Programa Para Calcular La Seguridad De Una Contraseña.

Stalin Maza, Mishel Centeno

Escuela de Formacion de Tecnologos,

Escuela Politecnica Nacional

Ecuador

[stalinct97@gmail.com](mailto:stalinct97@gmail.com)

[mishelcenteno96@hotmail.com](mailto:mishelcenteno96@hotmail.com)

# *INTRODUCCIÓN.*

Este documento contiene todos los pasos necesarios para la elaboración de un programa en el cual ingresamos una clave cualquier y nos indicara el nivel de seguridad de esta además de los elementos que componen dicha clave.

Asi podremos entender mejor como se compone una clave y la mejor manera de hacerla lo más segura posible.

Para esto calcularemos el tiempo estimado que le tomaría a cualquier computadora, realizar la decodificación de la clave, tomando en cuenta diferentes factores como por ejemplo: el rendimiento de la maquina al realizar las instrucciones, el tipo de caracteres ingresados, la longitud de la cadena.

# *DESARROLLO DE CONTENIDOS.*

1. *Librerías.*

Las bibliotecas contienen el código objeto de muchos programas que permiten hacer cosas comunes, como leer el teclado, escribir en la pantalla, manejar números, realizar funciones matemáticas, etc.

Para incluir una biblioteca debemos escribir:

#Include <”nombre de la biblioteca”>

Para este programa necesitaremos usar varias librerías las cuales son de gran utilidad para realzar algunas funciones, sabiendo que el sistema operativo utilizado para este ejercicio es Ubuntu.

La primera y principal biblioteca a utilizar será la de “<iostream>” la cual nos permite utilizar algunas cosas importantes como los “cout” y los “cin” para pedir y recibir datos o valores.

Otra librería utilizada es la de “<time.h>”, la cual nos permitirá utilizar la función clock, que veremos más adelante en que la utilizaremos.

Tambien utilizamos la librería” <string.h>” la cual nos permitirá usar la función “strlen” y así determinar la longitud del texto ingresado.

La librería” <fstream>” tambien será utilizada ya que nos servirá para escribir las permutaciones realizadas en un archivo txt.

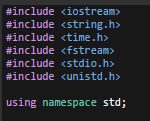


Ilustración 1.Librerias Utilizadas.

## Funciones.

Son un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor. Las funciones pueden recibir o no parámetros que modifiquen su funcionamiento. Las funciones son utilizadas para reducir la cantidad de código en la función principal. Cuando una función es invocada se le pasa el control a la misma, una vez que esta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.

### Función Main.

Esta función es la principal del programa, donde se recibirán la contraseña y la que mostrara los resultados.

Además se obtendrá la longitud de la cadena con la función strlen de la librería string.h, esto nos servirá mucho para los demás cálculos.

Dentro de ella tambien usaremos algunas funciones más, las cuales serán usadas de manera recursiva para optimizar el programa.

Aquí tambien implementaremos, que no se pueda observar la contraseña que escribimos por motivos de seguridad.

Para esto utilizamos la librería <”unistd.h”>, que nos permite utilizar la función password, que recoge lo que ingresa el usuario y después desactiva el eco de la consola, para que no se muestre en la consola lo que escribimos.

1. *Definición de las Variables.*

Para realizar el programa utilizaremos algunas variables y arreglos que nos permitirá que el programa realice todas las operaciones necesarias.

Entre ellas usaremos variables de tipo entero, de caracteres, arreglos de caracteres, unas variables long long int para poder almacenar números grandes.

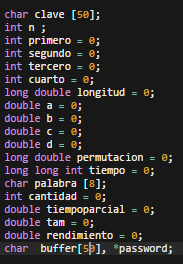


Ilustración 2. Lista de las Variables Utilizadas.

### Función Comparación.

Esta función es recursiva, que quiere decir que esta fuera del main y necesita primero declararla como cualquier otra variable.

Esta función recibirá un arreglo de caracteres que contiene la clave ingresada y la longitud del texto ingresado.

Lo que realizara es una comparación entre la clave ingresada y los diferentes conjuntos de letras, números y símbolos para determinar que elementos de estos conjuntos se encuentran en la palabra ingresada.

La Funcion es de tipo entero, ya que al finalizar me devuelve el número de comparaciones que hizo.

Dentro de los paréntesis, van los parámetros que reciben, los cuales son un arreglo de tipo char que contiene la clave ingresada, y recibe una variable entera que contiene la longitud de la cadena.

Para realizar esto utilizaremos dos bucles for.

El bucle for principal inicializara con la variable i = 0 y se repetirá mientras esta vari abre que aumenta en cada repetición sea menor que el valor de la longitud de la cadena, este bucle determinara el número de veces que se compararan los conjuntos.

El Segundo Bucle for inicializara una variable j = 0 y terminara cuando la variable “j” que aumentara en 1, cada vez que se repita el ciclo, sea menor que el tamaño del conjunto donde están ubicados los tipos de caracteres. Este bucle comparara letra por letra, el primer conjunto con el segundo y al final nos devolverá el número de caracteres ingresados que pertenece a dicho conjunto.

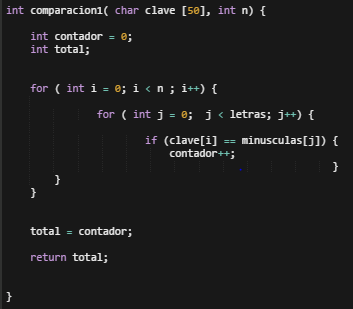


Ilustración 3.Codigo de la Funcion Comparación.

### Función Potencia.

Esta función me permitirá calcular el número de permutaciones o instrucciones que se realizaran, esta recibirá dos valores los cuales serán la base o el número de caracteres que se utilizara para las permutaciones y la longitud o número de elementos que deberán ser permutados de acuerdo a la palabra ingresada.

El Prototipo de esta función quedaría así:

double potencia (double longitud, double permutación);

Las instrucciones que realiza serian lo siguiente:

Si el valor de la permutación es igual a cero me retorna 1, caso contrario me retorna la longitud multiplicada por la misma función pero menos 1, esta se llamara a si misma hasta que llegue un punto en el cual al llamarse de nuevo, la permutación sea igual a cero y termine.

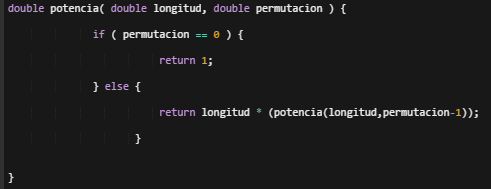


Ilustración 4.Función Permutación.

### Función Conversión.

Esta función me hará la transformación del tiempo que se nos da en milisegundos a valores más comprensibles como segundos o horas ya que no sería correcto mostrar que se demoraría 31536000 segundos en vez de decirle que se demoraría un año.

El Prototipo de esta función seria:

double conversion ( long long int tiempo)

Esto va porque la función recibe un valor entero muy grande y entonces agregarle los “long” para que este pueda guardar valores más grandes.

De acuerdo a esto, realizando una serie de comparaciones, según el tiempo que nos mande, nos devuelve el tiempo que se demoraría en realizar el total de permutaciones para descifrar la clave.

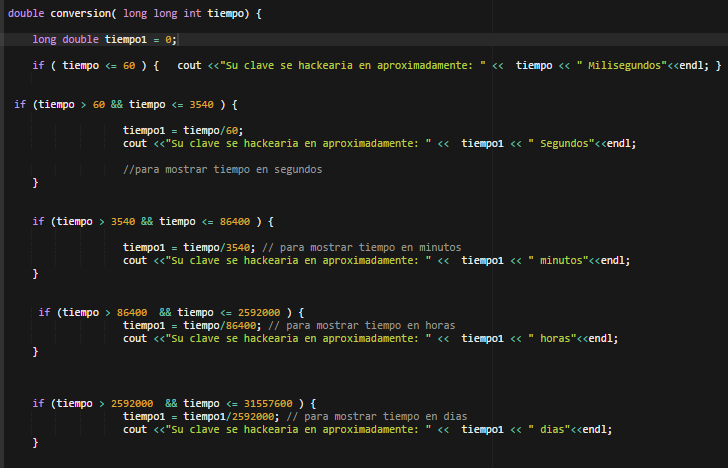


Ilustración 5.Codigo de la Funcion Conversion.

### Conjuntos.

Son arreglos en las cuales guardaremos de acuerdo a cada tipo, las letras minúsculas, mayúsculas, números o símbolos.

En este caso usamos arreglos de tipo char para guardar estos conjuntos.

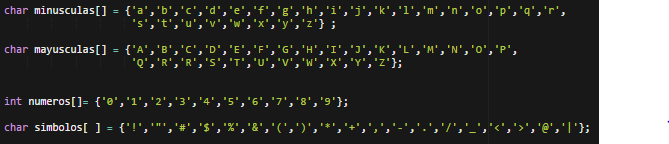


Ilustración 6.Codigo de los Arreglos de los Conjuntos Utilizados.

### Calculo del Número de Permutaciones.

Para realizar esto utilizaremos los resultados de las funciones comparación, los cuales me darán la longitud de la cadena y el número de caracteres.

El número de caracteres se determinara mediante algunos “if” los cuales si la palabra ingresada contiene alguno de los conjuntos entonces sumara el número de elementos de estos y estos serán el número de caracteres.

La Longitud en cambio será el tamaño de la cadena que se obtiene con la función strlen.

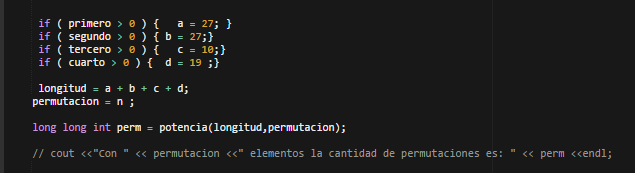


Ilustración 7.Codigo para calcular el número de permutaciones.

1. *Impresión de Permutaciones en archivo txt.*

Para realizar las impresiones de las permutaciones en un archivo txt, primero debemos crear un archivo txt en nuestro programa para lo cual sintaxis de esto es:

ofstream “nombre que guardara datos” (“nombrearchivo.txt”);

después de esto debemos colocar donde vamos a querer que se impriman las permutaciones, para lo cual debemos mandar el parámetro del nombre que guardara los datos, entonces en la impresión quedaría más o menos así:

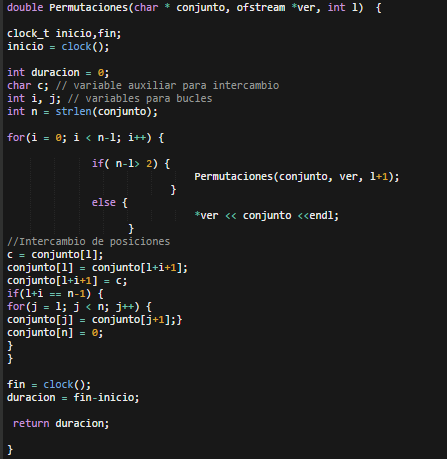


Ilustración 8.Impresion de las Permutaciones en un fichero txt.

### Calculo del tiempo.

Para realizar el cálculo de tiempo debemos usar la librería “<time.h>”, que nos permite usar la función “clock” que nos permite retornar el tiempo de ejecución de un programa.

Esto se realiza primero definiendo el inicio del reloj, para lo cual debemos usar la siguiente sintaxis:

clock “variable inicio” “variable fin”;

Después igualamos la variable de inicio al reloj.

Y en la parte del programa donde queremos que acabe de contar el tiempo, igualamos la variable de fin al reloj.

Para ver el tiempo solo debemos imprimir en la pantalla, la variable de fin menos la variable de inicio.

Ejemplo:

cout <<"tiempo " << fin-inicio << endl;

### Calculo del Rendimiento de la Maquina.

Para calcular el rendimiento de la maquina utilizaremos una formula sencilla, qué es:

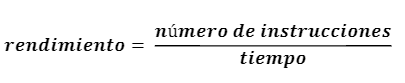


Ilustración 9.Fórmula para calcular el rendimiento.

Entonces para esto observamos que necesitaremos el número de instrucciones y el tiempo.

Estos ya los obtenemos con las funciones anteriores, por lo cual solo dividiremos el número de instrucciones ejecutadas durante un cierto tiempo, y ese será el rendimiento de la máquina.

1. *Calculo del tiempo requerido para hackear la clave.*

Para realizar este proceso solo debemos dividir el número de instrucciones o permutaciones que necesitaría hacer el computador sobre el rendimiento de la máquina.



Ilustración 10.Fórmula para Calcular el tiempo necesario para descifrar la clave.

1. *Función de seguridad.*

Esta función recibirá el tiempo que se demoraría en realizar las permutaciones totales de la palabra ingresada y nos permitirá ver el tiempo que se demora en descifrar dicha clave para eso tiene una variable del tiempo que tenemos en segundos que logramos convertir de milisegundos y vamos comparando un if, donde si ese valores igual a la opción: Si el tiempo es mayor e igual a 3540 lo que se ejecute será que su contraseña es muy débil Si el tiempo que da es mayor que 3540 pero menor que 86400 entonces su contraseña es débil Si el tiempo > 86400 && tiempo <= 2592000 entonces se demorara ese tiempo pero en días transformado esa cantidad dividendo para 86400. Si el tiempo > 86400 && tiempo <= 2629800 entonces se su contraseña es normal. Si el tiempo > 2629800 && tiempo <= 31557600 entonces se su contraseña es rígida. Si el tiempo > 31557600 && tiempo <= 315576000 entonces se su contraseña es fuerte. Si el tiempo > 315576 entonces se su contraseña es súper fuerte. Mediante esta comprobación podemos verificar si su clase tiene una buena seguridad



Ilustración 11.Niveles de Seguridad.

# *Conclusiones.*

El programa tiene utilidad en la vida diaria ya que ayuda a solucionar un problema al genera contraseñas que sea fuertes así impide que cualquier persona pueda acedera su cuenta y obtener sus datos privados. La implementación sirve para poder poner en práctica las herramientas aprendidas en clases como: cadenas, bucles, arreglos, funciones, librerías, etc. Se puede realizar un programa de acuerdo al rendimiento de la máquina con la aplicación que estemos realizando y podremos tener una mejor ejecución de este. Permite que pongamos en práctica nuestra mente para poder resolver los problemas usando la lógica al analizar las diferentes respuestas

# *Bibliografía.*

Vitor. Carlos, (2011) Los tipos de variables. [Online]. Available: <http://es.ccm.net/faq/3169-las-variables-en-c>  
(2006).Como usar time.h. Obtenido de El hacker: <http://foro.elhacker.net/programacion_cc/c_como_usar_timeh-t112394.0.html>  
Hyde. (2014).Funciones Cadenas De Caracteres [Online]. Available:<http://www.aprenderaprogramar.com/foros/index.php?topic=930.0>  
(2009). Función Strlen Obtenido de Geek Programador: <http://geekprogramador.blogspot.com/2009/07/funcion-strlen.html>  
(2015).Permutaciones en C++.Obtenido del Hater: <https://foro.elhacker.net/programacion_cc/permutaciones_en_c-t429978.0.html>  
Venegas. (2013). Permutaciones en c++.Obtenido de: <http://www.mediafire.com/download/gdh111epy3nps4y/Codigos.txt>