Tarea Promoción: Número Astronómico
Aspectos tecnicos

Integrantes:

- Tomás Berrojalviz Aguirre
- Tomás Carballo.

Materia: Sintaxis y semántica de los Lenguajes

Curso: K2055

Profesora: Ing. Roxana Leituz

Jefe de Cátedra: Dr. Oscar Bruno

Comenzamos este trabajo intentando visualizar para que podríamos usar las funciones requeridas, a decir verdad el cómo modelar y usar las mismas fue lo que más pensamos. Decidimos hacer un menu en el cual se podrían usar las funciones para crear números astronómicos, realizar las operaciones requeridas y poder leer los archivos que se vayan creando. Hicimos las funciones requeridas y las fuimos agregando a este menú.
Al ejecutar el programa se procede a la eliminación de cualquier archivo creado anteriormente, para esto previamente se crea un archivo en donde guardamos la cantidad de estos que hay disponible, originalmente inicializada en 0.
Si ya hay archivos, se restablece la cantidad de archivos devuelta a 0 y se los borrara.

El número astronómico está representado como una estructura que contiene una variable para almacenar la longitud o en caso de que haya error, su respectivo código, y otra variable donde se almacena la cadena de caracteres junto con sus flags al comienzo, uno para el overflow y otro para el carry respectivamente.

1. **Operaciones de Creación**
    a. **Crear desde cadena**

La misma recibe una cadena que será el entero del número astronómico. Tomamos la longitud de la cadena y la agregamos en la variable que contiene el largo de la cadena o el código de error.

Reservamos espacio en memoria para la cadena y además para los 2 flags y el centinela ('\0').En la variable destinada para la cadena primero incorporamos 2 ceros para los flags mencionados y luego lo concatenamos la cadena ingresada.

Después utilizamos la función EstablecerErrores que se encarga de analizar el número astronómico y asignarle sus errores si los tuviese o en el flag o en su código de error. Para ver cómo tratar los errores se verá más adelante en profundidad en el ítem "2. Operaciones de Manejo de Errores" . Por último se archiva si el usuario lo desea y se libera la memoria asignada a la cadena que ingresó al comienzo.

    b. **Crear desde cifra seguida de ceros**

Similar a la opción anterior pero con la diferencia que en este caso se pasará por parámetro una cifra y una cantidad de ceros, la primera serán los primeros dígitos y los ceros se agregaran a los mismos con la función darCeros que se encarga de concatenar la cantidad de ceros ingresada por el usuario a la cifra inicial.

    c. **CrearAleatorio**

Para crear esta cadena se utilizan las funciones rand y srand. El srand lo declaramos al comienzo del programa en el main, para tener una semilla diferente en cada uso usaremos la función time que nos dará la fecha y hora del sistema. Entonces al usar rand tendremos números al azar todo el tiempo.

Para que la longitud de la cadena sea aleatoria, se utiliza un if inicial donde tiene como condición la función rand()%2 , esto al devolver 0 y 1 permite que algunas veces entre y otras no. Para respetar el largo permitido del entero como máximo se hará hasta 100 veces.

Luego se va agregando de a un número a la vez. Al comienzo el rango será de 1 a 9 para que no comience con 0 ya que solo los octales comienzan con 0, luego con un rango de 0 a 9 obteniendo cualquier dígito de forma aleatoria. Estos son primeramente de tipo int, y luego se convierten a char utilizando la función itoa para luego ser concatenados al entero del número astronómico junto con las flags.

2. **Operaciones de Manejo de Errores**

Estos pueden ser los siguientes:

Ninguno: Este error no tiene sentido alguno en la implementación, ya que si no posee error el campo longitudError del número astronómico representa la longitud y no un código de error. Por ejemplo, si ponemos el número 10 no tiene error alguno entonces su campo longitudError debería ser 2 por su longitud o 0 porque no tiene error? Llegamos a la conclusión que poner un campo de error “Ninguno” es un absurdo.

Cadena nula: Si no tiene dígitos y el puntero del campo entero apunta a NULL.

Cadena invalida: Si posee caracteres que no son válidos, es decir, si no son dígitos.

Overflow: Si el número astronómico se excede del máximo, que es 102 (100 más los flags)

Puntero Nulo: En nuestra implementación este error no sucedería ya que el caso de puntero nulo es considerado en cadena nula.

Los codigos estan especificados por un enum y son los siguientes:

- Ninguno = 0
- CadenaNula = -1
- CadenaInvalida = -2
- Overflow = -3
- PunteroNulo = -4

Son negativos para que en la implementación sea más fácil diferenciar un código de error de la longitud del número astronómico. Para saber si un número astronómico tiene error o no usamos la función requerida NumeroAstronomico_EsError.

En caso de que haya overflow se agrega el código y además se modifica el flag correspondiente (el primero del número astronómico) por 1.

En el código fuente están las funciones EsCadenaNula, EsCadenaInvalida y EsOverflow que reciben el número astronómico y devuelven un valor de verdad 0 o 1.

Luego la función GetTipoDeError informa el código de error que posee el número astronómico.

Las funciones DefinirNulo, DefinirOverflow y DefinirInvalido agregan el código de error a la variable longitudError dependiendo de cuál es el error identificado por la función EstablecerErroresNumAstronomico. Para el caso de DefinirOverflow además se cambiará el flag indicador del mismo.

3. **Operaciones de Salida**
    a. **Mostrar**

Esta función se encarga de recibir un número astronómico, una cantidad de grupos a mostrar en la primera línea y un flujo. En nuestra implementación el flujo puede ser tanto el flujo por consola como un archivo.

La misma al comienzo debe verificar si el número que le enviamos posee error o no. En caso de no poseer error, definirá la cantidad de grupos que hay en el mismo.

Se podrá ver una explicación más detallada en el código:

Archivo: operaciones_nro_astronomico.c - Linea: 144-154

También puede pasar que tenga overflow o sea invalido pero igual queremos mostrarlo entonces utilizando una función auxiliar NumeroAstronomico_OpcionMostrarPorGrupos lo verificaremos antes de enviarla a la función requerida NumeroAstronomico_Mostrar.

Luego de saber cuántos grupos hay en ese entero, utilizaremos un array de int para almacenar los grupos en cada posición del vector. Tuvimos que evaluar los casos en los que el entero no era divisible por 3 haciendo que el primer grupo tenga o 1 o 2 dígitos en la funcion darGrupos

Para poder ir recorriendo los renglones utilizamos un for, y dentro para poder recorrer los grupos utilizamos otro. Dentro lo que haremos será mostrar un grupo separándolo de los demás por un “.” y cada renglón comienza con 2 tabulaciones (“\t\t”) y un fin de línea (“\n”). Aquí mismo se pasara el grupo de int a char* con la función itoa. Para los casos que hay grupos que tienen solo un dígito como int, deberemos agregarle los ceros al comienzo respectivamente, por ejemplo si tenemos 1001 será divido en 1.001 entonces al pasarlo a int* quedará {1,1} por eso al 2do 1 deberemos agregarle sus respectivos ceros al comienzo para mostrar “001”.

Se podrá ver una explicación más detallada en el código:

Archivo: operaciones_nro_astronomico.c - Linea: 162-207

La función devolverá el nuevo flujo.

4. **Operaciones Aritméticas**
    a. **Sumar**

La función NumeroAstronomico_Sumar recibirá dos números astronómicos y retorna el resultado de su suma.

Para esto creamos un nuevo número astronómico que contendrá el resultado el cual se le asignará con la función RealizarSumaEntero a la cual se le pasara por parámetro solo los enteros de cada número (le sumamos 2 a los punteros para que saltee los flags).

Para poder realizar una suma nos dimos cuenta que no sumamos digito a digito como haríamos en programación desde el primero al último, sino que se suma del último al primero. Entonces invertimos los enteros de cada uno e iremos sumando digito a digito con un auxiliar pasandolo a int para sumarlo y luego devolver ese resultado y agregarlo al entero del resultado. En caso de que haya carry se informará desde la función auxiliar y se le sumará 1 al próximo dígito.

Luego verificamos que ambos vectores ya se hayan sumado por completo sino se concatenara el resto del mismo al resultado.

Ya teniendo el resultado en el nuevo entero volveremos a invertirlo para que quede en el orden correcto y usaremos uno de los operandos como auxiliar. En él guardaremos los flags que tenga el resultado. Luego le concatenamos el entero que nos dio la suma al auxiliar. Por último copiaremos el resultado final con los flags al campo entero del resultado y estableceremos sus respectivos errores en caso de tenerlos.

b. Son iguales

Se utiliza internamente la función strcmp, por lo cual si los números son iguales devuelve 0 y agregando un not(!) cambiamos el valor de verdad del resultado.

La variable entero de los números astronómicos tienen un +2 para que de esta forma se omitan los flags.

c. Es menor

Se utiliza la misma función que en la opción anterior pero en este caso si devuelve un valor menor a 0, esto quiere decir que el primer número es menor al segundo, por lo tanto se retorna un 1.

5. Operaciones de persistencias

Para tener un control más simple de los archivos que creamos utilizaremos un archivo auxiliar que contendrá la cantidad de archivos de texto y binarios creados, nos referiremos al mismo como archivo auxiliar contador.

a. Read

Para esta función comenzaremos abriendo el archivo en el formato “rb” que representa lectura en binario y luego utilizaremos la función auxiliar BajarNumeroAstronomico que leerá el archivo y tomará primero el campo longitudError para saber cuánto leer, ya que estamos tratando con estructuras dinámicas. Se le asigna memoria a un auxiliar en el que guardaremos lo que leamos del archivo. Para obtener el número utilizaremos un while hasta que se termine el mismo e iremos leyendo de a un carácter. Luego pasaremos lo leído del auxiliar al campo entero del número astronómico.

b. Write

La función NumeroAstronomico_Write se encargara de subir un número astronómico a un archivo en formato binario. Utilizaremos un método de serialización enviando primero su longitud para luego al bajarlo saber cuánto leer en caso de que sea necesario. E incrementaremos el campo de cantidad de archivos de texto de nuestro archivo auxiliar contador.

c. Scan

Para esta función comenzaremos abriendo el archivo en el formato “r” que representa lectura en formato texto y luego utilizaremos la función BajarNumeroAstronomico que leerá el archivo y tomará primero el campo longitudError para saber cuánto leer, ya que estamos tratando con estructuras dinámicas. Se le asigna memoria a un auxiliar en el

que guardaremos lo que leamos del archivo. Para obtener el número utilizaremos un while hasta que se termine el mismo e iremos leyendo de a un carácter. Luego pasaremos lo leído del auxiliar al campo entero del número astronómico.

d. Print

La función NumeroAstronomico_Print se encargará de subir un número astronómico a un archivo en formato de texto. Utilizaremos un método de serialización enviando primero su longitud para luego al bajarlo saber cuánto leer en caso de que sea necesario. E incrementaremos el campo de cantidad de archivos de texto de nuestro archivo auxiliar contador.