

TADLineaCajas.pdf



Anónimo



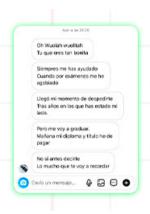
Análisis de Algoritmos y Estructuras de Datos



2º Grado en Ingeniería Informática



Escuela Superior de Ingeniería Universidad de Cádiz



Que no te escriban poemas de amor cuando terminen la carrera

(a nosotros por

(a nosotros pasa)

WUOLAH

Suerte nos pasa)







No si antes decirte Lo mucho que te voy a recordar

(a nosotros por suerte nos pasa)

TAD Linea Cajas

```
// 50 cajas 1-50
//numero cajas abiertas variable
// siempre hay entre 10-50 cajas en funcionamiento (depende afluencia de la gente)
// la media de cajas abiertas es 48 -> al dia
// 2 turnos de trabajo al dia
//cada turno suficientes cajeros demanda máxima + suplentes
// de cada caja :
// - num de caja
// - id cajero (tres digitos)
// - facturación desde el comienzo del turno
//definicion : solo se podra utilizar
class LineaCajas{
public:
  //postCondicion: crea una linea de cajas cerradas en el turno de mañana
  LineaCajas();
  // postCondición: abre una caja determinada del hipermercado,
  // (si ya esta el numero maximo de cajas abiertas no se abre ninguna --> realmente no
puedo abrir mas porque tengo que solo hy 50 cajas)
  bool abrirCaja (int numeroCaja, int idCajero[3]);
  //postCondición: cierra una caja determinada si hay más de 10 cajas abiertas y devuelve
la facturación desde que se abrió por última vez
  // si devuelve -1 la caja ya estaba cerrada de antes
  float cerrarCaja( int numeroCaja);
  //preCondicion : la caja debe de estar abierta de antemano (cerrarCaja != -1)
  //postCondición : se añade el importe cobrado al cliente a la facturacion de la caja en la
que se cobra
  void cobrarCliente (int numeroCaja, float importe);
  //preCondicion: el cajero no puede estar asignado ya a otra caja, si está ya asignado, el
cajero no se sustituirá
  //postCondición: se actualiza el id nuevo cajero por el que se sustituye
  void sustituirCajero(int numeroCaja, int idCajeroNuevo[3]); //continua funcionando
```

```
//postCondicion: devuelve la recaudacion de cajas total del turno, cierra todas las cajas
v cambia de turno
  float cambiarTurno();
  //postCondicion: devuelve la recaudación total del dia en el momento del cierre, cierra
todas las cajas, resetea la facturacion a 0,
  //y pone a 0 tambien el id del Cajero
  float cerrarCajas();
private:
  typedef struct {
    int numCaja, idCajero[3];
    bool abierta = false ; //caja cerrada false, caja abierta true
    float facturación = -1; // si la caja esta cerrada es -1, si esta abierta mayor o igual que
0
  }caja;
  char turnoDelDía ; // m-> mañana , t -> tarde
  lista<cajas> cajasTurnoActual(50); // cajasTurnoTarde(50); //estatica -> media 48 cajas
  float facturaTotalMañana, facturaTotalTarde;
  int numCajasAbiertas;
};
LineaCajas::LineaCajas(){
  turnoDelDía = 'm';
  numCajasAbiertas=0;
  facturaTotalMañana = 0;
  facturaTotalTarde = 0;
}
void LineaCajas::abrirCaja(int numeroCaja, int cajero[3]){
  if(numCajasAbiertas < 50){
    for(int i=0;i<3;i++){}
       cajasTurnoActual.elemento(numeroCaja-1).idCajero[i] = cajero[i];
    }
    cajasTurnoActual.elemento(numeroCaja-1).abierta = true;
    cajasTurnoActual.elemento(numeroCaja-1).facturacion = 0;
    numCajasAbiertas++;
  }
```





(a nosotros por suerte nos pasa)

Ayer a las 20:20

Oh Wuolah wuolitah Tu que eres tan bonita

Siempres me has ayudado Cuando por exámenes me he agobiado

Llegó mi momento de despedirte Tras años en los que has estado mi lado.

Pero me voy a graduar. Mañana mi diploma y título he de pagar

No si antes decirte Lo mucho que te voy a recordar













```
}
//postCondición: cierra una caja determinada si hay más de 10 cajas abiertas y devuelve la
facturación desde que se abrió por última vez
// si devuelve -1 la caja ya estaba cerrada de antes o no ha podido cerrarse.
float LineaCajas::cerrarCaja(int numeroCaja){
  float facturacionActualDeEstaCaja = -1;
  if(numCajasAbiertas > 10) {
    facturacionActualDeEstaCaja = cajasTurnoActual.elemento(numeroCaja -
1).facturacion;
    if(facturacionActualDeEstaCaja > -1){
       cajasTurnoActual.elemento(numeroCaja - 1).abierta = false;
       cajasTurnoActual.elemento(numeroCaja - 1).facturacion = -1;
       if (turnoDelDia == 'm') facturaTotalMañana += facturacionActualDeEstaCaja;
       else facturaTotalTarde += facturacionActualDeEstaCaja;
       numCajasAbiertas--;
    }
  }
  return facturacionActualDeEstaCaja;
//preCondicion : la caja debe de estar abierta de antemano (cerrarCaja != -1)
//postCondición : se añade el importe cobrado al cliente a la facturacion de la caja en la
que se cobra
void LineaCajas::cobrarCliente (int numeroCaja , float importe){
  assert(cajasTurnoActual.elemento(numeroCaja - 1).abierta);
  cajasTurnoActual.elemento(numeroCaja - 1).facturacion += importe;
}
//preCondicion: el cajero no puede estar asignado ya a otra caja, si está ya asignado, el
cajero no se sustituirá
//postCondición: se actualiza el id nuevo cajero por el que se sustituye
void LineaCajas::sustituirCajero(int numeroCaja, int idCajeroNuevo[3]){
  bool mismoCajero;
  int contadorCaracteresCajero;
  int i=0;
```







No si antes decirte Lo mucho que te voy a recordar

(a nosotros por suerte nos pasa)

```
while(!mismoCajero && i<50){
    if(cajasTurnoActual.elemento(i).abierta){
      mismoCajero = false;
      contadorCaracteresCajero = 0;
      for(int j=0; j<3; j++){
        if(cajasTurnoActual.elemento(i).idCajeroNuevo[j] == idCajeroNuevo[j])
contadorCaracteresCajero++;
      }
      if (contadorCaracteresCajero == 3) mismoCajero = true;
  }
  if(!mismoCajero){
    for(int i=0;i<3;i++){}
      cajasTurnoActual.elemento(numeroCaja-1).idCajero[i] = idCajeroNuevo[i];
  }
}
//postCondicion: devuelve la recaudacion de cajas total del turno, cierra todas las cajas y
cambia de turno
float LineaCajas::cambiarTurno(){
  float facturacionTurno = 0;
  for(int i=0;i<50;i++){
    if (cajasTurnoActual.elemento(i).abierta) {
      facturacionTurno += cajasTurnoActual.elemento(i).facturacion;
      cajasTurnoActual.elemento(i).abierta =false;
      cajasTurnoActual.elemento(i).facturacion =-1
    }
  }
  if(turnoActual == 'm') {
    facturaTotalManana = facturacionTurno;
    turnoActual = 't';
  }
    facturaTotalTarde = facturacionTurno;
```



```
turnoActual = 'm';
}

return facturacionTurno;
}

//postCondicion: devuelve la recaudación total del dia en el momento del cierre, cierra todas las cajas, resetea la facturacion a 0,
//y pone a 0 tambien el id del Cajero
float LineaCajas::cerrarCajas(){
  float recaudacionDelDia = ((turno == 'm') ? (cambiarTurno() + facturaTotalTarde) :
(cambiarTurno() + facturaTotalMañana));
  facturaTotalMañana = facturaTotalTarde = 0;
  return recaudacionDelDia;
}
```

