

Actividad | #3 | Código en Lenguaje C

Introducción al Desarrollo de Software

Ingeniería en Desarrollo de Software



TUTOR: Sandra Luz Lara Dévora

ALUMNO: Alan Giovanni Nieto Becerril

FECHA: 19/04/25

Índice

Introducción **3**

Descripción **4**

Justificación **5**

Desarrollo **6**

Conclusión **9**

Referencias **10**

Introducción

El lenguaje de programación C es un lenguaje estructurado que proporciona las construcciones fundamentales de control de flujo que se requiere en programas, como la agrupación de proposiciones, la toma de decisiones (if-else), la selección de un caso entre un conjunto de ellos (switch), la iteración con la condición de paro en las partes superiores (while, for) o inferiores (do) y la interrupción anticipada de ciclos (break).

Este lenguaje fue desarrollado entre 1969 y 1972 por Brian Kernighan y Dennis Ritchie en una computadora con el sistema operativo UNIX. El desarrollo de este lenguaje se basó en un lenguaje de programación previo, que era el BCPL.

Para utilizar alguna variable en C esta tiene que ser declarada y especificar el tipo de dato que tendrá. Se pueden declarar 5 tipos diferentes de datos con este lenguaje: char, int, float, double y void. También se pueden utilizar operadores aritméticos como el incremento, el decremento, el menos unario, la multiplicación, división, la suma y resta. Y operadores relaciones y lógicos como: not, mayor que, mayor o igual que, menor que, menor o igual que, igual, desigual, and y or.

Descripción

La empresa MathTech requiere que se elaboren tres tipos de calculadoras diferentes para implementar en diversas escuelas. La primera calculadora ‘Primos’ debe identificar si un número ingresado es primo o no. La segunda calculadora ‘Par/Impar’ debe solicitarle al usuario 10 números y determinar para cada número si se trata de un número par o de un número impar. La tercera calculadora ‘Al revés’ debe recibir un número entero de 4 dígitos y regresarlo de manera invertida por medio de operaciones matemáticas.

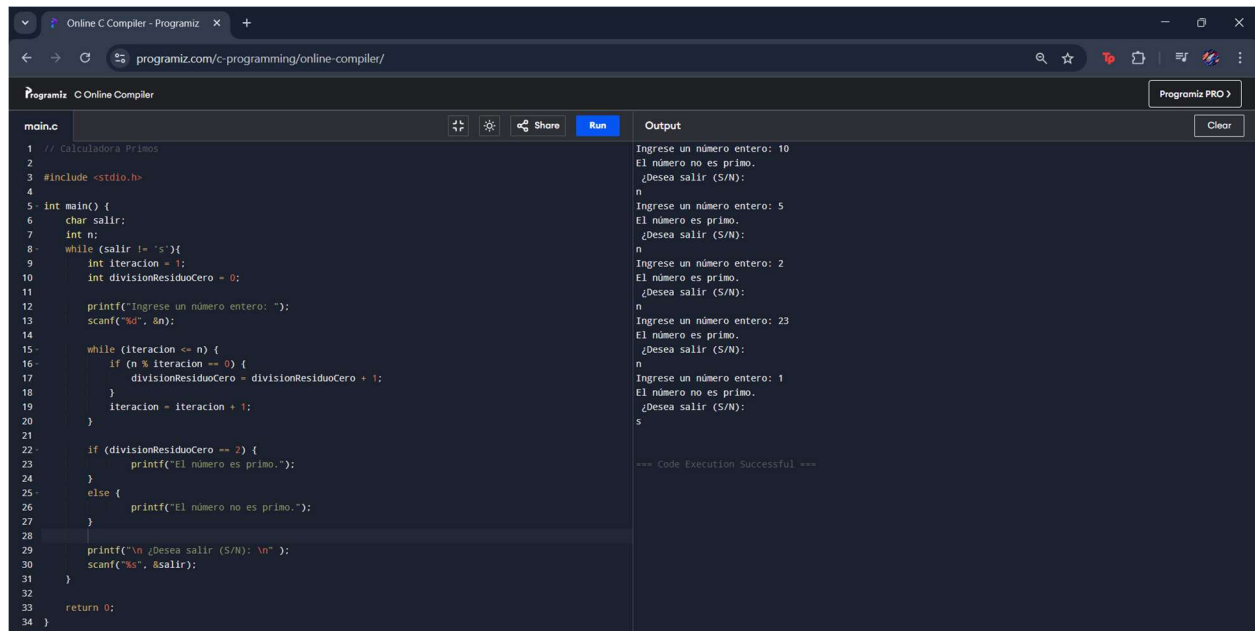
En las actividades anteriores se analizó una de las maneras en las cuales se podría resolver cada uno de los problemas y se elaboró su respectivo algoritmo y diagrama de flujo. En esta actividad el objetivo es desarrollar en lenguaje C, usando un compilador de código online, un programa para cada calculadora. Se mostrará la evidencia con capturas de pantalla indicando cual fue la lógica detrás de esa solución y también se mostrará lo que se ejecute con el compilador.

Justificación

Utilizar el lenguaje C es de suma importancia para nosotros, estudiantes de la Ingeniería en Desarrollo de Software y también para cualquier persona que quiera aprender a programar, debido a que da la esencia de como ocurre una secuencia de pasos en varios programas. Lleva varios años desde que fue desarrollado, por lo que ya cuenta con una comunidad de programadores que lo conocen bien. Es considerado un lenguaje de programación de nivel medio bajo, lo que significa que no requiere de muchos recursos como memoria, pero ejecuta programas de forma rápida y eficaz. Se considera uno de los lenguajes más populares y también es bastante flexible, utiliza varias bibliotecas que contribuyen a los usuarios a tener un mayor número de funciones, facilitando la programación.

En este proyecto final programar las 3 calculadoras utilizando el lenguaje C, me pareció una excelente idea para poder plasmar nuestras ideas que veníamos trabajando desde la elaboración de los algoritmos y los diagramas de flujo, sin tener que aprender de manera exhaustiva la sintaxis de algún otro lenguaje más complejo.

Desarrollo



The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Programiz C Online Compiler". The code editor on the left contains the following C code:

```

1 // Calculadora Primos
2
3 #include <stdio.h>
4
5 int main() {
6     char salir;
7     int n;
8     while (salir != 's'){
9         int iteracion = 1;
10        int divisionResiduoCero = 0;
11
12        printf("Ingrese un número entero: ");
13        scanf("%d", &n);
14
15        while (iteracion <= n) {
16            if (n % iteracion == 0) {
17                divisionResiduoCero = divisionResiduoCero + 1;
18            }
19            iteracion = iteracion + 1;
20        }
21
22        if (divisionResiduoCero == 2) {
23            printf("El número es primo.");
24        }
25        else {
26            printf("El número no es primo.");
27        }
28
29        printf("\n ¿Desea salir (S/N): ");
30        scanf("%s", &salir);
31    }
32
33    return 0;
34 }

```

The output window on the right shows the following text:

```

Ingrese un número entero: 10
El número no es primo.
¿Desea salir (S/N):
n
Ingrese un número entero: 5
El número es primo.
¿Desea salir (S/N):
n
Ingrese un número entero: 2
El número es primo.
¿Desea salir (S/N):
n
Ingrese un número entero: 23
El número es primo.
¿Desea salir (S/N):
n
Ingrese un número entero: 1
El número no es primo.
¿Desea salir (S/N):
s
--- Code Execution Successful ---

```

Figura 1. Calculadora primos.

En esta calculadora observamos del lado izquierdo que se inicializan dos variables antes del ciclo while. Char se usará para correr el programa cada vez que se presione una tecla distinta a 's'. Y n es el número que será ingresado por el usuario. Dentro del ciclo while inicializamos las variables iteracion y divisionResiduoCero que deberán regresar a estos valores cada que se ingrese un nuevo número. La calculadora busca cuantos divisores tiene el número y solo si tiene dos en total, significa que el número es primo. Podemos ver 5 ejemplos diferentes en la derecha, la calculadora se detuvo al presionar la tecla 's'.

The screenshot shows the Programiz C Online Compiler interface. On the left, the code editor displays a C program named 'main.c'. The code includes `<stdio.h>` and defines a `main` function. It initializes `int num = 0;` and `int numero;`. A `while (num != 10)` loop is used to process 10 numbers. Inside the loop, `printf` prompts the user to enter a number, `scanf` reads the input into `numero`, and an `if (numero % 2 == 0)` condition checks if the number is even. If even, it prints 'El número es par.', otherwise, it prints 'El número es impar.'. After each check, `num = num + 1;` increments the counter. After the loop, a final `printf` message states '***Ingresaste 10 números, terminó el ciclo.***'. The right side of the interface shows the 'Output' window, which displays the program's execution results, including 10 prompts and responses, and a final success message: '***Ingresaste 10 números, terminó el ciclo.***'. The status at the bottom indicates 'Code Execution Successful'.

```

1 #include <stdio.h>
2
3 int main() {
4     int num = 0;
5     int numero;
6
7     while (num != 10) {
8         printf("Ingrese un número: ");
9         scanf("%d", &numero);
10
11         if (numero % 2 == 0) {
12             printf("El número es par. \n");
13         } else {
14             printf("El número es impar. \n");
15         }
16         num = num + 1;
17     }
18
19     printf("***Ingresaste 10 números, terminó el ciclo.***");
20
21     return 0;
22 }

```

Output:

```

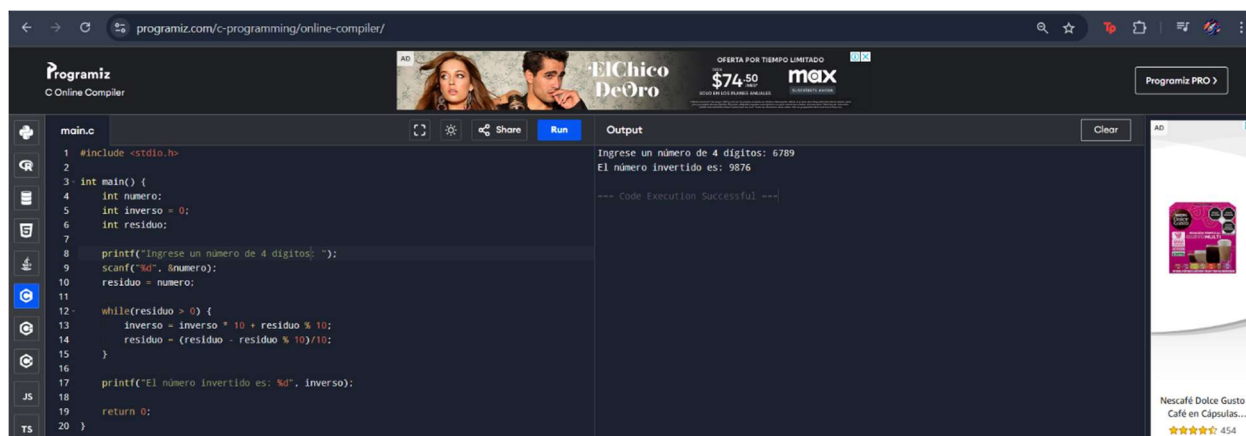
Ingrese un número: 1
El número es impar.
Ingrese un número: 2
El número es par.
Ingrese un número: 4
El número es par.
Ingrese un número: 5
El número es impar.
Ingrese un número: 10
El número es par.
Ingrese un número: 12
El número es par.
Ingrese un número: 99
El número es impar.
Ingrese un número: 1234
El número es par.
Ingrese un número: 11
El número es impar.
Ingrese un número: 777
El número es impar.
***Ingresaste 10 números, terminó el ciclo.***

```

=== Code Execution Successful ===

Figura 2. Calculadora par/impar.

En la segunda calculadora, del lado izquierdo solo inicializamos dos variables, `num` que valdrá 0 y es un iterador, y `numero`, que será el número ingresado por el usuario. La condición del programa se evalúa con un ciclo `while` y se ejecuta solo cuando `num` sea diferente a 10. Cuando corre este bucle, se solicita y se escanea el número ingresado por el usuario. Si el residuo de este número dividido por dos es 0, significa que es par. Caso contrario, es impar. Después, se suma una unidad a `num`, para que siga corriendo el ciclo `while` hasta llegar a 10. Del lado derecho observamos los 10 números que se solicitan al ejecutar la calculadora.



The screenshot shows the Programiz C Online Compiler interface. The code editor on the left contains a C program for reversing a number. The output window on the right shows the program's execution results.

```
1 #include <stdio.h>
2
3 int main() {
4     int numero;
5     int inverso = 0;
6     int residuo;
7
8     printf("Ingrese un número de 4 dígitos: ");
9     scanf("%d", &numero);
10    residuo = numero;
11
12    while(residuo > 0) {
13        inverso = inverso * 10 + residuo % 10;
14        residuo = (residuo - residuo % 10) / 10;
15    }
16
17    printf("El número invertido es: %d", inverso);
18
19    return 0;
20 }
```

Output

Ingrese un número de 4 dígitos: 6789
El número invertido es: 9876

--- Code Execution Successful ---

Figura 3. Calculadora al revés.

En esta última calculadora, inicializamos 3 variables: numero, inverso que vale 0 y residuo. Después le solicitamos al usuario ingresar un número, que se escanea y almacena en la variable numero. En este punto el valor de residuo es igual a numero. Posteriormente, comienza el bucle while siempre y cuando el residuo sea mayor a 0. La variable inverso empieza a acumular los valores de residuo y residuo queda vacía. Finalmente, se le regresa de forma invertida al usuario, el número que ingresó, como vemos del lado derecho.

Conclusión

En este proyecto final se muestra como fue el desarrollo de 3 calculadoras que resuelven distintos problemas matemáticos, a través del uso del lenguaje de programación C. En realidad, este proyecto es el resultado final de varias actividades que un programador debe seguir para poder codificar de manera correcta y eficiente. Es una muy buena práctica tomarse un tiempo para razonar como se podría resolver un problema (puede haber más de una solución), después se puede poner por escrito estas ideas para formar un algoritmo, detallando lo más posible cada uno de los pasos. También se pueden representar estas ideas a través de los diagramas de flujo. O utilizar ambas herramientas. Se le puede mostrar estas representaciones a otra persona o a nuestro equipo de colaboradores para ver si realmente se entiende la manera en que se planea resolver el problema e identificar algún error o pasos faltantes. Tras haber realizado esta serie de acciones, podemos tener más confianza al momento de codificar.

Referencias

- Kernighan, B. & Ritchie, D. (1991). El lenguaje de programación C. México: Pearson Educación
- Bonet, E. (s.f.). Lenguaje C. Universidad Veracruzana.
<https://informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf>

Liga Github: <https://github.com/AlanGNB/IDS>