Simulación De Un Auto-Lavado

Universidad Nacional Experimental de Guayana
Técnicas De Programación II
Jesús David Pérez
E-mail: jesusdavid2405 @gmail.com
Noviembre 14 del 2014
Ciudad Guayana, Edo. Bolívar

Resumen

El Código fuente ha sido desarrollado en Dev-C++ bajo Lenguaje C, contiene 7 archivos o códigos externos que han sido generados para simplificar a máxima escala las líneas de código en la función principal **main** del programa y su vez hacer más legible el código del programa, sin embargo cada archivo cumple funciones esenciales y muy importantes para la ejecución del mismo.

Palabras Claves: Código Fuente, Dev-C++, Archivos.

1. Introducción

Un auto lavado es un establecimiento dedicado a la limpieza de carros, el equipamiento de los mismos puede incluir varias modalidades en este caso:

1) Lavado y Aspirado.

2) Cambio de Aceite.

3) Lavado de Motor o Chasis.

En este establecimiento de servicios con una capacidad para cinco (5) puestos, que está rodeado por una pared excepto un lado por donde entran y salen los vehículos, el cual se comporta como una pila donde el ultimo carro en entrar es el primero en retirarlo.

Planteamiento del problema

- Elaborar una simulación de un Auto Lavado en el cual dentro de sus instalaciones funcionan tres servicios:

1) Lavado y Aspirado.
2) Cambio de Aceite.
3) Lavado de Motor o Chasis.

Donde primeramente los clientes hacen una cola en la entrada del local para ser atendidos, una vez adentro y de acuerdo a la necesidad de cada cliente por el tipo de servicio, estos son remitidos posteriormente a otra cola. El establecimiento cuenta con cuatro puestos simultáneos para el primer servicio y para los dos últimos servicios un puesto para cada uno. Por otra parte, posee un estacionamiento con capacidad de 5 puestos, el funcionamiento de éste es el comportamiento de una pila (LIFO), las operaciones que se realizan es estacionar un auto y sacar un auto del estacionamiento.

No obstante, es muy importante mostrar, el tiempo y la animación de la simulación, haciendo énfasis en el comportamiento de las colas. La interfaz debe ser modo texto y el usuario debe ingresar el tiempo antes de

comenzar la simulación. Se deben Utilizar estructuras dinámicas (Listas simplemente enlazadas con centinela). Para finalizar, se debe mostrar unas estadísticas generales de la simulación (Cantidad de autos atendidos, atendidos por servicio, tipos de pago, etc)

3. Resolución del problema:

Se pretende elaborar la simulación de la siguiente manera:

- Tener archivos externos en los cuales organizadamente cada uno de ellos posee líneas de código esenciales donde posteriormente les hacemos el llamado en el código fuente principal.
- Elaborar 3 listas simplemente enlazadas, las cuales representan cada servicio (Lavado y Aspirado, Cambio de Aceite, Lavado de Motor).
- Elaborar 1 Estructura Centinela, en la cual dentro de ella se desea tener 3 punteros que apuntan al primer nodo de cada lista, de esta manera nuestra estructura centinela se enlaza con cada una de las listas.
- Definir una función de tipo **bool** la cual tenga la función aleatoria predeterminada **srand** y sólo la limitemos para que genere un número entre el 0 y el 1. Dicho valor lo trabajamos de forma esencial para el manejo de la lista principal, y decisiones importantes en el código, en la cual le hagamos la llamada "N" veces en cada una de las funciones. .
- Elaborar una función en la que se inicialicen todos nuestros punteros y variables a utilizar antes del inicio de la simulación.
- Elaborar una función en la cual luego de haber finalizado la simulación limpie o borre las listas que aún se encuentran en el programa.
- Para finalizar mostrar los créditos de dicho programa.

4. Contenido

El programa inicia con la petición del tiempo de duración de dicha simulación en minutos y segundos, posteriormente se válida que los minutos y segundos no sean negativos ni mayores de 60, una vez correcto el minuto y segundo se procede con la simulación. Por Otra parte, hay destaca la creación de los 7 archivos que cumplen con varias funciones y que a su vez forman parte de función principal del programa.

Estos Archivos los especificamos como:

- **"Simulacion_Const.cpp":**
  Archivo que contiene sólo las constantes de todo el programa.
- **"Simulacion_Estruct.cpp":**
  Archivo que contiene la estructura centinela llamada "Cabecera", listas y estructura para el manejo del estacionamiento.
- **"Simulacion_Animacion.cpp":**
  Archivo que contiene la animación principal.
- **"Simulacion_Time.cpp**":
  Archivo que contiene todas las funciones relacionadas con el tiempo de la simulación.
- **"Simulacion_FuncF.cpp":**
  Archivo que contiene todas las funciones segundarias o finales del programa.

- **"Simulacion_FuncP.cpp"**: Archivo que contiene todas las funciones principales del programa**.**

5. Marco teórico

Es importante señalar las librerías utilizadas en el código fuente de la simulación.

Con respecto a las librerías, se utilizaron las siguientes: <stdio.h>, <conio.h>, <stdlib.h>, <string.h>, <time.h>, <windows.h>.

Entre las funciones más utilizadas con las librerías ya nombradas, tenemos:

- ✓ System("color "); utilizada para cambiar el color de la pantalla y el color de las letras.
- ✓ System("cls"); para borrar o limpiar la pantalla.
- ✓ Printf (""); para imprimir cualquier mensaje por pantalla.
- ✓ Scanf (); para leer cualquier dato y almacenarlo en la variable correspondiente.
- ✓ Getch(); para pausar la pantalla recibiendo un carácter.
- ✓ If(); para realizar una o varias instrucciones según sea la condición.
- ✓ Switch-case-break-default; para ejecutar un conjunto de instrucciones según sea el caso de la respuesta obtenida, en este caso la usamos para realizar los diferentes menús.
- ✓ For(){}, while(); son estructuras repetitivas para llevar a cabo uno o varios tipos de instrucciones, se repite según la condición definida.
- ✓ Sleep(); utilizada para esperar cierto tiempo en la ejecución del programa, definiendo el tiempo a esperar.
- ✓ Srand(time(NULL)); utilizada para generar valores aleatorios, en este caso esta función la utilizamos para generar un equipo contrincante aleatoriamente.

Funciones del programa:

- ✓ **Void** inicio_simulacion (); Función principal donde se ejecuta la mayor parte del programa, en ella se declara los punteros, y las variables más importantes, tales como el tiempo (min y seg).
- ✓ **Void** inicializar (); Esta función inicializa la función aleatoria **srand**, el tope para el manejo del estacionamiento, los punteros de las listas, los punteros de la estructura cabecera y cada variable que ella contiene**.**
- ✓ **Void** solicitar_time (&min,&seg); Función que solicita el tiempo para la simulación, primero los minutos y luego los segundos. Validando que el minuto y el segundo no sea negativo ni mayor a 60.
- ✓ **Void** mostrar_marco (); Imprime en pantalla un pequeño marco para mostrar el tiempo de la simulación.
- ✓ **Void** mostrar_tiempo (int *min, int *seg); Muestra el tiempo de la simulación en pantalla.
- ✓ **Void** animación (); Función que muestra en pantalla la animación principal.
- ✓ **Bool** aleatorio (); Función que genera un número aleatorio entre el 0 y el 1, importante para la toma de decisiones de forma aleatoria en todo el programa.
- ✓ **Void** nuevo_cliente (p_cabecera, ultm_cliente, &cont); Función que verifica si llego o no un cliente de acuerdo al resultado aleatorio retornado de la función **aleatorio()**. Sí el retorno de la función es

verdadero se procede a enviar a ese cliente a la cola para su atención.

- ✓ **Void** insertar_cliente (CABECERA *p_cabecera, CLIENTES* &ultm_cliente); Función que agrega a la cola el cliente y genera todos sus datos de forma aleatoria, definiendo de una vez el tiempo para el tipo de carro que tiene.
- ✓ **Void** distribución (CABECERA *p_cabecera, SERV1*&ultm_serv1, SERV2*&ultm_serv2, SERV3*&ultm_serv3, GARAJE *p_autos); Función que distribuye a los clientes de forma ordenada por el tipo de servicio solicitado, verificando que el cliente no está en otra cola y que no requiere de ningún servicio.
- ✓ **Void** cola_serv1 (CABECERA *p_cabecera, CLIENTES* &actual, SERV1* &ultm_serv1); Función que agrega al cliente a la cola para ser atendido en el servicio de lavado y aspirado.
- ✓ **Void** cola_serv2 (CABECERA *p_cabecera, CLIENTES* &actual, SERV2* &ultm_serv2); Función que agrega al cliente a la cola para ser atendido en el servicio de cambio de aceite.
- ✓ **Void** cola_serv3 (CABECERA *p_cabecera, CLIENTES* &actual, SERV3* &ultm_serv3); Función que agrega al cliente a la cola para ser atendido en el servicio de lavado de motor o chasis.
- ✓ **Void** atencion_serv1 (CABECERA *p_cabecera); Función que atiende a los clientes del servicio de" lavado de motor". Si ya termino el tiempo de atención en el servicio según el tipo de carro, se toman los datos para las estadísticas y se elimina de la cola, pasando al próximo cliente a atender posteriormente.
- ✓ **Void** atencion_serv2 (CABECERA *p_cabecera); Función que atiende a los clientes del servicio de "cambio de aceite". Si ya termino el tiempo de atención en el servicio según el tipo de carro, se toman los datos para las estadísticas y se elimina de la cola, pasando al próximo cliente a atender posteriormente.
- ✓ **Void** atencion_serv3 (CABECERA *p_cabecera); Función que atiende a los clientes del servicio de" lavado de motor o chasis". Si ya termino el tiempo de atención en el servicio según el tipo de carro, se toman los datos para las estadísticas y se elimina de la cola, pasando al próximo cliente a atender posteriormente.
- ✓ **Void** estadísticas (); Muestra las estadísticas generales de la simulación.
- ✓ **Void** limpiar (CABECERA *p_cabecera); Aseguramos de limpiar las listas que aún permanecen en el programa.
- ✓ **Void** mostrar_date (); Imprime en pantalla la fecha y hora final de la simulación.
- ✓ **Void** creditos(); muestra información acerca del programa.

Sin embargo, cabe aclarar que el programa en sí no está terminado ya que faltaba terminar la función del estacionamiento, y mostrar el comportamiento de los carros en la cola para los servicios. No obstante, con lo que se logró terminar se tuvo un buen resultado acerca de todo el manejo y distribución de los clientes en los servicios, ya que en las estadísticas los valores mostrados son exactos y concuerdan entre la cantidad de los clientes atendidos y la cantidad de atendidos por servicio y por tipo de pago.

6. Pruebas

Se realizaron diversas pruebas a lo largo de la elaboración del programa, estas pruebas corresponden a la visualización del tiempo, las estadísticas, los créditos, los carros y su movimiento hacia su servicio, y como se puede visualizar el establecimiento en general, obteniendo así respuestas satisfactorias luego de repetitivas y numerosas pruebas para la continuidad de la simulación. No obstante, donde hubo mayor énfasis fue en las pruebas del funcionamiento de las colas por servicio, el tiempo, la distribución de los clientes y las estadísticas. Ya que esta última nos indica si se están trabajando correctamente las funciones de distribución, colas de los servicios y atención de cada uno. Si los valores son correctos nos indica que el manejo de las listas es la correcta.

- **Consideraciones Especiales**:

1) Para el requerimiento de los 4 puestos simultáneos del servicio "Lavado y Aspirado" se considera que se puede trabajar en una sola lista y no en 4 listas. Por la siguiente razón:

En vez de atender a un cliente se atienden de a 4 clientes para este servicio, ya que equivale a trabajar con un cliente por puesto. De esta manera simplificamos el manejo de las listas.

2) Con respecto al tiempo de cada carro en un servicio, se tiene la opinión de fijar un tiempo predeterminado para cada tipo de carro y dentro de la función de atención de cada servicio se colocaría un condicional en el cual se pregunte si el tiempo predeterminado para ese tipo de vehiculo es nulo y aleatoriamente generar un valor entre 1 y 0, y trabajar el retorno de esté como una función booleana (verdadero=1, falso=0), de esta manera preguntaríamos en pocas palabras "Sí el tiempo es nulo Y el carro está listo".

Es importante mencionar que esta segunda consideración se hizo en el código pero debido al problema con el ejecutable mencionado y explicado más adelante, se tomó la decisión de quitarla para que los clientes lograrán fueran atendidos rápidamente y lograr terminar la simulación en un tiempo muy corto.

7. Conclusión

A lo largo de las diversas pruebas durante el desarrollo de dicha simulación, se observo unos errores en la programación del mismo. Uno de los más importantes es que existe algún error no determinado que hace que el ejecutable.exe deje de funcionar después de cierto tiempo de ejecución, el código compila exitosamente y no se detecta errores en el mismo, pero cuando se quiere ejecutar el programa directamente desde el ejecutable sucede ese caso, sin embargo es muy raro que cuando se compile y luego se ejecute desde el compilador el ejecutable.exe deje de funcionar, la mayoría de esas veces se logra terminar la simulación correctamente cuando se ejecuta de esa manera (compilando y ejecutando). No obstante, se estuvo revisando el código una y otra vez, pero no pude encontrar el error. Es preciso decir que seguramente no puede ser un puntero, ya que el manejo de listas es correcto por los valores generados en las estadísticas.

Para finalizar, se especificarán los requisitos faltantes de dicho proyecto:

- Falto mostrar el comportamiento visual de las colas por servicio, ya que los carros de los clientes generados aleatoriamente se muestran y se observan sus movimientos hacia un servicio y en tal caso si el servicio está ocupado empieza a hacer la cola, cosa que no se pudo mostrar. En este caso, "el mantener carros en cola" no se pudo imprimir pero se logró hacer

que el carro que iba a un servicio logrará parar en una posición relativa en la que no coincidiera con el anterior carro de la cola.

- Falto la función del estacionamiento, ya que su comportamiento era una pila, se sabía que había que apilar y desapilar.
- Por último y más importante falto arreglar la impresión del tiempo de función que coincidiera, cuando un carro se mueve hacia un servicio.