Practica No. 1

EL Algoritmo de Heap

Nombre(s): **César Eduardo Elias del Hoyo ID: 262045**

Ximena Rivera Delgadillo ID: 261261

José Luis Sandoval Pérez ID: 261731

Objetivo:

Con la realización de esta práctica se pretende que: se implemente el algoritmo de Heap en ANSI C para generar todas las permutaciones de N elementos de un conjunto en forma recursiva

Fundamento Teórico:

Matemáticamente, las permutaciones son un caso particular de las variaciones donde *m=n* así, las permutaciones son aquellas formas de agrupar los elementos de un conjunto teniendo en cuenta que:

1. Se toman todos los elementos disponibles.
2. Influye el orden en que se colocan.

Métodos basados en intercambios

Una forma natural de permutar una serie de elementos en una computadora es intercambiar dos de ellos. Los algoritmos de permutación más rápidos funcionan de esta manera: las permutaciones de *N!* de los *N* elementos son producidas por una secuencia de intercambios *N!-1*.

Métodos recursivos

Se va a analizar una clase de métodos de generación de permutaciones que son muy simples cuando se expresan como programas recursivos.

Para generar todas las permutaciones de *P[1l,...,P[N]*, se repiten *N* veces el paso: *primero generar todas las permutaciones de P[1],...,P [N-1], luego intercambiar P[N] con uno de los elementos P[1],...,P[N-1]*.

Como esto se repite, se pone un nuevo valor en *P[N]* cada vez. Los diferentes métodos difieren en sus enfoques para llenar *P[N]* con los *N* elementos originales.

Forma de trabajo:

Colaborativa en equipos de 3 personas

Material:

1. Computadora
2. Compilador ANSI C

Procedimiento:

Se va a crear un programa que ejecute el desarrollo de todas las permutaciones de un conjunto de N elementos.

Para ello se va a utilizar el modelo del algoritmo de Heap con un enfoque recursivo, en este algoritmo se da un intercambio de un par de elementos mientras que los restantes N-2 elementos no se modifican.

El siguiente pseudocódigo genera todas las permutaciones en forma recursiva de un arreglo de datos de longitud N:

```
procedure permutaciones(Num_ele: entero,  Array_perm; arreglo )
        if Num_ele =1
        salida (Array_perm)
        else
                for contador:= 0; contador<Num_ele – 1; contador+= 1 do
                        permutaciones(Num_ele -1, Array_perm)
                        if Num_ele is par then
                                swap (Array_perm[contador], Array_perm[Num_ele -1])
                        else
                                swap (Array_perm[0], Array_perm[Num_ele -1])
                        end if
                end for
                permutaciones(Num_ele -1, Array_perm)
        end if
```

Para la creación del programa deberán realizarse los siguientes pasos:

1. En las primeras líneas elaborar comentarios con la siguiente información:
    a. Nombre de la institución
    b. Nombre de la carrera
    c. Nombre de la materia
    d. Nombre(s) de quien(es) realiza(n) la práctica
    e. Nombre del profesor
    f. Una descripción breve de lo que realiza el programa
2. Incluir las librerías necesarias.
3. Declarar el prototipo de la función recursiva (permutaciones).
4. Implementar la función principal en la cual se debe desplegar la solicitud del número de elementos del conjunto a permutar (un máximo de 10 elementos).
5. Luego se debe solicitar el tipo de elementos a permutar (pueden ser números enteros o caracteres simples) y luego solicitar los elementos.
6. Declarar la función recursiva e implementarla.

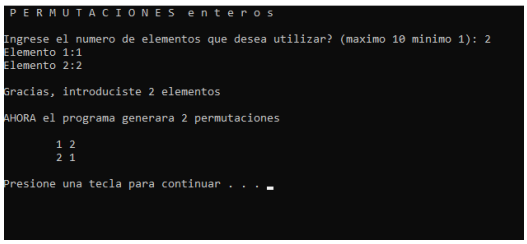
.

7. A continuación, se deben desplegar cada una de las permutaciones generadas por los grupos en cada iteración externa.
8. Cada vez que se realice una permutación completa se debe regresar al menú de inicio o una opción para terminar.
9. Al salir se debe detener el programa y luego regresar el control al sistema inicial.

Resultados:

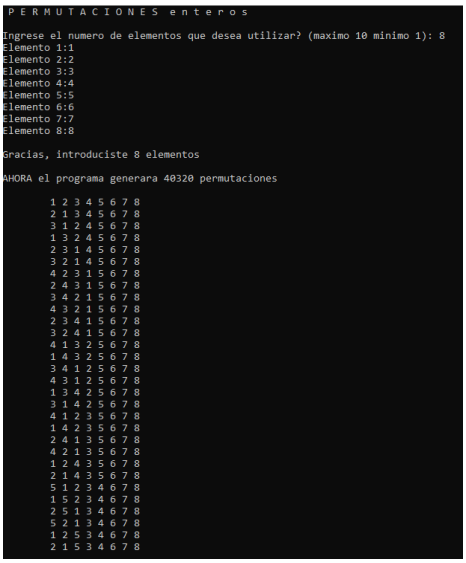
Realizar cuatro corridas de prueba para N=2,4,8,10 y pegar las imágenes de las pantallas de texto generadas en los siguientes recuadros.

Prueba con N = 2

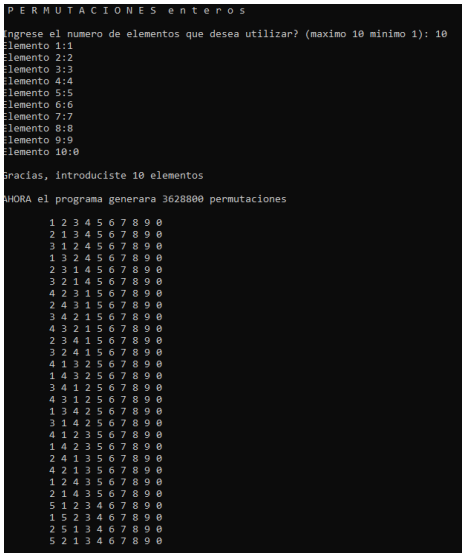


Prueba con N = 4

Prueba con N = 8



Prueba con N = 10



Nota: **Prueba con N=8 y N=10 no muestran todas las permutaciones en la imagen**

Conclusiones:

Durante esta practica comprendimos lo que son las permutaciones y como estas son representadas por el factorial de un numero. A su vez, comprendimos el algoritmo de Heap que nos ayuda a ordenar y afirmar el número de permutaciones posibles dentro de una lista de valores enteros o caracteres simples. Este algoritmo crea la lista de permutaciones posibles intercambiando los valores, siguiendo un patrón.

Para concluir, las permutaciones son parte importante en la programación que permiten analziar las diferentes maneras en que se puede ordenar una lista de N números y el algoritmo de Heap nos ayuda e tener un control de dichas permutaciones con un proceso corto y más sencillo

Una vez terminado el programa debe subirse a la plataforma de **aulavirtual** junto con este reporte.