    if (next == 0) //如果当前结点没有后继结点，则返回空指针
        return 0;
    next = tempPtr->next; //使当前结点的指针域指向 tempPtr 的后继结
    点
    return tempPtr; //返回被删除的结点的地址
}
#endif //NODE_H

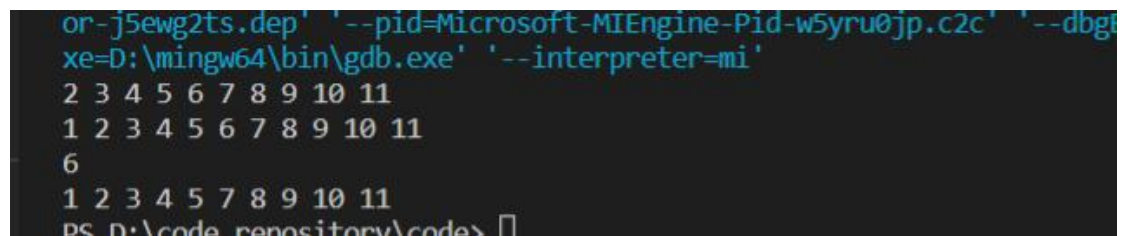
int main() {
    Node<int>*p;
    Node<int>a(1, 0);
    p=&a;

```

```

Node<int>b[10];
for(int i=0;i<10;i++){
    int item;
    cin>>item;
    b[i].data=item;
    p->insertAfter(&b[i]); //保存输入的整数，生成链表
    p=p->nextNode();
}
p=&a;
while(p->nextNode()!=0){ //顺序输出
    cout<<p->data<<" ";
    p=p->nextNode();
}
cout<<p->data<<" ";
cout<<endl;
int key;
cin>>key;
p=&a;
Node<int>*pre=&a;
while(p->data!=key){ //删除输入的整数
    pre=p;
    p=p->nextNode();
}
pre->deleteAfter();
p=&a;
while(p->nextNode()!=0){ //再顺序输出
    cout<<p->data<<" ";
    p=p->nextNode();
}
cout<<p->data<<" ";
cout<<endl;
}

```



```

or-j5ewg2ts.dep' '--pid=Microsoft-MIEngine-Pid-w5yru0jp.c2c' '--dbg
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
2 3 4 5 6 7 8 9 10 11
1 2 3 4 5 6 7 8 9 10 11
6
1 2 3 4 5 7 8 9 10 11
PS D:\code repository\code>

```

3.

```

int main() {
    LinkedList<int>a;
    LinkedList<int>b;
    for(int i=0;i<5;i++){

```

```

        int item;
        cin>>item;
        a.insertRear(item);
    }
    for(int i=0;i<5;i++){
        int item;
        cin>>item;
        b.insertRear(item);
    }
    b.reset();
    while(!b.endOfList()){
        int i=b.data();
        a.insertRear(i);
        b.next();
    }
    a.reset();
    while(!a.endOfList()){
        cout<<a.data()<<" ";
        a.next();
    }
    cout<<endl;
}

```

❌ 'Node<int>\* Node<int>::next' is private within this context gcc [行 156, 列 9]

❌ 'Node<int>\* Node<int>::next' is private within this context gcc [行 256, 列 15]

分析：报错，list 类不能用 node 类的 private 数据，可以把 node 里的 next 变成公有成员，或者把 list 类变成 node 类的友元

4.

```
#ifndef QUEUE_H
```

```
#define QUEUE_H
```

```
#include <cassert>
```

```
//类模板的定义
```

```
template <class T, int SIZE = 50>
```

```
class Queue {
```

```
private:
```

```
    Node<T> *front, *rear;
```

```
    LinkedList<T> list; //队列元素链表
```

```
public:
```

```
    Queue(); //构造函数，初始化队头指针、队尾指针、元素个数
```

```
    void insert(const T &item); //新元素入队
```

```
    T remove(); //元素出队
```

```
    void clear(); //清空队列
```

```

    const T &getFront() const; //访问队首元素
//测试队列状态
    int getLength() const; //求队列长度
    bool isEmpty() const; //判队队列空否
    bool isFull() const; //判断队列满否
};
//构造函数，初始化队头指针、队尾指针、元素个数
template <class T, int SIZE>
Queue<T, SIZE>::Queue() : list() { front = list.getrear(); rear =
list.getfront();}

template <class T, int SIZE>
void Queue<T, SIZE>::insert (const T& item) { //向队尾插入元素
    assert(list.getSize() != SIZE);
    list.insertRear(item); //向队尾插入元素
    list.getrear()->next = list.getfront(); //循环队列
}
template <class T, int SIZE>
T Queue<T, SIZE>::remove() {
    assert(list.getSize() != 0);
    return front->getData(); //返回首元素值
    list.deleteFront();
}
template <class T, int SIZE>
const T &Queue<T, SIZE>::getFront() const {
    return front->getData();
}
template <class T, int SIZE>
int Queue<T, SIZE>::getLength() const { //返回队列元素个数
    return list.getSize();
}
template <class T, int SIZE>
bool Queue<T, SIZE>::isEmpty() const { //测试队空否
    return list.getSize() == 0;
}
template <class T, int SIZE>
bool Queue<T, SIZE>::isFull() const { //测试队满否
    return list.getSize() == SIZE;
}
template <class T, int SIZE>
void Queue<T, SIZE>::clear() { //清空队列
    list.clear();
}
#endif //QUEUE_H

```

```

int main() {
    Queue<int>q;
    for (int i=0;i<5;i++) {
        int item;
        cin>>item;
        q.insert(item);
    }
    for (int i=0;i<5;i++) {
        cout<<q.remove()<<" ";
    }
}

```

```

=D:\mingw64\bin\gdb.exe --interpreter=mi
1 2 3 4 5
1 2 3 4 5

```

队列头应该指向链表尾，因为后进先出

6.

```

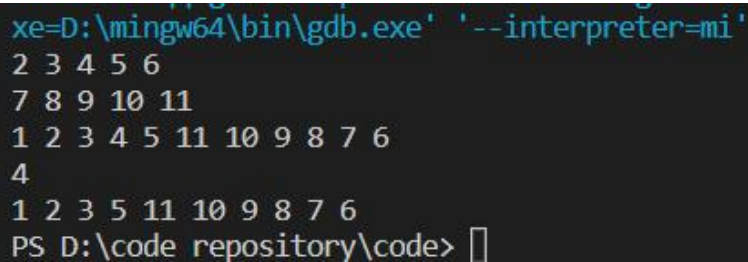
int main() {
    DNode<int>*p;
    DNode<int>a(1);
    p=&a;
    DNode<int>b[10];
    for (int i=0;i<5;i++) {
        int item;
        cin>>item;
        b[i].set(item);
        p->insertRight(&b[i]); //右插入
        p=p->nextNodeRight();
    }
    for (int i=5;i<10;i++) {
        int item;
        cin>>item;
        b[i].set(item);
        p->insertLeft(&b[i]); //左插入
        p=p->nextNodeLeft();
    }
    p=&a;
    while (p->nextNodeRight() !=&a) { //双向链表是个环?
        cout<<p->data<<" "; //尾结点指向了头结点
        p=p->nextNodeRight();
    } //顺序输出
    cout<<p->data<<" ";
    cout<<endl;
}

```

```

    int key;
    cin>>key;
    p=&a;
    DNode<int>*pre=&a;
    while (p->data!=key) {
        pre=p;
        p=p->nextNodeRight();
    }
    p->deleteNode();
    p=&a;
    while (p->nextNodeRight() !=&a) {
        cout<<p->data<<" ";
        p=p->nextNodeRight();
    }
    cout<<p->data<<" ";
    cout<<endl;
}

```



```

xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
2 3 4 5 6
7 8 9 10 11
1 2 3 4 5 11 10 9 8 7 6
4
1 2 3 5 11 10 9 8 7 6
PS D:\code repository\code>

```

7.

```

#include <iostream>
using namespace std;
int main() {
    int a[]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    for (int i=1; i<20; i++) {
        int temp=a[i];
        int j=i;
        while (j>0&& a[j-1]>temp) {
            a[j]=a[j-1];
            j--;
        }
        a[j]=temp;
        for (int i=0; i<20; i++) {
            cout<<a[i]<<" ";
        }
        cout<<endl;
    }
}

```

```

or-yhxga4ht.q40' '--pid=Microsoft-MIEngine-Pid-1knvfuii.eoc' '--dbg
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 2 3 5 7 9 11 13 15 17 19 4 6 8 10 12 14 16 18 20
1 2 3 4 5 7 9 11 13 15 17 19 6 8 10 12 14 16 18 20
1 2 3 4 5 6 7 9 11 13 15 17 19 8 10 12 14 16 18 20
1 2 3 4 5 6 7 8 9 11 13 15 17 19 10 12 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 13 15 17 19 12 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 15 17 19 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 19 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 19 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
PS D:\code repository\code>

```

8.

```

#include <iostream>
using namespace std;
template <class T>
void mySwap(T &x, T &y) {
    T temp = x;
    x = y;
    y = temp;
}

int main() {
    int a[]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    for (int i = 0; i < 19; i++) {
        int leastIndex = i;
        for (int j = i + 1; j < 20; j++)
            if (a[j] < a[leastIndex])
                leastIndex = j;
        mySwap(a[i], a[leastIndex]);
        for(int i=0;i<20;i++){
            cout<<a[i]<<" ";
        }
        cout<<endl;
    }
}

```



```

xe=D:\mingw64\bin\gdb.exe --interpreter=mi
1 3 5 7 9 11 13 15 17 19 2 4 6 8 10 12 14 16 18 20
1 2 5 7 9 11 13 15 17 19 3 4 6 8 10 12 14 16 18 20
1 2 3 7 9 11 13 15 17 19 5 4 6 8 10 12 14 16 18 20
1 2 3 4 9 11 13 15 17 19 5 7 6 8 10 12 14 16 18 20
1 2 3 4 5 11 13 15 17 19 9 7 6 8 10 12 14 16 18 20
1 2 3 4 5 6 13 15 17 19 9 7 11 8 10 12 14 16 18 20
1 2 3 4 5 6 7 15 17 19 9 13 11 8 10 12 14 16 18 20
1 2 3 4 5 6 7 8 17 19 9 13 11 15 10 12 14 16 18 20
1 2 3 4 5 6 7 8 9 19 17 13 11 15 10 12 14 16 18 20
1 2 3 4 5 6 7 8 9 10 17 13 11 15 19 12 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 13 17 15 19 12 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 17 15 19 13 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 15 19 17 14 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 19 17 15 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 19 16 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 19 17 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 19 18 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
PS D:\code repository\code>

```

9.

```

#include <iostream>
using namespace std;
template <class T>
void mySwap(T &x, T &y) {
    T temp = x;
    x = y;
    y = temp;
}

int main() {
    int a[]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    int i = 19;
    while (i > 0) {
        int lastExchangeIndex = 0;
        for (int j = 0; j < i; j++)
            if (a[j + 1] < a[j]) {
                mySwap(a[j], a[j + 1]);
                lastExchangeIndex = j;
            }
        i = lastExchangeIndex;
        for(int i=0;i<20;i++){
            cout<<a[i]<<" ";
        }
    }
}

```

```

        cout<<endl;
    }
}

```

```

or-gfthv77.kj3 --pid=Microsoft-MIEngine-Pid-qa0tnt2c.w3q --c
xe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
1 3 5 7 9 11 13 15 17 2 4 6 8 10 12 14 16 18 19 20
1 3 5 7 9 11 13 15 2 4 6 8 10 12 14 16 17 18 19 20
1 3 5 7 9 11 13 2 4 6 8 10 12 14 15 16 17 18 19 20
1 3 5 7 9 11 2 4 6 8 10 12 13 14 15 16 17 18 19 20
1 3 5 7 9 2 4 6 8 10 11 12 13 14 15 16 17 18 19 20
1 3 5 7 2 4 6 8 9 10 11 12 13 14 15 16 17 18 19 20
1 3 5 2 4 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 3 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
PS D:\code repository\code> int i = n - 1;

```

10.

```

#include <iostream>
using namespace std;
template <class T>
void mySwap(T &x, T &y) {
    T temp = x;
    x = y;
    y = temp;
}

int main() {
    int a[]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    int i = 19;
    while (i > 0) {
        int lastExchangeIndex = 0;
        for (int j = 0; j < i; j++)
            if (a[j + 1] < a[j]) {
                mySwap(a[j], a[j + 1]);
                lastExchangeIndex = j;
            }
        i = lastExchangeIndex;
        for(int i=0;i<20;i++){
            cout<<a[i]<<" ";
        }
        cout<<endl;
    }
    int key;
    cin>>key;
}

```

```

    int low = 0;
    int high = 19;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (key == a[mid]) {
            cout<<mid<<endl;
            break;
        }
        else if (key < a[mid])
            high = mid - 1;
        else
            low = mid + 1;
    }
}

```

```

xe=D:\mingw64\bin\gdb.exe --interpreter=mi
1 3 5 7 9 11 13 15 17 2 4 6 8 10 12 14 16 18 19 20
1 3 5 7 9 11 13 15 2 4 6 8 10 12 14 16 17 18 19 20
1 3 5 7 9 11 13 2 4 6 8 10 12 14 15 16 17 18 19 20
1 3 5 7 9 11 2 4 6 8 10 12 13 14 15 16 17 18 19 20
1 3 5 7 9 2 4 6 8 10 11 12 13 14 15 16 17 18 19 20
1 3 5 7 2 4 6 8 9 10 11 12 13 14 15 16 17 18 19 20
1 3 5 2 4 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 3 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
8
7
PS D:\code_repository\code>

```