

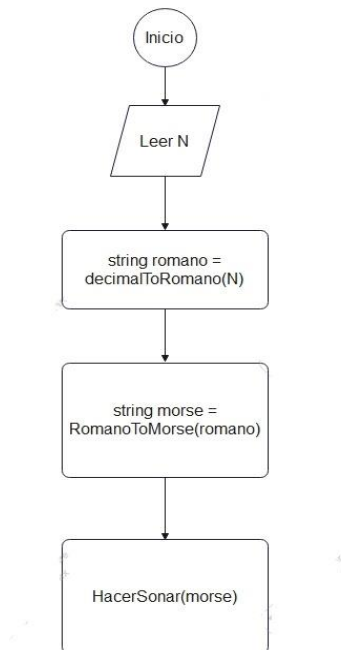
My amazing Project

1. Descripción del problema:

El problema se origina como el primer proyecto del portafolio del curso fundamentos de informática, para el primer ciclo 2022. El problema se divide en 4 partes. Para empezar, se debe de recibir por teclado un número en base decimal (del 1 al 3000), en segundo lugar, el número recibido se convierte en un número romano y se muestra el resultado en pantalla, posteriormente, este número romano se convierte en código Morse y para finalizar este se debe de hacer sonar.

2. Análisis de la solución del problema:

Para solucionar el problema se llevó a cabo una serie de procesos, los cuales están basados en el siguiente diagrama:



Para pasar de un número decimal a romano pensé en crear un tipo de diccionario con 2 arrays, pero debido a que en este trabajo no se puede hacer uso de arrays decidí hacerlo mediante el uso del condicional “IF” evaluando los principales valores (1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1) y ("M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I").

Hacer la conversión de romano a Morse fue tan sencillo cómo recorrer un String letra por letra y dependiendo de la letra agregaba el valor de esa letra en código Morse a una variable llamada “morse”.

Para hacer sonar el código Morse, simplemente debía de averiguar si era un punto o un guion (. o -), en caso de que sea un punto suena por un periodo más corto (200ms) y si es un guion suena por un periodo de tiempo mas extenso (el doble para ser precisos). Tuve que realizar una pequeña investigación con el fin de encontrar una forma de hacer sonar mi código.

3. Descripción de los aspectos más importante a utilizar:

Variables:

1. *inputNumero*: Se encarga de recibir el número entero introducido por el usuario.
2. *romano*: Almacena una cadena de texto, el valor final de esta variable es la conversión del valor de *inputNumero* en un número romano.
3. *morse*: Almacena una cadena de texto, el valor final de esta variable es la conversión del valor almacenado en la variable *romano* en clave Morse.

Funciones:

1. *decimalToRomano(N)*: Esta función recibe como parámetro un número entero N, con el fin de convertir este número (decimal) en un número romano, para cumplir este objetivo se hace uso del condicional IF y ELSE IF, estos condicionales se encargan de comprobar si el número (decimal) es mayor a ciertos números de referencia (1000, 900, 500, entre otros), en caso de que sea mayor se agrega el valor de este número, pero en símbolos romanos, ejemplo, si N = 1100, se evalúa si N es mayor a 1000, en este caso es cierto, por ende, se agregaría la M (1000 en números romanos), luego a 1100 se le resta 1000 y se repite el proceso. Al final de la función se devuelve la variable que almacena el número N en números romanos.
2. *romanoToMorse(X)*: Esta función recibe como parámetro el retorno de la función *decimalToRomano()*, con el objetivo de convertir el número romano en código Morse, para conseguir esto se hace uso de un WHILE, este WHILE se encarga de recorrer cada una de las letras del número romano, luego dentro de este WHILE con el uso del condicional IF y ELSE IF, se evalúan las posibles letras romanas, y se agrega la clave Morse correspondiente. Esta función retorna la variable que almacena el código Morse.
3. *hacerloSonar(M)*: Esta función recibe como parámetro el retorno de la función *romanoToMorse()*, y se encarga de hacer sonar el código Morse. Para hacer sonar el código Morse se hace uso de un WHILE, este WHILE se encarga de recorrer cada uno de los símbolos del código Morse, luego dentro de este WHILE con el uso del condicional IF y ELSE IF, se evalúa si el símbolo es un guión “-” o un punto “.”, luego con el uso de la función *Beep()* se hace sonar el código.

4. *Beep(F, D)*: Esta función se encarga de emitir un sonido, recibe 2 parámetros, F y D, donde F es la frecuencia del sonido y D la duración de este en milisegundos. Para hacer uso de la función *Beep()* se debe de incluir la biblioteca “windows.h”.

4. Descripción de datos de prueba:

El programa se probó con una gran variedad de números, desde números negativos hasta números mayores a 3000, en los casos en el que el número es mayor a 3000 o menor a 1 el programa imprime en consola “El número no está en el rango, inténtelo de nuevo” y le da la posibilidad al usuario de volver a introducir un número válido.

Cuando se introduce una letra seguida de un número el programa funciona correctamente, ejemplo: 544L, en casos como este el programa simplemente ignora la letra del final.

5. Limitaciones del programa:

En los casos en el que el usuario introduce un carácter o una cadena de texto, el programa NO funciona, entra en un bucle infinito.

6. Observaciones generales:

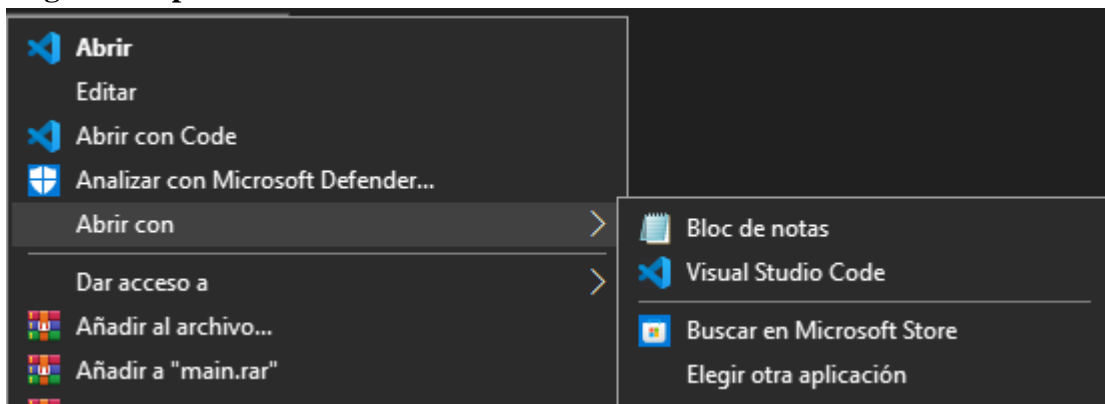
Al inicio del programa se incluye la biblioteca *windows.h*, con el fin de poder utilizar la función *Beep()*, función encargada de emitir sonidos.

7. Manual del usuario:

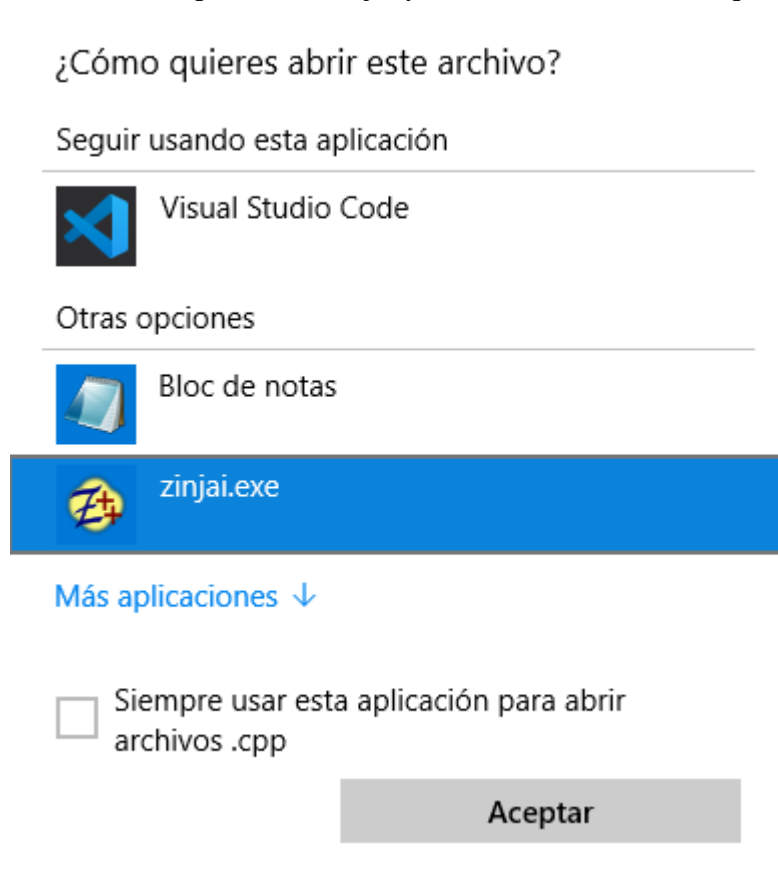
Descargar e instalar [Zinjai](#) o su entorno de desarrollo preferido, en este caso trabajaremos con [Zinjai](#).


Descargar el archivo [main.cpp](#).

Buscamos el archivo recientemente descargado, le damos **click derecho >> abrir con >> elegir otra aplicación**.



Buscamos la aplicación Zinjal y le damos doble click izquierdo.



Dentro de la aplicación Zinjal ejecutamos nuestro código presionando F9 o con el botón de Save, Compile and Run .

Una vez se haya ejecutado nuestra aplicación debemos introducir un número entero entre 1 y 3000.