



CÁTEDRA DE COMPUTACIÓN – CICLO 1 2020

FIUNA

EJERCICIO

Elaborar un programa en C++ que permita codificar una cadena de caracteres introducida desde teclado, según el esquema propuesto:

- Cada letra entre A y F deberá ser cambiada por la letra que está dos posiciones más adelante en el alfabeto (ejemplo: A por C); lo mismo aplica para las minúsculas correspondientes.
- Cada letra desde la G hasta antes de la S, deberá ser cambiada por la que está tres posiciones más adelante (ejemplo: G por J); lo mismo aplica para las minúsculas correspondientes.
- Cada letra desde la S en adelante (hasta Y), por la que está una posición más adelante (ejemplo: S por T); lo mismo aplica para las minúsculas correspondientes.
- Para el caso de la letra Z, cambiarla por la letra A; lo mismo aplica para las minúsculas correspondientes.
- En el caso de los números del 0 al 8, reemplazar un número dado por el consecutivo (ejemplo: el 1 por 2, etc.), y en caso del dígito 9 reemplazarlo por un 0.

Utilizar una función para la codificación, y mantener inalterados todos los demás caracteres que no sean letras o números (puntos, comas, etc.).

Imprimir finalmente el resultado de la codificación.

EJERCICIO DESARROLLADO – SEMANA 11 “CADENAS”

RESOLUCIÓN

Como siempre, deberemos declarar en la primera parte del código, las líneas necesarias para permitir el flujo de entrada y salida del programa, así como cada función (distinta a la main) que deseemos crear para cumplir con el objetivo del programa.

En este caso, deberemos desarrollar una función que realice la codificación, y sólo esa función es la necesaria (opcionalmente se puede crear otra que cuente el tamaño de la cadena ingresada en realidad, pero nosotros por practicidad realizaremos esta tarea dentro de la función de codificación misma).

Se debe recordar también que hay un montón de aproximaciones posibles a realizar para resolver este ejercicio, pero esta es la que encuentro más didáctica a los fines de nuestra materia.

Un esquema general, que cumple con lo solicitado en el enunciado en cuanto a funciones (y en el cual apenas se prototipa o declara la función, y aún no se la crea) es:

```
#include<iostream>
using namespace std;

void codificacion(char cad[]);

int main (){
    //En este caso, solo por practicidad, limitamos el tamaño de la cadena a 50 caracteres.
    char cad[50];
    cout<<"Ingrese la palabra o frase a codificar:\n";
    cin.getline(cad,50);
    codificacion(cad);
    //La cadena es "la misma" es decir se hacen los cambios sobre la cadena en sí, al utilizarla
    //como argumento de la función.
    cout<<"\nLa palabra o frase ya codificada es:\n";
    cout<<cad<<endl;
    return 0;
}
```

En dicha porción de código se tienen en cuenta todos los conceptos que se fueron presentando en la semana para el ingreso de cadenas (*cin.getline*) así como para la impresión de las mismas (con el *cout* tradicional), así como la sintaxis para el envío de cadenas a una función auxiliar para trabajar con las mismas, en este caso en particular, con la codificación, creando al efecto la función “codificacion” de tipo void, ya que trabajará sobre la cadena en sí teniéndola como argumento de llamada (modificando así sus elementos) y no devolverá valor alguno.

EJERCICIO DESARROLLADO – SEMANA 11 “CADENAS”

Una proposición de función de codificación que cumpla con todos los ítems que fueron especificados en el enunciado (y que aquí copiamos para mayor detalle a la hora de desarrollar la función es):

- Cada letra entre A y F deberá ser cambiada por la letra que está dos posiciones más adelante en el alfabeto (ejemplo: A por C); lo mismo aplica para las minúsculas correspondientes.
- Cada letra desde la G hasta antes de la S, deberá ser cambiada por la que está tres posiciones más adelante (ejemplo: G por J); lo mismo aplica para las minúsculas correspondientes.
- Cada letra desde la S en adelante (hasta Y), por la que está una posición más adelante (ejemplo: S por T); lo mismo aplica para las minúsculas correspondientes.
- Para el caso de la letra Z, cambiarla por la letra A; lo mismo aplica para las minúsculas correspondientes.
- En el caso de los números del 0 al 8, reemplazar un número dado por el consecutivo (ejemplo: el 1 por 2, etc.), y en caso del dígito 9 reemplazarlo por un 0.

Utilizar una función para la codificación, y mantener inalterados todos los demás caracteres que no sean letras o números (puntos, comas, etc.).

```
void codificacion(char cad[]){
    int i=0,tam;
    //Con esta porción contamos la cantidad de elementos de la cadena.
    while (cad[i]!='\0') i++;
    tam=i;
    //Aquí hacemos la codificación específica según los requerimientos, caso por caso.
    for (i=0;i<tam;i++){
        if ((cad[i]>='A' && cad[i]<='F') || (cad[i]>='a' && cad[i]<='f')) cad[i]=cad[i]+2;
        if ((cad[i]>='G' && cad[i]<='S') || (cad[i]>='g' && cad[i]<='s')) cad[i]=cad[i]+3;
        if ((cad[i]>='S' && cad[i]<='Y') || (cad[i]>='s' && cad[i]<='y')) cad[i]=cad[i]+1;
        //La diferencia entre la codificación ASCII entre la 'A' y la 'Z' y entre la 'a' y la 'z' es
        //la misma: 25, siendo el valor ASCII de las "zetas" mayor al de las "as".
        if (cad[i]=='Z' || cad[i]=='z') cad[i]=cad[i]-25;
        if (cad[i]>='0' && cad[i]<='8') cad[i]=cad[i]+1;
        if (cad[i]=='9') cad[i]='0';
        //Si no entra en ninguno de los casos de arriba es que es un caracter especial, y como no
        //se hacen cambios, se mantiene inalterado dentro de su posición en la cadena original.
    }
}
```

Básicamente, se cuenta la cantidad de caracteres útiles (antes del “cero terminador” ‘\0’ que indica la finalización de la cadena, como se vio extensamente en las lecciones de la semana) y luego se analiza caso a caso en qué parte de lo solicitado en el enunciado cae (rangos de letras o números), según el carácter, y de hecho como aclarado en el código, si no “cae” en ninguno de los esquemas de codificación implementados por las estructuras “if” entonces simplemente, ese carácter en cuestión no cambia dentro de la cadena original.

Al término de la ejecución de la función, la cadena “original” (en realidad estamos trabajando con la misma cadena) se vio sustituida en cada carácter, por una equivalente con las instrucciones de codificación.

EJERCICIO DESARROLLADO – SEMANA 11 “CADENAS”

Aclaración: como las estructuras “if” tienen una sola línea útil (la línea de codificación de cada caso) no utilizamos llaves para abrirlas y cerrarlas. Por otro lado, como se vio en clase de teoría, cada caracter tiene una representación numérica en el código ASCII universal, de allí que se resten o se sumen y se obtengan las letras correspondientes; la aclaración de las “z-Z” y las “a-A” está hecha en el código.

Finalmente, juntando las dos porciones de código tenemos el programa completamente funcional, que es el que abajo juntamos para practicidad:

```
#include<iostream>
using namespace std;

void codificacion(char cad[]);

int main (){
    //En este caso, solo por practicidad, limitamos el tamaño de la cadena a 50 caracteres.
    char cad[50];
    cout<<"Ingrese la palabra o frase a codificar:\n";
    cin.getline(cad,50);
    codificacion(cad);
    //La cadena es "la misma" es decir se hacen los cambios sobre la cadena en sí, al utilizarla
    //como argumento de la función.
    cout<<"\nLa palabra o frase ya codificada es:\n";
    cout<<cad<<endl;
    return 0;
}

void codificacion(char cad[]){
    int i=0,tam;
    //Con esta porción contamos la cantidad de elementos de la cadena.
    while (cad[i]!='\0') i++;
    tam=i;
    //Aquí hacemos la codificación específica según los requerimientos, caso por caso.
    for (i=0;i<tam;i++){
        if ((cad[i]>='A' && cad[i]<='F') || (cad[i]>='a' && cad[i]<='f')) cad[i]=cad[i]+2;
        if ((cad[i]>='G' && cad[i]<='S') || (cad[i]>='g' && cad[i]<='s')) cad[i]=cad[i]+3;
        if ((cad[i]>='T' && cad[i]<='Y') || (cad[i]>='t' && cad[i]<='y')) cad[i]=cad[i]+1;
        //La diferencia entre la codificación ASCII entre la 'A' y la 'Z' y entre la 'a' y la 'z' es
        //la misma: 25, siendo el valor ASCII de las "zetas" mayor al de las "as".
        if (cad[i]=='Z' || cad[i]=='z') cad[i]=cad[i]-25;
        if (cad[i]>='0' && cad[i]<='8') cad[i]=cad[i]+1;
        if (cad[i]=='9') cad[i]='0';
        //Si no entra en ninguno de los casos de arriba es que es un caracter especial, y como no
        //se hacen cambios, se mantiene inalterado dentro de su posición en la cadena original.
    }
}
```

EJERCICIO DESARROLLADO – SEMANA 11 “CADENAS”

Si compilamos y ejecutamos el código fuente propuesto, tendremos un programa que nos pide que ingresemos una frase cualquiera para aplicar la codificación.

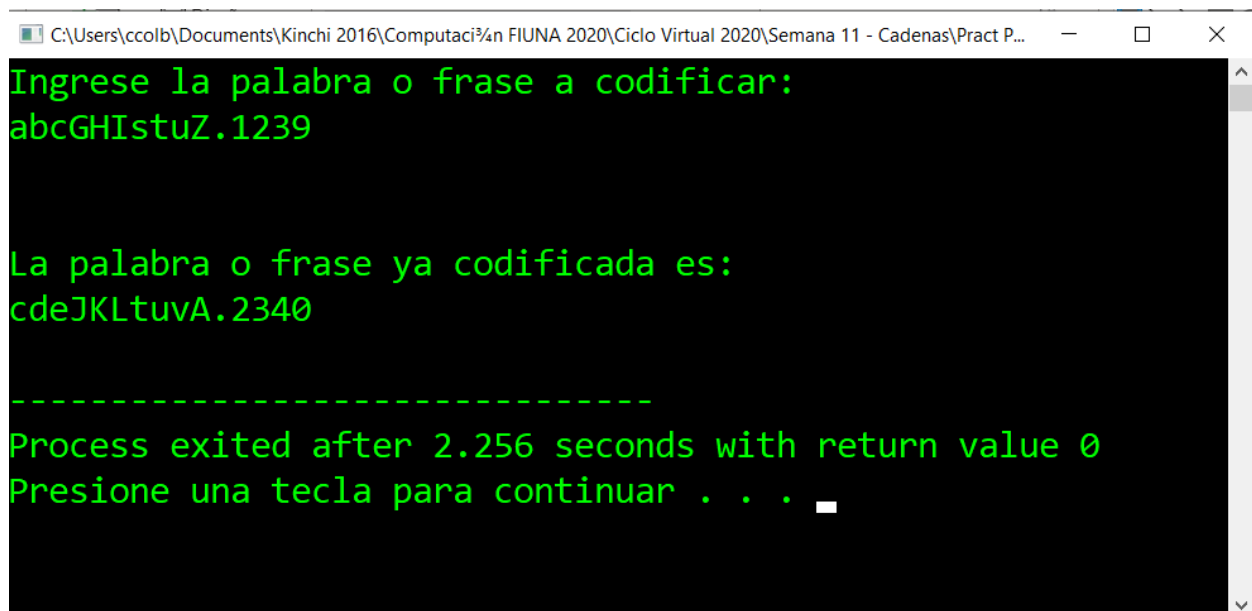
Según los requerimientos solicitados en el enunciado, si ingresamos la siguiente cadena:

abcGHIstuZ.1239

Entonces esperamos obtener el siguiente resultado:

cdeJKLtuvA.2340

Una ejecución del programa nos confirma esto, cuyo resultado en el ejecutable se ve de esta manera:



```
C:\Users\ccolb\Documents\Kinchi 2016\Computaci3n FIUNA 2020\Ciclo Virtual 2020\Semana 11 - Cadenas\Pract P...
Ingrese la palabra o frase a codificar:
abcGHIstuZ.1239

La palabra o frase ya codificada es:
cdeJKLtuvA.2340

-----
Process exited after 2.256 seconds with return value 0
Presione una tecla para continuar . . .
```

Lo cual confirma que el programa funciona como estuvimos diseñándolo para que lo haga.

Damos de esta manera una finalización al ejercicio desarrollado.

Recordar de nuevo que este código presentado fue **una** solución nada más del problema, que cumple con lo solicitado; pero podrían encontrarse varias maneras más de hacerlo.

Si existieran dudas relativas al tema, no duden en realizar las consultas utilizando el foro correspondiente en la sección de desarrollo de la semana en el Aula Virtual.

¡Muchas gracias!

