

## Proyecto # 3 | Código en Lenguaje C

### Introducción al desarrollo de software

Ingeniería en Desarrollo de Software



academiaglobal

TUTOR: SANDRA LARA DEVORA

ALUMNO: JOSE JOEL LANDEROS SANTOS

FECHA: 8 DICIEMBRE DE 2024

## **Introducción**

En este ejercicio se aprenderá sobre el lenguaje de programación C, una herramienta para entender la lógica y estructura detrás del funcionamiento de una computadora. A través de la resolución de ejercicios se explorarán conceptos como la sintaxis del lenguaje, el uso de variables, condicionales, bucles, métodos y las operaciones de entrada y salida. Estas actividades permiten desarrollar habilidades en programación, organizando los pasos necesarios para resolver problemas de forma lógica.

En este proyecto, se practicará cómo implementar ejercicios que refuercen el entendimiento de los fundamentos de C, descomponiendo procesos complejos en instrucciones más simples. Además, se destacará la importancia de comprender cómo los programas interactúan con el hardware, lo que proporciona una base para resolver problemas computacionales y diseñar soluciones que optimicen recursos. Este enfoque práctico es para dominar los principios básicos de la programación y avanzar en el desarrollo de software.

## **Descripción**

Un lenguaje de programación es un sistema formal compuesto por reglas y sintaxis específicas que permite a los desarrolladores comunicarse con las computadoras. Mediante este lenguaje, los programadores pueden escribir instrucciones que una máquina interpreta y ejecuta para realizar tareas específicas, como cálculos, manejo de datos, automatización de procesos o desarrollo de aplicaciones. Los lenguajes de programación actúan como un puente entre las ideas humanas y las operaciones que realiza el hardware, transformando conceptos abstractos en instrucciones concretas y ejecutables.

C, por su parte, es un lenguaje de programación de propósito general, imperativo y estructurado. Se considera un lenguaje de bajo nivel en comparación con lenguajes más modernos, ya que permite un control detallado del hardware y de los recursos del sistema. Su diseño también incorpora elementos de alto nivel que facilitan la gestión de estructuras de datos y algoritmos. Es ampliamente utilizado para desarrollar sistemas operativos, controladores de dispositivos y programas que requieren alta eficiencia, siendo una herramienta clave para programadores que buscan optimizar el rendimiento y la interacción directa con el hardware.

## **Justificación**

Realizar esta práctica con OnlineGDB es una buena manera de comenzar con C, ya que ofrece un entorno de desarrollo fácil de usar directamente desde el navegador sin necesidad de instalar software adicional. OnlineGDB permite a los principiantes escribir, depurar y probar sus programas en un entorno controlado y accesible. El lenguaje de programación C es una excelente opción para aprender a programar debido a su cercanía con el hardware y la eficiencia que ofrece en la manipulación de recursos del sistema. Al comenzar con C, los programadores adquieren una comprensión profunda de cómo funcionan las computadoras a nivel interno, lo que les permite optimizar sus programas y entender mejor la lógica de los sistemas operativos y el software de bajo nivel.

## Desarrollo

### Calculadora: Primos

La empresa MathTech requiere a un ingeniero en desarrollo de software que sea capaz de realizar la tarea de programar tres tipos de calculadoras diferentes para implementar en los colegios y escuelas públicas: - La primera calculadora deberá de llevar por nombre Primos, y su objetivo será identificar los números primos que se ingresen, por ejemplo si el usuario ingresa el número 83, deberá imprimir el siguiente mensaje: “El número (número ingresado) si es primo”, en caso de que no sea primo se imprimirá el siguiente mensaje “El número (número ingresado) no es primo”.

```

1 #include <stdio.h>
2 #include <math.h>
3
4 int main() {
5     int numero, esPrimo = 1;
6     printf("Ingrese un número entero: ");
7     scanf("%d", &numero);
8
9     if (numero <= 1) {
10         esPrimo = 0; // Los números <= 1 no son primos
11     } else {
12         for (int i = 2; i <= sqrt(numero); i++) {
13             if (numero % i == 0) {
14                 esPrimo = 0;
15                 break; // Salir del bucle si se encuentra un divisor
16             }
17         }
18     }
19     if (esPrimo) {
20         printf("El número %d es primo.\n", numero);
21     } else {
22         printf("El número %d no es primo.\n", numero);
23     }
24     return 0;
25 }
26

```

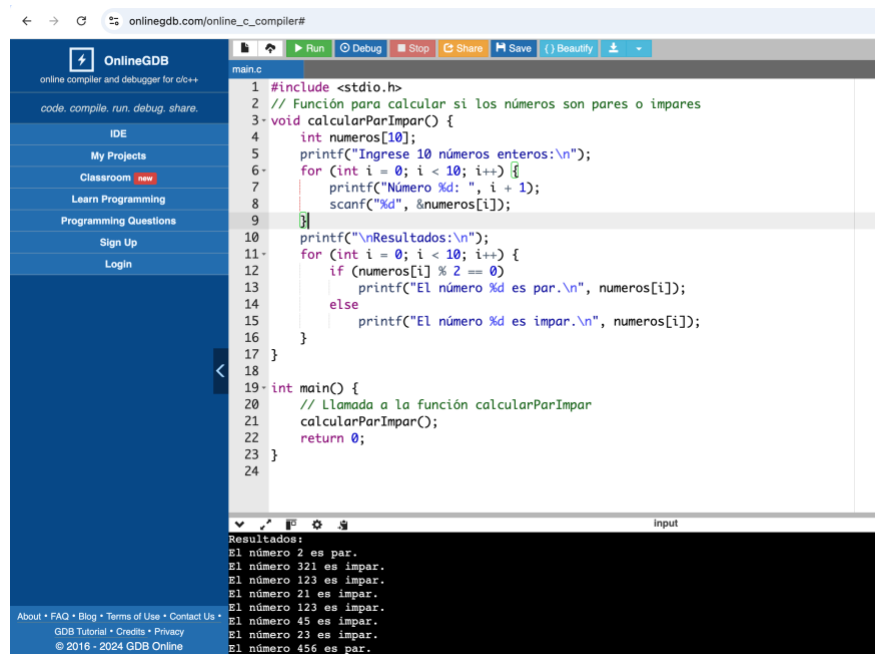
Input: 4  
Output: El número 4 no es primo.  
...Program finished with exit code 0  
Press ENTER to exit console.

Imagen 1.1 captura de pantalla de OnlineGDB donde muestra el código en C del programa de ejercicio de si es primo o no es primo.

Tomando en cuenta el código Imagen 1.1, podemos observar cómo se estructura la lógica con el lenguaje C solicitando al usuario que introduzca un número entero con una variable entera llamada “int” que se tuvo que inicializar y usar un método scan de una librería. También el uso del condicional “if” que nos sirvió para darle a entender a la computadora que, si el número es menor o igual a 1, en el cuerpo del if que da como resultado “true” se tenga que seguir la instrucción de cambiar el booleano como “no primo” que es falso que es equivalente a un 0. Además de que si es mayor que 1, se comprueba con una comparación dentro de un “if” si es divisible por algún número desde 2 hasta la raíz cuadrada del número utilizando métodos matemáticos de una librería que ya tiene C que nos sirve para automatizar las matemáticas como saber si se encuentra un divisor, y dar como resultado en el cuerpo del if que el número no es primo y se imprime el resultado con printf que es el comando para imprimir líneas de texto que nos permite darle vida al código para saber si el número es primo o no.

## Calculadora: Par/Impar

La segunda calculadora se llamará Par/Impar, su objetivo es que se ingresen 10 números, ya sean pares o impares, por ejemplo, si se ingresa el número 9, el programa deberá de indicar que es un número impar, pero si se trata del número 2, el programa deberá indicar que se trata de un número par. De 10 números enteros, se debe determinar cuáles son pares y cuáles son impares.



```

1 #include <stdio.h>
2 // Función para calcular si los números son pares o impares
3 void calcularParImpar() {
4     int numeros[10];
5     printf("Ingrese 10 números enteros:\n");
6     for (int i = 0; i < 10; i++) {
7         printf("Número %d: ", i + 1);
8         scanf("%d", &numeros[i]);
9     }
10    printf("\nResultados:\n");
11    for (int i = 0; i < 10; i++) {
12        if (numeros[i] % 2 == 0) {
13            printf("El número %d es par.\n", numeros[i]);
14        } else {
15            printf("El número %d es impar.\n", numeros[i]);
16        }
17    }
18 }
19 int main() {
20     // Llamada a la función calcularParImpar
21     calcularParImpar();
22     return 0;
23 }
24
Resultados:
El número 2 es par.
El número 321 es impar.
El número 123 es impar.
El número 21 es impar.
El número 123 es impar.
El número 45 es impar.
El número 23 es impar.
El número 456 es par.

```

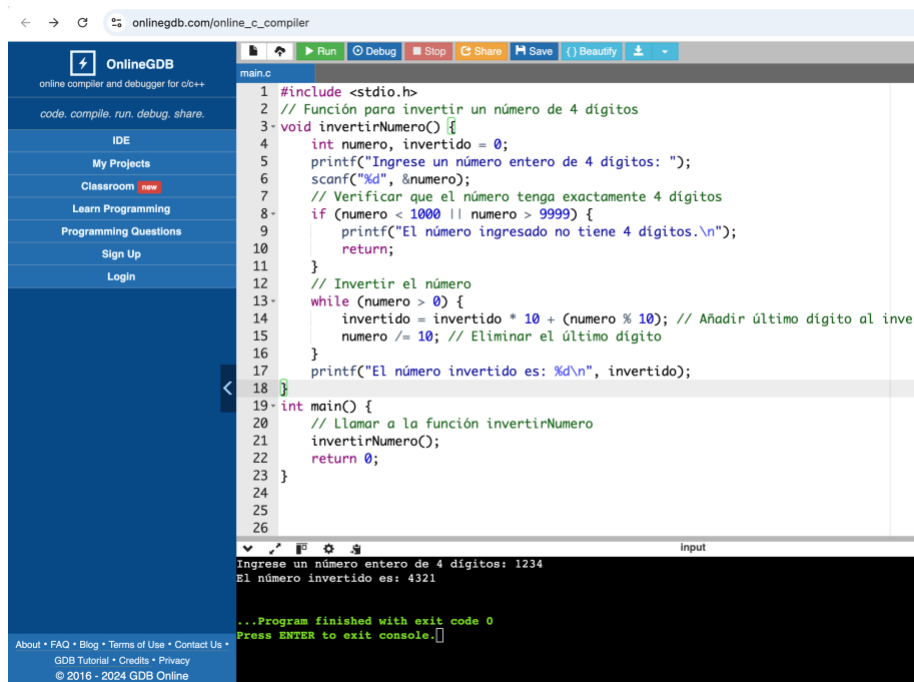
Imagen 1.2 captura de pantalla de OnlineGDB donde muestra el código en C del programa calcular par e impar.

Como vemos en la Imagen 1.2, se solicita al usuario que ingrese 10 números enteros, utilizando un ciclo for que recorre 10 iteraciones. En cada iteración: Se imprime un mensaje para que el usuario sepa qué número debe ingresar. El número ingresado por el usuario se guarda en el arreglo `numeros[i]`, utilizando `scanf` para capturar la entrada. Se utiliza el operador módulo `%` para verificar si el número es divisible por 2. Si el residuo de la división de `numeros[i]` entre 2 es 0 (`numeros[i] % 2 == 0`), entonces el número es par. Si la condición se cumple, se imprime que el número es par. Si la condición no se cumple (es decir, si el número no es divisible por 2), el número es impar, y en ese caso se imprime que el número es impar. Esto se hace para cada número en el arreglo `numeros[i]`, lo que equivale a verificar cada número y luego imprimir si es par o impar.

## Calculadora: Al Revés

El último programa se llamará Al Revés, su objetivo es que el usuario ingrese un número de 4 dígitos y que sea un número entero, y este programa se encargará de regresar los números al revés o invertidos. Por ejemplo, si se ingresa el número 7631, el programa matemático deberá regresar 1367.

### 1. Código



```

1 #include <stdio.h>
2 // Función para invertir un número de 4 dígitos
3 void invertirNumero() {
4     int numero, invertido = 0;
5     printf("Ingrese un número entero de 4 dígitos: ");
6     scanf("%d", &numero);
7     // Verificar que el número tenga exactamente 4 dígitos
8     if (numero < 1000 || numero > 9999) {
9         printf("El número ingresado no tiene 4 dígitos.\n");
10        return;
11    }
12    // Invertir el número
13    while (numero > 0) {
14        invertido = invertido * 10 + (numero % 10); // Añadir último dígito al invertido
15        numero /= 10; // Eliminar el último dígito
16    }
17    printf("El número invertido es: %d\n", invertido);
18 }
19 int main() {
20     // Llamar a la función invertirNumero
21     invertirNumero();
22     return 0;
23 }
24
25
26

```

Input

```

Ingrese un número entero de 4 dígitos: 1234
El número invertido es: 4321

...Program finished with exit code 0
Press ENTER to exit console.

```

Imagen 1.3 captura de pantalla de OnlineGDB donde muestra el código en C de invertir número.

En la Imagen 1.3 el código en solicita al usuario ingresar un número de 4 dígitos, utilizando la función scanf para capturar la entrada. Primero, verifica si el número tiene exactamente 4 dígitos, comprobando que esté entre 1000 y 9999; si no cumple esta condición, imprime un mensaje de error y termina la ejecución de la función. Luego, mediante un bucle while, invierte el número. En cada iteración, extrae el último dígito del número usando el operador módulo (%), lo agrega a la variable invertido y elimina el último dígito del número dividiéndolo entre 10. Finalmente, muestra el número invertido.

**Conclusión**

Este trabajo me enseñó que aprender a programar no es un proceso rápido ni fácil. Aunque la meta era solo conocer las bases, fue necesario enfrentarme a dificultades y cometer errores para aprender a solucionarlos. La experiencia me demostró que, para desarrollar una comprensión profunda de la programación, es necesario ser constante y enfocado, ya que cada concepto aprendido abre la puerta a un puesto laboral como futuro desarrollador.

La práctica con el lenguaje de programación C facilitó comprender lo importante de aprender los fundamentos de la programación, que sirven para desarrollar una base sólida. Aunque fue un reto al principio, este proceso me permitió conocer cómo los programas interactúan con el hardware y cómo optimizar los recursos del sistema. A medida que avanzaba, tuve que enfrentar varios errores, lo que, aunque desafiante, me ayudó a mejorar mis habilidades y a comprender más a fondo la lógica.

**Referencias**

Francisco Javier Pinales Delgado. (2014). ROBLEMARIO DE ALGORITMOS RESUELTOS CON DIAGRAMAS [Libro] Universidad Autónoma de Aguascalientes.