Logotipo, nombre de la empresa

Descripción generada automáticamenteMétodo de Quicksort

Nombre: Oscar Alejandro Penilla Skakievich

Fecha: 29/08/22

Grupo:5D1

Registro: 20300701

**Descripción del programa:**

Se va a crear un programa que ordene un conjunto de números enteros usando el método de Selección, los datos se los dará el usuario y este mismo decidirá hasta cuantos ingresar dentro de un rango de 1 a 50. El programa también podrá mostrar o no las iteraciones en cada vez que se repita para que el usuario pueda ver el proceso de su arreglo, este a diferencia del de burbuja anteriormente hecho será que el primer valor lo vamos a tomar como el mas grande y si hay uno que le gane por ende el nuevo valor será el mas grande.

**Requerimientos:**

a) Ordenará un conjunto de números enteros, la cantidad mínima deberá ser 1 y el máximo 50.

b) Deberá ingresar los valores correspondientes.

c) El o los módulos de ordenación deberán ser métodos.

d) El sistema está diseñado en C++.

e) La ordenación será de menor a mayor.

f) Mostrar el mensaje de error correspondiente a cada posibilidad.

g) Evite usar variables globales.

h) Responda lo siguiente y anexe sus respuestas al reporte de QUICKSORT, estas se revisarán como si fuera un caso 5.

a. Explique su algoritmo de solución.

Se crean otros dos arreglos para cada parte del arreglo, una mayor y una menor, y se crea un valor para que sepamos cuando parar que se le llama tope, ya que este nos dirá si seguir reptiendo el proceso o pararlo, y se ordena cada mitad ya sea la mayor o la menor con base al primer valor obtenido

b. ¿Cómo distribuye los elementos mayores y menores del pivote?

Por medio de dos arreglos ya con nombre para el mayor y para el menor

c. ¿Crea nuevos arreglos o todas las acciones las realiza sobre el mismo arreglo?

Ocupa manejar al menos dos arreglos mas y esos los ordena

**PSP:**

|  |  |  |  |
| --- | --- | --- | --- |
| Contenido del PSP – Quicksort | | | |
| Código fuente: | Estimado | Real | Error |
| (min) | (min) |
| Tiempo de Diseño | 40 | 60 | 50% |
| Tiempo de Codificación. | 120 | 180 | 50% |
| Tiempo estimado de fabricación | 150 | 180 | 20% |
| Total, de líneas de código nuevas. | 50 | 69 | 38% |
| Total, de líneas de código reusadas. | 100 | 44 | 56% |
| Total, de líneas de código del programa. | 150 | 113 | 25% |
| Cantidad de errores de compilación. | 60 | 90 | 50% |
| Cantidad de errores de ejecución. | 60 | 50 | 17% |

**Código fuente:**

// Metodo Quicksort.cpp : Este archivo contiene la función "main". La ejecución del programa comienza y termina ahí.

//Oscar Penilla 20300701

#include <iostream>

using namespace std;

class numeros

{

public:

int cantidad = 0;

int arreglo[50];

numeros(int, int\*);

~numeros();

void quicksort(int\*, int);

};

numeros::numeros(int, int\*)

{

//numeros::iteracion, iteracion; //1

numeros::cantidad = cantidad;

numeros::arreglo[50] = arreglo[50];

}

numeros::~numeros() { }

void numeros::quicksort(int\* arreglo, int cantidad) {

int tope, ini, fin, pos;

int may[25], menor[25];//usando la constante se crean arreglos mayor y menor con la cantidad de numeros que pueden tener

tope = 0;

menor[tope] = 0;//menor[0]=0

may[tope] = cantidad - 1;//may[0]=n(elemtent) menos 1

while (tope >= 0)//mientras el tope sea mayor o igual a 0

{//este es el mas importante, egloba a todas las repeticiones

ini = menor[tope];//inicio igual a menor[0]

fin = may[tope];//final igual a may[0]

tope--;//tope disminuye

int izq, der, aux;//izquierda, derecha auxiliar

bool b\_door;//booleano b\_door

izq = ini;//izq = inicial

der = fin;//derecha = fin

pos = ini;// posterior = inicial

//inicial------final

b\_door = true;//nos ayuda a parar

while (b\_door == true)//algo

{//while 1

while ((arreglo[pos] < arreglo[der]) && (pos != der))//arreglo[inicial=izquierda] menor a arreglo[derecha] &&and posterior no sea igual a derecha

der--;//derecha se reducira

if (pos == der)//si posicion es igual a derecha

b\_door = false;//sale si es igual

else

{

aux = arreglo[pos];//auxiliar sera arreglo posicion

arreglo[pos] = arreglo[der];//arreglo posicion igual a arreglo derecha

arreglo[der] = aux; //arreglo derecha igual a auxiliar

pos = der;//psicion igual a derecha pivote

}//termina el if

while ((arreglo[pos] > arreglo[izq]) && (pos != izq))//Entramos en otra repeticion, en la cual se va a repetir cada que

izq++;//izquierda aumenta(Contrario a la derecha)

if (pos == izq)//su posicion es igual a izquierda sale

b\_door = false;//salimos si ya llegamos a pasar todo el arreglo

else

{

aux = arreglo[pos];//cambios de variable

arreglo[pos] = arreglo[izq];

arreglo[izq] = aux;

pos = izq;

}

}

if (ini <= (pos - 1))//si el incio es menor o igual a la posicion actual menos 1

{

tope++;//el tope aumenta para

menor[tope] = ini;//del menor el topa va a ser el nuevo inicio

may[tope] = pos - 1;//y del mayor va a ser la posicion menos uno

}

if (fin >= (pos + 1))//ahora el final si es mayor o igual a la posicion a

{

tope++;//aumentamos nuestro tope

menor[tope] = pos + 1;//y de nuestro arreglo del lado menor le damos el valor de la posicion mas uno para seguir

may[tope] = fin;//y asi obtenemos que el arreglo de los mayores en la posicion tope es igual al final

}

}

for (int j = 0; j < cantidad; j++)//mostramos el resultado

{

cout << arreglo[j] << " ";

}

}

int main()

{

int cant = 0;

char iteracion = 'S';

int arreglo[50];

do

{

cout << "Cual es la cantidad de digitos que quieres ingresar? Min 1 max 50" << endl;

cin >> cant;

if (cant < 1 || cant > 50) {

cout << "Valor fuera de rango\n";

}

} while (cant < 1 || cant > 50);

cout << "Ingresa los datos: " << endl;

for (int i = 0; i < cant; i++)

{

cin >> arreglo[i];

}

numeros ob1(cant, arreglo);

ob1.quicksort(arreglo, cant);

}