Informe de las prácticas de experimentación y aplicación de los aprendizajes

1. Datos Informativos:

Facultad:	*CIENCIAS ADMINISTRATIVAS GESTIÓN EMPRESARIAL E INFORMÁTICA*
Carrera:	*SOFTWARE*
Asignatura:	*Algoritmos y lógica de programación*
Ciclo:	*Primero*
Docente:	*ING. MÓNICA BONILLA*
Título de la práctica:	SENTENCIAS DE REPETICIÓN
No. de práctica:	*No. 2*
Escenario o ambiente de aprendizaje de la practica	*DEV C++. - Entorno de desarrollo integrado para programar en lenguaje C/C++*
No. de horas:	***10***
Fecha:	***30-05-2024***
Estudiantes:	Ariel Alejandro Calderón Jacson Antonio Narváez
Calificación	

2. Introducción:

Las sentencias de repetición, también conocidas como bucles, son estructuras de control en los lenguajes de programación que permiten ejecutar un bloque de código múltiples veces de manera eficiente. Se utilizan para automatizar tareas repetitivas, iterar sobre colecciones de datos y gestionar flujos de trabajo que requieren repetición controlada.

3. Objetivo de la práctica:

Comprender y aplicar las sentencias de repetición para automatizar tareas repetitivas, optimizar la ejecución de bloques de código y manejar flujos de trabajo complejos en diferentes lenguajes de programación.

4. Descripción del desarrollo de la práctica:

4.1 Uso de Sentencias de Repetición en C

Las sentencias de repetición en el lenguaje de programación C son fundamentales para la creación de bucles, permitiendo ejecutar un bloque de código varias veces de manera controlada y eficiente. Los tres tipos principales de sentencias de repetición en C son `for`, `while` y `do...while`.

4.2 Estructuras condicionales más comunes

- Bucle `for`: Se utiliza cuando se conoce el número exacto de iteraciones que se desean realizar. En este bucle, la inicialización se ejecuta una vez al principio, la condición se evalúa antes de cada iteración, y el incremento se ejecuta después de cada iteración.

- Bucle `while`: Se utiliza cuando no se conoce de antemano el número de iteraciones y se desea ejecutar el bloque de código mientras se cumpla una condición. En este bucle, la condición se evalúa antes de cada iteración, y el bloque de código se ejecuta mientras la condición sea verdadera.

- Bucle `do...while`: Es similar al `while`, pero garantiza que el bloque de código se ejecute al menos una vez, ya que la condición se evalúa después de cada iteración. En este bucle, el bloque de código se ejecuta una vez antes de que la condición se evalúe.

4.3 Aplicación de sentencias de repetición en lenguaje C

4.3.1 Problema

*Realizar el análisis respectivo utilizar los ciclos de repetición en las diferentes operaciones (for, while, o do while) para generar las tablas de suma, resta, multiplicación, división y potenciación el usuario deberá ingresar los rangos de los números de las tablas que necesita imprimir por cada operación señalada y elegir la opción salir para finalizar la ejecución del programa. Ejemplo: 8*1=5 8*2=10 8*3=15…8*25=100*

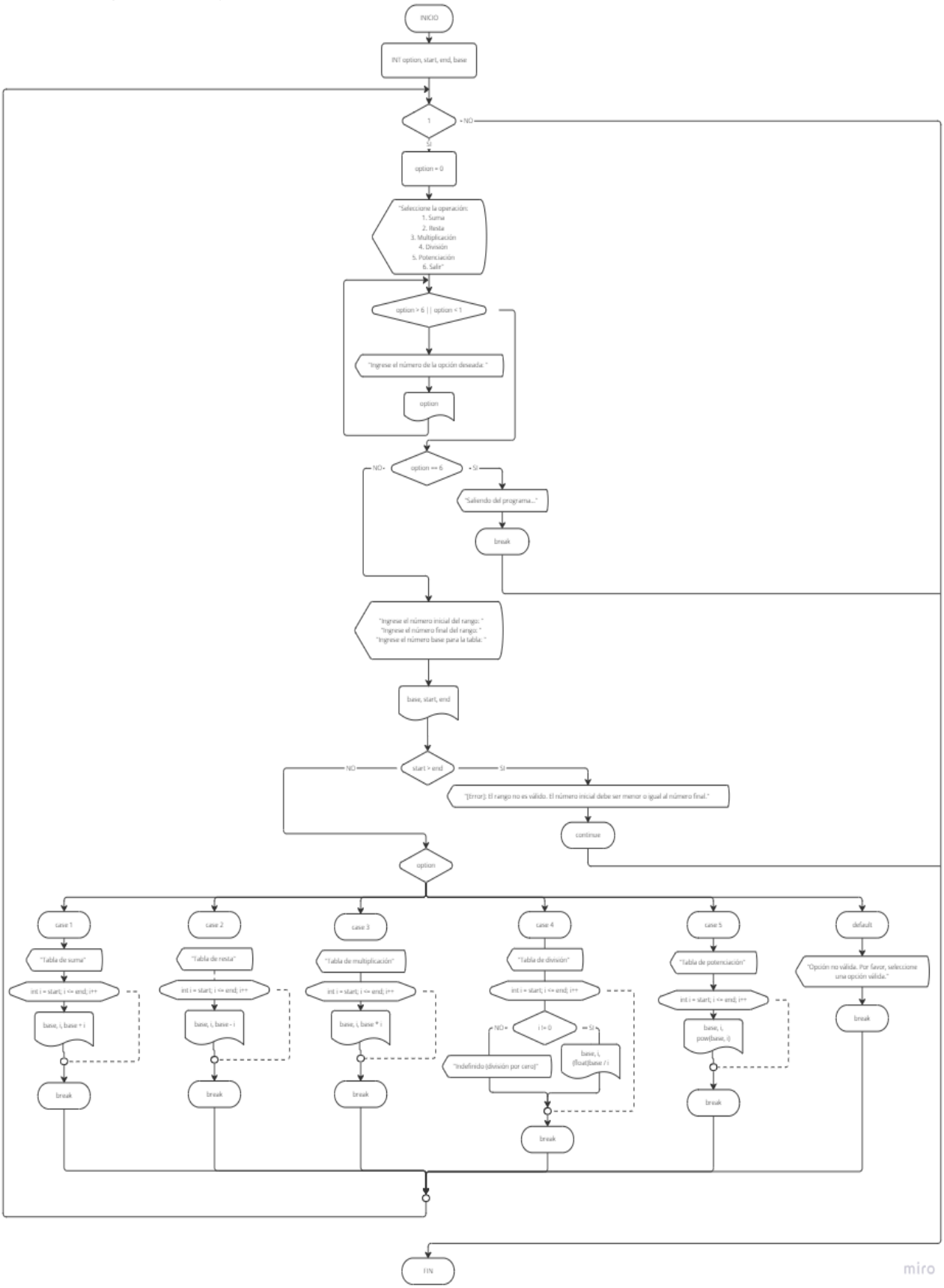
4.3.2 Pseudocódigo

```
INICIO
  DECLARAR entero option, start, end, base


  MIENTRAS verdadero
    ASIGNAR 0 a option
    IMPRIMIR "Seleccione la operación:"
```

```
    IMPRIMIR "1. Suma"
    IMPRIMIR "2. Resta"
    IMPRIMIR "3. Multiplicación"
    IMPRIMIR "4. División"
    IMPRIMIR "5. Potenciación"
    IMPRIMIR "6. Salir"

    MIENTRAS option > 6 O option < 1
      IMPRIMIR "Ingrese el número de la opción deseada: "
      LEER option

    SI option = 6 ENTONCES
      IMPRIMIR "Saliendo del programa..."
      SALIR DEL BUCLE
    FIN SI

    IMPRIMIR "Ingrese el número inicial del rango: "
    LEER start
    IMPRIMIR "Ingrese el número final del rango: "
    LEER end
    IMPRIMIR "Ingrese el número base para la tabla: "
    LEER base

    SI start > end ENTONCES
      IMPRIMIR "[Error]: El rango no es válido. El número inicial debe ser menor o igual al número
final."
      CONTINUAR
    FIN SI


    SEGÚN option HACER
      CASO 1:
        IMPRIMIR "Tabla de suma para base del start al end:"
        PARA i DESDE start HASTA end HACER
          IMPRIMIR "base + i = base + i"
        FIN PARA
```

```
    CASO 2:
      IMPRIMIR "Tabla de resta para base del start al end:"
      PARA i DESDE start HASTA end HACER
        IMPRIMIR "base - i = base - i"
      FIN PARA

    CASO 3:
      IMPRIMIR "Tabla de multiplicación para base del start al end:"
      PARA i DESDE start HASTA end HACER
        IMPRIMIR "base * i = base * i"
      FIN PARA

    CASO 4:
      IMPRIMIR "Tabla de división para base del start al end:"
      PARA i DESDE start HASTA end HACER
        SI i != 0 ENTONCES
          IMPRIMIR "base / i = base / i"
        SINO
          IMPRIMIR "base / i = indefinido (división por cero)"
        FIN SI
      FIN PARA

    CASO 5:
      IMPRIMIR "Tabla de potenciación para base del start al end:"
      PARA i DESDE start HASTA end HACER
        IMPRIMIR "base ^ i = pow(base, i)"
      FIN PARA

    OTRO CASO:
      IMPRIMIR "Opción no válida. Por favor, seleccione una opción válida."
  FIN SEGÚN

FIN MIENTRAS

FIN
```

4.3.3 Diagrama de flujo



4.3.4 Codificación

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

int main()
{
    int option, start, end, base;
    while (1)
    {
        option = 0;
        printf("\nSeleccione la operación:\n");
        printf("1. Suma\n");
        printf("2. Resta\n");
        printf("3. Multiplicación\n");
        printf("4. División\n");
        printf("5. Potenciación\n");
        printf("6. Salir\n");

        while (option > 6 || option < 1)
        {
            printf("Ingrese el número de la opción deseada: ");
            scanf("%d", &option);
        }
        if (option == 6)
        {
            printf("Saliendo del programa...\n");
            break;
        }
        printf("Ingrese el número inicial del rango: ");
        scanf("%d", &start);
        printf("Ingrese el número final del rango: ");
        scanf("%d", &end);
        printf("Ingrese el número base para la tabla: ");
        scanf("%d", &base);

        if (start > end)
```

```
        {
            printf("[Error]: El rango no es válido. El número inicial debe ser menor o
igual al número final.\n");
            continue;
        }
        switch (option)
        {
        case 1:
            printf("Tabla de suma para %d del %d al %d:\n", base, start, end);
            for (int i = start; i <= end; i++)
            {
                printf("%d + %d = %d\n", base, i, base + i);
            }
            break;
        case 2:
            printf("Tabla de resta para %d del %d al %d:\n", base, start, end);
            for (int i = start; i <= end; i++)
            {
                printf("%d - %d = %d\n", base, i, base - i);
            }
            break;
        case 3:
            printf("Tabla de multiplicación para %d del %d al %d:\n", base, start, end);
            for (int i = start; i <= end; i++)
            {
                printf("%d * %d = %d\n", base, i, base * i);
            }
            break;
        case 4:
            printf("Tabla de división para %d del %d al %d:\n", base, start, end);
            for (int i = start; i <= end; i++)
            {
                if (i != 0)
                {
                    printf("%d / %d = %.2f\n", base, i, (float)base / i);
                }
```

```
                else
                {
                    printf("%d / %d = indefinido (división por cero)\n", base, i);
                }
            }
            break;
        case 5:
            printf("Tabla de potenciación para %d del %d al %d:\n", base, start, end);
            for (int i = start; i <= end; i++)
            {
                printf("%d ^ %d = %.0f\n", base, i, pow(base, i));
            }
            break;
        default:
            printf("Opción no válida. Por favor, seleccione una opción válida.\n");
            break;
        }
    }
    return 0;
}
```

5. Metodología:

Experimentar con proyectos minimalistas donde se pueda poner en práctica el uso de sentencias repetitivas para optimizar el flujo de tu programa.

6. Resultados obtenidos:

Ejecución del programa en C

```
Seleccione la operación:
1. Suma
2. Resta
3. Multiplicación
4. División
5. Potenciación
6. Salir
Ingrese el número de la opción deseada: 5
Ingrese el número inicial del rango: 1
Ingrese el número final del rango: 9
Ingrese el número base para la tabla: 2
Tabla de potenciación para 2 del 1 al 9:
2 ^ 1 = 2
2 ^ 2 = 4
2 ^ 3 = 8
2 ^ 4 = 16
2 ^ 5 = 32
2 ^ 6 = 64
2 ^ 7 = 128
2 ^ 8 = 256
2 ^ 9 = 512

Seleccione la operación:
1. Suma
2. Resta
3. Multiplicación
4. División
5. Potenciación
6. Salir
Ingrese el número de la opción deseada:
```

7. Conclusiones:

Las sentencias de repetición son herramientas poderosas en C que, cuando se utilizan correctamente, pueden hacer que el código sea más eficiente, mantenible y fácil de entender.

8. Recomendaciones:

- Evitar bucles infinitos: Asegurarse de que la condición de terminación se alcanzará eventualmente.
- Optimización: Escribir bucles eficientes para minimizar el tiempo de ejecución y uso de recursos.

9. Bibliografía:

Autor	Titulo	Año	Ciudad	Editorial	ISBN	Código
Joyanes Aguilar, Luis	Programación en C, C++, Java y UML	2014	México, D.F	McGrawHill	9786071512123.	005.13 3 J88p
Joyanes Aguilar, Luis	Programación en C: Libro de problemas	2002	España	McGraw Hill Interamericana	8448136225	005.1 J88p

10. Anexos: