* **ANALISIS**

**Generación de Datos Aleatorios:**

El programa genera datos aleatorios para simular estudiantes que desean ingresar a las carreras universitarias.

Estos datos incluyen el nombre, apellido, carrera elegida y un puntaje aleatorio.

**Proceso de Admisión:**

El programa calcula la admisión de los estudiantes en función de sus puntajes y las restricciones de cada carrera.

Los estudiantes son admitidos si cumplen con ciertos criterios de puntaje para cada carrera.

Además, se asignan becas a los estudiantes que obtienen un puntaje igual o superior al 95%.

**Consideraciones Especiales:**

Se aplican descuentos del 25% en la matrícula para estudiantes cuyos padres o madres son profesores de la universidad.

Se otorgan puntos adicionales a los estudiantes en la carrera de Medicina si son abanderados, tienen afinidad con la carrera o poseen capacidades especiales.

**Escritura de Resultados:**

Los resultados de admisión se escriben en archivos CSV separados para cada carrera.

Cada archivo contiene información sobre los estudiantes admitidos, incluyendo su nombre, carrera, puntaje, estado de admisión, puntaje adicional, puntaje total y solicitud de beca.

**Impresión de Resultados:**

Los resultados de admisión también se imprimen en la consola en un formato tabular para cada carrera.

Uso de Funciones y Métodos:

El código utiliza funciones y métodos para modularizar el proceso de generación de datos, cálculo de admisiones y escritura de resultados.

Cada método tiene una función específica y contribuye a la ejecución del programa de manera organizada y estructurada.

Aleatoriedad:

Se utiliza la clase Random para generar números aleatorios en varios puntos del programa, como la generación de puntajes, la selección de carreras y la aplicación de criterios especiales.

Manipulación de Arreglos:

Se utilizan matrices para almacenar los datos de entrada y salida de los estudiantes, así como para organizar los resultados de admisión por carrera.

En resumen, el código simula un proceso de admisión universitaria completo, que incluye la generación de datos aleatorios, el cálculo de admisiones, la consideración de criterios especiales, la escritura e impresión de resultados. Está bien estructurado y utiliza funciones y métodos para mejorar la legibilidad y la mantenibilidad del código.

* ALGORITMO COMPUTACIONAL

1. Declarar y definir un arreglo de cadenas llamado NOMBRES con nombres de estudiantes

2. Declarar y definir un arreglo de cadenas llamado APELLIDOS con apellidos de estudiantes

3. Iniciar el método principal main

4. Declarar e inicializar matrices para almacenar datos de entrada y salida para cada carrera (Química, Fisioterapia y Rehabilitación, Medicina)

5. Generar los datos de entrada para los estudiantes, incluyendo nombre, apellido, carrera y puntaje

6. Calcular las admisiones para cada carrera basadas en los puntajes de los estudiantes

7. Escribir los resultados de admisión en archivos CSV para cada carrera

8. Imprimir los resultados de admisión en la consola para cada carrera

9. Fin del método principal main

10. Definir el método generarDatosEntrada que recibe una matriz de datosEntrada como parámetro

11. Inicializar un generador de números aleatorios

12. Para cada índice en la matriz de datosEntrada

13. Generar un nombre aleatorio

14. Generar un apellido aleatorio

15. Generar una carrera aleatoria

16. Generar un puntaje aleatorio entre 0 y 100

17. Almacenar los datos generados en la matriz de datosEntrada

18. Fin del método generarDatosEntrada

19. Definir el método calcularAdmisiones que recibe matrices de datosEntrada y datosSalida para cada carrera como parámetros

20. Inicializar contadores para cada carrera

21. Para cada estudiante en los datos de entrada

22. Obtener el nombre, la carrera y el puntaje del estudiante

23. Inicializar variables para admitido, beca y costo de la carrera

24. Según la carrera del estudiante

25. Caso "Química":

26. Si el puntaje es mayor o igual a 80 y el contador de Química es menor que 80

27. Almacenar los datos del estudiante como admitido en la matriz de salida de Química

28. Si el puntaje es mayor o igual a 95, asignar beca al estudiante

29. Incrementar el contador de Química

30. Caso "Fisioterapia y Rehabilitación":

31. Si el puntaje es mayor o igual a 90

32. Marcar al estudiante como admitido

33. Caso "Medicina":

34. Si el puntaje es mayor o igual a 85 y el contador de Medicina es menor que 80, o si el puntaje es mayor o igual a 95

35. Marcar al estudiante como admitido

36. Si el puntaje es mayor o igual a 95, asignar beca al estudiante

37. Incrementar el contador de Medicina

38. Verificar si el padre o madre del estudiante es profesor de la UTPL y aplicar un descuento del 25% al costo de la carrera si corresponde

39. Si el estudiante está admitido

40. Calcular el puntaje adicional para el estudiante

41. Calcular el puntaje total del estudiante

42. Almacenar los datos del estudiante en la matriz de salida correspondiente a su carrera

43. Fin del método calcularAdmisiones

44. Definir el método calcularPuntajeAdicional que recibe el nombre, la carrera y el puntaje del estudiante como parámetros

45. Inicializar un generador de números aleatorios

46. Inicializar el puntaje adicional como 0

47. Si el puntaje es menor que 95

48. Si la carrera es Medicina y el estudiante es abanderado, agregar 5 puntos adicionales al puntaje

49. Si la carrera es Medicina y el estudiante tiene afinidad, agregar 2 puntos adicionales al puntaje

50. Si la carrera es Medicina y el estudiante tiene capacidad especial, agregar 1 punto adicional al puntaje

51. Retornar el puntaje adicional

52. Fin del método calcularPuntajeAdicional

53. Definir el método esAbanderado que retorna verdadero o falso

54. Inicializar un generador de números aleatorios

55. Retornar verdadero si un número aleatorio es menor que 20, de lo contrario, retornar falso

56. Fin del método esAbanderado

57. Definir el método esAfines que retorna verdadero o falso

58. Inicializar un generador de números aleatorios

59. Retornar verdadero si un número aleatorio es menor que 10, de lo contrario, retornar falso

60. Fin del método esAfines

61. Definir el método esCapacidadEspecial que retorna verdadero o falso

62. Inicializar un generador de números aleatorios

63. Retornar verdadero si un número aleatorio es menor que 35, de lo contrario, retornar falso

64. Fin del método esCapacidadEspecial

65. Definir el método esPadreMadreProfesor que retorna verdadero o falso

66. Inicializar un generador de números aleatorios

67. Retornar verdadero si un número aleatorio es menor que 5, de lo contrario, retornar falso

68. Fin del método esPadreMadreProfesor

69. Definir el método obtenerNombreAleatorio que retorna un nombre aleatorio de la lista NOMBRES

70. Inicializar un generador de números aleatorios

71. Retornar un nombre aleatorio de la lista NOMBRES

72. Fin del método obtenerNombreAleatorio

73. Definir el método obtenerApellidoAleatorio que retorna un apellido aleatorio de la lista APELLIDOS

74. Inicializar un generador de números aleatorios

75. Retornar un apellido aleatorio de la lista APELLIDOS

76. Fin del método obtenerApellidoAleatorio

77. Definir el método obtenerCarreraAleatoria que retorna una carrera aleatoria entre Química, Fisioterapia y Rehabilitación, y Medicina

78. Inicializar un generador de números aleatorios

79. Retornar una carrera aleatoria entre las disponibles

80. Fin del método obtenerCarreraAleatoria

81. Definir el método escribirResultados que escribe los resultados de admisión en archivos CSV

82. Para cada fila de datosSalida

83. Si la fila no es nula y tiene un estado de admisión

84. Escribir la fila en el archivo CSV con el formato especificado

85. Fin del método escribirResultados

86. Definir el método imprimirResultados que imprime los resultados de admisión en la consola

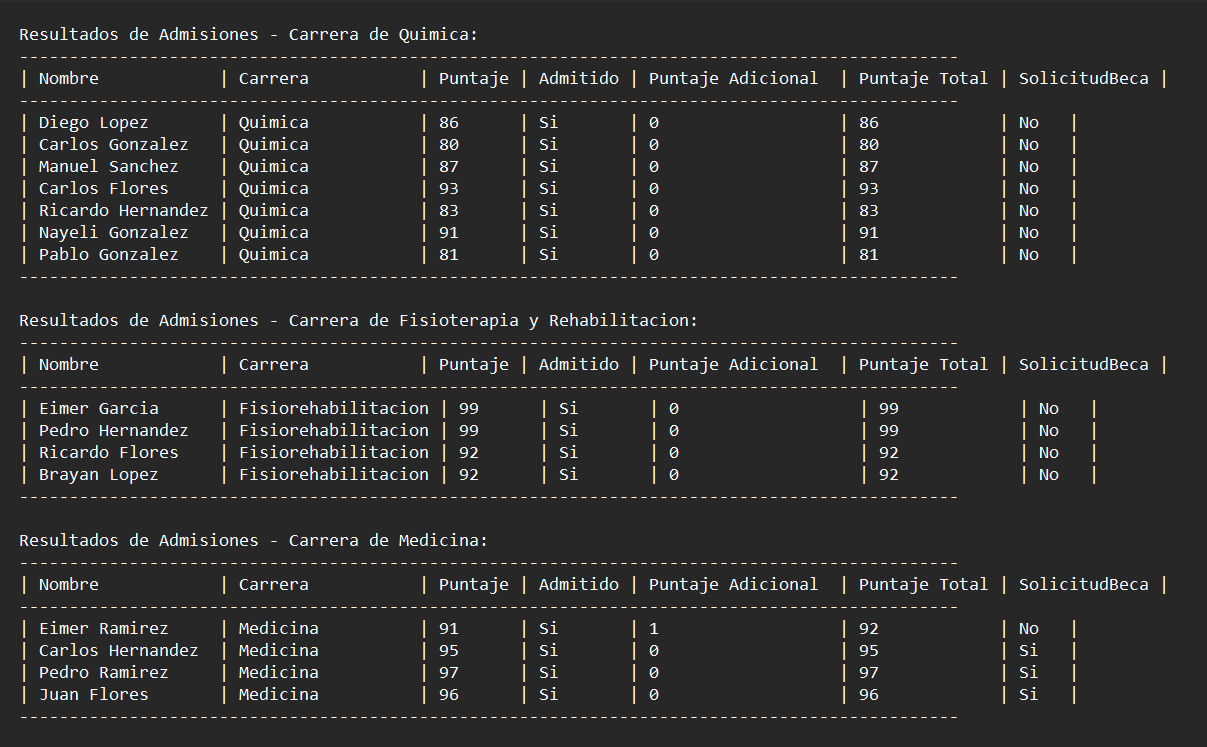
87. Para cada fila de datosSalida

88. Si la fila no es nula y tiene un estado de admisión

89. Imprimir la fila en la consola con el formato especificado

90. Fin del método imprimirResultados

**PRUEBA DE ESCRITORIO**

****