
PRACTICA 4

Objetivo General

Conocer y aplicar las virtudes que tiene la ejecución dinámica de procesos mediante la fusión de dos Lenguajes de Programación, C y Python.

Objetivos Particulares:

- Desarrollar un pequeño juego de cartas llamado “Siete y Medio” en lenguaje de programación **C PURO**.
- Implementar un pequeño script para ejecutar un esquema de maestro y esclavo para lograr la ejecución dinámica de procesos en Python mediante la instrucción *execlp()*,
- Tratar de emular la ejecución que realiza el Sistema Operativo con cada una de las aplicaciones que residen en una maquina moderna.

Documentos a entregar

Esta práctica se puede realizar de forma individual o por equipos previamente formados (enviar los integrantes por correo), no existe posibilidad de cambios (agregar más integrantes al equipo)

- El Proyecto Completo, el consiste en un programa en C y un programa en Python, ambos sin uso de IDE's (70 % de la calificación).
- Informe individual de cómo se abordaron cada uno de los problemas planteados (30 % de la calificación). El informe debe contener las mismas secciones de las practicas pasadas y debe realizarse a MANO.

Plazo de entrega

La hora y fecha límite para entregarla será el lunes 25 de febrero del 2019 enviada al correo.

Nota. No se recibirá ninguna práctica fuera de ese horario, sin ninguna excepción.

Especificaciones de las partes que conforman la práctica:

Esta práctica se conforma de dos programas, los cuales serán descritos a continuación:

SieteyMedio.c

Estructura Modular del Juego:

Apoyándonos del Diagrama de Módulos de la Figura 1 (a y b) (similar al Diagrama de Flujos) veremos cuál será la interacción entre los diferentes componentes del.

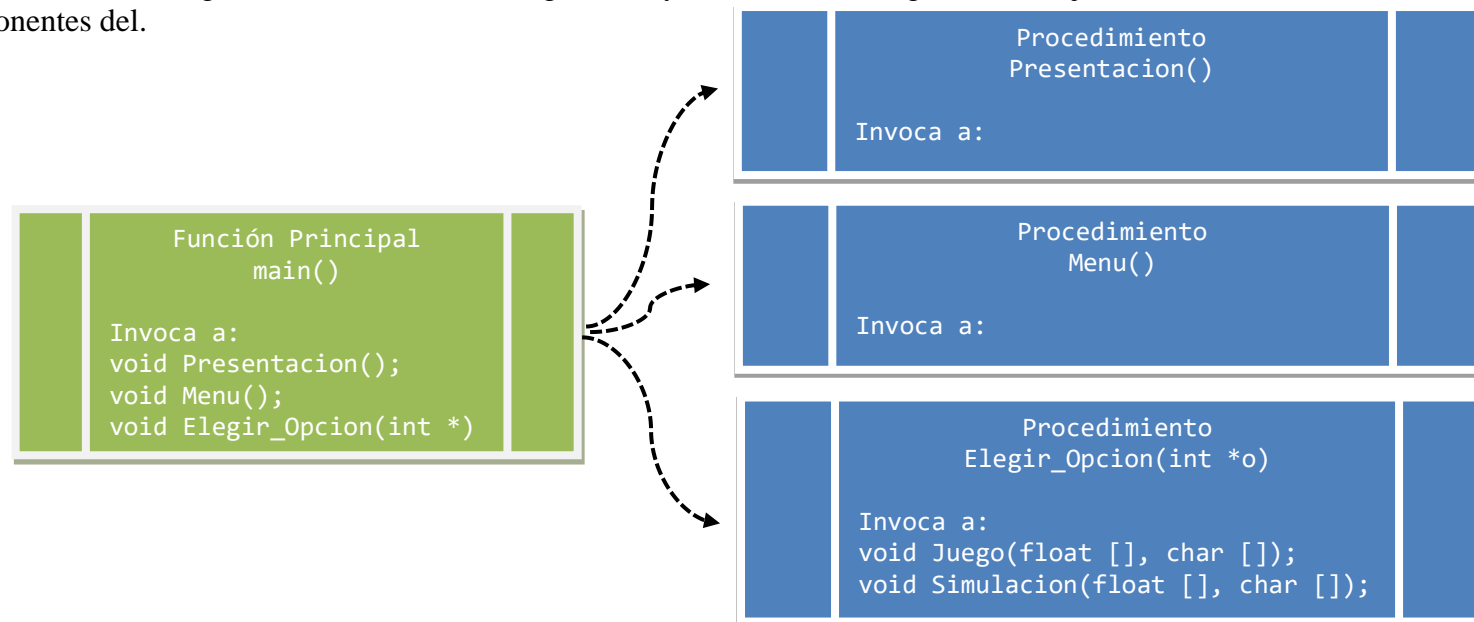


Figura 1_a. Primera interacción del Programa (Presentación y Dirección).

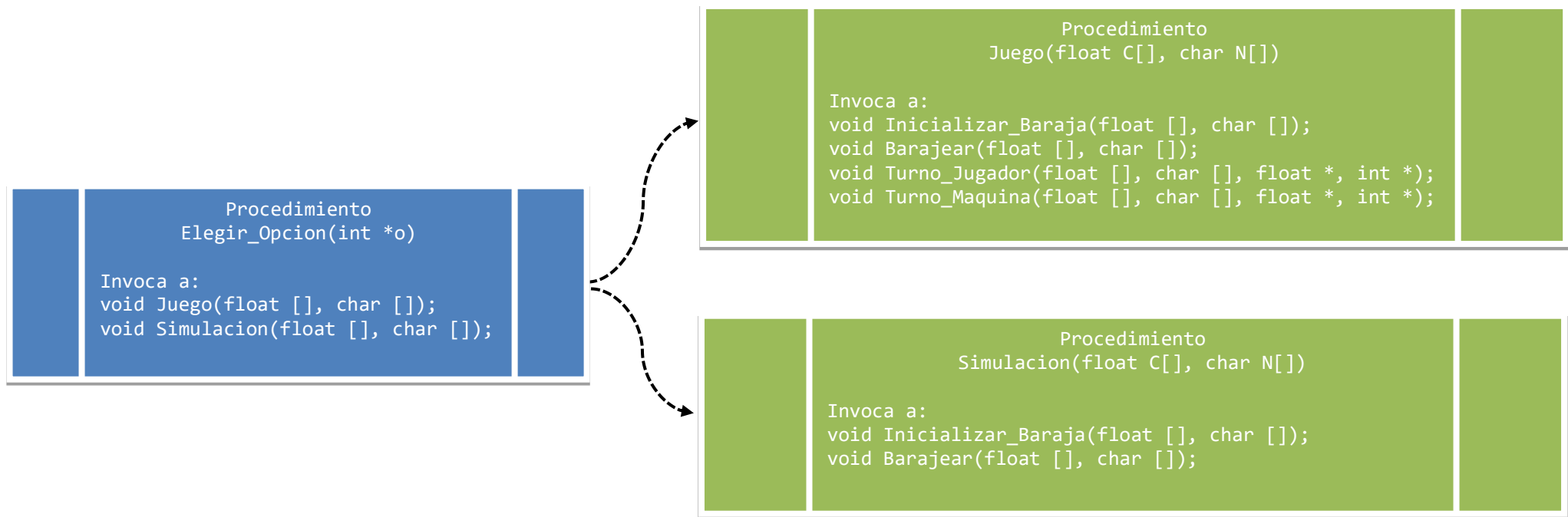
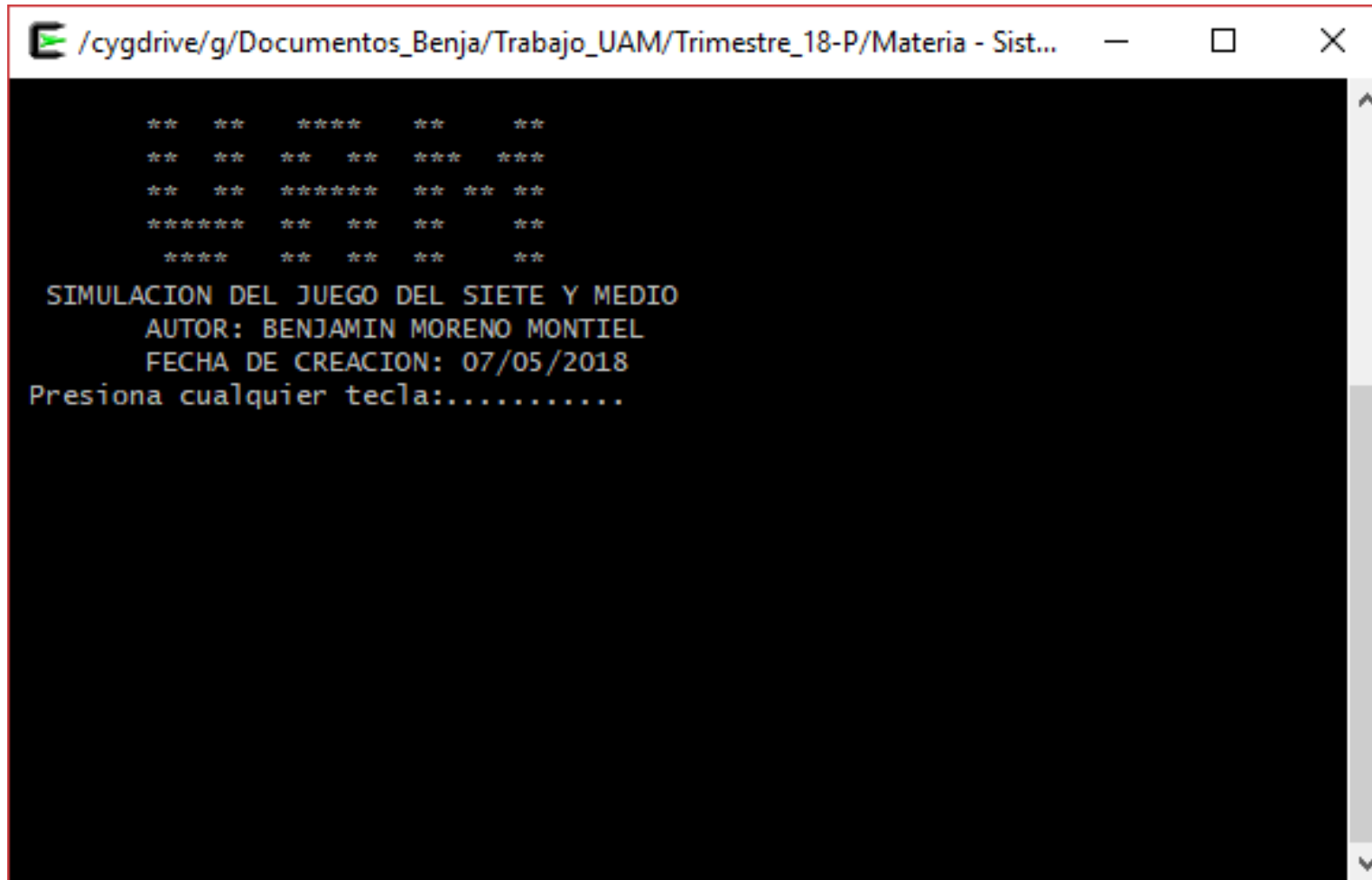


Figura 1_b. Segunda interacción del Programa (Parte Funcional).

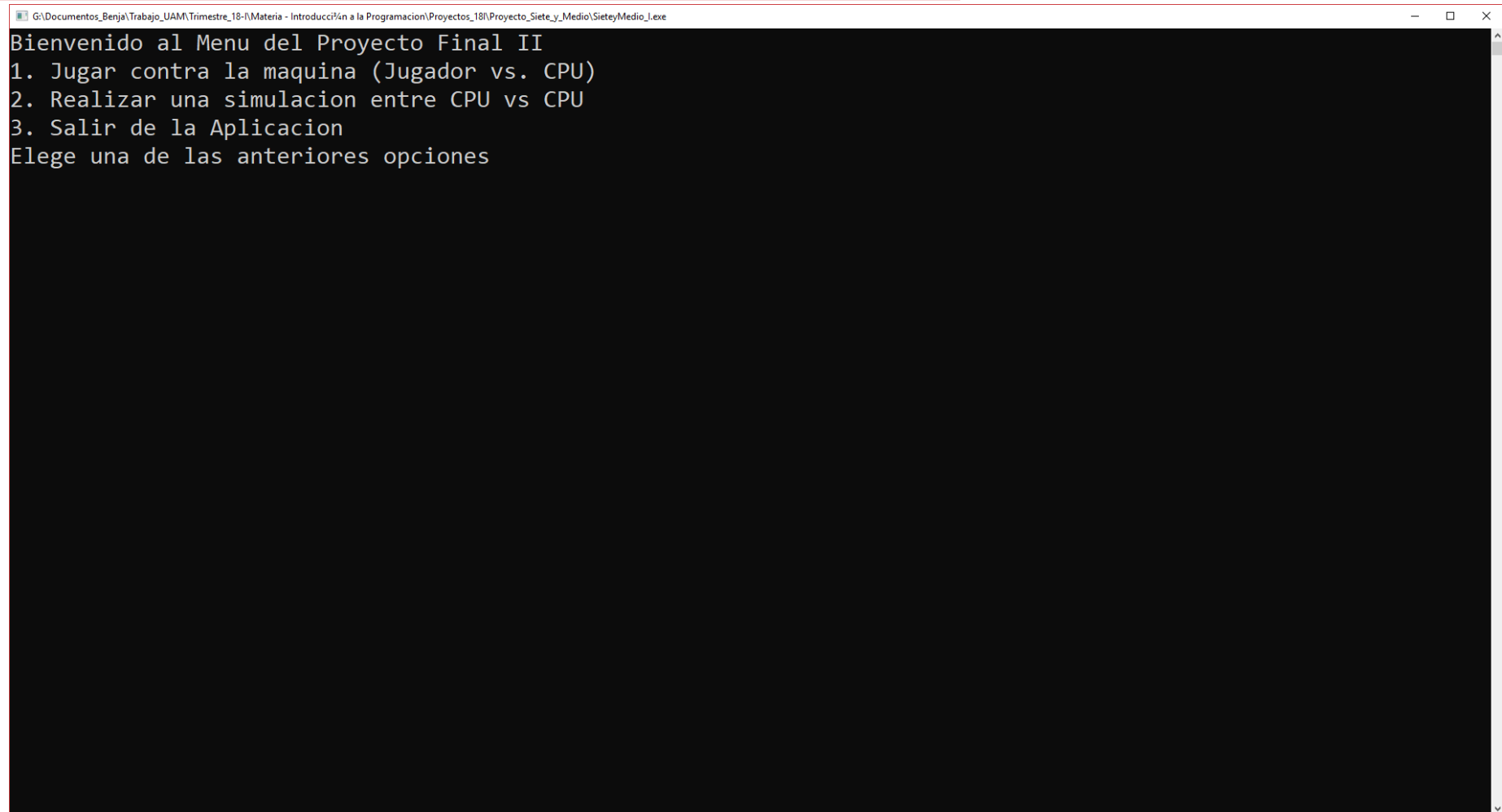
Podemos observar en el Diagrama de Módulo de la Figura 1_a que se tiene la interacción inicial entre la Función `main()` y los Procedimientos `Presentacion()`, `Menu()` y `Elegir_Opcion()`. Para la actividad del lunes se deben implementar las siguientes actividades:

1. El programa debe ser interactivo (ciclico) y esto se debe controlar en la Función Principal.
2. El Procedimiento `Presentacion()` mostrará una caratula del segundo Proyecto Final tal y como se muestra en la Figura 2_a
3. El Procedimiento `Menu()` debe tener 3 opciones que se le muestran al usuario, tal y como se muestran en la Figura 2_b.
4. El Procedimiento `Elegir_Opcion()` representa toda la parte funcional del proyecto, por ello se describirá de forma más detallada.



```
/cygdrive/g/Documents_Benja/Trabajo_UAM/Trimestre_18-P/Materia - Sist...  
  
**  **  *****  **  **  
**  **  **  **  *****  **  
**  **  *****  **  **  **  
*****  **  **  **  **  
*****  **  **  **  **  
  
SIMULACION DEL JUEGO DEL SIETE Y MEDIO  
AUTOR: BENJAMIN MORENO MONTIEL  
FECHA DE CREACION: 07/05/2018  
Presiona cualquier tecla:.....
```

Figura 2_a. Portada inicial de la Practica 4.



A screenshot of a terminal window with a black background and white text. The window title bar at the top shows the file path: G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducci%n a la Programaci%n\Proyectos_18\Proyecto_Siete_y_Medio\SieteyMedio_L.exe. The terminal content displays a menu for 'Proyecto Final II' with three numbered options: 1. Jugar contra la maquina (Jugador vs. CPU), 2. Realizar una simulacion entre CPU vs CPU, and 3. Salir de la Aplicacion. Below the options, it prompts the user to 'Elege una de las anteriores opciones'.

```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducci%n a la Programaci%n\Proyectos_18\Proyecto_Siete_y_Medio\SieteyMedio_L.exe
Bienvenido al Menu del Proyecto Final II
1. Jugar contra la maquina (Jugador vs. CPU)
2. Realizar una simulacion entre CPU vs CPU
3. Salir de la Aplicacion
Elege una de las anteriores opciones
```

Figura 2_b. Formato del menú del Practica 4.

Ahora describiremos el contenido de la Figura 1_b, el cual como habíamos mencionado anteriormente contiene toda la parte funcional (llamado a Módulos Funcionales del Programa).

El primer llamado en el Procedimiento Elegir_Opcion() es al Procedimiento Juego(), en el cual se establecerá la lógica del programa. De acuerdo con el Diagrama de Módulos de la Figura 1_b podemos observar que parámetros de entrada tendrá este Procedimiento, en este caso será un arreglo de reales y de caracteres.

Ya en la implementación del Procedimiento Elegir_Opcion() podemos observar que se harán 4 invocaciones a cuatro procedimientos. Tener en cuenta que el Diagrama de Módulos de la Figura 1_b nos proporciona información del tipo de modulo (Procedimiento o Función) y los parámetros de entrada. Esto anterior se verá reflejado en los prototipos que se deben declarar después de las librerías.

El segundo llamado en el Procedimiento Elegir_Opcion() es al Procedimiento Simulacion(). Si se observa el Diagrama de Modulos de la Figura 1_b, esto nos dará todas las características de los Módulos que son invocados, los cuales son los mismos que se invocan en el Procedimiento Juego(), por lo que NO HAY que reescribirlos o hacer otros diferentes, en particular serían Inicializar_Baraja(), Barajear().

Funciones de control del juego:

Una vez que se revisó la estructura modular del Juego se propone realizar dos de las funcionalidades con la cual se arranca todo juego relacionado con el uso de un tomo de cartas. La primera consiste en simular tener un mazo de cartas inicial y la segunda consiste en barajar un número determinado de veces las cartas. Para esto se recomienda que se observe la demostración del juego que se realice durante la sesión de laboratorio del lunes 4 de junio, algo similar a lo que se muestra en la Figura 3.



Figura 3. Método estándar para barajar un mazo de cartas.

Como se puede ver en la Figura 3 y en la demostración, dado un mazo de cartas es necesario realizar una serie de movimientos para asegurar que el evento sea totalmente aleatorio y sin ningún tipo de trampas de por medio. Pero antes de ahondar en el tema de barajar un Mazo de cartas, veamos cuales son los diferentes palos presentes en la Baraja española lo cuales se muestran en la Figura 4.



Figura 4. Palos presentes en la Baraja Española

En base a la Figura 4, podemos ver que se tienen los Palos: Oros, Copas, Espadas y Bastos, teniendo siempre este orden cuando se toma un mazo nuevo de cartas. En el juego del Siete y Medio como fue explicado en la demostración se tienen cartas con un respectivo valor, estas son:

- El As o la primera carta vale 1
- Las cartas del 2 al 7 su respectivo valor.
- Finalmente se tienen cartas de Figura las cuales son Sota, Caballo y Rey que valen medio punto respectivamente.

En un Mazo de cartas que se abre por primera vez el orden siempre es el mismo, se empiezan con los Oros desde el As hasta el Rey de Oros (orden creciente), se sigue con el de las Copas respetando el orden de numeración de las Cartas. Posteriormente se enumeran las cartas del Palo de las Espadas y finalmente el Palo de los Bastos. En algunos casos el orden puede ser inverso (decreciente), esto es, se empieza desde el Rey de Oros hasta el As de Oros y así sucesivamente hasta llegar al As de Bastos.

Lo que se pretende hacer para este Juego es buscar que tipo de estructura de datos es la más adecuada para emular el orden creciente, empezando con los Oros y Terminando con los bastos de las cartas. Se propone el uso de un arreglo de *float's* y un arreglo de *chas's* pero esto queda abierto a nuevas implementaciones. Tomando en cuenta que los valores del 1 al 7 valen su unidad correspondiente y después tres valores de 0.5 para las cartas de Figura (Sota, Caballo y Rey). El primer objetivo será implementar el funcionamiento del Procedimiento Inicializar_Baraja(), ideando una forma en el que se despliegue el contenido mostrado en la Figura 5.

```
El contenido de la baraja inicial es:
O      O      O      O      O      O      O      O      O      O
1.0    2.0    3.0    4.0    5.0    6.0    7.0    0.5    0.5    0.5
C      C      C      C      C      C      C      C      C      C
1.0    2.0    3.0    4.0    5.0    6.0    7.0    0.5    0.5    0.5
E      E      E      E      E      E      E      E      E      E
1.0    2.0    3.0    4.0    5.0    6.0    7.0    0.5    0.5    0.5
B      B      B      B      B      B      B      B      B      B
1.0    2.0    3.0    4.0    5.0    6.0    7.0    0.5    0.5    0.5
```

Figura 5. Formato del desplegado inicial de las cartas.

Podemos ver en la Figura 5, que se propone un diseño de desplegado en base a los dos arreglos que se utilizaran. En este caso la primera fila del desplegado (después del mensaje) se muestra el carácter ‘O’, haciendo referencia al Palo de los Oros. En la segunda fila se muestra el valor empezando con el 1.0 (As) y terminando con el 0.5 (Rey), esto se debe a que en efecto el parámetro que recibe el arreglo es de tipo *float*. Se muestran todas las cartas de los Oros, después las Copas (‘C’), posteriormente las espadas (‘E’) y finalmente los Bastos (‘B’).

El segundo objetivo que se propone es simular el proceso de barajar las cartas. Como se mostró de manera práctica, este proceso consiste en cambiar de lugar un numero de cartas determinadas por la maestría del tallador (persona que normalmente baraja y reparte las cartas en un casino). Podemos obtener de 0 a 40 intercambios como máximo, ya que el numero 40 representa el número máximo de cartas por tomo Mazo.

El objetivo del proceso de simulación de barajar la estructura de datos propuesta por cada equipo o alumno se realizará mediante el siguiente algoritmo:

Algoritmo de barajar cartas:

1. Se debe generar un numero aleatorio n , entre $[20, 1000]$, el cual representa el número de veces en el que se intercambiaron un par de cartas (una pareja).
2. En base al número n , se debe implementar un ciclo iterativo para que pueda simular cada cambio.
3. Para cada iteración del ciclo iterativo se tendrá que generar un par de valores $c1$ y $c2$, en un rango del $[0, 39]$ que representaran las dos cartas a ser intercambiadas. No importa que los dos valores en un momento dado sean los mismo esto no afectara en nada el funcionamiento del juego.
4. En base a $c1$ y $c2$, se tendrán que intercambiar los valores de la estructura de datos propuesta.
5. El algoritmo finaliza hasta que se alcanza el número máximo de iteraciones.

Juntando la salida anterior (generada en el Procedimiento Inicializar_Baraja()) y la salida del Procedimiento Barajar(), que es donde se debe implementar el algoritmo anterior, se tendría la salida mostrada en la Figura 6,

G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducción a la Programación\Proyectos_18\Proyecto_Siete_y_Medio\SieteMedio_11.exe

El contenido de la baraja inicial es:

0	0	0	0	0	0	0	0	0	0
1.0	2.0	3.0	4.0	5.0	6.0	7.0	0.5	0.5	0.5
C	C	C	C	C	C	C	C	C	C
1.0	2.0	3.0	4.0	5.0	6.0	7.0	0.5	0.5	0.5
E	E	E	E	E	E	E	E	E	E
1.0	2.0	3.0	4.0	5.0	6.0	7.0	0.5	0.5	0.5
B	B	B	B	B	B	B	B	B	B
1.0	2.0	3.0	4.0	5.0	6.0	7.0	0.5	0.5	0.5

El contenido despues de barajear es:

B	0	C	C	B	0	B	0	B	B
4.0	4.0	0.5	3.0	0.5	0.5	2.0	6.0	1.0	3.0
0	C	C	B	E	E	E	0	B	B
3.0	5.0	2.0	0.5	4.0	7.0	1.0	0.5	5.0	7.0
0	C	C	E	C	E	0	E	E	C
7.0	1.0	6.0	3.0	0.5	0.5	5.0	2.0	6.0	4.0
0	B	E	C	0	B	E	0	E	C
1.0	0.5	0.5	7.0	0.5	6.0	0.5	2.0	5.0	0.5

Figura 6. Desplegado del Procedimiento Barajear()

Desarrollo del Juego:

Como se mostró en la explicación práctica, el juego del Siete y Medio consiste en repartir una serie de cartas a un par de jugadores (La Casa y el Jugador) con el fin de que estos alcancen o estén los más cercano al valor de 7.5 (recordar que las figuras tienen el valor de medio punto).

Este juego muy similar al que se realiza con las cartas inglesas llamado Blackjack, en el cual se debe llegar a un puntaje lo más cercano al 21 (dado que hay más cartas disponibles). Es un juego muy popular en los casinos de todo el mundo y se puede realizar entre más de un par de oponentes.

Retomando el juego del Siete y Medio, vamos a establecer un modo de juego 1 a 1. El juego se realiza entre dos personas, el jugador y la casa, por lo que se empieza a repartirle al jugador. Este acumula las cartas necesarias para llegar lo más cercano al siete y medio, sin embargo, cabe la posibilidad de que se pase de este valor y pierda de forma automática. Una vez que se finalice el turno del jugador (sin que este se pase del 7.5), se procede a repartir las cartas de la casa, la cual tendrá en si una estrategia para lograr determinar si pide más cartas o se planta (existe la posibilidad que también se pase del 7.5)

La incertidumbre del juego consiste en que tanto la casa como el jugador no conozcan el puntaje del otro. Para obtener esto siempre se oculta una carta y así genera el real objetivo del juego, esto se puede observar en la Figura 7, en la cual se muestra el desarrollo completo de un juego del siete y medio.

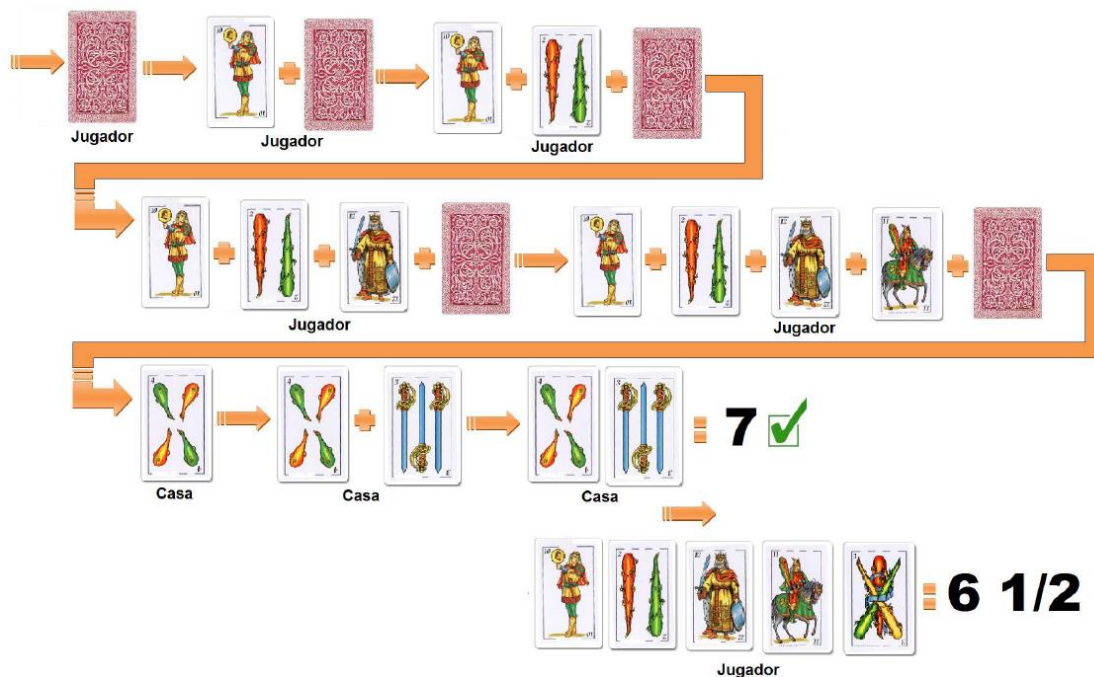


Figura 7. Desarrollo completo del juego del Siete y Medio.

Como se puede ver en la Figura 7, primero se reparte las cartas al jugador. Este decide parar de pedir cartas con un acumulado visible de 3.5. La casa empieza a repartirse sus cartas, no tiene caso ocultar su primera carta ya que solamente es un escenario de juego 1 a 1.

La casa se retira hasta alcanzar el puntaje de 7. Finalmente, el jugador destapa su carta oculta y se verifica que la casa gane. Cabe mencionar que la casa tiene preferencia en el caso de que se presente un empate en el resultado final de la partida.

Este desarrollo del juego se pretende llevar a la simulación mediante el siguiente algoritmo:

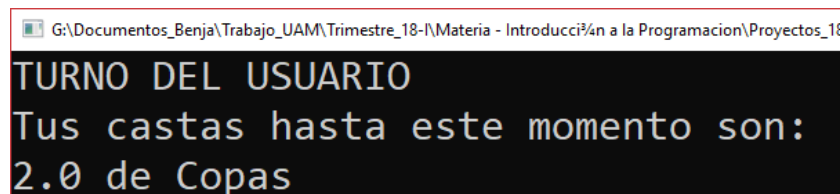
Algoritmo para el Procedimiento Juego():

1. Se debe invocar al Procedimiento Inicializar_Baraja() que se explicó anteriormente.
2. Una vez que se tienen las cartas inicializadas es turno de invocar al Procedimiento Barajear().
3. Se debe implementar una selección iterativa para que el usuario decida hasta qué punto desea terminar el juego (recordar que esto se realiza mediante el ciclo *do-while*).
 - 3.1. Plantear una estructura de tipo *if-else condicionado* para comprobar cada posible caso del desenlace del juego.

Como pudimos observar en el algoritmo anterior, todo el funcionamiento recae en dos funciones booleanas llamadas Turno_Jugador() y Turno_Maquina(). Se explicarán el funcionamiento de ambas funciones de manera separa:

Algoritmo de Función Turno_Jugador:

1. Crear un par de arreglos de reales y caracteres respectivamente. Estos arreglos tendrán un tamaño máximo de 13, ya que son el máximo número de cartas que se pueden acumular.
2. Como se pueden observar en los parámetros de entrada de la función Turno_Jugador() de la Figura 1.2, se tienen dos apuntadores, uno de tipo float (para el puntaje) y otro de tipo int (para recorrer las cartas y su respectivo palo). El apuntador de tipo entero nos permitirá simular la repartición de las cartas ya que ira recorriendo cada una de las cartas. En este segundo paso se necesita inicializar la primera carta que se le asigne al jugador, por lo que los arreglos que fueron declarados en el paso 1, serán inicializados con la primera carta (tanto el arreglo de valor como el de su respectivo Palo ('O', 'C', 'E' y 'B')).
3. Plantear un ciclo infinito para controlar el turno del jugador (preguntar al profesor acerca del ciclo infinito).
 - 2.1. Inicializar el apuntador de tipo *float* en 0 (*p = 0).
 - 2.2. Implementar un ciclo *for* para imprimir cada una de las cartas actuales que se le asignen al jugador.
 - 2.2.1. Dado que se quiere un mensaje más amigable, se tiene que implementar un *switch.case* con el valor del carácter ('O', 'C', 'E' y 'B') tomado del arreglo auxiliar declarado en el paso 1. Los casos de la estructura selectiva tendrán el funcionamiento de imprimir "3 de Oros", en vez de "3 de O". Para ir viendo cómo quedaría este despliegue se puede observar la salida de la Figura 8.



```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducci3n a la Programacion\Proyectos_18
TURNO DEL USUARIO
Tus castas hasta este momento son:
2.0 de Copas
```

Figura 8. Formato del despliegue de cartas.

- 2.2.2. Después de desplegar el valor se debe acumular el valor de carta en el apuntador de tipo *float* que fue pasado como parámetro de entrada.
- 2.3. El paso anterior se realizará un número indeterminado de veces hasta que el usuario decida finalizar el algoritmo (finalizar el ciclo infinito). Es por ello por lo que se le tiene que preguntar si quiere terminar, tal y como se muestra en la Figura 9.

```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-I\Materia - Introducci3n a la Programacion\Proyectos_18\Proy
TURNO DEL USUARIO
Tus castas hasta este momento son:
2.0 de Copas
Tu puntaje es: 2.0
Quieres pedir mas cartas???
1 -> si, 0 -> no: _
```

Figura 9. Contenido del ciclo infinito.

Como podemos ver en el caso anterior, dependiendo de la respuesta del jugador se puede terminar el ciclo infinito con una salida forzada.

- 2.4.Sin embargo, cabe la posibilidad que el usuario pida una carta más si opción es igual a 1, en cuyo caso pueden existir dos escenarios. El primero escenario es que el usuario no se pase del 7.5 y se regrese al paso 2.1. El segundo escenario por lo contrario indicaría que el jugador se pasó del 7.5, por lo que se implementaría una salida forzada regresando 1 (verdadero) al Procedimiento Juego(), por lo que se finalizaría el juego de forma automática, tal y como se observa en la Figura 10.

```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-I\Materia - Introducci3n a la Programacion\Proyectos_18\Proyecto_Siete_y_Medio\SieteyMedio_III
TURNO DEL USUARIO
Tus castas hasta este momento son:
2.0 de Copas
Tu puntaje es: 2.0
Quieres pedir mas cartas???
1 -> si, 0 -> no: 1
Te has exedido del 7.5 (Tu puntaje == 8.0)
HAS PERDIDO LA PARTIDA
VICTORIA PARA LA MAQUINA
Quieres jugar de nuevo???
[s]i, [n]o: _
```

Figura 10. Salida completa de una partida del Siete y Medio.

Podemos observar en la Figura 10, que el programa se queda pausado esperando una respuesta del usuario, lo cual corresponde ciclo *do-while* del paso 3 del Algoritmo del Procedimiento Juego().

El siguiente punto por tratar es el Procedimiento Turno_Maquina y se podrá pensar, que es una cosa muy difícil de realizar. Pues la respuesta es que no, ya que las bases son exactamente lo mismo que se desarrolló en el Procedimiento Turno_Jugador() con algunas modificaciones.

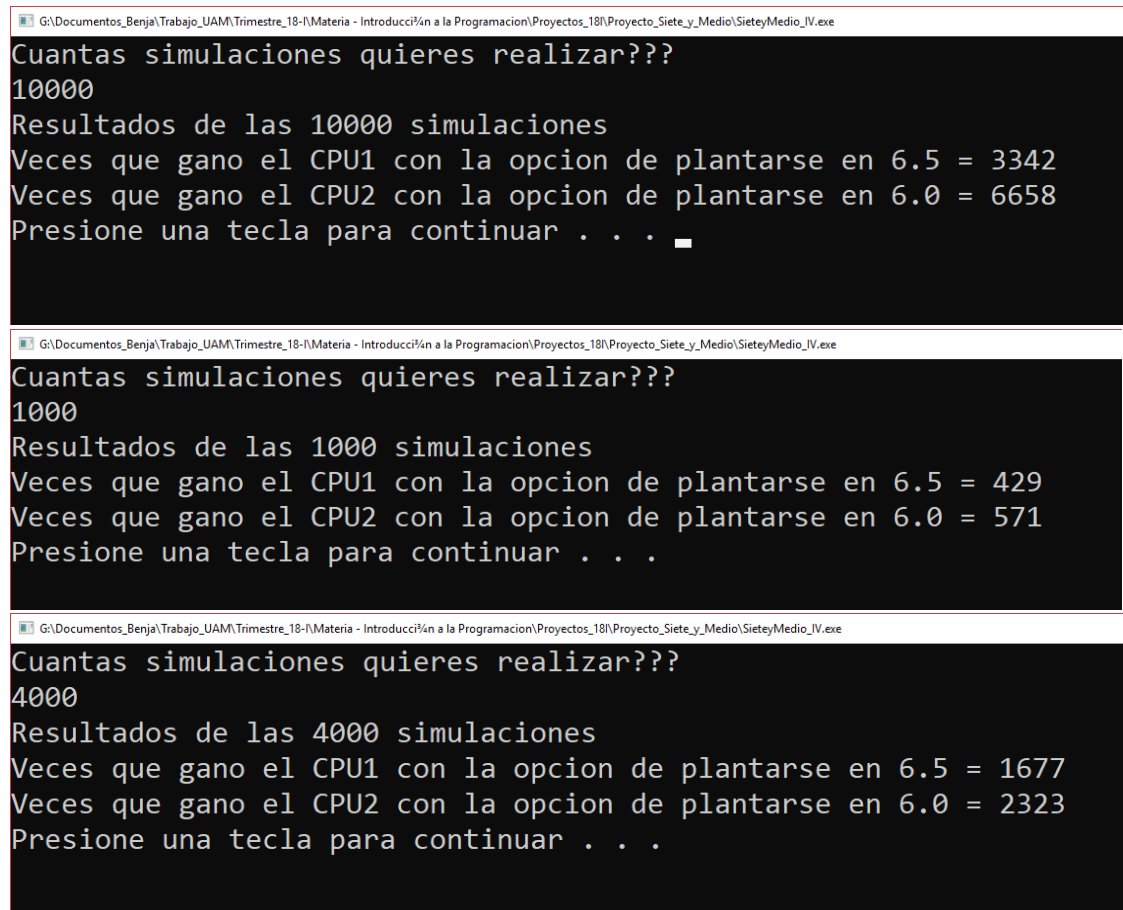
Simulación de Siete y Medio CPU vs CPU:

El objetivo de esta actividad es realizar una simulación entre dos entidades que representen a un par de máquinas con valores de plantarse diferentes (6.5 y 6). El algoritmo que describe el funcionamiento de esta simulación se muestra a continuación:

Algoritmo para el Procedimiento Simunlacion():

1. Ya que aquí no se necesitan desplegar los valores de cartas ni el palo de cada una de ellas, solo se necesitan variables auxiliares como por ejemplo: un par de variables de tipo flotante para almacenar los puntos de cada CPU, un par de variables de tipo entero para almacenar las veces que gana cada CPU, una variable de tipo entero para almacenar el valor del número de simulaciones (leído por teclado), una variable de tipo entero para recorrer las cartas y finalmente variables para los ciclos utilizados.
2. Teniendo el valor de las simulaciones obtenido del paso 1, declarar un ciclo adecuado para realizar cada iteración de forma ordenada y realizar lo siguiente:
 - 2.1. Inicializar la variable para recorrer las casillas del arreglo de valores y tipos de palo de las cartas.
 - 2.2. Invocar el Procedimiento Inicializar_Baraja().
 - 2.3. Invocar el Procedimiento Barajear().
 - 2.4. Implementar un segundo ciclo for para simular los dos turnos de los CPU's y realizar lo siguiente:
 - 2.4.1. Si es el caso del primer CPU usar el valor de plantarse igual a 6.5 y establecer un funcionamiento muy similar al que se desarrolló en la Función Truno_Maquina().}, salvo que en este caso no se pone la instrucción *return* y se pone *break*;
 - 2.4.2. SI se trata del segundo CPU se debe usar el valor de plantarse igual a 6 y el resto del funcionamiento es muy parecido al paso 2.4,1
3. Al llegar a este punto se debe implementar una estructura de tipo if-else condicionado para cubrir los escenarios de victoria o derrota de casa CPU, apoyarse de las Funciones Turno_Jugador() y Turno_Maquin().

4. Como paso final desplegar las veces que gana cada una de las entidades presentes en la simulación. En la Figura 11 se muestran varias ejecuciones diferentes de esta parte final del Segundo Proyecto Final.



The figure consists of three vertically stacked screenshots of a terminal window. Each screenshot shows the execution of a program named 'SieteyMedio_IV.exe'. The program prompts the user for the number of simulations to perform, and then displays the results of those simulations, including the number of wins for CPU1 and CPU2.

```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducci3n a la Programacion\Proyectos_18\Proyecto_Siete_y_Medio\SieteyMedio_IV.exe
Cuantas simulaciones quieres realizar???
10000
Resultados de las 10000 simulaciones
Veces que gana el CPU1 con la opcion de plantarse en 6.5 = 3342
Veces que gana el CPU2 con la opcion de plantarse en 6.0 = 6658
Presione una tecla para continuar . . .
```

```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducci3n a la Programacion\Proyectos_18\Proyecto_Siete_y_Medio\SieteyMedio_IV.exe
Cuantas simulaciones quieres realizar???
1000
Resultados de las 1000 simulaciones
Veces que gana el CPU1 con la opcion de plantarse en 6.5 = 429
Veces que gana el CPU2 con la opcion de plantarse en 6.0 = 571
Presione una tecla para continuar . . .
```

```
G:\Documentos_Benja\Trabajo_UAM\Trimestre_18-19\Materia - Introducci3n a la Programacion\Proyectos_18\Proyecto_Siete_y_Medio\SieteyMedio_IV.exe
Cuantas simulaciones quieres realizar???
4000
Resultados de las 4000 simulaciones
Veces que gana el CPU1 con la opcion de plantarse en 6.5 = 1677
Veces que gana el CPU2 con la opcion de plantarse en 6.0 = 2323
Presione una tecla para continuar . . .
```

Figura 11. Ejecuciones de la parte final del Juego Siete y Medio.

Ejecucion_Dinamica.py:

Para esta parte de la Practica, vamos a proponer un esquema de maestro y esclavo utilizando un par de procesos creados con la llamada al Sistema Operativo *os.fork()*. En la Figura 12 se muestra el esquema de programación multiprocesos que se propone realizar.

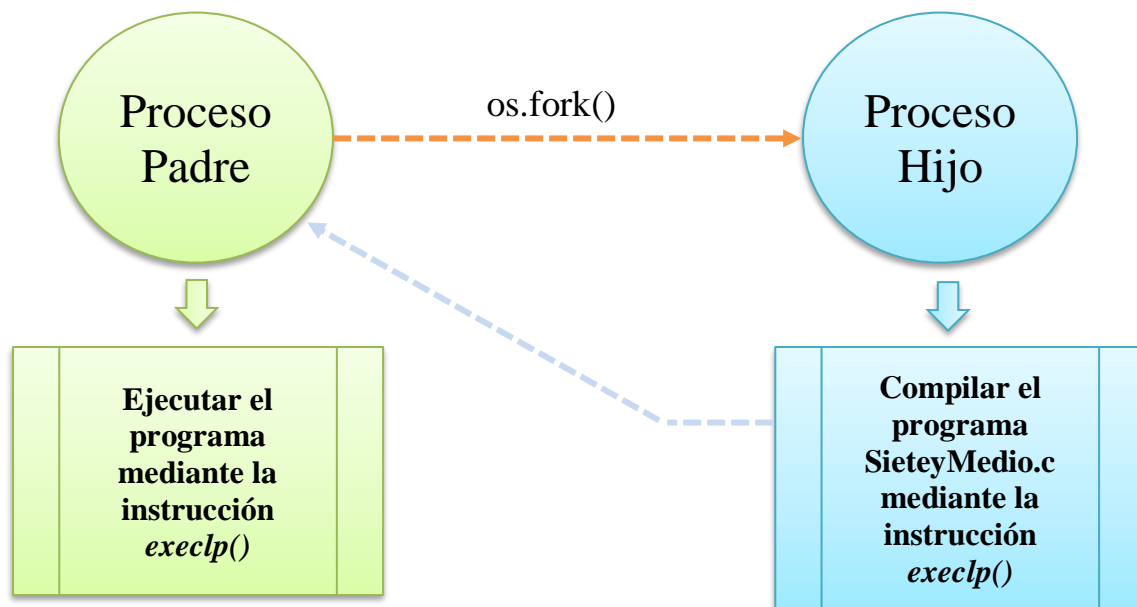


Figura 12. Esquema maestro y esclavo de la Practica 4.

En la Figura 12 podemos observar los siguientes puntos que representan la parte lógica de esta parte de la práctica.

1. El proceso Padre en una parte del código implementado hará una llamada a la función *os.fork()*, generando un hijo que será el esclavo en el esquema Maestro y Esclavo. El proceso quedará en la espera de la finalización del proceso hijo, ya que no podría ejecutar nada antes de que el proceso hijo finalizará el funcionamiento de compilación del programa fuente.
2. Una vez creado el hijo, este procederá a generarle a su Padre Maestro el ejecutable del Juego propuesto en la primera parte de esta práctica. Este proceso se llevará a cabo mediante la función *os.execlp()*, cambiando los parámetros de entrada de la siguiente forma:

execlp("gcc", "gcc", "SieteyMedio.c")

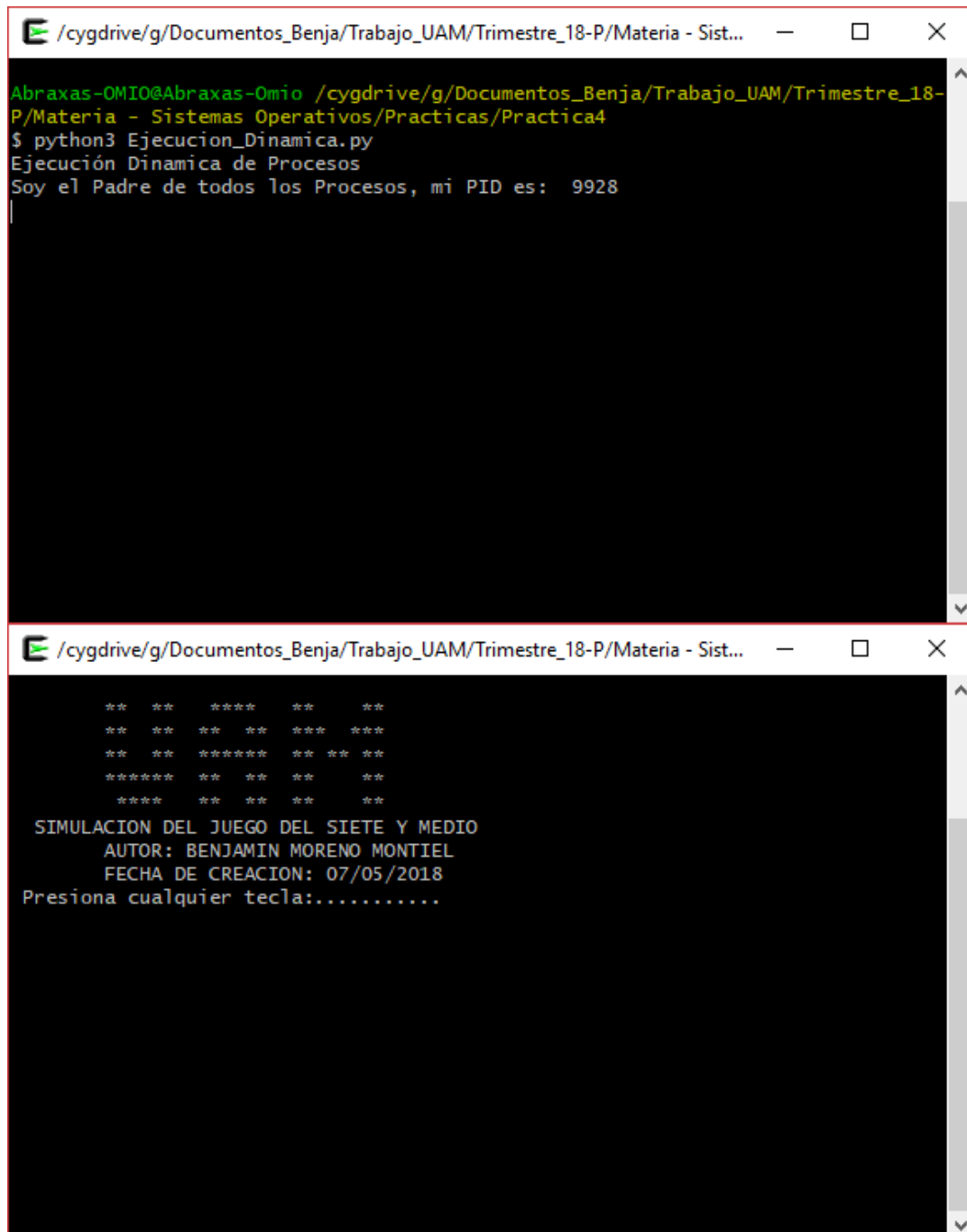
Puede tomar de apoyo la explicación de esta función que viene en la presentación revisada en la Semana 4 del curso

3. Una vez que finaliza de manera correcta el proceso de compilación, este generará un archivo ejecutable genérico llamado *a.exe* en Windows y *a.out* en Linux o Mac.

4. El proceso hijo terminará su funcionamiento y se tendrá que incorporar la función `os.exit(0)`
5. Una vez que el proceso hijo finalizó su tarea, el proceso padre podrá reanudar sus actividades y la mas importante es ejecutar el archivo *a.exe* utilizando el siguiente comando:

`execlp("./a.exe", "a.exe")`

La forma visual de este apartado de la practica se puede visualizar en la Figura 13.



The image shows two screenshots of a terminal window. The top screenshot shows the execution of a Python script named `Ejecucion_Dinamica.py`. The output indicates that the process is the parent of all other processes and its PID is 9928. The bottom screenshot shows the output of a game simulation titled "SIMULACION DEL JUEGO DEL SIETE Y MEDIO". The output includes the author's name, BENJAMIN MORENO MONTIEL, and the creation date, 07/05/2018. The prompt "Presiona cualquier tecla:....." is displayed.

```
/cygdrive/g/Documentos_Benja/Trabajo_UAM/Trimestre_18-P/Materia - Sist...  
Abraxas-OMIO@Abraxas-Omio /cygdrive/g/Documentos_Benja/Trabajo_UAM/Trimestre_18-P/Materia - Sistemas Operativos/Practicas/Practica4  
$ python3 Ejecucion_Dinamica.py  
Ejecución Dinámica de Procesos  
Soy el Padre de todos los Procesos, mi PID es: 9928  
|  
  
** ** ***** ** **  
** ** ** ** ** ***** **  
** ** ***** ** ** **  
***** ** ** ** **  
***** ** ** ** **  
SIMULACION DEL JUEGO DEL SIETE Y MEDIO  
AUTOR: BENJAMIN MORENO MONTIEL  
FECHA DE CREACION: 07/05/2018  
Presiona cualquier tecla:.....
```

Figura 13. Ejecución Final de la Practica 4