



INSTITUTO POLITECNICO NACIONAL

INSTITUTO POLITÉCNICO NACIONAL



ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

ANÁLISIS Y DISEÑO DE ALGORITMOS

PRÁCTICA 03

Rodrigo García Mayorga

2012630554

Introducción

El código presentado genera todas las posibles combinaciones de letras asociadas a una secuencia de dígitos, emulando el funcionamiento de un teclado de teléfono antiguo. En estos teclados, cada número del 2 al 9 corresponde a un conjunto específico de letras (por ejemplo, el 2 se asocia con "abc" y el 7 con "pqrs").

Funcionamiento del Código

1. Mapeo de Letras:

Se define un arreglo `mapaTeclado` donde cada posición (del 0 al 9) contiene las letras correspondientes al dígito en un teclado telefónico. Los índices 0 y 1 están vacíos ya que no tienen letras asignadas.

2. Función Recursiva:

La función `generarCombinaciones` se encarga de construir las combinaciones letra por letra:

- Si se ha alcanzado el último índice, se imprime la combinación actual.
- De lo contrario, se obtiene el dígito actual, se validan los rangos permitidos (2-9) y se recorre cada letra correspondiente a ese dígito.
- Se llama recursivamente con el siguiente índice, generando todas las combinaciones posibles.

3. Ejecución Principal:

En la función `main`, se solicita al usuario una cadena de dígitos válidos, se inicializa el array actual para almacenar la combinación en construcción y se llama a la función recursiva para empezar el proceso.

Ejemplo

Si el usuario ingresa "23", las posibles combinaciones generadas serían:

- ad, ae, af
- bd, be, bf
- cd, ce, cf

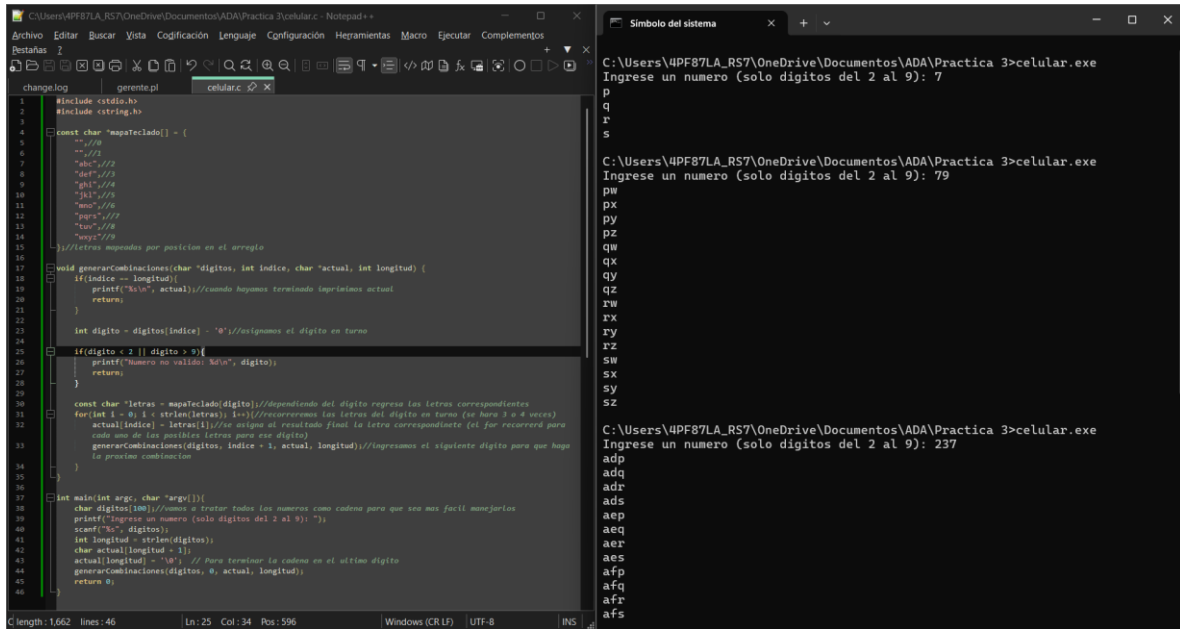
Relación con la Complejidad Algorítmica

La complejidad temporal del algoritmo es **exponencial**, específicamente $O(4^n)$, donde n es la longitud de la cadena de dígitos. Esto se debe a que cada dígito puede generar hasta 4 posibles letras, y estas combinaciones se multiplican recursivamente en cada nivel de profundidad.

En cuanto a la complejidad espacial, el uso de memoria es **lineal**, $O(n)$, ya que el array actual y la profundidad de la pila de llamadas recursivas dependen directamente de la longitud de la entrada.

El código, aunque eficiente en memoria, puede volverse ineficiente para secuencias largas debido a su crecimiento exponencial en tiempo de ejecución.

Capturas



The image shows a C++ program in Notepad++ and its execution in the Windows Command Prompt. The program is a recursive function that generates combinations of digits (2-9) mapped to letters (p-z). It uses a recursive function `generarCombinaciones` to build the combinations. The main function prompts the user to enter a number and then prints the resulting combinations.

```
1 #include <iostream>
2 #include <string.h>
3
4 const char *mapaTeclado[] = {
5     "", //
6     "", //
7     "abc", //
8     "def", //
9     "ghi", //
10    "jkl", //
11    "mno", //
12    "pqrs", //
13    "tuvw", //
14    "xyz" //
15 }; //letras mapeadas por posición en el arreglo
16
17 void generarCombinaciones(char *digitos, int indice, char *actual, int longitud) {
18     if(indice == longitud) {
19         printf("%s\n", actual); //cuando hayamos terminado imprimimos actual
20         return;
21     }
22     int digito = digitos[indice] - '0'; //asignamos el digito en turno
23     if(digito < 2 || digito > 9) {
24         printf("numero no valido: %d\n", digito);
25         return;
26     }
27     const char *letras = mapaTeclado[digito]; //dependiendo del digito repasa las letras correspondientes
28     for(int i = 0; i < strlen(letras); i++) { //recorreremos las letras del digito en turno (se hace 3 o 4 veces)
29         actual[indice] = letras[i]; //se asigna el resultado final la letra correspondiente (el for recorrerá para cada una de las posibles letras para ese digito)
30         generarCombinaciones(digitos, indice + 1, actual, longitud); //ingresamos el siguiente digito para que haga la proxima combinacion
31     }
32 }
33
34 int main(int argc, char *argv[]) {
35     char digitos[100]; //como se tratar todos los numeros como cadena para que sea mas facil manejarlos
36     printf("Ingrese un numero (solo digitos del 2 al 9): ");
37     scanf("%s", digitos);
38     int longitud = strlen(digitos);
39     char actual[longitud + 1];
40     actual[longitud] = '\0'; // Para terminar la cadena en el ultimo digito
41     generarCombinaciones(digitos, 0, actual, longitud);
42     return 0;
43 }
```

Execution output in the Command Prompt:

```
C:\Users\4PF87LA_RS7\OneDrive\Documentos\ADA\Practica 3>celular.exe
Ingrese un numero (solo digitos del 2 al 9): 7
p
q
r
s

C:\Users\4PF87LA_RS7\OneDrive\Documentos\ADA\Practica 3>celular.exe
Ingrese un numero (solo digitos del 2 al 9): 79
pw
px
py
pz
qw
qx
qy
qz
rw
rx
ry
rz
sw
sx
sy
sz

C:\Users\4PF87LA_RS7\OneDrive\Documentos\ADA\Practica 3>celular.exe
Ingrese un numero (solo digitos del 2 al 9): 237
adp
adq
adr
ads
aep
aeq
aer
aes
afp
afq
afr
afs
```

Enlace del código (github):

https://github.com/LowisN/ADA_Practica2.git