

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

PROGRAMACIÓN

FACULTAD DE INGENIERÍA

Práctica 6: Programas

Integrantes:

Rodríguez Ruíz Stefanny,

Mendoza Frías Luis

Fernando, Martínez Barras

Alexis, Mandujano Jiménez

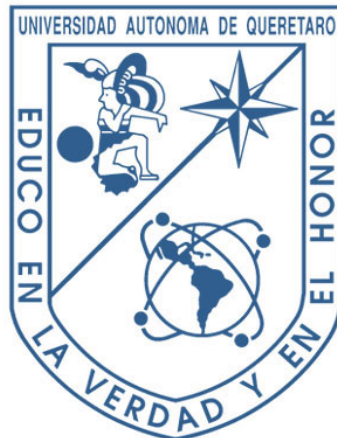
Daniel Cruz

Ingeniería en Nanotecnología

Profesor: José de Jesús

Santana Ramírez

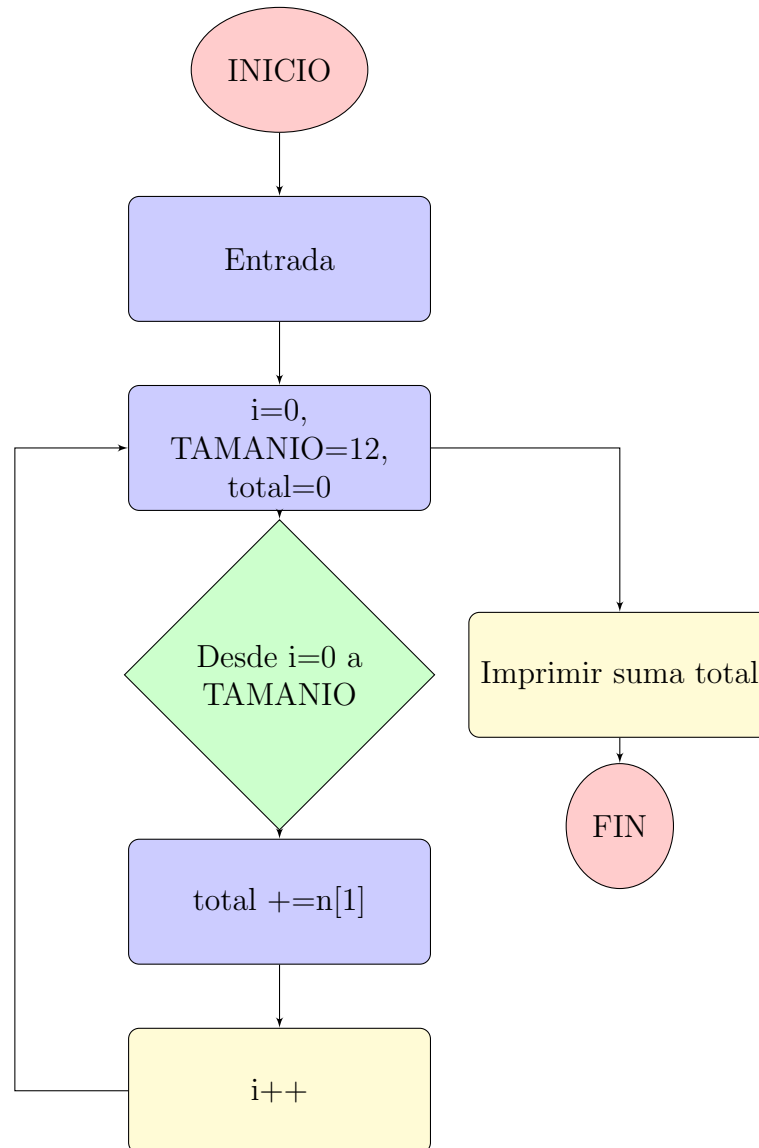
Abril 11, 2019



1 Suma de los elementos de un arreglo

Este programa se centra en sumar los elementos que se encuentran en un arreglo, un arreglo es una serie de números ordenados, que se guardan en una variable previamente definida. El propósito principal es el concepto del arreglo.

1.1 Diagrama de flujo



1.2 Código

```
//suma de los elementos de un arreglo  
#include <stdio.h>  
#define TAMANIO 12  
int main()  
{
```

```

int n[ TAMANIO ]={50,87,90,60,1,2,3,4,5,10,9,12};
int i;
int total=0;

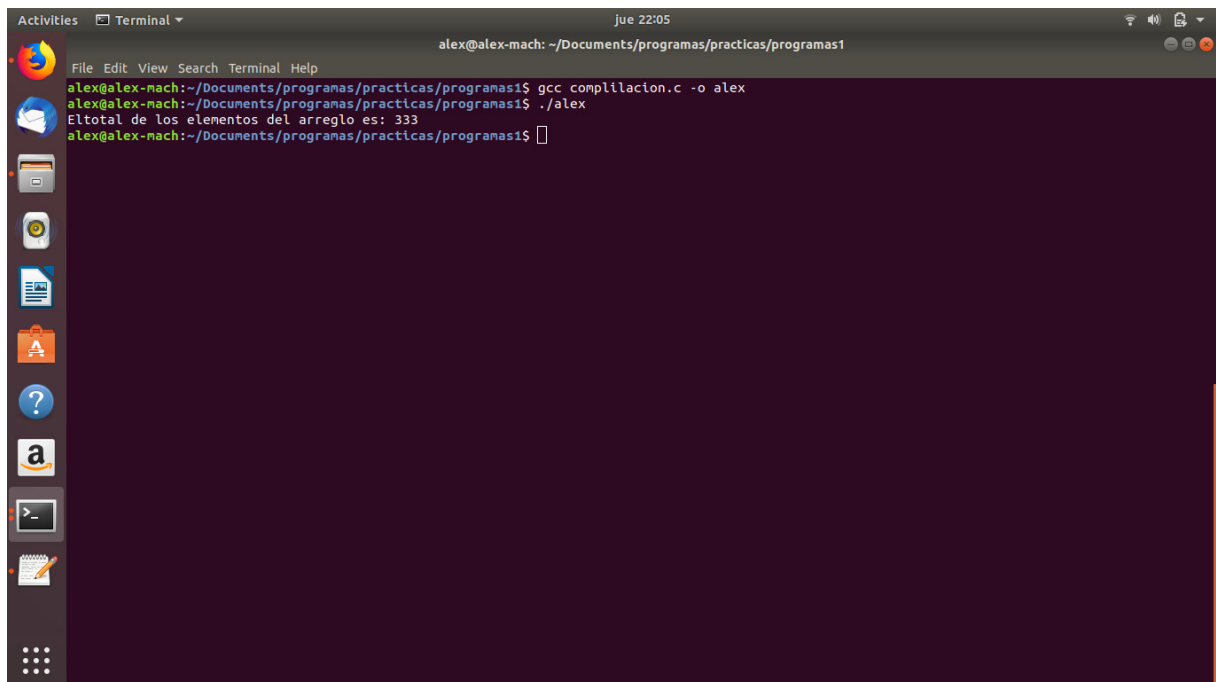
for(i=0;i<TAMANIO;i++)
{
total += n[i];
}

printf("El total de los elementos del arreglo es: %d\n",total);

return 0;
}

```

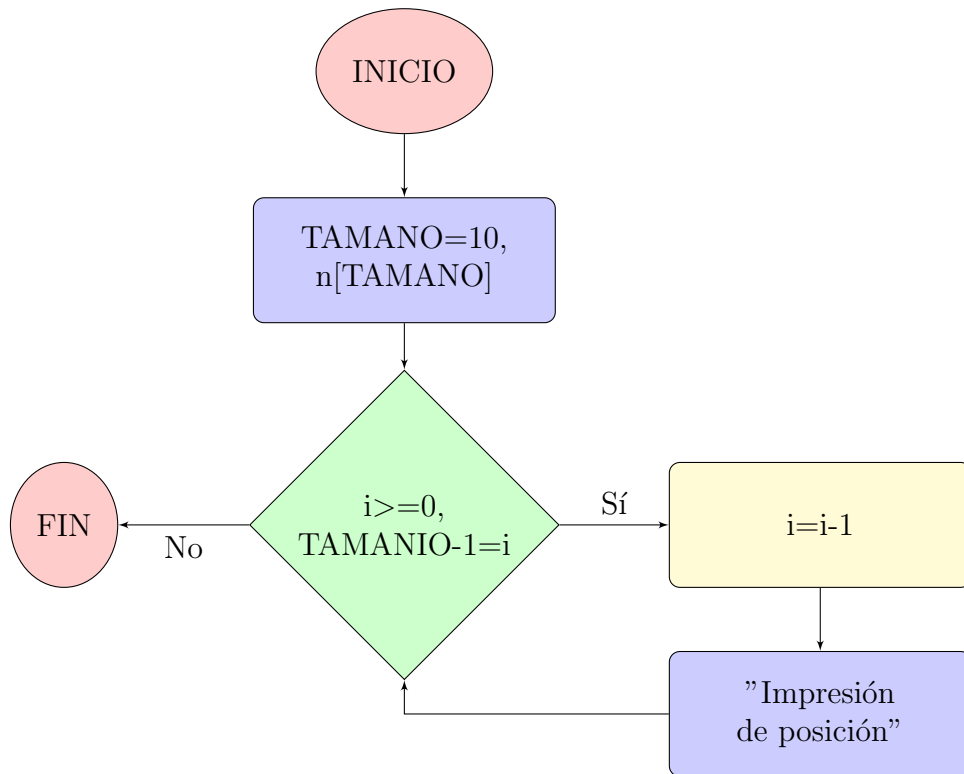
1.3 Terminal



2 Impresión de un arreglo

Previamente se definió el concepto de un arreglo, la diferencia principal se encuentra en el sentido de que no se le proporciona más ordenes al programa que la de imprimir los elemento de dicho arreglo.

2.1 Diagrama de flujo

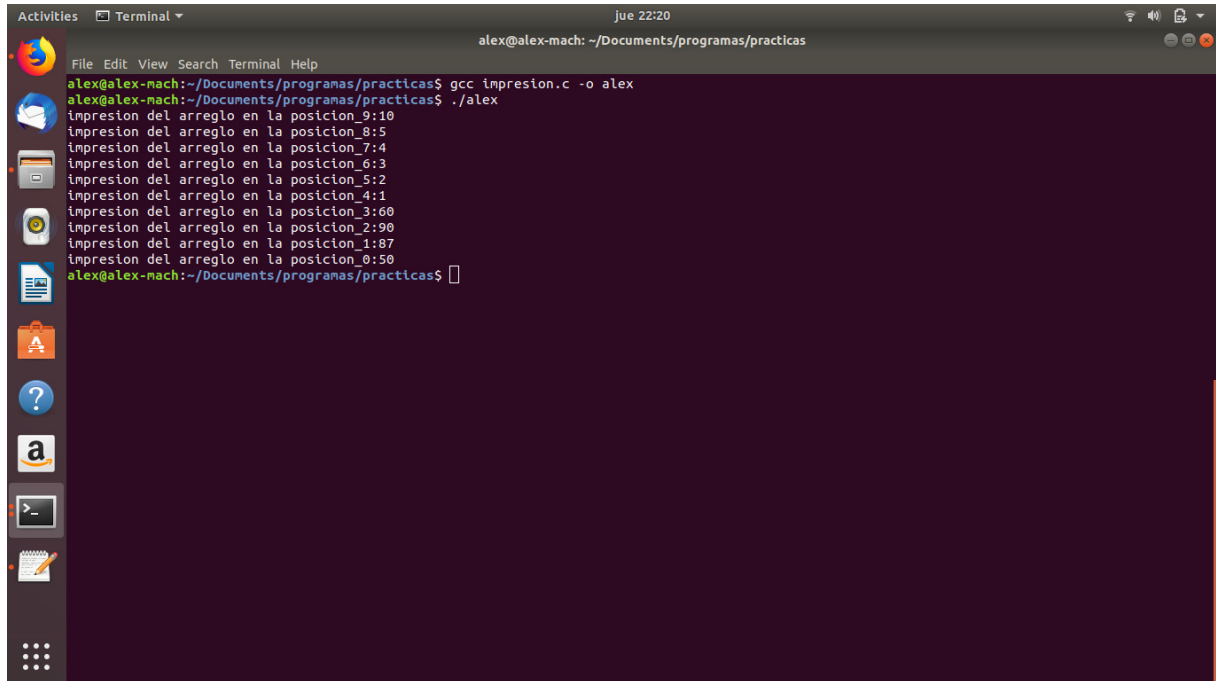


2.2 Código

```
//impresion de un arreglo
#include<stdio.h>
#define TAMANO 10
int main()
{
    int i, n[TAMANO]={50,87,90,60,1,2,3,4,5,10};

    for(i=0;i<TAMANO;i++)
    {printf("impresion del arreglo en la posicion_%d:%d\n",i,n[i])
      ;}
    return 0;
}
```

2.3 Terminal

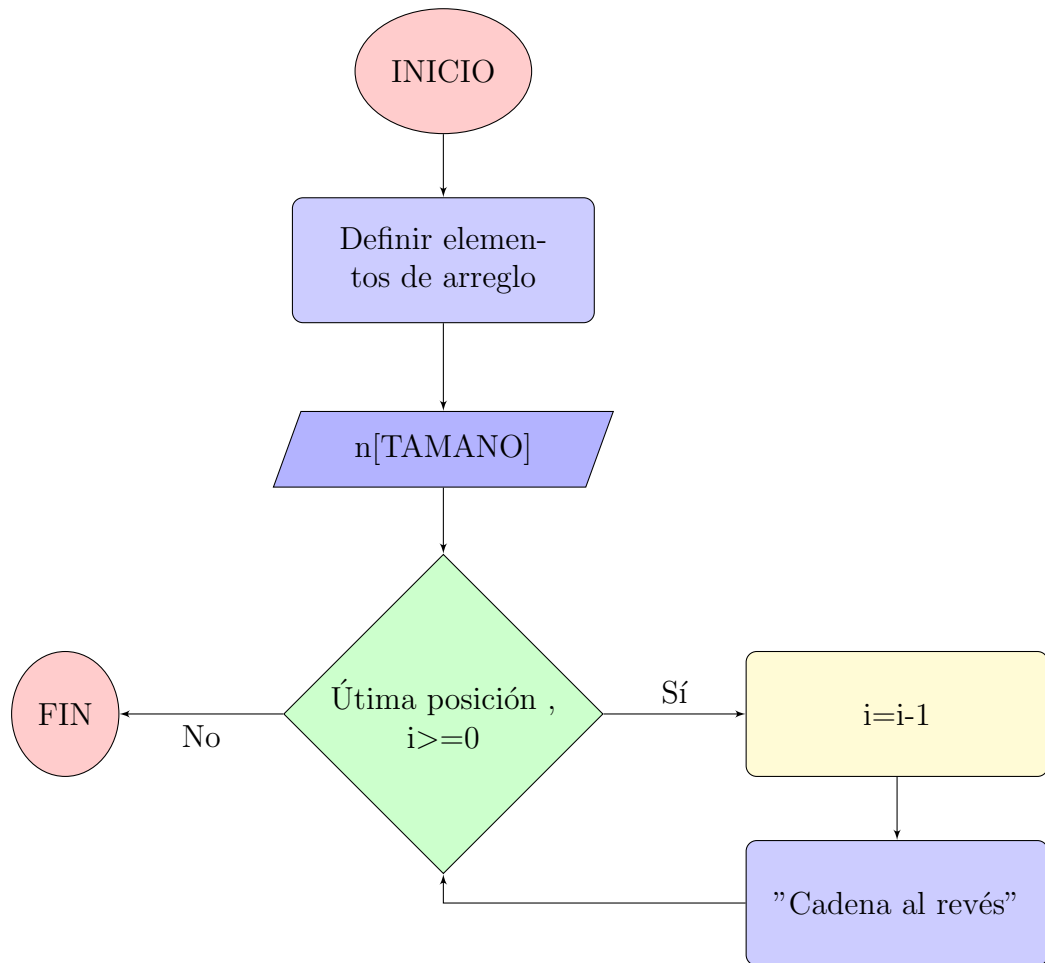


```
alex@alex-mach: ~/Documents/programas/practicas
alex@alex-mach:~/Documents/programas/practicas$ gcc impresion.c -o alex
alex@alex-mach:~/Documents/programas/practicas$ ./alex
impresion del arreglo en la posicion 9:10
impresion del arreglo en la posicion 8:5
impresion del arreglo en la posicion 7:4
impresion del arreglo en la posicion 6:3
impresion del arreglo en la posicion 5:2
impresion del arreglo en la posicion 4:1
impresion del arreglo en la posicion 3:60
impresion del arreglo en la posicion 2:90
impresion del arreglo en la posicion 1:87
impresion del arreglo en la posicion 0:50
alex@alex-mach:~/Documents/programas/practicas$
```

3 Impresión inversa de un arreglo

El programa se centra en la impresión del arreglo inverso, esto se logra a través modificaciones pequeñas al programa "impresión de un arreglo", para que en lugar de que el contador vaya hacia adelante, este retroceda y posteriormente imprima el resultado.

3.1 Diagrama de flujo

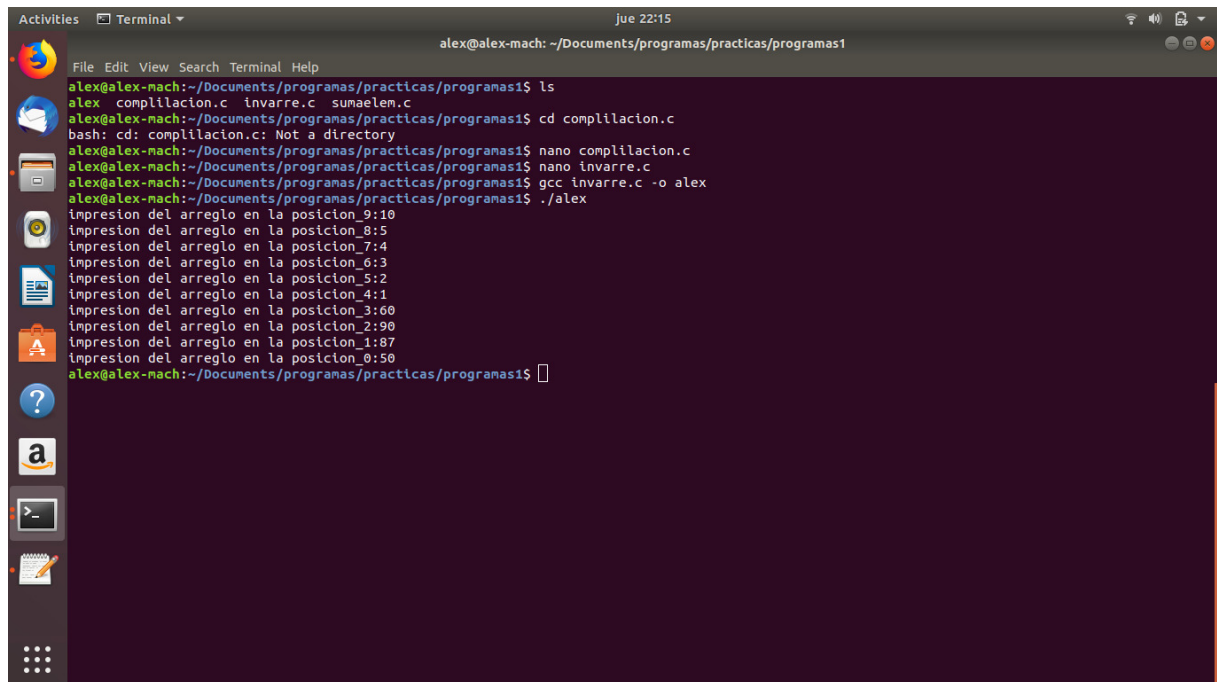


3.2 Código

```
//impresion inversa de un arreglo
#include<stdio.h>
#define TAMANO 10
int main()
{
    int i, n[TAMANO]={50,87,90,60,1,2,3,4,5,10};

    for(i=TAMANO-1;i>=0;i--)
    {printf("impresion del arreglo en la posicion_%d:%d\n",i,n[i])
      ;}
    return 0;
}
```

3.3 Terminal

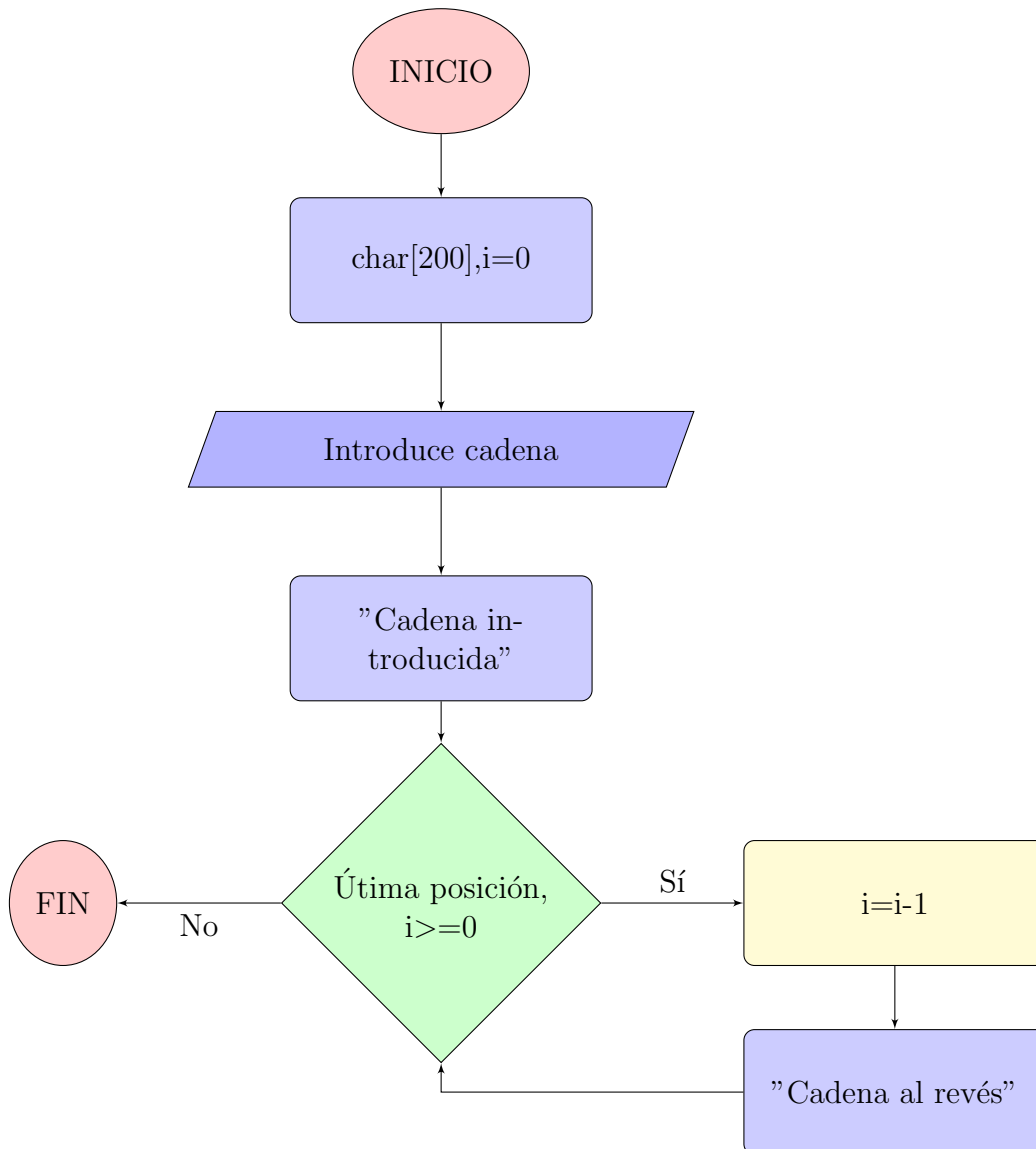


```
alex@alex-mach: ~/Documents/programas/practicas/programas1
alex@alex-mach:~/Documents/programas/practicas/programas1$ ls
alex  complilacion.c  invarre.c  sumaelm.c
alex@alex-mach:~/Documents/programas/practicas/programas1$ cd complilacion.c
bash: cd: complilacion.c: Not a directory
alex@alex-mach:~/Documents/programas/practicas/programas1$ nano complilacion.c
alex@alex-mach:~/Documents/programas/practicas/programas1$ nano invarre.c
alex@alex-mach:~/Documents/programas/practicas/programas1$ gcc invarre.c -o alex
alex@alex-mach:~/Documents/programas/practicas/programas1$ ./alex
impresion del arreglo en la posicion_9:10
impresion del arreglo en la posicion_8:5
impresion del arreglo en la posicion_7:4
impresion del arreglo en la posicion_6:3
impresion del arreglo en la posicion_5:2
impresion del arreglo en la posicion_4:1
impresion del arreglo en la posicion_3:60
impresion del arreglo en la posicion_2:90
impresion del arreglo en la posicion_1:87
impresion del arreglo en la posicion_0:50
alex@alex-mach:~/Documents/programas/practicas/programas1$
```

4 Impresión inversa de una cadena

La cadena podría definirse como un vector donde podemos guardar valores, el propósito principal de hacer esto es generar un espacio, donde el usuario puede introducir los valores y estos no necesariamente tienen que ser definidos desde el código fuente; no obstante la mecánica de funcionamiento es muy parecida al de "impresión inverso de un arreglo".

4.1 Diagrama de flujo



4.2 Código

```
#include <stdio.h>
#include <string.h>

int main()
{
    char cadena[200];
    int i = 0;
    printf("Introduce una cadena de texto: \n");
    scanf("%s", cadena);
    printf("\nLa cadena introducida es: %s\n\n", cadena);
    printf("La cadena al revés es:\n\t");
    for (i=strlen(cadena); i>=0; i--){
```



```

        printf("%c", cadena[i]);
    }
    puts(" ");
    return 0;
}

```

4.3 Terminal

```

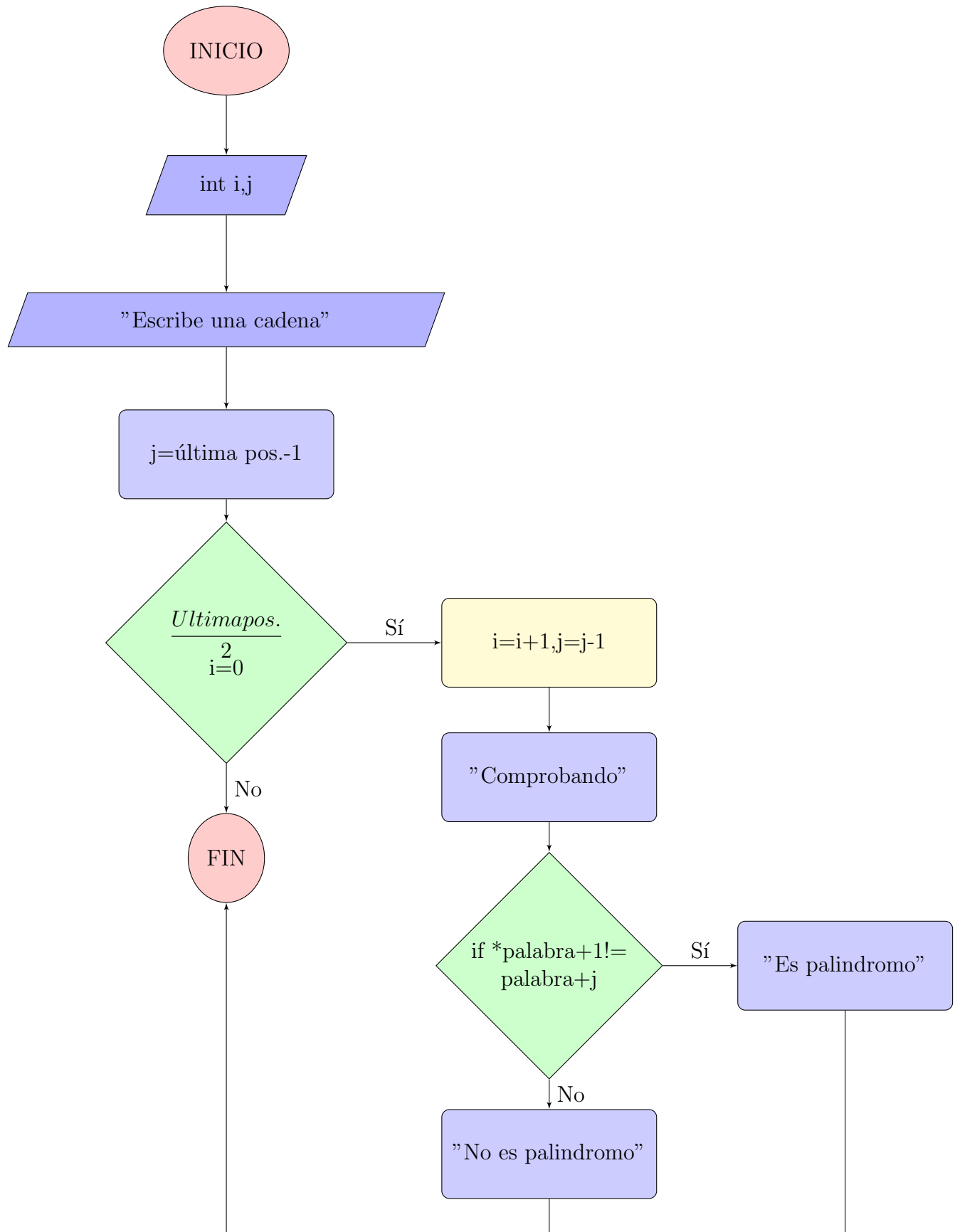
alex@alex-mach: ~/Documents/programas/practicas
File Edit View Search Terminal Help
alex@alex-mach:~/Documents/programas/practicas$ ls
alex  impresion.c  invcad.c  palindromo.c  programas1  suma  suma1.c  suma.c
alex@alex-mach:~/Documents/programas/practicas$ gcc invcad.c -o alex
alex@alex-mach:~/Documents/programas/practicas$ ./alex
Introduce una cadena de texto:
programacion
La cadena introducida es: programacion
La cadena al reves es:
noicamargorp
alex@alex-mach:~/Documents/programas/practicas$

```

5 Verificar si una cadena es un palíndromo

Un palíndromo se define como una palabra que se puede leer de izquierda a derecha o viceversa sin afectar el orden de las palabras, un ejemplo de ello es el nombre ANA, nótese que si se invierte el orden de lectura (de derecha a izquierda) la palabra conserva coherencia. El programa que se muestra enseguida tiene la función de verificar si las letras de una palabra conservan el mismo orden, si esta se invierte.

5.1 Diagrama de flujo



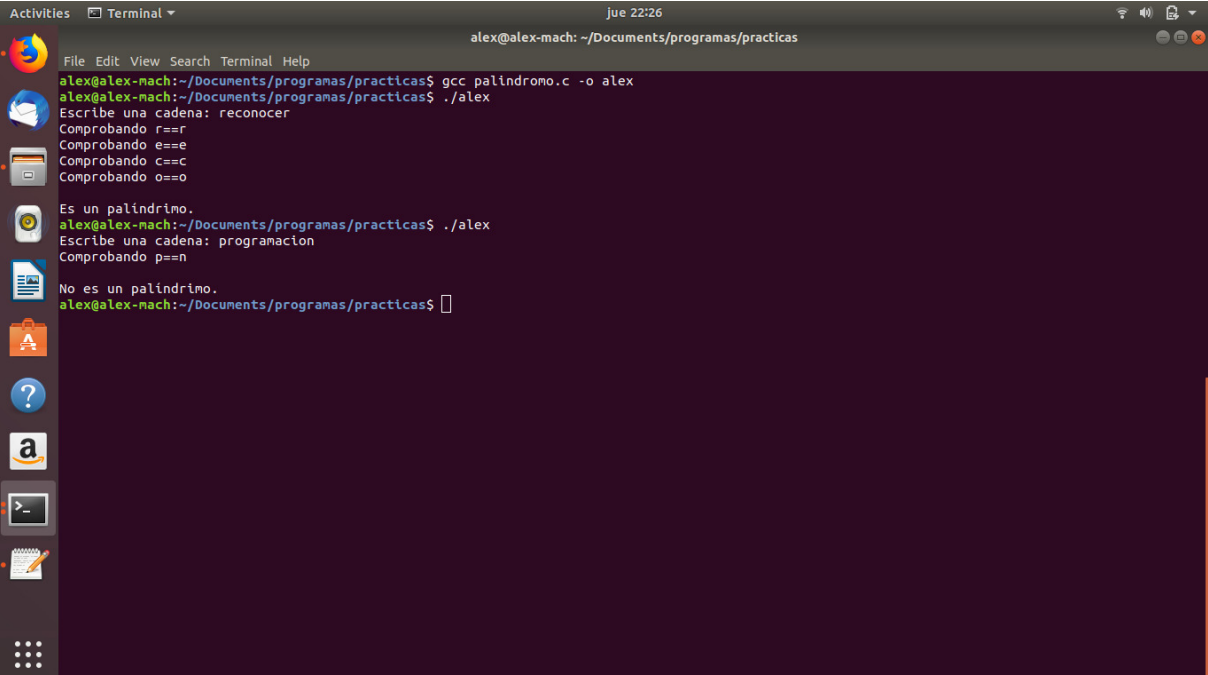
5.2 Código

```
#include <stdio.h>
#include <string.h>

int main()
{
    char palabra[20];
    int i, j;
    int palindromo = 1;
    printf("Escribe una cadena: ");
    scanf("%s", palabra);
    j=strlen(palabra)-1;
    for(i=0; i<strlen(palabra)/2; i++, j--) {
        printf("Comprobando %c==%c\n", *(palabra+i), *(palabra+
j));
        if (*(palabra+i) != *(palabra+j)) {
            palindromo = 0;
            break;
        }
    }
    if (palindromo)
        printf("\nEs un pal ndrismo.\n");
    else
        printf("\nNo es un pal ndrismo.\n");

    return (0);
}
```

5.3 Terminal



The screenshot shows a terminal window titled "alex@alex-mach: ~/Documents/programas/practicas". The user has compiled a C program named "alex" using "gcc palindromo.c -o alex". They then run the program twice. In the first run, they input "reconocer", and the program outputs "Es un palndrimo." (Note the typo in the original image). In the second run, they input "programacion", and the program outputs "No es un palndrimo." (Note the typo in the original image). The terminal window includes a sidebar with application icons and a top bar with system status.

```
alex@alex-mach:~/Documents/programas/practicas$ gcc palindromo.c -o alex
alex@alex-mach:~/Documents/programas/practicas$ ./alex
Escribe una cadena: reconocer
Comprobando r==r
Comprobando e==e
Comprobando c==c
Comprobando o==o

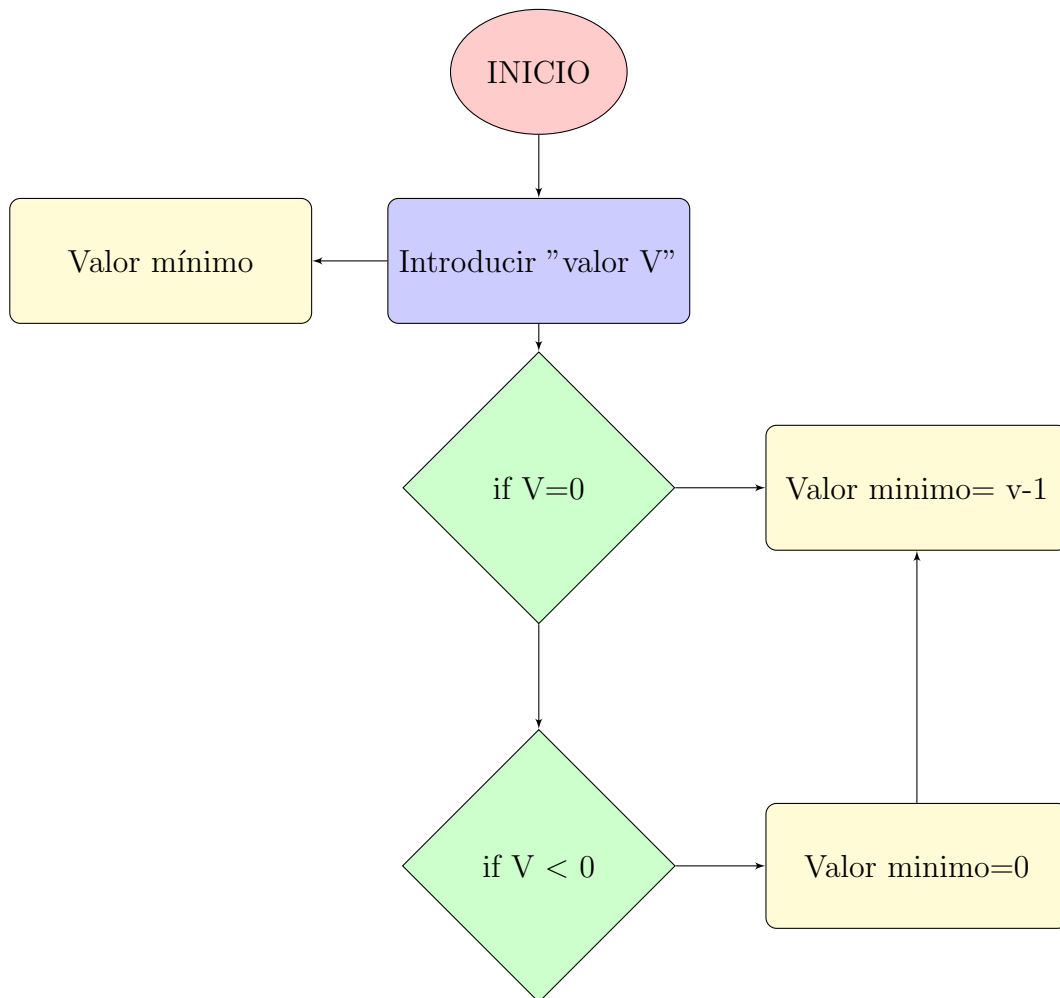
Es un palndrimo.
alex@alex-mach:~/Documents/programas/practicas$ ./alex
Escribe una cadena: programacion
Comprobando p==n

No es un palndrimo.
alex@alex-mach:~/Documents/programas/practicas$
```

6 El valor mínimo de un arreglo

Para calcular el máximo o el mínimo de un conjunto de valores, utilizaremos una variable del mismo tipo que el conjunto de valores que inicializamos con el valor del primer elemento y que modificamos cada vez que un elemento del conjunto supera el valor almacenado.

6.1 Diagrama de flujo



6.2 Código

```
#include <stdio.h>
#include <stdlib.h>
int minimo(int V[], int n)
{
    int min = V[0];
    if(n == 0) return min;
    else
    {
        if(V[n] < min) min = V[n];
    }
}
```

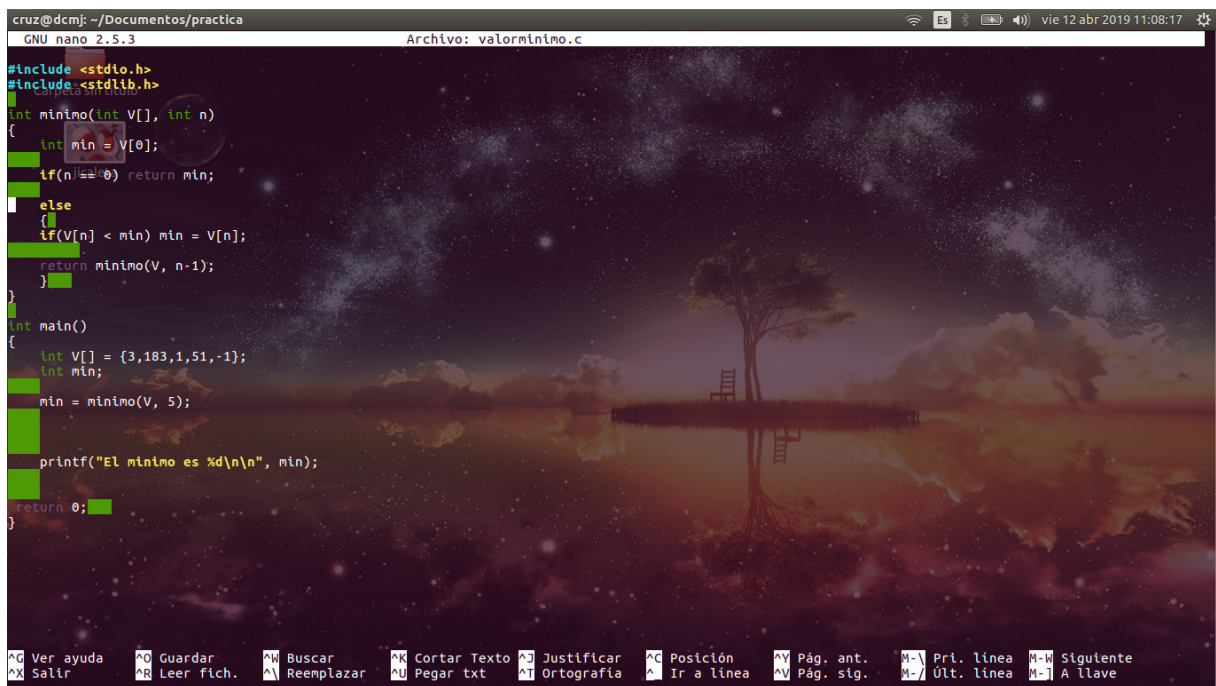
```

        return minimo(V, n-1);
    }
}

int main()
{
    int V[] = {3,183,1,51,-1};
    int min;
    min = minimo(V, 5);
    printf("El minimo es %d\n\n", min);
    return 0;
}

```

6.3 Terminal



```

cruz@dcml: ~/Documentos/practica
GNU nano 2.5.3 Archivo: valorminimo.c

#include <stdio.h>
#include <stdlib.h>

int minimo(int V[], int n)
{
    int min = V[0];
    if(n==0) return min;
    else
    {
        if(V[n] < min) min = V[n];
        return minimo(V, n-1);
    }
}

int main()
{
    int V[] = {3,183,1,51,-1};
    int min;
    min = minimo(V, 5);
    printf("El minimo es %d\n\n", min);
    return 0;
}

```

Ver ayuda Guardar Buscar Cortar Texto Justificar Posición Pág. ant. Pri. línea Siguiete
Salir Leer fich. Reemplazar Pegar txt Ortografía Ir a línea Pág. sig. Últ. línea A llave

7 Ordenamiento por selección

El método de ordenación por selección consiste en repetir los siguientes pasos:

- Se busca el elemento más pequeño del array y se coloca en la primera posición.
 - Entre los restantes, se busca el elemento más pequeño y se coloca en la segunda posición.
 - Entre los restantes se busca el elemento más pequeño y se coloca en la tercera posición.
- Este proceso se repite hasta colocar el último elemento.

7.1 Diagrama de flujo

7.2 Código

```

#include <stdio.h>
int main()

```

```

{
    int array[100], n, c, d, position, swap;
    printf("Introducir el numero de elementos: ");
    scanf("%d", &n);
    printf("Introduce %d enteros \n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
    for (c = 0; c < (n - 1); c++)
    {
        position = c;
        for (d = c + 1; d < n; d++)
        {
            if (array[position] > array[d])
                position = d;
        }
        if (position != c)
        {
            swap = array[c];
            array[c] = array[position];
            array[position] = swap;
        }
    }
    printf("El valor minimo de un arreglo es: ");
    for (c = 0; c < n; c++)
        printf("%d\n", array[c]);
    return 0;
}

```

7.3 Terminal

```

cruz@dcml: ~/Documentos/practica
GNU nano 2.5.3 Archivo: ordenamiento.c

#include <stdio.h>
int main()
{
    int array[100], n, c, d, position, swap;
    printf("Introducir el numero de elementos \n");
    scanf("%d", &n);
    printf("Introduce %d enteros\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
    for (c = 0; c < (n - 1); c++)
    {
        position = c;
        for (d = c + 1; d < n; d++)
        {
            if (array[position] > array[d])
                position = d;
        }
        if (position != c)
        {
            swap = array[c];
            array[c] = array[position];
            array[position] = swap;
        }
    }
    printf("el valor minimo de un arreglo es:\n");
    for (c = 0; c < n; c++)
        printf("%d\n", array[c]);
    return 0;
}

```

38 líneas leídas

Ver ayuda Guardar Buscar Cortar Texto Justificar Posición Pág. ant. Pri. línea Siguiente
Salir Leer fich. Reemplazar Pegar txt Ortografía Ir a línea Pág. sig. Últ. línea A llave

8 Quicksort

Se basa en la técnica divide y vencerás, que consiste en ir subdividiendo el array en arrays más pequeños, y ordenar éstos. Para hacer esta división, se toma un valor del array como pivote, y se mueven todos los elementos menores que este pivote a su izquierda, y los mayores a su derecha. A continuación se aplica el mismo método a cada una de las dos partes en las que queda dividido el array.

8.1 Diagrama de flujo

8.2 Código

```
#include<stdio.h>
void quicksort(int number[25],int first,int last){
    int i, j, pivot, temp;
    if(first<last){
        pivot=first;
        i=first;
        j=last;
        while(i<j){
            while(number[i]<=number[pivot]&& i<last)
                i++;
            while(number[j]>number[pivot])
                j--;
            if(i<j){
                temp=number[i];
                number[i]=number[j];
                number[j]=temp;
            }
        }
        temp=number[pivot];
        number[pivot]=number[j];
        number[j]=temp;
        quicksort(number,first,j-1);
        quicksort(number,j+1,last);
    }
}

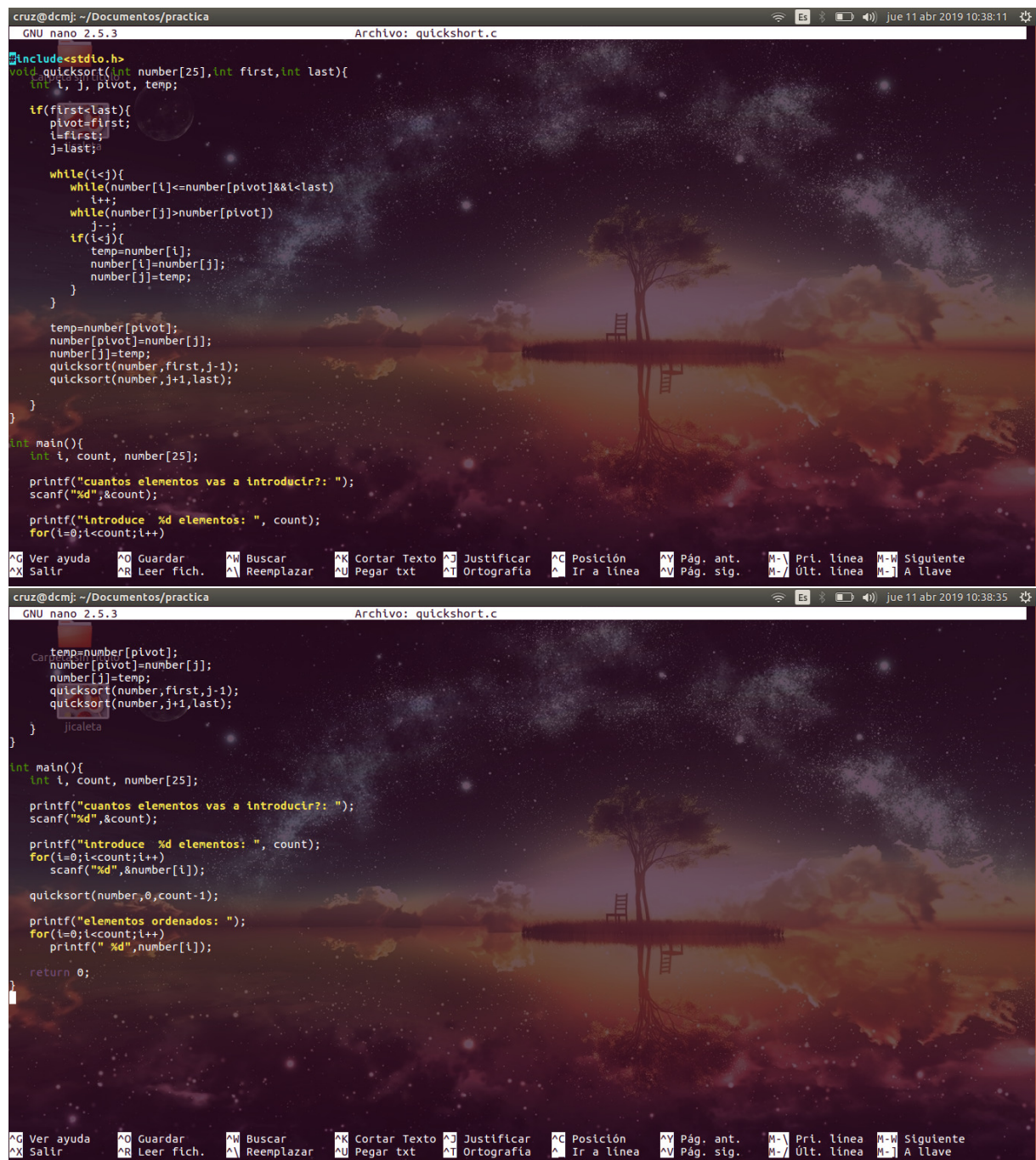
int main(){
    int i, count, temp, pivot, j, first, last, number[25];
    printf("Cuantos elementos vas a introducir?: ");
    scanf("%d",&count);
    printf("Introduce %d elementos: ", count);
    for(i=0;i<count;i++)
        temp=number[pivot];
        number[pivot]=number[j];
        number[j]=temp;
        quicksort(number,first,j-1);
        quicksort(number,j+1,last);
}
```

```

    }
int mo(){
    int i, count, number[25];
    printf("Cuantos elementos vas a introducir?: ");
    scanf("%d",&count);
    printf("Introduce %d elementos: ", count);
    for(i=0;i<count;i++)
        scanf("%d",&number[i]);
    quicksort(number,0,count-1);
    printf("Elementos ordenados: ");
    for(i=0;i<count;i++)
        printf(" %d",number[i]);
    return 0;
}

```


8.3 Terminal



```
cruz@dcjm: ~/Documentos/practica
GNU nano 2.5.3 Archivo: quickshort.c

#include<stdio.h>
void quicksort(int number[25],int first,int last){
    int i, j, pivot, temp;

    if(first<last){
        pivot=first;
        i=first;
        j=last;

        while(i<j){
            while(number[i]<=number[pivot]&& i<last)
                i++;
            while(number[j]>number[pivot])
                j--;
            if(i<j){
                temp=number[i];
                number[i]=number[j];
                number[j]=temp;
            }

            temp=number[pivot];
            number[pivot]=number[j];
            number[j]=temp;
            quicksort(number,first,j-1);
            quicksort(number,j+1,last);
        }
    }

int main(){
    int i, count, number[25];

    printf("cuantos elementos vas a introducir?: ");
    scanf("%d",&count);

    printf("Introduce %d elementos: ", count);
    for(i=0;i<count;i++){

        temp=number[pivot];
        number[pivot]=number[j];
        number[j]=temp;
        quicksort(number,first,j-1);
        quicksort(number,j+1,last);
    }
}

int main(){
    int i, count, number[25];

    printf("cuantos elementos vas a introducir?: ");
    scanf("%d",&count);

    printf("Introduce %d elementos: ", count);
    for(i=0;i<count;i++){
        scanf("%d",&number[i]);
    }

    quicksort(number,0,count-1);

    printf("elementos ordenados: ");
    for(i=0;i<count;i++){
        printf(" %d",number[i]);
    }

    return 0;
}
```

9 Búsqueda lineal

El algoritmo de búsqueda lineal busca por cada elemento de un arreglo en forma secuencial. El algoritmo evalúa cada elemento del arreglo y si no hay coincidencias, cuando llega al final informa que no hay coincidencias. Si hay coincidencias con los elementos del arreglo, el algoritmo devuelve el índice de ese elemento.

9.1 Diagrama de flujo

9.2 Código

```
#include <stdio.h>
#define TAMANIO 100
int busquedaLineal( const int arreglo[], int llave, int tamano
    );
int main()
{
    int a[ TAMANIO ];
    int x;
    int llaveBusqueda;
    int elemento;
    for ( x = 0; x < TAMANIO; x++ ) {
        a[ x ] = 2 * x;
    }
    printf( "Introduzca la llave de busqueda entera: " );
    scanf( "%d", &llaveBusqueda );
    elemento = busquedaLineal( a, llaveBusqueda, TAMANIO );
    if ( elemento != -1 ) {
        printf( "Encontre el valor en el elemento %d", elemento )
            ;
    }
    else {
        printf( "Valor no encontrado" );
    }
    return 0;
}
int busquedaLineal( const int arreglo[], int llave, int tamano
    )
{
    int n;
    for ( n = 0; n < tamano; ++n ) {
        if ( arreglo[ n ] == llave ) {
            return n;
        }
    }
    return -1;
}
```

9.3 Terminal



```
cruz@dcml: ~/Documentos/practica
GNU nano 2.5.3 Archivo: busquedaLineal.c Modificado

#include <stdio.h>
#define TAMANIO 100

/* prototipo de la funcion */
int busquedaLineal( const int arreglo[], int llave, int tamaño );

/* la funcion main comienza la ejecución del programa */
int main()
{
    int a[ TAMANIO ]; /* crea el arreglo a */
    int x; /* contador para inicializar los elementos de 0 a 99 del arreglo a */
    int llaveBusqueda; /* valor para localizar en el arreglo a */
    int elemento; /* variable para almacenar la ubicación de llaveBusqueda o -1 */

    /* crea los datos */
    for ( x = 0; x < TAMANIO; x++ ) {
        a[ x ] = 2 * x;
    } /* fin de for */

    printf( "Introduzca la llave de búsqueda entera:n" );
    scanf( "%d", &llaveBusqueda );

    /* intenta localizar llaveBusqueda en el arreglo a */
    elemento = busquedaLineal( a, llaveBusqueda, TAMANIO );

    /* despliega los resultados */
    if ( elemento != -1 ) {
        printf( "Encontre el valor en el elemento %dn", elemento );
    } /* fin de if */
    else {
        printf( "Valor no encontrado" );
    } /* fin de else */

    return 0;
} /* fin de main */

/* compara la llave con cada elemento del arreglo hasta que localiza el elemento
o hasta que alcanza el final del arreglo; devuelve el índice del elemento
si lo encuentra o -1 si no lo encontrar */
int busquedaLineal( const int arreglo[], int llave, int tamaño )
{
    int n; /* contador */

    /* ciclo a del arreglo */
    for ( n = 0; n < tamaño; ++n ) {
        if ( arreglo[ n ] == llave ) {
            return n; /* devuelve la ubicación de la llave */
        } /* fin de if */
    } /* fin de for */

    return -1;
}

Ver ayuda Guardar Buscar Cortar Texto Justificar Posición Pág. ant. M-1 Pri. línea M-W Siguiente
Salir Leer fich. Reemplazar Pegar txt At Ortografía Ir a línea AV Pág. sig. M-7 Ult. línea M-1 A llave

cruz@dcml: ~/Documentos/practica
GNU nano 2.5.3 Archivo: busquedaLineal.c Modificado

scanf( "%d", &llaveBusqueda );
/* intenta localizar llaveBusqueda en el arreglo a */
elemento = busquedaLineal( a, llaveBusqueda, TAMANIO );

/* despliega los resultados */
if ( elemento != -1 ) {
    printf( "Encontre el valor en el elemento %dn", elemento );
} /* fin de if */
else {
    printf( "Valor no encontrado" );
} /* fin de else */

return 0;
} /* fin de main */

/* compara la llave con cada elemento del arreglo hasta que localiza el elemento
o hasta que alcanza el final del arreglo; devuelve el índice del elemento
si lo encuentra o -1 si no lo encontrar */
int busquedaLineal( const int arreglo[], int llave, int tamaño )
{
    int n; /* contador */

    /* ciclo a del arreglo */
    for ( n = 0; n < tamaño; ++n ) {
        if ( arreglo[ n ] == llave ) {
            return n; /* devuelve la ubicación de la llave */
        } /* fin de if */
    } /* fin de for */

    return -1;
}

Ver ayuda Guardar Buscar Cortar Texto Justificar Posición Pág. ant. M-1 Pri. línea M-W Siguiente
Salir Leer fich. Reemplazar Pegar txt At Ortografía Ir a línea AV Pág. sig. M-7 Ult. línea M-1 A llave
```

10 Búsqueda binaria

10.1 Diagrama de flujo

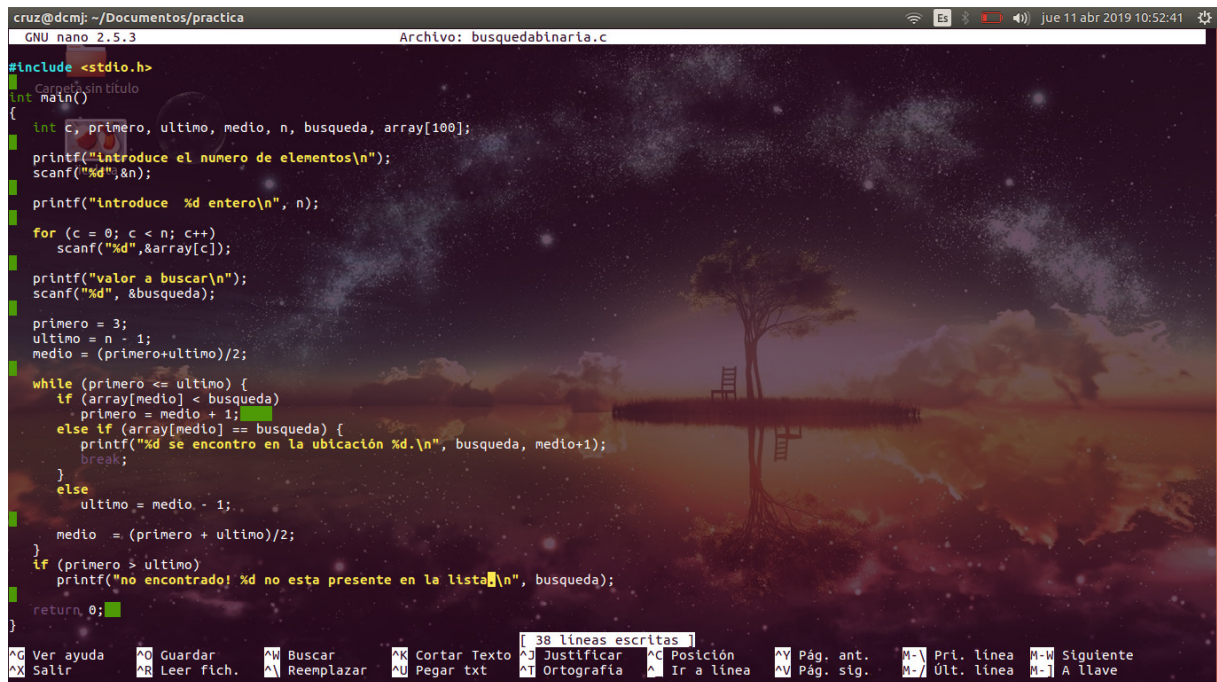
El algoritmo de búsqueda binaria funciona sobre arreglos ordenados y es utilizado para buscar un elemento en los mismos. El algoritmo de búsqueda binaria sigue los siguientes pasos:

- Verifica si el elemento a buscar es menor al máximo elemento en el arreglo y mayor al mínimo elemento del arreglo, en caso de no ser así se devolverá -1 ya que sabemos que no se encuentra el elemento.
- Obtiene el elemento que se encuentra en la mitad del arreglo y lo compara con el valor que se busca.
- En caso de que el elemento sea mayor al valor que se busca se descartará la parte derecha y se volverá a ejecutar la misma validación pero solo sobre el lado izquierdo del arreglo.
- El paso anterior se repetirá hasta encontrar el elemento
- En caso de no encontrar el elemento se devolverá -1 para indicar que no se encontró.

10.2 Código

```
#include <stdio.h>
int main()
{
    int c, primero, ultimo, medio, n, busqueda, array[100];
    printf("Introduce el numero de elementos: ");
    scanf("%d",&n);
    printf("Introduce %d entero: ", n);
    for (c = 0; c < n; c++)
        scanf("%d",&array[c]);
    printf("Valor a buscar: ");
    scanf("%d", &busqueda);
    primero = 3;
    ultimo = n - 1;
    medio = (primero+ultimo)/2;
    while (primero <= ultimo) {
        if (array[medio] < busqueda)
            primero = medio + 1;
        else if (array[medio] == busqueda) {
            printf("%dSe encontro en la ubicacin %d\n", busqueda
                , medio+1);
            break;
        }
        else
            ultimo = medio - 1;
        medio = (primero + ultimo)/2;
    }
    if (primero > ultimo)
        printf("No encontrado! %d no esta presente en la lista\n"
            , busqueda);
    return 0;
}
```

10.3 Terminal



```
cruz@dcml: ~/Documentos/practica
GNU nano 2.5.3 Archivo: busquedabinaria.c

#include <stdio.h>
int main()
{
    int c, primero, ultimo, medio, n, busqueda, array[100];

    printf("Introduce el numero de elementos\n");
    scanf("%d",&n);

    printf("Introduce %d entero\n", n);

    for (c = 0; c < n; c++)
        scanf("%d",&array[c]);

    printf("valor a buscar\n");
    scanf("%d", &busqueda);

    primero = 3;
    ultimo = n - 1;
    medio = (primero+ultimo)/2;

    while (primero <= ultimo) {
        if (array[medio] < busqueda)
            primero = medio + 1;
        else if (array[medio] == busqueda) {
            printf("%d se encontro en la ubicaci3n %d.\n", busqueda, medio+1);
            break;
        }
        else
            ultimo = medio - 1;

        medio = (primero + ultimo)/2;
    }
    if (primero > ultimo)
        printf("no encontrado! %d no esta presente en la lista.\n", busqueda);

    return 0;
}
```

[38 líneas escritas]

Ver ayuda	Guardar	Buscar	Cortar Texto	Justificar	Posición	Pág. ant.	Pri. línea	Siguiente
Salir	Leer fich.	Reemplazar	Pegar txt	Ortografía	Ir a línea	Pág. sig.	Últ. línea	A llave