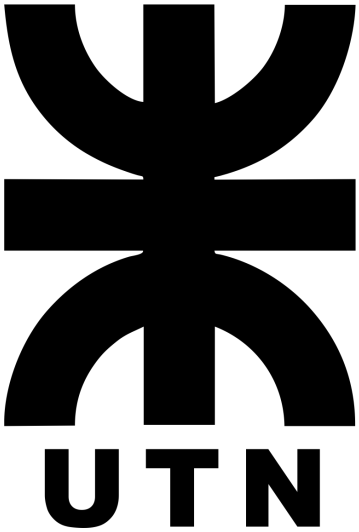
2017



FACULTAD REGIONAL CÓRDOBA

Informe de participación en Comunidad de Software Libre

PROYECTO PSEINT

EMILIANO ANDRADA – LEGAJO 59809

CÁTEDRA DE INGENIERÍA DE SOFTWARE DE FUENTES ABIERTAS / LIBRES

Curso 5K4

PROF. MEDEL, Ricardo Hugo

INTRODUCCIÓN

En el presente informe se muestra el trabajo desarrollado como participante de la comunidad del software libre **PSeInt**, en el contexto de la cátedra de Ingeniería de Software de Fuentes Abiertas / Libres de la UTN Facultad Regional Córdoba.

Se hará foco en los siguientes puntos:

CONTEXTO DEL PROYECTO

¿QUÉ ES PSEINT? OBJETIVO GENERAL

PSeInt es una herramienta que proporciona asistencia en los primeros pasos de la programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), 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.

FUNCIONALIDAD Y CARACTERÍSTICAS

Algunas de las funciones y particularidades más importantes del software:

- Herramientas de edición para escribir algoritmos en pseudocodigo en español.
- Listado de funciones, operadores y variables.
- Generación y edición del diagrama de flujo del algoritmo.
- Puede trabajar con diagramas clásicos y de Nassi-Shneiderman (representaciones gráficas de la programación estructurada)
- Edición simultánea de múltiples algoritmos.
- El pseudolenguaje utilizado es configurable.
- Interpreta y ejecuta los algoritmos escritos.
- Permite ejecutar el algoritmo paso a paso controlando la velocidad e inspeccionando variables y expresiones
- Ofrece un modo especial en el que describe las acciones realizadas en cada paso
- Determina y marca claramente los errores
- Señala errores de sintaxis en tiempo real (mientras escribe)
- Señala claramente errores en tiempo de ejecución
- Ofrece descripciones detalladas de cada error, con sus causas y soluciones más frecuentes.

- Permite convertir el algoritmo de pseudocodigo a código numerosos lenguajes de programación: C, C++, C#, Java, JavaScript, MatLab, Pascal, PHP, Python 2, Python 3, QBasic Visual Basic.
- Es multiplataforma (probado en Microsoft Windows, GNU/Linux y Mac OS X)
- Es totalmente libre y gratuito (licencia GPL versión 2)

LICENCIA GPL

Licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto. Esto garantiza a PSeInt y a sus usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar este software. Si embargo es posible protegerlo (mediante una práctica conocida como copyleft) de intentos de apropiación que restrinjan esas libertades a nuevos usuarios cada vez que la obra es distribuida, modificada o ampliada. Creada originalmente por Richard Stallman y aprobada por la Free Software Foundation (FSF).

DATOS DE ACCESO

Toda la información del proyecto se encuentra en la web, cuya dirección es http://pseint.sourceforge.net/.

Allí, en la sección descargas se puede obtener el paquete de instalación para Windows, Linux, y OS, y también descargar el código fuente.

- Dirección del repositorio:

https://sourceforge.net/p/pseint/code/ci/master/tree/

FUNCIONAMIENTO DE LA COMUNIDAD

ESTRUCTURA, PROCEDIMIENTOS Y TOMA DE DECISIONES

PSeInt no es un proyecto de software libre tal y como se describen en los libros, o como los que podemos encontrar en la mayoría de los repositorios de GitHub. Si bien se distribuye bajo GPL, el único desarrollador (y dueño del producto) es Pablo Novara, y ha sido así casi todo el tiempo que lleva el desarrollo del proyecto. Esto no es así porque no se acepten ayudas, sino porque es difícil encontrar colaboración. La comunidad se centra más en la participación a través de foros –

http://pseint.sourceforge.net/index.php?page=contacto.php, en donde sus miembros hacen aportes acerca de errores, situaciones particulares de algunos algoritmos, sugerencias y reporte de errores. El proyecto recibe retroalimentación constante de sus usuarios y las sugerencias para la mejora son tomadas directamente por Pablo. Sin obviar por supuesto, que el código fuente se encuentra disponible para cualquiera que lo quiera descargar. Pablo además cuenta con la colaboración activa de su esposa, quien es la reguladora de los foros.

Además, hay otro detalle no menor relacionado a PSeInt. Es un software para aprender a definir y ejecutar algoritmos, por lo cual no está en el centro de atención de los grandes desarrolladores y emprendedores de la comunidad, sino por el contrario, por aquellos que recién se inician en el mundo de la programación. Recordemos que el objetivo del programa es dar un soporte analítico e intuitivo a quienes estén empezando con sus primeras líneas de código.

Como consecuencia de lo relatado, la toma de decisiones y los procedimientos asociados a ella se simplifican en este proyecto, ya que hay un solo desarrollador. Es más sencillo para Pablo trabajar sobre su código y, caso de que alguien opte por realizar una colaboración de desarrollo, la puede hacer a través del repositorio cuyo acceso se indicó con anterioridad. Normalmente estos apoyos surgen a través de forks, que son creaciones paralelas del proyecto que incluyen nueva funcionalidad y correcciones en base al software oficial, pero que aguardan la aprobación de Pablo en caso de optar por incorporarlo al código fuente original.

HERRAMIENTAS UTILIZADAS

El código fuente es el mismo tanto para GNU/Linux, Microsoft Windows o Mac OS X (y cualquier otra plataforma en la que logren compilarlo).

Los paquetes que se distribuyen desde este sitio se construyen con gcc y en GNU/Linux (Ubuntu 12.04), aún la versión para de Windows (utilizando mingw y wine en Slackware 14) y las versiones para Mac (utlizando compiladores cruzados gracias a IMCROSS). El instalador para Windows se construye con NSIS. El framework utilizado para el desarrollo de la GUI es wxWidgets, en su versión 2.8 Ansi. El lenguaje empleado en todos los fuentes es C++.

En cada directorio de fuentes hay un archivo .zpr. Este es el archivo de proyecto. Se abren con un IDE llamado ZinjaI. Tambien se encuentran los respectivos makefiles para Windows y GNU/Linux para poder compilar directamente desde la consola.

INFORME FINAL

APORTE REALIZADO

Me pareció útil para la documentación del programa tener un instructivo que permita entender en simples pasos cómo funciona un algoritmo, cuáles son sus componentes, y cómo se aplica su teoría en el funcionamiento de PSeInt. Es por ello que realicé una recopilación de información sobre algoritmos computacionales y lo combiné con todas las funciones que tiene disponible hoy PSeInt para operar. Este es mi aporte:

QUÉ SON LOS ALGORITMOS Y CÓMO DESARROLLARLOS UTILIZANDO PSEINT

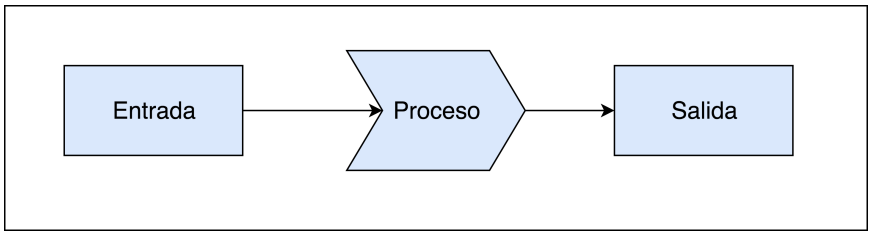
En el contexto matemático, los algoritmos son una serie de normas o leyes específicas que hace posible la ejecución de actividades, cumpliendo una serie de pasos continuos. Los algoritmos se pueden expresar de diversas formas:

- Lenguaje natural
- Lenguaje de programación
- Pseudocódigo
- Diagramas de flujo.

Características importantes de un algoritmo de programación:

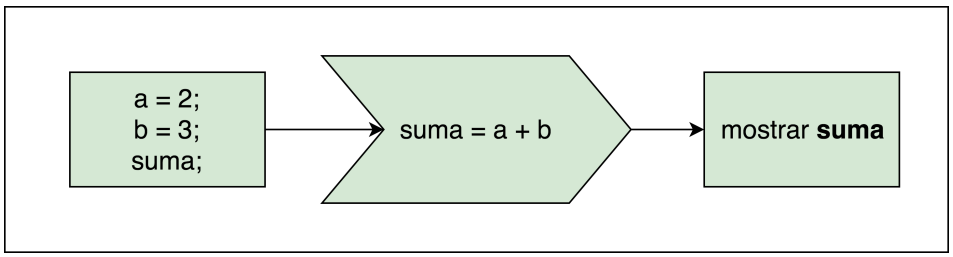
- Debe ser preciso e indicar el orden de realización en cada paso.
- Debe ser definido. Es decir, si ejecuto dos veces el algoritmo con las mismas entradas, tengo que obtener el mismo resultado cada vez.
- Finito: tiene que terminar en algún momento, con una cantidad finita de pasos.

Es muy importante entender cuáles son las propiedades de un algoritmo, cómo se estructura y cuáles son las opciones que tengo para realizarlo, ya que de eso depende el éxito de mi programa. A continuación, se muestra la estructura básica de cualquier algoritmo:



1. **Entrada**: insumo o datos necesarios que requiere el proceso para obtener el resultado.
2. **Proceso**: pasos necesarios para obtener una salida
3. **Salida**: resultados arrojados por el proceso como solución.

Podríamos llevar esto a un ejemplo sencillo, en el cual ingresan dos números, y se necesite sumarlos para mostrar su resultado:

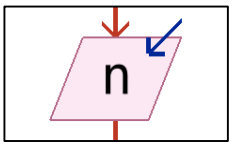


Como se observa parece algo sencillo, pero el proceso en este caso no es más que una simple suma de valores. Cuando tengamos situaciones más complejas, o validaciones que realizar, el bloque del proceso se complejiza y se incorporan nuevas formas de resolver situaciones dentro de un algoritmo.

En PSeInt, podemos encontrar las siguientes figuras asociadas a cada una de las estructuras de un algoritmo:

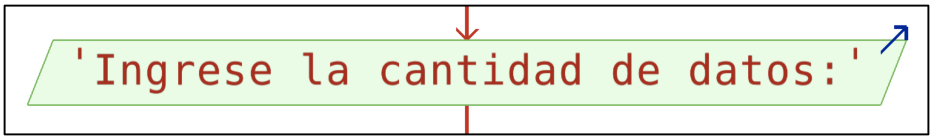
Entrada

- **Lectura de datos**. Aquí se guardará el valor que ingrese como entrada y que representa la cantidad solicitada anteriormente. También se representa con un **paralelogramo**.



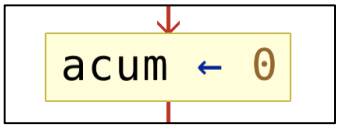
Salida

- **Escribir.** Se representa con un **paralelogramo**. El usuario aquí puede escribir el valor que se asignará a la variable. Se utiliza para generar salidas que luego podrán ser usadas como entrada.



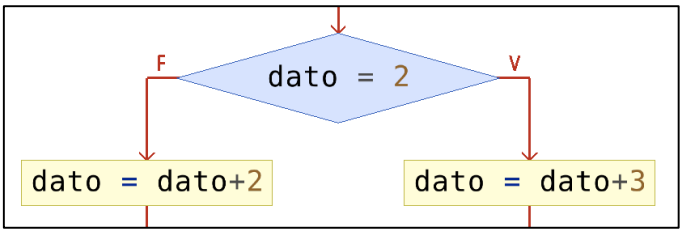
Proceso

- **Definición de variables y asignación de valores**. Se puede definir una variable que va a intervenir en el proceso, y asignarle un valor. Se representa mediante un **rectángulo**.

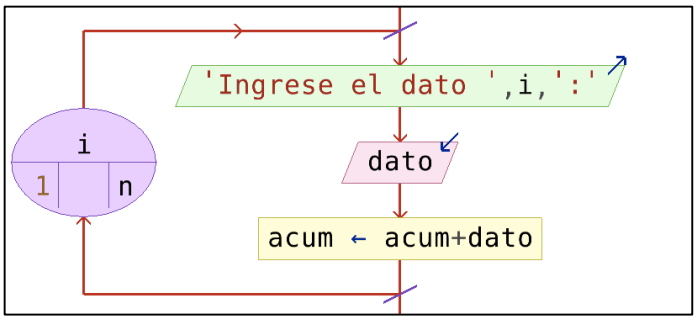


La flecha azul indica que está asignando el **valor 0** a la variable cuyo nombre es **acum.**

- **Estructuras de control**
  - **Control simple**
    - **Si – Entonces.** Se evalúa una condición en función de dos posibilidades: *Verdadero o Falso.* El algoritmo realizará una acción diferente en cada caso. En el ejemplo, si dato es un valor igual a 2, le suma a dicho valor un 2, sino, le suma 3.



- **Repetitivas:**
  - **Para (FOR).** Esta estructura repite **n** veces un conjunto de instrucciones:

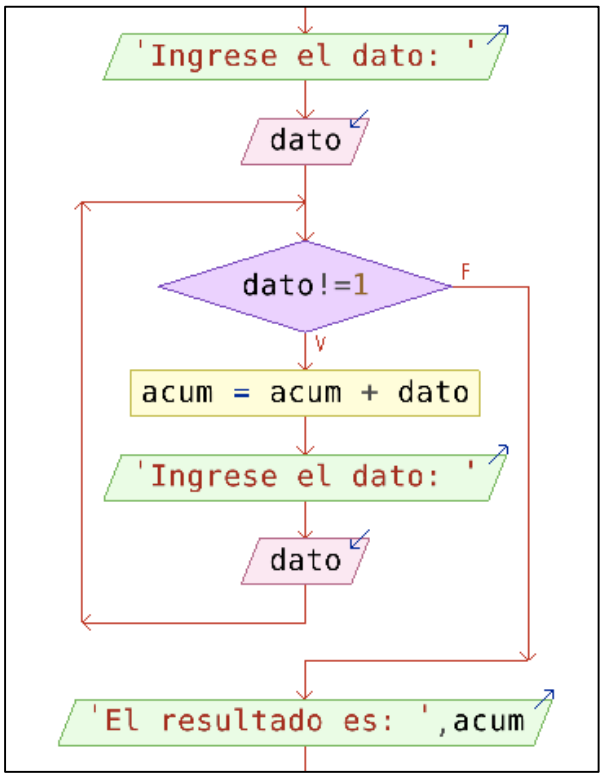


Su funcionamiento general es el siguiente:

1. **Se inicializa el contador.** Esto me dice desde dónde voy a empezar a contar las repeticiones. En nuestro ejemplo será desde el valor 1, asignado a la variable i *(según se ve en el lado izquierdo de la figura ovalada)*.
2. **Se evalúa la condición**. Se realiza para saber si se termina o no el ciclo de repetición. Si el resultado es falso, termina la ejecución de la estructura *para*. Si es verdadero, sigue por el paso 3. En el ejemplo, la condición es *"el proceso se repite **n** veces"*, según el lado derecho de la figura ovalada.
3. Se ejecuta el bloque de instrucciones dentro de la estructura. Para este caso, se suma a una variable **acum**, el valor ingresado en dato.
4. Se evalúa la expresión final, que normalmente altera la variable de control, y se vuelve al paso segundo.

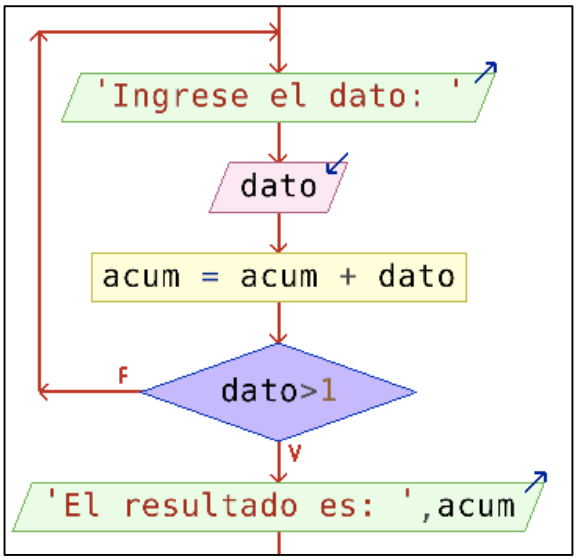
Es importante destacar que el ciclo *para* se ejecuta **siempre n veces.**

- **Mientras (WHILE).** La estructura repetitiva *mientras* funciona igual que el ciclo *para*, pero en este caso no necesariamente se ejecuta un número determinado de veces. Podemos escribir nuestro algoritmo para que así lo haga, pero podría ocurrir que la condición de finalización no esté relacionada con la cantidad de veces que se ha ejecutado el proceso. Por ejemplo:



En el caso anterior, la condición de corte es cuando dato sea igual a 1. Si ingresa un valor 1 en la primera iteración, el ciclo no se ejecuta y muestra el resultado. **Por lo cual, la estructura *mientras* se ejecuta como mínimo 0 veces, y como máximo, según lo que hayamos definido en la condición de control.**

- **Hacer hasta que (DO-WHILE).** Este ciclo funciona igual que la estructura *mientras*, pero el bloque de instrucciones **se ejecuta por lo menos una vez,** y luego se evalúa la condición. Por ejemplo:

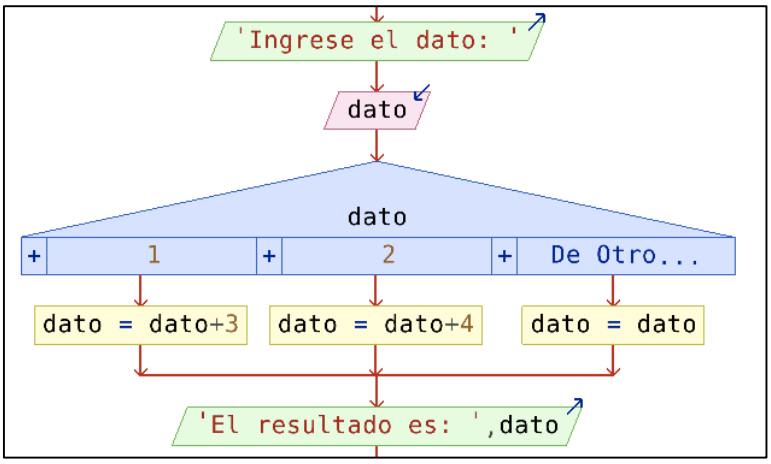


En el caso del ejemplo, sin importar el valor de la variable **dato,** se acumula a la variable **acum**, y luego se evalúa si continúa en el ciclo o no. Si el valor de dato es mayor a 1 se termina el ciclo, de lo contrario, repite hasta que se cumpla la condición.

Resumen:

Estructura	**Cantidad de repeticiones**
Para	**n**
Mientras	**0 a n**
Hacer mientras	**1 a n**

- **Selección múltiple.** En esta estructura el algoritmo sabrá que hacer en función del valor que toma como entrada. Es una forma simplificada de una muchas estructuras *Si-Entonces,* pero el razonamiento es el mismo. Veamos un ejemplo:



En este caso se escribe y se lee la variable dato. y se evalúa su valor. En función de ello, si es un 1 se suma al dato un 3, si es un 2, se suma un 4, y no se suma nada en cualquier otro caso. Esto reemplaza a una estructura más compleja de *Si-Entonces* en caso de que tengamos que bifurcar entre muchas posibilidades.

EXPERIENCIA PERSONAL

En lo personal creo que fue muy enriquecedor involucrarme en un proyecto de estas características. Trabajar en contacto con Pablo, y con miembros de la comunidad me hizo entender la importancia de este tipo de software en la actualidad. La gran mayoría de las veces, los softwares privativos disponen de foros y de lugares donde los usuarios podemos comentar, expresar necesidades y reportar errores, pero por lo general no tenemos una forma directa de llegar al equipo de desarrollo, o de realizar nuestro aporte para que también sirva al resto de la comunidad usuaria. Esto es lo que rescato fundamentalmente del software libre: proyectos que se desarrollan de manera colaborativa y que permiten interactuar a usuarios y desarrolladores con el objetivo de crear sistemas de calidad. Además, entendiendo que la retroalimentación del usuario es muy importante para evaluar el producto, y que sin ellos muchas veces las variables de calidad son poco medibles. Por otro lado, el hecho de que un desarrollador pueda colaborar o, en algunos casos, realizar modificaciones sobre programas o herramientas pensadas por terceras personas, forma una cadena de valor muy importante en nuestra industria, lo cual representa con claridad por qué los productos de software con licencias libres y abiertas son más confiables, robustos y funcionales que los privativos.

ANEXO I – CORREOS ELECTRÓNICOS E INFORMACIÓN PROVISTA POR PABLO

Por razones de simplicidad, acumulé los correos en un archivo almacenado en Drive, ya que algunas respuestas son extensas y ocuparían demasiado espacio en este informe.

El link de acceso a la carpeta es el siguiente: Trabajo Práctico ISWFAL_5K4_2017