Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos A
Ing. Luis Fernando Espino
Aux. Robinson Jonathan Perez Fuentes

Práctica 1: Manual Técnico
Segundo semestre 2017, Vacaciones Diciembre

Nombre: Osmel David Tórtola Tistoj
Carné: 201404218

Guatemala, viernes 15 de diciembre de 2017.

Sobre “Práctica 1”

El proyecto “Práctica 1” se trata de un simulador de procesos y actividades en un aeropuerto, así como el manejo de la información respectiva a estos procesos. El proyecto está desarrollado en el lenguaje C++, C y utiliza librerías de Qt.

Este simulador cuenta con opciones para poder visualizar la forma en que los aviones al aterrizar van desabordando los pasajeros, luego estos pasan al checkeo de sus documentos para finalmente tomar sus maletas y retirarse del sistema. Los aviones después de bajar todos sus pasajeros pasan a una estación de mantenimiento para posteriormente realizar otro vuelo.

El sistema da opción de manejar la cantidad de escritorios de atención a los pasajeros que hay en la simulación, así como estaciones de servicio a los aviones y también la cantidad de aviones que aterrizarán en esta simulación.

La información de la simulación es mostrada por medio de gráficas que se actualizan en tiempo real, así como también por medio de texto en una consola que lleva una bitácora de lo sucedido en cada momento.

Practica 1 (Simulador)

Manejo de la información:

El proyecto está realizado utilizando como base el comando *struct* de C/C++, y el principal objetivo es almacenar toda la información en tiempo real utilizando estructuras de datos dinámicas.

Cada estructura cumple un papel diferente dentro de la aplicación y entre ellas se comunican información.

Las estructuras de datos utilizadas en el proyecto son:

- Lista simple
- Lista doblemente enlazada y ordenada
- Lista doblemente enlazada circular
- Cola simple
- Cola doblemente enlazada
- Pila

La estructura del proyecto consiste en que cada estructura cuenta con su respectivo archivo header (.h) y clase C++ (.cpp) en la cual contiene todos sus objetos y métodos respectivos de ella, así como la importación de clases de otras estructuras si es que esta las necesita.

Luego, la parte principal del sistema se encuentra en el archivo mainwindow.cpp, la cual funciona como la parte fundamental del simulador al hacer todas las instanciaciones de las estructuras, así como contener todos los métodos necesarios para hacer que la simulación funcione. También cuenta con los métodos para generar las imágenes y cargarlas a la interfaz.

Código utilizado para la declaración de la cola simple (header);

```
typedef struct ColaSimple ColaSimple;
typedef struct csNodo csNodo;
typedef struct Pasajero Pasajero;

struct ColaSimple{
    csNodo * primero;
    csNodo * ultimo;
    int length;
};

struct csNodo{

    csNodo * siguiente;
    Pasajero *pasajero;

};
struct Pasajero{

    int id;
    int maletas;
    int documentos;
    int numeroTurnos;
    int avion;

};

int crearCola(ColaSimple * cola);
int queue(ColaSimple * cola, Pasajero *pasajero_);
int dequeue(ColaSimple * cola);
int esVacia(ColaSimple * cola);
Pasajero *primero(ColaSimple * cola);
int getSize(ColaSimple * cola);
QString escribirDOT(ColaSimple * cola);
Pasajero * crearPasajero(int id_, int avion_);
```

Esta clase se utiliza para generar una cola simple, que se utiliza para almacenar a los pasajeros que esperan para ser atendidos.

Código para la implementación de la lista simple:

```
typedef struct ListaSimple ListaSimple;
typedef struct lNodo lNodo;

struct sNodo{

    int id;
    sNodo * siguiente;
    Avion *avion;

};

struct ListaSimple{

    sNodo * primero;
    sNodo * ultimo;
    int length;

};

int crearLista(ListaSimple * lista);
int insertar(ListaSimple * lista, int id);
int eliminar(ListaSimple * lista);
QString escribirDOT(ListaSimple * lista);
int crearEstaciones(ListaSimple * lista, int cantidad);

QString escribirInformacion(ListaSimple * lista);
```

Esta lista es utilizado para contener las estaciones de servicio de los aviones, teniendo métodos para recibir aviones y para ir descontando su número de turnos restantes para posteriormente eliminarlos del sistema.

Código del header (.h) de la clase Lista doblemente enlazada:

```
typedef struct ListaDoblementeEnlazada ListaDoblementeEnlazada;
typedef struct ldNodo ldNodo;
typedef struct Escritorio Escritorio;

struct ListaDoblementeEnlazada{

    ldNodo * primero;
    ldNodo * ultimo;
    int length;
    int numeroEscritorios;


};

struct ldNodo{

    ldNodo * siguiente;
    ldNodo * anterior;
    Escritorio * escritorio;

};

struct Escritorio{

    char id;
    ColaSimple * cola;
    Pila * pilaDocumentos;


};

int crearLista(ListaDoblementeEnlazada * lista);
int insertar(ListaDoblementeEnlazada * lista, Escritorio * escritorio);
int eliminar(ListaDoblementeEnlazada * lista);
int esVacia(ListaDoblementeEnlazada * lista);
int getSize(ListaDoblementeEnlazada * lista);
QString escribirDOT(ListaDoblementeEnlazada * lista);
Escritorio * crearEscritorio(ListaDoblementeEnlazada * lista, char id_);
int crearEscritorios(ListaDoblementeEnlazada * lista, int cantidad);
int ingresar(ColaSimple *cola, Pasajero * pasajero);
int espaciosVacios(ListaDoblementeEnlazada * lista);
QString escribirInformacion(ListaDoblementeEnlazada * lista);
```

Esta estructura se utiliza para almacenar las ventanillas de atención a los pasajeros después de desabordar.

Código de la clase Lista Doble Circular:

```
typedef struct ListaDobleCircular ListaDobleCircular;
typedef struct ldcNodo ldcNodo;

struct ListaDobleCircular{

    ldcNodo * primero;
    ldcNodo * ultimo;
    int length;
    int contador;

};

struct ldcNodo{

    ldcNodo * siguiente;
    ldcNodo * anterior;
    int id;

};

int crearLista(ListaDobleCircular * lista);
int insertar(ListaDobleCircular * lista);
int eliminar(ListaDobleCircular * lista);
int esVacia(ListaDobleCircular * lista);
QString escribirDOT(ListaDobleCircular * lista);
```

Esta clase es la encargada de almacenar las maletas de los pasajeros, así como eliminarlos cuando el pasajero se retira del sistema.

Métodos principales dentro de la aplicación:

```
//Métodos
int graficar();
int escribirEnConsola(QString cadena);
int desabordaje(ColaDoblementeEnlazada * cola);
int registrarPasajeros();
int atender();
int darMantenimiento();
```

Estos métodos encontrados en el archivo mainwindow.h y mainwindow.cpp son los principales motores de la simulación y se describen a continuación:

Graficar: Este método inicia escribiendo la estructura básica de un archivo en lenguaje graph, para posteriormente llamar al método “escribirDOT” respectivo de cada una de las estructuras, el cual devuelve el contenido de cada estructura en formato graph, luego este método junta todo en un Qstring y lo escribe en un fichero .txt, ejecuta el comando dot y luego muestra la imagen generada en la interfaz.

Escribir en consola: Este método recibe un texto y lo envía directamente a la consola, en el botón turno se llama a este método para mostrar información en tiempo real en la consola.

Desabordaje: Este método reduce el contador de turnos de desabordaje del avión actual para luego generar todos los pasajeros en una cola.

Registrar pasajeros: Este método crea los pasajeros para luego ser atendidos.

Atender: Este método reduce el contador respectivo de cada personaje en cada ventanilla de atención.

Dar mantenimiento: Se encarga de ir reduciendo el contador de turnos de los aviones en las ventanillas de servicio para posteriormente eliminarlos del sistema.

Generación de gráficas:

Las gráficas del sistema se generan utilizando el comando dot de la aplicación *Graphviz*, el sistema recorre las estructuras de datos y escribe en forma de texto plano el contenido de ellas en el lenguaje graph, para luego ejecutar el comando dot sobre el fichero de texto, así generando la imagen para posteriormente cargarla a la aplicación utilizando un label y un scrollArea.

Ejemplo de un método utilizado para escribir el fichero .txt:

```
QString texto = "subgraph cluster_1 { ";
texto += "label = \"Desabordaje\";\n";

//PARA DECLARAR LAS CABECERAS Y LABELS
if(!esVacia(cola)){
    csNodo * aux = cola->primero;
    while (aux != NULL){
        texto += "\"Pasajero " +QString::number(aux->pasajero->id) +"\"";
        texto += "[label=\"{<f0>Pasajero: " +QString::number(aux->pasajero->id) +"|";
        texto += "<fi>Avion: " +QString::number(aux->pasajero->avion) + "|";
        texto += "<f2>Maletas: " +QString::number(aux->pasajero->maletas) +"|";
        texto += "<f3>Documentos: " +QString::number(aux->pasajero->documentos) +"|";
        texto += "<f4>Turnos: " +QString::number(aux->pasajero->numeroTurnos) +"";
        texto += "}\" shape=record];\n";
        aux = aux->siguiente;
    }
}
```

Ejemplo de un archivo generado con graphviz: