
南开大学

Nankai University

# 南　开　大　学

## 网 络 空 间 安 全 学 院

**数据结构实验报告**

---

## 第三次实验作业

---

罗功成 1910487

年级：2019 级

专业：信息安全

指导教师：王玮

2021 年 11 月 25 日

# 摘要

队列用链表方式实现：添加元素，求最后几个数的乘积，最后几个数的最大值。


**关键字：队列的链表实现，复杂度分析**


# 目录

# 一、 实验流程

## (一) 实验要求

1. 函数 A 在队列末尾添加数字，如 a(3) 是指在队列尾部添加 3.
2. 函数 B 是指定最后几个值的乘积，比如 b(2) 是最后两个数字的乘积。
3. 函数 C 是比较队列最后几个数字中最大值，如 c(4) 是队列中最后 4 个数最大值。
4. 注意，使用队列的链表结构实现列表。
5. 在实现 B,C 函数时，不能直接获取元素，必须通过入队列和出队列的操作进行。

## (二) 实验原理和思路

1. 以链表方式创建队列时要注意需要队头队尾两个指针，出队列需要判断一下当前队列是否已经为空。
2. A 函数：入队列操作就是调用 push 函数，这里创建一个中间节点 S，将读到的值保存在 s-data 中，并将 s 的下一个置为空值，同时将队尾指针指向新的 s 节点。每调用一次，完成一个元素的从队尾入队操作。
3. B 函数：测量当前队列长度 getsize 记为 y1, 用一个数组来接受队头元素，之后把队头元素 pop 出队列，下一个元素将作为新的队首元素，接着向数组的下一个位置存储后弹出，以此循环，直至队列为空。根据读到的 b(x), 可以将数组中的下标为 y-x 开始一直到 y-1 进行累乘，用 temp 做好记录后输出。(注意，要将 temp 重新置为 1，否则再次调用 b 函数时可能会保留着上一次的结果。)
4. B，C 函数结束后，都需要用循环（循环次数为前面 b,c 操作时队列长度）重新将数组中元素重新 push 入队列，为接下来的操作做准备.
5. C 函数：测量当前队列长度 getsize 记为 y1, 用一个数组来接受队头元素，之后把队头元素 pop 出队列，下一个元素将作为新的队首元素，接着向数组的下一个位置存储后弹出，以此循环，直至队列为空。之后根据读到的 c(x), 可以将数组中的下标为 y-x 开始一直到 y-1 进行比较，注意，初始比较 max 必须要设置为队列当前的首元素，用 max 比较后输出。

## (三) 实验核心代码

队列链表实现和函数声明

```
typedef struct Qnode {
    int data;
    struct Qnode* next;
};//创建链表

typedef struct LQueue {
    Qnode* front;
    Qnode* rear;
};//生成队头队尾指针

```

```
11 void initQueue(LQueue* q) {
12     q->front = q->rear = (Qnode*)malloc(sizeof(Qnode));
13     q->front->next = NULL;
14 }//链表队列的初始化，指向一个动态的节点
15
16 int empty(LQueue* t) {
17     return t->front->next == t->rear;
18 }//判断当前队列是否为空
19
20 void push(LQueue* t, int x) {
21     Qnode* s = (Qnode*)malloc(sizeof(Qnode));//S指向生成的Qnode
22     s->data = x;//输入值传给s的data
23     s->next = NULL;//把下一个置空
24     t->rear->next = s;//队尾指针的next指向s所指节点
25     t->rear = s;//队尾指针指向S
26 }//push，将节点插入队列，一次一个
27
28 void pop(LQueue* t) {
29     /*if (empty(t)) {
30         cout << "LQueue is empty,can't pop.\n";
31         return;
32     }*/
33     Qnode* q = t->front->next;
34     t->front->next = q->next;
35     free(q);
36     if (t->rear == NULL)
37         t->rear = t->front;
38 }//出队列，一次一个，FIFO
39
40 int getFront(LQueue* t) {
41     return t->front->next->data;
42 }//获取队头值
43
44 int getRear(LQueue* t) {
45     return t->rear->data;
46 }//获取队尾值
47
48 int getSize(LQueue* t) {
49     Qnode* q = t->front->next;
50     int k = 0;
51     while (q) {
52         k++;
53         q = q->next;
54     }
55     return k;
56 }//获取队列长度
57
58 void printQueue(LQueue* t) {
```

```
    Qnode* q = t->front->next;
    while (q) {
        cout << q->data << " ";
        q = q->next;
    }
    cout << "\n";
}//遍历打印所有值
```

函数 A

```
if (s == 'a')
        {
            int x1;
            cin >> x1;
            push(&L, x1);
            printQueue(&L);
            push(&L1, x1);
            count++;}
```

函数 B

```
if (s == 'b')
        {
            int x2;
            cin >> x2;//需要最后几个乘的个数
            int y1 = getSize(&L);//队列长度
            for (int i = 0; i < y1 ; i++)//所有出队列，然后剩下指定的那几个
                再进行相乘的操作。

            {
                z[i] = getFront(&L);//将队头元素依次保存
                pop(&L);//逐个出队列
            }
            int temp = 1;
            for (int j1 = y1-x2; j1 <y1; j1++)
            {

                temp = temp * z[j1];
            }
            cout << temp << endl;
            temp=1;
            for (int i = 0; i < count; i++)
            {
                int m=z[i];
                push(&L, m);//调用结束后，还要把原来的值重新放回队列去
            }

        }
```

函数 C

```
if (s == 'c')
        {
            int x3;
            cin >> x3;
                int z[100];
            for (int i = 0; i < count; i++)//所有元素都出队列，然后剩下指定的
                几个 再进行比大小的操作。

            {
                z[i] = getFront(&L);//将队头元素依次保存
                pop(&L);//逐个出队列
            }
            int max = z[count - x3];
            for (int i = count - x3; i < count; i++)
            {

                if (z[i] > max)
                    max = z2[i];
            }

            cout << max << endl;

            for (int i = 0; i < count; i++)
            {
                push(&L, z[i]);//调用结束后，还要把原来的值重新放回队列去
            }
        }
```

## （四） 实验结果与分析

输入指定次数，代表本次实验可以进行多少步。

输入 a(x) 时，可以看到将会遍历打印在队尾插入新元素 x 后当前的队列的元素情况。

输入 b(x) 后，可以看到将后 x 个数字相乘结果打印在下一行。

输入 c(x) 后，可以看到将后 x 个数字中最大值结果打印在下一行。

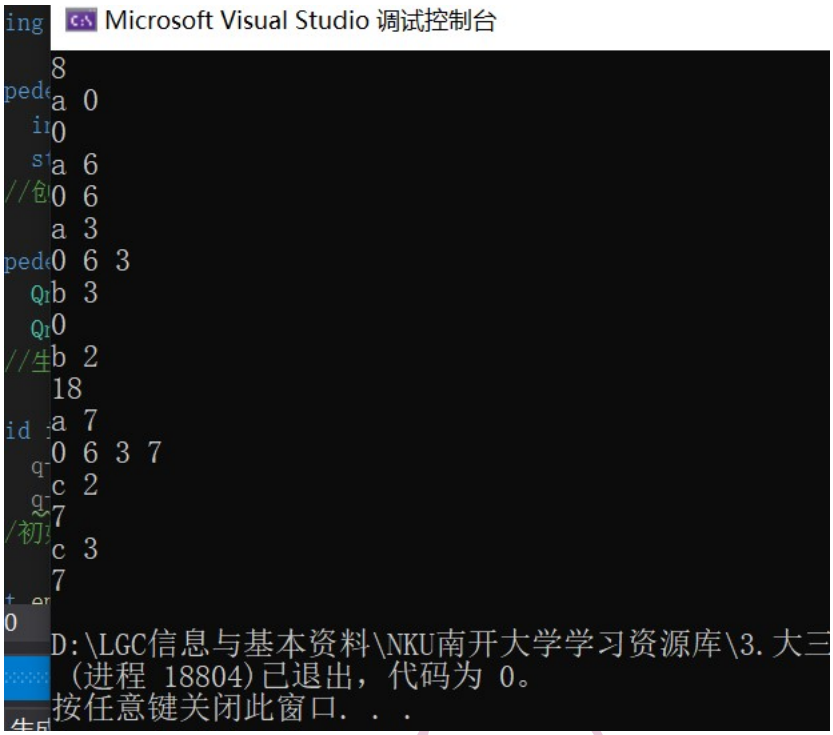


图 1: 函数 a,b,c 的实现效果

可见，以上代码可以满足本次实验的各项要求。

## 二、 复杂度解析

时间复杂度：O(n)

a 函数插入 n 次；b 函数出队列入队列各 n 次，至多进行 n 次乘法运算；c 函数出队列入队列各 n 次，至多进行 n 次比较，所以各项实际上都是 O(n)

空间复杂度：O(n)

用到 s,max,temp 三个中间变量，一个 int 类型的中间数组保留出队列的值，长度和队列长度一致为 n, 所以为 SP=n*(addr(z)+sizeof(int)).

## 三、 总结和收获

1. 对队列，链表等知识进行了回顾和总结。

2. 对 C++ 编程过程中函数，类，指针等的编写和调用能力得到了提高。