Análisis del Código C++ para la Comparación de Sumas de Dígitos

Víctor Manuel Rojas Trejos

September 11, 2024

1 Introducción

En este documento se presenta un código en C++ diseñado para verificar si las sumas de las mitades de los dígitos de un número entero son iguales. A continuación, se muestra el código junto con su explicación detallada.

2 Código en C++

El siguiente código está compuesto por tres funciones principales: `IntToVector`, `CompareSums`, y `main`.

2.1 Función `IntToVector`

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Funcion para convertir el numero en vector
vector<int> IntToVector(int n) {
    vector<int> digits;

    while (n > 0) {
        digits.push_back(n % 10);
        n /= 10;
    }
```

```
13
14      return digits;
15  }// end IntToVector
```

Listing 1: Función para convertir un número entero en un vector de dígitos.

Explicación: La función `IntToVector` toma un número entero `n` como argumento y lo convierte en un vector de dígitos. En el cuerpo de la función:

- Se declara un vector `digits` para almacenar los dígitos del número.
- Se utiliza un bucle `while` para dividir el número por 10 repetidamente y obtener el dígito más bajo (`n % 10`).
- Cada dígito extraído se añade al vector `digits`.
- Finalmente, el número se reduce en cada iteración dividiéndolo por 10 (`n /= 10`).
- La función retorna el vector de dígitos.

2.2 Función `CompareSums`

```
1  // Funcion que compara si las sumas de las mitades del numero son
       iguales
2  void CompareSums(vector<int>& digits) {
3      int sum1 = 0, sum2 = 0;
4
5      for (int i = 0; i < digits.size()/2; ++i) {
6          sum1 += digits[i];
7          sum2 += digits[i + digits.size()/2];
8      }
9
10     if (sum1 == sum2) {
11         cout << "result: " << sum1 << " " << 1 << endl;
12     } else {
13         cout << "result: " << sum1 << " " << 0 << endl;
14     }
15 }// end CompareSums
```

Listing 2: Función para comparar las sumas de las mitades del vector de dígitos.

Explicación: La función `CompareSums` recibe un vector de dígitos como argumento y compara las sumas de las dos mitades del vector:

- Se inicializan dos variables `sum1` y `sum2` para almacenar las sumas de las dos mitades del vector.
- Un bucle `for` recorre la primera mitad del vector, sumando los dígitos a `sum1` y a la segunda mitad a `sum2`.
- Luego, se compara `sum1` y `sum2`. Si son iguales, se imprime el valor de `sum1` y el número 1; de lo contrario, se imprime `sum1` y el número 0.

2.3 Función `main`

```
int main() {
    int number;
    cout << "Este programa verifica si las sumas de los digitos de
   un numero entero son iguales." << endl;
    cout << "Digita un numero entero: ";
    cin >> number;

    if (number > 0){
        vector<int> digitVector = IntToVector(number);
        if ((digitVector.size() % 2) != 0) {
            cout << "El numero de digitos del numero debe ser par."
     << endl;
        }
        else{
            CompareSums(digitVector);
            }
    }
    else{
        cout << "Debe ser un entero positivo." << endl;
    }
    return 0;
}
```

Listing 3: Función principal que gestiona la entrada y salida del programa.

Explicación: La función `main` gestiona la entrada del usuario y llama a las funciones anteriores:

- Se solicita al usuario que ingrese un número entero.
- Si el número ingresado es positivo, se convierte en un vector de dígitos utilizando la función `IntToVector`.
- Se verifica si el número de dígitos es par. Si es impar, se informa al usuario que el número debe tener un número par de dígitos.
- Si el número de dígitos es par, se llama a la función `CompareSums` para comparar las sumas.
- Si el número ingresado no es positivo, se muestra un mensaje de error.

3 Conclusión

Este código C++ proporciona una manera sencilla de verificar si las sumas de las mitades de los dígitos de un número entero son iguales. La conversión del número en un vector de dígitos y la comparación de las sumas de las mitades son realizadas mediante funciones claramente definidas, lo que permite una comprensión y mantenimiento más fácil del código.