AC

Practica 2

Fase 2 individual

Grado en ingeniería informática

Francisco Joaquín Murcia Gómez 48734281H

Grupo 1

Algoritmo de la burbuja

Este algoritmo es de los algoritmos de ordenación mas sencillos que hay; este consiste en recorrer el vector y si el elemento de la posición n-1 es menor al elemento de la posición n, este los intercambia de lugar pasando de la posición n-1 a la n y viceversa, esto se repite hasta que el vector esta ordenado, una manera de hacerlo, es repitiendo todo tantas veces como elementos tenga el vector, nuestro algoritmo esta hecho con esta lógica.

A continuación, una foto de del código:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

int main()
{
    int vector[20] = {2,30,45,2,7,4,8,1,4,5,7,9,5,4,0,6,7,6,1,3};

    __asm
    {
          mov esi, 0//contador posicion inicial del vector; controla la cantidad de veces que ejecuta el algoritmo
          mov edi, 19//contador posicion final del vector, le llamaremos n; controla la posicion de los accesos al vector

       recorrido:
          sub edi, 1//le resta 1 a n
          mov edx, [vector+4*edi]//accede a la posicion n-1 del vector y lo guarda en el registro edx
          add edi, 1//restable n
          mov eax, [vector+4*edi]//accede a la posicion n del vector y lo guarda en el registro eax
          cmp edx, eax//comparamos  los registros correspondientes a las posiciones n y n-1 del vector
          jl intercambio //si "lo que hay en n del vector" > "lo que hay en n-1 del vector" intercambia los valores en la etiqueta "intercambio"
          jmp decrementar // si no, decrementa n en la etiqueta "decrementar"

       incremento:
          inc esi// aumentamos el contador de inicio de vector
          mov edi, 19 //restablecemos el contador de final en 19
          cmp esi, 20//compara 20 con el contador de inicio del vector
          jl recorrido//si esi < 20 salta a la etiqueta "recorrido" y sigue ordenando el vector
          jmp fin//si no, salta a la etiqueta "fin"

       intercambio:
          mov [vector+4*edi], edx// guarda el numero menor (edx) en la posicion que ocupaba eax en el vector
          sub edi, 1//le resta 1 a n
          mov [vector + 4 * edi], eax// guarda el numero mayor (eax) en la posicion que ocupaba edx en el vector
          add edi, 1//restable n
          jmp decrementar // salta a la etiqueta "decrementar"

       decrementar:
          dec edi// decrementa en 1 el contador de final de vlector
          cmp edi, esi// comparamos los contadores
          jg recorrido// si edi>esi sigue recorriendo el vector saltando a la etiqueta "recorrido"
          jmp incremento// si no, salta a "incremento"

       fin://etiqueta para finalizar el programa

    }
    for (int i = 0; i < 20; i++) {
       printf_s(" %d ", vector[i]);
    }
}
```

El registro “esi” controla el numero de veces que recorre el vector.

El registro “edi” controla las posiciones del vector, este empieza en 19, ya que el vector contiene 20 elementos.

Nuestro programa consta de 4 etiquetas, que son las siguientes:

Recorrido: este es el encargado de recorrer el vector y mirar si esta ordenado o no, si lo esta salta a la etiqueta “decrementar”, y si no a la etiqueta “intercambio”.

Intercambio: este es el encargado de intercambiar los valores del vector, una vez realizado este intercambio, llama a la etiqueta “decrementar”

Decrementar: este es el encargado de continuar con la ejecución del programa, se encarga de decrementar el registro que controla las posiciones del vector y comprueba si ha acabado de recorrer el vector, en caso afirmativo salta a la etiqueta “incremento”, si no, a la de “recorrido”.

Incremento: esta etiqueta controla la finalización del programa, para ello incrementa el registro “esi” y resetea el “edi”, luego comprueba si “esi” es menor a 20, en caso afirmativo salta a la etiqueta “recorrido”, en caso contrario quiere decir esta todo el vector ordenado y salta a la etiqueta “fin” que básicamente finaliza el programa