PSeInt, el intérprete de lenguajes de programación basado en pseudocódigo.

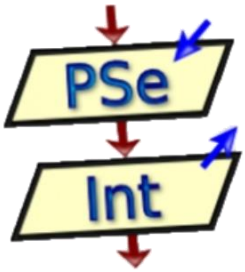

Fig1: Logo de PSeInt[1].

INTRODUCCIÓN

PSeInt se inicia como un proyecto final de la materia de programación I, en la carrera de Ingeniería e Informática de FICH-UNL, en noviembre del 2003.

Luego en diciembre del mismo año, se hace oficial la presentación para **Windows**, en la mesa del examen de programación I. En enero del 2005 se presenta la primera versión para **GNU/Linux.** En agosto de 2006 se agrega la posibilidad de generar **diagramas de flujo.** En febrero de 2008 se agrega un módulo con la posibilidad de convertir el pseudocódigo en **código C++,** en este mismo año se agrega la ejecución paso a paso. En marzo del 2011 sale a la luz primer versión para **Mac OS**, y se agregan los perfiles de lenguaje. En el 2012 ocurren muchos avances como: Se crea el repositorio git, se crea el blog Cucarachas Racing que tiene por objetivo difundir información relaciona a PSeInt, otros dos proyectos, y la programación en general. Primera versión con edición de diagramas de flujo. Primera versión con la posibilidad de crear Subprocesos/Funciones[1].

Primera versión con ejecución paso a paso explicada, que detalla el proceso de interpretación de un algoritmo.

¿QUÉ ES PSEINT?

Es una aplicación informática educativa de software libre, PSeInt es la abreviatura de los estados de computación de **PSe**udocódigo **Int**érprete.
PSeInt sirve para escribir algoritmos en **pseudocódigo** y ejecutarlos, y además genera **diagramas de flujo** de dichos algoritmos. La práctica de escribir algoritmos con PSeInt puede ayudarte a aprender a programar[1].

El pseudocódigo no es un lenguaje de programación, sin embargo, es la forma de escribir un algoritmo con un lenguaje más próximo al nuestro, y con el software PSeInt podemos escribir dicho algoritmo a distintos lenguajes de programación.

Un algoritmo es un conjunto de reglas definidas que permite solucionar un problema, de determinadas maneras, mediante operaciones sistemáticas (no necesariamente ordenadas) y finitas. Estas instrucciones, definidas y ordenadas en función de los datos, resuelven el problema o la tarea. Por ejemplo, hay un algoritmo para resolver el problema de sumar

dos números, uno para encontrar la distancia más corta entre dos puntos o uno para saber si una respuesta es cierta o falsa.

ALGORITMOS EN PSeInt

```
Proceso (nombre_algoritmo)
// Esto es un comentario
instrucción 1;
instrucción 2;
.
.
.
instrucción n;
FinProceso
```

Proceso y FinProceso son las palabras clave que se utilizan para abrir y cerrar, respectivamente, el algoritmo o programa.

Para escribir **comentarios** se empieza la línea con los símbolos «//».

Y después, cada **instrucción** puede consistir en: definir variables, escribir texto por pantalla, leer datos al usuario por teclado, borrar la pantalla, expresiones matemáticas o lógicas, estructuras de control (Si-Entonces, Mientras, Según, Repetir, …), etc.

VARIABLES.

Una **variable** es un espacio de la memoria donde se guarda información, esta información puede ser de diversos tipos y puede ir cambiando a lo largo del programa. A la variable hay que darle un nombre para identificarla, y ese nombre estará formado solo por letras, números y el guion bajo únicamente; no debe contener espacios ni operadores, ni palabras claves del pseudocódigo. Algunos ejemplos de variables correctas pueden ser: nombre, fecha_nacimiento, edad, N, promedio1, …

TIPOS DE DATOS

El **tipo de dato** es el que se le asocia a la variable, por lo que siempre guardará el mismo tipo de dato. Una variable que guarde una cadena no podrá guardar después otro tipo que no sea una cadena. Los tipos de datos en PSeInt pueden ser: NUMERO, ENTERO, REAL, CARACTER, TEXTO, CADENA y LOGICO.

DECLARAR VARIABLE

Se utiliza la palabra clave **Definir,** seguido del nombre de la variable, seguido de la palabra clave **como** y por último el tipo de dato. A continuación, veremos la sintaxis y algunos ejemplos de cómo declarar variables:

```
// Sintaxis para declarar variables
   Definir variable1 Como tipo_de_dato;
   Definir variable2 Como tipo_de_dato;

// Ejemplos
   Definir nombre Como Texto;
   Definir edad Como Entero;
```

ASIGNAR UN VALOR

Una vez declarada la variable se le puede asignar (**<-**) un valor.

```
// Asignación de variables
   variable1 <- valor;

// Ejemplos
   apellido <- "Abrego";
   edad <- 35;
```

LEER Y ESCRIBIR

Podemos pedir por teclado un valor (**Leer**). También existe una palabra clave para mostrar datos por pantalla (**Escribir**). A continuación, se muestra la sintaxis y ejemplos. También tenemos la palabra clave (**Sin Saltar**), la cual nos evita un salto de línea cuando la utilizamos

```
// Lectura de variables
   Leer variable1;
   Leer variable2 Sin Saltar;

// Mostrar datos por pantalla
   Escribir (cadena_texto);
   Escribir variable1;

// Ejemplos
   Escribir "Introduce tu apellido: ";
   Leer apellido;
   Escribir "¿Cuál es tu Edad?: " Sin Saltar;
   Leer edad;
```

ARREGLOS

Los tipos de datos anteriores son simples, además existen los arreglos, que son algo más complejos, son las matrices matemáticas, estructuras de datos homogéneos del mismo tipo que pueden tener varias dimensiones. Para declarar un arreglo se utiliza la palabra clave **Dimensión**, de la siguiente manera:

```
// Sintaxis declaración de un arreglo
   Dimension (nombre_arreglo) [(max1), (max2), ..., (maxN)];
```

```
// Arreglo para almacenar las notas de 2 alumnos para 3 asignaturas.
   Dimension notas [2,3];
        notas [0,0]<-8;
        notas [0,1]<-4;
        notas [0,2]<-8;
        notas [1,0]<-7;
        notas [1,1]<-3;
        notas [1,2]<-8;

// Para mostrarlo por pantalla
        Escribir "Las notas del alumno 1 son:";
        Escribir "Asignatura 1: " Sin Saltar;
        Escribir notas [0,0];
        Escribir "Asignatura 2: " Sin Saltar;
        Escribir notas [0,1];
        Escribir "Asignatura 3: " Sin Saltar;
        Escribir notas [0,2];
        Escribir "Las notas del alumno 2 son:";
        Escribir "Asignatura 1: " Sin Saltar;
        Escribir notas [1,0];
        Escribir "Asignatura 2: " Sin Saltar;
        Escribir notas [1,1];
        Escribir "Asignatura 3: " Sin Saltar;
        Escribir notas [1,2];
```

VEAMOS EL PROGRAMA EJECUTÁNDOSE EN EL SOFTWARE PSEINT

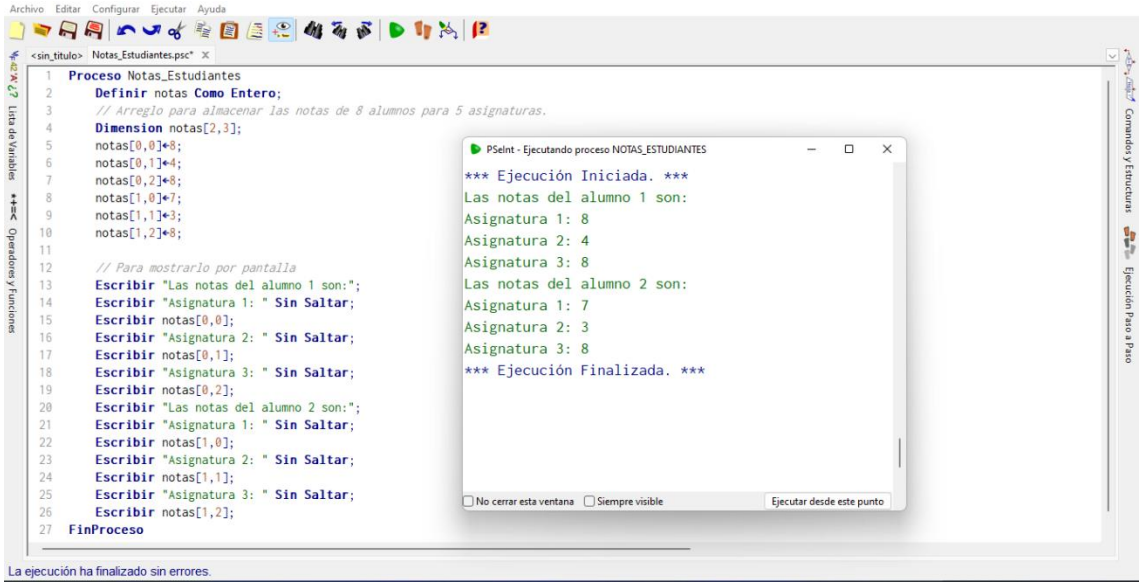

Fig2: Pseudocódigo para captura de notas de 2 estudiantes[1].

DE IGUAL FORMA TAMBIEN PODEMOS VER EL DIAGRAMA DE FLUJO GENERADO.

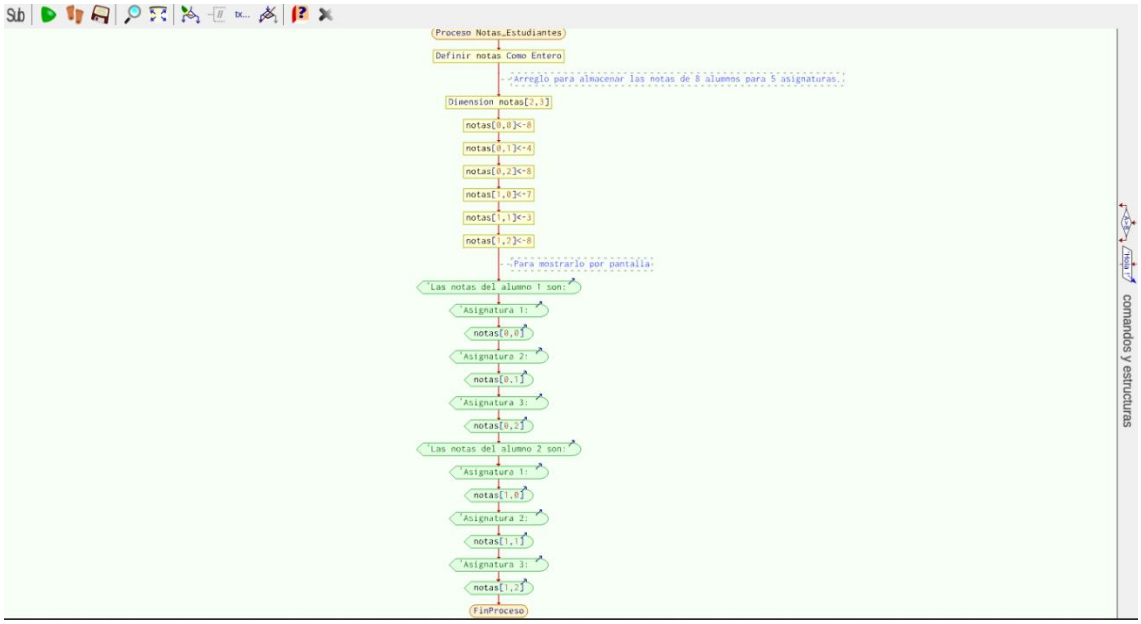

Fig3: Diagrama de Flujo de la captura de notas de 2 estudiantes[1].

EXPRESIONES Y OPERADORES DE LOS ALGORITMOS CON PSEINT

Las expresiones son combinaciones de constantes, variables y operadores que nos permiten trabajar con los datos. Dependiendo de los operadores utilizados en ellas, pueden ser de varios tipos: aritméticas, relacionales, lógicas, alfanuméricas.

Expresión aritmética
Aquella en la que se utilizan operadores aritméticos y como operandos datos numéricos.
+, -, *, /, ^, % o MOD

Expresión relacional
Aquella en la que se utilizan operadores relacionales y el resultado de esta expresión siempre será verdadero o falso.
>, <, >=, <=, =, <>

Expresión lógica
Aquella en la que se utilizan exclusivamente operadores lógicos y el resultado también será siempre verdadero o falso.
Y o &, O o |, NO o ~

Expresión alfanumérica
Aquella que se utiliza para unir cadenas de texto. Se usa el operador de concatenación y como operandos, cadenas de texto.
+

A continuación, algunos ejemplos:

```
// Expresiones aritméticas
   30 + 15;
   22 - 7;
   125 MOD 5;

// Expresiones relacionales
   49 < 35;
   12 = 19;
   10 >= 6;

// Expresiones lógicas
   9 > 1 O 3 < 9;
   15 < 25 Y variable1 = variable2;
   NO (18 < 10);

// Expresiones alfanuméricas
   "Hola Mundo" + ", es hora de triunfar";
```

ANEXOS

VIDEOS EXPLICATIVOS ADICIONALES

Video1

Video2

INFOGRAFÍA

[1] «PSeInt». http://pseint.sourceforge.net/ (accedido feb. 11, 2022).