ESCUELA POLITÉCNICA NACIONAL

PROGRAMACIÓN I

COMPONENTE PRÁCTICO

DOCENTE: Eddie Hans Yánez Quezada

ESTUDIANTE; Josune Antonella Singaña Tapia

FECHA: 21 de Agosto de 2022

Técnicas de ordenamiento

DEBER 10

Escuela Politécnica Nacional

Facultad De Ingeniería en Sistemas

PROGRAMACIÓN I (COMPONENTE PRÁCTICO)

DOCENTE: Eddie Hans Yánez Quezada
ESTUDIANTE: Josune Antonella Singaña Tapia
TEMA: Ejercicio técnicas de ordenamiento.
FECHA: 21 de agosto de 2022

INDICE

Objetivos
Problema 1

OBJETIVOS DEL DEBER:

- Conocer las técnicas de ordenamiento (intercambio, selección, burbuja, inserción, shell, quick sort, fusión).
- Analizar la estructura en lenguaje C del algoritmo desarrollado en pseint.

ACTIVIDAD EJERCICIO DE ALGORITMOS

Problema 1

1

Desarrollo			Etapa
Completar los algoritmos de ordenamiento los realizados en clase hasta la página 334. Diseñar un menú de opciones y codificar a C, cargar el archivos pdf y el archivo C			**Definición del problema**
Entrada	Proceso	Salida	**Análisis del problema**

Opc, lim, E enteros.	ordenIntercambio(vector,ne, niter por referencia)
	niter=0
	// ciclo va haste el penultimo elemento
	Para i=0 hasta ne-2 Hacer
	Para j=i+1 Hasta ne-1 Hacer
	// para orden ascendente > para orden descendente <
	Si vector[i]>vector[j]Entonces
	aux=vector[i]
	vector[i]=vector[j]
	vector[j]=aux
	niter=niter+1
	FinSi
	FinPara
	FinPara
	FinSubProceso
		vect[i], ni E enteros	

```
Subproceso ordenSeleccion( vector,ne, niter por referencia)
        niter=0
        // ciclo va haste el penultimo elemento
        Para i=0 hasta ne-2 Hacer
                pos=i
                Para j=i+1 Hasta ne-1 Hacer
        // para orden ascendente > para orden descendente <
                        Si vector[pos]>vector[j]Entonces
                                pos = j
                        FinSi
                FinPara
                aux=vector[i]
                vector[i]=vector[pos]
                vector[pos]=aux
                niter=niter+1
        FinPara
FinSubProceso

Subproceso ordenBurbuja( vector,ne, niter por referencia)
        niter=0
        i=1
        cambio = verdadero
        // ciclo va haste el penultimo elemento
        Mientras i< ne-1 y cambio = Verdadero Hacer
                cambio = falso
                Para j=0 Hasta ne-i-1 Hacer
// para orden ascendente > para orden descendente <
                        Si vector[j]>vector[j+1]Entonces
                                aux=vector[j]
                                vector[j]=vector[j+1]
                                vector[j+1]=aux
                                cambio = verdadero
                                niter=niter+1
                        FinSi
                FinPara
                i = i + 1
        FinMientras
FinSubProceso

Subproceso ordenInsercion( v, n, niter por referencia)
        Definir i Como Entero
        Definir j Como Entero
        Definir aux Como Entero
        niter=0
        Para i = 1 hasta n-1 hacer
                j= i - 1
                aux=v[i]
                Mientras j >= 0 y v[j] > aux  hacer
                        v[j+1]=v[j]
                        j=j-1
                        niter=niter+1
                FinMientras
                v[j+1]= aux
        FinPara
FinSubProceso

Subproceso ordenshell( v, n, nshell por referencia)
        nshell=0
        salto=trunc(n/2)
        Mientras salto > 0 hacer
                Para j=(salto) hasta n-1 hacer
                i= j - salto
                aux=v[j]
                seguir=Verdadero
                Mientras i >= 0 y seguir=verdadero hacer
                        Si aux< v[i] Entonces
                        v[i+salto]=v[i]
                        i=i-salto
                        nshell=nshell+1
                Sino
                        seguir=falso
                FinSi
                FinMientras
                v[i+salto]= aux
        FinPara
        salto=trunc(salto/2)
FinMientras
FinSubProceso
```

```
Subproceso ordenquicksort( v, inf, sup,nquick por referencia)
        m=inf
        n=sup
        pivote=trunc((v[m]+v[n])/2)
        Mientras m<n Hacer
                Mientras v[m]<pivote Hacer
                        m=m+1
                FinMientras
                Mientras v[n]>pivote Hacer
                        n=n-1
                FinMientras
                Si m<=n Entonces
                        aux=v[m]
                        v[m]=v[n]
                        v[n]=aux
                        m=m+1
                        n=n-1
                        nquick=nquick+1
                FinSi
        FinMientras
        Si inf<n Entonces
                ordenquicksort(v,inf,n,nquick)
        FinSi
        Si sup>m Entonces
                ordenquicksort(v,m,sup,nquick)
        FinSi
Finsubproceso

Subproceso ordenFusionar( a, b, v, m, ns, nifu por referencia)
        Definir j,k, i, x Como Entero
        i=0
        j=0
        k=0
        x=0
        Mientras i <= m-1 y j <= ns-1 hacer
                Si a[i] <= b[j] entonces
                        v[k] = a[i]
                        i = i + 1
                        Sino
                                v[k] = b[j]
                                j = j + 1
                                nifu=nifu+1
                                Fin si
                               k = k + 1
        Fin mientras
                Si i <= m-1 entonces
                 Para x = i hasta m-1 hacer
                        v[k] = a[x]
         k = k + 1

nifu=nifu+1
Fin para
Fin si
        Si j <= ns entonces
        Para x = j hasta ns-1 hacer
                v[k] = b[x]
                k = k + 1
                nifu=nifu+1
        Fin para
Fin si
FinSubProceso
```

Imagen obtenida en pseint

```
1  Subproceso ordenIntercambio( vector,ne, niter por referencia)
2      Definir i Como Entero
3      Definir j Como Entero
4      Definir aux Como Entero
5      niter=0
6      // ciclo va haste el penultimo elemento
7      Para i=0 hasta ne-2 Hacer
8          Para j=i+1 Hasta ne-1 Hacer
9              // para orden ascendente > para orden descendente <
10             Si vector[i]>vector[j]Entonces
11                 aux=vector[i]
12                 vector[i]=vector[j]
13                 vector[j]=aux
14                 niter=niter+1
15             FinSi
```

Diseño del algoritmo

```
        FinPara
    FinPara
FinSubProceso

Subproceso ordenSeleccion( vector,ne, niter por referencia)
    Definir i Como Entero
    Definir j Como Entero
    Definir pos Como Entero
    Definir aux Como Entero
    niter=0
    // ciclo va haste el penultimo elemento
    Para i=0 hasta ne-2 Hacer
        pos=i
        Para j=i+1 Hasta ne-1 Hacer
            // para orden ascendente > para orden descendente <
            Si vector[pos]>vector[j]Entonces
                pos = j
            FinSi
        FinPara
        aux=vector[i]
        vector[i]=vector[pos]
        vector[pos]=aux
        niter=niter+1
    FinPara
FinSubProceso

Subproceso ordenBurbuja( vector,ne, niter por referencia)
    Definir i Como Entero
    Definir j Como Entero
    Definir aux Como Entero
    Definir cambio Como Logico
    niter=0
    i=1
    cambio = verdadero
    // ciclo va haste el penultimo elemento
    Mientras i< ne-1 y cambio = Verdadero Hacer
        cambio = falso
        Para j=0 Hasta ne-i-1 Hacer
            // para orden ascendente > para orden descendente <
            Si vector[j]>vector[j+1]Entonces
                aux=vector[j]
                vector[j]=vector[j+1]
                vector[j+1]=aux
                cambio = verdadero
                niter=niter+1
            FinSi
        FinPara
        i = i + 1
    FinMientras
FinSubProceso

Subproceso ordenInsercion( v, n, niter por referencia)
    Definir i Como Entero
    Definir j Como Entero
    Definir aux Como Entero
    niter=0
    Para i = 1 hasta n-1 hacer
        j= i - 1
        aux=v[i]
        Mientras j ≥ 0 y v[j] > aux  hacer
            v[j+1]=v[j]
            j=j-1
            niter=niter+1
        FinMientras
        v[j+1]= aux
    FinPara
FinSubProceso
```

```
83
84  Subproceso ordenshell( v, n, nshell por referencia)
85      Definir i Como Entero
86      Definir j Como Entero
87      Definir salto Como Entero
88      Definir aux Como Entero
89      Definir seguir Como Logico
90      nshell=0
91      salto=trunc(n/2)
92      Mientras salto > 0 hacer
93          Para j=(salto) hasta n-1 hacer
94          i= j - salto
95          aux=v[j]
96          seguir=Verdadero
97          Mientras i ≥ 0 y seguir=verdadero hacer
98              Si aux< v[i] Entonces
99              v[i+salto]=v[i]
100             i=i-salto
101             nshell=nshell+1
102         Sino
103             seguir=falso
104         FinSi
105         FinMientras
106         v[i+salto]= aux
107     FinPara
108     salto=trunc(salto/2)
109 FinMientras
110 FinSubProceso
111
112
113 Subproceso ordenquicksort( v, inf, sup,nquick por referencia)
114 definir n, m, pivote,aux Como Entero
115     m=inf
116     n=sup
117     pivote=trunc((v[m]+v[n])/2)
118     Mientras m<n Hacer
119         Mientras v[m]<pivote Hacer
120             m=m+1
121         FinMientras
122         Mientras v[n]>pivote Hacer
123             n=n-1
124         FinMientras
125         Si m≤n Entonces
126             aux=v[m]
127             v[m]=v[n]
128             v[n]=aux
129             m=m+1
130             n=n-1
131             nquick=nquick+1
132         FinSi
133     FinMientras
134     Si inf<n Entonces
135         ordenquicksort(v,inf,n,nquick)
136     FinSi
137     Si sup>m Entonces
138         ordenquicksort(v,m,sup,nquick)
139     FinSi
140 FinSubProceso
```

```
141
142 Subproceso ordenFusionar( a, b, v, m, ns, nifu por referencia)
143     Definir j,k, i, x Como Entero
144     i=0
145     j=0
146     k=0
147     x=0
148     Mientras i ≤ m-1 y j ≤ ns-1 hacer
149         Si a[i] ≤ b[j] entonces
150             v[k] = a[i]
151             i = i + 1
152             Sino
153                 v[k] = b[j]
154                 j = j + 1
155                 nifu=nifu+1
156                 Fin si
157                 k = k + 1
158     Fin mientras
159         Si i ≤ m-1 entonces
160                         Para x = i hasta m-1 hacer
161                             v[k] = a[x]
162                             k = k + 1
163                             nifu=nifu+1
164                             Fin para
165                             Fin si
166     Si j ≤ ns entonces
167     Para x = j hasta ns-1 hacer
168         v[k] = b[x]
169         k = k + 1
170         nifu=nifu+1
171     Fin para
172 Fin si
173 FinSubProceso
174
175  SubProceso LLenarvector(vector, ne)
176      Definir i Como Entero
177      Para i=0 hasta ne-1 Hacer
178          vector[i]=azar(100)+1
179      FinPara
180  FinSubProceso
181  SubProceso igualarvector(vect1, vect2, ne)
182      Definir i Como Entero
183      Para i=0 hasta ne-1 Hacer
184          vect2[i]=vect1[i]
185      FinPara
186  FinSubProceso
187  SubProceso escribirvector(vect, ne, titulo)
188      Definir i Como Entero
189      Escribir " ", titulo, ": { " Sin Saltar
190      Para i=0 hasta ne-1 Hacer
191          Escribir vect[i], " " Sin Saltar
192      FinPara
193      Escribir " }"
194  FinSubProceso
195  Funcion opc =Menu()
196      definir opc Como Entero
197      Escribir" ---------------------------------------- "
198      Escribir"  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO "
```

```
199       Escribir " 1   INTERCAMBIO"
200       Escribir " 2   SELECCION"
201       Escribir " 3   BURBUJA"
202       Escribir " 4   INSERCION"
203       Escribir " 5   SHELL"
204       Escribir " 6   QUICK SORT"
205       Escribir " 7   FUSION"
206       Escribir " 8   SALIR"
207       Escribir " "
208       Escribir " INGRESAR OPCION:" Sin Saltar
209       Leer opc
210       Escribir" ---------------------------------------- "
211   FinFuncion
212   Algoritmo tecnicasOrdenamiento
213       Definir listaOrd Como Entero
214       Definir listaDes Como Entero
215       Definir mitad Como Entero
216       Definir lim Como Entero
217       Definir dim Como Entero
218       Definir ni Como Entero
219       Definir i Como Entero
220       Definir v2 Como Entero
221       Definir v3 Como Entero
222       Definir opc Como Entero
223       //autor= Singaña Josune
224       // fecha= 15-agosto-2022
225       opc=0
226       Escribir "ALGORITMOS DE ORDENAMIENTO "
227       Escribir "CUANTOS DATOS DESEA ORDENAS: " Sin Saltar
228       Leer lim
229       ni=0
230       Dimension listaOrd[lim]
231       Dimension listaDes[lim]
232       LLenarvector(listaDes,lim)
233     Repetir
234         ni=0
235         escribirvector(listaDes,lim,"VECTOR DATOS DESORDENADOS ")
236         igualarvector(listaDes, listaOrd, lim)
237         opc=menu()
238         Segun opc
239     1: ordenIntercambio(listaOrd,lim, ni)
240     escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR INTERCAMBIO")
241     Escribir "NUMERO DE INTERCAMBIOS: ",ni
242     2:ordenSeleccion(listaOrd,lim, ni)
243     escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR SELECCION")
244     Escribir "NUMERO DE INTERCAMBIOS: ",ni
245
246     3:ordenBurbuja(listaOrd,lim, ni)
247     escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR BURBUJA")
248     Escribir "NUMERO DE INTERCAMBIOS: ",ni
249
250     4:ordenInsercion(listaOrd,lim, ni)
251     escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR INSERCION")
252     Escribir "NUMERO DE INTERCAMBIOS: ",ni
253
254     5:ordenshell(listaOrd,lim, ni)
255     escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR SHELL")
256     Escribir "NUMERO DE INTERCAMBIOS: ",ni
257     6:ordenquicksort(listaOrd, 0,lim-1,ni)
258     escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR QUICK SORT")
259     Escribir "NUMERO DE INTERCAMBIOS: ",ni
260   7:mitad=trunc(lim/2)
261     Dimension v2[mitad]
```

```
262       Para i=0 hasta mitad-1 hacer
263           v2[i]=listaOrd[i]
264       FinPara
265       dim=lim-mitad
266       Dimension v3[dim]
267       Para i=0 hasta dim-1 hacer
268           v3[i]=listaOrd[i+mitad]
269       FinPara
270       ordenSeleccion(v2,mitad, ni)
271       ordenSeleccion(v3,dim, ni)
272       ordenFusionar(v2, v3, listaOrd, mitad, dim, ni)
273       escribirvector(listaOrd,lim,"VECTOR DATOS ORDENADOS POR FUSION")
274       Escribir "NUMERO DE INTERCAMBIOS: ",ni
275   8:Escribir">>>>>>>>>> GRACIAS POR USAR ESTA APLICACION <<<<<<<<<<<<<<<"
276   FinSegun
277   Si opc<8 Entonces
278       Escribir"<<<<<<<<< PRESIONA CUALQUIER TECLA PARA CONTINUAR >>>>>>>>>>>>>>>"
279       Esperar Tecla
280   FinSi
281   Mientras Que opc<8
282   FinAlgoritmo
```

Imagen ejecución pseint

PROCESO Sin limpiar pantalla

```
PSeInt - Ejecutando proceso TECNICASORDENAMIENTO
*** Ejecución Iniciada. ***
ALGORITMOS DE ORDENAMIENTO
CUANTOS DATOS DESEA ORDENAS: > 10
 VECTOR DATOS DESORDENADOS : { 57 78 19 14 41 50 23 8 86 54  }
 ----------------------------------------
  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO
 1   INTERCAMBIO
 2   SELECCION
 3   BURBUJA
 4   INSERCION
 5   SHELL
 6   QUICK SORT
 7   FUSION
 8   SALIR

 INGRESAR OPCION:> 5
 ----------------------------------------
 VECTOR DATOS ORDENADOS POR SHELL: { 8 14 19 23 41 50 54 57 78 86  }
NUMERO DE INTERCAMBIOS: 15
<<<<<<<<< PRESIONA CUALQUIER TECLA PARA CONTINUAR >>>>>>>>>>>>>>>
 VECTOR DATOS DESORDENADOS : { 57 78 19 14 41 50 23 8 86 54  }
 ----------------------------------------
  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO
 1   INTERCAMBIO
 2   SELECCION
 3   BURBUJA
 4   INSERCION
 5   SHELL
 6   QUICK SORT
 7   FUSION
 8   SALIR
 ----------------------------------------
  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO
 1   INTERCAMBIO
 2   SELECCION
 3   BURBUJA
 4   INSERCION
 5   SHELL
 6   QUICK SORT
 7   FUSION
 8   SALIR

 INGRESAR OPCION:> 8
 ----------------------------------------
>>>>>>>>>> GRACIAS POR USAR ESTA APLICACION <<<<<<<<<<<<<<<
*** Ejecución Finalizada. ***
```

Con limpiar pantalla

```
PSeInt - Ejecutando proceso TECNICASORDENAMIENTO

 INGRESAR OPCION:> 1
 ----------------------------------------
 VECTOR DATOS ORDENADOS POR INTERCAMBIO: { 8 14 19 23 41 50 54 57 78 86  }
NUMERO DE INTERCAMBIOS: 23
<<<<<<<<< PRESIONA CUALQUIER TECLA PARA CONTINUAR >>>>>>>>>>>>>>>
 VECTOR DATOS DESORDENADOS : { 57 78 19 14 41 50 23 8 86 54  }
 ----------------------------------------
  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO
 1   INTERCAMBIO
 2   SELECCION
 3   BURBUJA
 4   INSERCION
 5   SHELL
 6   QUICK SORT
 7   FUSION
 8   SALIR

 INGRESAR OPCION:> 3
 ----------------------------------------
 VECTOR DATOS ORDENADOS POR BURBUJA: { 8 14 19 23 41 50 54 57 78 86  }
NUMERO DE INTERCAMBIOS: 23
<<<<<<<<< PRESIONA CUALQUIER TECLA PARA CONTINUAR >>>>>>>>>>>>>>>
 VECTOR DATOS DESORDENADOS : { 57 78 19 14 41 50 23 8 86 54  }
```

Verificación del algoritmo

```
PSeInt - Ejecutando proceso TECNICASORDENAMIENTO
*** Ejecución Iniciada. ***
ALGORITMOS DE ORDENAMIENTO
CUANTOS DATOS DESEA ORDENAS: > 10
 VECTOR DATOS DESORDENADOS : { 70 67 62 10 57 30 9 49 98 48  }
 ----------------------------------------
  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO
 1   INTERCAMBIO
 2   SELECCION
 3   BURBUJA
 4   INSERCION
 5   SHELL
 6   QUICK SORT
 7   FUSION
 8   SALIR

 INGRESAR OPCION:> 7
 ----------------------------------------
 VECTOR DATOS ORDENADOS POR FUSION: { 9 10 30 48 49 57 62 67 70 98  }
NUMERO DE INTERCAMBIOS: 9
<<<<<<<<< PRESIONA CUALQUIER TECLA PARA CONTINUAR >>>>>>>>>>>>>>>
```

```
PSeInt - Ejecutando proceso TECNICASORDENAMIENTO
 VECTOR DATOS DESORDENADOS : { 31 17 79 100 42 45 1 20 92 30  }
 ----------------------------------------
  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO
 1   INTERCAMBIO
 2   SELECCION
 3   BURBUJA
 4   INSERCION
 5   SHELL
 6   QUICK SORT
 7   FUSION
 8   SALIR

 INGRESAR OPCION:> 8
 ----------------------------------------
>>>>>>>>>> GRACIAS POR USAR ESTA APLICACION <<<<<<<<<<<<<<<
*** Ejecución Finalizada. ***
```

Imagen obtenida en Dev c++

```
#include<iostream>
#include<cmath>
#include<cstdlib>
#include<conio2.h>
using namespace std;
#define ARREGLO_MAX 100
#define SIN_TIPO string
void ordenintercambio(int vector[], int ne, int &niter);
void ordenseleccion(int vector[], int ne, int &niter);
void ordenburbuja(int vector[], int ne, int &niter);
void ordeninsercion(int v[], int n, int &niter);
void ordenshell(int v[], int n, int &nshell);
void ordenquicksort(int v[], int inf, int sup, int &nquick);
void ordenfusionar(int a[], int b[], int v[], int m, int ns, int &nifu);
void llenarvector(int vector[], int ne);
void igualarvector(int vect1[], int vect2[], int ne);
void escribirvector(int vect[], int ne, char titulo[]);
int menu();

void ordenintercambio(int vector[], int ne, int &niter) {
    int aux, i, j;
    niter = 0;
    // ciclo va haste el penultimo elemento
    for (i=0;i<=ne-2;i++) {
        for (j=i+1;j<=ne-1;j++) {
            // para orden ascendente > para orden descendente <
            if (vector[i]>vector[j]) {
                aux = vector[i];
                vector[i] = vector[j];
                vector[j] = aux;
                niter = niter+1;
            }
        }
    }
}

void ordenseleccion(int vector[], int ne, int &niter) {
    int aux, i, j, pos;
    niter = 0;
    // ciclo va haste el penultimo elemento
    for (i=0;i<=ne-2;i++) {
        pos = i;
        for (j=i+1;j<=ne-1;j++) {
            // para orden ascendente > para orden descendente <
            if (vector[pos]>vector[j]) {
                pos = j;
            }
        }
        aux = vector[i];
        vector[i] = vector[pos];
        vector[pos] = aux;
        niter = niter+1;
    }
}

```

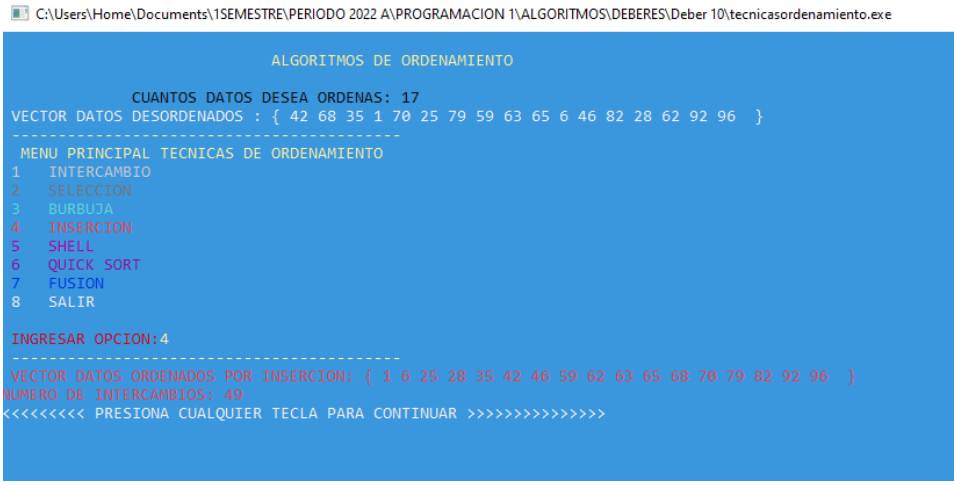
Diseño del algoritmo

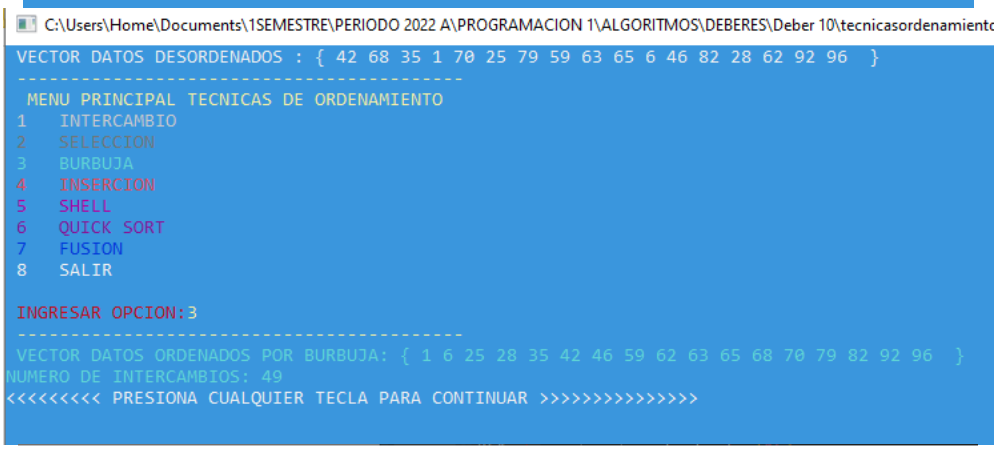
```
void ordenburbuja(int vector[], int ne, int &niter) {
    int aux, i, j;
    bool cambio;
    niter = 0;
    i = 1;
    cambio = true;
    // ciclo va haste el penultimo elemento
    while (i<ne-1 && cambio==true) {
        cambio = false;
        for (j=0;j<=ne-i-1;j++) {
            // para orden ascendente > para orden descendente <
            if (vector[j]>vector[j+1]) {
                aux = vector[j];
                vector[j] = vector[j+1];
                vector[j+1] = aux;
                cambio = true;
                niter = niter+1;
            }
        }
        i = i+1;
    }
}

void ordeninsercion(int v[], int n, int &niter) {
    int aux, i, j;
    niter = 0;
    for (i=1;i<=n-1;i++) {
        j = i-1;
        aux = v[i];
        while (j>=0 && v[j]>aux) {
            v[j+1] = v[j];
            j = j-1;
            niter = niter+1;
        }
        v[j+1] = aux;
    }
}

void ordenshell(int v[], int n, int &nshell) {
    int aux, i, j, salto;
    bool seguir;
    nshell = 0;
    salto = int(n/2);
    while (salto>0) {
        for (j=(salto);j<=n-1;j++) {
            i = j-salto;
            aux = v[j];
            seguir = true;
            while (i>=0 && seguir==true) {
                if (aux<v[i]) {
                    v[i+salto] = v[i];
                    i = i-salto;
                    nshell = nshell+1;
                } else {
                    seguir = false;
                }
            }
            v[i+salto] = aux;
        }
        salto = int(salto/2);
    }
}

void ordenquicksort(int v[], int inf, int sup, int &nquick) {
    int aux, m, n, pivote;
    m = inf;
    n = sup;
    pivote = int((v[m]+v[n])/2);
```

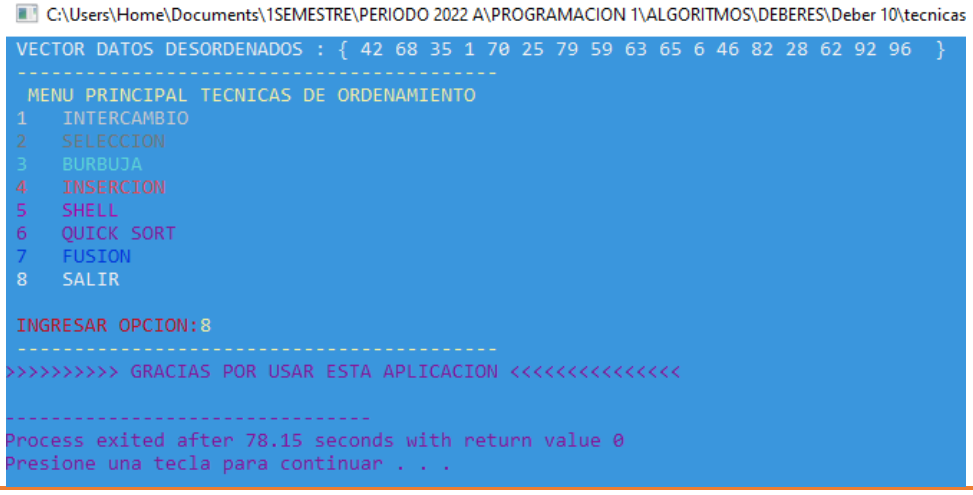
```
    while (m<n) {
        while (v[m]<pivote) {
            m = m+1;
        }
        while (v[n]>pivote) {
            n = n-1;
        }
        if (m<=n) {
            aux = v[m];
            v[m] = v[n];
            v[n] = aux;
            m = m+1;
            n = n-1;
            nquick = nquick+1;
        }
    }
    if (inf<n) {
        ordenquicksort(v,inf,n,nquick);
    }
    if (sup>m) {
        ordenquicksort(v,m,sup,nquick);
    }
}

void ordenfusionar(int a[], int b[], int v[], int m, int ns, int &nifu) {
    int i = 0;
    int j = 0;
    int k = 0;
    int x = 0;
    while (i<=m-1 && j<=ns-1) {
        if (a[i]<=b[j]) {
            v[k] = a[i];
            i = i+1;
        } else {
            v[k] = b[j];
            j = j+1;
            nifu = nifu+1;
        }
        k = k+1;
    }
    if (i<=m-1) {
        for (x=i;x<=m-1;x++) {
            v[k] = a[x];
            k = k+1;
            nifu = nifu+1;
        }
    }
    if (j<=ns) {
        for (x=j;x<=ns-1;x++) {
            v[k] = b[x];
            k = k+1;
            nifu = nifu+1;
        }
    }
}

void llenarvector(int vector[], int ne) {
    int i;
    for (i=0;i<=ne-1;i++) {
        vector[i] = (rand()%100)+1;
    }
}

void igualarvector(int vect1[], int vect2[], int ne) {
    int i;
    for (i=0;i<=ne-1;i++) {
        vect2[i] = vect1[i];
    }
}

void escribirvector(int vect[], int ne, char titulo[]) {
    int i;
    cout << " " << titulo << ": { ";
    for (i=0;i<=ne-1;i++) {
        cout << vect[i] << " ";
    }
    cout << " }" << endl;
}

int menu() {
    int opc;
        textcolor(14);
```

```
    cout << " ---------------------------------------- " << endl;
    cout << "  MENU PRINCIPAL TECNICAS DE ORDENAMIENTO " << endl;
        textcolor(7);cout << " 1   INTERCAMBIO" << endl;
        textcolor(8);cout << " 2   SELECCION" << endl;
        textcolor(11);cout << " 3   BURBUJA" << endl;
        textcolor(12);cout << " 4   INSERCION" << endl;
        textcolor(13);cout << " 5   SHELL" << endl;
        textcolor(5);cout << " 6   QUICK SORT" << endl;
        textcolor(1);cout << " 7   FUSION" << endl;
        textcolor(15);cout << " 8   SALIR" << endl;
       cout << " " << endl;
        textcolor(4);cout << " INGRESAR OPCION:";
        textcolor(14);cin >> opc;
        textcolor(14);cout << " ---------------------------------------- " << endl;
    return opc;
}

int main() {
        textbackground(3);
    clrscr();
    textcolor(14);
    int dim, i, lim, mitad, ni, opc;
    int v2[ARREGLO_MAX];
    int v3[ARREGLO_MAX];
    // autor= Singaña Josune
    // fecha= 15-agosto-2022
    opc = 0;
     gotoxy (30,2); cout << "ALGORITMOS DE ORDENAMIENTO " << endl;
        textcolor(0);
       gotoxy (15,4);cout << "CUANTOS DATOS DESEA ORDENAS: ";
    cin >> lim;
    ni = 0;
        int listades[lim];
    int listaord[lim];
    llenarvector(listades,lim);
    do {
        ni = 0;
            textcolor(15);
        escribirvector(listades,lim,"VECTOR DATOS DESORDENADOS ");
        igualarvector(listades,listaord, lim);
        opc = menu();
      switch (opc) {
      case 1:
          textcolor(7);
          ordenintercambio(listaord,lim,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR INTERCAMBIO");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 2:
          textcolor(8);
          ordenseleccion(listaord,lim,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR SELECCION");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 3:
          textcolor(11);
          ordenburbuja(listaord,lim,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR BURBUJA");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 4:
          textcolor(12);
          ordeninsercion(listaord,lim,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR INSERCION");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 5:
          textcolor(13);
          ordenshell(listaord,lim,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR SHELL");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 6:
          textcolor(5);
          ordenquicksort(listaord,0,lim-1,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR QUICK SORT");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 7:
          textcolor(1);
          mitad = int(lim/2);
          for (i=0;i<=mitad-1;i++) {
      case 7:
          for (i=0;i<=mitad-1;i++) {
              v2[i] = listaord[i];
          }
          dim = lim-mitad;
          for (i=0;i<=dim-1;i++) {
              v3[i] = listaord[i+mitad];
          }
          ordenseleccion(v2,mitad,ni);
          ordenseleccion(v3,dim,ni);
          ordenfusionar(v2,v3,listaord,mitad,dim,ni);
          escribirvector(listaord,lim,"VECTOR DATOS ORDENADOS POR FUSION");
          cout << "NUMERO DE INTERCAMBIOS: " << ni << endl;
          break;
      case 8:
              textcolor(5);
          cout << ">>>>>>>>>> GRACIAS POR USAR ESTA APLICACION <<<<<<<<<<<<<<<" << endl;
          break;
      }
      if (opc<8) {
              textcolor(15);
          cout << "<<<<<<<<< PRESIONA CUALQUIER TECLA PARA CONTINUAR >>>>>>>>>>>>>>>" << endl;
      getch();
          clrscr();
      }
    } while (opc<8);
    return 0;
}
```

Imagen ejecución Dev C++







Verificación del algoritmo