

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

GARZÓN DOMÍNGUEZ GERARDO ISMAEL

2CV13

TEORÍA COMPUTACIONAL

PRÁCTICA 1:

OPERACIONES CON CADENAS

INTRODUCCIÓN:

Nosotros como humanos estamos familiarizados con el concepto de lenguajes naturales, pues estos son los que usualmente usamos para comunicarnos entre nosotros. El hecho es que se necesita un estudio mas riguroso sobre lenguajes desde el punto de vista matemático y de la computación, así que a continuación vamos a mostrar las definiciones más básicas para su estudio.

Un alfabeto es un conjunto de símbolos, tiene la característica de ser finito y no vacío, usualmente un alfabeto está notado con el símbolo: [1]

A partir de los símbolos de un alfabeto, se pueden construir cadenas, las cuales son secuencias finitas que contienen los símbolos de un alfabeto.

La longitud de una cadena es la cantidad de símbolos que esta contiene. Existe una cadena especial, la cual no contiene símbolos y su longitud es 0, a esta la conocemos como la cadena vacía A continuación, se muestra la notación común para la longitud de una cadena y de la cadena vacía:

: Denota la longitud de la cadena

: Es la cadena vacía y

[1]

Existen ciertas operaciones que se pueden realizar sobre una cadena, a continuación, se listarán cuales son y en que consisten:

Concatenación: La concatenación consiste en agregar los símbolos de una cadena al final de otra cadena [1], puede ser denotada como:

Prefijo: En términos prácticos, un prefijo se obtiene al remover cero o más símbolos desde el final de una cadena. [2]

Sufijo: Es una cadena obtenida al remover cero o más símbolos desde el inicio de una cadena. [2]

Subcadena: Se obtiene al remover cualquier sufijo o prefijo de una cadena. [2]

Subsecuencia: Es una cadena formada al remover cero o más símbolos no necesariamente contiguos dentro de una cadena [2], pero como su nombre lo indica, estos símbolos deben por lo menos estar ordenados en secuencia, o en otras palabras deben seguir el orden original de la cadena, si lo notamos, las subsecuencias generalizan la definición de sufijos, prefijos y subcadenas. Como nota adicional, una subsequencia de una cadena, es propia si esta es distinta a la cadena vacía o a la cadena original. [2]

IMPLEMENTACIÓN DE CADENAS

En esta sección describiré de manera sencilla la implementación propia de cadenas y sus operaciones, así como algunos de los retos al escribir el código de la práctica.

Decidí implementar las cadenas como un tipo abstracto de dato, cuyos atributos son invisibles para el usuario, esto usando una “declaración hacia enfrente” de una estructura en un archivo de cabecera “.h”. La implementación y el tipo completo se encuentran en un archivo “.c”, el tipo completo es una estructura que mantiene un apuntador al tipo char y dos variables de tipo size\_t que mantienen registro del numero de elementos en el bloque de memoria apuntado por la variable tipo char\*, y el numero de elementos disponibles en dicho bloque de memoria. En el archivo “array.h” (es código que escribí como mini proyecto) se encuentran unos macros que se encargan de calcular cuando es necesario llamar “realloc” dependiendo de los elementos que se inserten en un arreglo.

Uno de los retos que me enfrenté al programar los métodos para las cadenas fue decidir si implementar una función de concatenación, o usar una función que generaliza ese comportamiento, como lo es “memmove” de la librería estándar “string.h”, el problema fue que está función realiza comparación relacional de apuntadores (operadores como >, <), y según el estándar de C, comparar apuntadores que no apunten dentro de un mismo bloque de memoria es comportamiento indefinido [3], esto puede ser un problema para constructores que copian un arreglo de caracteres normal (const char\*) a un bloque de memoria alojado por “malloc”, sin embargo, decidí confiar en que la mayoría de compiladores, al menos en plataformas comunes, no fallan al comparar apuntadores en distintos bloques de memoria.

FUNCIONAMIENTO DEL PROGRAMA

A continuación, se muestra el programa en acción realizando algunas de las operaciones de cadenas descritas anteriormente.

Text, letter

Description automatically generated

Figura 1 El programa en funcionamiento, despliega un menú con instrucciones y como usar, se debe introducir una palabra correspondiente a una acción para desplegar un “asistente” para la función, o la tecla return para volver a imprimir el menú.

Graphical user interface, text, application

Description automatically generated

Figura 2 Se inicia con la cadena vacía, se puede usar el comando “new” o “cc” para, crear una nueva cadena, o concatenar a la cadena en uso, en este caso a la cadena vacía le concatenamos la cadena “Hola” y después la cadena “Mundo”.

Text

Description automatically generated

Figura 3: Generamos una subcadena mediante la instrucción “ss”, que llamará a las funciones correspondientes para eliminar n caracteres desde el final y el inicio de la cadena.

Chart, text

Description automatically generated

Figura 4: Potencia de una cadena mediante el comando “pw”, llamará una función para recibir un número tipo long en C, después llamará al método correspondiente con dicho argumento como potencia para la cadena.

Text

Description automatically generated with medium confidence

Figura 5: Genera una subsecuencia a partir de una cadena, la función llamada al introducir “sq” se encargará de recibir una lista de índices validos para la cadena, en este caso se generó una subsecuencia a partir del primer símbolo y el ultimo de la cadena original.

CONCLUSION

Me ha resultado bastante útil implementar mi propia interface para cadenas, ya que me puede ser útil en el futuro si deseo extenderla a caracteres Unicode por ejemplo, aunque no es muy relevante para los propósitos de esta práctica. Ahora respecto al tema de la práctica, me ayudo a comprender bien las operaciones de cadenas y de alguna manera como ciertas operaciones generalizan a otras como es el caso de las subsecuencias, además de que también me ayudo a ampliar un poco más mi conocimiento sobre el estándar de C, en este caso, mi práctica se supone que por lo menos compila para el estándar de C99

Referencias:

[1] Peter Linz - An Introduction to Formal Languages and Automata [6th ed.]-Jones & Bartlett (2017)

[2] Alfred V. Aho – Compilers Principles, Thechniques & Tools, [2nd ed.] Pearson Addison Wesley (2006)

[3]<https://stackoverflow.com/questions/58321641/why-does-comparing-pointers-with-undefined-behavior-still-give-correct-results>