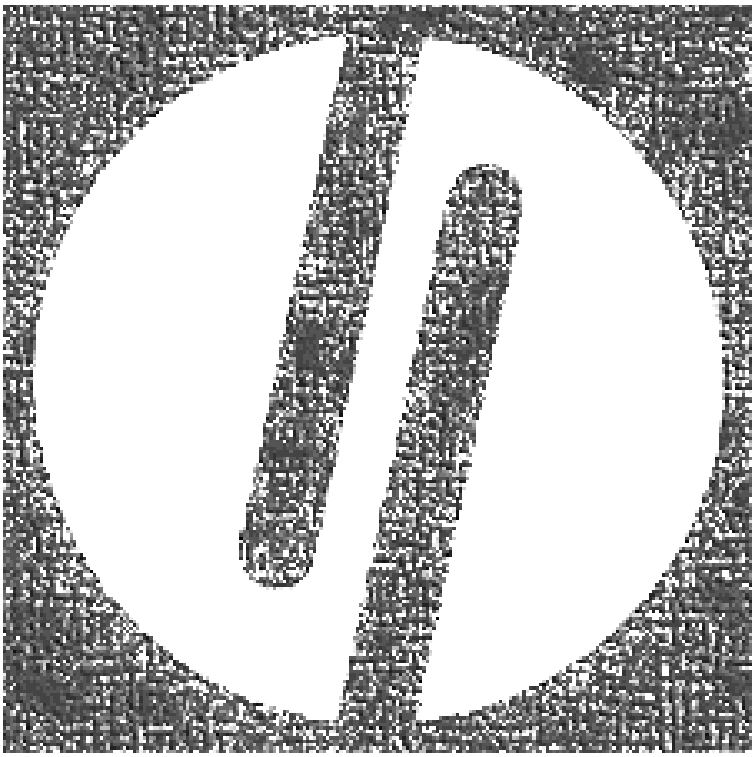
Fecha de Entrega: 24/11/2023

UNIVERSIDAD NACIONAL GENERAL SARMIENTO

Organización del Computador

Trabajo Practico Final



Alumno: Guillermo Ezequiel Sheil
Documento: 40.308.544
Comisión: COM-04

Organización del Computador
Trabajo Práctico Final

Por medio de este informe se pretende presentar el proyecto final de la materia Organización del Computador. En este se han utilizado diferentes prácticas aprendidas durante la cursada del segundo cuatrimestre del año 2023.
El trabajo busca resolver el algoritmo de encriptamiento Cesar Mejorado siguiendo las consignas detalladas en el trabajo designado por el profesor Mariano Vargas e Ignacio Tula.
Para la resolución de este trabajo se buscó generar una estructura basada en subrutinas, las cuales facilitaron el proceso ya que estas cuentan con la suficiente automatización e independencia necesaria para poder reutilizarse según sea necesario. Con respecto al main del programa, sabemos que este podría ser reducido a más subrutinas, encapsulando de esta manera las impresiones por pantalla, pero tomando en cuenta que este programa es independiente de otro, es decir que no es una herramienta implementada en un código más grande, se decidió por utilizar el main como el encargado de llamar a las distintas subrutinas y de organizar las distintas llamadas para imprimir por pantalla.

Si se buscase resumir el trabajo este consta de 2 subrutinas fundamentales, la codificación y la decodificación, estas subrutinas son las encargadas de codificar y decodificar todo, independientemente si se utiliza palabra secreta o código numérico, estas convierten las letras asignadas en otras shiftiandolas (según sea la clave). Ambas toman en consideración, los espacios, los signos de exclamación y de interrogación, las comas y las comillas dobles. Todas estas no son convertidas, pero si son contadas como letras procesadas. También toma en cuenta las mayúsculas y minúsculas, reiniciando el ciclo del abecedario si la conversión supera la letra Z en caso de codificar o la A en caso de decodificar. Tener en cuenta que este programa codifica y decodifica desplazamiento del 0 al 9 por letra. La clave para decodificar y codificar no tiene límite salvo el espacio reservado en memoria, es decir se puede usar tanto Hola;1;2;3;c como Hola;1;2;3;4;5;6;c. El programa entra en un bucle en el cual si se le acaba la clave arranca nuevamente por el primer número.
Si bien ya se tienen ambas subrutinas principales se necesitan muchas otras para poder conformar e incluso para poder llegar la instancia de utilizarlas, ejemplo de esto último son

las subrutinas para extraer mensaje, clave, palabra clave y código. Todas estas y el resto de las subrutinas que se mencionan serán detalladas al final del trabajo en más detalle.
El programa también cuenta con un menú interactivo, la implementación de la subrutina lengthS es clave para el funcionamiento de este. Esta subrutina cuenta caracteres hasta toparse con "~", esto permite crear un menú el cual se puede visualizar en el editor ya que si el length terminara en "/n" o en "0" no habría posibilidad ni de hacer saltos de línea ni de visualizarlo de manera clara y cómoda en el editor. También mencionar que cada pestaña del menú debe ser cargada para luego imprimirse.
El programa también cuenta con funciones para convertir número enteros a cadenas y cadenas a enteros. En el caso de los números enteros a cadenas hay 2 subrutinas, una recibe un número del 1 al 9 y lo convierte, esto es debido a la representación en ascii, la otra es un tanto más complejo, toma un número y se fija cuantos dígitos tiene, en base a eso va dividiendo en el caso del 221, primero se divide por 100, luego se multiplica la parte entera del número obtenido en la división y se lo multiplica por 100 obteniendo de esta forma 200. Luego al número total se le resta el 200 para obtener el 21 y se repite el proceso, pero reduciendo el número por el cual se divide y multiplica en este caso sería 10, esto se repite sucesivamente salvo en el último número ya que no es necesario hacer cuentas para obtener el 1, simplemente restar 20 (resultado de dividir y multiplicar) a 21. Este algoritmo está pensado para números de hasta 999 ya que para números mayores simplemente habría que agregar una subrutina que cuente los exponentes, pero dado que no se traducen más de 1000 letras no era necesario implementarlo.
Para la traducción de palabra clave simplemente fue utilizada una función la cual traduce las letras al número correspondiente y luego se utiliza el mecanismo convencional de desencriptamiento. Es importante tener en cuenta que para llamar a esta función se debe tener extraído el mensaje previamente, es por eso que se llama a extraer el mensaje, clave y palabra secreta desde el main.
El programa también cuenta con una subrutina para verificar que la paridad sea correcta, esta simplemente compara la paridad que recibe por teclado con la paridad real (cantidad de letras procesadas, incluye símbolos).

El programa también asumirá que los datos ingresados son previamente verificados y no contienen error en su sintaxis, Se asume que todos los datos son verificados atreves de una

función ficticia y que se encuentra en el programa, Esta función toma el input y un código que indica si se quiere codificar/decodificar con clave o si se quiere decodificar con palabra secreta. Esta devuelve si la palabra está escrita dentro de los estándares solicitados dando como resultado la letra c/d (para que se inicie el proceso ya programado). En el caso contrario se devuelve la letra correspondiente para que se genere el loop y el programa pida nuevamente que se ingrese por teclado.

Subrutinas:

cleanRegisterSub: Inicializa en 0 los registros {r0, r1, r2, r3}.

cleanRegister: Inicializa en 0 todos los registros.

write: Recibe por r3. Recibe por r3 la pos en memoria de una cadena. Deja de escribir cuando la cadena es nula (Termina en 0).

read: Recibe por teclado y almacena en inputKey.

lengthZero: Recibe r3 y devuelve: r0. Cuenta los caracteres de una palabra dada. Deja de recorrer al toparse con 0.

lengthtS: Recibe r3 y devuelve: r0. Length pero corta de recorrer en 0 no en ~.

toInt: Recibe r3 y devuelve: r0. Recibe un la pos en memoria de una cadena .asciz y devuelve un int.

saveParameters: Guarda en el Stack {r0, r1, r2, r3}.

loadParameters: Guarda en {r0, r1, r2, r3} los primeros 4 datos del Stack.

saveRegisters: Guarda en el Stack {r4, r5, r6, r7, r8, r9, r10, r11}.

loadRegisters: Guarda en {r4, r5, r6, r7, r8, r9, r10, r11} los primeros 8 datos del Stack.

printMenu: Recibe r3. Recibe la dirección del menú. Escribe en pantalla el menú y guarda en inputKey lo escrito por teclado.

toIntOne: Recibe r3 y devuelve r0. Recibe un número en asciz y lo transforma a int.

paridad: Recibe r0 y devuelve r0. Recibe un número y devuelve '0' si es par o '1' si es impar (Devuelve el carácter)

limpiar: Limpia la pantalla.

toStrOne: Recibe r3 y devuelve r0. Recibe un número entre 1 y 9 y lo convierte en str. Lo devuelve por r0

toStr999: Recibe r3. Recibe un número hasta 999 y lo convierte en str los guarda en procesadas.

vaciarPantalla: Imprimer 100 saltos de línea para empujar el texto hacia abajo

se_validar: Valida que lo que está en la direccion inputKey: sea válido. Se asume que esta función está en el programa.

se_extraerMensaje: Extrae el mensaje dado por el usuario.

se_extraerClave: Extrae la clave dada por el usuario.

se_extraerOpcion: Extrae la opción dada por el usuario.

se_decodificar: Codifica el mensaje.

se_codificar: Decodifica el mensaje.

se_extraerPSecreta: Convierte la palabra secreta en número.

se_verificarParidad: Recibe r0 y devuelve r0. Verifica que la paridad del mensaje y la paridad dada sean iguales, Si son iguales se guarda en r0 el numero 0.

Para la organización a la hora de programar se utilizó una tabla Exel la cual especificaba todas las subrutinas (Su nombre, lo que recibe, lo que devuelve, lo que hace, si está terminada, lo que le falta y por último si ya se probó). También se utilizó un cuadro con todas las etiquetas a memoria que se usaron (Nombre, “tipo” y descripción).

SUBRUTINA	RECIBE	DEVUELVE	DESCRIPCION	¿TERMINADA?	¿QUE FALTA?	¿PROBADA?
cleanRegisterSub	-	-	Inicializa en 0 los registros (r0, r1, r2, r3)	Si	-	s
cleanRegister	-	-	Inicializa en 0 todos los registros	Si	-	s
write	r3	-	Recibe por r3 la pos en memoria de una cadena. Deja de escribir cuando la cadena es nula (Termina en 0)	Si	-	s
read	-	-	Recibe por teclado y almacena en inputKey:	Si	-	s
writeRead	-	-	Imprime por pantalla inputKey:	Si	-	s
length	r3	r0	Recibe la posicion en memoria de un .asciz y devulve la cantidad de caracteres que este posee. Corta en \n	Si	-	s
lengthZero	r3	r0	Length pero corta de recorrer en 0 no en \n	Si	-	s
lengthS	r3	r0	Length pero corta de recorrer en 0 no en “	Si	-	s
toint	r3	r1	Recibe un la pos en memoria de una cadena .asciz y devuelve un int	Si	-	s
saveParameters	-	-	Guarda en el Stack (r0, r1, r2, r3)	Si	-	s
loadParameters	-	-	Guarda en (r0, r1, r2, r3) los primeros 4 datos del Stack	Si	-	s
saveRegisters	-	-	Guarda en el Stack (r4, r5, r6, r7, r8, r9, r10, r11)	Si	-	s
loadRegisters	-	-	Guarda en (r4, r5, r6, r7, r8, r9, r10, r11) los primeros 8 datos del Stack	Si	-	s
printMenu	r3	-	Recibe por r3 la direccion del menu. Escribe en pantalla el munu y guarda en inputKey: los escrito por teclado	Si	-	s
toIntOne	r3	r0	Recibe por r3 un numero en ascii y lo tranforma a int	Si	-	s
paridad	r0	r0	Recibe un numero y devuelve '0' si es par o '1' si es impar (Devuelve el caracter)	Si	-	s
limpiar	-	-	Limpia la pantalla	Si	-	s
toStrOne	r3	r0	Recibe un numero entre 1 y 9 y lo conviente en str. Lo devueve por r0	Si	-	s
toStr999	r3	-	Recibe un numero hasta 999 y lo convierte en str los guarda en procesadas:	Si	-	s
vaciarPantalla	-	-	Imprimer 100 saltos de linea para empujar el texto hacia abajo	Si	-	s
SUBRRUTINAS ESPECIFICAS	**RECIBE**	**DEVUELVE**	**DESCRIPCION**	**¿TERMINADA?**	**¿QUE FALTA?**	**¿PROBADA?**
se_validar	-	-	Valida que lo que esta en la direccion inputKey: sea valido	Falta Incluir	Se asume que se ingresa todo bien	
se_extraerMensaje	-	-	Extrae el mensaje dado por el usuario	Si	-	s
se_extraerClave	-	-	Extrae la clave dada por el usuario	Si	-	s
se_extraerOpcion	-	-	Extrae la opcion dada por el usuario	Si	-	s
se_decodificar	-	-	Codifica el mensaje	Si	-	s
se_codificar	-	-	Decodifica el mensaje	Si	-	s
se_extraerPSecreta	-	-	Convierte la palabra secreta en numero	Si	-	s
se_verificarParidad	r0	r0	Verifica que la paridad del mensaje y la paridad dada sean iguales, Si son iguales r0 vale 0	Si	-	s

ETIQUETA	TIPO	DESCRIPCION
convertida	.asciz	Palabra que fue encriptada/desencriptada
clave	.asciz	Donde se extrae la clave
inputKey	.asciz	Ingreso de Teclado
procesadas	.asciz	Tengo el numero de la cantidad procesadas
paridadMjs	.byte	Se guarda la paridad que me dieron
mensajeConvertida	.asciz	"Tu nuva palabra es: "
mensajeReser	.asciz	"PRESIONE ENTER PARA CONTINUAR"
cls	.asciz	Almacena la cadena para limpiar la pantalla
mensajeProcesadas	.asciz	"La cantidad de letras procesadas son:
enter	.asciz	"/n"
vParidad	.asciz	Almacene el ascii de la paridad que corresponde

Como conclusión, este trabajo pretende utilizar todas las herramientas obtenidas por la materia dando como resultado un trabajo funcional, el cual está conformado por una estructura mayoritariamente subrutinada. Creemos haber logrado lo que se pretende del trabajo.