Ingeniero en computación

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Hernandez Ceseña Ivan Fernando

Matrícula: 373077

Maestro: Pedro Núñez Yépiz

Actividad No. : 11

Tema - Unidad :FUNCIONES y METODOS DE ORDENACION Y BUSQUEDA ESTRUCTURAS Y LIBRERÍAS (p2)

Ensenada Baja California a 01 de noviembre del 2023

1. INTRODUCCIÓN

Se van a usar funciones para tener un programa con un operador múltiple (switch) y poder acceder por medio de este a la acción que se quiere ejecutar. acompañados de ciclos para automatizar selecciones y entradas de datos. Utilizando cadenas.

Se implementa el uso de una librería (.h) hecha por el alumno. En ella se encontrarán las funciones que se vayan a usar durante el programa.

2. COMPETENCIA

usar funciones para hacer un código más claro fácil de entender, además de poner en práctica la optimización de código. utilizando cadenas , validaciones y estructuras.

3. FUNDAMENTOS

- sintaxis de ciclos.
- funcionamiento de ciclos.
- proceso de ciclos.
- movilización de una función a otra.
- funcionamiento de vectores y matrices y cómo utilizarlas correctamente.
- funcionamiento de cadenas
- declaraciones de vectores y matrices.
- uso de librerías propias
- funcionamiento de estructuras.
- manipulación de estructuras.

4. PROCEDIMIENTO

Realiza el programa que contenga el siguiente menú

M E N Ú

1.- Agregar
a) manual (1)
b) Automatico (100)
c) Regresar
2.- Eliminar Registro
3.- Buscar
4.- Ordenar
5.- Imprimir
6.- Archivo Texto
0.- Salir

El programa deberá poder almacenar en un arreglo (máximo 2,000 registros) los datos para generar el CURP la estructura debe contener 2 estructuras anidadas, nombre y fecha nacimiento y un campo donde se escribirá automáticamente el curp basado en los datos proporcionados

MENÚ DESCRIPCIÓN:

1.- Cargar: Se deberá agregar 100 registros en forma automáticamente y aleatorios (cuidar no se desborde Arreglo)

2.- Eliminar: La búsqueda se realizará por matrícula, Imprimir el registro encontrado en forma de registro y preguntar si quiere eliminar si o no. (Eliminado Lógico x campo status)

3.- Buscar: La búsqueda se realizará por matrícula, el programa deberá ser capaz de realizar la búsqueda secuencial o Binaria según sea el caso. Imprimir el registro encontrado en forma de registro.

4.- Ordenar: La ordenación será por MATRICULA usar función de ordenación más adecuada según sea el caso usar 2 métodos de ordenación y el programa decidirá cuál es el que usará dependiendo del estado y tamaño de registros dentro del arreglo.
Nota: (validar si el arreglo ya está ordenado no volver ordenar por el mismo campo)

5.- Imprimir: El programa deberá imprimir los datos del arreglo (solo registros activos) en forma de tabla en pantallas de 40 registros y presionando la tecla de continuar en cada uno de los casos.

6.- Archivo de Texto: El programa deberá generar un archivo de texto con los datos del arreglo (solo registros activos) formatear salida.

NOTA: forma de registro es de la siguiente manera:

MATRICULA : 300523
NOMBRE : YAREMI
NOMBRE2 : GHIZETH
AP PATERNO : GARCIA
AP MATERNO : GUERRERO
FECHA NAC : 03-04-2010
EDAD : 19
SEXO : MUJER
LUGAR NAC : BAJA CALIFORNIA SUR
CURP : GAGY030410MBCRRRA5

NOTA : Librería Propia, Usar funciones, no se permiten variables global

5. RESULTADOS Y CONCLUSIONES

en este programa se declaran prototipos de funciones que vamos a usar para mostrar menu's, la estructura, y todas las funciones que se usaran en el programa.

```
typedef struct _names{
    char apellidoPaterno[30];
    char apellidoMaterno[30];
    char nombre[30];
} Tnamepeople;

typedef struct _fecha{
    int anio;
    int mes;
    int dia;
} Tdate;

typedef struct _alumnos{
    int status;
    int matricula;
    Tnamepeople  nombres;
    Tdate fecha;
    int edad;
    int sexo;
    int estado;
    char curp[18]; } Talumns;
```

```
int msg(void);
int msg2(void);

void generarCurpManual(Talumns reg[], int i);
void generarCurpAleatoria(Talumns reg[], int posicion, int n);
void imprimirUnAlumno(Talumns reg[], int posicion);
int existeValor(Talumns reg[], int n, int numero);
void eliminarRegistro(Talumns reg[], int posicion);
void buscarElemento(Talumns reg[], int posicion, int bandera);
void imprimirEncontrado(Talumns reg[], int buscar);
void imprimirRegistro(Talumns reg[], int posicion);
void ordenarAscendente(Talumns reg[], int n);
void cambiar(Talumns reg[], int i, int j);
int particion(Talumns reg[], int menor, int mayor);
void quicksort(Talumns reg[], int menor, int mayor);
int busquedaBinaria(Talumns reg[], int izquierda, int derecha);
void archivoTexto(Talumns reg[], int posicion);
```

funcion principal controla nuestro programa

```c
int main()
{
    srand(time(NULL));

    int op, op2, n=2000;
    int posicion = 0, aumentar;
    int bandera = 0;
    Talumns reg[n];

    do{

        op = msg();

        switch(op)
        {
            case 1:
            op2 = msg2();
            switch(op2)
            {
                case 1:
                aumentar = 1;
                if((posicion + aumentar) > n)
                {
                    aumentar = n - posicion;
                }
                if(aumentar == 0)
                {
                    printf("registro lleno.\n");
                }
                else
                {
                    generarCurpManual(reg, posicion);
                    posicion = posicion + aumentar;
                }
                break;
```

```
switch(op)
{
    case 1:
    op2 = msg2();
    switch(op2)
    {
        case 1:
        aumentar = 1;
        if((posicion + aumentar) > n)
        {
            aumentar = n - posicion;
        }
        if(aumentar == 0)
        {
            printf("registro lleno.\n");
        }
        else
        {
            generarCurpManual(reg, posicion);
            posicion = posicion + aumentar;
        }
        break;

        case 2:
        aumentar = 100;
        if((posicion + aumentar) > n)
        {
            aumentar = n - posicion;
        }
        if(aumentar == 0)
        {
            printf("registro lleno.\n");
        }
        else
        {
            generarCurpAleatoria(reg, posicion, aumentar);
            posicion = posicion + aumentar;
        }
        break;
    }
    break;

    case 2:
    eliminarRegistro(reg, posicion);
    break;

    case 3:
    buscarElemento(reg, posicion, bandera);
    break;

    case 4:
    if(posicion > 500)
    {
        printf("quicksort\n");
        quicksort(reg, 0, posicion-1);
    }
    else
```

funciones para el metodo de ordenacion quicksort

```
void cambiar(Talumns reg[], int i, int j)
{
    Talumns temp = reg[i];
    reg[i] = reg[j];
    reg[j] = temp;
}

int particion(Talumns reg[], int menor, int mayor)
{
    Talumns pivot;
    pivot.matricula = reg[mayor].matricula;
    int i = menor - 1;

    for (int j = menor; j <= mayor - 1; j++)
    {
        if (reg[j].matricula <= pivot.matricula)
        {
            i++;
            cambiar(reg, i, j);
        }
    }
    cambiar(reg, i + 1, mayor);
    return i + 1;
}

void quicksort(Talumns reg[], int menor, int mayor)
{
    if (menor < mayor)
    {
        int pi = particion(reg, menor, mayor);

        quicksort(reg, menor, pi - 1);
        quicksort(reg, pi + 1, mayor);
    }
}
```

El uso de una librería facilita mucho el entendimiento claro del código, también ayuda a que no se vea muy abultado. Esto nos ayuda a corregir errores de manera más rápida.

La lectura de todo como cadena facilita mucho su validación.

El separar acciones específicas también ayuda a poder usarlas de mejor manera dentro de otros bloques de código y tener códigos más limpios y fáciles de entender.

6. ANEXOS

anexos en el otro archivo:

nombre del archivo: anexo_HCIF_RP11_PE

7. referencias

Diseño de algoritmos y su codificación en lenguaje C

Corona, M.A. y Ancona, M.A. (2011)..

España: McGraw-Hill.

ISBN: 9786071505712

Programación estructurada a fondo:implementación de algoritmos en C

:Pearson Educación.Sznajdleder, P. A. (2017)..

Buenos Aires,Argentina: Alfaomega

Como programar en C/C++

H.M. Deitel/ P.J. Deitel

Segunda edición

Editorial: Prentice Hall.

ISBN:9688804711

Programación en C.Metodología, estructura de datos y objetos

Joyanes, L. y Zahonero, I. (2001)..

España:McGraw-Hill.

ISBN: 8448130138