Práctica 1. Programa para calcular sumas consecutivas y cambios de base

1st José Luis Aguilar
Departamento de aprender a programar
Universidad MdG
Ciudad de México, México
aguilarch.joseluis@gmail.com

***Resumen*—Este documento presenta la documentación de un programa en C que pueda calcular las sumas consecutivas dado un número, y también una herramienta para el cambio de base decimal a cualquier otra. Se muestran además los procesos que sigue el programa y las herramientas de diseño usadas para la solución.**

***Index Terms*—programación, Clang, introducción, cambio de base, sumas de números consecutivos.**

I. Descripción del problema

Se requiere de un programa que tenga la capacidad de llevar a cabo tres funciones manejadas a través de un menú:

1. Recibir un número entero positivo del usuario y encontrar todas las series de 3 o más números consecutivos que sumados den como resultado este valor.
2. Recibir un número entero positivo del usuario, convertirlo a 3 bases (de igual manera leídas por el usuario) entre base 2 y 16.
3. Salir del programa.

Además, antes de mostrar el menú, se muestra el nombre del creador, título del programa y fecha.

II. Comportamiento detallado

III. Procesos

A continuación se muestra el proceso detallado del programa, separado en pasos para facilitar su lectura:

-

IV. Alcances y limitaciones

El programa puede:

- Convertir de base decimal a cualquier base numérica entre 2 y 16.
- Mostrar todas las series de 3 o más sumas de números consecutivos.
- Valida que los valores introducidos sean números enteros positivos.
- Valida que no puedan ingresarse opciones inexistentes.

Las limitaciones del programa son las siguientes:

- Solamente puede tomar números enteros positivos, no convierte otros tipos de datos.
- No utiliza arreglos de datos, ni recursión, ni variables globales.
- Si el programa detecta un valor válido, regresa al menú y no directamente a la función que estaba usándose.

V. Diseño de pantalla

El programa muestra una descripción al inicio, con la siguiente forma[1]: @default

```
------------------------------------------

Este programa fue desarrollado por Jose
Luis Aguilar el 23/12/2020.
Presione [Enter] para continuar.

------------------------------------------
```

Que después menciona que se muestra un menú. El menú se ve de la forma siguiente: @default

```
------------------------------------------
                Menu
               ------
Escoge una opcion escribiendo el numero
de la accion que desees tomar:

    1. Cambio de base a un numero
    2. Calcular sumas consecutivas de un
numero
    3. Salir
------------------------------------------
```

Cada opción muestra la acción que toma. A continuación se muestran en orden las distintas opciones: @default

```
Introduzca el numero entero positivo a
convertir: 10423
Introduzca la base (entre 2 y 16) a la
que desea convertir: 12
El numero en base 12 es: 6047
Presione [Enter] para continuar.
```

Cada distinta opción elegida regresa nuevamente al menú. @default

```
Introduzca el numero entero positivo a
mostrar sus sumas: 3456
3 sumas consecutivas:
1151 + 1152 + 1153
```

[1]Se omiten los acentos en las pantallas del programa aquí descritas, pero en el ejecutable sí se observan.

```
9 sumas consecutivas:
380 + 381 + 382 + 383 + 384 + 385 + 386
+ 387 + 388
27 sumas consecutivas:
115 + 116 + 117 + 118 + 119 + 120 + 121
+ 122 + 123 + 124 + 125 + 126 + 127 +
128 + 129 + 130 + 131 + 132 + 133 + 134
+ 135 + 136 + 137 + 138 + 139 + 140 + 141
Presione [Enter] para continuar
```

Por último, la opción de salir del programa pide al usuario que confirme su decisión: @default

```
------------------------------------------


                Desea salir?

               1 = si, 0 = no.


------------------------------------------
```

Debido al tamaño usual de los displays en la terminal de 80 caracteres, el diseño del menú y las descripciones se hizo de 80 caracteres de ancho.

VI. Diseño de solución

Antes de mostrar el pseudocódigo y los algoritmos utilizados en el desarrollo del programa, conviene comprender el fundamento matemático sobre el cual se basan ambas soluciones. A continuación se muestra el fundamento de cada uno, donde se mostrará un pequeño pseudocódigo correspondiente al final de cada algoritmo.

VI-A. Cambio de base

Una base numérica se basa en la notación dada por la siguiente ecuación:

$$X_b = \sum_{n=-\infty}^{d} a_n b^n$$

Donde X_b es el número escrito en la base b, a_n es el múltiplo de cada una de las potencias de la base b, donde suele representarse con un símbolo comprendido en el conjunto $a_n \in \{0, ..., b-1\}$[2], d es el dígito correspondiente. Si se asume que solamente se trabaja con enteros, entonces se reduce a la siguiente expresión:

$$X_b = \sum_{n=0}^{d} a_n b^n \tag{1}$$

Por lo que para representar al número, se requieren $n+1$ dígitos. El número de dígitos puede encontrarse usando la siguiente propiedad de los logaritmos:

$$\log_b(x) = \frac{\log(x)}{\log(b)}$$

[2]Nos remitimos solamente a bases enteras, ya que aunque existen aquellas que no son enteras, suelen no ser utilizadas fuera de ámbitos de investigación matemática.

Entonces, considerando que $\lfloor \log_b(x) \rfloor$ es el número de dígitos necesarios para expresar a x en la base b, puede encontrarse el valor de cada uno con un sencillo algoritmo descrito en el siguiente pseudocódigo:

```
cambio de base (int numero, int base)
    ValidarEnteroPositivo(numero);
    ValidarEnteroPositivo(base);
    int digito = log(numero)/log(base);
    int multiplo = 0;
    int residuo = 0;

    while(numero>0)
        if(residuo>=numero)
            residuo=residuo +
power(base,digito);
            multiplo=multiplo+1;
        else
            // Se resta porque al contar
existe un overshoot.
            residuo = residuo -
power(base,digito);
            multiplo = multiplo-1;
            digito = digito-1;
            numero = numero-residuo;
            PRINT("_entero_",multiplo);
            residuo=0;
            numero=0;
        FIN
    FIN
    // Para considerar los que son
multiplos de la base:
    for(entero i = 0, i<=digito,i++)
        print("0")
    FIN
```

Esto muestra un algoritmo relativamente sencillo, y la implementación en el programa es muy similar, solamente con la diferencia que considera también las entradas de números y formatos para hacerlo más legible.

VI-B. Sumas consecutivas

Para poder considerar las sumas consecutivas, se puede hacer un algoritmo relativamente sencillo. Consideremos primero la suma de números enteros positivos consecutivos más grande posible:

$$X = \sum_{n=1}^{N} n$$

Se omite el cero porque si se considera, entonces cualquier serie pudiera ser más grande. Sin embargo, esta serie de números forzosamente es la más grande porque considera a todos los enésimos enteros. Esto impone un límite superior sobre las iteraciones que puede realizar el programa, que puede expresarse de una forma más sencilla, descrita en la siguiente ecuación[3]:

$$X = \frac{n(n+1)}{2} \tag{2}$$

Esta suma considera todos los enteros. Sin embargo, solamente son de interés en este programa las sumas que tengan tres o más enteros. Además, pueden considerarse dos casos distintos; cuando el número de términos consecutivos sea un número par, y cuando sea impar. Hecha esta consideración, entonces se pueden considerar los números de dos en dos casos, y se sigue respetando el límite superior. Haciendo el cambio de variable $n = 2k + 1$ en (2):

$$X = \frac{(2k+1)(2k+2)}{2} = (2k+1)(k+1) \tag{3}$$

Y esto es el límite superior tomando las sumas consecutivas de dos en dos. Entonces vale la pena considerar en qué casos existen estas sumas consecutivas. Esto ocurre para un número X con un número de sumas consecutivas n bajo los siguientes casos:

$$\begin{cases} n \text{ par} & X \bmod n \equiv \frac{n}{2} \\ n \text{ impar} & X \bmod n \equiv 0 \end{cases}$$

Estas condiciones son relativamente sencillas de encontrar, y se cumplen siempre. Suponiendo primero que el número de términos sea impar, entonces puede pensarse en el número X dividido entre el número de términos n, que forzosamente va a ser el término medio de las sumas consecutivas. Entonces la suma de n términos puede ponerse de la siguiente forma:

$$X = \frac{1}{n}\left[(X - \tfrac{n-1}{2}) + \cdots + X + \cdots + (X + \tfrac{n-1}{2})\right]$$

Por el otro lado, si número de términos es par, y solamente se trabaja con enteros, entonces pueden tomarse solamente los cocientes enteros, y como la siguiente relación se cumple cuando el residuo es distinto a cero: $\left\lfloor \frac{X}{n} \right\rfloor < \frac{X}{n}$ Entonces también puede medirse todo con respecto a un dígito (en este caso, el número $\frac{n}{2}$). Siguiendo este criterio, entonces puede escribirse un número de la siguiente forma:

$$X = \sum_{i=1}^{n} \left(\left\lfloor \frac{X}{n} \right\rfloor - \frac{n}{2} + i + 1 \right)$$

Tomando en cuenta estas dos relaciones, entonces puede hacerse un sencillo algoritmo para poder calcular las sumas consecutivas dependiendo si el número es par o no. A continuación se muestra el pseudocódigo:

```
void SumasConsecutivas(int *X)
    n = 1;
    while((2*n+1)*(n+1)<= *X)
        SumasImpares(*X, 2*n+1);
        SumasPares(*X, 2*(n+1));
        n++;
    FIN

void SumasImpares(int X, int N)
    if(X % N == 0) // % es la operacion modulo.

        int offset = (N-1)/2;
        int ValorInicial = X/N-offset;
        int sumas = ValorInicial;

        /* El primer termino esta fuera del loop,
        con el proposito de que al imprimir los
        valores de la suma, no salga un termino de mas. */

        print("_num_ sumas consecutivas: _num_ ", N, sumas);

        for(int i = 1, i < N, i++)
            sumas = ValorInicial+i;
            printf("+ _num_ ", sumas);
        FIN
        print("_newline_");
    FIN

void SumasPares(entero X, entero N)
    if(X %N==N/2 AND X>=N*(N+1)/2)

        int offset = N/2-1;
        int ValorInicial = X/N - offset;
        int sumas = ValorInicial;
        print("_num_ sumas consecutivas: _num_ ", N, sumas);
        for(int i = 1, i < N, i++)
            sumas = ValorInicial+i;
            printf("+ _num_ ", sumas);
        FIN
        printf("_newline_");
    FIN
```

[3]En los anexos se hace una breve demostración por si el lector no se encuentra familiarizado.