

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
suerte nos pasa)*

WUOLAH

WUOLAH

Oh Wuolah wuolilah
Tu que eres tan bonita

```
//postCondicion: devuelve la recaudacion de cajas total del turno, cierra todas las cajas
y cambia de turno
float cambiarTurno();
```

```
//postCondicion: devuelve la recaudación total del dia en el momento del cierre, cierra
todas las cajas, resetea la facturación a 0,
//y pone a 0 también 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).facturación = 0;

        numCajasAbiertas++;
    }
}
```

**Que no te escriban poemas de amor
cuando terminen la carrera ▶▶▶▶▶▶▶▶**
(a nosotros por suerte nos pasa) 😊



WUOLAH



```

}

//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;

```

WUOLAH

Oh Wuolah wuolithah
Tu que eres tan bonita

```
        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;
}
```