UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
Facultad de arquitectura, ingeniería y diseño.(FIAD).

Actividad 10: Funciones de validar números y cadenas.

Alumna: Luisa Lizeth Zerega Soto.

Materia: Laboratorio de Programación Estructurada.

Clave: 36276.

Fecha de entrega:

Matricula: 356491.

Maestro: Pedro Nunez Yepiz.

Tema-Unidad: Funciones de validar números y cadenas.

Introducción.

El lenguaje en c ya proporciona formas de convertir números o averiguar si un **número es válido**, pero tiene algunas deficiencias.
Por ejemplo, algunas funciones dejan de leer cuando encuentran una letra e indican que el número es válido.
Entonces, leeremos el número como una cadena y luego lo recorreremos.
Para cada letra de la cadena comprobaremos si es un número, y en caso contrario comprobaremos si es la coma o el signo negativo.
Finalmente invocamos la función que convierte una cadena en un número, sin miedo a errores de conversión, ya que ya la hemos validado previamente.

Los **arrays** son variables estructuradas, donde cada elemento se almacena en un archivo, consecutivamente en la memoria.
Las cadenas de caracteres se declaran en C en forma de matrices de caracteres y permite uso de una serie de nociones y funciones especiales.

Una **arrays (unidimensional, también llamada vector)** es una variable estructurada formada por un número “n” de variables simples del mismo tipo que llamamos los componentes o elementos de arrays. Por tanto, el número de componentes es “n”. tamaño de arrays. Al igual que en matemáticas, decimos que "A" es una vector de dimensión “n”.

Competencia.

El alumno deberá utilizar funciones de validar números y cadenas para el correcto funcionamiento de datos de caracteres en menús con funciones y métodos de ordenación y búsqueda estructura y librerías.

Fundamentos.

- ★ Estructuras. (n.d.). http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/control/lengua_C/estructuras.htm
- ★ Estructuras en c. (n.d.). kesquivel.files.wordpress.com. Retrieved October 15, 2023, from https://kesquivel.files.wordpress.com/2013/05/estructuras2013final2.pdf

★ Principios de Programación: El lenguaje C. (n.d.). www.fing.edu.uy. Retrieved October 15, 2023, from https://www.fing.edu.uy/tecnoinf/mvd/cursos/prinprog/material/teo/prinprog-teorico08.pdf

Procedimiento.

ACTIVIDAD 10
REALICE EL SIGUIENTE PROGRAMA QUE CONTENGA UN MENÚ.

MENÚ
1.- AGREGAR (AUTOM 10 REGISTROS)
2.- AGREGAR MANUAL
3- ELIMINAR REGISTRO (lógico)
4.- BUSCAR
5- ORDENAR
6.- IMPRIMIR
0.- SALIR

UTILIZAR UN ARREGLO DE 500 REGISTROS,SE DEBERÁ UTILIZAR ESTRUCTURAS CON LOS DATOS BÁSICOS DE UN ALUMNO (status, Matricula, ApPat, ApMat, Nombre, Edad, Sexo) y Busqueda y Ordenacion por campo MATRICULA.

nota: usar librería propia

Resultados.

```
// LUISA LIZETH ZEREGA SOTO.
// MAT:356491
// FECHA: 17-OCT-2023
// DESCRIP: Funciones de validar numeros y cadenas.
// ZSL_ACT10.CPP

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

#define MAX_REGISTROS 500
```

```c
struct Alumno {
    int status;
    int matricula;
    char apPat[50];
    char apMat[50];
    char nombre[50];
    int edad;
    char sexo;
};

void agregarAutom(int cantidad, struct Alumno alumnos[]) {
    for (int i = 0; i < cantidad; i++) {
        alumnos[i].status = 1;
        alumnos[i].matricula = i + 1;
        sprintf(alumnos[i].apPat, "Apellido%d", i + 1);
        sprintf(alumnos[i].apMat, "Apellido%d", i + 1);
        sprintf(alumnos[i].nombre, "Nombre%d", i + 1);
        alumnos[i].edad = 20 + i;
        alumnos[i].sexo = 'M';
    }
}

void agregarManual(struct Alumno alumnos[], int *cantidad) {
    if (*cantidad < MAX_REGISTROS) {
        printf("Ingrese los datos del alumno:\n");
        printf("Matricula: ");
        scanf("%d", &alumnos[*cantidad].matricula);
        printf("Apellido Paterno: ");
        scanf("%s", alumnos[*cantidad].apPat);
        printf("Apellido Materno: ");
        scanf("%s", alumnos[*cantidad].apMat);
        printf("Nombre: ");
        scanf("%s", alumnos[*cantidad].nombre);
        printf("Edad: ");
        scanf("%d", &alumnos[*cantidad].edad);
        printf("Sexo (M/F): ");
        scanf(" %c", &alumnos[*cantidad].sexo);

        alumnos[*cantidad].status = 1;
        (*cantidad)++;

        printf("Registro agregado correctamente.\n");
```

```
    } else {
        printf("No se pueden agregar más registros. El arreglo está
lleno.\n");
    }
}

void eliminarRegistro(struct Alumno alumnos[], int cantidad) {
    int matricula;
    printf("Ingrese la matricula del alumno a eliminar: ");
    scanf("%d", &matricula);

    for (int i = 0; i < cantidad; i++) {
        if (alumnos[i].matricula == matricula) {
            alumnos[i].status = 0;
            printf("Registro eliminado correctamente.\n");
            return;
        }
    }

    printf("No se encontró ningún registro con esa matricula.\n");
}

void buscar(struct Alumno alumnos[], int cantidad) {
    int matricula;
    printf("Ingrese la matricula del alumno a buscar: ");
    scanf("%d", &matricula);

    for (int i = 0; i < cantidad; i++) {
        if (alumnos[i].matricula == matricula && alumnos[i].status ==
1) {
            printf("Registro encontrado:\n");
            printf("Matricula: %d\n", alumnos[i].matricula);
            printf("Apellido Paterno: %s\n", alumnos[i].apPat);
            printf("Apellido Materno: %s\n", alumnos[i].apMat);
            printf("Nombre: %s\n", alumnos[i].nombre);
            printf("Edad: %d\n", alumnos[i].edad);
            printf("Sexo: %c\n", alumnos[i].sexo);
            return;
        }
    }

    printf("No se encontró ningún registro con esa matricula.\n");
}
```

```c
void ordenar(struct Alumno alumnos[], int cantidad) {
    struct Alumno temp;

    for (int i = 0; i < cantidad - 1; i++) {
        for (int j = 0; j < cantidad - i - 1; j++) {
            if (alumnos[j].matricula > alumnos[j + 1].matricula) {
                temp = alumnos[j];
                alumnos[j] = alumnos[j + 1];
                alumnos[j + 1] = temp;
            }
        }
    }

    printf("Registros ordenados correctamente.\n");
}

void imprimir(struct Alumno alumnos[], int cantidad) {
    for (int i = 0; i < cantidad; i++) {
        if (alumnos[i].status == 1) {
            printf("Registro %d:\n", i + 1);
            printf("Matricula: %d\n", alumnos[i].matricula);
            printf("Apellido Paterno: %s\n", alumnos[i].apPat);
            printf("Apellido Materno: %s\n", alumnos[i].apMat);
            printf("Nombre: %s\n", alumnos[i].nombre);
            printf("Edad: %d\n", alumnos[i].edad);
            printf("Sexo: %c\n", alumnos[i].sexo);
        }
    }
}

int main() {
    struct Alumno alumnos[MAX_REGISTROS];
    int cantidad = 0;
    int opcion;

    do {
        printf("\n");
        printf("MENÚ\n");
        printf("1.- AGREGAR (AUTOM 10 REGISTROS)\n");
        printf("2.- AGREGAR MANUAL\n");
        printf("3.- ELIMINAR REGISTRO (lógico)\n");
        printf("4.- BUSCAR\n");
```

```
        printf("5.- ORDENAR\n");
        printf("6.- IMPRIMIR\n");
        printf("0.- SALIR\n");
        printf("Ingrese una opción: ");
        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                agregarAutom(10, alumnos);
                printf("Registros agregados automáticamente.\n");
                break;
            case 2:
                agregarManual(alumnos, &cantidad);
                break;
            case 3:
                eliminarRegistro(alumnos, cantidad);
                break;
            case 4:
                buscar(alumnos, cantidad);
                break;
            case 5:
                ordenar(alumnos, cantidad);
                break;
            case 6:
                imprimir(alumnos, cantidad);
                break;
            case 0:
                printf("Saliendo del programa...\n");
                break;
            default:
                printf("Opción inválida. Por favor, ingrese una opción
válida.\n");
                break;
        }
    } while (opcion != 0);

    return 0;
}
```

Conclusiones.

Se aprendió a utilizar funciones de validar números y cadenas para el correcto funcionamiento de datos de caracteres en menús con funciones y métodos de ordenación y búsqueda estructura y librerías.

Anexos.