




MANUAL TÉCNICO

BANKSIM V.1

Instituto Tecnológico de Ciudad Madero
Ingeniería en Sistemas Computacionales

EQUIPO 1

- Constantino Morales Angel Jesus
 - Del Angel Flores Diego Emmanuel
 - Gonzales Garces Christian Michelle
- 



INDICE

ACERCA DE ESTE MANUAL	3
ANTES DE EMPEZAR.....	4
Simulación en lenguajes de alto nivel.....	5
GpSS WORLD.....	6
Requerimientos del sistema	7
codigo en gpss world	8
EJECUCIÓN Y RESULTADOS EN GPSS WORLD.....	11
Simulación EN PYTHON	14
codigo EN PYTHON	18
DIAGRAMA.....	22
EJECUCION Y RESULTADOS EN PYTHON.....	23
VALIDACIÓN	26
CONCLUSIÓN.....	29
REFERENCIAS	30
SOPORTE TECNICO	31

ACERCA DE ESTE MANUAL

En este manual se describirá las funciones utilizadas en GPSS WORLD y Para poder hacer la simulación deseada del siguiente enunciado:

Un banco emplea 3 cajeros para servir a sus clientes, los clientes arriban de acuerdo con un proceso poisson a una razón media de 40 por hora, si un cliente encuentra todos los cajeros ocupados entonces se incorpora a la cola que alimenta a todos los cajeros. El tiempo que dura la transacción entre un cajero y un cliente sigue una distribución uniforme entre 0 y 1 minutos. Para esta información. Calcular el tiempo promedio en el sistema y la cantidad promedio de clientes en el sistema.

En este caso tomaremos los resultados de un programa en Python como los del sistema real y los resultados de GPSS WORLD como del sistema simulado.

ANTES DE EMPEZAR

Instalación de GPSS WORLD y Visual Studio Code

Antes de empezar, debe tener un sistema operativo, compatible con Microsoft Windows instalado en su computadora y debe tener un conocimiento básico de ese sistema

Para poder proceder con crear la simulación del enunciado mencionado anteriormente, es importante realizar la instalación del Gpss World , y tener el conocimiento básico. Para esto es importante tener en cuenta el Manual de Usuario, donde se explican conceptos básicos y procesos de instalación.

SIMULACIÓN EN LENGUAJES DE ALTO NIVEL

La simulación de eventos discretos es una técnica utilizada para modelar y analizar sistemas dinámicos en los que los eventos ocurren en puntos específicos en el tiempo. Se utiliza para simular y comprender el comportamiento de sistemas complejos, como líneas de producción, redes de comunicación, sistemas de transporte y procesos de negocios.

Python es un lenguaje de programación de alto nivel que ofrece muchas bibliotecas y herramientas que facilitan la implementación de simulaciones de eventos discretos.

La simulación de eventos discretos en lenguajes de alto nivel, como Python, presenta varias ventajas que la convierten en una opción atractiva para modelar y analizar sistemas complejos. A continuación, se enumeran algunas de estas ventajas:

Beneficios Clave

1. **Facilidad de programación:** Python es conocido por su sintaxis clara y legible, lo que facilita la implementación de algoritmos de simulación. Su naturaleza de lenguaje de alto nivel permite expresar conceptos complejos de manera más sencilla y concisa, lo que ahorra tiempo y esfuerzo durante el desarrollo de la simulación.

2. **Amplia comunidad y bibliotecas especializadas:** Python cuenta con una comunidad activa de desarrolladores y una amplia gama de bibliotecas especializadas que facilitan la simulación de eventos discretos. La biblioteca `simpy`, utilizada en el ejemplo anterior, es una de las opciones más populares y ofrece una gran cantidad de funciones y herramientas para construir modelos de simulación.

3. **Flexibilidad y extensibilidad:** Python es un lenguaje flexible que permite adaptarse a diferentes requerimientos y escenarios de simulación. Puede combinarse con otras bibliotecas de Python, como `numpy` y `pandas`, para realizar análisis estadísticos y visualizaciones de los resultados de la simulación. Además, Python permite la integración con otras tecnologías y sistemas, lo que facilita la conexión con bases de datos, servicios web y otras herramientas externas.

4. **Eficiencia en el tiempo de desarrollo:** Gracias a la amplia gama de bibliotecas y herramientas disponibles, la simulación de eventos discretos en Python puede ser más rápida en términos de tiempo de desarrollo en comparación con otros lenguajes. Python proporciona estructuras de datos y funciones incorporadas que agilizan la implementación de modelos de simulación, lo que permite concentrarse en la lógica y el análisis del sistema en lugar de preocuparse por detalles de bajo nivel.

GPSS WORLD

GPSS (**General Purpose Simulation System**, en español: **Simulación de Sistemas de Propósito General**) es un lenguaje de simulación discreta de eventos / paquete que opera no principalmente dando un paso a través del tiempo, pero mediante la generación de transacciones (las personas que sirven, a puestos de trabajo, proceso, etc.) y luego progresa a través de varias etapas (o bloques de utilizar el término correcto).

Los elementos que se inyectan al modelo (transacciones) mediante el bloque GENERATE son puestos en la cadena de eventos futuros (Future Event Chain) con el instante de su futuro nacimiento, las tareas que estos elementos realizan mediante el bloque ADVANCE generan eventos futuros con el instante de terminación de la tarea.

El GPSS detiene el reloj y simula todo lo que tiene que simular para un determinado instante (eventos corrientes), cuando no tiene más nada por simular mira en la cadena de eventos futuros qué es lo próximo que tiene que hacer y coloca el reloj para ese instante con lo que los eventos que eran futuros se transforman en corrientes. El GPSS ejecuta todos los eventos corrientes hasta que no haya ningún evento y así se repite el proceso hasta que un TERMINATE con operando distinto de cero alcance la cantidad pedida en el START.

El usuario describe las acciones que los elementos que se mueven por el sistema realizan (lo hace mediante los bloques que representan a esas acciones) y el GPSS se encarga de realizar la simulación al momento que recibe el START con la cantidad de transacciones terminadas que el usuario quiere simular.

Beneficios Clave

El GPSS tiene varias características que lo hacen un lenguaje de simulación adecuado para introducirse en la programación de modelos de simulación con un lenguaje de propósito específico. Estas cualidades se pueden agrupar en tres que son:

1. GPSS es completo: Esto quiere decir que es un lenguaje muy rico ya que contiene incorporadas muchas opciones estadísticas y se puede aplicar en una gran variedad de situaciones prácticas.
2. Es de fácil empleo y tiene poder: GPSS es fácil de aprender y de usar ya que se requiere invertir una cantidad mínima de tiempo en la curva de aprendizaje para poder simular y modelar de una manera tan compleja como necesite.
3. Tiempo de Prueba: El GPSS es uno de los primeros lenguajes de simulación que se desarrolló, así que se ha probado durante mucho tiempo. (Gordon lo implementó en 1962).

REQUERIMIENTOS DEL SISTEMA

Los requerimientos del sistema en este manual técnico son una parte fundamental de la documentación que describe los recursos de hardware, software y configuraciones necesarios para instalar y ejecutar correctamente la aplicación del software. Estos requerimientos definen los mínimos y recomendaciones para el entorno en el cual se debe implementar el software, los requerimientos son los siguientes :

Requerimientos para GPSS WORLD

Recomendado
Procesador Pentium 200 MMX o mejor
64 megabytes en RAM
65 megabytes de Espacio libre en Disco Duro
Monitor SVGA (1024 x 768 X 16 millones de colores)
Conexión a internet
Ratón (Mouse) y teclado

Requerimientos para Visual Studio Code

Recomendado
1.8 GHz de velocidad de procesador
Sistema operativo Windows 8 o posterior
8 GB de RAM
Conexión a internet
Monitor SVGA (1024 x 768 X 16 millones de colores)
Ratón (Mouse) y Teclado

CODIGO EN GPSS WORLD

*Segmento de generación de clientes

GENERATE (POISSON(1,1.5)) ;Llegada de clientes

*Elección de cajero o encolamiento(q_cajero)

SELECT NU cajero,1,3,,F,q_cajero

TRANSFER, atención

q_cajero **SELECT** MIN cajero,1, 3,,Q; Elección de cola

atención **QUEUE** P\$cajero; Cola teléfono elegido

SEIZE P\$cajero; Servicio del cajero

DEPART P\$cajero;Abandono de cola

ADVANCE 1,0;Demora de la transacción

RELEASE P\$cajero;Fin de la transacción CAJA ;Libera el cajero

TERMINATE

*Segmento de parada

GENERATE 480; Cierre después de 8 horas

TERMINATE 1

START 1

1. Para ejecutar nuestro problema en GPSS WORLD el mismo sistema se ejecutará con los procesos de cada etapa del proyecto donde mostrará los datos conforme se está ejecutando al momento de iterar la cantidad de elementos que procesa la simulación del proyecto.

En nuestras líneas de código contienen lo siguiente:

1. Una cola
2. Generador de llegada de clientes
3. 3 cajeros
4. Tiempo de transacción
5. Reloj de la simulación

En el siguiente proceso obtenemos la generación de clientes mediante su tiempo entre llegadas por el método poisson.

```
1  *Segmento de generación de clientes
2  |  GENERATE (POISSON(1,1.5)) ;Llegada de clientes
```

En nuestra simulación tenemos tres cajeros a disposición de los clientes, donde elegirán el que se encuentre disponible.

```
4  *Elección de cajero o encolamiento(q_cajero)
5  |  SELECT NU cajero,1,3,,F,q_cajero
```

Con la instrucción TRANSFER verificaremos el trayecto de las transacciones, que ingresen a esta instrucción de bloque.

```
7  |  TRANSFER,atencion
```

Seleccionaremos el cajero disponible con la siguiente instrucción

```
9  q_cajero  SELECT MIN cajero,1,3,,Q; Eleccion de cola
10 atencion  QUEUE P$cajero;Cola telefono elegido
```

Registraremos cada transacción que entre de tal forma que el cajero quede ocupado hasta que ingrese a una instrucción RELEASE

```
11 |  SEIZE P$cajero; Servicio del cajero
```

Con DEPART simularemos que un elemento de la cola se desincorpora de la misma y entra a un cajero disponible.

```
12 |  DEPART P$cajero;Abandono de cola
```

ADVANCE nos dará el tiempo de servicio en el cajero atendido

13 | **ADVANCE 1,0;Demora de la transacción**

Con RELEASE se desocupará el cajero por la transacción

14 | **RELEASE P\$cajero;Fin de la transacción CAJA ;Libera el cajero**

Finalmente, el cliente sale del banco

15 | **TERMINATE**

17 *Segmento de parada

18 GENERATE 480;Cierre después de 8 horas

19 TERMINATE 1

20 START 1

EJECUCIÓN Y RESULTADOS EN GPSS WORLD

A continuación, se presenta una tabla con los resultados que se obtuvieron después de 10 ejecuciones del programa BANKSIM usando combinaciones de números aleatorios distintos:

Ejecución	Promedio de clientes en el sistema	Tiempo promedio en el sistema
1	42	1.0
2	40	1.0
3	40	0.995
4	37	1.0
5	39	0.995
6	41	1.0
7	38	1.0
8	41	0.995
9	42	1.0
10	39	0.98

EVIDENCIAS DE LOS RESULTADOS

1.

FACILITY	ENTRIES	UTIL.	AVE.	TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	218	0.454	1.000	1	339	0	0	0	0	0
2	92	0.192	1.000	1	0	0	0	0	0	0
3	28	0.058	1.000	1	0	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY		
1	2	0	218	213	0.013	0.028	1.200	0		
2	2	0	92	90	0.004	0.022	1.000	0		
3	1	0	28	27	0.002	0.036	1.000	0		

2.

FACILITY	ENTRIES	UTIL.	AVE.	TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	220	0.458	1.000	1	0	0	0	0	0	0
2	83	0.173	1.000	1	0	0	0	0	0	0
3	24	0.050	1.000	1	0	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY		
1	1	0	220	219	0.002	0.005	1.000	0		
2	1	0	83	83	0.000	0.000	0.000	0		
3	1	0	24	24	0.000	0.000	0.000	0		

3.

FACILITY	ENTRIES	UTIL.	AVE.	TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	220	0.456	0.995	1	327	0	0	0	0	0
2	87	0.181	1.000	1	0	0	0	0	0	0
3	19	0.040	1.000	1	0	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY		
1	1	0	220	218	0.004	0.009	1.000	0		
2	1	0	87	87	0.000	0.000	0.000	0		
3	1	0	19	19	0.000	0.000	0.000	0		

4.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	210	0.438	1.000	1	302	0	0	0	0
2	76	0.158	1.000	1	0	0	0	0	0
3	15	0.031	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	1	0	210	208	0.004	0.010	1.000	0	
2	1	0	76	76	0.000	0.000	0.000	0	
3	1	0	15	15	0.000	0.000	0.000	0	

5.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	214	0.444	0.995	1	318	0	0	0	0
2	83	0.173	1.000	1	0	0	0	0	0
3	20	0.042	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	1	0	214	209	0.010	0.023	1.000	0	
2	1	0	83	83	0.000	0.000	0.000	0	
3	1	0	20	20	0.000	0.000	0.000	0	

6.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	226	0.471	1.000	1	0	0	0	0	0
2	85	0.177	1.000	1	0	0	0	0	0
3	17	0.035	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	1	0	226	226	0.000	0.000	0.000	0	
2	1	0	85	85	0.000	0.000	0.000	0	
3	1	0	17	17	0.000	0.000	0.000	0	

7.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	205	0.425	0.995	1	305	0	0	0	0
2	86	0.179	1.000	1	0	0	0	0	0
3	13	0.027	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	1	0	205	203	0.004	0.010	1.000	0	
2	1	0	86	86	0.000	0.000	0.000	0	
3	1	0	13	13	0.000	0.000	0.000	0	

8.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	223	0.465	1.000	1	332	0	0	0	0
2	90	0.188	1.000	1	0	0	0	0	0
3	18	0.037	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	1	0	223	219	0.008	0.018	1.000	0	
2	1	0	90	89	0.002	0.011	1.000	0	
3	1	0	18	18	0.000	0.000	0.000	0	

9.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	217	0.452	1.000	1	0	0	0	0	0
2	93	0.194	1.000	1	0	0	0	0	0
3	28	0.058	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	1	0	217	214	0.006	0.014	1.000	0	
2	1	0	93	92	0.002	0.011	1.000	0	
3	1	0	28	28	0.000	0.000	0.000	0	

10.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
1	211	0.440	1.000	1	0	0	0	0	0
2	83	0.171	0.988	1	314	0	0	0	0
3	19	0.040	1.000	1	0	0	0	0	0
QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)	RETRY	
1	2	0	211	206	0.013	0.028	1.200	0	
2	1	0	83	82	0.002	0.012	1.000	0	
3	1	0	19	18	0.002	0.053	1.000	0	

SIMULACIÓN EN PYTHON

Python ofrece una variedad de bibliotecas y herramientas que facilitan la simulación de sistemas, lo que lo convierte en una opción popular para la simulación en diferentes dominios.


En Python, existen varias bibliotecas populares para la simulación, como NumPy, SciPy y Matplotlib, que proporcionan funcionalidades para manipulación de datos, cálculos científicos y visualización. Estas bibliotecas ofrecen una base sólida para desarrollar modelos y algoritmos de simulación.

Un entorno de desarrollo popular para este lenguaje es Visual Studio Code sin duda, es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Para poder comprender mejor el código que se presentará en la siguiente sección es importante conocer algunos de los comandos sintácticos que Python emplea para llevar a cabo programas de propósito general.

Def

En Python las funciones se escriben como def

A screenshot of a code editor window with a dark background. At the top left, there are three colored circles: red, yellow, and green. Below them, the code is written in a light blue font. The code consists of two lines: '1 def funcion():' and '2 print('Funcion')'. The line numbers '1' and '2' are on the left, and the code is indented to the right.

```
1 def funcion():
2     print('Funcion')
```

IF

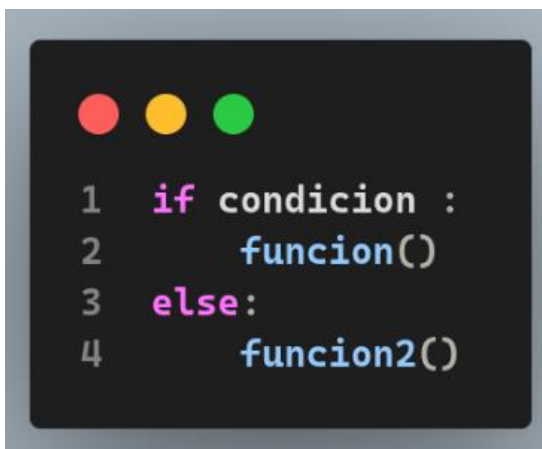
Se usa para decidir si una determinada declaración o bloque de enunciados se ejecutará o no; es decir, si una determinada condición es verdadera (true), se ejecutará un bloque de enunciado y, de ser falsa (false).

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains two lines of code:

```
1  if condicion :  
2      funcion()
```

IF ELSE

La declaración if solo nos dice que, si una condición es verdadera ejecutará un bloque de instrucciones y si la condición es falsa, no lo hará. Pero ¿y si queremos hacer otra cosa cuando la condición sea falsa? Aquí viene la declaración else. Podemos usar la instrucción else con la instrucción if para ejecutar un bloque de código cuando la condición es falsa.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains four lines of code:

```
1  if condicion :  
2      funcion()  
3  else:  
4      funcion2()
```

IF ELSE IF


Las sentencias if se ejecutan desde arriba hacia abajo. Tan pronto como una de las condiciones que controlan el if sea verdadera, se ejecuta la instrucción asociada con ese if, y el resto de la escalera se pasa por alto. Si ninguna de las condiciones es verdadera, se ejecutará la sentencia final else.



```
1  if condicion :
2      funcion()
3  elif condicion:
4      funcion2()
5  else:
6      funcion3()
```

IF ANIDADO

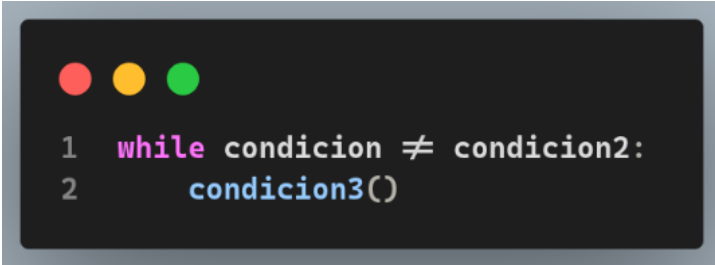
Es una declaración if que se deriva de otro if o else. Las declaraciones if anidadas significan una instrucción if dentro de una declaración if, es decir, podemos colocar una instrucción if dentro de otra instrucción if.



```
1  if condicion :
2      funcion()
3      if condicion2:
4          funcion2()
```


WHILE

Es un ciclo que mientras la condición se cumpla hará las instrucciones que se encuentra dentro.



```
1 while condicion ≠ condicion2:  
2     condicion3()
```

CODIGO EN PYTHON

```
import heapq
import math

# Configuración de los parámetros del generador congruencial multiplicativo
semilla = 111
constante_multiplicativa = 53
modulo = 32768

# Función para generar números aleatorios utilizando el método congruencial
multiplicativo
def generar_aleatorio():
    global semilla
    semilla = (constante_multiplicativa * semilla) % modulo
    return semilla / modulo

# Configuración de los parámetros del generador de llegadas
tasa_llegada = 60 / 40 # convertir a llegadas por minuto

# Función para generar tiempo entre llegadas
def generar_llegada():
    llegada = -1 / tasa_llegada * math.log(generar_aleatorio())
    return llegada

# Configuración de los parámetros del generador de tiempos de transacciones
a = 0 # Tiempo mínimo entre transacciones
b = 1 # Tiempo máximo entre transacciones
# Función para generar tiempo de transacciones entre el cajero y el cliente.
def generar_transaccion():
    global a
    global b
    tiempo_transaccion = a + (b - a) * generar_aleatorio()
    return tiempo_transaccion
```

Generador de
variables aleatorias

Generador de
tiempos entre
arribos

Generador de
tiempos de
transacciones

```
# Configuración de los parámetros de la simulación
num_atms = 3
cola = []
reloj = 0
siguiente_tiempo_llegada = generar_llegada()
siguiente_tiempo_salida = [float('inf')] * num_atms
atms_ocupados = 0
```

Elementos de
la simulación

```
# Inicializar variables para el cálculo de estadísticas
tiempo_total_en_sistema = 0
num_total_clientes = 0
```

Variables de
interés

```
# Mensajes de la simulación
bienvenida = "¡Hola! 😊 La simulacion ha comenzado"
print(bienvenida)
```

```
# Ejecutar la simulación durante 8 horas (480 minutos)
tiempo_simulacion = 480
while reloj < tiempo_simulacion:
    # Avanzar el reloj al siguiente tiempo de evento
    reloj = min(siguiente_tiempo_llegada + 1, min(siguiente_tiempo_salida))
```

Aquí comienza la
simulación

```
if siguiente_tiempo_llegada < min(siguiente_tiempo_salida):
    # Llega un cliente
    print("----> Llegó un cliente")
    if atms_ocupados < num_atms:
        # Asignar un cajero automático disponible al cliente
        atms_ocupados += 1
        cajero = siguiente_tiempo_salida.index(float('inf'))
        tiempo_servicio = generar_transaccion()
        siguiente_tiempo_salida[cajero] = reloj + tiempo_servicio
        print("*** Un cliente utilizó un cajero")
    else:
        # Todos los cajeros automáticos están ocupados, el cliente se une a la cola
```

```

heapq.heappush(cola, reloj)
print("## Un cliente se unió a la cola de espera")
# Programar la próxima llegada
siguiente_tiempo_llegada = reloj + generar_llegada()
else:
# Un cliente termina su transacción
num_total_clientes += 1
tiempo_total_en_sistema += min(siguiete_tiempo_salida) -
siguiente_tiempo_llegada
cajero = siguiente_tiempo_salida.index(min(siguiete_tiempo_salida))
siguiente_tiempo_salida[cajero] = float('inf')
if len(cola) > 0:
# Asignar el próximo cliente en la cola al cajero automático
siguiente_tiempo_cliente = heapq.heappop(cola)
tiempo_servicio = generar_aleatorio()
siguiente_tiempo_salida[cajero] = max(reloj, siguiente_tiempo_cliente) +
tiempo_servicio
print("*** Un cliente utilizó un cajero")
else:
# No hay clientes en la cola, el cajero automático queda libre
atms_ocupados -= 1
print("<- Un cliente dejó el cajero libre")

# Mensaje de fin de la simulación
print("*** La simulación ha finalizado. ***")

# Calcular las estadísticas
tiempo_promedio_en_sistema = abs(tiempo_total_en_sistema / num_total_clientes)
num_promedio_clientes = num_total_clientes / (reloj / 60)

# Imprimir los resultados
print() # Salto de línea
print("Resultados relevantes:")
print(f"Tiempo promedio que un cliente pasa en el sistema:
{tiempo_promedio_en_sistema:.2f} minutos")

```

Cálculo de
estadísticas

Visualización de
resultados

```
print(f"Cantidad promedio de clientes en el sistema:
{num_promedio_clientes:.2f}")

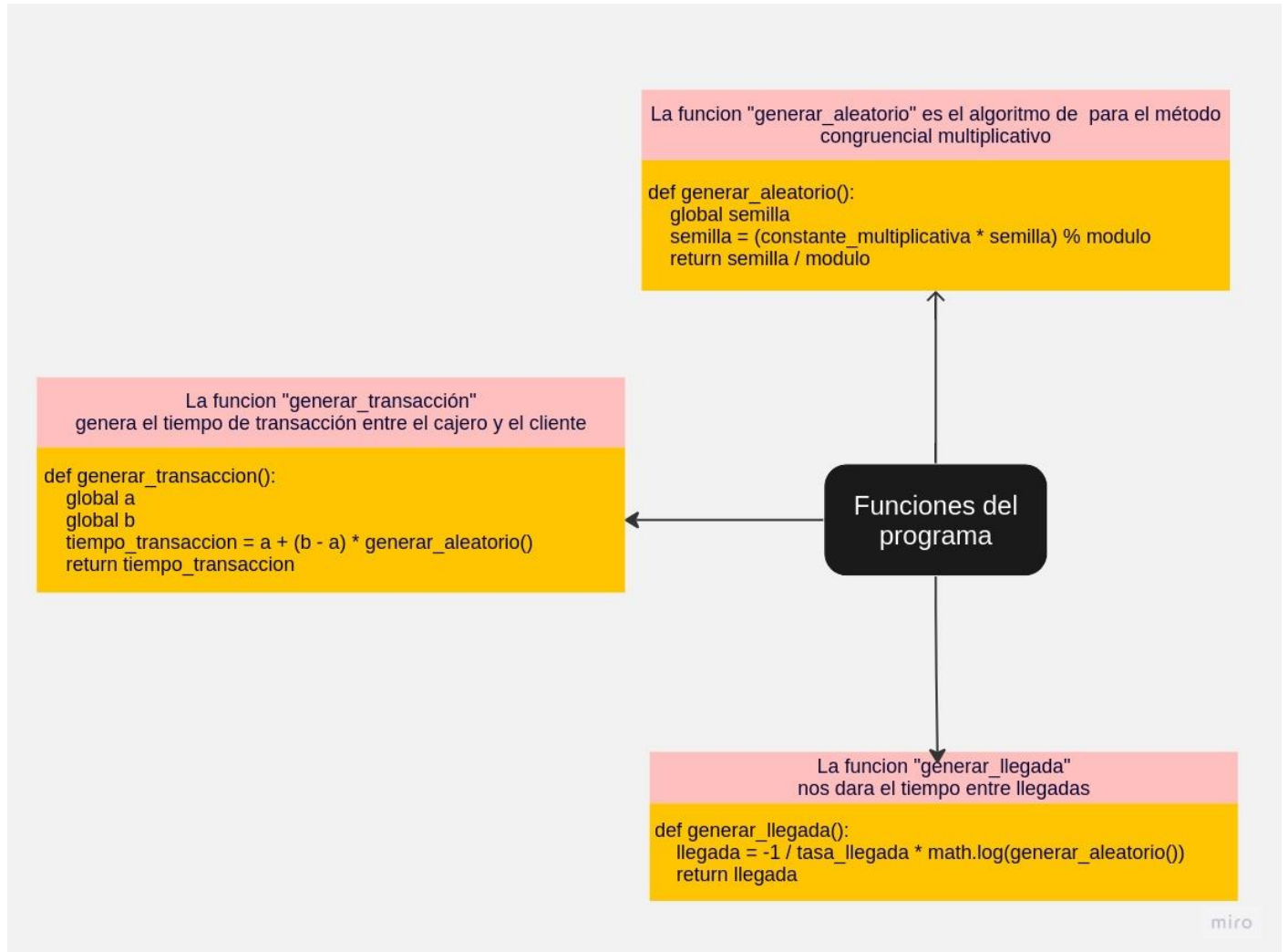
print() # Salto de línea

print("Otros datos:")

print(f"Tiempo total en el sistema de todos los clientes:
{abs(tiempo_total_en_sistema):.2f} minutos")

print(f"Cantidad total de clientes que ingresaron en 8 horas:
{num_total_clientes}")
```

DIAGRAMA



EJECUCION Y RESULTADOS EN PYTHON.

A continuación, se presenta una tabla con los resultados que se obtuvieron después de 10 ejecuciones del programa BANKSIM en Python usando combinaciones de números aleatorios distintos:

Ejecución	Promedio de clientes en el sistema	Tiempo promedio en el sistema
1	41	0.69
2	42	0.66
3	43	0.66
4	46	0.64
5	46	0.65
6	44	0.73
7	45	0.67
8	44	0.65
9	42	0.66
10	44	0.68

Las combinaciones de números aleatorios utilizados en las ejecuciones anteriores son los siguientes respectivamente:

Ciclo	XN	A
1	111	53
2	127	69
3	95	51
4	149	165
5	69	67
6	155	163
7	37	45
8	73	21
9	195	99
10	97	109

EVIDENCIAS DE LOS RESULTADOS EN VISUAL STUDIO

1.

```
Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.69 minutos
Cantidad promedio de clientes en el sistema: 41.62

Otros datos:
Tiempo total en el sistema de todos los clientes: 231.34 minutos
Cantidad total de clientes que ingresaron en 8 horas: 333
PS C:\Users\Angel>
```
2.

```
Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.66 minutos
Cantidad promedio de clientes en el sistema: 42.17

Otros datos:
Tiempo total en el sistema de todos los clientes: 222.15 minutos
Cantidad total de clientes que ingresaron en 8 horas: 338
PS C:\Users\Angel>
```
3.

```
Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.66 minutos
Cantidad promedio de clientes en el sistema: 43.48

Otros datos:
Tiempo total en el sistema de todos los clientes: 229.18 minutos
Cantidad total de clientes que ingresaron en 8 horas: 348
PS C:\Users\Angel>
```
4.

```
Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.64 minutos
Cantidad promedio de clientes en el sistema: 45.92

Otros datos:
Tiempo total en el sistema de todos los clientes: 235.61 minutos
Cantidad total de clientes que ingresaron en 8 horas: 369
PS C:\Users\Angel>
```


- Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.65 minutos
Cantidad promedio de clientes en el sistema: 46.23
- Otros datos:
Tiempo total en el sistema de todos los clientes: 240.15 minutos
Cantidad total de clientes que ingresaron en 8 horas: 370
5. PS C:\Users\Angel> █
- Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.73 minutos
Cantidad promedio de clientes en el sistema: 44.61
- Otros datos:
Tiempo total en el sistema de todos los clientes: 260.43 minutos
Cantidad total de clientes que ingresaron en 8 horas: 357
6. PS C:\Users\Angel> █
- Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.67 minutos
Cantidad promedio de clientes en el sistema: 44.81
- Otros datos:
Tiempo total en el sistema de todos los clientes: 242.72 minutos
Cantidad total de clientes que ingresaron en 8 horas: 360
7. PS C:\Users\Angel> █
- Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.65 minutos
Cantidad promedio de clientes en el sistema: 44.22
- Otros datos:
Tiempo total en el sistema de todos los clientes: 230.04 minutos
Cantidad total de clientes que ingresaron en 8 horas: 354
- 8.
- Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.66 minutos
Cantidad promedio de clientes en el sistema: 42.33
- Otros datos:
Tiempo total en el sistema de todos los clientes: 223.75 minutos
Cantidad total de clientes que ingresaron en 8 horas: 339
- 9.
- Resultados relevantes:
Tiempo promedio que un cliente pasa en el sistema: 0.68 minutos
Cantidad promedio de clientes en el sistema: 43.98
- Otros datos:
Tiempo total en el sistema de todos los clientes: 239.14 minutos
Cantidad total de clientes que ingresaron en 8 horas: 352
- 10.

VALIDACIÓN

La validación que se ha realizado a continuación involucra el cálculo de la media, varianza y pruebas estadísticas como la prueba F de Fisher y la prueba t de Student es un enfoque válido para comparar los resultados obtenidos en ambos programas, GPSS y Python. Estos análisis nos permiten evaluar si existe una diferencia significativa entre las muestras recopiladas

1.- Se recopilaron 10 resultados usando combinaciones distintas de números aleatorios, en ambos programas, GPSS y Python, con esto obtuvimos lo siguiente:

Cantidad promedio de clientes en el sistema.				8
			clientes/hora	
	total clientes	Python	GPSS WORLD	total clientes
1	333	41.625	42.250	338
2	338	42.25	40.875	327
3	348	43.5	40.750	326
4	369	46.125	37.625	301
5	370	46.25	39.625	317
6	357	44.625	41.000	328
7	360	45	38.000	304
8	354	44.25	41.375	331
9	339	42.375	42.250	338
10	352	44	39.125	313

2.- De acuerdo con los datos anteriores se calculó la media de ambos resultados, obteniendo lo siguiente.

Media Python	Media GPSS
44	40.2875

3.- Seguido de eso calculamos la varianza de ambos resultados, obteniendo los siguientes resultados.

VARIANZA PYTHON	
5.640625	22.625
3.0625	2.51388889
0.25	
4.515625	
5.0625	
0.390625	
1	
0.0625	
2.640625	
5.04871E-29	

VARIANZA GPSS WORLD	
3.85140625	24.0640625
0.34515625	2.673784722
0.21390625	
7.08890625	
0.43890625	
0.50765625	
5.23265625	
1.18265625	
3.85140625	
1.35140625	

4.- Una vez que tenemos los resultados de ambas varianzas, uno de los resultados en Python y GPSS, Procedemos a calcular la F de Fisher, esta es una prueba estadística utilizada para comparar la variabilidad entre dos o más muestras. Esta prueba se basa en la relación de las varianzas de las muestras y permite evaluar si existen diferencias significativas en las varianzas.

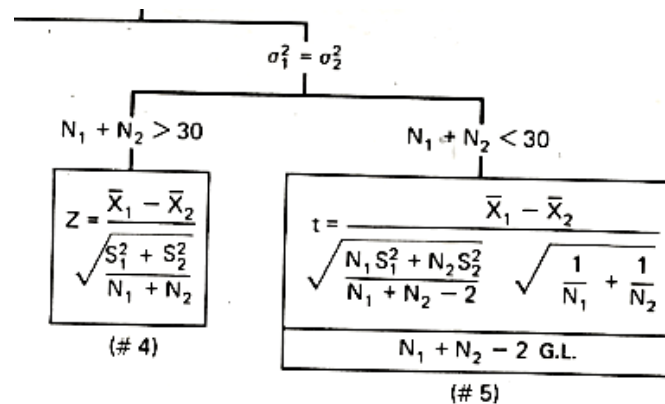
F FISHER
VARIANZA MAYOR SOBRE VARIANZA MENOR
1.063604972

5.- Este valor obtenido se compara en la tabla de valores críticos de la distribución Fisher (0.05)

g.d.l	Grados de libertad del Numerador								
	1	2	3	4	5	6	7	8	9
1	161,4	199,5	215,7	224,6	230,2	234,0	236,8	238,9	240,5
2	18,513	19,000	19,164	19,247	19,296	19,330	19,353	19,371	19,385
3	10,128	9,552	9,277	9,117	9,013	8,941	8,887	8,845	8,812
4	7,709	6,944	6,591	6,388	6,256	6,163	6,094	6,041	5,999
5	6,608	5,786	5,409	5,192	5,050	4,950	4,876	4,818	4,772
6	5,987	5,143	4,757	4,534	4,387	4,284	4,207	4,147	4,099
7	5,591	4,737	4,347	4,120	3,972	3,866	3,787	3,726	3,677
8	5,318	4,459	4,066	3,838	3,687	3,581	3,500	3,438	3,388
9	5,117	4,256	3,863	3,633	3,482	3,374	3,293	3,230	3,179
10	4,965	4,103	3,708	3,478	3,326	3,217	3,135	3,072	3,020

6.- El valor crítico para este tipo de muestra es 3.179, nuestro valor actual de Fisher es 1.0633, se dice que las varianzas de las poblaciones son iguales debido a que la F de Fisher es menor que el valor critico de la tabla.

7.- Ahora que conocemos que nuestras varianzas son iguales, con los datos que ya conocemos aplicamos la prueba de medias dada por el siguiente algoritmo.



8.- Debido a que nuestra muestra es menor a 30, entonces aplicamos la fórmula de la derecha, con la que obtenemos el siguiente resultado **0.977983191**

9.- Comparamos el resultado anterior con la tabla T de Student con 18 grados de libertad y significancia de 0.05.

Grados de libertad	0.25	0.1	0.05	0.025	0.01	0.005
1	1.0000	3.0777	6.3137	12.7062	31.8210	63.6559
2	0.8165	1.8856	2.9200	4.3027	6.9645	9.9250
3	0.7649	1.6377	2.3534	3.1824	4.5407	5.8408
4	0.7407	1.5332	2.1318	2.7765	3.7469	4.6041
5	0.7267	1.4759	2.0150	2.5706	3.3649	4.0321
6	0.7176	1.4398	1.9432	2.4469	3.1427	3.7074
7	0.7111	1.4149	1.8946	2.3646	2.9979	3.4995
8	0.7064	1.3968	1.8595	2.3060	2.8965	3.3554
9	0.7027	1.3830	1.8331	2.2622	2.8214	3.2498
10	0.6998	1.3722	1.8125	2.2281	2.7638	3.1693
11	0.6974	1.3634	1.7959	2.2010	2.7181	3.1058
12	0.6955	1.3562	1.7823	2.1788	2.6810	3.0545
13	0.6938	1.3502	1.7709	2.1604	2.6503	3.0123
14	0.6924	1.3450	1.7613	2.1448	2.6245	2.9768
15	0.6912	1.3406	1.7531	2.1315	2.6025	2.9467
16	0.6901	1.3368	1.7459	2.1199	2.5835	2.9208
17	0.6892	1.3334	1.7396	2.1098	2.5669	2.8982
18	0.6884	1.3304	1.7341	2.1009	2.5524	2.8784
19	0.6876	1.3277	1.7291	2.0930	2.5395	2.8609
20	0.6870	1.3252	1.7247	2.0860	2.5280	2.8452

10.- Al ser el resultado menor que encontramos en la tabla t de student, decimos que pasa las pruebas.

CONCLUSIÓN

En conclusión, el simulador de eventos discretos desarrollado como parte del proyecto integrador de la materia de simulación y el cual es un componente fundamental para comprender y estudiar sistemas complejos en diferentes áreas. A lo largo del proceso de desarrollo, se han abordado conceptos esenciales como números aleatorios, pruebas estadísticas, variables aleatorias y validación del simulador.

El conocimiento de programación es esencial para llevar a cabo un proyecto de simulación exitoso. Comprender los fundamentos de la programación y la estructura de datos es crucial para implementar de manera efectiva los modelos y algoritmos en el simulador. A lo largo del proceso de desarrollo, es probable que se encuentren desafíos y problemas, lo que supone un desafío que fomenta el desarrollo del pensamiento en programación y la resolución de problemas.

Es importante destacar que los simuladores de eventos discretos tienen una amplia gama de aplicaciones en el mundo real. Pueden utilizarse para estudiar y optimizar sistemas de transporte, operaciones logísticas, sistemas de salud, procesos industriales, entre otros. Estos simuladores permiten realizar experimentos y que ayudan a comprender el comportamiento de los sistemas y tomar decisiones informadas en la planificación y mejora de los procesos.

En resumen, el desarrollo y uso de un simulador de eventos discretos como parte de un proyecto integrador de simulación es una experiencia enriquecedora que combina conocimientos de programación, conceptos estadísticos y modelado de sistemas complejos. Esta herramienta tiene un gran potencial para aplicaciones prácticas en el mundo real, brindando la oportunidad de explorar y comprender mejor el comportamiento de sistemas complejos y tomar decisiones más informadas.

REFERENCIAS

Bu, R. C. (1994b). Simulación: un enfoque práctico. Editorial Limusa.

https://books.google.ie/books?id=iY6dI3E0FNUC&printsec=frontcover&dq=Raul+Coss+B%C3%BA&hl=&cd=1&source=gbs_api#v=onepage&q=Raul%20Coss%20B%C3%BA&f=false

Johnsonbaugh, R. (1993). Discrete Mathematics. Simon & Schuster Books For Young Readers.

Beazley, D. M. (2001). Python Essential Reference. Sams Publishing.

Alfonso, U. M., & Carla, M. V. (2013). MODELADO Y SIMULACIÓN DE EVENTOS DISCRETOS. Editorial UNED.

Law, A. M. (2014). Simulation Modeling and Analysis. McGraw-Hill Education.

https://books.google.ie/books?id=5clDnAEACAAJ&dq=simulation+modeling+and+analysis&hl=&cd=1&source=gbs_api

SOPORTE TECNICO

Para obtener soporte técnico, tienes la opción de contactar a los programadores a través de dos vías:

1. Programador: Angel Constantino

- Correo electrónico: l20070572@cdmadero.tecnm.mx
- GitHub: [consmor7](#)

2. Programador: Diego Flores

- Correo electrónico: l21070366@cdmadero.tecnm.mx
- GitHub: [floressdi](#)

Puedes comunicarte con cualquiera de los programadores utilizando su dirección de correo electrónico proporcionada o visitando su perfil de GitHub. Estas vías de contacto están disponibles para que puedas realizar consultas, plantear dudas o hacer sugerencias relacionadas con el programa.

Los programadores estarán encantados de ayudarte y responder a tus inquietudes en la medida de sus posibilidades. Recuerda proporcionar una descripción clara del problema o consulta que tengas para que puedan brindarte la mejor asistencia posible.