Tarea 1
Encriptación
Entrega: 11 de abril a las 21:00

1. Introducción

En este laboratorio deberán que desarrollar un programa que realice encriptación y desencriptación usando algoritmos básicos. Para esto deberán utilizar los siguientes métodos:

1.1. Cifrado por Desplazamiento

El cifrado por desplazamiento consiste en desplazar un carácter α por otro cuyo código ASCII sea $\beta + \alpha$. Para efectos de nuestro programa el desplazamiento lo realizaremos a nivel de **chars**, por lo que el desplazamiento es cíclico alrededor del numero 255 (max char). Ejemplo:

```
Desplazamiento de 4 caracteres:
original:   Me gusta programar en C
Codificado: Qi\$kywxe\$tvskveqev\$ir\$G
```

1.2. Negación de Bits

Este tipo de cifrado consiste en negar los bits del mensaje, obteniendo caracteres completamente distintos. Ejemplo:

```
Negaremos la letra a:
a:  0110 0001
~a: 1001 1110
```

1.3. XOR

Este tipo de cifrado consiste en realizar la operación XOR sobre el mensaje con un char conocido, esta operación se aplica sobre todo el texto. Ejemplo:

```
Haremos XOR a la letra a con la letra b:
a:   0110 0001
b:   0110 0010
a^b: 0000 0011
```

1.4. Invertir Caracteres

Este tipo de cifrado consiste en invertir cada letra del texto con su opuesta en el alfabeto, sin contar la ñ, las letras con acento, ni los números o símbolos. Si existe diferencia entre mayúsculas y minúsculas.

```
Invertiremos las letras:
original:   aAbz12 ñá
Codificado: zZya12 ñá
```

2. Tarea

Su objetivo en este laboratorio es crear un programa en C que:

2.1. Semana 1

1) Pregunte si desea encriptar o desencriptar
2) Pida al usuario una frase para realizar la accion. (*Hint: investigue la funcion gets.*)
3) Pida al usuario la cantidad de caracteres para hacer el desplazamiento
4) Pida al usuario un char para realizar XOR **Encriptar**
   a) Realice el desplazamiento
   b) Realice la negación de bits
   c) realice el XOR

   Desencriptar

   a) realice el XOR
   b) Realice la negación de bits
   c) Realice el desplazamiento inverso
5) Imprima la frase encriptada/desencriptada

Cualquier error que se produzca durante la ejecución del programa debe ser impreso en la consola y el programa debe terminar.

2.2. Semana 2

Modifique su programa, para que ahora en vez de pedirle una frase al usuario, pídale un archivo y trabaje sobre dicho archivo, además, antes de realizar el desplazamiento, primero deberá realizar la inversión de caracteres. El resultado en vez de imprimirlo, escríbalo a un archivo nuevo.

1) **Pida al usuario un archivo.**
2) Pregunte si desea encriptar o desencriptar
3) Pida al usuario la cantidad de caracteres para hacer el desplazamiento
4) Pida al usuario un char para realizar XOR **Encriptar**
   a) **Realice inversión de caracteres**
   b) Realice el desplazamiento
   c) Realice la negación de bits
   d) realice el XOR

   Desencriptar

   a) realice el XOR
   b) Realice la negación de bits
   c) Realice el desplazamiento inverso
   d) **Realice inversión de caracteres**
5) Escriba el archivo encriptado/desencriptado a un nuevo archivo

Ahora cualquier error que se produzca durante la ejecución del programa debe ser informado por el **estandar error**, en lugar de la consola, y el programa debe terminar.

3. Consideraciones Generales

Su programa debe ser robusto frente a datos que no corresponden. Esto quiere decir que su programa no debe caerse en caso de que el usuario haga operaciones matemáticas no válidas (p.ej., dividir por cero si fuera el caso). Sin embargo, pueden asumir que el usuario será amigable, en el sentido de que cuando le soliciten un número este no ingresará letras, por ejemplo.

4. Evaluación y Entrega

Formato de entrega: Subir un solo archivo comprimido con todo el código fuente de su programa al módulo de tareas de la página del curso en https://saf.uandes.cl/ing/, con el nombre de archivo "NOMBRE-APELLIDO-Tarea1.tar.gz" el cual deberá contener los archivos de código fuente en C (archivos .c y .h), reemplazando "NOMBRE" y "APELLIDO" según corresponda (ej. ALAN-TURING-Tarea1.tar.gz). Los archivos compilados no serán tomados en cuenta, si se llega a subir solo un archivo compilado, este será ignorado, y será evaluado con nota 1.
En el momento de compilación, se recomienda incluir las siguientes flags al compilador:

```
-Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self
```

Junto a los archivos de código base, debe venir incluido un archivo **Makefile** el cual permita realizar la compilación de manera automática con el comando **make**, este debe incluir las siguientes flags:

```
 -Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self
```

Al compilar se debe generar un ejecutable llamado "tarea_1". Si este paso falla y se debe realizar la compilación de forma manual, o el ejecutable no se llama como se indica se procederá a descontar 1 punto de la nota obtenida.

Aquellas tareas que no compilen de la forma normal (gcc -o salida entrada) serán evaluadas con nota 1. Las que compilen, pero se caigan durante la ejecución, serán evaluadas con nota máxima 3.

Su programa será evaluado con múltiples casos de prueba y deberá ser capaz de ejecutarlos todos de manera correcta. De fallar en algún caso de prueba serán descontados los puntos correspondientes a dicho caso.

5. Consideraciones de Trabajo

El trabajo en las tareas debe ser hecho solo por su grupo, por lo que cuide su tarea para que no sea copiada parcial o íntegramente por otros. Todas las tareas entregadas serán comparadas por un sistema automático de detección de plagios. Cualquier copia será penalizada, recibiendo el mismo castigo tanto quien copia como quien permite que le copien. También es considerada copia cualquier ayuda externa recibida directamente en la tarea, sin importar si proviene de un alumno del curso, de la universidad, o de otro lugar. El castigo será establecido por el Consejo de la Facultad, siendo como mínimo un 1,0 de promedio en el curso.