Logotipo, nombre de la empresa

Descripción generada automáticamenteMétodo de Radix

Nombre: Oscar Alejandro Penilla Skakievich

Fecha: 05/09/22

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. Un programa hecho para ordenar grandes cantidades de datos, curiosamente solamente puede o positivos o negativos, además este su ordenamiento es por medio de las unidades, decenas, centenas etc etc. En este caso vamos a ordenar números de 1000 a 9999 tanto positivos como negativos

**Requerimientos:**

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

b)El rango de datos deberá ser entre1000-9999o-9999 y -1000.

c)Deberá ingresar los valores correspondientes.

d)Deberá tener dos opciones de ordenación (Elija como preguntar esta opción).

a. Los valores que se van a ordenar serán positivos.

b. Los valores que se van a ordenar serán negativos.

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

f)El sistema está diseñado en c++.

g)La ordenación será de menor a mayor.

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

**PSP:**

|  |  |  |  |
| --- | --- | --- | --- |
| Contenido del PSP – Radix | | | |
| Código fuente: | Estimado | Real | Error |
| (min) | (min) |
| Tiempo de Diseño | 30 | 40 | 33% |
| Tiempo de Codificación. | 60 | 103 | 72% |
| Tiempo estimado de fabricación | 90 | 143 | 59% |
| Total, de líneas de código nuevas. | 80 | 119 | 49% |
| Total, de líneas de código reusadas. | 50 | 20 | 60% |
| Total, de líneas de código del programa. | 130 | 139 | 7% |
| Cantidad de errores de compilación. | 50 | 120 | 140% |
| Cantidad de errores de ejecución. | 50 | 70 | 40% |

**Código fuente:**

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

//radix

//Oscar Alejandro Penilla 20300701

#include <iostream>

#include<cmath>

#include<list>

using namespace std;

class numeros

{

public:

int cantidad;

int arreglo[50];

numeros(int, int\*);

~numeros();

void radixSort(int\*, int, int, char);

void ver(int\*, int, char);

};

numeros::numeros(int, int\*)

{

numeros::cantidad = 0;

}

numeros::~numeros() { }

void numeros::ver(int\* arradix, int cant, char neg)

{

if(neg=='N'){

// cout << "Entro2"<<endl;

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

arradix[i]=arradix[i]\*-1;

cout << arradix[i] <<endl;

}

cout <<"Arreglo ordenado: "<<endl;

for (int i = cant-1; i >= 0; i--){

cout << " ["<<arradix[i] << "] ";

}

}else{

cout <<"Arreglo ordenado: "<<endl;

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

cout << " ["<<arradix[i] << "] ";

}

}

void numeros::radixSort(int \*arr, int n, int max, char neg) {

if(neg=='N'){

// cout << "Entro1"<<endl;

for(int i=0; i<n; i++){

arr[i]=arr[i]\*-1;

cout << arr[i] <<endl;

}

}

int i, j, m, p = 1, index, temp, count = 0;

list<int> pocket[10]; //vamos a hacer un arreglo para 10

for(i = 0; i< max; i++) {

m = pow(10, i+1);

p = pow(10, i);

for(j = 0; j<n; j++) {

temp = arr[j]%m;

index = temp/p; //encontrar el índice para el pocket arreglo

pocket[index].push\_back(arr[j]);

}

count = 0;

for(j = 0; j<10; j++) {

//eliminar de listas vinculadas y almacenar en el arreglo

while(!pocket[j].empty()) {

arr[count] = \*(pocket[j].begin());

pocket[j].erase(pocket[j].begin());

count++;

}

}

}

}

int main()

{

int cant = 0;

char posinega = '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<<"Ingresa un valor correspondiente"<<endl;

}

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

do

{

cout << "Los valores seran positivos? (S/N)" << endl;

cin >> posinega;

switch (posinega)

{

case 'S':

{

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

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

{

cin >> arreglo[i];

if(arreglo[i] <= 1000 || arreglo[i] > 9999){

i--;

cout << "Ingresa un valor del 1000 a 9999"<<endl;

}

}

numeros ob1(cant, arreglo);

ob1.radixSort(arreglo, cant, 4, posinega);

ob1.ver(arreglo, cant, posinega);

}

break;

case 'N':

{

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

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

{

cin >> arreglo[i];

if(arreglo[i] >= (-1000) || arreglo[i] <= (-9999) ){

i--;

cout << "Ingresa un valor del -1000 a -9999"<<endl;

}

}

numeros ob1(cant, arreglo);

ob1.radixSort(arreglo, cant, 5, posinega);

ob1.ver(arreglo, cant, posinega);

}

break;

}

} while (posinega != 'S' && posinega != 'N');

}