Práctica 5

Hector Javier Salazar Alvarez

Modificaciones realizadas al código original.

En el texto original, hice varias correcciones importantes para mejorar el funcionamiento y la claridad del código. Primero, en la función “factorial”, eliminé la variable “number”, ya que no se utilizaba..

Luego, en la función “printPascalTriangle”, realicé ajustes en cómo se imprimen los coeficientes del triángulo de Pascal. Originalmente, el cálculo de los coeficientes usaba la operación “factorial(i)/factorial(j)*factorial(i-j)”, pero esto no estaba correctamente agrupado debido a la falta de paréntesis. Esto podía llevar a resultados incorrectos en la división y multiplicación. Para corregirlo, añadí paréntesis alrededor de “factorial(j) * factorial(i-j)”, de modo que la operación se realice correctamente como “factorial(i) / (factorial(j) * factorial(i-j))”. Esto asegura que primero se multipliquen “factorial(j)” y “factorial(i-j)”, y luego se divida “factorial(i)” por el resultado, lo que produce el valor correcto para cada coeficiente.

Además, cambié el especificador de formato en “printf” de “%d” a “%llu” para que se ajuste al tipo “unsigned long long int” que retorna la función “factorial”. También ajusté la impresión, reemplazando la coma con una tab y eliminando el símbolo “:”, y en su lugar coloqué un salto de línea “\n” para que el triángulo de Pascal se muestre de manera más clara y legible.

En la función “main”, corregí el uso de “scanf” para leer el valor de “n_rows”. Cambié el especificador de formato de “%d” a “%u” para manejar correctamente el tipo “unsigned int” y arreglé el paso de la dirección de “n_rows” usando “&n_rows” en lugar de simplemente “n_rows”. También modifiqué la llamada a “printPascalTriangle”, reemplazando el valor fijo “10” por “n_rows”, permitiendo que el número de filas del triángulo de Pascal sea dinámico según la entrada del usuario.

Finalmente, añadí un mensaje para el usuario solicitando el número de columnas, lo que hace que el programa sea más amigable y claro sobre su propósito.

En la línea 17 del código original, el bucle “for” calcula el factorial de un número entero no negativo “n”. El proceso es el siguiente:

El bucle empieza con el valor inicial de “n” y se repite mientras “n” sea mayor que 0. En cada repetición, el valor actual de “n” se multiplica con la variable “product”, que se inicializa en 1 antes de comenzar el bucle. Así, en cada paso del bucle, “product” acumula el producto de “n” con el valor decreciente de “n”. Después de cada iteración, “n” se reduce en 1. Cuando “n” llega a 0, el bucle termina. Al final, la función devuelve el valor de “product”, que es el resultado del cálculo del factorial de “n”, es decir, el producto de todos los números enteros desde “n” hasta 1.

En la corrección de identación del código, se realizaron ajustes para mejorar la claridad y la estructura del mismo. En la función factorial, se añadieron llaves {} alrededor del cuerpo del bucle for para asegurar que el bloque de código se ejecute correctamente. Esto ayuda a delimitar claramente el alcance del bucle, que ahora está indentado de manera que se refleja adecuadamente el cuerpo del bucle. En la función printPascalTriangle, el bucle interno for también fue indentado dentro del bucle externo, con una tabulación adicional para separar visualmente el código del bucle interno del externo. Esto facilita la comprensión de que el código dentro del bucle interno pertenece a este bucle, mejorando así la legibilidad del código. Finalmente, en la función main, se alineará el código de manera que las llamadas a printf, scanf, y printPascalTriangle estén correctamente indentadas dentro del bloque main, y se añadieron comentarios explicativos para cada línea de código. Estos ajustes aseguran que el código no solo sea más fácil de leer, sino que también mantenga una estructura lógica y ordenada.