Manual técnico

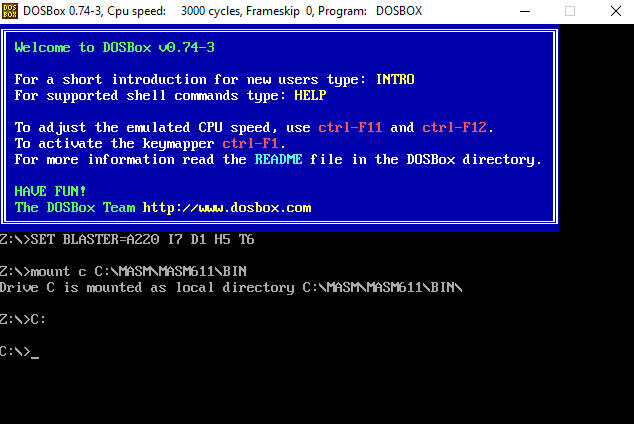
Proyecto Final

Andhy Lizandro Solís Osorio

Arqui1

**Entorno De Trabajo**

**DosBox**



Es un emulador de los procesadores 8086 y similares de Intel de a finales de los 80 y parte de los 90, permite la ejecución de programas casi con total naturalidad como si se tratara de una computadora antigua ejecutándolos

**MASM**

Fue el compilador usado para la creación del juego, fue un compilador desarrollado por Microsoft, la versión que fue utilizada solamente soporta registros de 16 bits, se compilaba por medio del comando ML y nombre de código fuente y generaba un ejecutable y el código objeto.

**Interrupciones Usadas**

**21H**

Interrupción utilizada para diversos métodos, permite entrada y salida de información del programa con el sistema

**10H**

Interrupción utilizada para mostrar información en pantalla colocando el emulador en modo.

**16H**

Interrupción para manejar las entradas del teclado cuando el elemento no esta representado por caracteres ASCII.

**Explicación Algoritmo Para Movimiento Del Vehículo**

Si se detecta tecla izquierda restar posición en X del vehículo por 1 si se detecta flecha derecha incrementar posición por 1 en caso contrario de tecla no esperada limpiar el buffer, si se detecta ESC poner en pausa.

**Explicación Algoritmo Para Colisión**

Si se esta en el largo del vehículo en Y o menos pasar a la siguiente verificación si se está entre el borde izquierdo o derecho indicar colisión.

**Explicación Algoritmo Para Graficar Barras**

Cargar en un vector los valores a ordenar y graficar esos valores ejecutar iteración de posicionamiento del ordenamiento y volver a graficar con el vector actualizado.

include Macros.asm ; archivo con los macros

include PJuego.asm

include DJuego.asm

include EJuego.asm

include GJuego.asm

include Usuario.asm

include Archivo.asm

include Estadi.asm

include Top.asm

include Barras.asm

include Orden.asm

.model small

.stack 500h

.data

;=======================Titulos

Titulo1  db "UNIVERSIDAD DE SAN CARLOS DE GUATEMALA",10,13,"$"

Titulo2 db "ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1A",10,13,"NOMBRE: ANDHY LIZANDRO SOLIS OSORIO",10,13,"$"

Titulo3 db "CARNET: 201700886",10,13,"SECCION: A",10,13,"$"

TituloTiempoBarras db "Ingrese La Velocidad Del Ordenamiento(0-9)","$"

Velocidad db 0

SinCarga db "No Hay Niveles Cargados","$"

Top10Puntos db "Top 10 Puntos","$"

Top10Tiempo db "Top 10 Tiempos","$"

BanderaCargado db 0

MenuPrincipal1 db "1)Ingresar","$"

MenuPrincipal2 db "2)Registrar","$"

MenuPrincipal3 db "3)Salir","$"

MenuUsuario1 db "1)Iniciar Juego","$"

MenuUsuario2 db "2)Cargar Juego","$"

MenuUsuario3 db "3)Salir","$"

MenuAdmin1 db "1)Top 10 puntos","$"

MenuAdmin2 db "2)Top 10 tiempo","$"

MenuAdmin3 db "3)Salir","$"

TituloUsuario db "Ingrese Credenciales Para Nuevo Usuario","$"

TituloCreado db "El Usuario Fue Creado Exitosamente","$"

TituloErrorArchivo db "El Archivo No Se Pudo Abrir","$"

TituloIngreseRuta db "Ingrese La Ruta Del Archivo .PLY","$"

TituloExtension db "La Extension Debe Ser .PLY","$"

TituloLogin db "Ingrese Sus Credenciales Para Ingresar","$"

TituloBienvenido db "Bienvenido Usuario: ","$"

TItuloLoginMalo db "Credenciales Erroneas","$"

Pausa db "Juego En Pausa Presione, Espacio Para Salir O Presione ESC para Regresar"

ArchivoDita db 400 dup(0)

RellenoUsu db "Usuario:","$"

RellenoContra db "Contra:","$"

ErrorContra db "Error La Contrasenia No Es Numerica","$"

ErrorUsuario db "El Usuario Ya Fue Creado","$"

AdvertenciaUsuario db "Advertencia Se Alncanzo El Limite De Usuarios","$"

Contra db "Ingrese Contrasenia","$"

User db "Ingrese Nombre De Usuario","$"

MaximoNivelPunteo db 0

MaximoNivelTiempo db 0

ArchivoUsuario db "User.txt",00h

ArchivoPunteo db "Puntos.txt",00h

ArchivoTiempo db "Tiempos.txt",00h

ReportePuntos db "Puntos.rep",00h

ReporteTiempo db "Tiempos.rep",00h

;=============Variables Del Juego=============

JugadorActual db 12 dup(0)

PosX db 0

Handle dw 0

Punteo dw 0

MaximoPunteo dw 0

MaximoTiempo dw 0

Obstaculo db 20 dup(0)

ObstaculoY db 20 dup(0)

ObstaculoX db 20 dup(0)

Bonus db 20 dup(0)

BonusY db 20 dup(0)

BonusX db 20 dup(0)

TiempoMaximo dw 0

ActualBonus db 0

BanderaBonus dw 0

BanderaBonusSub dw 0

ACtualObstaculo db 0

BanderaObstaculo dw 0

BanderaObstaculoSub dw 0

ActualTiempo dw 0

Hora db 0

Minuto db 0

Segundo db 0

Colorcito db 0

Registro dw 0

Rex db 0

Longitud db 0

Teclado db 25 dup (0)

ValX db 0

TopeX db 0

TopeY db 0

BarraAncho db 0

BarraLargo db 0

MaxBarraLargo db 0

MaxBarraAncho db 0

TituloOrden1 db "Seleccion el Tipo De Ordenamiento Que Desea Usar","$"

TituloOrden2 db "1)Ordenamiento Bubblesort","$"

TituloOrden3 db "2)Ordenamiento Quicksort","$"

TituloOrden4 db "3)Ordenamiento Shellsort","$"

TituloMovimiento1 db "Orientacion Del Ordenamiento","$"

TituloMovimiento2 db "1)Descendente","$"

TituloMovimiento3 db "2)Ascendente","$"

TipoOrden db 0

Orientacion db 0

TecladoTemporal db 10 dup (0)

;=============Almacenamiento Temporal===================

InformacionUsuario db 1100 dup (0)

InformacionTiempo db 200 dup (0)

InformacionPuntos db 200 dup (0)

InformacionNiveles db 500 dup (0);6 Niveles de 17

InformacionRealNiveles db 50 dup (0);6 Niveles de 17

ValoresBarras db 20 dup(0)

CantidadBarras db 0

;=============Variables De La PantallaJuego==============

PantallaUsuarioActual db 12 dup (0)

PantallaNivelActual db 0

PantallaPuntosActual db 4 dup(0)

PantallaTiempoActual db 8 dup(0)

TiempoNivel dw 0

ChoqueObstaculo db 0

ChoqueBonus db 0

Regis16 dw 0

Regis8 db 0

IndexNivel dw 0

;===============Delay===============

inputi db 50

mm dw 0

rr dw 0

;=================================================

 ArregloTemporal dw 20 dup(0)

 LetreroFinal db 70 dup(0)

;-----------------------------------------

RegistroSort1   dw  ?

RegistroSort2   dw  ?

RegistroSort3   dw  0

RegistroSort4   dw  19

RegistroSort5   dw  ?

RegistroSort6   dw  ?

;=========================================

gap db 0

j db 0

k db 0

n db 19

.code

main  proc

xor ax,ax

mov   ax, @data ;hmm ¿seg?

mov   ds,ax  ;ds = ax = saludo

;xor ah,ah

;int 16h

tip:

;ArchivoCargarUsuario

;CargarPunteo

;CargarTiempo

;GraficarBarras

;Call CargaValores;Cargar Al Temporal

;EstablecerTiempos

;call BurbujaAscendente

;call BurbujaDescendente

;call SSAscendente

;call SSDescendente

;mov RegistroSort3,0

;mov RegistroSort4,19

;call QSAscendente

;mov RegistroSort3,0

;mov RegistroSort4,19

;call QSDescendente

;jmp Exec

ssa:

mov BanderaCargado,0

LabelPrincipal:

Print Titulo1

Print Titulo2

Print Titulo3

NuevaLinea

Print MenuPrincipal1

NuevaLinea

Print MenuPrincipal2

NuevaLinea

Print MenuPrincipal3

NuevaLinea

mov ah,1

int 21h

cmp al,'1'

jz LabelIngresar

cmp al,'2'

jz LabelRegistrar

cmp al,'3'

jz Exec

jmp LabelPrincipal

LabelIngresar:

NuevaLinea

ArchivoCargarUsuario

LoginUsuario

cmp TecladoTemporal[0],'a'

jnz LabelMenuUsuario

cmp TecladoTemporal[1],'d'

jnz LabelMenuUsuario

cmp TecladoTemporal[2],'m'

jnz LabelMenuUsuario

cmp TecladoTemporal[3],'i'

jnz LabelMenuUsuario

cmp TecladoTemporal[4],'n'

jnz LabelMenuUsuario

jmp LabelMenuAdmin

LabelRegistrar:

NuevaLinea

ArchivoCargarUsuario

CrearUsuario

jmp LabelPrincipal

;==============================================Operaciones Usuario

LabelMenuUsuario:

NuevaLinea

Print Titulo1

Print Titulo2

Print Titulo3

NuevaLinea

Print MenuUsuario1

NuevaLinea

Print MenuUsuario2

NuevaLinea

Print MenuUsuario3

NuevaLinea

mov ah,1

int 21h

cmp al,'1'

jz LabelJuego

cmp al,'2'

jz LabelCargaJuego

cmp al,'3'

jz LabelPrincipal

jmp LabelMenuUsuario

LabelJuego:

cmp BanderaCargado,0

jz NoJugar

Juego

jmp LabelMenuUsuario

NoJugar:

NuevaLinea

Print SinCarga

mov ah,1

int 21h

NuevaLinea

jmp LabelMenuUsuario

LabelCargaJuego:

CargarNivel

jmp LabelMenuUsuario

;===================================================Operaciones Admin

LabelMenuAdmin:

NuevaLinea

Print Titulo1

Print Titulo2

Print Titulo3

NuevaLinea

Print MenuAdmin1

NuevaLinea

Print MenuAdmin2

NuevaLinea

Print MenuAdmin3

NuevaLinea

mov ah,1

int 21h

cmp al,'1'

jz LabelTopPuntos

cmp al,'2'

jz LabelTopTiempos

cmp al,'3'

jz LabelPrincipal

jmp LabelMenuAdmin

LabelTopTiempos:

NuevaLinea

TopTiempo

jmp LabelMenuAdmin

LabelTopPuntos:

NuevaLinea

TopPunteo

jmp LabelMenuAdmin

;cmp al, 'a'

;jne a

Exec:

NuevaLinea

.exit

main  endp  ;Termina proceso

QSAscendente proc

xor ax,ax

xor bx,bx

xor cx,cx

xor dx,dx

xor si,si

mov  ax,RegistroSort3

cmp  ax,RegistroSort4

jnl  mayor1

call NuevoPivoteAscendente

mov RegistroSort5, ax

add ax,1

mov Regis8,0

Push Regis16

push ax

push RegistroSort4

mov ax,RegistroSort5

mov RegistroSort4,ax

dec RegistroSort4

call QSAscendente

pop Regis16

Push Regis16

pop RegistroSort4

pop RegistroSort3

call QSAscendente

pop Regis16

mayor1:

ret

call DesCargaValores

DibujarBarras

QSAscendente endp

NuevoPivoteAscendente proc

lea  si,ArregloTemporal

mov  ax,RegistroSort4

shl  ax,1 ;Desplazar

add  si,ax

mov  ax,[si]

mov  RegistroSort6,ax

mov  ax,RegistroSort3

mov  RegistroSort1,ax

dec  RegistroSort1

mov  ax,RegistroSort3

mov  RegistroSort2,ax

;===================Inicio Del Ciclo

CicloPara:

lea  si,ArregloTemporal

mov  ax,RegistroSort2

shl  ax,1

add  si,ax

mov  ax,[si]

cmp  ax,RegistroSort6

jnl   mayor;Condicional De Cambio

jz mayor

inc  RegistroSort1

lea  di,ArregloTemporal

mov  cx,RegistroSort1

shl  cx,1

add  di,cx

mov  cx,[di]

mov  [di], ax

mov  [si], cx

cmp ax,0

jz silencio1

cmp cx,0

jz silencio1

push ax

push bx

push cx

push dx

push si

xor ax,ax

mov ax,[si]

mov Regis16,ax

SonidoBarra Regis16

mov ax,[di]

mov Regis16,ax

SonidoBarra Regis16

pop si

pop dx

pop cx

pop bx

pop ax

push ax

push bx

push cx

push dx

push si

call DesCargaValores

push rr

push mm

QSD3:

dec rr

jz QSD1

QSD2:

dec mm

jnz QSD2

jmp QSD3

QSD1:

pop mm

pop rr

DibujarBarras

pop dx

pop cx

pop bx

pop ax

pop si

silencio1:

mayor:

inc  RegistroSort2

mov  ax,RegistroSort4

cmp  RegistroSort2,ax

jl   CicloPara

inc  RegistroSort1

lea  si,ArregloTemporal

mov  ax, RegistroSort1

shl  ax, 1

push cx

add  si, ax

mov  ax, [si]

lea  di,ArregloTemporal

mov  cx, RegistroSort4

shl  cx, 1

add  di, cx

mov  cx, [di]

mov  [di],ax;Cambio Pos entre di y si

mov  [si],cx

mov  ax, RegistroSort1;Retorno Del Siguiente

pop cx

PivoFin:

ret

NuevoPivoteAscendente endp

DesCargaValores proc

mov cx,20

xor si,si

LupDesCargaValores:

xor ah,ah

mov bx,si

shl bx,1

mov ax,ArregloTemporal[bx]

mov ValoresBarras[si],al

inc si

dec cx

jnz LupDesCargaValores

ret

DesCargaValores endp

CargaValores proc

mov cx,20

xor si,si

LupCargaValores:

xor ah,ah

mov al,ValoresBarras[si]

mov bx,si

shl bx,1

mov ArregloTemporal[bx],ax

inc si

dec cx

jnz LupCargaValores

ret

CargaValores endp

;==============================================Shell Sort

SSDescendente proc

DibujarBarras

mov gap,0

mov j,0

mov k,0

xor ax,ax

xor bx,bx

mov al,n

mov bl,2

div bl

mov gap,al

SSlup1:

mov al,gap

mov j,al

SSlup2:

xor cx,cx

mov cl,j

push cx

mov al,gap

sub j,al

jc SSPasar;Evitar Negativos

mov al,j

mov k,al

pop cx

mov j,cl

SSlup3:

xor bx,bx

xor si,si

xor ax,ax

mov al,k

add al,gap

mov si,ax

mov bl,k

mov al,ValoresBarras[si]

cmp al,ValoresBarras[bx]

jz SSPasar

jc SSPasar

mov al,ValoresBarras[si]

mov ah,ValoresBarras[bx]

mov ValoresBarras[si],ah

mov ValoresBarras[bx],al

cmp al,0

jz Silencio

cmp ah,0

jz Silencio

push ax

push bx

push cx

push dx

push si

xor ax,ax

mov al,ValoresBarras[si]

mov Regis16,ax

SonidoBarra Regis16

mov al,ValoresBarras[bx]

mov Regis16,ax

SonidoBarra Regis16

pop si

pop dx

pop cx

pop bx

pop ax

;==================Cambiar Posiciones

push rr

push mm

SSD3:

dec rr

jz SSD1

SSD2:

dec mm

jnz SSD2

jmp SSD3

SSD1:

pop mm

pop rr

DibujarBarras

Silencio:

mov al,gap

sub k,al;k=-=gap

jc SSPasar;k>0

cmp k,0

jnz SSlup3;k=0

SSPasar:

inc j;j++

mov al,n

cmp j,al

jc SSlup2;j<n

xor ah,ah

mov al,gap

mov bl,2

div bl

mov gap,al;gap=gap/2

cmp gap,0

jnz SSlup1

DibujarBarras

ret

SSDescendente endp

SSAscendente proc

DibujarBarras

mov gap,0

mov j,0

mov k,0

xor ax,ax

xor bx,bx

mov al,n

mov bl,2

div bl

mov gap,al

SSlup1:

mov al,gap

mov j,al

SSlup2:

xor cx,cx

mov cl,j

push cx

mov al,gap

sub j,al

jc SSPasar;Evitar Negativos

mov al,j

mov k,al

pop cx

mov j,cl

SSlup3:

xor bx,bx

xor si,si

xor ax,ax

mov al,k

add al,gap

mov si,ax

mov bl,k

mov al,ValoresBarras[si]

cmp al,ValoresBarras[bx]

jz SSPasar

jnc SSPasar

mov al,ValoresBarras[si]

mov ah,ValoresBarras[bx]

mov ValoresBarras[si],ah

mov ValoresBarras[bx],al

cmp al,0

jz Silencio

cmp ah,0

jz Silencio

push ax

push bx

push cx

push dx

push si

xor ax,ax

mov al,ValoresBarras[si]

mov Regis16,ax

SonidoBarra Regis16

mov al,ValoresBarras[bx]

mov Regis16,ax

SonidoBarra Regis16

pop si

pop dx

pop cx

pop bx

pop ax

;==================Cambiar Posiciones

push rr

push mm

SSD3:

dec rr

jz SSD1

SSD2:

dec mm

jnz SSD2

jmp SSD3

SSD1:

pop mm

pop rr

DibujarBarras

Silencio:

mov al,gap

sub k,al;k=-=gap

jc SSPasar;k>0

cmp k,0

jnz SSlup3;k=0

SSPasar:

inc j;j++

mov al,n

cmp j,al

jc SSlup2;j<n

xor ah,ah

mov al,gap

mov bl,2

div bl

mov gap,al;gap=gap/2

cmp gap,0

jnz SSlup1

DibujarBarras

ret

SSAscendente endp

;====================================Metodo Burbuja=========================

BurbujaDescendente proc

DibujarBarras

xor ax,ax

xor bx,bx

xor si,si

xor dx,dx

Blup1:

xor bx,bx

Blup2:

mov al,ValoresBarras[bx]

cmp al,ValoresBarras[bx+1]

jnc Bpasar

jz Bpasar

Bcambio:

mov ah,ValoresBarras[bx+1]

mov ValoresBarras[bx+1],al

mov ValoresBarras[bx],ah

cmp al,0

jz Silencio

cmp ah,0

jz Silencio

push ax

push bx

push cx

push dx

push si

xor ax,ax

mov al,ValoresBarras[bx+1]

mov Regis16,ax

SonidoBarra Regis16

mov al,ValoresBarras[bx]

mov Regis16,ax

SonidoBarra Regis16

pop si

pop dx

pop cx

pop bx

pop ax

push rr

push mm

QSD3:

dec rr

jz QSD1

QSD2:

dec mm

jnz QSD2

jmp QSD3

QSD1:

pop mm

pop rr

push si

push bx

DibujarBarras

pop bx

pop si

Silencio:

Bpasar:

inc bx

cmp bx,19

jnz Blup2

inc si

cmp si,20

jnz Blup1

DibujarBarras

ret

BurbujaDescendente endp

BurbujaAscendente proc

DibujarBarras

xor ax,ax

xor bx,bx

xor si,si

xor dx,dx

BAlup1:

xor bx,bx

BAlup2:

mov al,ValoresBarras[bx]

cmp al,ValoresBarras[bx+1]

jc BApasar

jz BApasar

BAcambio:

mov ah,ValoresBarras[bx+1]

mov ValoresBarras[bx+1],al

mov ValoresBarras[bx],ah

cmp al,0

jz Silencio

cmp ah,0

jz Silencio

push ax

push bx

push cx

push dx

push si

xor ax,ax

mov al,ValoresBarras[bx+1]

mov Regis16,ax

SonidoBarra Regis16

mov al,ValoresBarras[bx]

mov Regis16,ax

SonidoBarra Regis16

pop si

pop dx

pop cx

pop bx

pop ax

push rr

push mm

BAD3:

dec rr

jz BAD1

BAD2:

dec mm

jnz BAD2

jmp BAD3

BAD1:

pop mm

pop rr

push si

push bx

DibujarBarras

pop bx

pop si

Silencio:

BApasar:

inc bx

cmp bx,19

jc BAlup2

jnz BAlup2

inc si

cmp si,20

jnz BAlup1

DibujarBarras

ret

BurbujaAscendente endp

QSDescendente proc

xor ax,ax

xor bx,bx

xor cx,cx

xor dx,dx

xor si,si

mov  ax,RegistroSort3

cmp  ax,RegistroSort4

jnl  mayor1

call NuevoPivoteDescendente

mov RegistroSort5, ax

add ax,1

mov Regis8,0

Push Regis16

push ax

push RegistroSort4

mov ax,RegistroSort5

mov RegistroSort4,ax

dec RegistroSort4

call QSDescendente

pop Regis16

Push Regis16

pop RegistroSort4

pop RegistroSort3

call QSDescendente

pop Regis16

mayor1:

call DesCargaValores

DibujarBarras

ret

QSDescendente endp

NuevoPivoteDescendente proc

lea  si,ArregloTemporal

mov  ax,RegistroSort4

shl  ax,1 ;Desplazar

add  si,ax

mov  ax,[si]

mov  RegistroSort6,ax

mov  ax,RegistroSort3

mov  RegistroSort1,ax

dec  RegistroSort1

mov  ax,RegistroSort3

mov  RegistroSort2,ax

;===================Inicio Del Ciclo

CicloPara:

lea  si,ArregloTemporal

mov  ax,RegistroSort2

shl  ax,1

add  si,ax

mov  ax,[si]

cmp  ax,RegistroSort6

jl   mayor;Condicional De Cambio

jz mayor

inc  RegistroSort1

lea  di,ArregloTemporal

mov  cx,RegistroSort1

shl  cx,1

add  di,cx

mov  cx,[di]

mov  [di], ax

mov  [si], cx

cmp ax,0

jz silencio1

cmp cx,0

jz silencio1

push ax

push bx

push cx

push dx

push si

xor ax,ax

mov ax,[si]

mov Regis16,ax

SonidoBarra Regis16

mov ax,[di]

mov Regis16,ax

SonidoBarra Regis16

pop si

pop dx

pop cx

pop bx

pop ax

push ax

push bx

push cx

push dx

push si

call DesCargaValores

push rr

push mm

QSD3A:

dec rr

jz QSD1A

QSD2A:

dec mm

jnz QSD2A

jmp QSD3A

QSD1A:

pop mm

pop rr

DibujarBarras

pop si

pop dx

pop cx

pop bx

pop ax

silencio1:

mayor:

inc  RegistroSort2

mov  ax,RegistroSort4

cmp  RegistroSort2,ax

jl   CicloPara

inc  RegistroSort1

lea  si,ArregloTemporal

mov  ax, RegistroSort1

shl  ax, 1

push cx

add  si, ax

mov  ax, [si]

lea  di,ArregloTemporal

mov  cx, RegistroSort4

shl  cx, 1

add  di, cx

mov  cx, [di]

mov  [di],ax;Cambio Pos entre di y si

mov  [si],cx

mov  ax, RegistroSort1;Retorno Del Siguiente

pop cx

PivoFin:

ret

NuevoPivoteDescendente endp

end main

Código Traducido Para Los Algoritmos De Ordenamiento

**QUICKSORT**

**Fuente:** <https://www.geeksforgeeks.org/quick-sort/>

/\* This function takes last element as pivot, places

   the pivot element at its correct position in sorted

    array, and places all smaller (smaller than pivot)

   to left of pivot and all greater elements to right

   of pivot \*/

int partition (int arr[], int low, int high)

{

    int pivot = arr[high];    // pivot

    int i = (low - 1);  // Index of smaller element

    for (int j = low; j <= high- 1; j++)

    {

        // If current element is smaller than the pivot

        if (arr[j] < pivot)

        {

            i++;    // increment index of smaller element

            swap(&arr[i], &arr[j]);

        }

    }

    swap(&arr[i + 1], &arr[high]);

    return (i + 1);

}

/\* The main function that implements QuickSort

 arr[] --> Array to be sorted,

  low  --> Starting index,

  high  --> Ending index \*/

void quickSort(int arr[], int low, int high)

{

    if (low < high)

    {

        /\* pi is partitioning index, arr[p] is now

           at right place \*/

        int pi = partition(arr, low, high);

        // Separately sort elements before

        // partition and after partition

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

**SHELLSORT**

**Fuente:** <https://www.geeksforgeeks.org/shellsort/>

/\* function to sort arr using shellSort \*/

int shellSort(int arr[], int n)

{

    // Start with a big gap, then reduce the gap

    for (int gap = n/2; gap > 0; gap /= 2)

    {

        // Do a gapped insertion sort for this gap size.

        // The first gap elements a[0..gap-1] are already in gapped order

        // keep adding one more element until the entire array is

        // gap sorted

        for (int i = gap; i < n; i += 1)

        {

            // add a[i] to the elements that have been gap sorted

            // save a[i] in temp and make a hole at position i

            int temp = arr[i];

            // shift earlier gap-sorted elements up until the correct

            // location for a[i] is found

            int j;

            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)

                arr[j] = arr[j - gap];

            //  put temp (the original a[i]) in its correct location

            arr[j] = temp;

        }

    }

    return 0;

}