UNIVERSIDAD POLITÉCNICA DE PÉNJAMO(UPP)

Nombre del alumno: Brandon Iván Márquez Morales.

Grado y Grupo: 1° “A”

Carrera: Ingeniería en Software.

Materia: Algoritmos.

Tema: Reporte de Métodos de ordenamiento

Fecha de inicio: 25/11/2020.

Fecha de término: 26/11/2020.

Docente: Miguel Ángel Saldaña Cabeza.

Contenido

Introducción. 3
Marco teórico. 4
¿Qué es un arreglo? 4
¿Qué es un arreglo vectorial? 4
¿Cómo es el uso de un arreglo vectorial en C++? 4
Declaración de un Array o Vector en C++ 5
Línea 1 5
Línea 2 6
Línea 3 6
Línea 4 6
Línea 5 6
¿Cómo inicializar un Array o Vector en C++? 6
Forma 1 de declarar un Array o Vector en C++ 7
Forma 2 de declarar un Array o Vector en C++ 7
Descripción del algoritmo 8
Ventajas, Desventajas de Quicksort 9
Ventajas: 9
Desventajas: 9
Desarrollo. 10
Diagrama de flujo (Quicksort): 10
Código explicado en el lenguaje C++ 11
Conclusiones. 14
Bibliografía 15

Introducción.

En el presente documento se presentará una de las formas de ordenar los números de un array o arreglo en este caso se trata del método de ordenamiento “Quicksort”.

Los métodos de ordenamiento son necesarios para que luego de ordenar, se puedan buscar datos de una manera mucho más rápida y eficiente aplicando distintas técnicas.

Para poder ordenar una cantidad determinada de números almacenadas en un vector o matriz, existen distintos métodos (algoritmos) con distintas características y complejidad.

Existe desde el método más simple, como el Bubbleshort (o Método Burbuja), que son Simples iteraciones, hasta el Quickshort (Método Rápido), que al estar optimizado usando recursión, su tiempo de ejecución es menor y es más efectivo.

Para el ejemplo que se realizó principalmente el vector ya puede tener los valores establecidos o bien, también principalmente se le puede pedir al usuario la dimensión del vector y posteriormente pedir los valores para almacenarlos en el vector.

Marco teórico.

¿Qué es un arreglo?

Un arreglo (vector, array, matriz) es un conjunto de datos o una estructura de datos homogéneos que se encuentran ubicados en forma consecutiva en la memoria RAM (sirve para almacenar datos en forma temporal). (EcuRed, 2018+)

¿Qué es un arreglo vectorial?

Los arreglos se dividen en 2 grupos, los vectores las matrices. Los vectores son arreglos que contienen una sola dimensión y las matrices 2 o más dimensiones.

Los Arreglos se utilizan para almacenar un conjunto de variables, que sean del mismo tipo de dato, y todas estas bajo un mismo nombre.

Para crear un arreglo se debe en primer lugar declararlo como cualquier otra variable, la única diferencia es que debemos indicar la cantidad de elementos que contendrá el arreglo, colocando el número de índice entre paréntesis. (Wordpress, 2017)

¿Cómo es el uso de un arreglo vectorial en C++?

¿Cómo declarar un Array o Vector en C++?

Para declarar un vector en C++, se deben seguir las mismas normas básicas que se siguen para declarar una variable cualquiera, con un pequeño cambio en la sintaxis. Para declarar un vector, arreglo o como lo quieras llamar, necesitaremos saber el tipo de los datos que irán al interior de este, es decir, serán número enteros, o número decimales o cadenas de texto, etc. necesitamos también, como siempre, un

nombre para el vector y un tamaño máximo. La sintaxis para declarar un vector en C++ es la siguiente:

```
tipo_de_dato nombre_del_vector[tamanio];
```

Tenemos entonces, tal como mencioné antes, que para declarar un vector en C++, debemos definirle un tipo de los datos, sea entero, float, string, etc., debemos darle un nombre y al interior de los corchetes "[]" debemos poner el tamaño máximo que tendrá el vector, es decir la cantidad máxima de datos que podrá contener (recuerda que en C++ esto es necesario hacerlo). Veamos un ejemplo en el cual pondré la declaración de varios vectores de diferentes tipos y tamaños en C++. (ProgramarYa, 2019)

Declaración de un Array o Vector en C++

```
int my_vector1[10];
float my_vector2[25];
string my_vector3[500];
bool my_vector4[1000];
char my_vector5[2];
```

Veamos rápidamente que representa cada línea del código anterior.

Línea 1

Esta línea contiene la declaración de un vector llamado my_vector1, el cual contendrá un máximo de 10 elementos de tipo entero.

Línea 2

Esta línea contiene la declaración de un vector llamado my_vector2, el cual contendrá un máximo de 25 elementos de tipo float.

Línea 3

Esta línea contiene la declaración de un vector llamado my_vector3, el cual contendrá un máximo de 500 elementos de tipo string.

Línea 4

Esta línea contiene la declaración de un vector llamado my_vector4, el cual contendrá un máximo de 1000 elementos de tipo booleano.

Línea 5

Esta línea contiene la declaración de un vector llamado my_vector5, el cual contendrá un máximo de 2 elementos de tipo char.

Ya que está claro cómo se declara un vector, vamos a ver cómo inicializarlo, es decir inicializar un vector en C++ o en otras palabras darle valores a un vector. (ProgramarYa, 2019)

¿Cómo inicializar un Array o Vector en C++?

En cuanto tenemos declarado un vector, es posible asignarle valores, evidentemente estos valores deben coincidir con el tipo de dato que le asignamos a dicho vector, no tendría sentido ingresar como valores de un vector cadenas de caracteres si el tipo de dato de dicho vector es numérico.

Se va a mostrar a continuación otra forma distinta de inicializar un vector, todas son válidas, ya es cuestión de nuestras necesidades y conocimientos determinar cuál es útil y en qué momento. (ProgramarYa, 2019)

Forma 1 de declarar un Array o Vector en C++

```
string vector[5] = {"5", "hola", "2.7", "8,9", "adios"};
```

Aquí hemos declarado un vector de tipo string tamaño 5 y lo hemos inicializado con diferentes valores, es necesario notar que cada valor va entre comillas dobles "" puesto que son strings. El valor inicial corresponde a la casilla o índice 0 y tiene el valor de "5", el índice 1 el valor es "hola" y el índice 4 el valor es "adiós", es importante notar que el primer índice de n array o vector no es el UNO sino que es el CERO. (ProgramarYa, 2019)

Forma 2 de declarar un Array o Vector en C++

```
int vector2[] = {1,2,3,4,10,9,80,70,19};
```

Aquí hemos declarado un vector de tipo int y no especificamos su tamaño, si el tamaño no se especifica entre los corchetes, el vector tendrá como tamaño el número de elementos incluidos en la llave, para este caso es 9.

(ProgramarYa, 2019)

Descripción del algoritmo

El algoritmo consta de los siguientes pasos:

1.- Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.

2.- Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.

3.- La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.

4.- Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados. (Ecured2, 2018)

Ventajas, Desventajas de Quicksort

Ventajas:

Requiere de pocos recursos en comparación a otros métodos de ordenamiento.

En la mayoría de los casos, se requiere aproximadamente N log N operaciones.

Ciclo interno es extremadamente corto.

No se requiere de espacio adicional durante ejecución (in-place processing). (wordpress2, 2019)

Desventajas:

Se complica la implementación si la recursión no es posible.

Peor caso, se requiere N2

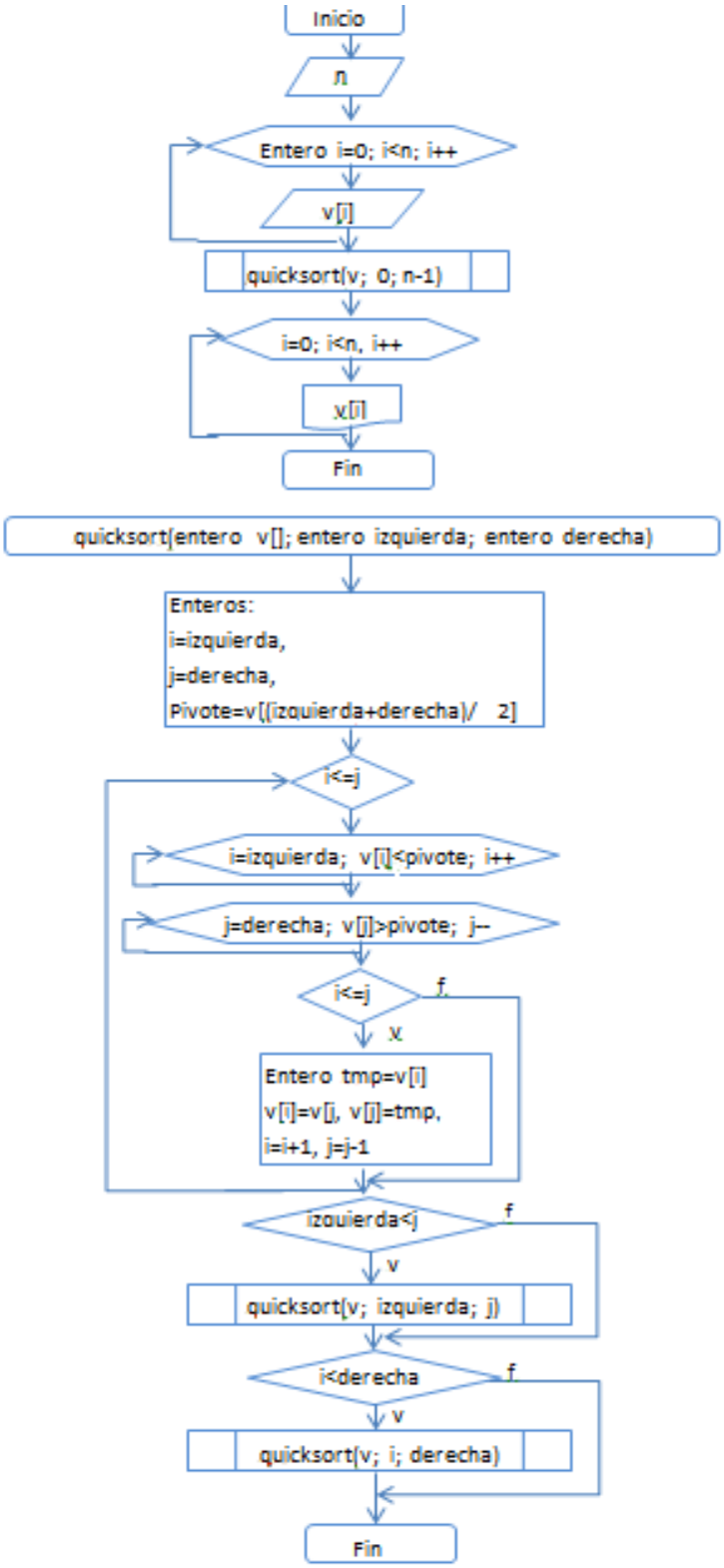
Un simple error en la implementación puede pasar sin detección, lo que provocaría un rendimiento pésimo.

No es útil para aplicaciones de entrada dinámica, donde se requiere reordenar una lista de elementos con nuevos valores.

Se pierde el orden relativo de elementos idénticos. (wordpress2, 2019)

Desarrollo.

Diagrama de flujo (Quicksort):



Código explicado en el lenguaje C++

```
        //EN ESTE BLOQUE SE COMPARA SI LOS ELEMENTOS YA TIENEN UNA POSICIÓN ORDENADA CON LOS ANTERIORES DE SER ASÍ-
        //SE HACE EL INTERCAMBIO DE LOS ELEMENTOS
        if (i <= j){
            tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i++; j--;
        }
    }
    //se comprueba si se cumplió lo anterior
    //de ser asi a nuestra funcion "quicksort" le ponemos COMO PARAMETROS los parametros "arr", "izq", "j", "i", "der
    if(izq < j){
        Quicksort(arr, izq, j);
    }
    //se comprueba si se cumplió lo anterior
    //de ser asi a nuestra funcion "quicksort" le ponemos COMO PARAMETROS los parametros "arr", "izq", "j", "i", "der
    if(i < der){
        Quicksort(arr, i, der);
    }
}

int main()
{
    setlocale(LC_CTYPE, "Spanish");

    //PEDIMOS LA DIMENSIÓN AL USUARIO Y LA LEEMOS
    cout<<"Digita de cuantos elementos quieres que tenga tu vector: "; cin>>NumeroDeElementosdelVector;
```

```
int main()
{
    setlocale(LC_CTYPE, "Spanish");

    //PEDIMOS LA DIMENSIÓN AL USUARIO Y LA LEEMOS
    cout<<"Digita de cuantos elementos quieres que tenga tu vector: "; cin>>NumeroDeElementosdelVector;

    //PEDIMOS LOS ELEMENTOS DEL VECTOR...
    cout<<"Digita los elemento de Vector:"<<endl;

    //ESTE CICLO for SIRVE PARA ALMACENAR LOS VALORES DADOS POR EL USUARIO AL VECTOR
    //INICIAMOS EN CERO HASTA LA LONGITUD DEL VECTOR QUE HALLA DICHO EL USUARIO
    for(int i = 0; i < NumeroDeElementosdelVector; i++){

        cout<<"V["<<i<<"] = ";
        //AQUI ES DONDE SE ALMACENA EL VALOR
        cin>>Vector[i];
    }

    //AQUI SIMPLEMENTE SE MUESTRAN LOS ELEMENTOS INGRESADOS AL VECTOR
    cout<<"Estos son elementos que ingresó..."<<endl;
    for(int i = 0; i < NumeroDeElementosdelVector; i++){
        cout<<Vector[i]<<endl;
    }

    cout<<"Este el es orden ascendentemente..."<<endl;

    //FINALMENTE SE LLAMA A LA FUNCION Quicksort, INVOCAMOS A NUESTRO "Vector" INICIANDO EN 0 HASTA EL NÚMERO DE ELEMENTOS DEL VECTOR
```

```
    //PEDIMOS LOS ELEMENTOS DEL VECTOR...
    cout<<"Digita los elemento de Vector:"<<endl;

    //ESTE CICLO for SIRVE PARA ALMACENAR LOS VALORES DADOS POR EL USUARIO AL VECTOR
    //INICIAMOS EN CERO HASTA LA LONGITUD DEL VECTOR QUE HALLA DICHO EL USUARIO
    for(int i = 0; i < NumeroDeElementosdelVector; i++){

        cout<<"V["<<i<<"] = ";
        //AQUI ES DONDE SE ALMACENA EL VALOR
        cin>>Vector[i];
    }

    //AQUI SIMPLEMENTE SE MUESTRAN LOS ELEMENTOS INGRESADOS AL VECTOR
    cout<<"Estos son elementos que ingresó..."<<endl;
    for(int i = 0; i < NumeroDeElementosdelVector; i++){
        cout<<Vector[i]<<endl;
    }

    cout<<"Este el es orden ascendentemente..."<<endl;

    //FINALMENTE SE LLAMA A LA FUNCION Quicksort, INVOCAMOS A NUESTRO "Vector" INICIANDO EN 0 HASTA EL NÚMERO DE ELEMENTOS DEL VECTOR
    Quicksort(Vector, 0, NumeroDeElementosdelVector);

    //SE IMPRIMEN LOS VALORES YA ORDENADAMENTE
    for(int i = 1; i <= NumeroDeElementosdelVector; i++){
        cout<<Vector[i]<< " ";
     }
     return 0;
}
```

Conclusiones.

A comparación de los otros métodos de ordenamiento claramente este es el método más rápido a la hora de compilarlo, ya que hice una comparación con el método de ordenamiento de la burbuja y hay una diferencia de 5 o más segundos.

Tal vez sea un poco complejo de entender el algoritmo además de desarrollarlo, personalmente si entender como funciona el algoritmo no me resulto tan difícil lo entendí muy bien gracias a un videotutorial en cuál entendí de se trata y como es su desarrollo.

A la hora de desarrollar el código si me costó un poco de trabajo al hacer el intercambio de valores así que tuve que consultar diferentes fuentes de información para desarrollar el código, pero una vez el haber desarrollado el código la verdad no tendría ningún problema para traducirlo a otro lenguaje de programación.

Finalmente considero que este es el método que se debe de utilizar si o si, si lo queremos aplicar a alguna aplicación para el ordenamiento de valores en un array ya que es muy efectivo y rápido de compilar.

Bibliografía

EcuRed. (2018+). *EcuRed*. Obtenido de EcuRed: https://www.ecured.cu/Arreglos_(Inform%C3%A1tica)

Ecured2. (2018). *Ecured2*. Obtenido de Ecured2: https://www.ecured.cu/QuickSort

ProgramarYa. (2019). *ProgramarYa*. Obtenido de ProgramarYa: https://www.programarya.com/Cursos/C++/Estructuras-de-Datos/Arreglos-o-Vectores

Wordpress. (2017). *Wordpress*. Obtenido de Wordpress: https://pragracomputacional.wordpress.com/arreglos-y-vectores/

wordpress2. (2019). *wordpress2*. Obtenido de wordpress2: https://quicksortweb.wordpress.com/2017/10/07/ventajas-desventajas-y-aplicaciones/