银行业务模拟实验报告

班级：计算机系 1 班　　　　姓名：杨林卓　　　学号：PB15111629

题目：编写一个模拟银行业务的程序

一、需求分析

（1） 以链表 EventList 表示客户到达和离开事件，head 指示链表头元素，length 表示链表中元素个数。链表中元素 Event 有 type 和 occur_time 两个变量，均为整型。

（2） 以队列 CustomerQueue 表示银行中的客户队列。head 和 tail 分别指示队列中的头元素和尾元素，length 表示队列的长度。

（3） 模拟过程中用户成功办理业务则输出他的办理业务的类型——存款或者取款，办理业务等待时间，办理业务所在窗口。

（4） 最后输出进入银行的客户总数，成功办理业务的客户总数，所有进入银行客户等待的总时间，客户平均等待时间。

（5） 银行拥有的金额总数和营业结束时间以及客户到达银行的时间间隔可以在源文件适当处进行修改。

（6） 程序运行过程：

1） 模拟；2）输出结果。

二、概要设计

1. 设定事件和事件链表的抽象数据类型

```
class Event
{
    public:
        int type;       //事件类型  0 表示到达，1 表示离开
        int occur_time;         //事件发生时间
    Event *next;        //指向下一个事件
    Event()
    {
        type = 0;       //类型默认为到达
        occur_time = 0;         //发生时间默认为 0
        next = 0;       //指针默认为空
    }
    void Random(int);           //随机生成顾客的到达时间
    int Compare(Event);         //比较两个事件的先后顺序
};

class EventList         //事件表
{
        public:
              Event *head;      //指向链表头元素
              int length;       //链表长度
              EventList()
              {
                    head = new Event;
                    head->next = 0;
                    head->type = 0;
                    head->occur_time = 0;
              }
              bool OrderInsert(Event *);    //将元素按顺序插入表中
              Event GetFirst();             //获取表头元素
              bool DeleteFirst();           //删除表头元素
    };
```

2.　设定顾客和顾客队列的抽象数据类型

```
class Customer
{
   public:
        int type;  //表示顾客办理的业务类型，0 表示存钱，1 表示取钱
        int money;          //顾客办理业务的金额
        int arrive_time;        //顾客到达银行时间
        int cost_time;          //顾客办理业务需要花费的时间
        int leave_time;         //顾客离开银行时间
        Customer *prior;        //指向前一个顾客
        Customer *next;         //指向后一个顾客
        Customer(int ArriveTime = 0)
        {
              type = 0;             //顾客办理业务类型默认为存钱
              money = 0;            //业务金额默认为 0
              arrive_time = 0;  //到达时间默认为 0
              cost_time = 0;    //花费时间默认为 0
              leave_time = 0;         //默认值为 0，若为 0，说明未办理业
                                        务，否则成功办理了业务
              prior = 0;
              next = 0;             //指针默认为空
        }
        void Random();          //随机生成顾客的业务类型和金额总数
};

   class CustomerQueue
   {
        public:
```

```
        Customer *head;      //指向队列头元素
        Customer *tail;      //指向队列尾元素
        int length;          //队列长度
        CustomerQueue()
        {
            head = new Customer;
            tail = new Customer;
            head->prior = 0;
            head->next = tail;
            tail->prior = 0;
            tail->next = head;
            length = 0;
        }
        Customer GetFirst();            //获取头元素
        bool DeleteFirst();                 //删除头元素
        bool AddEnd(Customer *);            //在尾结点后插入
};
```

3．　设定银行抽象数据类型

```
class Bank
{
    public:
        int open_time;          //开始营业时间，以分钟为单位
        int close_time;         //停止营业时间
        int total_time;         //所有达到银行顾客等待的总时间
        int customer_number;    //到达银行的顾客总数
        int customer_success;   //成功办理业务的顾客总数
        int current_time;       //当前时间
        double total_money;     //银行存款总额，以元为单位
        EventList CustomerEvent;      //顾客事件链表
        CustomerQueue Window1;        //业务窗口 1
        CustomerQueue Window2;        //业务窗口 2
        Bank(int OpenTime = 0, int CloseTime = 600,
             double TotalMoney = 10000)
        {
            open_time = OpenTime;
            close_time = CloseTime;
            total_money = TotalMoney;
            total_time = 0;
            customer_number = 0;
            customer_success = 0;
            current_time = 0;
        }
```

```
                void OpenForDay();   //生成客户到达事件
                void CloseForDay();   //处理停止营业后还未办理业务的客户
                void DealEvent(Event);     //处理客户事件
                void Check();          //处理客户队列，检查是否有离开客户
                void DealBusiness();        //处理业务
                bool DealWindow1();       //检查窗口 1 顾客是否办理完业务
                bool DealWindow2();       //检查窗口 2 顾客是否可办业务
                bool Business1(Customer *);   //处理窗口 1 的业务
                bool Business2(Customer *);   //处理窗口 2 的业务
                bool IsMoreEvent();         //判断是否还有更多事件
};
```

4. 程序包含模块

（1） 主程序模块

（2） 银行模拟模块

（3） 事件模块

（4） 顾客模块

调用关系：

主程序 → 银行模拟模块 → 事件模块、顾客模块

三、函数详细设计

1. 事件类型

```
void Event::Random(int last_arrive_time)
{
    //随机生成顾客的到达时间
    //参数中 last_arrive_time 是前一个顾客到达时间

    //增加的时间在 min_time  到  max_time 之间

    int min_time = 1;
    int max_time = 10;
    //顾客到达时间在 last_arrive_time 的基础上随机增加一定的时间
    //初始设定为  min_time = 1, max_time = 10

    type = 0;       //生成的事件均为到达事件

    occur_time = last_arrive_time +
                min_time +
                (int) (max_time - min_time) * rand() / (RAND_MAX    + 1);
                //产生 1~max_time 之间的数  到达时间在上一个顾客到
                达之后若干时间
}
```

```
int Event::Compare(Event b)
{
    //比较两个事件发生的的先后顺序
    if (occur_time < b.occur_time)
    {   //若比事件 b 的发生时间早，返回-1
        return -1;
    }
    else if (occur_time == b.occur_time)
    {
        //若发生时间相同，返回 0
        return 0;
    }
    else
    {
        //若比事件 b 的发生时间晚，返回 1
        return 1;
    }
}

bool EventList::OrderInsert(Event *e)
//在事件表中安时间顺序插入
{
    if (!e)
    {
        //若指针 e 为空，返回 false
        return false;
    }

    if (length == 0)        //表为空
    {
        head->next = e;
        length++;
        return true;
    }

    Event *ptr = head->next;  //表不为空
    Event *ptr_pre = head;
    while(ptr)
    {
        if ((*ptr).Compare(*e) < 0)         //比表中元素大则继续寻找
        {
            ptr_pre = ptr;
            ptr = ptr->next;
        }
        else                                    //找到位置后插入
        {
            ptr_pre->next = e;
            e->next = ptr;
            length++;
            return true;
        }
    }

    //若比表中元素都大，则插入末尾
    ptr_pre->next = e;
    length++;
    return true;
}


Event EventList::GetFirst()
{   //获取表头元素
    Event e;
    if (length == 0)
    {
        return e;           //表为空返回初始值为 0 的事件
    }

    e = *(head->next);
    return e;
}


bool EventList::DeleteFirst()
{   //删除表头元素
    if (length == 0)
    {
        return false;   //表为空返回错误
    }

    Event *tmp;
    if (length == 1)    //表中只有一个元素
    {
        tmp = head->next;
        head->next = 0;
        delete tmp;
        length--;
        return true;
    }


        tmp = head->next;
        head->next = head->next->next;
        delete tmp;
        length--;
        return true;
    }
```

2. 顾客类型

```
    void Customer::Random()
    {   //随机生成顾客的业务类型和金额总数
        //金额在 min_money 到 max_money 之间
        int min_money = 100;
        int max_money = 5000;
        //花费时间在 min_cost_time  到  max_cost_time 之间
        int min_cost_time = 1;
        int max_cost_time = 10;
        //类型有存款和取款两种
        type = rand() % (1 + 1);
        money = min_money +
            (int) (max_money - min_money) * rand() / (RAND_MAX + 1);
                    //金额在  100~5000  之间
        cost_time = min_cost_time +
         (int) (max_cost_time - min_cost_time) * rand() / (RAND_MAX + 1);
                    //办理业务时间在  1~10  分钟之间
}
```

3. 银行类型

```
    void Bank::OpenForDay()
    {   //生成客户到达事件
        int time = 0;        //初始时间为 0
        while (time < close_time)   //时间未超过 close_time 则继续生成
        {
            Event *event = new Event;
            (*event).Random(time);
            time = event->occur_time;      //time  记录客户的到达时间
            CustomerEvent.OrderInsert(event); //向顾客事件表中插入
        }
    }
```

```
void Bank::CloseForDay()
{   //处理停止营业时还未办理业务的客户
    while (Window1.length != 0)  //当窗口 1 中有客户时
    {
        Customer tmp;
        tmp = Window1.GetFirst();
        total_time += (current_time - tmp.arrive_time);
        //将客户等待时间加到 total_time 中
        Window1.DeleteFirst();
        //处理完从窗口 1 队列中删除
    }

    //和窗口 1 同样处理窗口 2 为离开客户
    while (Window2.length != 0)
    {
        Customer tmp;
        tmp = Window2.GetFirst();
        total_time += (current_time - tmp.arrive_time);
        Window2.DeleteFirst();
    }

    //输出一天银行营业结果
    cout << "Customer Number:" << customer_number << endl;
    cout << "Total Time:" << total_time << endl;
    cout << "Average Time:";
    cout << (total_time * 1.0) / customer_number << endl;
    cout << "Customer Success Number:" << customer_success << endl;
}


void Bank::DealEvent(Event e)
{   //处理事件
    if (e.type == 0)    //客户到达事件
    {
        customer_number++;
        Customer *ptr = new Customer;
        (*ptr).Random();
        //随机生成客户具体信息
        ptr->arrive_time = current_time;
        //将客户到达时间设置为当前时间
        Window1.AddEnd(ptr);
        //向窗口 1 添加客户
    }

    else                    //客户离开事件
    {
        //检查窗口删除离开客户
        Check();
    }
}


void Bank::Check()          //处理客户队列，检查是否有离开客户
{
    Customer *ptr;
    //处理窗口 1
    ptr = Window1.head->next;
    if (ptr && ptr->leave_time == current_time)
    //队列不为空且离开时刻为当前时刻
    {
        //从窗口 1 中删除客户
        Window1.DeleteFirst();
    }

    //同样的方式处理窗口 2 客户
    ptr = Window2.head->next;
    if (ptr && ptr->leave_time == current_time)
    //队列不为空且离开时刻为当前时刻
    {
        //从窗口 2 中删除客户
        Window2.DeleteFirst();
    }
}


void Bank::DealBusiness()           //处理业务
{
    if(DealWindow1())
    //若窗口 1 办理的业务为存钱，则检查窗口 2 是否能够办理
    {
        //处理窗口 2
        DealWindow2();
    }
}
```

```
bool Bank::DealWindow1()             //办理窗口 1 业务
{
    if (Window1.length == 0)  //当前窗口没有客户
    {
        return false;
    }

    Customer *ptr;
    ptr = Window1.head->next;
    if (ptr->leave_time == 0)  //未办理业务
    {
        if ( ! Business1(ptr))
        //办理业务不成功，从窗口 1 中离开
        {
            Window1.DeleteFirst();
            ptr = 0;
            return false;
        }
        else           //办理成功则在事件表中插入离开事件
        {
            Event *p = new Event;
            p->type = 1;       //事件类型为离开
            p->occur_time = ptr->leave_time;
            //发生时间设置为顾客离开事件
            CustomerEvent.OrderInsert(p);
            //向事件表中顺序插入
            return true;
        }
    }
    return false;
}


bool Bank::DealWindow2()
//检查窗口 2 中客户是否能够成功办理业务
{
    if(Window2.length == 0)  //无客户则返回错误
    {
        return false;
    }

    Customer *ptr = Window2.head->next;
    Customer *ptr_pre = Window2.head;
    if (ptr->leave_time == 0)   //寻找是否有金额满足条件的顾客
    {
        while (ptr->money > total_money && ptr_pre !=
                Window2.tail->next)
        //当未找到满足要求的顾客且未全部搜索完，则继续搜索
        {
            ptr_pre = ptr;
            ptr = ptr->next;
        }

        if (ptr->money <= total_money)
            //如果能够成功办理则办理业务
        {
            //头尾指针移位指向新的队列头元素
            Window2.head->next = ptr;
            Window2.tail->next = ptr->prior;
            if (Business2(ptr))     //办理业务成功插入离开事件
            {
                Event *p = new Event;
                p->type = 1; //事件类型为离开
                p->occur_time = ptr->leave_time;
                CustomerEvent.OrderInsert(p);
                return true;
            }
        }
    }
}

bool Bank::Business1(Customer *ptr)       //处理窗口 1 的业务
{
    if (ptr->type == 0)     //顾客存钱
    {
        total_money += ptr->money;
        ptr->leave_time = current_time + ptr->cost_time;
        total_time += ptr->leave_time - ptr->arrive_time;
        cout << "Window1\t\t" << "Type: Saving" << "\t\tMoney: "
               << ptr->money << "\t\tStayingTime: "
               << ptr->leave_time - ptr->arrive_time << endl;
        //输出客户信息
        customer_success++;
        return true;
    }
    else      //顾客取钱
    {
        if (ptr->money <= total_money)    //检查是否可以办理业务
        {
            total_money -= ptr->money;
            ptr->leave_time = current_time + ptr->cost_time;
            total_time += ptr->leave_time - ptr->arrive_time;
            cout << "Window1\t\t" << "Type: Withdrawing"
                << "\tMoney: " << ptr->money << "\t\tStayingTime: "
                << ptr->leave_time - ptr->arrive_time << endl;
            //输出客户信息
            customer_success++;
            return true;
        }
        else      //银行存款不够，加入窗口 2 队列
        {
            Customer *tmp = new Customer;
            tmp->type = ptr->type;
            tmp->arrive_time = ptr->arrive_time;
            tmp->money = ptr->money;
            tmp->cost_time = ptr->cost_time;
            Window2.AddEnd(tmp);
            return false;
        }
    }
}



bool Bank::Business2(Customer *ptr)      //处理窗口 2 的业务
{
    if (!ptr)  //指针为空
    {
        return false;
    }

    total_money -= ptr->money;
    ptr->leave_time = current_time + ptr->cost_time;
    total_time += ptr->leave_time - ptr->arrive_time;
    cout << "Window2\t\t" << "Type: Withdrawing" << "\tMoney: "
            << ptr->money << "\t\tStayingTime: " << ptr->leave_time
            - ptr->arrive_time << endl;
        //输出客户信息
        return true;
    }



    bool Bank::IsMoreEvent()                //判断是否还有顾客
    {
        if (CustomerEvent.length == 0)
        {
            return false;
        }
        else
        {
            return true;
        }
}
```

4. 主程序

```
int main()
{
    srand((unsigned int)time(NULL));       //生成随机数种子
    Bank B;
    B.OpenForDay();           //初始化事件表
    while (B.IsMoreEvent() && B.current_time <= B.close_time)
    {   //当有客户且处于营业时间
        Event e;
        Customer c;
        e = B.CustomerEvent.GetFirst();    //获取当前事件
        if (e.occur_time > B.close_time)
        //事件发生的时间超过营业时间
        {
            break;
        }

        //处理业务
        B.current_time = e.occur_time;
        B.DealEvent(e);
        B.DealBusiness();
        B.CustomerEvent.DeleteFirst();
    }
    //停止营业，输出营业信息
    B.current_time = B.close_time;
```

```
        B.CloseForDay();
        return 0;
    }
```

四、调试分析

1. 产生的随机数随机性不够

   一开始我将生成随机数种子的操作放在了生成事件和顾客具体信息的函数里面，而计算机执行速度快，导致测试时生成的随机数都相同。然而手动调试时，因为单步执行的缘故，生成的随机数又有了随机性。因此导致用了很长的时间才找到问题的原因。

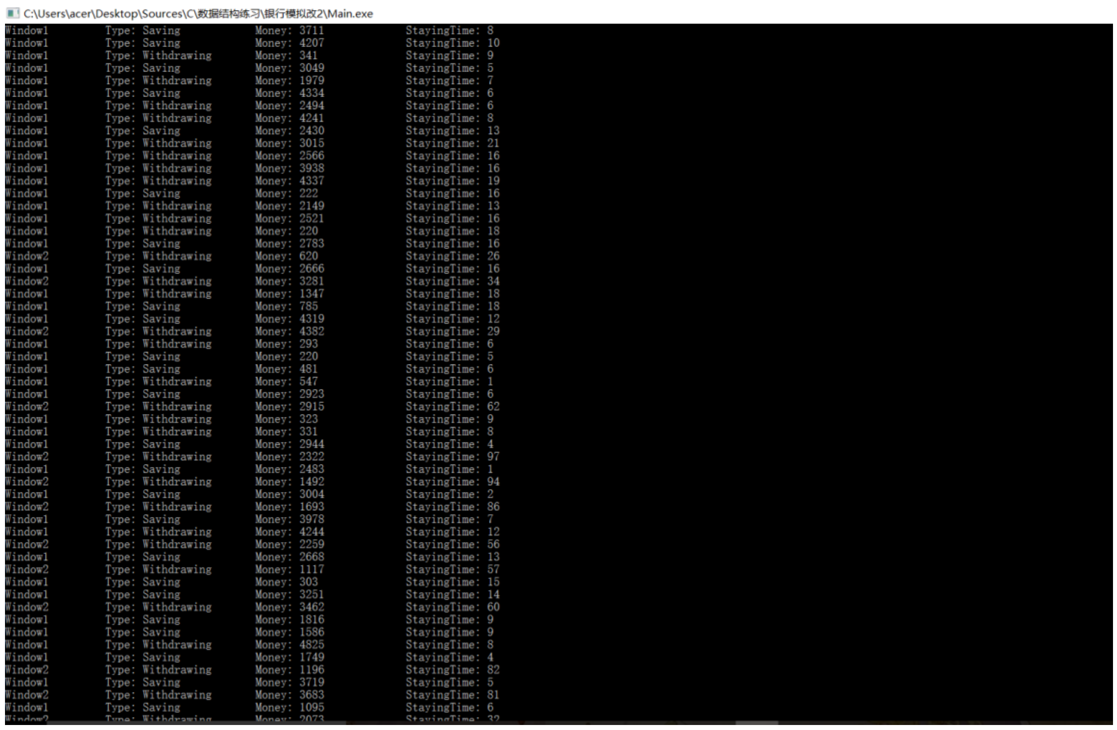
2. 插入操作函数和删除操作函数出错

   主要是编写函数时忽视了边界条件，比如表为空或者长度为 1 时情况比较特殊，需要特殊处理。再有处理的指针为头指针或者尾指针时也需要特别注意。在调试的时候发现了这些问题。

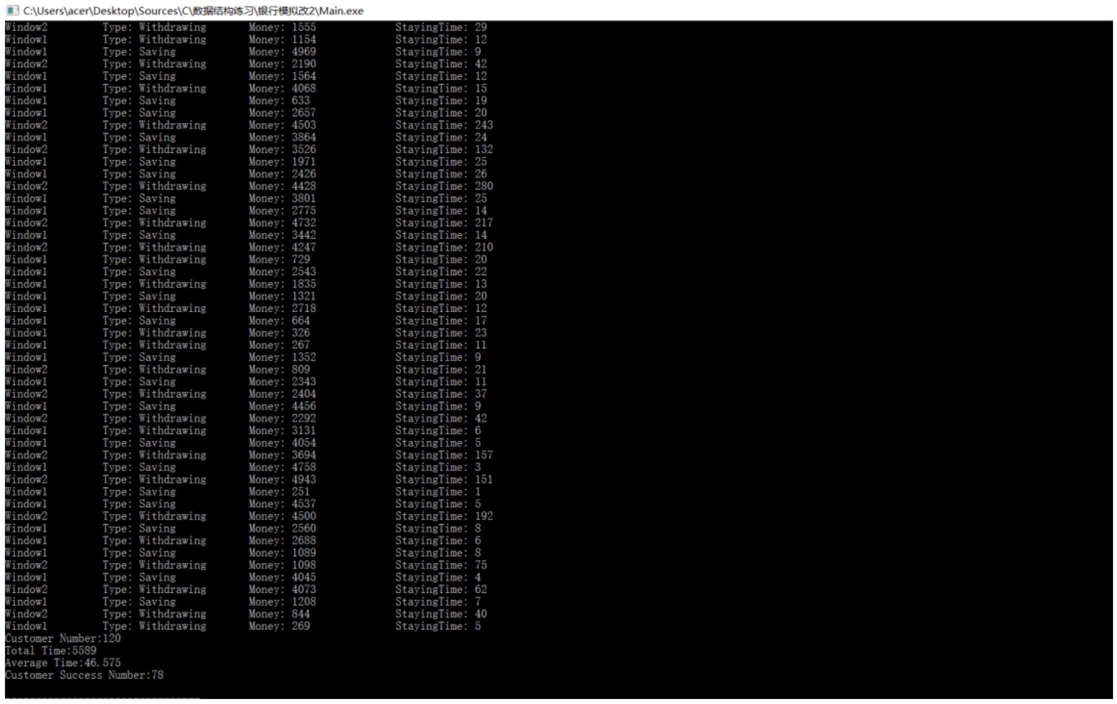
3. 窗口设置问题

   一开始我认为只需要设置一个窗口 2 便能够完成模拟，后来发现这样模拟出来的情况不够真实。因为办理业务需要花费时间，这段时间其他客户需要等待，因此不能忽略窗口 1 排队等待的时间。故后来又加入了窗口 1，使模拟出来的情况符合实际。

五、测试结果



```
C:\Users\acer\Desktop\Sources\C\数据结构练习\银行模拟改2\Main.exe
Window1         Type: Saving            Money: 3711             StayingTime: 8
Window1         Type: Saving            Money: 4207             StayingTime: 10
Window1         Type: Withdrawing       Money: 341              StayingTime: 9
Window1         Type: Saving            Money: 3049             StayingTime: 5
Window1         Type: Withdrawing       Money: 1979             StayingTime: 7
Window1         Type: Saving            Money: 4334             StayingTime: 6
Window1         Type: Withdrawing       Money: 2494             StayingTime: 6
Window1         Type: Withdrawing       Money: 4241             StayingTime: 8
Window1         Type: Saving            Money: 2430             StayingTime: 13
Window1         Type: Withdrawing       Money: 3015             StayingTime: 21
Window1         Type: Withdrawing       Money: 2566             StayingTime: 16
Window1         Type: Withdrawing       Money: 3938             StayingTime: 16
Window1         Type: Withdrawing       Money: 4337             StayingTime: 19
Window1         Type: Saving            Money: 222              StayingTime: 16
Window1         Type: Withdrawing       Money: 2149             StayingTime: 13
Window1         Type: Withdrawing       Money: 2521             StayingTime: 16
Window1         Type: Withdrawing       Money: 220              StayingTime: 18
Window1         Type: Saving            Money: 2783             StayingTime: 16
Window2         Type: Withdrawing       Money: 620              StayingTime: 26
Window1         Type: Saving            Money: 2666             StayingTime: 16
Window2         Type: Withdrawing       Money: 3281             StayingTime: 34
Window1         Type: Withdrawing       Money: 1347             StayingTime: 18
Window1         Type: Saving            Money: 785              StayingTime: 18
Window1         Type: Saving            Money: 4319             StayingTime: 12
Window2         Type: Withdrawing       Money: 4382             StayingTime: 29
Window1         Type: Withdrawing       Money: 293              StayingTime: 6
Window1         Type: Saving            Money: 220              StayingTime: 5
Window1         Type: Saving            Money: 481              StayingTime: 6
Window1         Type: Withdrawing       Money: 547              StayingTime: 1
Window1         Type: Saving            Money: 2923             StayingTime: 6
Window2         Type: Withdrawing       Money: 2915             StayingTime: 62
Window1         Type: Withdrawing       Money: 323              StayingTime: 9
Window1         Type: Withdrawing       Money: 331              StayingTime: 8
Window1         Type: Saving            Money: 2944             StayingTime: 4
Window2         Type: Withdrawing       Money: 2322             StayingTime: 97
Window1         Type: Saving            Money: 2483             StayingTime: 1
Window2         Type: Withdrawing       Money: 1492             StayingTime: 94
Window1         Type: Saving            Money: 3004             StayingTime: 2
Window2         Type: Withdrawing       Money: 1693             StayingTime: 86
Window1         Type: Saving            Money: 3978             StayingTime: 7
Window1         Type: Withdrawing       Money: 4244             StayingTime: 12
Window2         Type: Withdrawing       Money: 2259             StayingTime: 56
Window1         Type: Saving            Money: 2668             StayingTime: 13
Window2         Type: Withdrawing       Money: 1117             StayingTime: 57
Window1         Type: Saving            Money: 303              StayingTime: 15
Window1         Type: Saving            Money: 3251             StayingTime: 14
Window2         Type: Withdrawing       Money: 3462             StayingTime: 60
Window1         Type: Saving            Money: 1816             StayingTime: 9
Window1         Type: Saving            Money: 1586             StayingTime: 9
Window1         Type: Withdrawing       Money: 4825             StayingTime: 8
Window1         Type: Saving            Money: 1749             StayingTime: 4
Window2         Type: Withdrawing       Money: 1196             StayingTime: 82
Window1         Type: Saving            Money: 3719             StayingTime: 5
Window2         Type: Withdrawing       Money: 3683             StayingTime: 81
Window1         Type: Saving            Money: 1095             StayingTime: 6
Window2         Type: Withdrawing       Money: 2073             StayingTime: 32
```


C:\Users\acer\Desktop\Sources\C\数据结构练习\银行模拟改2\Main.exe

```
Window2         Type: Withdrawing       Money: 1555             StayingTime: 29
Window1         Type: Withdrawing       Money: 1154             StayingTime: 12
Window1         Type: Saving            Money: 4969             StayingTime: 9
Window2         Type: Withdrawing       Money: 2190             StayingTime: 42
Window1         Type: Saving            Money: 1564             StayingTime: 12
Window1         Type: Withdrawing       Money: 4068             StayingTime: 15
Window1         Type: Saving            Money: 633              StayingTime: 19
Window1         Type: Saving            Money: 2657             StayingTime: 20
Window2         Type: Withdrawing       Money: 4503             StayingTime: 243
Window1         Type: Saving            Money: 3864             StayingTime: 24
Window2         Type: Withdrawing       Money: 3526             StayingTime: 132
Window1         Type: Saving            Money: 1971             StayingTime: 25
Window1         Type: Saving            Money: 2426             StayingTime: 26
Window2         Type: Withdrawing       Money: 4428             StayingTime: 280
Window1         Type: Saving            Money: 3801             StayingTime: 25
Window1         Type: Saving            Money: 2775             StayingTime: 14
Window2         Type: Withdrawing       Money: 4732             StayingTime: 217
Window1         Type: Saving            Money: 3442             StayingTime: 14
Window2         Type: Withdrawing       Money: 4247             StayingTime: 210
Window1         Type: Withdrawing       Money: 729              StayingTime: 20
Window1         Type: Saving            Money: 2543             StayingTime: 22
Window1         Type: Withdrawing       Money: 1835             StayingTime: 13
Window1         Type: Saving            Money: 1321             StayingTime: 20
Window1         Type: Withdrawing       Money: 2718             StayingTime: 12
Window1         Type: Saving            Money: 664              StayingTime: 17
Window1         Type: Withdrawing       Money: 326              StayingTime: 23
Window1         Type: Withdrawing       Money: 267              StayingTime: 11
Window1         Type: Saving            Money: 1352             StayingTime: 9
Window2         Type: Withdrawing       Money: 809              StayingTime: 21
Window1         Type: Saving            Money: 2343             StayingTime: 11
Window2         Type: Withdrawing       Money: 2404             StayingTime: 37
Window1         Type: Saving            Money: 4456             StayingTime: 9
Window2         Type: Withdrawing       Money: 2292             StayingTime: 42
Window1         Type: Withdrawing       Money: 3131             StayingTime: 6
Window1         Type: Saving            Money: 4054             StayingTime: 5
Window2         Type: Withdrawing       Money: 3694             StayingTime: 157
Window1         Type: Saving            Money: 4758             StayingTime: 3
Window2         Type: Withdrawing       Money: 4943             StayingTime: 151
Window1         Type: Saving            Money: 251              StayingTime: 1
Window1         Type: Saving            Money: 4537             StayingTime: 5
Window2         Type: Withdrawing       Money: 4500             StayingTime: 192
Window1         Type: Saving            Money: 2560             StayingTime: 8
Window1         Type: Withdrawing       Money: 2688             StayingTime: 6
Window1         Type: Saving            Money: 1089             StayingTime: 8
Window2         Type: Withdrawing       Money: 1098             StayingTime: 75
Window1         Type: Saving            Money: 4045             StayingTime: 4
Window2         Type: Withdrawing       Money: 4073             StayingTime: 62
Window1         Type: Saving            Money: 1208             StayingTime: 7
Window2         Type: Withdrawing       Money: 844              StayingTime: 40
Window1         Type: Withdrawing       Money: 269              StayingTime: 5
Customer Number:120
Total Time:5589
Average Time:46.575
Customer Success Number:78

--------------------------------
```

上面为程序执行一次之后的两张截图，输出了客户办理的业务信息。最左边是客户办理业务的窗口，Type 表示客户办理的业务类型，Saving 表示存钱，Withdrawing 表示取款。Money 表示办理的业务金额。StayingTime 表示客户待在银行的时间。包含这次模拟在内，十次模拟的结果分别为：

到达客户总数	成功办理业务客户	平均等待时间
120	78	46.575
117	110	13.333
117	79	45.017
108	106	8.648
118	105	15.636
119	100	12.496
117	84	18.239
123	84	28.545
130	105	18.239
111	109	9.766

10 次模拟平均到达客户总数为 118，成功办理业务客户总数为 96，平均等待时间为 21.650 分钟。办理成功率为 81.36%。总的来说，测试结果比较理想。

六、附录

源程序文件名清单：

Bank_Simulation.h　//所有数据类型的声明文件
Event_Class.h　　　//事件类型成员函数实现
Event_List_Class.h　//事件链表成员函数实现

```
Customer_Class.h        //顾客类型成员函数实现
Customer_Queue_Class.h      //顾客队列成员函数实现
Bank_Class.h            //银行类型成员函数实现
Main.cpp                //主程序
```