INGENIERIA EN COMPUTACION

Materia: Programación Estructurada

Alumno: Ayon Dimas Andres

Matricula: 366845

Profesor: Pedro Nunes Yepiz

Actividad: #9

Tema: Librerías, métodos de búsqueda y ordenación

Ensenada B.C 10 de octubre del 2023

INTRODUCCION

En esta ocasión podremos ver lo que son las librerías propias en C, así como algunos métodos de ordenación y búsqueda. Donde podremos aprender a implementar etas sentencias en nuestros programas hechos en C, ya que esto nos ayuda a la hora de cuando nosotros queramos buscar algún dato en uno de nuestros arreglos, es aquí donde los métodos de ordenación y búsqueda nos son de suma importancia ya que con estos lo podremos hacer, además de que son datos dentro de un arreglo, datos solicitudes por el usuario.

COMPETENCIA

El alumno deberá comprender los métodos de ordenación y búsqueda, así como hacer nuestras propias librerías en C, en base a funciones para poder utilizarlas en nuestros programas y que estas cumplan la función que nosotros deseamos, debemos aprender a implementar los métodos de búsqueda y ordenación y así mismo combinar estos mismo para una búsqueda más eficiente.

FUNDAMENTOS

Métodos de búsqueda

La búsqueda es una operación que tiene por objeto la localización de un elemento dentro de la estructura de datos (arreglo). A menudo un programador estará trabajando con grandes cantidades de datos almacenados en arreglos y pudiera resultar necesario determinar si un arreglo contiene un valor que coincide con algún valor clave o buscado.

búsqueda secuencial

La **búsqueda secuencial** es un método de búsqueda utilizado en programación para encontrar un elemento específico en una lista o arreglo. En el contexto de C o cualquier otro lenguaje de programación, la búsqueda secuencial se realiza siguiendo estos pasos:

- Se comienza desde el primer elemento de la lista o arreglo.
- Se compara el elemento actual con el elemento que estás buscando.
- Si los elementos son iguales, la búsqueda se detiene y se devuelve la posición en la que se encontró el elemento.
- Si los elementos son diferentes, se pasa al siguiente elemento en la lista.

- Se repiten los pasos 2 a 4 hasta que se encuentre el elemento buscado o se llegue al final de la lista sin encontrarlo.
- La búsqueda secuencial es simple y fácil de implementar, pero puede ser ineficiente en listas grandes, ya que debe revisar cada elemento uno por uno. La cantidad de comparaciones en el peor caso es igual al tamaño de la lista.

búsqueda binaria

La búsqueda binaria, también conocida como búsqueda dicotómica, es un algoritmo de búsqueda eficiente utilizado en programación para encontrar un elemento específico en una lista o arreglo ordenado. La principal característica de la búsqueda binaria es que reduce a la mitad el espacio de búsqueda en cada iteración, lo que la hace mucho más rápida que la búsqueda secuencial en listas grandes. El proceso de búsqueda binaria en C (u otro lenguaje) se realiza de la siguiente manera:

- Se comienza con una lista ordenada (por ejemplo, en orden ascendente).
- Se establece un rango de búsqueda, generalmente utilizando dos índices, uno para el inicio y otro para el final del rango.
- Se calcula el índice del elemento medio en el rango actual (la mitad entre el índice de inicio y el índice de final).
- Se compara el elemento medio con el elemento que se busca.
- Si el elemento medio es igual al elemento buscado, la búsqueda se detiene, y se devuelve la posición del elemento medio como resultado.
- Si el elemento medio es menor que el elemento buscado, se ajusta el índice de inicio al índice del elemento medio + 1, descartando la mitad inferior del rango.
- Si el elemento medio es mayor que el elemento buscado, se ajusta el índice de final al índice del elemento medio - 1, descartando la mitad superior del rango.
- Se repiten los pasos 3 a 7 hasta que se encuentre el elemento buscado o hasta que el rango de búsqueda se reduzca a cero (lo que indica que el elemento no está en la lista).

La búsqueda binaria es especialmente eficiente en listas grandes, ya que reduce el espacio de búsqueda a la mitad en cada iteración. La cantidad máxima de comparaciones requeridas es logarítmica en el tamaño de la lista, lo que la hace mucho más rápida que la búsqueda secuencial, que puede requerir comparaciones lineales.

Métodos de ordenación

La ordenación o clasificación es el proceso de organizar datos en algún orden o secuencia específica, tal como creciente o decreciente, para datos numéricos, o alfabéticos, para datos de caracteres. Los métodos de ordenación más directos son los que se realizan en el espacio ocupado por el array.

Método de la burbuja

El método de ordenación de burbuja, también conocido como "bubble sort" en inglés, es un algoritmo simple de ordenación que se utiliza para ordenar elementos en una lista o

arreglo en orden ascendente o descendente. Aunque es fácil de entender y de implementar, el ordenamiento de burbuja no es muy eficiente en comparación con otros algoritmos de ordenación, especialmente en listas grandes.

Es más adecuado para fines educativos y de aprendizaje que para aplicaciones prácticas en la mayoría de los casos. La idea principal detrás del ordenamiento de burbuja es iterar a través de la lista varias veces, comparando pares de elementos adyacentes y realizando un intercambio si es necesario para colocar los elementos en su posición correcta.

Método de selección

El método de ordenación por selección (Selection Sort) es un algoritmo de ordenación simple que funciona seleccionando repetidamente el elemento más pequeño (o más grande, dependiendo del orden deseado) de la lista no ordenada y colocándolo al principio (o al final) de la lista ordenada. Este proceso se repite hasta que toda la lista esté ordenada.

Librerías propias en C

En C, las "librerías propias" se refieren a bibliotecas de funciones y códigos escritos por el programador o el equipo de desarrollo específicamente para un proyecto o aplicación en particular. Estas bibliotecas contienen funciones personalizadas que se adaptan a las necesidades específicas del programa que se está desarrollando.

Las librerías propias pueden contener una variedad de funciones que realizan tareas específicas y que son utilizadas repetidamente en el código de la aplicación.

PROCEDIMIENTO

Búsqueda secuencial

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define TAM 10

void main() {
int a[TAM], temp, i, j, num;

randomize(); //Inicializa el generador de numeros aleatorios
printf ("Llenando arreglo con números aleatorios\n");
for (i=0; i< TAM; i++)
    a[i]=random(100);
printf ("Numero a buscar? ");
scanf ("%d", &num);
for (i=0; i< TAM; i++)
    if (a[i] == num){
        printf ("\nValor encontrado");
        printf ("\nPosicion: %d", i);
    }
```

```
        else
            printf ("\nNo existe");
        printf ("El arreglo era:\n");
        for (i=0; i< TAM; i++)
            printf ("%d ", a[i]);
    getch();
    }
```

Búsqueda binaria

```
    #include <stdio.h>
    #include <conio.h>
    #include <stdlib.h>
    #define TAM 15

    void main(){
        int a[TAM], busca, temp, bajo, alto, central;

    printf("Llenando el arreglo con números aleatorios\n");
    randomize(); //Inicializa el generador de aleatorios
    for (int i=0; i< TAM; i++)
        a[i]=random(100);
            //Implementacion de Ordenamiento por burbuja de menor a mayor
            printf ("Ordenando arreglo...\n");
            for (int j=1; j <= TAM; j++)
                for (i=0; i< TAM-1; i++)
                    if (a[i] > a[i+1]){
                        temp = a[i];
                        a[i] = a[i+1];
                        a[i+1] = temp;
                    }

            //Implementacion de busqueda binaria
            printf ("\nIntroduce un numero a buscar: ");
            scanf ("%d", &busca);

            bajo = 0;
            alto = TAM-1;
            central = (bajo+alto)/2;
            while (bajo < alto && busca != a[central]){
                if(busca > a[central])
                    bajo = central+1;
                else
                    alto = central-1;
                central=(bajo+alto)/2;
            }

            if (busca == a[central])
                printf ("\n%d encontrado en posicion %d", busca, central);
            else
                printf ("\n%d no existe", busca);
            printf ("\n\nEl arreglo ordenado era\n\n");
            for (i=0; i< TAM; i++)
                printf ("%d ", a[i]);
    getch();
    }
```

Método de la burbuja

```
int
main(void)
{
    // El arreglo
    int arreglo[] = {30, 28, 11, 96, -5, 21, 18, 12, 22, 30, 97, -1, -40, -500};
    /*
    Calcular la longitud, puede ser definida por ti o calculada:
    https://parzibyte.me/blog/2018/09/21/longitud-de-un-arreglo-en-c/
    */
    int longitud = sizeof arreglo / sizeof arreglo[0];
    /*
    Imprimirlo antes de ordenarlo
    */
    printf("Imprimiendo arreglo antes de ordenar...\n");
    for (int x = 0; x < longitud; x++) {
      printf("%d ", arreglo[x]);
    }
    // Un salto de línea
    printf("\n");


    /*
    Invocar al método de la burbuja indicando todo el arreglo, desde 0 hasta el
    índice final
    */
    burbuja(arreglo, longitud);
    /*
    Imprimirlo después de ordenarlo
    */
    printf("Imprimiendo arreglo despues de ordenar...\n");
    for (int x = 0; x < longitud; x++)
      printf("%d ", arreglo[x]);
    return 0;
  }
```

Método de selección

```
void selection_sort(int *vector, int taille)

{
```

```
    int actual, mas_pequeno, j, temp;


    for (actual = 0; actual < taille - 1; actual++)
    {
        mas_pequeno = actual;
        for (j = actual 1; j < taille; j++)
            if (vector[j] < vector[mas_pequeno])
            mas_pequeno = j;
    temp = vector[actual];
            vector[actual] = vector[mas_pequeno];
    vector[mas_pequeno] = temp;
  }
}
```

RESULTADOS Y CONCLUSIONES

Aquí pudimos ver cómo es que se realizan estas sentencias en C y la utilidad que le podemos dar a estas mismas, ya que nos sirven para poder buscar algún dato en específico que nosotros requiramos para algo en específico. Así mismo los métodos de ordenación nos son de utilidad a la hora de ordenar algún arregle de la manera que nosotros deseamos, ya sea de forma ascendente y de forma descendente, esto nos ayuda para tener cierto control de estos mismos y que al aplicar un métodos d busqueda pueda llegar a ser más preciso.