Tarea 4. Programación y Algoritmos

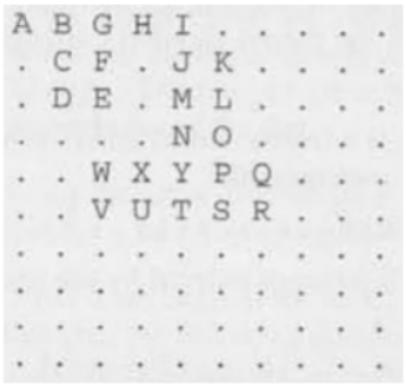
Carlos Giovanny Encinia González

20 de septiembre 2021

Problema 1

Escribir un programa que genere una caminata aleatoria en una matriz de 10x10. El arreglo debe contener inicialmente puntos ´.´, y debe recorrerse basado en el residuo de un número aleatoria (usar srand y rand()) cuyos resultado puede ser 0 (arriba), 1 (abajo), 2 (izq), 3 (der), que indican la dirección a moverse. A) Verificar que el movimiento no se salga del arreglo de la matriz, y B) No se puede visitar el mismo lugar más de una vez. Si alguna de estas condiciones intentar moverse hacia otra dirección; si todas las posiciones están ocupadas, finalizar el programa e imprimir el resultado.

Figure 1: Ejemplo print



´Y´ esta bloqueado por todos los lados, por lo que no puede continuar con la ´Z´. Puede hacer uso de letras o números en orden creciente para mostrar los lugares visitados. Hacer uso de funciones cuyo argumento incluya el arreglo 2D; esto es, una función para inicializar el arreglo, otra para realizar la caminata aleatoria, y otra ara escribir la caminata.

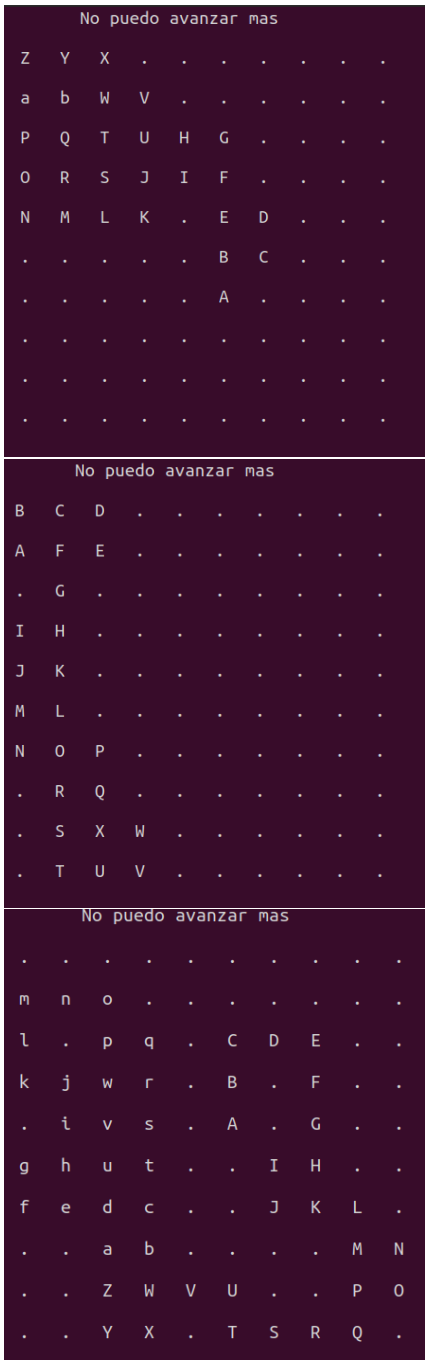
Respuesta

La matriz en la que se tendrá la información de los puntos que se visitan, es creada con memoria dinámica. El programa crea las posiciones iniciales con la función rand(), una vez creadas las posiciones iniciales el programa comienza la caminata aleatoria, existen solo cuatro movimientos permitidos los cuales son los descritos en el problema. Si no se esta permitido el movimiento debido a las restricciones del mismo problema entonces se realiza otro movimiento aleatorio distinto del original, o distinto de movimientos no permitidos anteriores. El resultado se imprime cada vez que hay un cambio de estado en el sistema, esto sumado a funciones de usleep() y system(clear), hacen una pequeña animación del recorrido de la caminata, al final se imprime el recorrido total. Es importante mencionar, que solo se utilizan letras (mayúsculas y minúsculas) para saber el recorrido, y se utiliza el mismo orden de aparición que en el código ASCII. Para ser mas especifico la primer letra del recorrido sera A, una vez terminado el recorrido con el abecedario en mayúsculas, entonces se imprime el abecedario en minúscula pero si se termina esto, de nuevo comienza con el abecedario en mayúsculas y comienza el ciclo.

Para poder ejecutar el programa podemos utilizar CodeBlocks, compilando y ejecutando de manera estándar.

En seguida se muestran algunos de los resultados obtenidos, solamente se mostrara el resultado final, si se requiere ver la animación entonces se debe de ejecutar el programa.

Figure 2: Resultados



Problema 2

Dado un archivo de entrada, escribir un programa que encuentre los siguiente:

a) Probabilidad (Pbb) de aparición de cada una de las letras del alfabeto (no haga diferencia entre minúsculas y mayúsculas).

b) $Pbb(x|y)$, para $(x, y) \in (a...z, A...Z)$ de las 10 letras (x) mas frecuentes.

El archivo debe ser recorrido solo una vez (se evaluará la eficiencia de su código).

Respuesta

El programa lee el archivo una vez usando un while que funciona con un scanf(), este se detiene hasta que hay un EOF, es importante mencionar, que es lo que se tomo en consideración para poder hacer la clasificación de las letras. Se considero la posibilidad de los caracteres especiales del español, entonces el programa considera las vocales acentuadas, la ñ, y diéresis en la vocal u, todo estos caracteres especiales son considerados en un switch(), en el cual al inicio aparecen los caracteres que creo que son los mas probables que aparezcan, por ejemplo las vocales i, o, e acentuadas. Las letras mayúsculas y minúsculas fueron consideradas como una misma. Es de suma importancia tener en cuenta que **se tomo en cuenta los caracteres como el espacio, y otros imprimibles de ASCII** como caracteres permitidos, todos estos se describen en nuestro programa como], se tomo en consideración estos espacios y caracteres debido a que la mayoría de los procesadores de texto toman en cuenta los espacios y signos como exclamación como caracteres.

Cada uno de los caracteres es almacenado en un arreglo dinámico, de una dimensión (simula ser de 2 dimensiones), la matriz generada es de 27 * 26, la ultima fila contiene la suma de cada aparición de alguna letra del abecedario.

Después de que se almacena la información, se pasa a calcular las probabilidades de cada una de las letras del abecedario, después se pasa a calcular las 10 probabilidades mayores de las 10 variables que mas aparecen.

Para correr el programa es necesario hacerlo desde la terminal, ejecutando un makefile. Se debe usar el comando **make**, una vez hecho esto se creara un ejecutable llamado **problema2**, entonces para poder hacer la lectura del archivo debemos ingresar

./problema2 < 996.txt. 996.txt es el nombre del archivo con el que se pidió probar el programa.

Figure 3: Como ejecutar

```
./problema2 < 996.txt
```

En seguida se muestran los resultados obtenidos al leer el libro que se nos entregó.

Figure 4: Resultados

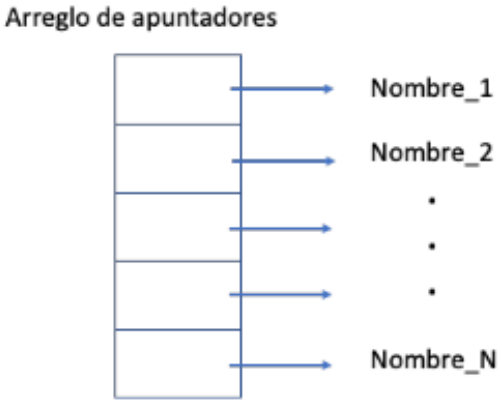
```
============================================================
                Las probabilidades de cada letra:
============================================================
P(A) = 0.063453,  P(B) = 0.010869,  P(C) = 0.017560,

P(D) = 0.034751,  P(E) = 0.093489,  P(F) = 0.017779,

P(G) = 0.014587,  P(H) = 0.053383,  P(I) = 0.051918,

P(J) = 0.000704,  P(K) = 0.005331,  P(L) = 0.028441,

P(M) = 0.019550,  P(N) = 0.053159,  P(O) = 0.062572,

P(P) = 0.011201,  P(Q) = 0.001794,  P(R) = 0.042643,

P(S) = 0.048873,  P(T) = 0.072169,  P(U) = 0.021359,

P(V) = 0.007967,  P(W) = 0.017297,  P(X) = 0.001943,

P(Y) = 0.014157,  P(Z) = 0.000471,  P([) = 0.232579,
```

```
============================================================
            Probabilidades condicionales
============================================================

P(E|H) = 0.413106, P(E|R) = 0.233507, P(E|S) = 0.129341, P(E|T) = 0.085206, P(E|V) = 0.737382,
P(E|M) = 0.244842, P(E|L) = 0.163329, P(E|B) = 0.385556, P(E|N) = 0.074164, P(E|D) = 0.100641,
P(E|[) = 0.013265, P(E|E) = 0.032457,

P(T|[) = 0.132246, P(T|A) = 0.131426, P(T|I) = 0.129590, P(T|S) = 0.103370, P(T|N) = 0.085758,
P(T|O) = 0.066581, P(T|U) = 0.140188, P(T|E) = 0.023122, P(T|R) = 0.046186, P(T|H) = 0.028987,
P(T|T) = 0.014873, P(T|C) = 0.053483,

P(A|[) = 0.091982, P(A|H) = 0.178400, P(A|E) = 0.052954, P(A|S) = 0.078741, P(A|M) = 0.163816,
P(A|W) = 0.151100, P(A|R) = 0.060299, P(A|L) = 0.088434, P(A|C) = 0.132677, P(A|T) = 0.025272,
P(A|P) = 0.127087, P(A|F) = 0.068256,

P(O|[) = 0.049972, P(O|T) = 0.108619, P(O|H) = 0.106183, P(O|N) = 0.084557, P(O|F) = 0.200958,
P(O|R) = 0.077070, P(O|C) = 0.183759, P(O|D) = 0.080871, P(O|S) = 0.054435, P(O|L) = 0.073518,
P(O|W) = 0.116354, P(O|M) = 0.101745,

P(H|T) = 0.356008, P(H|[) = 0.059473, P(H|C) = 0.249418, P(H|A) = 0.000409, P(H|W) = 0.234923,
P(H|S) = 0.054243, P(H|G) = 0.152514, P(H|A) = 0.000409, P(H|P) = 0.016390, P(H|E) = 0.001449,
P(H|R) = 0.001948, P(H|N) = 0.000785,

P(N|A) = 0.245219, P(N|I) = 0.241066, P(N|O) = 0.128458, P(N|E) = 0.077666, P(N|[) = 0.017788,
P(N|U) = 0.102618, P(N|K) = 0.188159, P(N|R) = 0.022723, P(N|W) = 0.034992, P(N|N) = 0.007852,
P(N|G) = 0.013841, P(N|A) = 0.245219,

P(I|[) = 0.054221, P(I|H) = 0.149755, P(I|W) = 0.198232, P(I|T) = 0.044908, P(I|A) = 0.047519,
P(I|R) = 0.066892, P(I|L) = 0.095847, P(I|S) = 0.039915, P(I|D) = 0.053918, P(I|U) = 0.075618,
P(I|M) = 0.075710, P(I|N) = 0.027788,

P(S|[) = 0.059867, P(S|I) = 0.141215, P(S|A) = 0.101831, P(S|E) = 0.064385, P(S|U) = 0.108322,
P(S|R) = 0.053148, P(S|S) = 0.042469, P(S|N) = 0.032787, P(S|O) = 0.025847, P(S|T) = 0.016502,
P(S|M) = 0.040785, P(S|D) = 0.022614,

P(R|E) = 0.124474, P(R|O) = 0.132597, P(R|A) = 0.080384, P(R|[) = 0.015341, P(R|U) = 0.160747,
P(R|T) = 0.024806, P(R|I) = 0.033735, P(R|P) = 0.128988, P(R|F) = 0.060901, P(R|G) = 0.066256,
P(R|R) = 0.019677, P(R|B) = 0.068772,

P(D|N) = 0.203503, P(D|E) = 0.072217, P(D|[) = 0.026588, P(D|A) = 0.051486, P(D|I) = 0.046033,
P(D|L) = 0.072290, P(D|R) = 0.034160, P(D|O) = 0.018264, P(D|U) = 0.015455, P(D|D) = 0.007232,
P(D|G) = 0.002657, P(D|W) = 0.001231,
```

Problema 3

Dado una lista de nombres (strings) de N personas (apellido_paterno, apellido_materno, Nombre(s)), escribir un programa que ordene los nombres alfabéticamente usando un arreglo de apuntadores:

Figure 5: Ejemplo



Los nombres pueden tener distintas longitudes; cuando un nombre sea prefijo de otro, considerar al nombre mas corto como menor. El ordenamiento debe ser a través de una función que reciba el arreglo de apuntadores.

Respuesta

El programa lee el archivo y almacena cada nombre en un arreglo de dos dimensiones de manera dinámica, una vez hecho esto se pasa a ordenar el arreglo, se utiliza el método de merge sort para hacer esto. La manera en que se considera el orden es a través de los nombres de las personas(no apellidos), es importante mencionar que se espera que los nombres comienzan con letra mayúscula, si inician con minúsculas, y se hace una comparación entre nombre con mayúsculas y minúsculas, siempre se tomara como mayor aquel que tenga letra minúscula, esto debido a que se utiliza el código ASCII como referencia.

El programa espera que el archivo que se entrega contenga el formato especificado en la descripción

del problema. Para ejecutar el programa se necesita hacerlo desde terminal. Primero ejecutaremos el comando **make** seguido de esto se creara un ejecutable llamado **programa3** el cual lo ejecutaremos de la siguiente manera figura (6), haciendo direccionamiento del archivo que queremos leer.

Figure 6: Ejecuta

```
./problema3 < nombres.txt
```

En seguida se muestran los resultados obtenidos al leer un archivo.

Figure 7: Resultado

```
La lista original es:
=======================================
Encinia,Gonzalez,Carlos,Giovanny
Zapata,Meddina,Andrea
Mendez,Perez,Gabriel,Andres
Lopez,Lopez,Rafael,Carlos
Lozano,Hernandez,Carl,Javier
Lopez,Lopez,Andes,Javier
Romero,Flores,Miriam



La lista ordenada por nombre es
=======================================

Lopez,Lopez,Andes,Javier
Zapata,Meddina,Andrea
Lozano,Hernandez,Carl,Javier
Encinia,Gonzalez,Carlos,Giovanny
Mendez,Perez,Gabriel,Andres
Romero,Flores,Miriam
Lopez,Lopez,Rafael,Carlos
```

Problema 4

Programa que encuentre los tres números mayores de un arreglo de enteros, especificando su posición (índice) original en el arreglo de entrada.

El prototipo de la función debe ser:

void find_three_largest(const int *a, int len_a, int *largest, int *index_largest int *second_largest, int *second_index_largest, int *third_largest, int *third_index_largest);

Respuesta

El algoritmo primero crea un arreglo con 10 elementos, esto puede cambiar si se desea, después se llena el arreglo con elementos aleatorios entre 0 y 3000, esto también puede ser modificado si se requiere.

Después de que se crea el vector se manda a la función descrita en el problema y se empieza a utilizar un algoritmo parecido a selection sort el cual solamente busca los tres primero valores mayores y sus respectivos índices. La verificación de estos valores se hace a través de una función la cual compara un arreglo temporal el cual guarda los resultados obtenidos, esto para evitar repeticiones. El algoritmo al igual que los últimos dos se debe de ejecutar desde consola, primero ejecutamos el archivo **make** después este generara un ejecutable llamado **problema4** ya solamente debemos de ejecutarlo de la siguiente manera **./problema4**. En seguida se muestran los resultados obtenidos de diferentes ejecuciones.

Figure 8: Resultados

```
============================================================
El arreglo es:

2046 1878 4806 3074 3534 1286 1359 3394 3821 3732 2522 605
============================================================
Estos son los numeros mas altos y sus correspondientes indices
indice 2, numero 4806
indice 8, numero 3821
indice 9, numero 3732
============================================================
El arreglo es:

4432 1189 3016 1519 3413 647 1130 450 3537 182 3241 1044
============================================================
Estos son los numeros mas altos y sus correspondientes indices
indice 0, numero 4432
indice 8, numero 3537
indice 4, numero 3413
============================================================
El arreglo es:

969 2453 2304 2701 2964 3829 1932 3076 918 1804 4399 3576
============================================================
Estos son los numeros mas altos y sus correspondientes indices
indice 10, numero 4399
indice 5, numero 3829
indice 11, numero 3576
============================================================
El arreglo es:

183 740 4213 1600 1199 191 2396 3190 2924 2993 4905 1228
============================================================
Estos son los numeros mas altos y sus correspondientes indices
indice 10, numero 4905
indice 2, numero 4213
indice 7, numero 3190
```