

## Project Title: Bank Deposit Simulation

### Requirement Analysis

Based on the Project I- class Queue, this project will accomplish a more complicated simulation program about 'Bank Deposit'. Assume that in a bank there is only one staff that is responsible for handling all the businesses, and a customer arrives in every 3-8 min. and takes deposits in turn. All the customers only apply for one business of 'Certificate Deposit'(定期存款) whose amount is from 1,000 to 50,000 Yuan with the limitation of integer. The staff spends 2-7min. for each deposit business. The table below is the annual interest rates for deposit before and after it's raised on Aug 19th last year.

Certificate Deposit	Annual Interest Rates for Deposit before It's Raised (%)	Annual Interest Rates for Deposit after It's Raised(%)
Three months	1.71	1.80
Half a year	2.07	2.25
One year	2.25	2.52
Two years	2.70	3.06
Three years	3.24	3.69
Five years	3.60	4.14

Please write a program to carry out the whole process that all the customers take deposit. And the requirement is listed as follows:

Read the references about how to create random tasks. You can use one second or a 'tigtag' on a clock to instead of one minute. The timing starts from zero.

Design Class SavingsAccount, it is required that in the class it saves

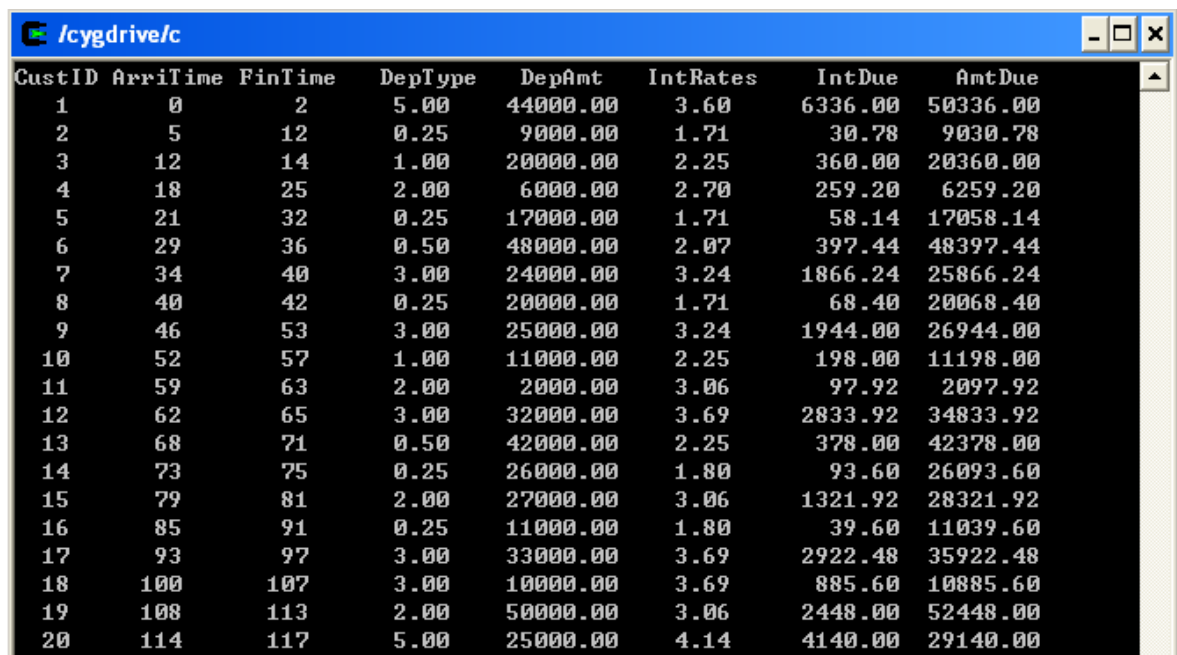
the business number of every deposit, the amount deposited, deposit type etc.

Class SavingsAccount should use the functions for calculating interest due, aggregate amount due (don't forget to deduct 20% interest tax), using inline function.

All kinds of interest rates need to be presented by constant in Class SavingsAccount, and you should design a function that can change interest rate to reflect its increase.

The program will simulate deposit cases of 20 customers: the first 10 customers will use the annual interest rates for deposit before it's raised, and the last 10 customers will use the annual interest rates for deposit after it's raised.

The outputs of program are the deposit event of every customer, please print them as the following order of the items:



CustID	ArrTime	FinTime	DepType	DepAmt	IntRates	IntDue	AmtDue
1	0	2	5.00	44000.00	3.60	6336.00	50336.00
2	5	12	0.25	9000.00	1.71	30.78	9030.78
3	12	14	1.00	20000.00	2.25	360.00	20360.00
4	18	25	2.00	6000.00	2.70	259.20	6259.20
5	21	32	0.25	17000.00	1.71	58.14	17058.14
6	29	36	0.50	48000.00	2.07	397.44	48397.44
7	34	40	3.00	24000.00	3.24	1866.24	25866.24
8	40	42	0.25	20000.00	1.71	68.40	20068.40
9	46	53	3.00	25000.00	3.24	1944.00	26944.00
10	52	57	1.00	11000.00	2.25	198.00	11198.00
11	59	63	2.00	2000.00	3.06	97.92	2097.92
12	62	65	3.00	32000.00	3.69	2833.92	34833.92
13	68	71	0.50	42000.00	2.25	378.00	42378.00
14	73	75	0.25	26000.00	1.80	93.60	26093.60
15	79	81	2.00	27000.00	3.06	1321.92	28321.92
16	85	91	0.25	11000.00	1.80	39.60	11039.60
17	93	97	3.00	33000.00	3.69	2922.48	35922.48
18	100	107	3.00	10000.00	3.69	885.60	10885.60
19	108	113	2.00	50000.00	3.06	2448.00	52448.00
20	114	117	5.00	25000.00	4.14	4140.00	29140.00

There is no input and the output could probably different each time the program gets executed. However, the format should remain the same.

## 1. Design Description

SavingsAccount
-id -type -arriveTime -lasting -finishTime -amount -rate -interestDue -amountDue -raised
+SavingsAccount(int,int,int,int) +SavingsAccount() +opreator<<(ostream&, SavingsAccount) +raise() +process() +finishat(int)

Queue
-data[] -size -served
+Queue() +enqueue(SavingsAccount& sa) +opreator<<(ostream&, SavingsAccount) +raise() +process() +finishat(int) +dequeue() +noMoreClients() +isEmpty() +process(int)

The main part of the program is a loop in which a countdown for 3-8 minutes (randomly chosen) ticks first to simulate the arrival of a new customer, then the bank stuff does his work of this minute and if a certain deposit is finished, the information of that customer will be printed on the screen, then the program wait for a second instead of a minute to represent that minute in reality.

## 2. Debug Analysis

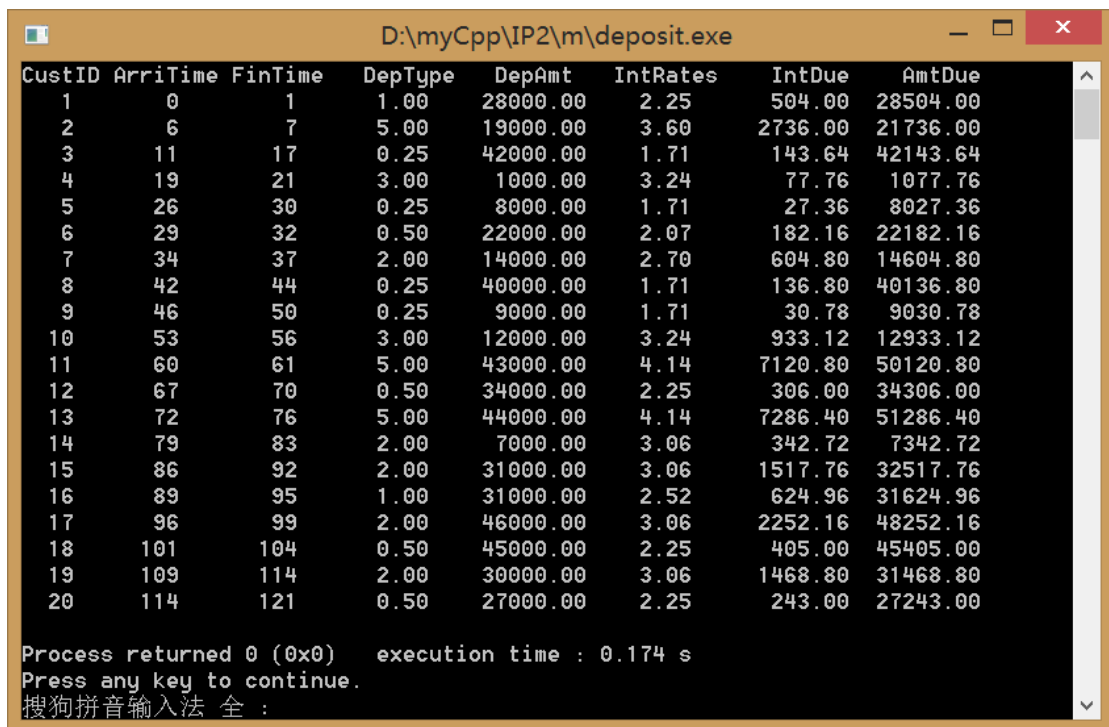
1. All the code should be written under the condition of 'using namespace std;'

2. Only if the header file, its actualization file and the main function have been correctly associated will the program operates smoothly. And to achieve this, 'deposit.cpp' should include 'queue.h' and 'queue.cpp' should be included by 'queue.h'.

3. The classes and constants should not be redefined.

## 3. Test Result

a )



CustID	ArrTime	FinTime	DepType	DepAmt	IntRates	IntDue	AmtDue
1	0	1	1.00	28000.00	2.25	504.00	28504.00
2	6	7	5.00	19000.00	3.60	2736.00	21736.00
3	11	17	0.25	42000.00	1.71	143.64	42143.64
4	19	21	3.00	1000.00	3.24	77.76	1077.76
5	26	30	0.25	8000.00	1.71	27.36	8027.36
6	29	32	0.50	22000.00	2.07	182.16	22182.16
7	34	37	2.00	14000.00	2.70	604.80	14604.80
8	42	44	0.25	40000.00	1.71	136.80	40136.80
9	46	50	0.25	9000.00	1.71	30.78	9030.78
10	53	56	3.00	12000.00	3.24	933.12	12933.12
11	60	61	5.00	43000.00	4.14	7120.80	50120.80
12	67	70	0.50	34000.00	2.25	306.00	34306.00
13	72	76	5.00	44000.00	4.14	7286.40	51286.40
14	79	83	2.00	7000.00	3.06	342.72	7342.72
15	86	92	2.00	31000.00	3.06	1517.76	32517.76
16	89	95	1.00	31000.00	2.52	624.96	31624.96
17	96	99	2.00	46000.00	3.06	2252.16	48252.16
18	101	104	0.50	45000.00	2.25	405.00	45405.00
19	109	114	2.00	30000.00	3.06	1468.80	31468.80
20	114	121	0.50	27000.00	2.25	243.00	27243.00

Process returned 0 (0x0) execution time : 0.174 s  
Press any key to continue.  
搜狗拼音输入法 全 :

b )

```

D:\myCpp\IP2\m\deposit.exe
CustID ArrTime FinTime DepType DepAmt IntRates IntDue AmtDue
1 0 4 2.00 46000.00 2.70 1987.20 47987.20
2 7 10 3.00 38000.00 3.24 2954.88 40954.88
3 14 17 1.00 17000.00 2.25 306.00 17306.00
4 17 21 5.00 31000.00 3.60 4464.00 35464.00
5 21 25 0.50 31000.00 2.07 256.68 31256.68
6 25 27 0.25 7000.00 1.71 23.94 7023.94
7 28 31 0.25 39000.00 1.71 133.38 39133.38
8 32 34 3.00 23000.00 3.24 1788.48 24788.48
9 39 43 0.25 18000.00 1.71 61.56 18061.56
10 47 49 0.50 17000.00 2.07 140.76 17140.76
11 55 58 5.00 43000.00 4.14 7120.80 50120.80
12 60 61 0.50 19000.00 2.25 171.00 19171.00
13 67 68 2.00 27000.00 3.06 1321.92 28321.92
14 71 75 0.50 32000.00 2.25 288.00 32288.00
15 74 78 1.00 1000.00 2.52 20.16 1020.16
16 78 85 2.00 49000.00 3.06 2399.04 51399.04
17 83 89 0.50 20000.00 2.25 180.00 20180.00
18 87 93 0.50 6000.00 2.25 54.00 6054.00
19 94 95 2.00 4000.00 3.06 195.84 4195.84
20 98 101 0.50 10000.00 2.25 90.00 10090.00

Process returned 0 (0x0) execution time : 0.080 s
Press any key to continue.
搜狗拼音输入法 全 :

```

## 4. Appendix

All the files are packed together with this report in the compressed files. However, the code as well as annotation is still provided in this report in case.

```

[queue.h]
//File name:queue.h
//Header file of queue library
#ifndef QUEUE
#define QUEUE
#include <iostream>
using namespace std;
const double Rates[2][6]={ { 1.71,2.07,2.25,2.70,3.24,3.60},{ 1.80,2.25,2.52,3.06,3.69,4.14 } };
//Different rates
const double Types[6]={0.25,0.5,1,2,3,5}; //Different maturities
class SavingsAccount
{
private:
    int id,type,arriveTime,lasting,finishTime;
    double amount,rate,interestDue,amountDue;
    static int raised;
public:
    SavingsAccount(int ID,int AMOUNT,int TYPE,int ATIME);
    SavingsAccount():id(0),amount(0),type(1),arriveTime(0){ }
    friend ostream& operator<<(ostream& os,const SavingsAccount& sa);
    void raise(){raised=1;}
    bool process();
    void finishat(int FTime){finishTime=FTime;}
}; //To simulate a single client

```

```

class Queue
{
private:
    SavingsAccount data[20];
    int size,served;
public:
    Queue(){size=0;served=0;}
    void enqueue(const SavingsAccount& sa);
    SavingsAccount dequeue();
    bool noMoreClients(){return (served>=20);}
    bool isEmpty(){return (size==0);}
    void process(int currentTime);
}; //To simulate a queue of clients
int random(int inf,int sup); //To generate a random number between inf and sup (both
included)
#include "queue.cpp"
#endif

```

```

[queue.cpp]
//file name:queue.cpp
//This file realizes the queue library.
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;
int random(int inf,int sup)
{
    return inf+(rand()*(sup-inf+1)/(RAND_MAX+1));
} //To generate a random number between inf and sup (both included)
inline SavingsAccount::SavingsAccount(int ID,int AMOUNT,int TYPE,int ATIME)
{
    id=ID;
    amount=AMOUNT;
    type=TYPE;
    arriveTime=ATIME;
    lasting=random(2,7);
    rate=Rates[raised][type];
    interestDue=amount*0.008*rate*Types[type];
    amountDue=amount+interestDue;
} //Initialize the client's session
bool SavingsAccount::process()
{
    lasting-=1;
    return (lasting==0);
} //To simulate the work of the stuff
void Queue::process(int currentTime)
{

```

```

        if (data[0].process())
        {
            data[0].finishat(currentTime);
            ++served;
            cout<<dequeue();
        }
    } //To simulate the work of the stuff
void Queue::enqueue(const SavingsAccount& sa)
{
    if (size<20)
    {
        data[size]=sa;
        ++size;
    }
} //Add a client into the queue
SavingsAccount Queue::dequeue()
{
    SavingsAccount temp;
    if (size>0)
    {
        temp=data[0];
        int i;
        for (i=0;i<size-1;++i) data[i]=data[i+1];
        --size;
    }
    return temp;
} //To remove the first client in the queue and return it
ostream& operator<<(ostream& os,const SavingsAccount& sa)
{
    os<<setw(4)<<sa.id<<setw(8)<<sa.arriveTime<<setw(9)<<sa.finishTime<<setw(10)<<Types[sa.type]<<setw(12)<<sa.amount<<setw(8)<<sa.rate<<setw(12)<<sa.interestDue<<setw(10)<<sa.amountDue<<endl;
    return os;
} //Out put a line that has been formatted

```

[deposit.cpp]

//File name:deposit.cpp

//This program tests the application of the class Queue.

#include <iostream>

#include "queue.h" //To utilize classes and fuctions in the Queue library

#include "windows.h" //To use the function Sleep()

using namespace std;

int SavingsAccount::raised=0; //The interest rate has not been raised yet

int main()

```

{
    srand((unsigned)time(NULL)); //To initialize the random number generator
    cout<<"CustID  ArriTime  FinTime    DepType    DepAmt    IntRates    IntDue
AmtDue\n";
    cout.precision(2);

```

```

cout<<fixed; //To make the output formatted
Queue clients;
int arrival=0,timing=0,count=0;
SavingsAccount client;
for (timing=0;(!((count>=20)&&(clients.isEmpty())));++timing)
{
    if ((arrival==0)&&(count<20))
    {
        ++count;
        if (count==11) client.raise();
        client=SavingsAccount(count,random(1,50)*1000,random(0,5),timing);
        clients.enqueue(client);
        arrival=random(3,8);
    }
    if (!clients.isEmpty()) clients.process(timing);
    if (arrival>0) --arrival;
    Sleep(1000);
}
}

```