INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN
ÁREA ACADÉMICA DE INGENIERÍA EN COMPUTADORES
ALGORITMOS Y ESTRUCTURAS DE DATOS II (CE-2103)

TAREA CORTA #1
Arreglos paginados

Realizado por:
Eduardo Moya Bello - 2015096664
María Fernanda Ávila Marín - 2014089607
Gabriel Abarca Aguilar - 2017110442

Profesor:
Ing. Isaac Ramírez, M.SI.

Cartago, 7 de Septiembre de 2018

Tabla de contenidos

Introducción

Antes de hablar de arreglos paginados, se debe comprender el concepto de paginación de memoria. La paginación de memoria es un método de administración de memoria que consiste en la división de los programas en pequeñas partes o páginas. La memoria es dividida en trozos del mismo tamaño que las páginas (marcos de página). (Arpaci-Dusseau, 2014).

En la presente tarea, dicha técnica se aplicará a un problema común de arreglos en el lenguaje de programación C++. Esto permitirá aprender cómo implementar un un arreglo que simule paginación de memoria y mejorar los conocimientos en el lenguaje mencionado anteriormente.

La estructura del trabajo se divide en un índice, una descripción breve del problema a solucionar, una descripción breve de la solución implementada, así como problemas encontrados (retos encontrados al implementar la solución, abordajes para las soluciones y aquello que no haya sido posible implementar).

Descripción del problema

Se tiene un archivo que contiene una gran cantidad de números enteros separados por coma. La tarea consiste en ordenar dicho archivo utilizando quick sort, insertion sort y selection sort. Se supondrá que el computador en el que se ejecuta el programa tiene una memoria máxima de 12 KB por lo que el programa no podrá tener más de 6 páginas de 1 KB (256 enteros) en memoria y deberá ejecutar el algoritmo de ordenamiento bajo esta restricción (Ramírez, I. 2018).

Para los algoritmos, la paginación debe ser transparente (su código no debe ser modificado de tal forma que refleje lo que ocurre detrás de escena). El estudiante deberá implementar una clase `PagedArray` que sobrecargue el operador [ ], para que el algoritmo simplemente pida entradas del arreglo. Además, la clase

`PagedArray` se encargará de buscar la página correspondiente, reemplazando alguna de las que ya estaban cargadas (Ramírez, I. 2018)

Asimismo, se implementará un algoritmo de reemplazo. El programa se ejecutará de la siguiente forma:

```
paged-sort -i <archivo> -a {QS|IS|SS} -o <archivo_resultado>
```

Cuando el programa se ejecute, copiará el contenido de archivo a `archivo_resultado` y aplicará el algoritmopaged-sort -i <archivo> -a {QS|IS|SS} -o <archivo_resultado>paged-sort -i <archivo> -a {QS|IS|SS} -o <archivo_resultado> de ordenamiento (QS para Quicksort, IS para insertion sort o SS para selection sort) sobre el archivo de resultado. No puede generar ningún archivo intermedio (Ramírez, I. 2018).

La solución

La tarea consta de 4 clases principales: `Memory`, `PagedArray`, `Sorting` y `main`. La clase `main` instancia a `PagedArray` quien se encarga de la administración de memoria. `PagedArray` puede considerarse como un análogo a la Unidad de Manejo de Memoria o MMU.

`Memory` por su parte, se encarga de leer archivo y de agregar los datos a un array interno. `Memory`, al igual que `PagedArray`, puede verse como un análogo, en este caso a la memoria virtual. Por otra parte, `Sorting` se encarga de la implementación de los métodos de ordenamiento.

En el `main`, el usuario tiene la posibilidad de seleccionar cuál método de ordenamiento utilizar. Después de seleccionar el método y haber instanciado el `PagedArray`, se realiza una validación para verificar que el dato ingresado por el usuario es correcto y seguidamente se llama al método estático de `Sorting` según

el algoritmo de ordenamiento deseado. Después de haber ordenado la página, el `main` imprime dicha página.

Problemas encontrados

El primer problema consistió en el manejo de conceptos. Los punteros hacen referencias a direcciones en memoria, y cuando éstas se desean manipular se debe utilizar el operador asterisco.

Durante un tiempo, se olvidó de esto, por lo que se trabajaba con direcciones en memoria y no con el dato en memoria. Este es un error común para quienes están iniciando a trabajar en el lenguaje C++, pero al mismo tiempo es un aprendizaje.

Otro problema en la implementación de la solución fue el uso de dos sistemas operativos diferentes. El proyecto se inicó en Microsoft Visual Studio utilizando el sistema operativo Windows 10 y posteriormente se continuó el proyecto utilizando CLion en Linux (Ubuntu).

Esto generó una gran cantidad de problemas, no solo por el manejo de proyectos sino por el funcionamiento de los compiladores, ya que codifican de manera diferente. Esto es un hallazgo interesante ya que demuestra las diferencias entre manejo de memoria.

El método QuickSort ordena cada página, pero no se encarga de ordenar todos los datos. Por motivos de tiempo, no fue posible encontrar una solución a este problema.

Conclusiones

1. La programación orientada a objetos simplifica mucho la solución de problemas. En este caso el enfoque se basó en simular el funcionamiento de una computadora (con analogías).
2. El manejo de conceptos y los errores comunes son fuentes de error que requieren especial atención, ya que pueden consumir mucho tiempo.
3. Es preferible que el manejo de proyectos se realice en los mismos sistemas operativos y/o programas, ya que no todos funcionan de la misma manera, lo que puede generar errores.

Bibliografía

Arpaci-Dusseau, Remzi H.; Arpaci-Dusseau, Andrea C. (2014), Operating Systems: Three Easy Pieces. Recuperado de: http://pages.cs.wisc.edu/~remzi/OSTEP/vm-paging.pdf

Ramírez, I. (2018). *Tarea Corta - Arreglos Paginados*. Tecnológico de Costa Rica. Cartago, Costa Rica.