Homework 3

Punto 1 - Punto 4

Casó con First Fit

```cpp
int main() {
    int particiones = 5;
    int tamanoMemoria = 68; //Tamano de la memoria
    GestorMemoria gestor(tamanoMemoria);
    int listParticionesInicial[] = {10, 25, 18, 15};
    int listParticiones[] = {10, 25, 18, 15};

    if (gestor.asignarMemoriaFirstFit(1, 20, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 1 de tamaño 20" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaFirstFit(2, 3, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 2 de tamaño 3" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaFirstFit(3, 15, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 3 de tamaño 15" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaFirstFit(4, 10, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 4 de tamaño 10" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaFirstFit(5, 30, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 5 de tamaño 19" << std::endl;
    }
    else {std::cout << "No se pudo asignar" << std::endl;}
    gestor.imprimirEstadoMemoria();

    gestor.liberarMemoria(4, listParticiones, particiones, listParticionesInicial);
    std::cout << "Proceso 4 liberado" << std::endl;

    if (gestor.asignarMemoriaFirstFit(6, 9, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 6 de tamaño 9" << std::endl;
    }

    gestor.imprimirEstadoMemoria();
    return 0;
```

```
Asignado proceso 1 de tamaño 20
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 2 de tamaño 3
Estado de la Memoria: 2 2 2 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 3 de tamaño 15
Estado de la Memoria: 2 2 2 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 -1 -1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 4 de tamaño 10
Estado de la Memoria: 2 2 2 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 -1 -1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 -1 -1 -1 4 4 4 4 4 4 4 4 4
4 -1 -1 -1 -1 -1
No se pudo asignar
Estado de la Memoria: 2 2 2 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 -1 -1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 -1 -1 -1 4 4 4 4 4 4 4 4 4
4 -1 -1 -1 -1 -1
Proceso 4 liberado
Asignado proceso 6 de tamaño 9
Estado de la Memoria: 2 2 2 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 -1 -1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 -1 -1 -1 6 6 6 6 6 6 6 6 6
-1 -1 -1 -1 -1 -1
Program ended with exit code: 0
```

Casó con Best Fit

```
203 int main() {
204     int particiones = 5;
205     int tamanoMemoria = 68; //Tamano de la memoria
206     GestorMemoria gestor(tamanoMemoria);
207     int listParticionesInicial[] = {10, 25, 18, 15};
208     int listParticiones[] = {10, 25, 18, 15};
209 
210     if (gestor.asignarMemoriaBestFit(1, 20, listParticiones, particiones, listParticionesInicial)){
211         std::cout << "Asignado proceso 1 de tamaño 20" << std::endl;
212     }
213     gestor.imprimirEstadoMemoria();
214 
215     if (gestor.asignarMemoriaBestFit(2, 3, listParticiones, particiones, listParticionesInicial)){
216         std::cout << "Asignado proceso 2 de tamaño 3" << std::endl;
217     }
218     gestor.imprimirEstadoMemoria();
219 
220     if (gestor.asignarMemoriaBestFit(3, 15, listParticiones, particiones, listParticionesInicial)){
221         std::cout << "Asignado proceso 3 de tamaño 15" << std::endl;
222     }
223     gestor.imprimirEstadoMemoria();
224 
225     if (gestor.asignarMemoriaBestFit(4, 10, listParticiones, particiones, listParticionesInicial)){
226         std::cout << "Asignado proceso 4 de tamaño 10" << std::endl;
227     }
228     gestor.imprimirEstadoMemoria();
229 
230     if (gestor.asignarMemoriaBestFit(5, 19, listParticiones, particiones, listParticionesInicial)){
231         std::cout << "Asignado proceso 5 de tamaño 19" << std::endl;
232     }
233     else {std::cout << "No se pudo asignar" << std::endl;}
234     gestor.imprimirEstadoMemoria();
235 
236     gestor.liberarMemoria(4, listParticiones, particiones, listParticionesInicial);
237     std::cout << "Proceso 4 liberado" << std::endl;
238 
239     if (gestor.asignarMemoriaBestFit(6, 9, listParticiones, particiones, listParticionesInicial)){
240         std::cout << "Asignado proceso 6 de tamaño 9" << std::endl;
241     }
242 
243     gestor.imprimirEstadoMemoria();
244     return 0;
```

```
Asignado proceso 1 de tamaño 20
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 2 de tamaño 3
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 2 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 3 de tamaño 15
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 2 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3
Asignado proceso 4 de tamaño 10
Estado de la Memoria: 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3
No se pudo asignar
Estado de la Memoria: 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3

Proceso 4 liberado
Asignado proceso 6 de tamaño 9
Estado de la Memoria: 6 6 6 6 6 6 6 6 6 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3
```

Caso con Worst Fit

```cpp
int main() {
    int particiones = 5;
    int tamanoMemoria = 68; //Tamano de la memoria
    GestorMemoria gestor(tamanoMemoria);
    int listParticionesInicial[] = {10, 25, 18, 15};
    int listParticiones[] = {10, 25, 18, 15};

    if (gestor.asignarMemoriaWorstFit(1, 20, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 1 de tamaño 20" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaWorstFit(2, 3, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 2 de tamaño 3" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaWorstFit(3, 15, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 3 de tamaño 15" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaWorstFit(4, 10, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 4 de tamaño 10" << std::endl;
    }
    gestor.imprimirEstadoMemoria();

    if (gestor.asignarMemoriaWorstFit(5, 19, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 5 de tamaño 19" << std::endl;
    }
    else {std::cout << "No se pudo asignar" << std::endl;}
    gestor.imprimirEstadoMemoria();

    gestor.liberarMemoria(4, listParticiones, particiones, listParticionesInicial);
    std::cout << "Proceso 4 liberado" << std::endl;

    if (gestor.asignarMemoriaWorstFit(6, 9, listParticiones, particiones, listParticionesInicial)){
        std::cout << "Asignado proceso 6 de tamaño 9" << std::endl;
    }

    gestor.imprimirEstadoMemoria();
    return 0;
```

```
Asignado proceso 1 de tamaño 20
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 2 de tamaño 3
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 2 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 3 de tamaño 15
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
Asignado proceso 4 de tamaño 10
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4
4 -1 -1 -1 -1 -1
No se pudo asignar
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4
4 -1 -1 -1 -1 -1
Proceso 4 liberado
Asignado proceso 6 de tamaño 9
Estado de la Memoria: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 -1 -1 -1 -1 -1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6
-1 -1 -1 -1 -1 -1
Program ended with exit code: 0
```

Informe:

Estructura del código:

1. El código comienza con la inclusión de las bibliotecas necesarias para la entrada/salida, manejo de vectores, algoritmos y archivos.
2. Luego, se define una clase llamada `Reader`, que contiene una función `lectura(int posicion)` para leer un valor numérico de un archivo llamado "archivo.txt" en una posición específica. Esta clase se utiliza para obtener el tamaño de la memoria en el programa principal además de otros parámetros necesarios para la ejecución del programa.
3. A continuación, se define otra clase llamada `GestorMemoria`, que se utiliza para gestionar la asignación y liberación de memoria. La clase incluye tres métodos principales: `asignarMemoriaFirstFit`, `asignarMemoriaBestFit` y `asignarMemoriaWorstFit` para asignar memoria utilizando diferentes estrategias (First Fit, Best Fit, Worst Fit), `liberarMemoria` para liberar memoria y `imprimirEstadoMemoria` para imprimir el estado actual de la memoria.
4. En la función `main`, se instancia la clase `Reader` para obtener el tamaño de la memoria y se inicializa la clase `GestorMemoria` con ese tamaño y una lista de particiones iniciales.

5. Luego, se realizan varias asignaciones de memoria y liberaciones para probar las estrategias de asignación. Se imprime el estado de la memoria después de cada operación.

Funcionamiento:

- El código utiliza diferentes estrategias (First Fit, Best Fit, Worst Fit) para asignar memoria a procesos. Las estrategias intentan encontrar el segmento de memoria más adecuado para acomodar el proceso dado su tamaño.
- La función `liberarMemoria` se utiliza para liberar memoria previamente asignada por procesos.
- Se utiliza un vector llamado `memoria` para representar el estado de la memoria. Cada posición del vector corresponde a una posición de memoria, y el valor en esa posición indica el proceso al que pertenece.
- El código muestra mensajes de salida para informar sobre la asignación y liberación de memoria, así como el estado actual de la memoria.

Observaciones:

- El código está diseñado para simular la gestión de memoria y las estrategias de asignación, por lo que no se comunica con un sistema operativo real ni realiza asignaciones de memoria física.

En general, este código muestra un ejemplo simple de cómo se podrían implementar diferentes estrategias de asignación de memoria en un programa en C + +.