Laboratorio 13

Programación de Computadores (2023-2)

Temas del Laboratorio 13

- Uso de estructuras en C
- Serialización de datos
- Ejercicios

Structs (Estructuras)

Una estructura (struct) es un tipo de dato que permite combinar diferentes tipos de datos bajo un solo nombre.
Es una manera de organizar datos de manera que sea más fácil de entender y manipular.

```
struct NombreDeLaEstructura {
    tipoDeDato1 campo1;
    tipoDeDato2 campo2;
    // ... más campos si es necesario
};
```

Declaración e inicialización de structs

Se puede de declarar una variable del tipo struct y luego inicializar los valores de cada campo usando el operador punto (.)

También se puede inicializar los valores de los campos del struct al momento de declararlo, ingresando los valores en el mismo orden que se define en el struct.

La lectura o modificación de los campos del struct se realiza mediante el operador punto (.)

```c
# include <stdio.h>

struct punto{
    int x;
    int y;
};

#include <stdio.h>

int main() {

    // Declaración de la variable de tipo struct punto
    struct punto p;

    // Inicialización de la variable
    p.x = 1;
    p.y = 2;

    // Declaración e inicialización de la variable
    struct punto q = {3, 4};

    // Impresión de los valores
    printf("punto p = %d, %d\n", p.x, p.y);
    printf("punto q = %d, %d\n", q.x, q.y);

    return 0;
}
```

Typedef en Structs

El uso de typedef en C es común cuando se trabaja con structs para crear un alias legible para el tipo de datos de la estructura.

Esto ayuda a simplificar la declaración de variables y hace que el código sea más claro.

```c
#include <stdio.h>

// Definición de la struct con typedef
typedef struct {
    int x;
    int y;
} Punto;  // Punto es ahora un alias para struct Punto

int main() {
    // Declaración e inicialización de una variable de tipo Punto
    Punto punto1 = {1, 2};

    // Accediendo a los miembros de la struct
    printf("Coordenadas del punto: (%d, %d)\n", punto1.x, punto1.y);

    return 0;
}
```

Arreglos de Structs

Al igual que con otros tipos de datos, es posible declarar arreglos de structs en C.

Se puede acceder a cada campo de cada elemento del arreglo mediante el índice del elemento.

```c
#include <stdio.h>

typedef struct {
    int x;
    int y;
} Punto;

int main() {
    // Declaración del arreglo de 3 puntos
    Punto puntos[3];

    // Ingreso de valores por el usuario para cada punto
    for (int i = 0; i < 3; ++i) {
        printf("Ingrese las coordenadas del punto %d (x y): ", i + 1);
        scanf("%d %d", &puntos[i].x, &puntos[i].y);
    }

    // Imprimir los valores ingresados
    printf("\nValores ingresados:\n");
    for (int i = 0; i < 3; ++i) {
        printf("Punto %d: (%d, %d)\n", i + 1, puntos[i].x, puntos[i].y);
    }

    return 0;
}
```

Structs en memoria dinámica

Cuando se declara un struct en memoria dinámica se utiliza el operador flecha (**->**) en vez del operador punto (**.**) para acceder a los campos del struct.

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

typedef struct {
    int x;
    int y;
} Punto;

int main() {
    // Declaración de un struct punto en memoria dinámica
    Punto *p = malloc(sizeof(Punto));

    // Ingreso de los valores de x e y desde la terminal
    printf("Ingrese el valor de x: ");
    scanf("%d", &p->x);
    printf("Ingrese el valor de y: ");
    scanf("%d", &p->y);

    // Impresión de los valores de x e y
    printf("Las coordenadas del punto son: (%d, %d)\n", p->x, p->y);

    free(p);

    return 0;
}
```

Arreglo de Structs en memoria dinámica

Cuando se declara un arreglo de structs en memoria dinámica se accede a los campos de cada elemento mediante el operador punto (**.**), de la misma manera como se accede en memoria estática.

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

typedef struct {
    int x;
    int y;
} Punto;

int main() {
    // Declaración de un arreglo de 5 puntos en memoria dinámica
    Punto *puntos = malloc(sizeof(Punto) * 5);

    // Ingreso de los valores de los puntos desde la terminal
    for(int i = 0; i < 5; i++){
        printf("Ingrese el valor de x del punto %d: ", i+1);
        scanf("%d", &puntos[i].x);
        printf("Ingrese el valor de y del punto %d: ", i+1);
        scanf("%d", &puntos[i].y);
    }

    // Impresión de los valores de los puntos
    for(int i = 0; i < 5; i++){
        printf("Punto %d: (%d, %d)\n", i+1, puntos[i].x, puntos[i].y);
    }

    free(puntos);

    return 0;
}
```

Ejercicio 1

Considere la siguiente función

$$f(x, y, z) = x^4 - 2x^2 + y^2 + z^2.$$

Cree un programa que reciba N puntos y decida cuál es el punto que alcanza el mayor y el menor valor dentro de la función e imprime el punto y el valor correspondiente

Debe crear un arreglo de structs usando malloc, para guardar los N puntos

Ejercicio 2

Cree un programa para guardar datos de un número N de estudiantes usando struct.

La struct de cada estudiante debe contener:
Nombre del estudiante,
Matrícula,
Evaluación 1,
Evaluación 2,
Evaluación 3,
Promedio.

La struct debe tener un alias usando typedef

Pedir al usuario que ingrese los datos para 5 estudiantes (el promedio no se ingresa, se calcula con los datos ingresados de las tres evaluaciones).

Luego se deben leer los datos, calcular el promedio para luego mostrar el nombre, matrícula, promedio y si aprobaron o no. (promedio >=3.95).