Informe – Tarea 7: Montículo binario

**Caso real y justificación**

Yo elegí como caso real un restaurante, porque ahí los pedidos llegan todo el tiempo y hay que decidir en qué orden atenderlos. Cada pedido tiene un número que representa la prioridad, la urgencia trabaja con un min-heap (donde los números más chicos son más urgentes).

El heap me sirve porque organiza los pedidos automáticamente y siempre me deja ver cuál es el primero que hay que atender (la raíz). Además, puedo extraer ese pedido y el heap se reacomoda solo, manteniendo el orden sin que yo tenga que ordenar todo de nuevo.

El código refleja cómo se manejaría la atención de pedidos en un restaurante, según la prioridad que se quiera dar.

**Uso del programa**

* Primero se le pide que ingrese los nodos (uno a la vez)
* Si quiere agregar más, se le pregunta con un s/n: con *s* vuelve a ingresar otra tanda, y con *n* pasa directo a la siguiente etapa.
* Recién después de eso el usuario llega al menú, que ahora es más simple y solo tiene estas opciones:

1. **Consultar pedido más urgente:** Busca en el heap el valor más pequeño del vector y lo muestra sin extraerlo.
2. **Consultar pedido menos urgente:** Busca en el heap el valor más grande del vector y lo muestra sin extraerlo.
3. **Atender pedido:** saca la raíz y actualiza el heap con el nuevo valor en la cima (extrae el más urgente).
4. **Mostrar todos los pedidos:** imprime el vector con todos los pedidos actualizados.
5. **Dibujar heap:** visualización SFML
6. **Salir:** cierra el programa.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### **Prompt 1**

**Lo que pregunté:**

*"¿Cómo hago para que cuando no haya pedidos el programa no se rompa al mostrar el más urgente?"*

**Respuesta del asistente:**

Me sugirió usar runtime\_error("No hay pedidos.") y capturar la excepción con catch en el menú.

**Qué tomé:**

La idea de lanzar una excepción y manejarla en el menú.

**Qué descarté:**

La propuesta de devolver -1 como valor especial, porque no quería que el usuario confunda un pedido válido con un error.

### **Prompt 2**

**Lo que pregunté:**

*"Si dos pedidos tienen la misma prioridad, cómo los maneja el heap?"*

**Respuesta del asistente:**

Me explicó que el heap simplemente los guarda como están: ambos quedan en el vector y se atenderán en cualquier orden válido según la estructura.

**Qué tomé:**

La aclaración de que los duplicados sí están permitidos y el heap no se rompe.

**Qué descarté:**

La sugerencia de implementar un segundo criterio de orden (como “hora de llegada”), porque no era necesario para lo que quería hacer

### **Prompt 3**

**Lo que pregunté:**  
 *"¿Cómo puedo probar que el heap funciona bien con muchos pedidos, aunque mi código solo los carga uno por uno?"*

**Respuesta del asistente:**  
 Me recomendó dos opciones:

1. Seguir usando insertarPedidos() pero cargar manualmente varios valores de prioridad.
2. Crear una función adicional de prueba que genere pedidos automáticos en un bucle, solo para testear el heap.

**Qué tomé:**  
 La idea de que para la entrega basta con ingresar varios pedidos manualmente y probar qué pasa si hay muchos.

**Qué descarté:**  
 La propuesta de programar la generación automática de pedidos, porque mi sistema ya cumple con pedirlos uno a uno y no necesitaba código extra.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Casos de prueba:**

Dataset: 10, 20, 30, 15, 50, 12, 40, 7, 18, 45, 1

El usuario inicia el programa y se muestra el mensaje de bienvenida al sistema de pedidos del restaurante. Luego comienza a ingresar las prioridades de los pedidos, donde el número menor ingresado significa mayor urgencia. Los valores que ingresa son:

10, 20, 30, 15, 50, 12, 40, 7, 18, 45, 1

Después de cada ingreso, el programa pregunta si desea agregar otro pedido (s/n). El usuario responde s hasta el último valor, donde responde n. Una vez completado el ingreso, el programa confirma que los pedidos fueron agregados.

Imagen de la pantalla de un computador

El contenido generado por IA puede ser incorrecto.

Una vez tenemos los datos necesarios, el programa nos lleva a un menú donde el usuario puede realizar diferentes acciones sobre los pedidos almacenados en el heap.

Texto

El contenido generado por IA puede ser incorrecto.

opción 1

Texto

El contenido generado por IA puede ser incorrecto.

opción 2



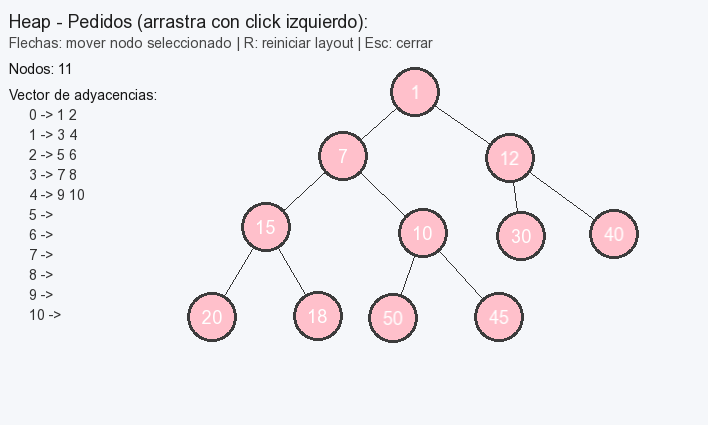
opción 3



opción 4

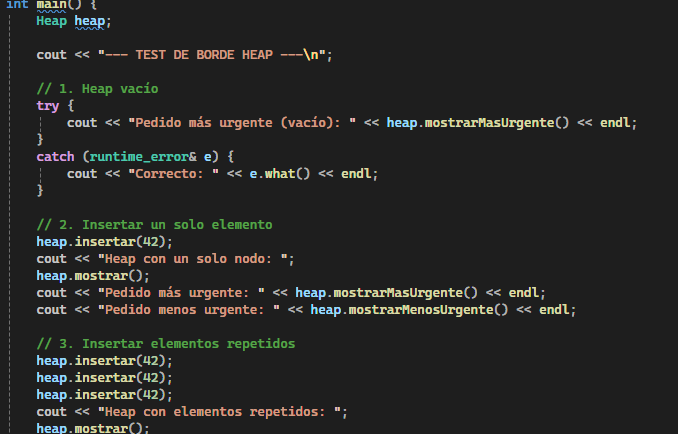


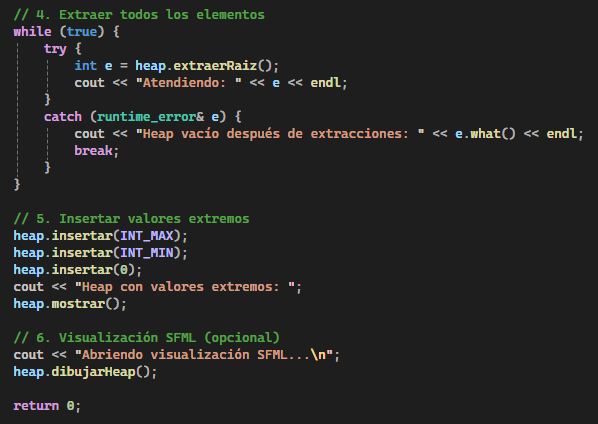
opción 5



**Test de borde sugerido por IA**

Le pedí a la IA que me genere un test de borde y pensó lo siguiente:





✅ Qué prueba este test de borde:

1. Intentar mostrar el más urgente en heap vacío → debería lanzar excepción.
2. Heap con un solo nodo.
3. Insertar valores repetidos y verificar orden.
4. Extraer todos los elementos hasta vacío.
5. Insertar valores extremos (INT\_MAX, INT\_MIN, 0) y mostrar heap.
6. Abrir la visualización SFML para verificar que todo se dibuje correctamente incluso con bordes.

**Resultados**

