

Actividad | 2 | Código en Lenguaje C

Introducción al Desarrollo de Software

Ingeniería en Desarrollo de Software



TUTOR: Sandra Luz Lara Dévora

ALUMNO: Cynthia Jeanette García Torres

FECHA: Sábado, 9 de Agosto de 2025

Índice

Introducción.....	3
Descripción	3
Justificación.....	4
Desarrollo	5
Números primos.....	5
Número par e impar	7
Números invertidos	10
Conclusión	12
Referencias	13

Introducción

En esta fase final del trabajo, puse en práctica todo lo que había desarrollado en las actividades anteriores para convertir los algoritmos y diagramas de flujo en programas funcionales escritos en lenguaje C. Esta etapa es clave porque marca el paso de la planeación a la ejecución, lo que me permitió comprobar que la lógica diseñada anteriormente realmente funciona cuando se lleva a un entorno de programación real.

Los programas que realicé fueron tres: uno para identificar números primos, otro para determinar si un número es par o impar y uno más para invertir un número entero de cuatro cifras. Para el desarrollo utilicé el compilador online Programiz, que me facilitó escribir, ejecutar y revisar cada código de forma inmediata. Con este trabajo no solo reforcé mis conocimientos de C, sino que también entendí la importancia de seguir una metodología ordenada para obtener resultados precisos y confiables.

Descripción

El objetivo de esta actividad fue llevar la lógica y estructura de los programas diseñados en las etapas anteriores a un código real y ejecutable.

Para el caso de los números primos, se solicitó al usuario ingresar un número entero y, mediante un ciclo, se verificó si este tenía divisores distintos de 1 y de sí mismo. Según el resultado obtenido, se mostró en pantalla si el número era primo o no.

En el ejercicio de par o impar, se configuró un ciclo que pidió diez números al usuario, utilizando el operador módulo (%) para determinar si cada uno era divisible entre 2, mostrando el resultado correspondiente en cada caso.

Por último, en el algoritmo de números invertidos, se solicitó un número de cuatro cifras y, a través de operaciones de división y módulo, se separaron sus dígitos para reorganizarlos en orden inverso.

En cada situación, el código se probó directamente en Programiz y se guardaron evidencias tanto de su escritura como de su ejecución para comprobar su correcto funcionamiento.

Justificación

Llevar los programas al lenguaje C me permitió confirmar que la planificación es un paso fundamental antes de escribir código. Al tener los algoritmos y diagramas ya definidos, la programación fue más rápida y con menos errores. El lenguaje C es ideal para este tipo de prácticas porque obliga a pensar en cada detalle, desde la sintaxis hasta el flujo lógico del programa. Además, trabajar en Programiz me dio la ventaja de poder ejecutar y ver resultados de forma inmediata, lo que facilita la corrección de errores y la optimización del código.

Este tipo de ejercicios fortalecen las habilidades para resolver problemas, organizar ideas y traducirlas en instrucciones claras para la computadora. En resumen, esta metodología asegura que el trabajo final cumpla con los objetivos planteados y que los programas sean funcionales y fáciles de comprender.

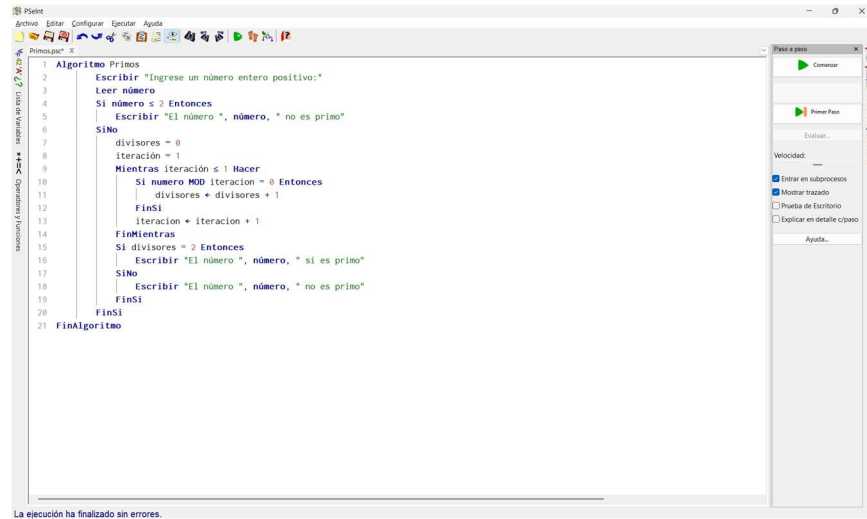
Desarrollo

En esta sección se presentarán los pseudocódigos desarrollados en PSeInt, los diagramas de flujo elaborados en PSDraw y, finalmente, el código en lenguaje C junto con su ejecución en la plataforma Programiz. Cada elemento permite observar el proceso de transformación de la lógica inicial en una solución programada y funcional. A continuación, se explica brevemente el funcionamiento de cada caso.

Números primos

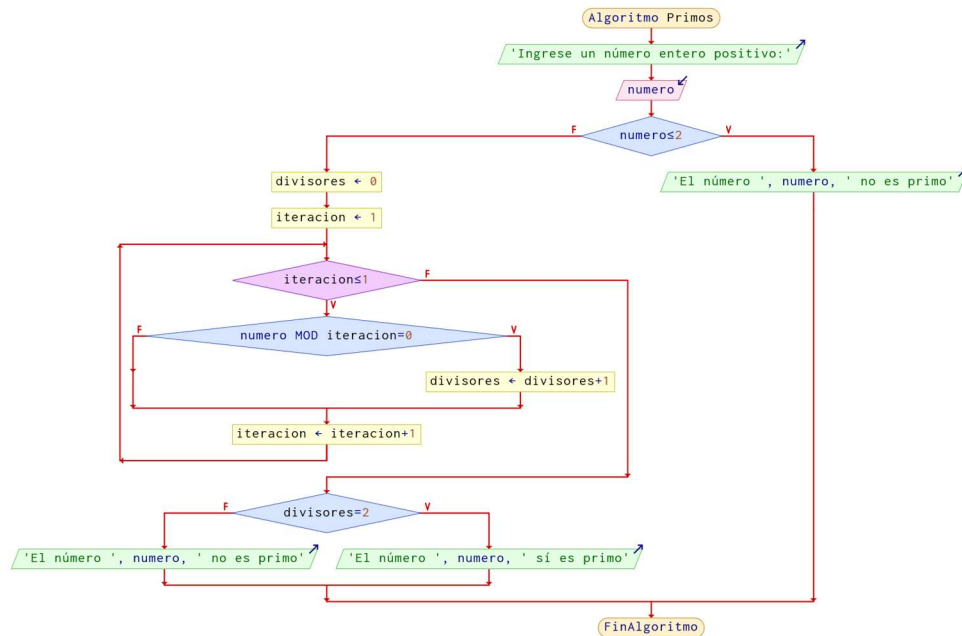
Para el caso de números primos, primero se elaboró el pseudocódigo en PSeInt, solicitando un número entero y utilizando un ciclo para contar sus divisores. Luego, en PSDraw, se representó la misma lógica en un diagrama de flujo que muestra la entrada de datos, el recorrido de posibles divisores y la decisión final. Finalmente, el algoritmo se implementó en C y se probó en Programiz, verificando su correcto funcionamiento con ejemplos y guardando evidencias del código y su ejecución.

Figura 1. *Pseudocódigo del algoritmo de números primos escrito en PSeInt.*



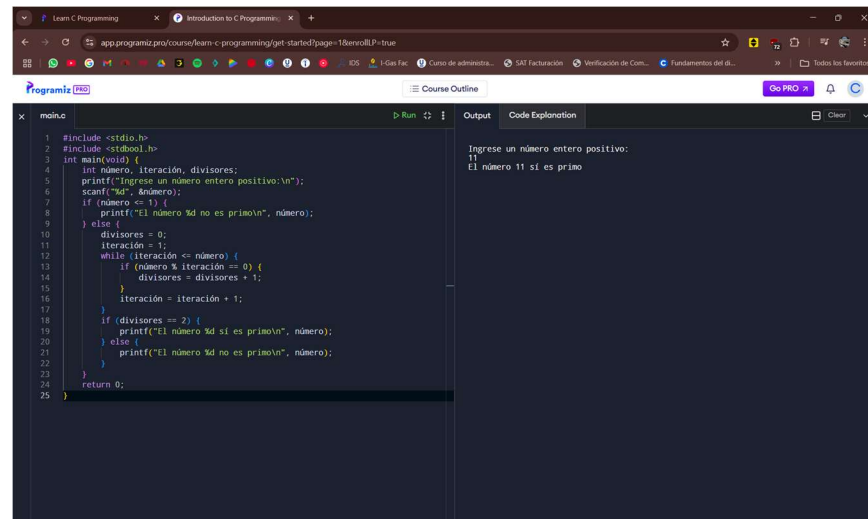
Nota. En el código se observa el uso de estructuras condicionales y un ciclo Mientras para evaluar divisores y determinar si el número es primo o no.

Figura 2. Diagrama de flujo del algoritmo “Primos” elaborado en PSeDraw.



Nota. El diagrama muestra las condiciones y ciclos utilizados para identificar un número primo.

Figura 3. Código en C para determinar si un número es primo.



```
1 #include <stdio.h>
2 #include <stdbool.h>
3 int main(void) {
4     int numero, iteración, divisores;
5     printf("Ingrese un número entero positivo:\n");
6     scanf("%d", &numero);
7     if (numero <= 1) {
8         printf("El número %d no es primo\n", numero);
9     } else {
10        divisores = 0;
11        iteración = 1;
12        while (iteración <= numero) {
13            if (numero % iteración == 0) {
14                divisores = divisores + 1;
15            }
16            iteración = iteración + 1;
17        }
18        if (divisores == 2) {
19            printf("El número %d sí es primo\n", numero);
20        } else {
21            printf("El número %d no es primo\n", numero);
22        }
23    }
24    return 0;
25 }
```

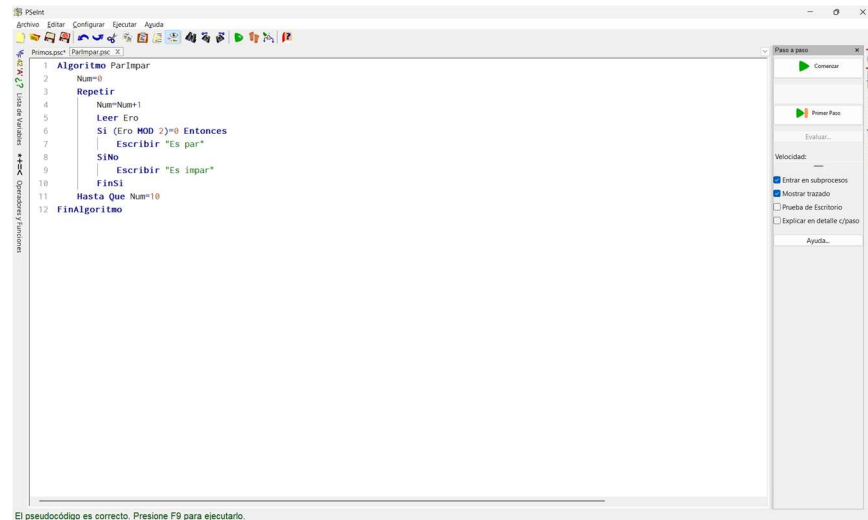
Output: Ingrese un número entero positivo:
11
El número 11 sí es primo

Nota. El programa solicita un número entero positivo, verifica sus divisores y determina si cumple la condición de número primo.

Número par e impar

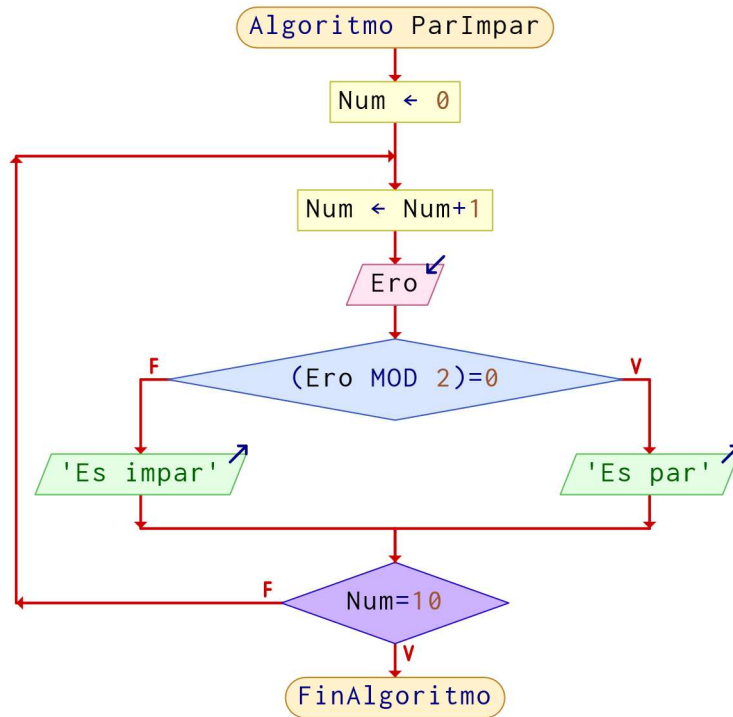
En este caso se desarrolló un programa que solicita al usuario ingresar una serie de números y, mediante el uso del operador módulo (%), determina si cada valor es par o impar. La lógica se implementó en un ciclo que se repite hasta completar la cantidad establecida de entradas, evaluando en cada iteración si el número es divisible entre 2. Dependiendo del resultado, se muestra un mensaje que indica la clasificación correspondiente. El procedimiento se plasmó primero en pseudocódigo, luego se representó en un diagrama de flujo y finalmente se codificó en lenguaje C para ejecutarlo y verificar su correcto funcionamiento.

Figura 4. Pseudocódigo del algoritmo para clasificar un número como par o impar en PSeInt.



Nota. El código utiliza una estructura repetitiva para solicitar diez números y evaluar cada uno mediante el operador módulo.

Figura 5. Diagrama de flujo del algoritmo “ParImpar” elaborado en PSDraw.



Nota. El diagrama ilustra la estructura condicional y el ciclo repetitivo usado para clasificar los números.

Figura 6. Código en lenguaje C para determinar si un número es par o impar y resultado de ejecución en Programiz.

```

1  /*Programa: Número par o impar */
2  #include <stdio.h>
3  int main() {
4      int Num = 0;
5      int Ero;
6      do {
7          Num = Num + 1;
8
9          printf("Ingrese un numero:\n");
10         scanf("%d", &Ero);
11
12         if (Ero % 2 == 0) {
13             printf("Es par\n");
14         } else {
15             printf("Es impar\n");
16         }
17     } while (Num < 10);
18     printf("**** Terminó el ciclo ****\n");
19     return 0;
20 }
  
```

Output:

```

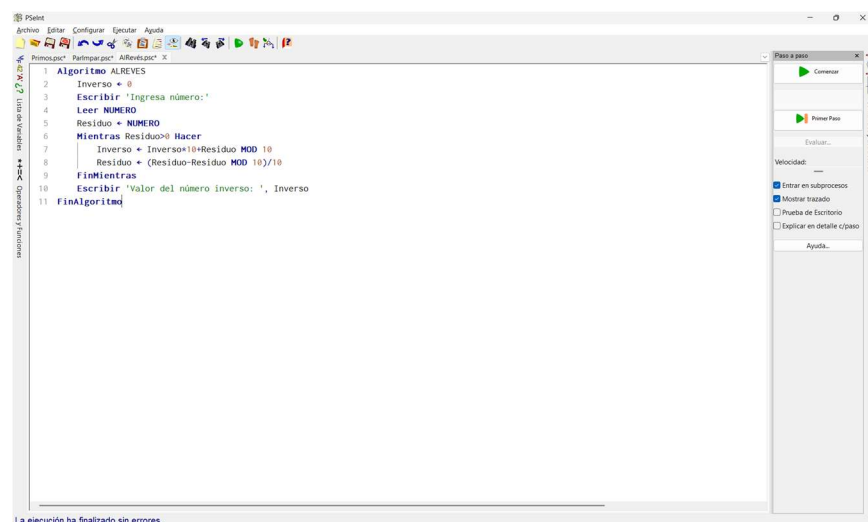
Ingrese un numero:
3
Es impar
Ingrese un numero:
13
Es impar
Ingrese un numero:
25
Es par
Ingrese un numero:
19
Es impar
Ingrese un numero:
26
Es par
Ingrese un numero:
12
Es par
Ingrese un numero:
14
Es par
Ingrese un numero:
10
Es par
Ingrese un numero:
15
Es impar
Ingrese un numero:
25
Es impar
**** Terminó el ciclo ****
  
```

Nota. El programa solicita un número entero, evalúa si es par o impar mediante el operador módulo y repite el proceso hasta alcanzar el límite establecido.

Números invertidos

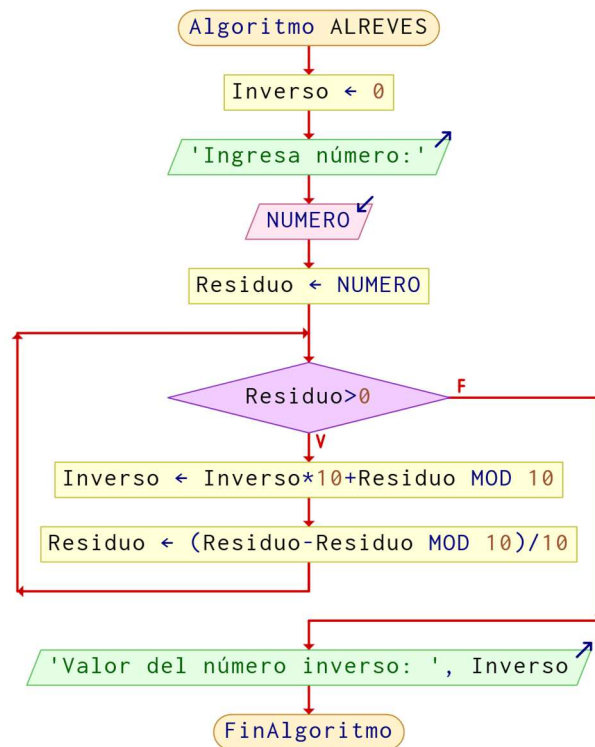
En el caso de los números invertidos, el procedimiento comienza solicitando un número entero al usuario y asignándolo a una variable. A partir de ahí, se establece un ciclo que se ejecuta mientras el número tenga dígitos por procesar. En cada iteración, se obtiene el último dígito mediante la operación módulo, se incorpora al valor acumulado del número inverso multiplicándolo previamente por 10, y posteriormente se elimina ese dígito mediante una división entera. Este proceso continúa hasta que el número original se reduce a cero, momento en el que se muestra en pantalla el valor final invertido. El algoritmo se representa en pseudocódigo, diagrama de flujo y código en C, evidenciando la misma lógica implementada de forma visual y ejecutable.

Figura 7. Pseudocódigo del algoritmo de inversión de números en PSeInt.



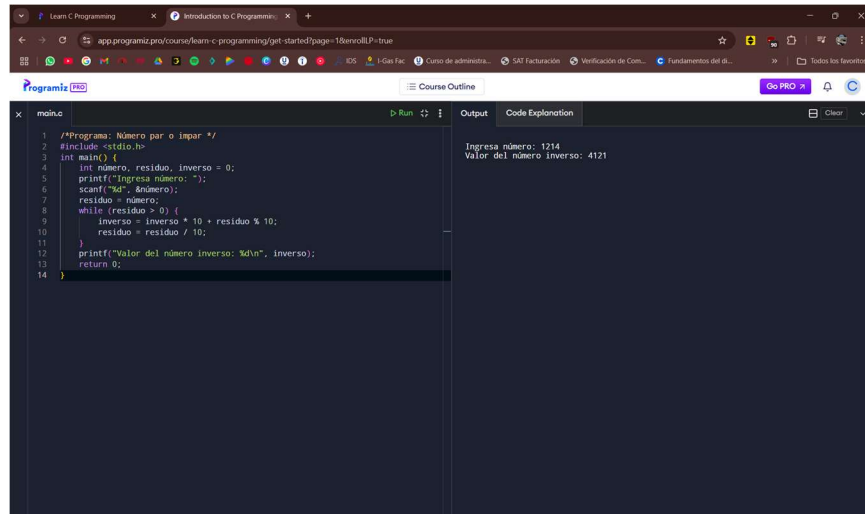
Nota. En el código se observa el uso de un ciclo Mientras para extraer y reposicionar dígitos hasta invertir por completo el número original.

Figura 8. Diagrama de flujo del algoritmo “ALREVES” elaborado en PSDraw.



Nota. El diagrama detalla el uso de un ciclo y operaciones matemáticas para invertir un número entero.

Figura 8. Código en lenguaje C para invertir un número y resultado de ejecución en Programiz.



```
1 //Programa: Número par o impar */
2 #include <stdio.h>
3 int main() {
4     int número, residuo, inverso = 0;
5     printf("Ingres número: ");
6     scanf("%d", &número);
7     residuo = número;
8     while (residuo > 0) {
9         inverso = inverso * 10 + residuo % 10;
10        residuo = residuo / 10;
11    }
12    printf("valor del número inverso: %d\n", inverso);
13    return 0;
14 }
```

Output: Ingres número: 1214
Valor del número inverso: 4121

Nota. El código solicita un número entero, calcula su valor invertido mediante operaciones módulo y división entera, y muestra el resultado.

Conclusión

Finalizar esta actividad fue la oportunidad de cerrar un ciclo de aprendizaje que inició con la creación de algoritmos, continuó con diagramas de flujo y concluyó con la implementación en un lenguaje de programación real. Programar en C me ayudó a reforzar el manejo de estructuras de control, ciclos y condicionales, así como operaciones matemáticas básicas. También confirmé que un buen diseño previo hace que la programación sea más fluida y con menos errores.

La experiencia de ver cómo una idea se transforma en un programa que se ejecuta y devuelve resultados correctos es muy gratificante y demuestra el valor de seguir una metodología clara. Considero que lo aprendido en esta actividad no solo sirve en el ámbito académico, sino que es aplicable a cualquier proyecto en el que se requiera desarrollar soluciones lógicas y eficientes.

Referencias

PSeInt. (2023). PSDraw [Software]. PSeInt Project. <https://pseint.sourceforge.net/>

Programiz. (s. f.). Online C compiler. Programiz. <https://programiz.pro/learn/master-c-programming>

Academia Global-MX. (2025, 31 de julio). Introducción al Desarrollo de Software #3-
Pantalla compartida con vista del orador [Grabación de Zoom].