

Tongji University

Data Structure Course Curriculum Design Document

---

## Task5: Bank business

---

Author:

嘉杰 李

Jiajie Li

Supervisor:

颖 张

Prof. Ying Zhang

A companion use documentation for a program submitted in fulfillment  
of the requirements for the Data Structure curriculum design

in the

School of Software Engineering, Tongji



December 27, 2019

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

# Contents

1	Analysis	1
1.1	Background Analysis . . . . .	1
1.2	Functional Analysis . . . . .	1
2	Design	3
2.1	Data structure design . . . . .	3
2.2	Member function design . . . . .	3
2.3	System design . . . . .	5
3	Implementation	7
3.1	Main Function . . . . .	7
3.1.1	Main code . . . . .	7
3.1.2	Main flowchart . . . . .	8
4	Test	9
4.1	Function Tests . . . . .	9
4.1.1	A has more people than B . . . . .	9
4.1.2	B has more people than A . . . . .	9
4.1.3	Only one person . . . . .	10
A	Frequently Asked Questions	11
A.1	Why this document looks so sparse? . . . . .	11
A.2	Why is there header files in this project? . . . . .	11



## Chapter 1

# Analysis

### 1.1 Background Analysis

In the banking business, due to the differences in the speed of processing services and the types of processing services at each service window, when customers queue up to process business, the order of processing business is a question worth studying in order to improve the efficiency of the bank.

### 1.2 Functional Analysis

Suppose a bank has two business windows, A and B, and the speed of processing business is different. The processing speed of window A is twice that of window B. That is, when window A has processed 2 customers, window B processes Finish 1 customer. Given a sequence of customers arriving at the bank, output the sequence of customers in the order in which the business was completed. Assume that the time interval after the customer's letter arrives is not taken into account, and when two customers are processed in different windows at the same time, the customer in window A has priority to output.

This project requires input as a line of positive integers, where the first number  $N$  ( $N \leq 1000$ ) is the total number of customers, followed by the number of  $N$  customers. Odd numbered customers need to go to window A for business, even numbered customers go to window B. The numbers are separated by spaces. The program outputs the customer numbers in the order in which the business process is completed. The number keys are separated by spaces. There are extra spaces.

Since this problem needs to consider the queuing of the two windows, consider using a queue to solve the problem.



## Chapter 2

# Design

### 2.1 Data structure design

As described in the functional analysis above, design the A and B services to design two circular queues to process two types of service requests respectively; then assign each integer to these two queues according to the parity of the input sequence integers. In addition, it is necessary to design a process for two queue processing processes, that is, a loop. In the loop, two elements are output from the A queue, and then one element is output from the B queue. When the elements in one queue are found to be empty, all the elements in the other queue are output.

### 2.2 Member function design

ListNode Class:

```
template <class T> struct __listNode {
    __listNode<T>* prev;
    __listNode<T>* next;
    T data;
};
```

Linked List Class:

```
template <class T> class list{
private:
    typedef __listNode<T> list_node;
    typedef list_node* link_type;
    link_type get_node(){
        link_type p=(link_type)malloc(sizeof(list_node));
        p->next=NULL;
        p->prev=NULL;
        return p;
    }
    void put_node(link_type p){
        p->prev=NULL;
        p->next=NULL;
        p=NULL;
    }
};
```

```

        free(p);
    }
    link_type create_node(const T& x){
        link_type p = get_node();
        p->data = x;
        return p;
    }
    void destroy_node(link_type p){
        put_node(p);
    }
    void __insert(link_type p, const T &x){
        len++;
        link_type temp = create_node(x);
        p->next->prev = temp;
        temp->next = p->next;
        temp->prev = p;
        p->next = temp;
    }
    void __del(link_type p){
        len--;
        p->prev->next=p->next;
        p->next->prev=p->prev;
        destroy_node(p);
    }
    link_type loc(int pos){
        pos+=1;
        if (pos>len) return NULL;
        link_type temp = hnode;
        while (pos--) temp=temp->next;
        return temp;
    }

public:
    list();
    ~list();
    int Class_test(T temp);
    int push_back(const T& x) {__insert(hnode->prev, x); return 0;}
    int empty();
    int plist();
    bool isEmpty() const {return hnode==hnode->next;}

    int del(int pos);
    int insert(int pos, const T &x);
    int modify(int pos, const T &x);

    int len;
    link_type hnode; //head node
};

```



Queue class:

```
template <class T, class Sequence=list<T> >
class queue {
protected:
    Sequence c;
public:
    bool empty() const {return c.isEmpty();}
    int size() const {return c.len;}
    T front() {return c.isEmpty()?~1:c.hnode->next->data;}
    void push(const T& x) {c.push_back(x);}
    void pop() {c.del(0);}
};
```

## 2.3 System design

The program first enters a line of positive integers, separated by spaces, and the first number is the number of users, and then the output is performed in the order that the business process is completed.



## Chapter 3

# Implementation

### 3.1 Main Function

#### 3.1.1 Main code

```
int main(){
    int n,temp;
    queue<int> a,b;
    cin >> n;
    for (int i=0;i<n;++i) cin>>temp,temp%2==1?a.push(temp):b.push(temp);
    while (!(a.empty()&&b.empty())) {
        if (!a.empty()) cout<<a.front()<<" ",a.pop();
        if (!a.empty()) cout<<a.front()<<" ",a.pop();
        if (!b.empty()) cout<<b.front()<<" ",b.pop();
    }
    return 0;
}
```

## 3.1.2 Main flowchart

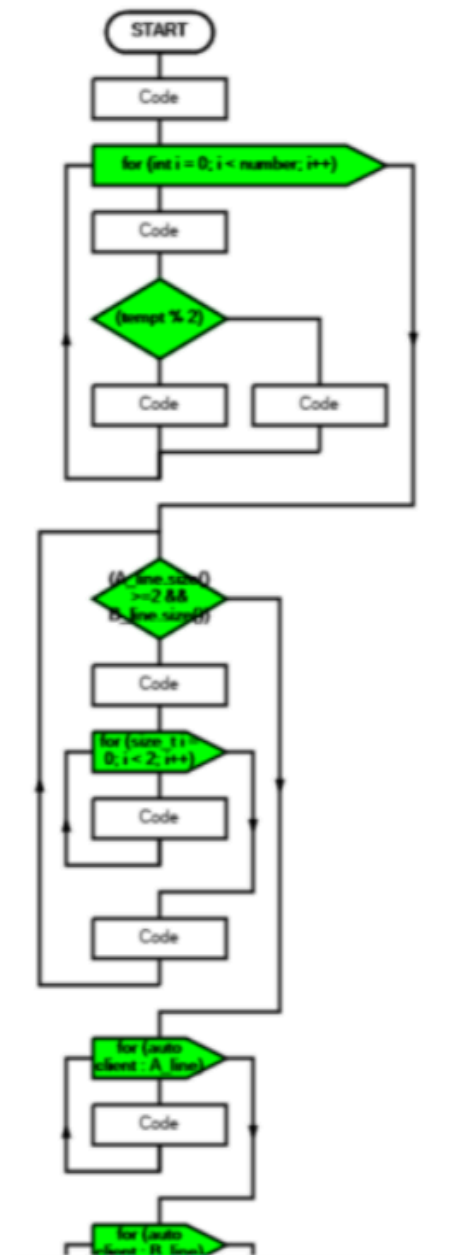


Figure 3.1: Main flowchart

## Chapter 4

# Test

### 4.1 Function Tests

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

#### 4.1.1 A has more people than B

Test Case:

8 2 1 3 9 4 11 13 15

Expected Result:

1 3 2 9 11 4 13 15

Program Result:

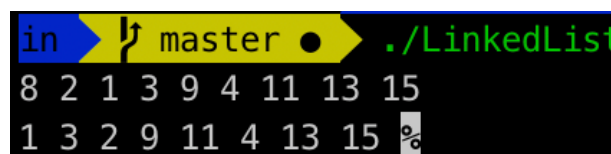


Figure 4.1: Screenshots

#### 4.1.2 B has more people than A

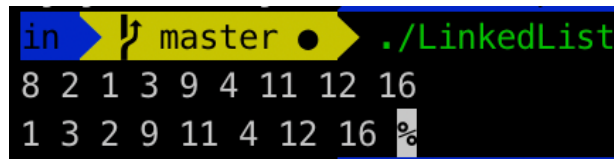
Test Case:

8 2 1 3 9 4 11 12 16

Expected Result:

1 3 2 9 11 4 12 16

Program Result:



```
in> ./LinkedList
8 2 1 3 9 4 11 12 16
1 3 2 9 11 4 12 16 %
```

Figure 4.2: Screenshots

#### 4.1.3 Only one person

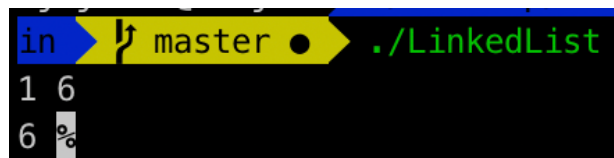
Test Case:

1 6

Expected Result:

6

Program Result:



```
in> ./LinkedList
1 6
6 %
```

Figure 4.3: Screenshots

## Appendix A

# Frequently Asked Questions

### A.1 Why this document looks so sparse?

This document is built using  $\text{\LaTeX}$  and is arranged in book format. There may be blank pages before and after each chapter.

### A.2 Why is there header files in this project?

I wrote basic stl header files, such as `vector.h`, etc. Except for the header files I wrote, the only header files used in all projects are `iostream` and `string.h`