Revista Ciencia & Tecnología

No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

CC BY NC

Ordenamiento Matemático basado en C++ mediante Code::Blocks con aplicación al área contable

Mathematical ordering based on C ++ using Code :: Blocks with application to the accounting area

Eduardo Toscano Guerrero[1]
fe.toscano@uta.edu.ec
https://orcid.org/0000-0002-3951-7774

Fabián Londo Yachambáy[2]
fabianlondo@yahoo.com
https://orcid.org/0000-0002-5753-2855

Recibido: 28/9/2021, Aceptado: 28/12/2021

RESUMEN
La Tecnología hace que los procesos se optimicen cada vez más, logrando disminuir tiempos en cada uno de los procedimientos de trabajo de cualquier índole como por ejemplo en producción, planificación, administración, etc. Dicha optimización se hace posible mediante la utilizando softwares de programación como lenguajes computacionales que son herramientas ahorradoras de tiempo, el momento de diagramar un proceso específico el cual genera un crecimiento increíble de nueva tecnología. Por esta razón se creó el Lenguaje C++, como herramienta de trabajo para optimización de procesos diversos, el cual es un mejoramiento de Lenguaje original "C". El propósito de la investigación es analizar la versatilidad y velocidad del ordenamiento para varios casos específicos en el área contable, como por ejemplo: Una tasa cambiaria, en el ámbito de producción, de bodegaje e inventarios, en pedidos según fechas específicas, en fin. La necesidad de optimizar tiempo en cada uno de los procedimientos dentro de los procesos implica un análisis más profundo de lograr agilitar el trabajo diario, logrando de esta manera ahorrar recursos humanos, materiales y económicos. Se realizan dentro del estudio de investigación el análisis de seis casos de ordenamiento de tal manera que se puedan registrar los tiempos de ejecución.

Palabras clave: Optimización, programación, lenguaje, ordenamiento, velocidad, sistematización

[1] Magister en Matemática Aplicada, Universidad Técnica de Ambato, Ecuador
[2] Magister en Matemática Aplicada, Universidad Técnica de Ambato, Ecuador

Revista Ciencia & Tecnología

No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

Abstract
Technology makes processes more and more optimized, managing to reduce times in each of the work procedures of any kind such as in production, planning, administration, etc. This optimization is made possible by using programming software such as computer languages that are time-saving tools, the moment of diagramming a specific process which generates an incredible growth of new technology. For this reason, the C ++ Language was created as a work tool for optimization of various processes, which is an improvement of the original "C" Language. The purpose of the research is to analyze the versatility and speed of the ordering for various specific cases in the accounting area, such as: An exchange rate, in the field of production, warehousing and inventories, in orders according to specific dates, in short. The need to optimize time in each of the procedures within the processes implies a more in-depth analysis to streamline daily work, thus saving human material and financial resources. The analysis of six ordering cases is carried out within the research study in such a way that the execution times can be recorded.

Keywords: Optimization, programming, language, ordering, speed, systematization.

Introducción

El presente trabajo de investigación tiene como finalidad realizar un análisis teórico-práctico de 6 métodos de ordenamiento con un arreglo de enteros con 100 elementos, de tal manera que se puedan comparar los tiempos de ejecución de cada uno de ellos que están programados en lenguaje C++, que es una herramienta poderosa para elaborar proyectos de cualquier tipo. Por lo tanto, el objetivo general de este trabajo es analizar cada uno de los métodos, su programación, su funcionamiento, su velocidad de resolución y poder comparar la versatilidad de los ordenamientos en los diferentes casos. Además, revisar la literatura para poder crear un marco lógico que sirva como apoyo conceptual y de esta manera poder realizar un análisis mucho más sobrio del tema en mención. Para alcanzar este objetivo se realiza un análisis objetivo de cada uno de los casos de ordenamiento como son: De Selección, Inserción, Intercambio, Burbuja, Shell y Quicksort, los cuales tienen diferentes principios conceptuales con respecto a su análisis conceptual. Para ello se hace uso del software CodeBlocks que es una herramienta configurada para desarrollar el lenguaje C++. Para Talens. (1995). Inicialmente, se desarrollan las codificaciones de cada uno de los métodos de ordenamientos anteriormente mencionados para los arreglos de enteros de 100 números y posteriormente se los compila y se hace correr en programa CodeBlocks para posteriormente tomar el tiempo de ejecución de cada programación en cada ordenamiento y compararlos entre sí, y de esta manera tener un criterio de optimalidad en base a la velocidad de resolución y sus principios conceptuales. Desde un punto de vista analítico, el presente trabajo se compone de 3 fases, además de la introducción y las conclusiones.

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

En la primera parte se realiza una revisión sistematizada de la literatura referencial con el fin de comprender mejor, el cómo se desarrollan las programaciones y la forma de programación de cada una. En la segunda parte se describe la metodología aplicada para el desarrollo de esta investigación, así como las programaciones de los diferentes casos de ordenamiento para el arreglo de enteros de 100 elementos. Finalmente, en la tercera parte se describen los resultados de las programaciones luego de ser compiladas y hacerle correr el programa, a este tiempo de ejecución de los resultados del programa se lo denomina scoup.

Para estudiar cualquier lenguaje de programación se debe conocer cuáles son los conceptos que soporta, es decir el tipo de programación que se va a realizar. Como el C++ incorpora características nuevas con respecto a los lenguajes Pascal o , en primer lugar se da una descripción de los conceptos a los que este lenguaje da soporte.

Dentro de la metodología de programación clásica se definen una serie de fases en el proceso de desarrollo de aplicaciones. No es mi intención repetirlas ahora, sólo quiero indicar que las nuevas metodologías de programación orientadas a objetos han modificado la forma de trabajar en estas etapas. El llamado ciclo de vida del software exigía una serie de etapas a la hora de corregir o modificar los programas, trabajando sobre todas las etapas del proceso de desarrollo. A mi modo de ver estas etapas siguen existiendo de una manera u otra, pero el trabajo sobre el análisis y diseño (que antes eran textos y diagramas, no código) es ahora posible realizarlo sobre la codificación: la idea de clase y objeto obliga a que los programas tengan una estructura muy similar a la descrita en las fases de análisis y diseño, por lo que un código bien documentado junto con herramientas que trabajan sobre el código (el browser, por ejemplo, nos muestra la estructura de las jerarquías de clases de nuestro programa) puede considerarse un modelo del análisis y el diseño (sobre todo del diseño, pero las clases nos dan idea del tipo de análisis realizado) Talens (1995).

Debido a que las estructuras de datos son utilizadas para almacenar información, para poder recuperar esa información de manera eficiente es deseable que aquella esté ordenada. Existen varios métodos para ordenar las diferentes estructuras de datos básicas. Los métodos sencillos por lo general requieren de aproximadamente n x n pasos para ordenar n elementos.
Los métodos simples son: insertion sort (o por inserción directa) selection sort, bubble sort, y shellsort, en dónde el último es una extensón al insertion sort, siendo más rápido. Los métodos más complejos son el quick-sort, el heap sort, radix y address-calculation sort. El ordenar un grupo de datos significa mover los datos o sus referencias para que queden en una secuencia tal que represente un orden, el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente. Talems, (1995).

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

Guardati, (2006) Manifiesta que la ordenación de burbuja (Bubble Sort en inglés) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas "burbujas". También es conocido como el método del intercambio directo. Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo el más sencillo de implementar.

Roldán, (2015) enfoca el ordenamiento Shell (Shell sort en inglés) como un algoritmo de ordenamiento. El método se denomina Shell en honor de su inventor Donald Shell. Su implementación original, requiere O(n2) comparaciones e intercambios en el peor caso. Un cambio menor presentado en el libro de V. Pratt produce una implementación con un rendimiento de O(n log2 n) en el peor caso. Esto es mejor que las O(n2) comparaciones requeridas por algoritmos simples pero peor que el óptimo O(n log n). Aunque es fácil desarrollar un sentido intuitivo de cómo funciona este algoritmo, es muy difícil analizar su tiempo de ejecución. El algoritmo Shell sort mejora el ordenamiento por inserción comparando elementos separados por un espacio de varias posiciones. Esto permite que un elemento haga "pasos más grandes" hacia su posición esperada. Los pasos múltiples sobre los datos se hacen con tamaños de espacio cada vez más pequeños. El último paso del Shell sort es un simple ordenamiento por inserción, pero para entonces, ya está garantizado que los datos del vector están casi ordenados.

Dentro de la Ordenación Por Selección considérese el algoritmo para ordenar un array A de enteros en orden ascendente, es decir, del número más pequeño al mayor. Es decir, si el array A tiene n elementos, se trata de ordenar los valores del array de modo que el dato contenido en A[0] sea el valor más pequeño, el valor almacenado en A[1] el siguiente más pequeño, y así hasta A[n-1], que ha de contener el elemento de mayor valor. El algoritmo se apoya en sucesivas pasadas que intercambian el elemento más pequeño sucesivamente con el primer elemento de la lista, A[0] en la primera pasada. En síntesis, se busca el elemento más pequeño de la lista y se intercambia con A[0], primer elemento de la lista. A[0] A[1] A[2].... A[n-1]. Después de terminar esta primera pasada, el frente de la lista está ordenado y el resto de la lista A[1], A[2]...A[n-1] permanece desordenado. La siguiente pasada busca en esta lista desordenada y selecciona el elemento más pequeño, que se almacena entonces en la posición A[1]. De este modo los elementos A[0] y A[1] están ordenados y la sublista A[2], A[3]...A[n-1] desordenada; entonces, se selecciona el elemento más pequeño y se intercambia con A[2]. El proceso continúa n − 1 pasadas y en ese momento la lista desordenada se reduce a un elemento (el mayor de la lista) y el array completo ha quedado ordenado. Benjumea, (2006).

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

El método de ordenación por inserción es similar al proceso típico de ordenar tarjetas de nombres (cartas de una baraja) por orden alfabético, que consiste en insertar un nombre en su posición correcta dentro de una lista o archivo que ya está ordenado. Así el proceso en el caso de la lista de enteros A = 50, 20, 40, 80, 30. El algoritmo correspondiente a la ordenación por inserción contempla los siguientes pasos: 1. El primer elemento A[0] se considera ordenado; es decir, la lista inicial consta de un elemento. 2. Se inserta A[1] en la posición correcta, delante o detrás de A[0], dependiendo de que sea menor o mayor. 3. Por cada bucle o iteración i (desde i=1 hasta n-1) se explora la sublista A[i-1] . . A[0] buscando la posición correcta de inserción; a la vez se mueve hacia abajo (a la derecha en la sublista) una posición todos los elementos mayores que el elemento a insertar A[i], para dejar vacía esa posición. 4. Insertar el elemento a la posición correcta. Guardati, (2006).

El algoritmo conocido como quicksort (ordenación rápida) recibe el nombre de su autor, Tony Hoare. La idea del algoritmo es simple, se basa en la división en particiones de la lista a ordenar, por lo que se puede considerar que aplica la técnica divide y vencerás. El método es, posiblemente, el más pequeño de código, más rápido, más elegante, más interesante y eficiente de los algoritmos de ordenación conocidos. El método se basa en dividir los n elementos de la lista a ordenar en dos partes o particiones separadas por un elemento: una partición izquierda, un elemento central denominado pivote o elemento de partición, y una partición derecha. La partición o división se hace de tal forma que todos los elementos de la primera sublista (partición izquierda) son menores que todos los elementos de la segunda sublista (partición derecha).

Las dos sublistas se ordenan entonces independientemente. Para dividir la lista en particiones (sublistas) se elige uno de los elementos de la lista y se utiliza como pivote o elemento de partición. Si se elige una lista cualquiera con los elementos en orden aleatorio, se puede seleccionar cualquier elemento de la lista como pivote, por ejemplo, el primer elemento de la lista. Si la lista tiene algún orden parcial conocido, se puede tomar otra decisión para el pivote. Idealmente, el pivote se debe elegir de modo que se divida la lista exactamente por la mitad, de acuerdo al tamaño relativo de las claves. Por ejemplo, si se tiene una lista de enteros de 1 a 10, 5 o 6 serían pivotes ideales, mientras que 1 o 10 serían elecciones «pobres» de pivotes. Una vez que el pivote ha sido elegido, se utiliza para ordenar el resto de la lista en dos sublistas: una tiene todas las claves menores que el pivote y la otra, todos los elementos (claves) mayores que o iguales que el pivote (o al revés). Estas dos listas parciales se ordenan recursivamente utilizando el mismo algoritmo; es decir, se llama sucesivamente al propio algoritmo quicksort. La lista final ordenada se consigue concatenando la primera sublista, el pivote y la segunda lista, en ese orden, en una única lista. La primera etapa de quicksort es la división o «particionado» recursivo de la lista hasta que todas las sublistas constan de sólo un elemento. Guardati, (2006).

Metodología

El método que se utiliza para esta investigación es el Método Analítico-Cuantitativo que consiste en la desmembración de un todo, descomponiéndolo

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

en sus partes o elementos para observar las causas, la naturaleza y los efectos. ... Es necesario conocer la naturaleza del fenómeno y objeto que se estudia para comprender su esencia. Se desarrolla la investigación inicialmente con la programación de los diferentes tipos de ordenamiento para caso como son: Selección, Inserción, Intercambio, Burbuja, Shell, y Quicksort. Para cada uno de estos métodos se ha desarrollado la programación respectiva.

Problema: Ordenamiento con un arreglo de enteros con 100 elementos.
Se Programa el problema bajo los parámetros y conceptos de cada método de ordenamiento para posteriormente obtener los resultados como son los tiempos "coup" de resolución en cada caso.

//METODO DE SELECCIÓN (1)

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
   int numeros[] =
{7,2,9,3,4,10,5,6,8,1,17,12,19,13,14,20,15,16,18,11,27,22,29,23,24,30,25,26
,28,21,37,32,39,33,34,40,35,36,38,31,47,42,49,43,44,50,45,46,48,41,57,52,5
9,53,54,60,55,56,58,51,67,62,69,63,64,70,65,66,68,61,77,72,79,73,74,80,75,
76,78,71,87,82,89,83,84,90,85,86,88,81,97,92,99,93,94,100,95,96,98,91};
   int i,j,aux,min;
   for(i=0;i<100;i++){
      min = i;
      for(j=i+1;j<100;j++){
         if(numeros[j] < numeros[min]){
            min = j;
         }
      }
      aux = numeros[i];
      numeros[i] = numeros[min];
      numeros[min] = aux;
   }
   cout<<"Orden Ascendente: ";
   for(i=0;i<100;i++){
         cout<<numeros[i]<<" ";
   }
   cout<<"\nOrden Descendente: ";
   for(i=99;i>=0;i--){
      cout<<numeros[i]<<"  ";
   }
   getch();
   cout<<"pulse una tecla para continuar: ";
   return 0;
}
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

// METODO DE INSERCION (2)

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
   int numeros[] =
{7,2,9,3,4,10,5,6,8,1,17,12,19,13,14,20,15,16,18,11,27,22,29,23,24,30,25,26
,28,21,37,32,39,33,34,40,35,36,38,31,47,42,49,43,44,50,45,46,48,41,57,52,5
9,53,54,60,55,56,58,51,67,62,69,63,64,70,65,66,68,61,77,72,79,73,74,80,75,
76,78,71,87,82,89,83,84,90,85,86,88,81,97,92,99,93,94,100,95,96,98,91};
   int i,pos,aux;
   for(i=0;i<100;i++){
      pos =i;
   aux = numeros[i];
   while((pos>0) && (numeros[pos-1] > aux)){
      numeros[pos]= numeros[pos-1];
      pos--;
      }
      numeros[pos] = aux;
   }
   cout<<"Orden Ascendente: ";
   for(i=0;i<99;i++){
      cout<<numeros[i]<<"  ";
   }
   cout<<"\nOrden Descendente: ";
   for(i=99;i>=0;i--){
      cout<<numeros[i]<<"  ";
   }
   getch();
   return 0;
}
```

// METODO DE INTERCAMBIO (3)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int n=100;
void ordIntercambio(int a[]);
void imprimirArreglo(int a[]);

int main() {
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

```
    bool condicion=true;
    while(condicion) {
       int arreglo[n]=
{7,2,9,3,4,10,5,6,8,1,17,12,19,13,14,20,15,16,18,11,27,22,29,23,24,30,25,26
,28,21,37,32,39,33,34,40,35,36,38,31,47,42,49,43,44,50,45,46,48,41,57,52,5
9,53,54,60,55,56,58,51,67,62,69,63,64,70,65,66,68,61,77,72,79,73,74,80,75,
76,78,71,87,82,89,83,84,90,85,86,88,81,97,92,99,93,94,100,95,96,98,91};
       imprimirArreglo(arreglo);
       cout << "Escoja el metodo de ordenamiento:" << endl;
       cout << "1. Intercambio" << endl;

       int c;
       cin >> c;
       switch(c) {
       case 1:
          ordIntercambio(arreglo);
          imprimirArreglo(arreglo);
          break;
       }
       system("pause");
       system("cls");
    }
    return 0;
  }
  void ordIntercambio(int a[]) {
    int i,j;
    for (i=0; i<=n-2; i++)
       for (j=i+1; j<=n-1; j++)
          if(a[i]>a[j]) {
             int aux;
             aux=a[i];
             a[i]=a[j];
             a[j]=aux;
          }
  }
  void imprimirArreglo(int a[]) {
    int i;
    for (i=0; i<n; i++)
       cout << a[i] << "\t" ;
    cout << endl;
}
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

// METODO BURBUJA (4)

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
   int numeros[]=
{7,2,9,3,4,10,5,6,8,1,17,12,19,13,14,20,15,16,18,11,27,22,29,23,24,30,25,26
,28,21,37,32,39,33,34,40,35,36,38,31,47,42,49,43,44,50,45,46,48,41,57,52,5
9,53,54,60,55,56,58,51,67,62,69,63,64,70,65,66,68,61,77,72,79,73,74,80,75,
76,78,71,87,82,89,83,84,90,85,86,88,81,97,92,99,93,94,100,95,96,98,91};
   int i,j,aux;
   for(i=0;i<100;i++){
      for(j=0;j<100;j++){
         if(numeros[j] > numeros[j+1]){
            aux = numeros[j];
            numeros[j] = numeros[j+1];
            numeros[j+1] = aux;
         }
      }
   }
   cout<<"Orden Ascencente:  ";
   for(i=0;i<100;i++){
      cout<<numeros[i]<<"  ";
   }
   cout<<"\nOrden Descencente:  ";
   for(i=99;i>=0;i--){
      cout<<numeros[i]<<"  ";
   }
      return 0;
}
```

// METODO SHELL (5)

```
// Ordenamiento Shell
#include <iostream>
#include <iostream>
#include <cstdlib>

using namespace std;

int n=100;
void imprimirArreglo(int a[]);
void ordenacionShell(int a[]);

int main() {
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

```
    bool condicion=true;

    while(condicion) {
       // Grupo de 100 Numeros enteros
       int arreglo[n]= {7,2,9,3,4,10,5,6,8,1,17,12,19,13
,14,20,15,16,18,11,27,22,29,23,24,30,25,26,28,21,37,32,39,33,34,40,35,36,3
8,31,47,42,49,43,44,50,45,46,48,41,57,52,59,53,54,60,55,56,58,51,67,62,69,
63,64,70,65,66,68,61,77,72,79,73,74,80,75,76,78,71,87,82,89,83,84,90,85,8
6,88,81,97,92,99,93,94,100,95,96,98,91
                    };
       imprimirArreglo(arreglo);
       cout << "Escoja el metodo de ordenamiento:" << endl;
       cout << "1. Shell" << endl;
       cout << "2. Salir" << endl;
       int c;
       cin >> c;
       //Condicion
       switch(c) {

       case 1:
          ordenacionShell (arreglo);
          imprimirArreglo(arreglo);
          break;

       case 2:
          condicion=false;
          cout << "Adios" << endl;
          break;
       default:
          cout << "Debe escoger una opcion valida" << endl;
          break;
       }
       system("pause");
       system("cls");
    }
    return 0;
 }


 void imprimirArreglo(int a[]) {
    int i;
    for (i=0; i<n; i++)
       cout << a[i] << "\t" ;
    cout << endl;
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

```
}
void ordenacionShell(int a[]) {

   {
      int intervalo, i, j, k;
      intervalo = n / 2;
      while (intervalo > 0) {

         for (i = intervalo; i < n; i++) {
            j = i - intervalo;
            while (j >= 0) {
               k = j + intervalo;
               if (a[j] <= a[k])
                  j = -1; /* así termina el bucle, par ordenado */
               else {
                  int temp;
                  temp = a[j];
                  a[j] = a[k];
                  a[k] = temp;
                  j -= intervalo;
               }
            }
         }
         intervalo = intervalo / 2;
      }
   }
}
```

// METODO QUICKSORT (6)

```
#include <iostream>
using namespace std;
void Quicksort(int* arr, int izq, int der)
{
   int i = izq, j = der, tmp;
   int p = arr[(izq + der) / 2];
  while (i <= j){
      while (arr[i] < p) i++;
      while (arr[j] > p) j--;
      if (i <= j){
        tmp = arr[i];
        arr[i] = arr[j];
        arr[j] = tmp;
        i++; j--;
      }
   }
  if (izq < j)
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

```
            Quicksort(arr, izq, j);
        if (i < der)
          Quicksort(arr, i, der);
  }
  int main ()
  {
     cout<< "Este es el Metodo Quicksort\n\n";
     cout<< "Esta es la lista a ordenar\n";
     cout<<
"7,2,9,3,4,10,5,6,8,1,17,12,19,13,14,20,15,16,18,11,27,22,29,23,24,30,25,26
,28,21,37,32,39,33,34,40,35,36,38,31,47,42,49,43,44,50,45,46,48,41,57,52,5
9,53,54,60,55,56,58,51,67,62,69,63,64,70,65,66,68,61,77,72,79,73,74,80,75,
76,78,71,87,82,89,83,84,90,85,86,88,81,97,92,99,93,94,100,95,96,98,91\n\n
";
     cout<< "La lista de abajo ya se ha ordenado:\n\n";
       int                                arreglo[100]                                =
{7,2,9,3,4,10,5,6,8,1,17,12,19,13,14,20,15,16,18,11,27,22,29,23,24,30,25,26
,28,21,37,32,39,33,34,40,35,36,38,31,47,42,49,43,44,50,45,46,48,41,57,52,5
9,53,54,60,55,56,58,51,67,62,69,63,64,70,65,66,68,61,77,72,79,73,74,80,75,
76,78,71,87,82,89,83,84,90,85,86,88,81,97,92,99,93,94,100,95,96,98,91};
       Quicksort(arreglo,0,99);
         for(int i = 0; i < 100; i++)
             cout << arreglo[i] << " ";
    return 0;
 }
```

Resultados y discusión

Tabla 1. Mediciones de tiempo "coup"

MÉTODO DE ORDENAMEINTO	TIEMPO DE PROMEDIO	ANÁLISIS DE RESULTADOS
Selección	8 seg.	Óptimo
Inserción	9 seg.	Óptimo
Intercambio	8 seg.	Óptimo
Burbuja	14 seg.	Bueno
Shell	7 seg.	Muy Bueno
Quicksort	9 seg.	Óptimo

Fuente: Elaboración propia a partir del trabajo en laboratorio de la aplicación del software Code::Blocks bajo C++.

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

Aplicación en el Área Contable

El desarrollo de la tecnología hace posible optimizar tiempos de ejecución en los procesos contables, es por esta razón que se quiere sugerir una programación que cuente el número de tipos de compra en un almacén de electrodomésticos, de tal manera que realice un resumen de las compras de contado y las compras a crédito en mensualidades de 6, 12, 16, y 24 meses, contabilizando el número de facturas en cada caso, la programación hace referencia a la contabilización de ventas de contado y crédito que se realizan, tomando en cuenta que si son ventas a crédito se deberá especificar si son a cada plazo anteriormente propuesto.

Caso Práctico de Aplicación 1

En la feria del hogar se ha encontrado una tienda que vende al contado y al crédito en 6, 12, 18, 24 mensualidades sus artefactos: televisores, refrigeradoras, hornos micro hondos. Se desea saber en cualquier momento cuantas ventas se han realizado al contado, cuantas, al crédito en 6, cuantas, en 12, cuantas, en 18, cuantas en 24 mensualidades.

Programación en C++

```
//[fpermissive]
int main() {
   char ventas='S';
   int tipoventa, tipocred;
   int ncont=0, ncre6=0,ncre12=0,ncre18=0,ncre24=0;
   while(ventas=='S') {
      printf("Ingrese en tipo de venta (1)=Contado, (2)=Credito:");
      scanf("%d",&tipoventa);
      if(tipoventa==2) {
         printf("Ingrese el tipo de credito en mensualidades
(6),(12),(18),(24):");
         scanf("%d",&tipocred);
         switch (tipocred) {
         case 6:
            ncre6++;
            break;
         case 12:
            ncre12++;
            break;
         case 18:
            ncre18++;
            break;
         case 24:
            ncre24++;
            break;
         }
      } else
         ncont++;
      printf("\n Desea realizar mas ventas? (S)=Si, (N)=No:");
      scanf("%c",&ventas);
```

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

```
    }
    printf("\n Numero de ventas al contado:%d",ncont);
    printf("\n Numero de ventas a credito en 06 meses:%d",ncre6);
    printf("\n Numero de ventas a credito en 12 meses:%d",ncre12);
    printf("\n Numero de ventas a credito en 18 meses:%d",ncre18);
    printf("\n Numero de ventas a credito en 24 meses:%d",ncre24);
    getch();
}
```

A continuación se presentan los gráficos de secuencia de la ejecución del programa en Code::Blocks de manera que se entienda como se ingresan los datos y cuáles son los resultados a obtener en la corrida de programa. Se cuenta con un menú para poder decidir cual opción es la que desea obtener como resultado. Es importante tomar en cuenta que este tipo de programaciones aportan considerablemente en el desarrollo y la productividad de las empresas ya que minimizan el tiempo de los procesos de recolección de datos para los informes finales de contabilidad. Cabe resaltar que se pueden desarrollar programas completos de análisis financieros contables que coadyuven con la importancia de la Eficiencia y Eficacias de las Empresas.

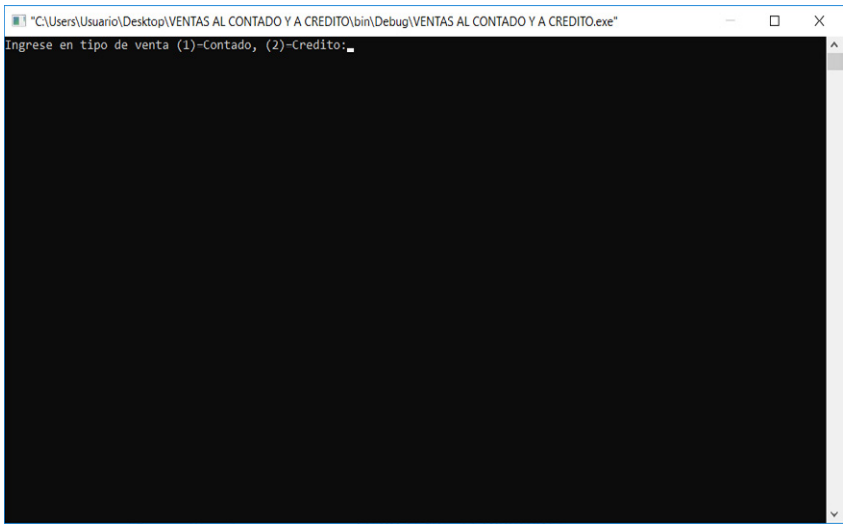

Gráfico 1. Ingreso en tipo de venta

Fuente: *Elaboración propia a partir del trabajo en laboratorio de la aplicación del software Code::Blocks bajo C++.*

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

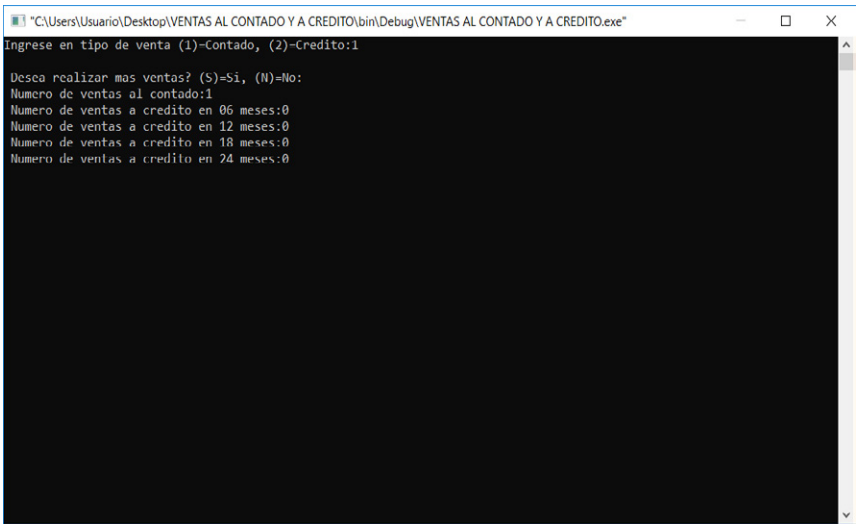

Gráfico 2. Opción de ventas adicionales

Fuente: *Elaboración propia a partir del trabajo en laboratorio de la aplicación del software Code::Blocks bajo C++.*

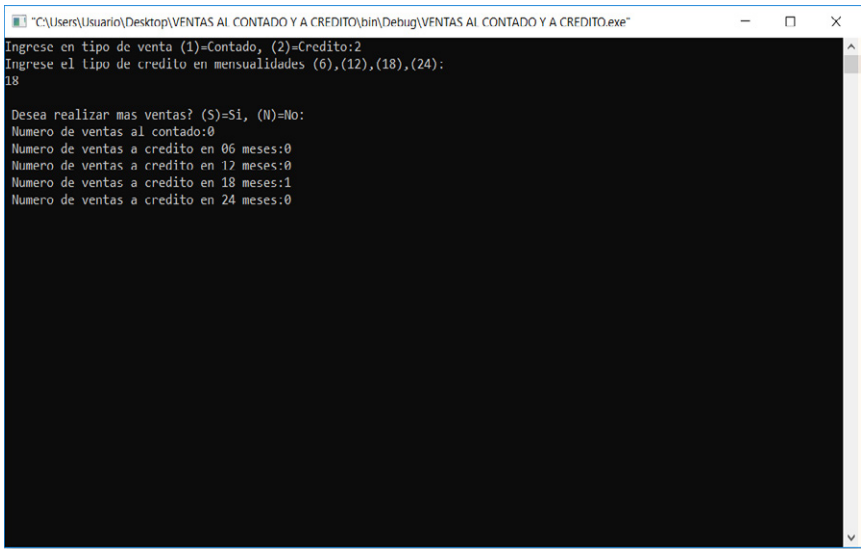

Gráfico 3. Ventas a 18 meses

Fuente: *Elaboración propia a partir del trabajo en laboratorio de la aplicación del software Code::Blocks bajo C++.*

Caso Práctico de Aplicación 2

También se puede analizar una Aplicación de conversión de monedas, en este caso de realiza la codificación de la transformación monetaria de la equivalencia de Euros a Dólares Norteamericanos. Al tratarse de una aplicación pequeña se establece una codificación bastante simple que puede ser acrecentada en las diferentes moneadas de mundo entero, comprobando nuevamente que es esta una herramienta poderosa y crear la conciencia en los estudiantes, profesionales

Revista Ciencia & Tecnología
No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734

y público en general que se pueden tener diferentes tipos de Aplicaciones que nuevamente se realza la idea de optimización de recursos, concretamente se está hablando de la variable tiempo.

Programación en C++

```
#include <iostream>
using namespace std ;
const double dolares = 1.1327 ;
int main() {
    cout << "Introduce la cantidad (en euros): " ;
    double euros ;
    cin >> euros ;
    double pesetas = euros * dolares ;
    cout << euros << " Euros equivalen a " << pesetas << " dolares" << endl
;
    // return 0 ;
}
```

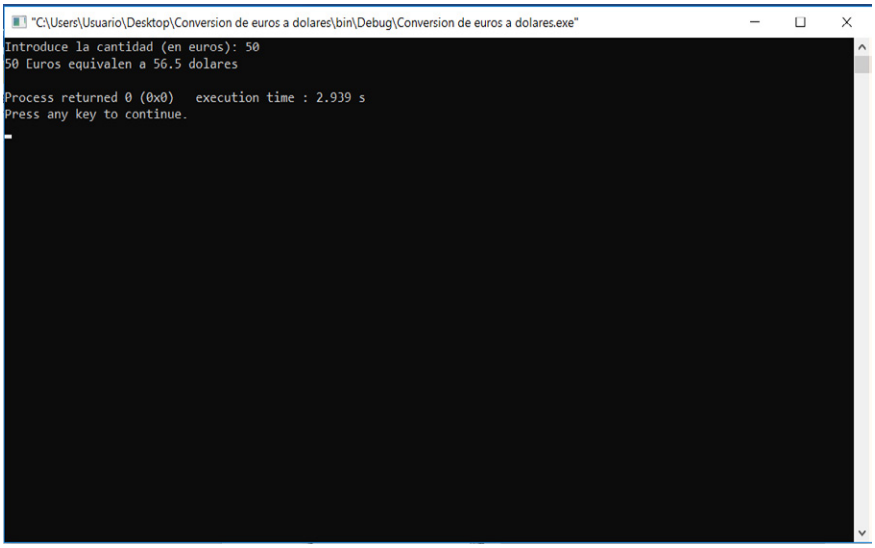

Gráfico 4. Tiempo de ejecución

Fuente: *Elaboración propia a partir del trabajo en laboratorio de la aplicación del software Code::Blocks bajo C++.*

Conclusiones

Al analizar los elementos de la presente investigación se puede denotar los siguientes puntos: Las Programaciones de cada uno de los métodos de ordenamiento en el lenguaje C++, tienen sus propios criterios y fundamentos científico-analíticos los cuales describen la solución al problema de la misma manera, pero con diferente programación sin embargo el resultado siempre es el esperado, ordenar los 100 elementos numéricos. Correlacionado los resultados de cada uno de los métodos de ordenamiento se puede establecer que La Selección, Inserción e Intercambio son los métodos más óptimos para hallar la

solución al problema planteado de una manera eficiente, posteriormente el método Burbuja y el Quicksort que son método no tan buenos como los primeros citados y por último el método que presenta problemas de eficiencia para su desarrollo y para entregar la respuesta es el método Shell que presenta más duración en su ejecución. Cabe resaltar que los 6 métodos cumplieron el objetivo final que es, ordenar los 100 elementos numéricos con sus programaciones específicas bajo los parámetros conceptuales individuales. La complejidad computacional que presenta cada uno de los métodos refleja la velocidad con la cual entregan los resultados finales con respecto a su límite asintótico (teórico) y los resultados experimentales (tiempo de ejecución reportado por el Code::Block). Es importante tomar en cuenta que se corrió la rutina de ejecución de los diferentes métodos de ordenamiento tres veces para luego obtener un promedio de los tiempos de ejecución de resultados experimentales. Se puede concluir que la presente investigación evidencia la versatilidad y potencialidad de una herramienta informática que puede llegar a reducir costos de operación en diferentes procesos y procedimientos contables, los mismos que representan un valor monetario para las empresas en la actualidad.

Referencias bibliográficas

Basanta-Val, P., García-Valls, M., & López-Anastasio, P. (2013). Herramienta Web Ligera para La Programación en C-Concurrente. Revista Iberoamericana de Automática e Informática Industrial RIAI. https://doi.org/10.1016/j.riai.2013.05.010

Vicente Benjumea y Manuel Roldán (2017), Fundamentos de Programación con el Lenguaje de Programación C++.

CONAF. (2014). Plan de manejo Reserva Nacional Coyhaique. Corporación Nacional Forestal. Gobierno de Chile.

Cruz, H., Making Supercomputing Available to All Cuban Researchers. Computing in Science & Engineering, 2018, 20 (3): p. 25-30. [Links]

Galeno, G. (2013). Programación de sistemas embebidos en C. Journal of Chemical Information and Modeling.

Golemon, S. libssh2. 2020 [En línea]. [Consultado el: 20/01/2020] Disponible en: Disponible en: https://www.libssh2.org . [Links]

Guardati, 2006. Estructura de datos. Tercera edición.

Gutiérrez, R. (2018). Manual de Programación. ArduinoBot.

Joyanes Aguilar, L., & Zahonero Martínez, I. (2014). Programación en C, C++, Java y UML. In BMC Public Health.

Mantilla Güiza, R. R. (2014). Una mirada entre programación paralela y a tradicional práctica educativa. I+D Revista de Investigaciones. https://doi.org/10.33304/revinv.v04n2-2014003

Martín Valverde, A. (2002). Fronteras y "zonas grises" del Derecho del Trabajo en la jurisprudencia actual (1980-2001). Revista Del Ministerio de Trabajo e Inmigración.

Oliag, S. T. (1995). Curso de programación en C ++ Apuntes de clase. EUI (UPV) Valencia.

Olivares Flores, L. I. (2018). Manual de programación en Lenguaje C++. Métodos de Funciones de Base Radial Para La Solución EDP.

Potter, J. ParWeb: a front-end interface for cluster computing. Honors Theses, Western Michigan University, Michigan, 2014. [Links]

Raffo Lecca, E. (2014). Programación genérica en C++, usando Metaprogramación. Industrial Data. https://doi.org/10.15381/idata.v10i1.6353

Ragni, M. . Force[En línea]. 2013, [Consultado el: 22/01/2020] Disponible en: Disponible en: https://www.ragni.me/Force/ . [Links]

Rebora, A. (1978). El ordenamiento territorial y urbano en México Problemas y perspectivas. Comercio Exterior.

Rodríguez Castillo, I., & Rodríguez Alcantar, E. (2019). Programación en C++, paradigma estructurado. In Programación en C++, paradigma estructurado. https://doi.org/10.47807/unison.49

Roldán, (2015). Fundamentos de Programación con el Lenguaje de programación C++.

Ruiz Rodríguez, R. (2019). Una Introducción a la Programación Estructurada en C. In Journal of Chemical Information and Modeling.

Sangopanta Cajas, P. R., Mérelo Gil, Antonio, B., & Quinatoa Arequipa, E. E. (2019). Analizar un código a través de lenguajes de programación C ++ y Code :: Blocks. Ciencias de La Ingeniería y Aplicadas.

Sampedro, Z.; Hauser, T.; Sood, S. Sandstone HPC: A domain-general gateway for new HPC users. En: Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact. ACM, 2017, p. 33. [Links]

Sarasty, H. Documentación y análisis de los principales frameworks de arquitectura de software en aplicaciones empresariales. Tesis Doctoral, Universidad Nacional de La Plata, 2016. [Links]

Sidhu, R., et al. Machine Learning Based Datacenter Monitoring Framework. Master Theses, The University of Texas, 2016. [Links]

Sterling, T.; Anderson, M.; Brodowicz, M. Chapter 1 Introduction. En: High performance computing: Modern Systems and Practices. San Francisco, C.A.: Morgan Kaufmann Publishers Inc, 2017. p. 1-12. [Links]

Universidad De Oriente. Guía de Usuario del HPC UO. [En línea]. 2019, [Consultado el: 22/01/2020] Disponible en: Disponible en: https://wiki.hpc.uo.edu.cu/doku.php?id=hpc-guia#referencia_en_publicaciones . [Links]

Universidad Nacional de Rosario. (2008). Programación Orientada a Objetos en C++. Facultad de Ciencias Exactas, Ingeniería y Agrimensura.

(Talens, 1995). Curso de Programación en C++.

Revista Ciencia & Tecnología

No. 33, 31 de enero de 2022
ISSN impreso: 1390 - 6321
ISSN online: 2661 - 6734