银行业务模拟实验报告

姓名：孙天艺 学号：15030410038

**一、需求分析**

1. 客户业务分为两种：第一种是申请从银行得到一笔资金，即取款或借款；第二种是向银行投入一笔资金，即存款或还款。

2. 银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队。处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足时，则立即排入第二个队等候，直至满足时才离开银行， 否则业务处理完后立即离开银行。

3. 每接待完一个第二种业务的客户，则顺序检查和处理第二个队列中的客户,对能满足的申请者予以满足，不能满足者重新排到第二个队列的队尾。

4. 假设检查不需要时间,在此检查过程中,一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查（因为此时已不可能还有满足者）,转而继续接待第一个队列的客户。

5. 任何时刻都只开一个窗口,营业时间结束时所有客户立即离开银行。通过离散的模拟方法求出客户在银行内逗留的平均时间

**二、概要设计**

1.单链表的抽象数据类型定义为：

ADT eventlist {

数据对象：D={qi|qi∈ElemSet,i=1,2,……,n,n>=o}

数据关系：R={〈qi-1,qi>|qi-1,qi∈D,i=2,3,……,n}

队列的抽象数据类型定义：

ADT Queue{

数据对象：D={ai|ai∈ElemSet,i=1,2,……,n,n>=o}

数据关系：R={〈ai-1,ai>|ai-1,ai∈D,i=2,3,……,n}

基本操作：

Initqueue(linkqueue &q)

操作结果：构造一个空的队列 q；

Initlist(linklist &ev)

操作结果：构造一个空的链表 ev；

push(queue &q,int d)

初始条件：队列数组已存在

操作结果：在队尾插入一个元素 e

pop(queue &q)

初始条件：队列已存在

操作结果：获取队列 q 的队头

service\* front(queue &q)

初始条件：队列已存在

操作结果：返回队首元素

service\* back(queue &q)

初始条件：队列已存在

操作结果：返回队尾元素

service\* searchAndDel(queue &q,int m)

初始条件：队列已存在

操作结果：在队里寻找可处理元素

void putMoney()

操作结果：处理存款事件

void getMoney()

操作结果：处理取款事件

void arrive()

初始条件：初始化操作完毕

操作结果：处理客户到达事件

｝ADT

2．本程序包含五个模块：①主程序模块：Void main(){ 接受命令； 处理命令； }

②事件链表表单元模块——实现链表的抽象数据类型；

③队列单元模块——实现队列的抽象数据类型；

④事件结点结构单元模块——定义链表的结点结构；

⑤队列结点结构单元模块——定义队列的结点结构；

各模块之间的调用关系如下：主程序模块—〉链表、队列表单元模块—> 队列、链表结点结构单元模块。

**三、详细设计**

#include<stdio.h>

#include<stdlib.h>

#include <string>

#include <time.h>

using namespace std;

int total; //初始时银行现存资金总额

int closeTime; //营业结束时间

int arriveTime; //两个到达事件之间的间隔上限

int dealTime; //客户之间交易的时间上限

int dealMoney = 5000; //交易额上限

int currentTime = 0; //当前时间

int totalTime = 0; //客户逗留总时间

int counter = 0; //客户总数

int number = 1; //初始客户序列号

struct service {

int num; //客户号

string type; //到达或离开

int beginTime;

int endTime;

int money; //正数为存款，负数为取款

service\* next;

};

struct queue { //队列

service\* head;

service\* rear;

};

void push(queue &q,int d) {// 插入元素d为Q的新的队尾元素

service\* temp = new service;

temp->money = d;

temp->next = NULL;

if(NULL == q.head) {//队列为空，初始化

q. head = temp;

q. rear = temp;

}//if

else {//队列不为空，插入元素d

q. rear->next = temp;

q. rear = q.rear->next;

}//else

}

void pop(queue &q) {// 若队列不空,出队列函数

service\* temp;

temp = q. head;

if(NULL ==q. head->next)

q.head = q. rear =NULL;

else

q. head=q. head->next;

delete temp;

}

service\* front(queue &q) {//返回队首元素

return q. head;

}

service\* back(queue &q) {//返回队尾元素

return q. rear;

}

service\* searchAndDel(queue &q,int m) {//在对列中寻找可处理元素

service\* sign = q. head; //标记头节点

service\* temp;

while(NULL != q. head) {

if((-(q. head->money)) <m) {//队首元素可以处理

if(q. head==q.rear) {

temp = q. head;

q. head = q. rear = NULL;

return temp;

}

else {//队首元素出列

temp = q. head;

q. head = q. head->next; // 首节点后移一位,返回原首节点

return temp;

}

}

else {//队首元首不能被处理

if(q. head == q. rear){}

else {//首节点移到队列尾部

q. rear->next = q. head;

q. rear = q. rear->next;

q. head =q. head->next;

q. rear->next = NULL;

}

}

if(q. head == sign)//队列循环一周时停止

return NULL;

}

return NULL;

}

bool state =1; //用于判断是否有窗口在处理

int currentTimeOfDeal = 0;

int theArriveTime = 0;

queue eq; //事件队列

queue fq; //队列一

queue sq; //对列二

//初始化三个队列

void arrive() {/\*"到达"函数 随机产生顾客，进入队列一

产生到达事件 进入事件队列\*/

push(fq,(rand()% (2\*dealMoney) -dealMoney)); //随机产生顾客加入第一队列

back(fq)->beginTime = currentTime;

back(fq)->num = number;

push(eq,(back(fq)->money)); //将产生事件加入事件队列

back(eq)->beginTime = currentTime;

back(eq)->type = "到达";

back(eq)->num = number;

++number;

}

void putMoney() { //存款函数

total += front(fq)->money; //更新资金总额

push(eq,front(fq)->money); //加入事件队列 离开

back(eq)->type = "离开";

back(eq)->num = front(fq)->num;

back(eq)->endTime = (front(fq)->beginTime + rand()%dealTime +1);

++counter; //更新客户总数

totalTime += (back(eq)->endTime - front(fq)->beginTime); //更新逗留时间

pop(fq); //删除第一队列第一个业务

currentTimeOfDeal = back(eq)->endTime;

state =0;

}

void getMoney() {//取款函数

if( (-fq.head->money) > total ) {//资金短缺 加入第二队列

push( sq,front(fq)->money );

back(sq)->beginTime = front(fq)->beginTime;

back(sq)->num = front(fq)->num;

pop(fq);

}

else {

total += back(fq)->money;

push(eq,front(fq)->money); //加入事件队列 离开

back(eq)->type = "离开";

back(eq)->num = front(fq)->num;

back(eq)->endTime = (front(fq)->beginTime + rand()%dealTime +1);

back(eq)->beginTime = 0;

currentTimeOfDeal = back(eq)->endTime;

++counter; //更新客户总数

totalTime += ( back(eq)->endTime - back(fq)->beginTime ); //更新逗留时间

pop(fq); //删除第一队列第一个业务

state =0;

}

}

service\* temped ;

int randomTemp;

void findAndDeal() {//在对列中寻找可处理元素，对其进行处理

while( (temped= searchAndDel(sq,total))&&NULL != temped ) {//查找可处理取款

total += temped->money; //更新资金总额

push(eq,temped->money); //加入事件队列训 离开

back(eq)->type = "离开";

back(eq)->num = temped->num;

randomTemp = rand()%dealTime +1;

back(eq)->endTime = currentTime + randomTemp ;

currentTimeOfDeal += randomTemp;

++counter; //更新客户总数

totalTime += ( back(eq)->endTime - temped->beginTime ); //更新逗留时间

delete temped; //删除节点

temped = NULL;

}

state = 0;

}

int main() {

printf("\n\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" \* \*\n");

printf(" \* 欢迎进入银行模拟系统 \*\n");

printf(" \* \*\n");

printf(" \* 1.开始模拟 0.退出 \*\n");

printf(" \* \*\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

int n;

printf("请输入你的选择：");

scanf("%d",&n);

while(n==1) {

srand(time(NULL)); //初始化随机函数

printf("输入银行的初始存款：\n");

scanf("%d",&total);

printf("输入银行的营业时间：\n");

scanf("%d",&closeTime);

printf("输入最大到达时间间隔：\n");

scanf("%d",&arriveTime);

printf("输入最大的处理时间：\n");

scanf("%d",&dealTime);

theArriveTime +=rand()%arriveTime + 1; //首次到达时间

while(currentTime < closeTime) {

++currentTime;

if( currentTimeOfDeal < currentTime ) currentTimeOfDeal = currentTime ;

if( currentTimeOfDeal == currentTime ) state = 1;

if( currentTime == theArriveTime ) {//到达事件

arrive();

theArriveTime +=rand()%arriveTime +1;

}

if( 1 == state && NULL != fq.head) {

if(fq.head->money >= 0) {

putMoney();

findAndDeal();

}

else

getMoney();

}

}

printf("客户逗留平均时间为: %d\n",totalTime/counter);

printf("银行当前余额: %d\n",total);

break;

}

return 0;

}

**四、调试分析**

1. 我开始没有处理系统时间， 即直接使用随机生成的客户到达时间，经查阅资料才知道，原来分钟用 ti/60 显示，秒用 ti%60 显示即可。

2. 变成过程中犯了一个极其简单的错误，把栈和队列竟然弄反了，在运行结果不对的情况下很难发现错误，花了我好长时间才找出来。

**五、测试结果**

