银行排队系统实验报告

问题描述

程序能够完成以下功能：首先随机生成85-115名顾客，以及每名顾客的名字、到达时间、办理业务所需时间、忍耐时间、是否为VIP等内容.然后模拟一个单队列、多窗口的银行排队系统一天的运行情况，窗口数量在控制台输入.接着选择是否输出各窗口运行时的详细信息，最后，统计VIP顾客和普通顾客的平均等待时间、窗口占用率、顾客离开率等并输出.

实验内容

程序中主要使用的数据结构是队列，在头文件 queue 中定义.

用户信息的储存方式为:

```
typedef struct {
  int VIP;
  string name;
  int arr_time;
  int ser_time;
  int tol_time;
  int cur_tol_time;
} customer;
```

其中VIP为0时为普通用户，VIP为1时为VIP用户，arr_time代表到达时间，ser_time代表办理业务所需时间，tol_time代表最大容忍时间，cur_tol_time代表顾客已经等待的时间.

窗口信息的存储方式为:

```
enum status { idle, busy };
typedef struct {
  status cur_status;
  int cur_start_time;
  int cur_customer;
  int cur_cust_type;
} window;
```

其中cur_status代表窗口是空闲还是工作中，cur_start_time代表当前状态开始的时间，cur_customer代表当前窗口的顾客，cur_cust_type代表当前窗口顾客的类型，即是否为VIP.

程序中的主要函数有：

```
void Window_init(window *&, int);
void Time_update(int &, int &);
void End_Serve(window *&, int, int);
void Serve(queue<int> &, window *&, int, customer *, int);
void Wait_tolerance(queue<int> &, customer *&, int &, int &);
void Print_win(window *, customer *, queue<int>, queue<int>, int );
void Percent_print(double );
```

1.窗口初始化函数： 函数原型为 `void Window_init(window *&win, int num)` ，输入为窗口数组 `win` 和窗口数量 `num` .

算法流程：将每个窗口的信息进行初始化，cur_status置为idle，cur_customer置为-1，cur_start_time置为0，cur_start_type置为-1.

算法的时间复杂度为O(num),空间复杂度为O(1).

2.时间更新函数： 函数原型为 `void Time_update(int &h, int &m)` ，输入为当前时间，返回更新后的时间.

算法流程：将输入的分钟加1，如果等于60则进位，之后将时间按照 "hh:mm" 的格式输出.

算法的时间复杂度为O(1),空间复杂度为O(1).

3.结束服务函数：

均为随机生成，名字为3-6位，到达时间相邻顾客控制在0-15分钟之内，办理业务所需时间为10-35分钟，忍耐时间为5-35分钟，VIP率为25%，