|  |
| --- |
| Universidad Católica del Uruguay |
| Transformador de cadenas de caracteres- DOCUMENTO DE ANALISIS |
|  |

|  |
| --- |
| Agustín PICOS Micaela OLIVERA  31/05/2017 |



Contenido

* Planteo del problema ………………………………………………………………………………PAG.3
* Análisis de soluciones alternativas……………………………………………………………PAG.4
* Justificación de la solución elegida……………………………………………………………PAG.5
* Desarrollo de la solución………………………………………………………..…………………PAG.6
* Conclusiones…………………………………………………………………….………………………PAG.7

Planteo del problema

Lo que nosotros comprendimos que se debe cumplir para lograr la consigna del primer obligatorio es lo siguiente:

* Lo que se debe lograr en este trabajo es poder elaborar un programa que permita a un usuario cualquiera aplicar modificaciones sobre una cadena de caracteres que el mismo determine. Las modificaciones estarán integradas en un menú principal pudiendo observar la opción y una breve descripción de su resultado. A su vez el programa tendrá un menú opcional (modo comando) en el cual el usuario puede realizar las modificaciones ingresando comandos manualmente. Este nuevo modo también debe permitir al usuario concatenación entre varios comandos.
* El programa debe permitir pasar del menú estándar al modo comando y viceversa, también debe incluir un comando que despliegue contacto de los creadores y en el caso de desear salir del programa, el usuario podrá ver la cadena inicial (la cadena que él ingresó al comienzo del programa) y a su vez verá la cadena final.

Los problemas que nos planteamos antes de comenzar el primer obligatorio fueron:

* Realizar un programa que satisfaga los requerimientos de la consigna y que mantenga un adecuado equilibro de eficacia y eficiencia.
* Hacer que nuestro programa sea de sencillo entendimiento para que cualquier usuario pudiera manejarlo sin ningún problema. Esto incluye que el programa sea claro de forma que no se generen ambigüedades al utilizarlo.
* Lograr todos los requerimientos expresados en la consigna ya sea mediante los conocimientos obtenidos en clase como también investigando fuera de clase.

Análisis de soluciones alternativas

Durante el desarrollo del programa surgieron varias alternativas:

* Al principio nuestra idea fue realizar una única función que contenga todas las opciones y comandos, esa función se encargaría de comparar las entradas del usuario con todos los comandos.
* Otra opción era que los comandos del programa y los ingresados por el usuario se trabajasen como cadenas de caracteres. De esa forma la manera de que el programa pudiera reconocer la orden del usuario seria comparando las cadenas y cuando estas fueran idénticas se ejecutarían las modificaciones.
* Luego se creó una lista con los comandos y cuando el usuario ingresaba un comando,el programa comprobaría si el comando ingresado se encontraba en la lista de comandos. A su vez a esta alternativa se le agregaba una extensión compuesta por una lista que contenía todas las opciones posibles de sustitución para las listas de los comandos de cifrado, descifrado y de sustitución de caracteres.
* Dificultades que se presentaron fueron principalmente las condiciones de bucle, tanto en el menú principal, como en el modo comando, también hubo confusiones sobre si utilizar entradas de tipo (int) para el menú principal y tipo (str) para el modo comando.
* Finalmente, otra alternativa era crear una lista para las opciones principales y verificar si las opciones ingresadas correspondían con las opciones de transformación. De esta forma el programa recorrería cada uno de los comandos ingresados y ejecutaría cada transformación en el orden que el usuario las ingresó (esto no ocurría cuando los comandos eran cadenas de caracteres). Esta alternativa permitiría que si o si las transformaciones se efectuaran en el orden en que son ingresados los comandos.

Justificación de la solución elegida

La solución final es producto de varias modificaciones sobre el borrador inicial. No se tuvo la oportunidad de elegir entre dos opciones, sino que se tomó cada elemento del programa y se mejoró hasta que no se produjeran errores en la ejecución.

Es de nombrar, la supresión de las listas que contenían todas las transformaciones y cifrados, ya que el método en el que se utilizaban funcionaba correctamente, pero era engorroso.

En el trabajo final, se culminó con un programa subdividido en dos partes: las opciones del menú principal, y el modo comando. De esta forma, las entradas podían referirse solamente a la sección donde se encontraba el usuario. Esta elección permitió realizar un bucle de transformaciones dentro del modo comando, independientemente del menú principal y viceversa.

El trabajar con los comandos ingresados, transformándolos en una lista permitió ingresar los comandos y trabajar con ellos recorriendo cada uno de los elementos y en orden de presentación.

Cerca del final del trabajo, se realizaron unas correcciones con los métodos “while” ya que, por ejemplo, dentro de las opciones se encontraban repetidas varias veces, sentencias para el ingreso de datos y verificación de los datos ingresados.

Desarrollo de la solución

Conclusiones

Al finalizar el primer obligatorio podemos concluir que durante el desarrollo del trabajo se presentó la siguiente evolución; al principio desarrollábamos un modelo tal como se nos ocurría, y a medida que aparecían errores en su ejecución, lo descartábamos y lo volvíamos a hacer pero mejorado, y si ese nuevo modelo funcionaba lo seguíamos utilizado. Si veíamos que el modelo no era compatible con las funciones que más adelante se debían agregar lo descartábamos y hacíamos uno nuevo.

Esa situación se repitió varias veces hasta el punto de crear un total de cuatro modelos, diferentes en su estructura general pero a la vez similar en varios conceptos.

Otro detalle a destacar es que durante el desarrollo del trabajo creamos cosas redundantes, que en el afán de mejorar el programa terminaban siendo engorrosas o estando de más. En algunas etapas desarrollamos cosas mediante un camino largo y costoso de comprender, pero que luego de conseguir comprenderlo se pudo llegar a simplificar, sin embargo el programa funcionaban sin que el usuario viera que ese camino era el más largo.

Finalmente se podría decir que la mayor dificultad que se nos presentó no era hacer que el programa hiciera los que nosotros queríamos cuando le ingresábamos comando correcto, sino que el programa diera el error correspondiente cuando un usuario ingresaba un comando erróneo por accidente.