Nosotros somos el grupo **The Unknowns** y está compuesto por: Juan Belich, Sol Bilyk, Lucia Graffo, Lautaro Ravanal, Micaela Reartes y Arturo Riera.

Para realizar este proyecto utilizamos el programa de pseint, que es una herramienta educativa creada en [Argentina](https://es.wikipedia.org/wiki/Argentina), utilizada principalmente por estudiantes para aprender los fundamentos de la programación y el desarrollo de la lógica.

PSeInt está pensado para ayudar a los estudiantes que se inician en el mundo de la programación.

El pseudocódigo se suele utilizar como primer contacto para introducir conceptos básicos como el uso de estructuras de control, expresiones, variables, etc, sin tener que lidiar con las particularidades de la [sintaxis](https://es.wikipedia.org/wiki/Sintaxis_(lenguajes_de_programaci%C3%B3n)) de un lenguaje real. Este software pretende facilitarle al principiante la tarea de escribir algoritmos en este pseudolenguaje presentando un conjunto de ayudas y asistencias, y brindarle además algunas herramientas adicionales que le ayuden a encontrar errores y comprender la lógica de los algoritmos.

PSeInt es una herramienta para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos. **MICA**

PSeint incorpora diferentes herramientas de edición: listado de funciones, operadores y variables, resaltado de bloques lógicos, coloreador de sintaxis, indentado inteligente y autocompletado.

Otra característica a destacar de PSeint es que los pseudocódigos que se emplean para escribir los algoritmos pueden transformarse en un diagrama de flujo, facilitando así la tarea de analizarlos y comprenderlos mucho mejor desde un punto de vista gráfico. Los diagramas son editables y es posible escoger entre dos tipos, los clásicos o de Nassi-Shneiderman.

Además de ejecutar el algoritmo para observar cómo funciona y comprobar el resultado final, existe la opción de modificarlo y observar cómo cambia el resultado de inmediato

Tiene algunas funciones importantes como:

**No permitir utilizar variables sin inicializar:** Evita que se utilicen variables que no han sido inicializadas, En caso de estar desactivado, las variables no inicializadas tendrán por defecto los valores 0.

**Controlar el uso del punto y coma:** en nuestro caso es importante porque al utilizar el programa en modo estricto nos indica cuando hemos olvidado colocar un punto y coma.

**Permitir palabras clave como operadores (relacionales):** y, o, no, si.

Palabras reservadas

**Ejecutar:** Ejecuta el algoritmo en una nueva ventana de consola. Si el algoritmo contiene errores de sintaxis, no será ejecutado.

**Ejecutar Paso a Paso:** La ejecución paso a paso permite realizar un seguimiento más detallado de la ejecución del algoritmo. Es decir, permite observar en tiempo real qué instrucciones y en qué orden se ejecutan, así como también observar el contenido de variables o expresiones durante el proceso.

**Verificar Sintaxis**: La verificación de sintaxis permite realizar un seguimiento más detallado de la sintaxis del algoritmo y en caso de encontrar un error mostrará una descripción de la instrucción inválida y sugerencias de cómo solucionar dicho error sintáctico. **SOL**

Habiendo entendido como funciona este programa mediante teoría explicada en clase y la practica con ejercicios durante el primer semestre, tuvimos la idea de crear una juegoteca intectactiva ya sea entre 2 jugadores o un jugador y la computadora aplicando temas aprendidos como Tipos de datos, variables, condicionales simples, dobles y múltiples, estructuras repetitivas como ciclos para mientras y repetir, vectores, ordenamiento, matrices, cadena de caracteres, modularidad.

El programa consiste en un juego donde podremos encontrar 4 opciones diferentes que podrán elegirse desde el menú principal. *(ejecutar y mostrar menú, grabar la pantalla)*

El menú principal está dentro de un ciclo mientras que le permite al usuario poder continuar jugando luego de haber terminado el juego anterior, el ciclo se rompe cuando se ingresa un valor diferente de 5, que del 1 al 5 son las posibles opciones. **JUANO + 5 MIL**

**SUERTE 🡨 CARA O CRUZ sol**

Lo primero que podemos observar es la salida de información que se le indica al usuario ingresar un número siendo el 1 para Suerte, 2 para la Búsqueda del Tesoro, 3 para el 5 mil, 4 para el Ahorcado y 5 para salir del menú. Luego con la función leer el jugador es dirigido el juego que seleccionó.

El primer juego que tenemos es el de suerte o también cara o cruz, nos dirige al subproceso suerte(moneda, tipeado) por valor, *que sabemos que en este tipo de parámetro aunque durante el procedimiento su valor cambie el valor no será asignado a la variable del programa principal.* (teoría)

Se definen las variables num y cara como enteros, agregamos un salto de línea y con el comando Escribir se le pide al usuario que elija entre cara siendo la opción 1 y cruz la opción 2, haciendo la ilusión como si se arrojara una moneda al aire. Con el comando leer, que nos permite ingresar información desde afuera se guarda la opción que el usuario ingreso.

A continuación podemos ver como se usó un ciclo mientras con la finalidad de poder controlar que el usuario solo ingrese las opciones disponibles en la pantalla el 1 o el 2, indicándole que el valor ingresado es incorrecto, en caso de que este ingrese valores que no corresponde el ciclo se continuara hasta que se tipee el 1 o el 2.

Luego para determinar el resultado del juego se utiliza una función, *que sabemos que en programación, es un subprograma o subalgoritmo que toma uno o más valores (argumentos) y devuelve un resultado (valor de la función para los argumentos dados) (teoría),* llamada cara 🡨 azar(1), la función azar guarda los números entre el cero y el número que está entre paréntesis, que en este caso es el 1.

Se colocó un condicional para comparar una variable contra otro(s) valor(es), para que en base al resultado de esta comparación, se siga un curso de acción dentro del programa (teoría), si cara es igual a 0 entonces a moneda se le asigna el valor de 1, sino, si cara es igual a 1 a moneda se le asigna el valor de 2, *La instrucción de asignación permite almacenar una valor en una variable, Al ejecutarse la asignación, primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda. El tipo de la variable y el de la expresión deben coincidir.* (teoría).

Entonces si moneda es igual a 1 con el comando escribir se le muestra al usuario “El resultado del azar es: cara” y si no, si moneda es igual a 2 entonces se le muestra “El resultado del azar es: cruz”.

Finalmente se compara la opción ingresada (tipeado) con el resultado de la función azar (moneda) para saber si el usuario adivino “de qué lado cayo la moneda”, en caso de haber acertado se le muestra un mensaje de felicitaciones por haber ganado y si no acertó vera un mensaje que dice perdiste y dándole la opción de volver a jugar o volver al menú principal.

**EJECUTAR EL JUEGO Y MOSTRARLO**

**BUSQUEDA DEL TESORO 🡪 mica**

El segundo juego es la búsqueda del tesoro, que consiste en encontrar un tesoro dentro de una matriz de 3x3, *este es un arreglo de M\*N elementos organizados en dos dimensiones donde “M” es el número de filas y “N” es el número de columnas. Para representar una matriz se necesita un nombre de matriz acompañado de 2 índices.* (teoría).

En el proceso principal se pueden observar las reglas y luego se llama al subproceso tesoro(mat, intentos). Definimos las variables que vamos a utilizar en proceso, i, j, pos, posTablero como enteros. A posTablero lo inicializamos en 1. Y de esta forma le damos inicio al juego donde habran 4 intentos.

Se comienza con un ciclo para, donde el iterador, desde la posición 0 hasta 2 con paso 1 y el iterador j también desde la posición 0 hasta 2 con paso 1, para cargar a la matriz con espacios en blanco.

Lo siguiente que se hace es crear la función llenarMat para ingresar el tesoro en una posición aleatoria, se le asigna un cero en una posición al azar. Este ciclo sirve para poder mostrar en pantalla la matriz con los números del 1-9 para intentar encontrar el tesoro perdido. Se guarda la variable pos. Luego se realiza un ciclo mientras de control para que no se ingrese un numero fuera del rango que tienen las matrices.

Y dentro de este subproceso llamamos al subproceso comparar(mat, pos, intentos).

Después usando el comando azar se le asigna el cero de forma aleatoria a una posicion en la matriz, que sería el tesoro a encontrar.

A continuación se crea un subproceso nuevo llamado comparar(mat, pos, intentos) donde recorremos la matriz Hasta Que en cierta posicion se encuentre el tesoro y si se encuentra y es la posicion ingresada por el usuario sale por Verdadero con un mensaje que le avisa al usuario que encontro el tesoro, sino se descuenta un intento, tendrá 4 intentos y en caso de no encontrarlo aparecerá en pantalla un mensaje diciendo Perdiste, de lo contrario el ultimo subproceso de este juego llamado mostrarMatGanadora(mar por referencia, pos por valor) donde si encontraste el tesoro.

**5 MIL 🡪 ARTURO Y LUCIA**

**Explicación del código si quieren**

**AHORCADO 🡪 juano + Lautaro (un solo corazón) <3**

En la opción 4 del menú principal podemos encontrar el juego del ahorcado, donde utilizando el comando escribir le damos la bienvenida al usuario, le indicamos las reglas y la cantidad de vidas que tiene.

A continuación le pedimos al jugador 1 que escriba una palabra, esta queda guardada en la variable palabra y para que el segundo jugador no vea la palabra ingresada en la pantalla aparecen **muñequitos** caminando,

Luego a la variable sum2 le asignamos el valor 0, que es la que va a contar las letras.

En la línea siguiente se utiliza un ciclo repetir para pedirle al usuario una letra, que quedara guardada en la variable letra.

Luego la variable h que fue definida como lógica, se le asigna falso, esta es con la que se van a descontar las vidas.

Ademas, aparecerá en pantalla con el comando escribir un mensaje para el usuario indicándole cuantas letras tiene la palabra, con la función longitud. (escribir algo de teoría de cadenas).

Luego llamamos al subproceso recorrerPalabra(palabra, letra por referencia), en este definimos a i como entero. Empezamos con un ciclo Para donde el iterador comienza desde 0 hasta la longitud de la palabra, con paso 1, usando la función subcadena(palabra, i,i) para recorrer la palabra letra por letra, de esta buscamos si la letra ingresada por el usuario coincide con alguna de la palabra. En caso de estar se mostrara en consola la o las posiciones de esa letra.

Despues llamamos a la función vof 🡨 retorno(palabra, letra) que aquí definimos las variables i, j, suma como enteros, vof (verdadero o falso) como lógico.

A la variable suma se le asigna un 0, esta variable es la que va sumando las letras. Continuamos con un ciclo para donde el iterador va desde 0 hasta la longitud de la palabra con paso 1, a la variable j se le asigna cero. Luego dentro de un condicional usamos la funcion subcadena para recorrer la palabra, posicion por posicion y se compara por la letra ingresada por el jugador dos, entonces a la variable j se le asigna un uno porque tiene que incrementar de 1 cuando la letra este en la palabra, después en la variable suma se acumula las veces que se repite la letra en la palabra.

Con otro condicional si decimos que si el acumulador suma es distinto a cero entonces la letra esta en alguna posición de la palabra. Sino, será falso o sea que la letra no esta en la palabra.

(ANDA AL PROCESO PRINCIPAL linea 955 aprox)

Entonces en el juego principal vemos que a la variable h se le asigna la función retorno, esto nos va a decir que si la letra no esta en la palabra se va a descontar una vida en la variable **a**, o sea si h es falso. Sino a sigue siendo a, se siguen teniendo los mismos intentos (no se descuenta nada).

La variable **a** aparece dentro de un condicional si y representa las vidas.

En la siguiente línea a la variable sum1 se le asigna el valor que retorna la función ganaste, en esta función definimos los iteradores i y j como enteros, a j se le asigna el cero y luego dentro de un ciclo para donde al acumulador **i** va desde cero hasta la longitud de la palabra dentro de un condicional Si, se recorre la palabra posicion por posicion comparando si la letra que ingreso el usuario, entonces j es igual a j más uno.

Volvemos al proceso principal y vemos que a la variable sum2 se le acumula el valor que ya tenia mas el valor que tiene la variable sum1 la cual es la que la función ganaste guardo el ultimo valor.

Dentro de un condicional si decimos si sum2 es igual a la longitud de la palabra y **a** es distinto a 1 entonces el usuario adivino las letras de la palabra por lo tanto ha ganado HIZO LA GANACION DEL JUEGO, el juego finaliza porque a la variable a se le asigno un cero, esto es para salir del bucle y volver al menú principal.

Si al valor de **a** se le van restando valores quiere decir que el jugador no esta adivinando las letras, por lo tanto pierde intentos, a medida que va perdiendo va apareciendo en consola un dibujo ilustrativo que representa cada vida que va perdiendo hasta llegar a **a i**gual a 1 que se le terminan las vidas, pierde, saliendo del ciclo repetir, se le muestra al jugador en la pantalla cual era la palabra que tenia que adivinar y finalmente el ciclo se rompe cuando la cantidad de intentos es igual a 1, si es igual a 1 es porque perdió y si es igual a 0 es porque gano pero de una u otra forma sale del ciclo y se muestra el menú principal.

Por ultimo para finalizar el programa principal, el usuario puede elegir la opción 5 del menú y vera el mensaje de gracias por jugar, vuelva pronto.

De esta forma queda terminado el proyecto.