Documentación del Proyecto

Adrover Ramiro, LU: 125925 – Mosconi Alvaro, LU: 126687

Organización de Computadoras

INDICE

DEFINICIONES Y ESPECIFICACIÓN DE REQUERIMIENTOS

DESCRIPCIÓN GENERAL DEL PROYECTO DE SOFTWARE:

El objetivo de este proyecto de software es diseñar un sistema que a partir de una cadena ingresada por el usuario se realice una conversión de un sistema de representación a otro de acuerdo a los parámetros especificados en la cadena, indicando de ser necesario los cómputos realizados durante la conversión.

ESPECIFICACIÓN DE REQUERIMIENTOS DEL PROYECTO DE SOFTWARE

El software permite realizar la conversión del número ingresado expresado en la base numérica origen a otra base numérica destino, cambiando de esta manera el sistema de representación de dicho número.

El programa ofrece un apartado de ayuda, también accesible mediante la línea de comandos, utilizando el indicador "-h" que como resultado mostrara una serie de consideraciones a tener en cuenta respecto al proyecto de software de forma que quien lo necesite pueda entender la sintaxis especifica y los límites del programa.

El sistema no posee la capacidad de convertir número negativos a ningún sistema de presentación.

El software cumple con los objetivos del enunciado, sin restricciones o modificaciones agregadas.

ESPECIFICACIONES DE PROCEDIMIENTOS

PROCEDIMIENTOS DEL DESARROLLO:

HERRAMIENTAS UTILIZADAS

El proyecto está elaborado en su totalidad en el entorno de programación de lenguaje C provisto por la catedra, CodeBlocks, en su versión 20.03 y el compilador usado es el GNU GCC versión 11.2.

ARQUITECTURA DEL SISTEMA:

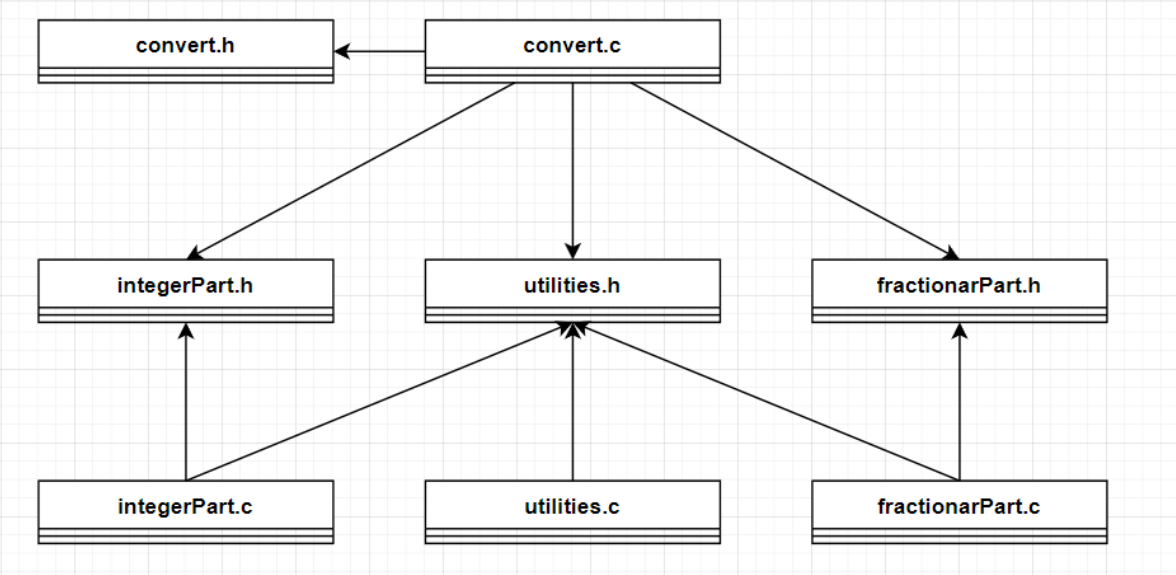
DESCRIPCIÓN JERÁRQUICA:

El sistema está dividido en diferentes módulos (extensiones .c y .h).

Por su parte, en los módulos .h están establecidos los encabezados de las diferentes funciones y procedimientos.

Por otro lado, en los módulos .c se encuentran las implementaciones de dichas funciones y procedimientos.

DIAGRAMA DE MÓDULOS:



DESCRIPCIÓN INDIVIDUAL DE LOS MÓDULOS

- utilities.h: En este módulo se encuentran definidas las funciones y procedimientos implementadas por utilities.c.

- convert.h: en este módulo están definidas las funciones y procedimientos implementados por convert.c.

- integerPart.h: en este módulo están definidas las funciones y procedimientos implementados por integerPart.c.

- fractionalPart.h: en este módulo están definidas las funciones y procedimientos implementados por fractionalPart.c.

- utilities.c: En este módulo se llevaron a cabo la implementación de distintos procedimientos y/o funciones que son de uso común entre los distintos módulos del proyecto.

- convert.c: este módulo funciona como el módulo principal del proyecto, alertando al usuario de posibles errores en la sintaxis especifica del programa, descomponiendo la cadena ingresa y determinar que método de conversión debe utilizarse para cada parte del número teniendo en cuenta las bases utilizadas delegando y repartiendo las tareas de conversión a otras funciones y/o procedimientos. Hace uso de utilities.h.

- integerPart.c: en este módulo están implementadas las funciones y los procedimientos que permiten la correcta conversión de la parte entera del número ingresado por el usuario independientemente de en qué base estén representados, teniendo en cuenta las condiciones provistas por el enunciado del proyecto. Hace uso de utilities.h.

- fractionalPart.c: en este módulo están implementadas las funciones y procedimientos que permiten la correcta conversión de la parte fraccionaria del número ingresado por el usuario independientemente de la base en que estén representados, teniendo en cuenta las condiciones provistas por el enunciado del proyecto. Hace uso de utilities.h

DEPENDENCIAS EXTERNAS:

Este software hace uso de las librerías stdlib.h, stdio.h y Math, para la conversión de arreglos de caracteres a enteros, para la conversión de un número real en un puntero a caracteres, para hacer uso de la función logarítmica y para hacer uso de "pow()" para el cálculo de potencias.

DESCRIPCIÓN DE PROCESOS Y SERVICIOS OFRECIDOS POR EL SISTEMA

El sistema brinda la posibilidad de convertir un numero expresado en una base en el rango [2,16] a cualquier otra base en ese mismo rango, cabe denotar que el sistema hace uso de la base decimal como punto intermedio para las conversiones.

Para ejecutar el programa se lo debe llamar mediante la consola en la ubicación del sistema "…/carpetaDelProyecto/src/bin/Debug/ "usando el comando:

convert -n <number> -s <source_base> -d <destination_base>

Consideraciones:

- El orden en el que se ingresan los argumentos por línea de comandos es irrelevante.
- La concatenación entre dígitos en el software se denota como [Digito1][Digito2].

Parámetros opcionales	Descripción
-s	Indica al sistema que la base origen será el parámetro siguiente. En caso de no ser ingresado, se asume que la base origen es base decimal (base 10).
-d	Indica al sistema que la base destino será el parámetro siguiente. En caso de no ser ingresado, se asume que la base destino es base decimal (base 10).
-v	Muestra los cómputos intermedios en la consola a medida que se realiza la conversión.
-h	Muestra por consola una ayuda al usuario que especifica la sintaxis y los límites del sistema. Tiene prioridad sobre todo lo demás.

En caso de que se algún parámetro sea ingresado de forma incorrecta, se mostrara un mensaje de error acorde. En caso contrario se realizará la conversión del número ingresado a la base correspondiente.

A continuación, se pueden observan diferentes capturas probando las diferentes posibilidades de conversión que posee el sistema.

EJEMPLOS CON CAPTURAS DE PANTALLA

CONVERSIÓN DE CUALQUIER BASE ORIGEN A BASE A DECIMAL:

Cadena ingresada: convert -n 1010.4 -s 16 -d 10 -v

```
C:\Users\alvar\Desktop\Álvaro\Organización de Computadoras\Practicos\Practico B\convertidor\bin\Debug>convert -n 1010.4 -s 16 -d 10 -v


 -------------------- INICIO DE LA CONVERSION (PARTE ENTERA) --------------------

 R0 --> 0 * 16^0 = 0

 R1 --> 1 * 16^1 = 16

 R2 --> 0 * 16^2 = 0

 R3 --> 1 * 16^3 = 4096

 Numero Convertido --> R0 + R1 + R2 + R3  = (4112)b10

 ---------------------- FIN DE LA CONVERSION (PARTE ENTERA) ----------------------

 -------------------- INICIO DE LA CONVERSION (PARTE FRACCIONARIA) --------------------

 R-1 --> 4 * 16^-1 = 0.250000

 Numero Convertido --> R-1 = (0.250000)b10

 ---------------------- FIN DE LA CONVERSION (PARTE FRACCIONARIA) ----------------------

 NUMERO OBTENIDO: (4112.250000)b10
```

CONVERSIÓN DE BASE ORIGEN DECIMAL A CUALQUIER BASE DESTINO

Cadena ingresada: convert -n 1010.4 -s 10 -d 14

```
C:\Users\alvar\Desktop\Álvaro\Organización de Computadoras\Practicos\Practico B\convertidor\src\bin\Debug>convert -n 1010.4 -s 10 -d 14 -v


 -------------------- INICIO DE LA CONVERSION (PARTE ENTERA) --------------------

1.   1010 / 14  --- Resto1 = 2 <=> (2)b14 --- Cociente: 72
2.   72 / 14    --- Resto2 = 2 <=> (2)b14 --- Cociente: 5
3.   5 / 14     --- Resto3 = 5 <=> (5)b14 --- Cociente: 0

Numero Convertido --> [Resto3][Resto2][Resto1] = (522)b14

 ---------------------- FIN DE LA CONVERSION (PARTE ENTERA) ----------------------

 -------------------- INICIO DE LA CONVERSION (PARTE FRACCIONARIA) --------------------

0. --> 0.400000 * 14 = 5        D0 ==> (5)b14
1. --> 0.600000 * 14 = 8        D1 ==> (8)b14
2. --> 0.400000 * 14 = 5        D2 ==> (5)b14
3. --> 0.600000 * 14 = 8        D3 ==> (8)b14
4. --> 0.400000 * 14 = 5        D4 ==> (5)b14
5. --> 0.600000 * 14 = 8        D5 ==> (8)b14
6. --> 0.400000 * 14 = 5        D6 ==> (5)b14
7. --> 0.600000 * 14 = 8        D7 ==> (8)b14
8. --> 0.400000 * 14 = 5        D8 ==> (5)b14
9. --> 0.600001 * 14 = 8        D9 ==> (8)b14

Numero Convertido --> [Digito0][Digito1][Digito2][Digito3][Digito4][Digito5][Digito6][Digito7][Digito8][Digito9] = (5858585858)b14

 ---------------------- FIN DE LA CONVERSION (PARTE FRACCIONARIA) ----------------------


 NUMERO OBTENIDO: (522.5858585858)b14
```

CONVERSIÓN DE CUALQUIER BASE ORIGEN A CUALQUIER BASE DESTINO

Cadena ingresada: convert -n 4242.42 -s 8 -d 16

```
C:\Users\alvar\Desktop\Álvaro\Organización de Computadoras\Practicos\Practico B\convertidor\bin\Debug>convert -n 4242.42 -s 8 -d 16 -v


 -------------------- INICIO DE LA CONVERSION (PARTE ENTERA) --------------------

 R0 --> 2 * 8^0 = 2

 R1 --> 4 * 8^1 = 32

 R2 --> 2 * 8^2 = 128

 R3 --> 4 * 8^3 = 2048

 Numero Convertido --> R0 + R1 + R2 + R3  = (2210)b10

 ---------------------- FIN DE LA CONVERSION (PARTE ENTERA) ----------------------

 -------------------- INICIO DE LA CONVERSION (PARTE FRACCIONARIA) --------------------

 R-1 --> 4 * 8^-1 = 0.500000

 R-2 --> 2 * 8^-2 = 0.031250

 Numero Convertido --> R-2 + R-1 = (0.531250)b10

 ---------------------- FIN DE LA CONVERSION (PARTE FRACCIONARIA) ----------------------

 -------------------- INICIO DE LA CONVERSION (PARTE FRACCIONARIA) --------------------

R0 --> 0.531250 * 16 = 8        <==> (8)b16
R1 --> 0.500000 * 16 = 8        <==> (8)b16

Numero Convertido --> [Resto0][Resto1] = (88)b16

 ---------------------- FIN DE LA CONVERSION (PARTE FRACCIONARIA) ----------------------
```

```
 ---------------------- INICIO DE LA CONVERSION (PARTE ENTERA) ----------------------

1.   2210 / 16  --- Resto1 = 2 <=> (2)b16 --- Cociente: 138
2.   138 / 16   --- Resto2 = 10 <=> (A)b16 --- Cociente: 8
3.   8 / 16     --- Resto3 = 8 <=> (8)b16 --- Cociente: 0

Numero Convertido --> [Resto3][Resto2][Resto1] = (8A2)b16

 ----------------------- FIN DE LA CONVERSION (PARTE ENTERA) -----------------------

 NUMERO OBTENIDO: (8A2.88)b16
```

DOCUMENTACIÓN TÉCNICA – ESPECIFICACIÓN API:

CONVERT.H

FUNCTIONS

- int **main** (int argc, char *argv[])
- int * **parseArguments** (int nArg, char *argv[])
- void **buildNumber** (char *integerPart, char *fractionalPart, int *sourceBase, int *destinationBase, int *detailed)
- void **help** ()
- int * **validateNumber** (const char *number, const int *base)
- char * **getFractionalSide** (const char *number, char *destination)
- char * **getIntegerSide** (const char *number, char *destination)
- int * **stringLength** (char *string)

FUNCTION DOCUMENTATION

VOID BUILDNUMBER (CHAR * INTEGERPART, CHAR * FRACTIONALPART, INT * SOURCEBASE, INT * DESTINATIONBASE, INT * DETAILED)

Procedimiento encargado de hacer los llamados a las funciones y/o procedimientos que convierten los números expresados en base origen a el número equivalente en base destino.

PARAMETERS

integerPart	Puntero a char que apunta al primer carácter de la parte entera del número ingresado por el usuario.
fractionalPart	Puntero a char que apunta al primer carácter de la parte fraccionaria del número ingresado por el usuario.
sourceBase	Puntero a entero que apunta al número que representa la base numérica origen ingresada por el usuario.
destinationBase	Puntero a entero que apunta al número que representa la base numérica destino ingresada por el usuario.
detailed	Puntero a entero que apunta al número 1 si el usuario ingreso el parámetro "-v", 0 si el usuario no ingreso el parámetro "-v".

CHAR * GETFRACTIONALSIDE (CONST CHAR * NUMBER, CHAR * DESTINATION)

Función encargada de extraer la parte fraccionaria del número ingresado.

PARAMETERS

number	Puntero a char que apunta al primer carácter del número ingresado por el usuario (expresado en cadena de caracteres).
destination	Puntero auxiliar para ubicar el inicio de la cadena a retornar.

RETURNS

Puntero a char que apunta al primer carácter de la parte fraccionaria.

CHAR * GETINTEGERSIDE (CONST CHAR * NUMBER, CHAR * DESTINATION)

Función encargada de extraer la parte entera del número ingresado.

PARAMETERS

number	Puntero a char que apunta al primer carácter del número ingresado por el usuario (expresado en cadena de caracteres).
destination	Puntero auxiliar para ubicar el inicio de la cadena a retornar.

RETURNS

Puntero a char que apunta al primer carácter de la parte entera.

VOID HELP ()

Procedimiento encargado de proveer la correcta sintaxis y semántica al usuario.

INT MAIN (INT ARGC, CHAR * ARGV[])

Procedimiento encargado de analizar los argumentos e invocar a otros/as procedimientos y funciones.

PARAMETERS

argc	Indica el número de argumentos ingresados por el usuario.
argv	Puntero a arreglo de punteros (Contiene los argumentos ingresados por el usuario).

RETURNS

0 Si la ejecución fue exitosa, 1 si se produjo algún error.

INT * PARSEARGUMENTS (INT NARG, CHAR * ARGV[])

Función encargada de analizar los argumentos e invocar a otros/as procedimientos y funciones.

PARAMETERS

nArg	Indica el número de argumentos ingresados por el usuario.
argv	Puntero a arreglo de punteros (Contiene los argumentos ingresados por el usuario).

RETURNS

Puntero a entero que apunta al entero 0 si la ejecución fue exitosa, 1 si se produjo algún error.

INT * STRINGLENGTH (CHAR * STRING)

Función encargada de contar la cantidad de caracteres de una cadena de caracteres.

PARAMETERS

string	Cadena de caracteres ingresada por el usuario.

RETURNS

Cantidad de caracteres de la cadena recibida.

Función encargada de contar la cantidad de caracteres de una cadena de caracteres.

PARAMETERS

string	Puntero a char que apunta al primer carácter de la cadena ingresada por el usuario.

RETURNS

Puntero a entero que apunta a un entero que representa la cantidad de caracteres de la cadena recibida.

INT * VALIDATENUMBER (CONST CHAR * NUMBER, CONST INT * BASE)

Función encargada de verificar si el número recibido por parámetro corresponde a la base recibida por parámetro.

PARAMETERS

number	Puntero a char que apunta al primer carácter del número ingresado por el usuario (expresado en cadena de caracteres).
base	Puntero a entero que apunta al número que representa la base numérica origen ingresada por el usuario.

RETURNS

Puntero a entero que apunta al entero 1 si el número corresponde a la base, 0 en caso contrario.

CONVERT.C

```
#include <stdio.h>
#include <stdlib.h>
#include "convert.h"
#include "integerPart.h"
#include "fractionalPart.h"
#include "utilities.h"
```

FUNCTIONS

- int **main** (int argc, char *argv[])
- int * **parseArguments** (int nArg, char *argv[])
- void **buildNumber** (char *integerPart, char *fractionalPart, int *sourceBase, int *destinationBase, int *detailed)
- void **help** ()
- int * **validateNumber** (const char *number, const int *base)
- char * **getFractionalSide** (const char *number, char *destination)
- char * **getIntegerSide** (const char *number, char *destination)

FRACTIONALPART.H

FUNCTIONS

- double * **divisionMethodFractional** (char *number, int *base, int *detailed)
- char * **multiplicationMethodFractional** (char *number, int *base, int *detailed)

FUNCTION DOCUMENTATION

DOUBLE * DIVISIONMETHODFRACTIONAL (CHAR * NUMBER, INT * BASE, INT * DETAILED)

Función encargada de llevar a cabo el método de la división para la parte fraccional del número ingresado por el usuario.

PARAMETERS

number	Puntero a char que apunta al primer carácter de la parte fraccionaria del número ingresado por el usuario (expresado en cadena de caracteres).
base	Puntero a entero que apunta al número que representa la base numérica origen ingresada por el usuario.
detailed	Puntero a entero que apunta al número 1 si el usuario ingreso el parámetro "-v", 0 si el usuario no ingreso el parámetro "-v".

RETURNS

Puntero a float que apunta al número fraccional ya convertido a base 10.

CHAR * MULTIPLICATIONMETHODFRACTIONAL (CHAR * NUMBER, INT * BASE, INT * DETAILED)

Funcion encargada de llevar a cabo el metodo de la multiplicación para la parte fraccional del número ingresado por el usuario.

PARAMETERS

number	Puntero a char que apunta al primer carácter de la parte fraccionaria del número ingresado por el usuario (expresado en cadena de caracteres).
base	Puntero a entero que apunta al número que representa la base numérica destino ingresada por el usuario.
detailed	Puntero a entero que apunta al número 1 si el usuario ingreso el parámetro "-v", 0 si el usuario no ingreso el parámetro "-v".

RETURNS

Puntero a char que apunta al primer carácter del número fraccional ya convertido a base destino.

FRACTIONALPART.C

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "fractionalPart.h"
#include "utilities.h"
```

FUNCTIONS

- double * **divisionMethodFractional** (char *number, int *base, int *detailed)
- char * **multiplicationMethodFractional** (char *number, int *base, int *detailed)

INTEGERPART.H

FUNCTIONS

- int * **transformNumberToDecimal** (char *characterToTransform)
- int * **multiplicationMethodInteger** (char *number, int *base, int *detailed)
- void **setEquivalentDigit** (char *currentDigit, int *reminder)
- void **reverseString** (char *stringToReverse)
- char * **divisionMethodInteger** (char *number, int *base, int *detailed)

FUNCTION DOCUMENTATION

CHAR * DIVISIONMETHODINTEGER (CHAR * NUMBER, INT * BASE, INT * DETAILED)

Funcion encargada de llevar a cabo el metodo de la división para la parte entera del número ingresado por el usuario.

PARAMETERS

number	Puntero a char que apunta al primer carácter de la parte entera del número ingresado por el usuario (expresado en cadena de caracteres).
base	Puntero a entero que apunta al entero que representa la base numérica destino ingresada por el usuario.
detailed	Puntero a entero que apunta al número 1 si el usuario ingreso el parámetro "-v", 0 si el usuario no ingreso el parámetro "-v".

RETURNS

Puntero a char que apunta al primer carácter del número entero ya convertido a base destino.

INT * MULTIPLICATIONMETHODINTEGER (CHAR * NUMBER, INT * BASE, INT * DETAILED)

Funcion encargada de llevar a cabo el metodo de la multiplicación para la parte entera del número ingresado por el usuario.

PARAMETERS

number	Puntero a char que apunta al primer carácter de la parte entera del número ingresado por el usuario (expresado en cadena de caracteres).
base	Puntero a entero que apunta al entero que representa la base numérica origen ingresada por el usuario.
detailed	Puntero a entero que apunta al número 1 si el usuario ingreso el parámetro "-v", 0 si el usuario no ingreso el parámetro "-v".

RETURNS

Puntero a char que almacena el número fraccional ya convertido a base 10.

VOID REVERSESTRING (CHAR * STRINGTOREVERSE)

Procedimiento encargado de invertir una cadena de caracteres.

PARAMETERS

stringToReverse	Puntero a char que apunta a el primer carácter de la parte entera del número ingresado por el usuario ya convertido a base destino (expresado en cadena de caracteres).

VOID SETEQUIVALENTDIGIT (CHAR * CURRENTDIGIT, INT * REMINDER)

Procedimiento encargado de asignar el carácter equivalente al número recibido por parámetro.

PARAMETERS

currentDigit	Puntero a char que almacena la posición de memoria correspondiente al digito a transformar a base destino.
reminder	Puntero a entero que apunta al resto resultante de la división (expresado en base origen)

INT * TRANSFORMNUMBERTODECIMAL (CHAR * CHARACTERTOTRANSFORM)

Funcion encargada de transformar un carácter ASCII a su entero decimal equivalente.

PARAMETERS

characterToTransform	Puntero a char que apunta al carácter a transformar.

RETURNS

Puntero a entero que apunta a un entero decimal equivalente al char recibido por parámetro.

INTEGERPART.C

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "integerPart.h"
#include "utilities.h"
```

FUNCTIONS

- void **reverseString** (char *stringToReverse)
- char * **divisionMethodInteger** (char *number, int *base, int *detailed)
- int * **multiplicationMethodInteger** (char *number, int *base, int *detailed)

UTILITIES.H

FUNCTIONS

- int * **transformNumberToDecimal** (char *characterToTransform)
- int * **stringCompare** (const char *a, const char *b)
- int * **stringLength** (char *string)
- void **setEquivalentDigit** (char *currentDigit, int *reminder)
- int * **getRequiredSizeForNumber** (char *number, int *destinationBase)

FUNCTION DOCUMENTATION

INT * GETREQUIREDSIZEFORNUMBER (CHAR * NUMBER, INT * DESTINATIONBASE)

Funcion encargada de retornar la cantidad de dígitos necesarias para representar el número ingresado en base destino.

PARAMETERS

number	Puntero a char que apunta al primer carácter del número ingresado por el usuario (expresado en cadena de caracteres).
destinationBase	Puntero a entero que apunta al entero que representa la base numérica destino ingresada por el usuario.

RETURNS

Puntero a entero que apunta al entero que indica la cantidad de dígitos necesarios para representar el número ingresado en base destino.

VOID SETEQUIVALENTDIGIT (CHAR * CURRENTDIGIT, INT * REMINDER)

Procedimiento encargado de asignar el carácter equivalente al número recibido por parámetro.

PARAMETERS

currentDigit	Puntero a char que almacena la posición de memoria correspondiente al digito a transformar a base destino.
reminder	Puntero a entero que apunta al resto resultante de la división (expresado en base origen)

INT * STRINGCOMPARE (CONST CHAR * STRINGA, CONST CHAR * STRINGB)

Funcion encargada de comparar cadena de caracteres (strings).

PARAMETERS

stringA	Puntero a char que apunta al primer carácter de la stringA.
stringB	Puntero a char que apunta al primer carácter de la stringB.

RETURNS

Puntero a entero que apunta a un 1 si las cadenas eran iguales y a un 0 en caso contrario.

INT * STRINGLENGTH (CHAR * STRING)

Funcion encargada de contar la cantidad de caracteres de una cadena de caracteres.

PARAMETERS

string	Cadena de caracteres ingresada por el usuario.

RETURNS

Cantidad de caracteres de la cadena recibida.

Funcion encargada de contar la cantidad de caracteres de una cadena de caracteres.

PARAMETERS

string	Puntero a char que apunta al primer carácter de la cadena ingresada por el usuario.

RETURNS

Puntero a entero que apunta a un entero que representa la cantidad de caracteres de la cadena recibida.

INT * TRANSFORMNUMBERTODECIMAL (CHAR * CHARACTERTOTRANSFORM)

Funcion encargada de transformar un carácter ASCII a su entero decimal equivalente.

PARAMETERS

characterToTransform	Puntero a char que apunta al carácter a transformar.

RETURNS

Puntero a entero que apunta a un entero decimal equivalente al char recibido por parámetro.

UTILITIES.C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "utilities.h"
```

FUNCTIONS

- int * **transformNumberToDecimal** (char *characterToTransform)
- int * **stringCompare** (const char *stringA, const char *stringB)
- int * **stringLength** (char *string)
- void **setEquivalentDigit** (char *currentDigit, int *reminder)
- int * **getRequiredSizeForNumber** (char *number, int *destinationBase)

CONCLUSIONES FINALES:

Se opto por almacenar las bases de origen y destino ingresadas por el usuario en punteros a enteros y almacenar el numero ingresado por el usuario en un puntero a char para luego fragmentarlo en 2 partes; la parte entera y la parte fraccional, tales partes serían enviadas a distintas funciones y/o procedimientos de acuerdo a las bases ingresadas donde se efectuarán los métodos de conversión acordes por separado para que luego las partes sean unidas y mostradas por consola al usuario.

Además, hemos observado a lo largo del desarrollo del sistema, que las acciones que una persona puede realizar de manera relativamente sencilla pueden presentar complicaciones al momento de adaptarlas a un algoritmo a replicar por una computadora. Ej. Separar la parte entera y la fraccionaria a la hora de convertir un numero fraccionario mediante el método de la multiplicación.

Para la implementación del sistema, se opto utilizar 20 dígitos para representar la parte fraccionaria de los números convertidos, el objetivo de hacer esto es evitar que ocurran redondeos no deseados.

Gracias a la realización del proyecto creemos haber obtenido:

- Habilidad en la manipulación de punteros.
- Experiencia en la modularización en un lenguaje imperativo como lo es C.
- Experiencia en la utilización de herramientas para la creación de la documentación del código.