



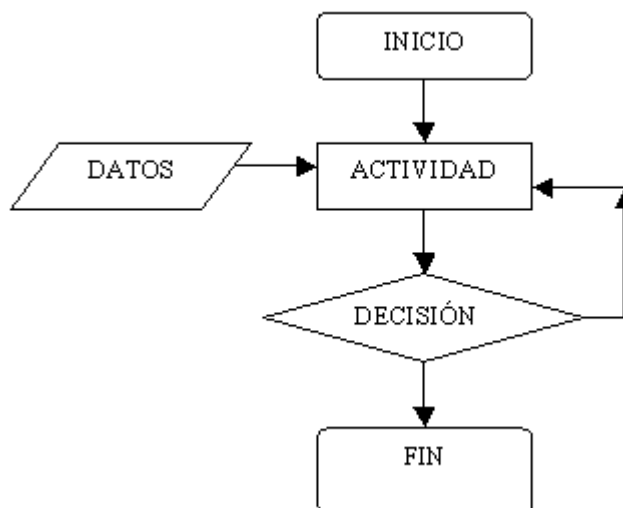
Expresión de Problemas y Algoritmos



Curso de:

**Expresión de
Problemas y
Algoritmos**

- Algoritmos, conceptos teóricos
- Diagramas Nassi-Schneiderman
- Guía de Ejercicios



Profesor : Mgtr. Norberto Charczuk
Ayudante: Lic. Joan Defelippe

Alumno:

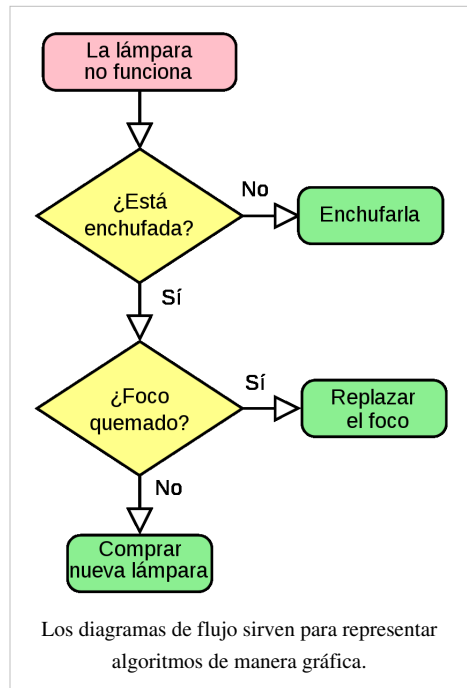


Licenciatura en Sistemas

Algoritmo

En matemáticas, ciencias de la computación y disciplinas relacionadas, un **algoritmo** (del griego y latín, *dixit algorithmus* y éste a su vez del matemático persa Al Juarismi^[1]) es un conjunto preescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.^[2] Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución. Los algoritmos son el objeto de estudio de la **algoritmia**.^[1]

En la vida cotidiana, se emplean algoritmos frecuentemente para resolver problemas. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, o las instrucciones que recibe un trabajador por parte de su patrón. Algunos ejemplos en matemática son el algoritmo de la división para calcular el cociente de dos números, el algoritmo de Euclides para obtener el máximo común divisor de dos enteros positivos, o el método de Gauss para resolver un sistema lineal de ecuaciones.



Definición formal

En general, no existe ningún consenso definitivo en cuanto a la definición formal de algoritmo. Muchos autores los señalan como listas de instrucciones para resolver un problema abstracto, es decir, que un número finito de pasos convierten los datos de un problema (entrada) en una solución (salida).^{[1] [2] [3] [4] [5] [6]} Sin embargo cabe notar que algunos algoritmos no necesariamente tienen que terminar o resolver un problema en particular. Por ejemplo, una versión modificada de la criba de Eratóstenes que nunca termine de calcular números primos no deja de ser un algoritmo.^[7]

A lo largo de la historia varios autores han tratado de definir formalmente a los algoritmos utilizando modelos matemáticos como máquinas de Turing entre otros.^{[8] [9]} Sin embargo, estos modelos están sujetos a un tipo particular de datos como son números, símbolos o gráficas mientras que, en general, los algoritmos funcionan sobre una vasta cantidad de estructuras de datos.^{[3] [1]} En general, la parte común en todas las definiciones se puede resumir en las siguientes tres propiedades siempre y cuando no consideremos algoritmos paralelos:^[7]

Tiempo secuencial. Un algoritmo funciona en tiempo discretizado –paso a paso–, definiendo así una secuencia de estados "*computacionales*" por cada entrada válida (la *entrada* son los datos que se le suministran al algoritmo antes de comenzar).

Estado abstracto. Cada estado computacional puede ser descrito formalmente utilizando una estructura de primer orden y cada algoritmo es independiente de su implementación (los algoritmos son objetos abstractos) de manera que en un algoritmo las estructuras de primer orden son invariantes bajo isomorfismo.

Exploración acotada. La transición de un estado al siguiente queda completamente determinada por una descripción fija y finita; es decir, entre cada estado y el siguiente solamente se puede tomar en cuenta una cantidad fija y limitada de términos del estado actual.

En resumen, un algoritmo es cualquier cosa que funcione paso a paso, donde cada paso se pueda describir sin ambigüedad y sin hacer referencia a una computadora en particular, y además tiene un límite fijo en cuanto a la

cantidad de datos que se pueden leer/escribir en un solo paso. Esta amplia definición abarca tanto a algoritmos prácticos como aquellos que solo funcionan en teoría, por ejemplo el método de Newton y la eliminación de Gauss-Jordan funcionan, al menos en principio, con números de precisión infinita; sin embargo no es posible programar la precisión infinita en una computadora, y no por ello dejan de ser algoritmos.^[10] En particular es posible considerar una cuarta propiedad que puede ser usada para validar la tesis de Church-Turing de que toda función calculable se puede programar en una máquina de Turing (o equivalentemente, en un lenguaje de programación suficientemente general).^[10]

Aritmetizabilidad. Solamente operaciones innegablemente calculables están disponibles en el paso inicial.

Medios de expresión de un algoritmo

Los algoritmos pueden ser expresados de muchas maneras, incluyendo al lenguaje natural, pseudocódigo, diagramas de flujo y lenguajes de programación entre otros. Las descripciones en lenguaje natural tienden a ser ambiguas y extensas. El usar pseudocódigo y diagramas de flujo evita muchas ambigüedades del lenguaje natural. Dichas expresiones son formas más estructuradas para representar algoritmos; no obstante, se mantienen independientes de un lenguaje de programación específico.

La descripción de un algoritmo usualmente se hace en tres niveles:

1. **Descripción de alto nivel.** Se establece el problema, se selecciona un modelo matemático y se explica el algoritmo de manera verbal, posiblemente con ilustraciones y omitiendo detalles.
2. **Descripción formal.** Se usa pseudocódigo para describir la secuencia de pasos que encuentran la solución.
3. **Implementación.** Se muestra el algoritmo expresado en un lenguaje de programación específico o algún objeto capaz de llevar a cabo instrucciones.

También es posible incluir un teorema que demuestre que el algoritmo es correcto, un análisis de complejidad o ambos.

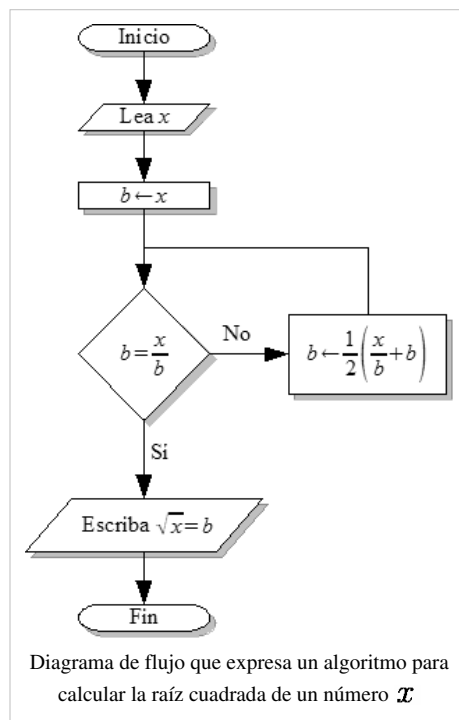
Diagrama de flujo

Los diagramas de flujo son descripciones gráficas de algoritmos; usan símbolos conectados con flechas para indicar la secuencia de instrucciones y están regidos por ISO.

Los diagramas de flujo son usados para representar algoritmos pequeños, ya que abarcan mucho espacio y su construcción es laboriosa. Por su facilidad de lectura son usados como introducción a los algoritmos, descripción de un lenguaje y descripción de procesos a personas ajenas a la computación.

Pseudocódigo

El pseudocódigo (*falso lenguaje*, el prefijo *pseudo* significa *falso*) es una descripción de alto nivel de un algoritmo que emplea una mezcla de lenguaje natural con algunas convenciones sintácticas propias de lenguajes de programación, como asignaciones, ciclos y condicionales, aunque no está regido por ningún estándar. Es utilizado para describir algoritmos en libros y publicaciones científicas, y como producto intermedio durante el desarrollo de un algoritmo, como los diagramas de flujo, aunque presentan una ventaja importante sobre estos, y es que los algoritmos descritos en pseudocódigo requieren menos espacio para representar instrucciones complejas.



El pseudocódigo está pensado para facilitar a las personas el entendimiento de un algoritmo, y por lo tanto puede omitir detalles irrelevantes que son necesarios en una implementación. Programadores diferentes suelen utilizar convenciones distintas, que pueden estar basadas en la sintaxis de lenguajes de programación concretos. Sin embargo, el pseudocódigo en general es comprensible sin necesidad de conocer o utilizar un entorno de programación específico, y es a la vez suficientemente estructurado para que su implementación se pueda hacer directamente a partir de él.

Sistemas formales

La teoría de autómatas y la teoría de funciones recursivas proveen modelos matemáticos que formalizan el concepto de *algoritmo*. Los modelos más comunes son la máquina de Turing, máquina de registro y funciones μ -recursivas. Estos modelos son tan precisos como un lenguaje máquina, careciendo de expresiones coloquiales o ambigüedad, sin embargo se mantienen independientes de cualquier computadora y de cualquier implementación.

Implementación

Muchos algoritmos son ideados para implementarse en un programa. Sin embargo, los algoritmos pueden ser implementados en otros medios, como una red neuronal, un circuito eléctrico o un aparato mecánico y eléctrico. Algunos algoritmos inclusive se diseñan especialmente para implementarse usando lápiz y papel. El algoritmo de multiplicación tradicional, el algoritmo de Euclides, la criba de Eratóstenes y muchas formas de resolver la raíz cuadrada son sólo algunos ejemplos.

Variable

Un elemento que toda pertenece a un dato específico correcto. La declaración se realiza comenzando con **var**. Principalmente, existen dos maneras de otorgar valores iniciales a variables:

1. Mediante una sentencia de asignación.
2. Mediante uno de los procedimientos de entrada de datos (**read** o **readln**).

Ejemplo:

```
...
i:=1;
readln(n);
while i < n do begin
    (* cuerpo del bucle *)
    i := i + 1
end;
...
```

Estructuras secuenciales

La estructura secuencial es aquella en la que una acción sigue a otra en secuencia. Las operaciones se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. La asignación de esto consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor. La asignación se puede clasificar de la siguiente forma:

1. **Simples**: Consiste en pasar un valor constante a una variable ($a \leftarrow 15$)
2. **Contador**: Consiste en usarla como un verificador del número de veces que se realiza un proceso ($a \leftarrow a + 1$)
3. **Acumulador**: Consiste en usarla como un sumador en un proceso ($a \leftarrow a + b$)
4. **De trabajo**: Donde puede recibir el resultado de una operación matemática que involucre muchas variables ($a \leftarrow c + b*2/4$).

Un ejemplo de estructura secuencial, como obtener la área de un triángulo:

```
Inicio
...
    float b, h, a;
    printf("Diga la base");
    scanf("%f", &b);
    printf("Diga la altura");
    scanf("%f", &h);
    a = (b*h)/2;
    printf("El área del triángulo es %f", a)
...
Fin
```

Algoritmos como funciones



Un algoritmo se puede concebir como una función que transforma los datos de un problema (entrada) en los datos de una solución (salida). Más aún, los datos se pueden representar a su vez como secuencias de bits, y en general, de símbolos cualesquiera.^{[1] [9] [11]} Como cada secuencia de bits representa a un número natural (véase Sistema binario), entonces los algoritmos son en esencia funciones de los números naturales en los números naturales que sí se pueden calcular. Es decir que todo algoritmo calcula una función $f : \mathbb{N} \rightarrow \mathbb{N}$ donde cada número natural es la codificación de un problema o de una solución.

En ocasiones los algoritmos son susceptibles de nunca terminar, por ejemplo, cuando entran a un bucle infinito. Cuando esto ocurre, el algoritmo nunca devuelve ningún valor de salida, y podemos decir que la función queda indefinida para ese valor de entrada. Por esta razón se considera que los algoritmos son funciones parciales, es decir, no necesariamente definidas en todo su dominio de definición.

Cuando una función puede ser calculada por medios algorítmicos, sin importar la cantidad de memoria que ocupe o el tiempo que se tarde, se dice que dicha función es computable. No todas las funciones entre secuencias de datos son computables. El problema de la parada es un ejemplo.

Análisis de algoritmos

Como medida de la eficiencia de un algoritmo, se suelen estudiar los recursos (memoria y tiempo) que consume el algoritmo. El análisis de algoritmos se ha desarrollado para obtener valores que de alguna forma indiquen (o especifiquen) la evolución del gasto de tiempo y memoria en función del tamaño de los valores de entrada.

El análisis y estudio de los algoritmos es una disciplina de las ciencias de la computación y, en la mayoría de los casos, su estudio es completamente abstracto sin usar ningún tipo de lenguaje de programación ni cualquier otra implementación; por eso, en ese sentido, comparte las características de las disciplinas matemáticas. Así, el análisis de los algoritmos se centra en los principios básicos del algoritmo, no en los de la implementación particular. Una forma de plasmar (o algunas veces "codificar") un algoritmo es escribirlo en pseudocódigo o utilizar un lenguaje muy simple tal como Lexico, cuyos códigos pueden estar en el idioma del programador.

Algunos escritores restringen la definición de algoritmo a procedimientos que deben acabar en algún momento, mientras que otros consideran procedimientos que podrían ejecutarse eternamente sin pararse, suponiendo el caso en el que existiera algún dispositivo físico que fuera capaz de funcionar eternamente. En este último caso, la finalización con éxito del algoritmo no se podría definir como la terminación de éste con una salida satisfactoria, sino que el éxito estaría definido en función de las secuencias de salidas dadas durante un periodo de vida de la ejecución del algoritmo. Por ejemplo, un algoritmo que verifica que hay más ceros que unos en una secuencia binaria infinita debe ejecutarse siempre para que pueda devolver un valor útil. Si se implementa correctamente, el valor devuelto por el algoritmo será válido, hasta que evalúe el siguiente dígito binario. De esta forma, mientras evalúa la siguiente secuencia podrán leerse dos tipos de señales: una señal positiva (en el caso de que el número de ceros sea mayor que el de unos) y una negativa en caso contrario. Finalmente, la salida de este algoritmo se define como la devolución de valores exclusivamente positivos si hay más ceros que unos en la secuencia y, en cualquier otro caso, devolverá una mezcla de señales positivas y negativas.

Ejemplo de algoritmo

El problema consiste en encontrar el máximo de un conjunto de números. Para un ejemplo más complejo véase Algoritmo de Euclides.

Descripción de alto nivel

Dado un conjunto finito C de números, se tiene el problema de encontrar el número más grande. Sin pérdida de generalidad se puede asumir que dicho conjunto no es vacío y que sus elementos están numerados como c_0, c_1, \dots, c_n .

Es decir, dado un conjunto $C = \{c_0, c_1, \dots, c_n\}$ se pide encontrar m tal que $x \leq m$ para todo elemento x que pertenece al conjunto C .

Para encontrar el elemento máximo, se asume que el primer elemento (c_0) es el máximo; luego, se recorre el conjunto y se compara cada valor con el valor del máximo número encontrado hasta ese momento. En el caso que un elemento sea mayor que el máximo, se asigna su valor al máximo. Cuando se termina de recorrer la lista, el máximo número que se ha encontrado es el máximo de todo el conjunto.

Descripción formal

El algoritmo puede ser escrito de una manera más formal en el siguiente pseudocódigo:

Sobre la notación:

- " \leftarrow " representa una asignación: $m \leftarrow x$ significa que la variable m toma el valor de x ;
- "**devolver**" termina el algoritmo y devuelve el valor a su derecha (en este caso, el máximo de C).

Implementación

En lenguaje C++:

```
int max(int c[], int n) {
    int i, m = c[0];
    for (i = 1; i < n; i++)
        if (c[i] > m) m = c[i];
    return m;
}
```

Referencias

- [1] Brassard, Gilles; Bratley, Paul (1997). *Fundamentos de Algoritmia*. Madrid: PRENTICE HALL. ISBN 84-89660-00-X.
- [2] Real Academia Española. Diccionario de la lengua española (http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&TIPO_BUS=3&LEMA=algoritmo) "*Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.*"
- [3] Cormen, Thomas; Leiserson, Charles; Rivest, Ronald; Stein, Clifford (2009). *Introduction to algorithms*. Cambridge, Massachusetts: The MIT Press. ISBN 978-0-262-53305-8.
- [4] Ralph P. Grimaldi (1998). «Propiedades de los números enteros: Inducción matemática». *Matemáticas Discreta y Combinatoria*. México: Addison Wesley Longman de México. ISBN 968-444-324-2.
- [5] Johnsonbaugh, Richard (2005). «Introducción a la teoría de números». *Matemáticas Discretas*. México: PEARSON EDUCACIÓN. ISBN 970-26-0637-3.
- [6] Carl Reynolds & Paul Tymann (2008). *Schaum's Outline of Principles of Computer Science*. McGraw-Hill. ISBN 978-0-07-146051-4.
- [7] Gurevich, Yuri (2000). « Sequential Abstract State Machines capture Sequential Algorithms (<http://research.microsoft.com/en-us/um/people/gurevich/Opera/141.pdf>)». *ACM Transactions on Computational Logic* **1** (1). ISSN 1529-3785, 77-111. .
- [8] John E. Savage (1987). *The Complexity of Computing*. Krieger Publishing Co.. ISBN 089874833X.
- [9] [Sipser, Michael (<http://www-math.mit.edu/~sipser/>)] (2005). *Introduction to the Theory of Computation* (2 edición). Course Technology. ISBN 978-0534950972.
- [10] Nachum Dershowitz & Yuri Gurevich (2008). « A natural axiomatization of computability and proof of Church's Thesis (<http://research.microsoft.com/en-us/um/people/gurevich/Opera/188.pdf>)». *Bulletin of Symbolic Logic* **14** (3). ISSN 10798986, 299-350. .
- [11] [Kelley, Dean (<http://krypton.mnsu.edu/~kelled/>)] (1995). *Teoría de Autómatas y Lenguajes Formales*. Prentice Hall. ISBN 0-13-497777-7.

Bibliografía

- *Fundamentos de Algoritmia*, G. Brassard y P. Bratley. (ISBN 848966000)
- *The Art of Computer Programming*, Knuth, D. E. [quien fue también, el creador del TeX]
- *Introduction to Algorithms (2nd ed)*, Cormen, T. H., Leiserson, C. E., Rivest, R. L. y Stein, C.
- *Introduction to Algorithms. A Creative Approach*, Mamber, U.
- *Algorithms in C (3r ed)*, Sedgewick, R. (también existen versiones en C++ y Java)
- *The Design and Analysis of Computer Algorithms*, Aho, A.

Véase también

Tipos de algoritmos según su función

- Algoritmo de ordenamiento
- Algoritmo de búsqueda

Técnicas de diseño de algoritmos

- Algoritmos voraces (greedy): seleccionan los elementos más prometedores del conjunto de candidatos hasta encontrar una solución. En la mayoría de los casos la solución no es óptima.
- Algoritmos paralelos: permiten la división de un problema en subproblemas de forma que se puedan ejecutar de forma simultánea en varios procesadores.
- Algoritmos probabilísticos: algunos de los pasos de este tipo de algoritmos están en función de valores pseudoaleatorios.
- Algoritmos determinísticos: el comportamiento del algoritmo es lineal: cada paso del algoritmo tiene únicamente un paso sucesor y otro antecesor.
- Algoritmos no determinísticos: el comportamiento del algoritmo tiene forma de árbol y a cada paso del algoritmo puede bifurcarse a cualquier número de pasos inmediatamente posteriores, además todas las ramas se ejecutan simultáneamente.
- Divide y vencerás: dividen el problema en subconjuntos disjuntos obteniendo una solución de cada uno de ellos para después unirlos, logrando así la solución al problema completo.

- Metaheurísticas: encuentran soluciones aproximadas (no óptimas) a problemas basándose en un conocimiento anterior (a veces llamado experiencia) de los mismos.
- Programación dinámica: intenta resolver problemas disminuyendo su coste computacional aumentando el coste espacial.
- Ramificación y acotación: se basa en la construcción de las soluciones al problema mediante un árbol implícito que se recorre de forma controlada encontrando las mejores soluciones.
- Vuelta atrás (backtracking): se construye el espacio de soluciones del problema en un árbol que se examina completamente, almacenando las soluciones menos costosas.

Temas relacionados



- Cota superior asintótica
- Cota inferior asintótica
- Cota ajustada asintótica
- Complejidad computacional
- Diagramas de flujo
- Máquina de Turing

Disciplinas relacionadas

- Ciencias de la Computación
- Análisis de algoritmos
- Complejidad computacional
- Informática
- Inteligencia artificial
- Investigación operativa
- Matemáticas
- Programación
- *Método de las 6'D. Modelamiento - Algoritmo - Programación. Enfoque orientado a las estructuras lógicas (2da ed.)* (<http://jjflorescueto.googlepages.com/3erlibro>), Juan José Flores Cueto y Carmen Bertolotti Zuñiga, 2008.
- *Método de las 6'D. Modelamiento - Algoritmo - Programación. Enfoque orientado a los arreglos (1ra ed.)* (<http://jjflorescueto.googlepages.com/4tolibro>), Juan José Flores Cueto y Gustavo Tataje Salas, 2009.

Enlaces externos

Wikilibros

-  Wikilibros alberga un libro o manual sobre **Algoritmia**.
-  Wikcionario tiene definiciones para **algoritmo**. Wikcionario
- Portal de algoritmia (<http://www.algoritmia.net>)
- Portal de algoritmos básicos (<http://www.algoritmos.tk>)
- Técnicas de Diseño de Algoritmos (<http://www.lcc.uma.es/~av/Libro/>) manual que explica y ejemplifica los distintos paradigmas de diseño de algoritmos. Rosa Guerequeta y Antonio Vallecillo (profesores de la Universidad de Málaga).
- Transparencias de la asignatura "Esquemas Algorítmicos", Campos, J. (<http://webdiis.unizar.es/asignaturas/EDA/>)
- Apuntes y problemas de Algorítmica por Domingo Giménez Cánovas (<http://dis.um.es/~domingo/alg.html>)
- Curso de Diseño de Algoritmos de Carlos Pes (http://www.carlospes.com/curso_de_algoritmos/)
- Algoritmos y Diagramas de Flujo (<http://snippets-tricks.org/algoritmos-y-diagramas-de-flujo/>)

-
- Cómo hacer (<http://www.martinturnes.com.ar/home/?tpl=home&area=como-hacer>)

Fuentes y contribuyentes del artículo

Algoritmo *Fuente:* <http://es.wikipedia.org/w/index.php?oldid=44128063> *Contribuyentes:* .Sergio, AS990, Abel406, AchedDamiman, Adrianantoniors, Airunp, Aitorzubiaurre, AlbertMA, Albertochoa, Aleator, Alexav8, AlfonsoERomero, Alhen, AlphaWiki, Alvaro qc, Amadís, Angel GN, Angus, Antur, Aquiel, Arlm1, Açigni-Lovrij, Baiji, Balderai, Banfield, BlackBeast, BuenaGente, CASF, Caizer, Calitb, Camilo, Camima, Carlo el calvo, Carmin, Carutsu, Chabacano, Chuchot, Cinabrium, Clarad, Cratón, Crescent Moon, Ctrl Z, Danielba894, David0811, Der Kreole, DerHexer, Dferg, Diegusjaimes, Dodo, Dogor, Dorico, Drake 81, Dromero, Ecemaml, Edgar, Eduardosalg, Edub, Eduman, Efegé, Ejmeza, Elabra sanchez, Elisardojm, Elwikipedista, Emijrp, Er Komandante, Ezarate, FAR, Farisori, Fegc77, Fide07, Flakinho, Francisco Mochis, Fsd141, GermanX, Gizmo II, Gothmog, Guillervf91, Gusgus, H3r3dia, HUB, Haitike, Hantartico, Heynry1, Huhsunqu, Humberto, Ictlogist, Ignacio Icke, Imperioonepiece, Ingenioso Hidalgo, Irvinopuma, Isha, JAAC, JMPerez, Jarisleif, Jarke, Javierito92, Jecanre, Jesuja, Jhoelito14, Jjlflorescueto, Jkbw, JorgeGG, Jorgeu, Jsanchezes, Jstitch, JuanRodríguez, Jugones55, Julie, Junior1209, Kn, KnightRider, Komputisto, Kved, Laura Fiorucci, Lcpousa, Lecuona, Libertad y Saber, Llull, Lourdes Cardenal, M.heda, Mansonce, ManuelGR, Manwë, Mar del Sur, Matdroses, McMalamute, Miguel hdez, MiguelAngel fotografo, Mipataentutrasero, MorZilla, Moriel, Mortadelo2005, Muro de Aguas, Mutari, Netito777, Ninovolador, Nixon, Nocturnogatuno, Obueno, Orgullomoore, Paintman, Pan con queso, Pedrito suarez, Pedrovicenterosero, Peregring-lk, PeruProfe, Pit, PoLuX124, Qix, Queninosta, Raystorm, Rbonvall, Rellu, Riviera, RoyFocker, RoyFokker, Rsg, Ríos-Ortega, S3v3r-1, Sabbut, Sancebau, Sauron, Schummy, Shining.Star, Shooke, Snakeyes, Sophie kowalsky, Speedplus, Super braulio, Superzerocool, Tano4595, Technopat, Tirithel, Tomatejc, Tostadora, Triku, Valentin estevanez navarro, Veon, Vic Fede, Virgi, Vitamine, Wilfredor, Willtron, XalD, Xavigivax, Xpress500, Xxim, Xxxmagicianxxx, Yeza, YoaR, Zam, ZrzlKing, Zupez zeta, conversion script, proxy1.unizar.es, 905 ediciones anónimas

Fuentes de imagen, Licencias y contribuyentes

Archivo:LampFlowchart-es.svg *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:LampFlowchart-es.svg> *Licencia:* Creative Commons Attribution-Sharealike 3.0 *Contribuyentes:* User:Booyabazooka, User:Huhsunqu

Archivo:AlgoritmoRaiz.png *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:AlgoritmoRaiz.png> *Licencia:* Creative Commons Attribution-Sharealike 2.5 *Contribuyentes:* Kn

Archivo:EsquemáticaAlgoritmo1.svg *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:EsquemáticaAlgoritmo1.svg> *Licencia:* GNU Free Documentation License *Contribuyentes:* User:Kn

Image:Wikibooks-logo.svg *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Wikibooks-logo.svg> *Licencia:* logo *Contribuyentes:* User:Bastique, User:Ramac

Archivo:Wiktionary-logo-es.png *Fuente:* <http://es.wikipedia.org/w/index.php?title=Archivo:Wiktionary-logo-es.png> *Licencia:* logo *Contribuyentes:* es:Usuario:Pybalo

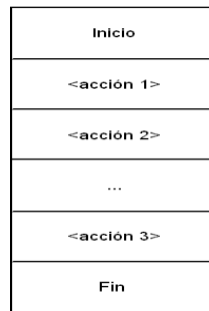
Licencia

Creative Commons Attribution-Share Alike 3.0 Unported
<http://creativecommons.org/licenses/by-sa/3.0/>

Diagramas Nassi-Schneiderman

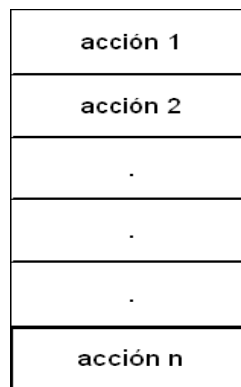
Los diagramas Nassi-Schneiderman son una técnica para la especificación de algoritmos que combina la descripción textual del pseudocódigo con la representación gráfica del diagrama de flujo.

Todo algoritmo se representa de la siguiente forma:

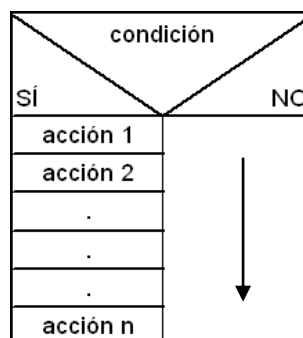


Existe una representación para cada una de las 3 instrucciones permitidas en la programación estructurada.

- **Secuenciales.** Recordemos que aquí tenemos: **declaración de variables** (tipo: nombre_variable), **asignación** (nombre_variable = valor), **lectura** (Leer <lista de variables>) y **escritura de datos** (Escribir <lista de constantes y variables>).



- **Alternativas.**
 - Alternativa simple.



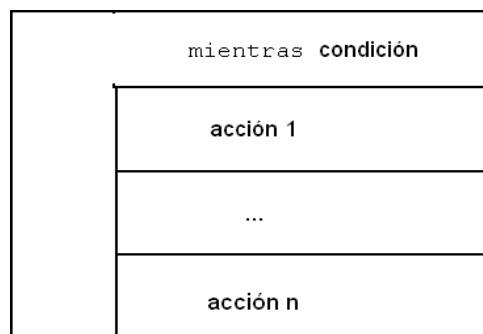
- Alternativa doble

condición	
SÍ	NO
acción 1	acción 1
acción 2	acción 2
.	.
.	.
.	.
acción n	acción n

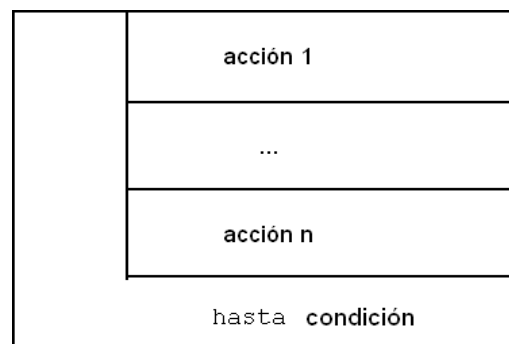
- Alternativa múltiple

condición			
caso 1	caso 2	caso 3	caso 4
acciones 1	acciones 2	acciones 3	acciones 4

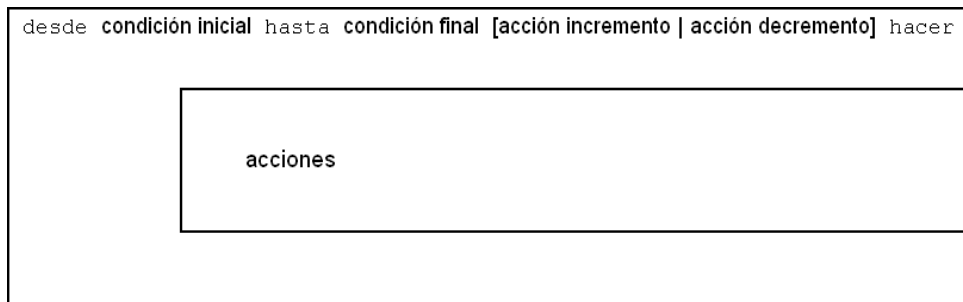
- Iterativas.
 - Ciclo Mientras



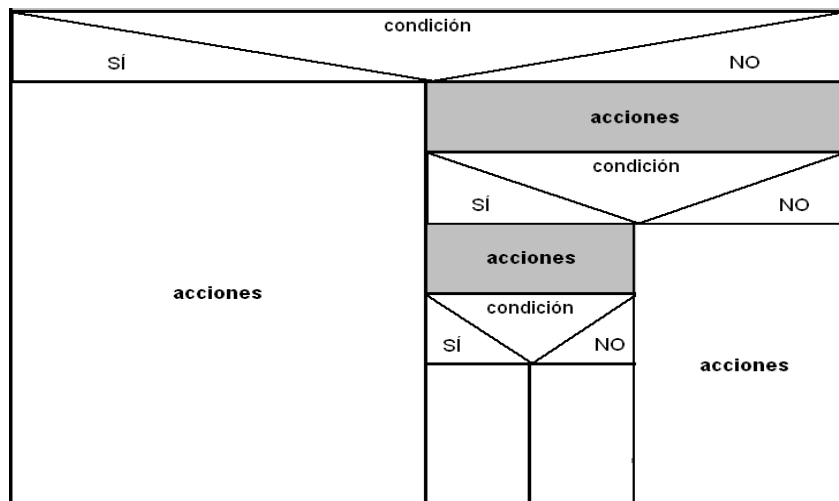
- Ciclo Repetir



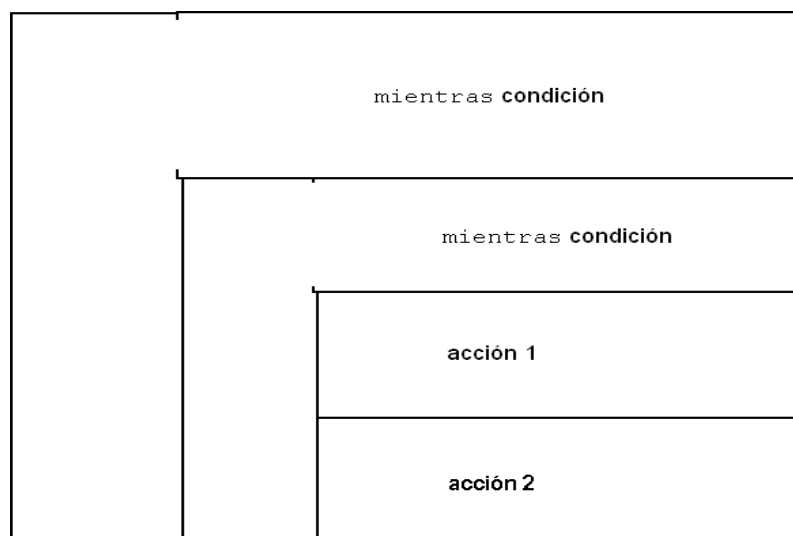
- Ciclo Desde / Para



- Alternativas anidadas. Consta de una serie de estructuras si, unas interiores a otras; a su vez, dentro de cada estructura pueden existir diferentes acciones. Se utiliza para diseñar estructuras que contengan más de dos alternativas.

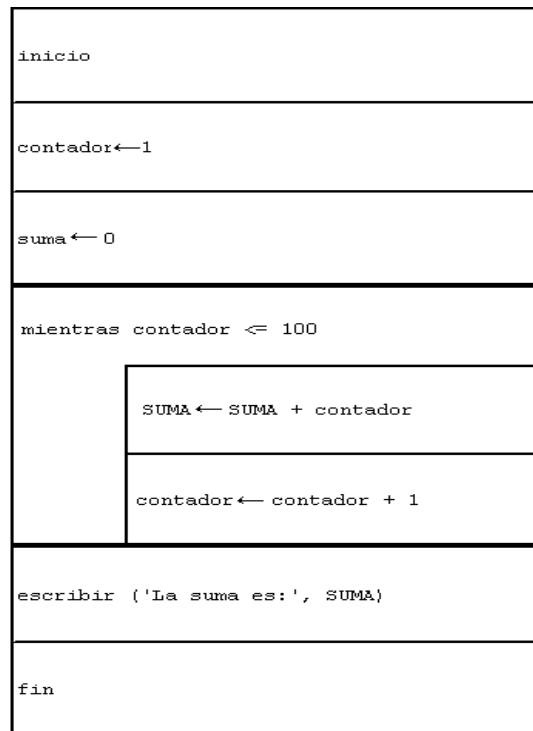


- Iterativas anidadas. Consta en anidar un ciclo dentro de otro. En este caso la estructura interna debe estar incluida totalmente dentro de la externa y no puede existir solapamiento.

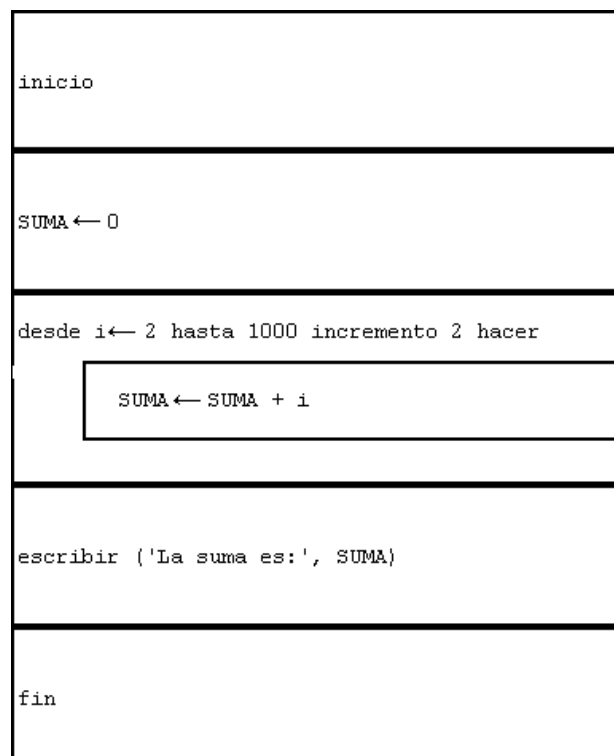


Ejemplos:

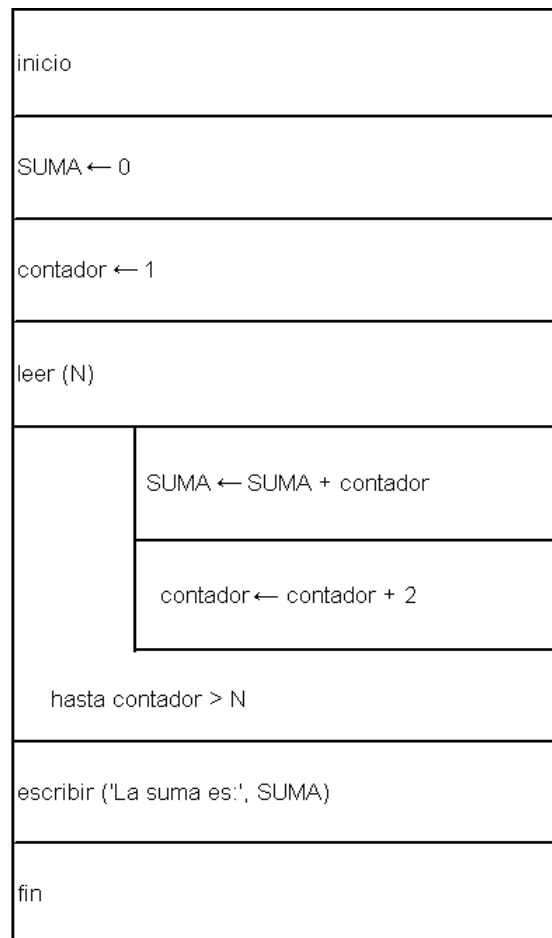
algoritmo suma_1_a_100



Elabora una solución, la más conveniente, para calcular la suma de todos los números pares entre 2 y 1000, utilizando la estructura desde.



Elabora una solución, la más conveniente, para calcular la suma de los números impares hasta N (inclusive), utilizando la estructura repetir.



Nomenclatura	Uso de Variables	Expresiones	Estructuras	Programación (varios)	Sangrado y Documentación
1. Si el nombre consta de varias palabras, escribirlas todas seguidas y separadas por mayúsculas	1. Variables de clase u objeto jamás deben ser públicas	1. Evitar expresiones numéricas o condiciones lógicas complejas. Dividir en subexpresiones	1. No usar “break” y “continue” salvo que sea estrictamente necesario	1. Definir clases y métodos pequeños	1. No escribir líneas con más de 70 caracteres aproximadamente. En otro caso, partir la línea
2. Nombres de los paquetes en minúsculas <code>package lprg.p7;</code>	2. Las variables deben declararse en su ámbito de uso (más reducido posible)	2. En una construcción condicional el “if” llevará la condición que se cumple más comúnmente	2. En un método que tiene que devolver un resultado dependiendo de varias condiciones, devolver el resultado en cuanto se sepa	2. Reemplazar estructuras o expresiones no triviales que se repiten en el código por un método equivalente	2. Utilizar dos espacios en blanco como medida básica de sangrado. No emplear el tabulador
3. Nombres para las Clases que sean sustantivos <code>ListaDeNombres</code>	3. Inicializar las variables inmediatamente <code>int nMuestras= 0</code>	3. Evitar comparar con TRUE y FALSE para tomar decisiones. Mejor: <code>if (! condicion)</code>	3. Evitar el anidamiento excesivo de estructuras “for”, “while”, “if”, “switch”. Si hay más de tres de estas estructuras anidadas, introducir métodos auxiliares para reducir	3. Aclarar el orden de las operaciones con paréntesis	3. En estructuras de bloque con una sola sentencia, se pueden obviar las llaves <code>if (condicion) hazAlgo();</code>
4. El nombre de las variables empieza por minúscula <code>int numeroDeCarneros;</code>	4. Variables con un ámbito muy amplio, que tengan nombre largo y significativo			4. Para comprobar si dos objetos son iguales usar el método “equals()” y no el operador “==”. Lo último es para comparar referencias.	
5. Constantes (“final”) íntegramente en mayúsculas <code>static final int COLOR_ROJO= 0xFF0000;</code>	5. Variables con un ámbito muy restringido o de usar y tirar que tengan nombre corto <code>int i, j;</code>		4. Un bucle no debería ocupar más de una pantalla (unas 20 líneas de código). En otro caso crear métodos auxiliares para reducir	5. Las excepciones en tiempo de ejecución no deben silenciarse	4. Ver los distintos patrones a seguir para sangrar: clases, interfaces, métodos, bucles, estructuras if y switch, captura de excepciones, etc
6. Nombres Métodos que sean verbos y primera letra en minúscula <code>String getNombre();</code> <code>void setNombre(String string);</code>	6. Las variables de iteración también deben tener nombres muy cortos <code>for (int i= 0; i < 10; i++) { ... }</code>				5. Hay que añadir explicaciones a todo lo que no es evidente 6. No hay que repetir lo que se hace, sino explicar por qué se hace Obligatorio (javadoc): - al principio de cada clase - al principio de cada método - ante cada variable de clase
7. Si la variable es un acrónico, sólo primera letra en mayúscula <code>class Dni;</code>	7. No usar nunca una variable para dos cosas distintas <code>int numero= 0; numero= numMesas(); numero= numHojas();</code>				Conveniencia (1 línea): - al principio de fragmento de código no evidente - a lo largo de los bucles

A los efectos de construir buenos programas y permitir comprender las ideas que el o los programadores desean expresar o comunicar y también facilitar la lectura, sugerimos esta pequeña guía que ha demostrado ser útil en el desarrollo de proyectos en varios ciclos lectivos.

Esta guía se incluye para que los alumnos minimicen la pérdida de tiempo en las tareas grupales y faciliten la evaluación del trabajo por los docentes y permitan realizar correcciones, depuraciones o mantenimientos que generalmente presentan los trabajos prácticos realizados.

Generalmente, las tareas de corrección y depuración son realizadas por otras personas con criterios distintos y modalidades de trabajo también distintas, entonces, para minimizar éstos problemas y sin quitar importancia a la labor creativa necesaria en el diseño de cualquier software, sugerimos una serie de recomendaciones generales, que necesariamente son subjetivas y por tanto discutibles en su validez, pero que han demostrado ser útiles durante la experiencia acumulada.

Estas recomendaciones, fundamentalmente van dirigidas a alumnos diseñadores de software sin experiencia en proyectos con cierto grado de dificultades y magnitud mediana, por lo que pretendemos crearles un buen hábito desde el inicio y no descuidar los detalles que pudieran ser causa de ineficiencias futuras.

Los aspectos a tener en cuenta son cuatro, a saber, organizativas, de nomenclatura, de identificadores, técnicas y de presentación.

ASPECTO ORGANIZATIVO

Este aspecto es importante a los efectos de que el grupo alumnos aúnen esfuerzos en el diseño y se constituya un real equipo de trabajo con roles definidos y objetivos comprendidos por todos.

Las fases son:

- a) Reunión previa: de constitución del grupo de trabajo y toma de conocimiento de la situación de cada integrante y donde se establecen lugares, horarios de reuniones, restricciones, los medios de comunicación, etc.. Logrado esto puede continuarse con la siguiente tarea.*
- b) Comprensión del problema: en el que se analiza globalmente el problema a ser resuelto, generalmente bosquejado en un conjunto de especificaciones o deseos que se entregan inicialmente a los integrantes del grupo.*
- c) Fase de análisis del problema: en la que se estudia conjuntamente el planteo inicial, los sucesivos objetivos, los eventos que se encadenan, la inicialización, la secuencialidad, el paralelismo, el uso y la desactivación o destrucción de objetos, etc.*

En esta fase se deben resolver los siguientes puntos:

1. *Eliminar ambigüedades e incoherencias*
2. *Controlar que las especificaciones iniciales del problema contengan toda la información pertinente sin lagunas y si las hubieran determinar como se las cubriría y quien será el responsable de resolverla.*
3. *Utilizar una estructura (algoritmo) conocida y sencilla con una nomenclatura (codificación) clara y precisa como la que se explica mas adelante (método húngaro).*

NOMENCLATURA DE IDENTIFICADORES

Una buena codificación de identificadores, permite verificar y leer adecuadamente un programa, para ello proponemos un esquema que permite la generación de identificadores basado en la notación húngara (Anexo I) en la que la codificación es realizada mediante prefijos y sufijos. Existen programas utilitarios (ejemplo CodeCheck con la regla prefix.cc.) que permiten analizar la codificación de acuerdo a esta nomenclatura.

La estructura del nombre puede contener hasta de 8 letras en la que las primeras tres indican rótulos (tags) que definen tipo y subtipo de acuerdo a la siguiente regla:

id -> tipo[subtipo]nombre

Las siguientes tablas definen tags para tipos y sub-tipos más comúnmente utilizados. Estos tags están definidos en base a abreviaciones o acrónimos del nombre de cada tipo de dato y los nombres de más de una palabra deben separarse usando letras Mayúsculas.

TIPO	VARIABLES DE TIPO
<i>c</i>	<i>char</i>
<i>u</i>	<i>unsigned</i>
<i>l</i>	<i>long</i>
<i>s</i>	<i>short</i>
<i>i</i>	<i>int</i>
<i>f</i>	<i>float</i>
<i>d</i>	<i>double</i>
<i>st</i>	<i>struct</i>
<i>en</i>	<i>enum</i>
<i>b</i>	<i>boolean</i>
<i>n</i>	<i>variable de tipo int usada como indice dentro de un vector</i>

<i>SUBTIPO</i>	<i>define</i>
<i>np</i>	<i>near pointer</i>
<i>fp</i>	<i>far pointer</i>
<i>p</i>	<i>pointer</i>
<i>t</i>	<i>nobre de clase o tipo definido por el usuario</i>
<i>v</i>	<i>vector</i>
<i>m</i>	<i>matriz</i>
<i>h</i>	<i>handle o puntero doble</i>
<i>r</i>	<i>argumento de una funcion</i>

Ejemplo:

```
char      *cpStr;
unsigned  uvTabla[100];
void      f(int irNumero);
```

Calificadores

Calificadores distinguen cualidades con tipos idénticos en un contexto nominal.

En otras palabras, son nombres que vienen de un pequeño conjunto de nombres estándares que tienen una semántica bien definida (tal como las llamadas constantes).

Por ejemplo, en el nombre de variable cpLim, el tipo es cp y el calificador es Lim.

Criterio para elegir calificadores

- *Para variables booleanas (b): describir la condición bajo la cual la variable es verdadera (ej: bOpen, bExiste).*
- *Para variables en conjuntos enumerados: Describir el elemento particular. Considerar, por ejemplo, un tipo de valor de color con el prefijo co. Los elementos constantes de este tipo pueden llamarse coRojo, coVerde y así sucesivamente.*

Calificadores standard

- *Temp (o T) Temporario. Típicamente usado para anidar variables.*
- *Sav. Un temporario del cual se restaurará la variable.*
- *Prev. Un valor para grabar el contenido previo de la variable en la última iteración (la actual corriente)*
- *Cur. valor actual en una enumeración.*
- *Next. próximo valor en una enumeración.*
- *Dest, Src. destinación y fuente u origen en una relación consumidor/Productor.*
- *Nil. un valor especial inválido que puede ser distinguido de un grupo de valores válidos y que típicamente denota alguna ausencia.*
- *Min. mínimo índice válido. Típicamente definido como 0.*

- *Mac.*(máximo corriente) un valor superior estricto para índices válidos, es el tope de algún stack (pila). También indica el número de elementos de un arreglo (array) cuando $Min = 0$.
- *Max.* es el límite de almacenamiento de algún stack. $Max > \text{ó} = Mac$.
- *First.* primer elemento de un intervalo [*First*, *Last*].
- *Last.* último elemento de un intervalo.
- *Lim.* Límite superior estricto de un intervalo [*First*, *Lim*). $Lim - First$ es el número de elementos en un intervalo.

Para una mayor profundización sobre este tema sugerimos al alumno leer el artículo "The Hungarian Revolution" aparecido en la revista Byte de Agosto del 1991, (páginas 131 a 138) que figura como Anexo I de la presente introducción.

ASPECTO TÉCNICO

En cuanto al aspecto técnico, el software se hará **SIEMPRE REUBICABLE Y ESTRUCTURADO**. Estos dos requisitos facilitan la depuración, la comprensión y el mantenimiento de los programas.

PRESENTACIÓN

Recomendamos observar los siguientes puntos para la presentación de los programas:

1. Cada módulo, rutina, función, etc, deberá estar presentado de la siguiente forma:

- Encabezamiento: (Entre símbolos de comentarios)
 - a) Nombre de la función, rutina, módulo, etc.
 - b) Objetivo de la misma
 - c) Parámetros de entrada
 - d) Parámetros que devuelve
 - e) Apellidos y nombres del o los autores
 - f) Fecha de creación
 - g) Fecha de última modificación o reléase (versión)
 - h) Escudo comentario escrito sobre la modificación
- Comentarios documentados:
En cada línea de codificación en que se realiza una operación compleja, se deberá incorporar un comentario a modo de documentar en pocas palabras lo que se pretende realizar con el código en ese renglón. Esto facilita enormemente la lectura y comprensión del código.

2. Los programas deberán estar estructurados en secciones o módulos bien separados en forma que sea totalmente modular y que facilite las tareas de depuración cuando se

modifique una función o parte de ella y no requiere retocar todo el programa en su conjunto.

3. El lenguaje que se utiliza para la codificación tiene predefinido un conjunto de palabras llamadas palabras claves o palabras reservadas (que son las primitivas de dicho lenguaje) dado que solo pueden ser usadas en una determinada y particular forma. Para construir los módulos se requiere cumplir cuatro pasos fundamentales, a saber:

- *primer paso: desarrollo de las especificaciones*
 - a) Estudiar el problema a resolver*
 - b) definir un nombre de la función, programa, etc.*
 - c) definir las interfases de entrada y salida que usará este módulo (variables globales)*
 - d) definir las variables locales*
- *segundo paso: desarrollo del programa*
 - a) programar el algoritmo que realizará el módulo*
- *tercer paso: prueba del programa*
 - a) funcionamiento*
 - b) depuración*
 - c) optimización*
- *cuarto paso: catalogación (no siempre se implementa)*
 - consiste en incorporar el módulo en una estructura de forma tal que pueda ser fácilmente utilizable en otros códigos a través de un editor de enlaces (linkeditor).*

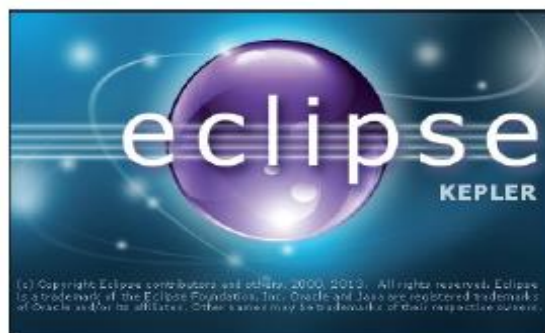
4. En cuanto a los programas completos deberán considerarse

- 1. su instalación por única vez*
- 2. su inicialización todas las veces que se lo invoca*
- 3. su desinvocación*

5. Y desde el punto de vista del usuario se deberá considerar todas las interfases que permitan un fácil uso e interacción con las presentaciones (teclas, mouse, pantallas sensibles, etc.)

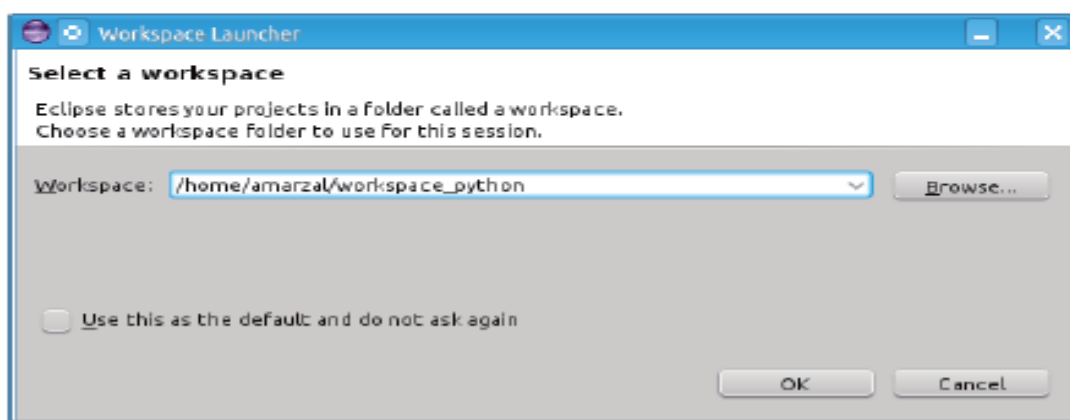
3.1.1. Instalar y preparar Eclipse para el trabajo con la extensión Pydev

Arranca Eclipse. Aparece un pantallazo de presentación similar al siguiente (que es el de la versión 4.3 de Eclipse, conocida como Kepler):

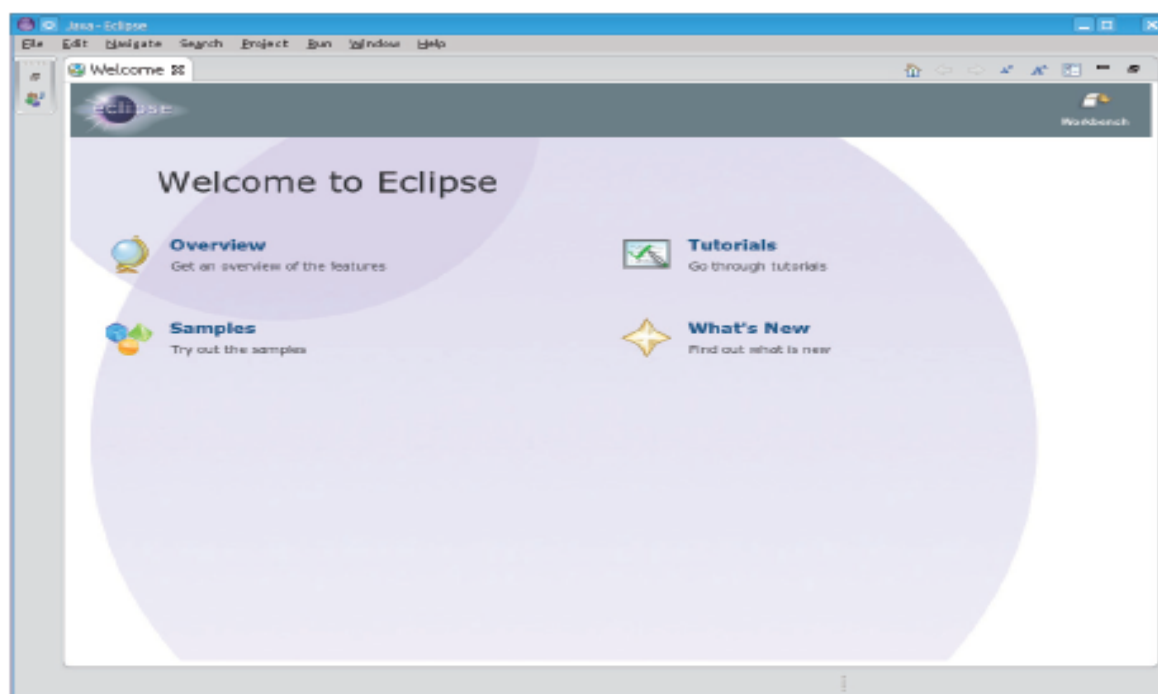


¹También se suele denominar *scripts* a los programas Python.

A continuación aparecerá un cuadro de diálogo en el que se te pedirá que indiques el espacio de trabajo («workspace») en el que vas a moverte en esta sesión. El espacio de trabajo es una carpeta (un directorio) en el que puedes agrupar proyectos relacionados entre sí. Es recomendable que uses un único espacio de trabajo para el trabajo con este texto. Cada programa con suficiente entidad, cuando los haya, podrá crearse en su propio proyecto dentro del espacio de trabajo. Como es la primera vez que trabajamos con Eclipse, vamos a crear un espacio de trabajo propio al que denominaremos `workspace_python`. Modifica el nombre que te propone por defecto para que quede así² y pulsa el botón OK:

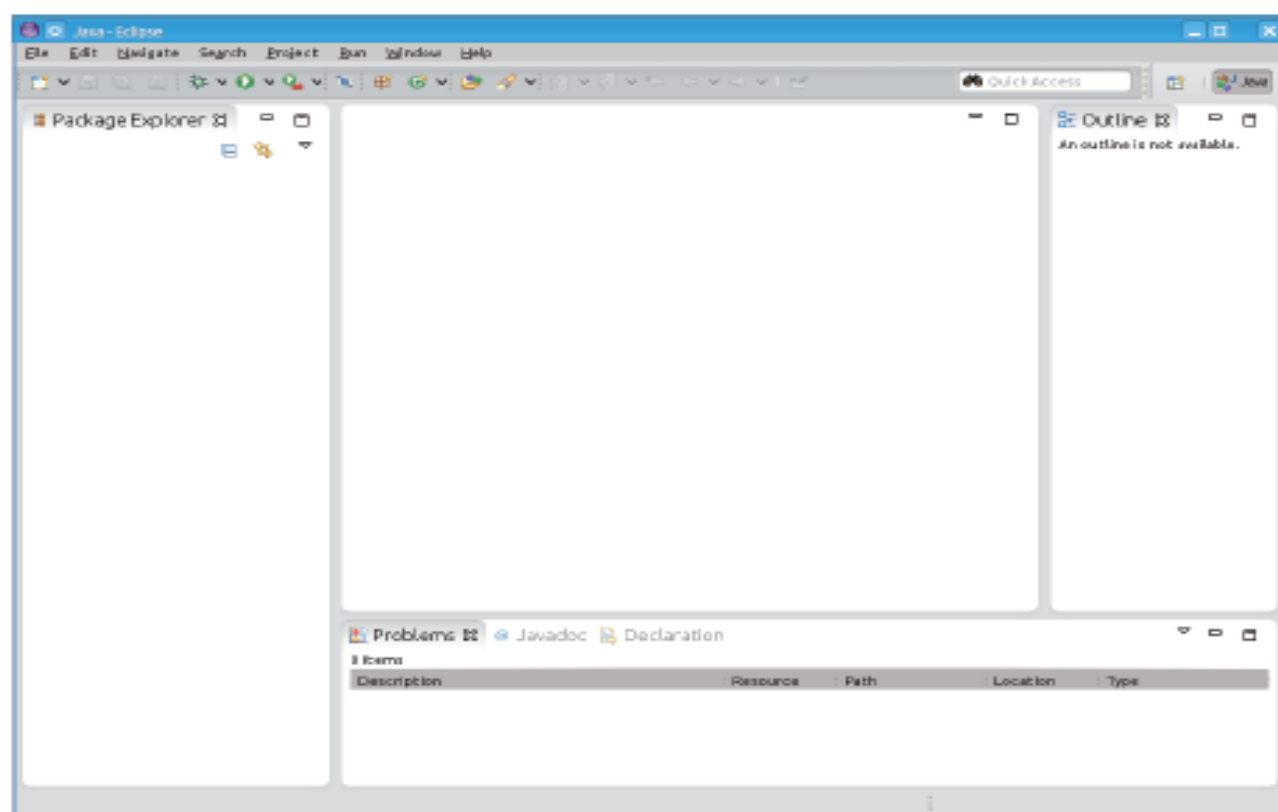


Probablemente Eclipse arranque con una pantalla como esta:

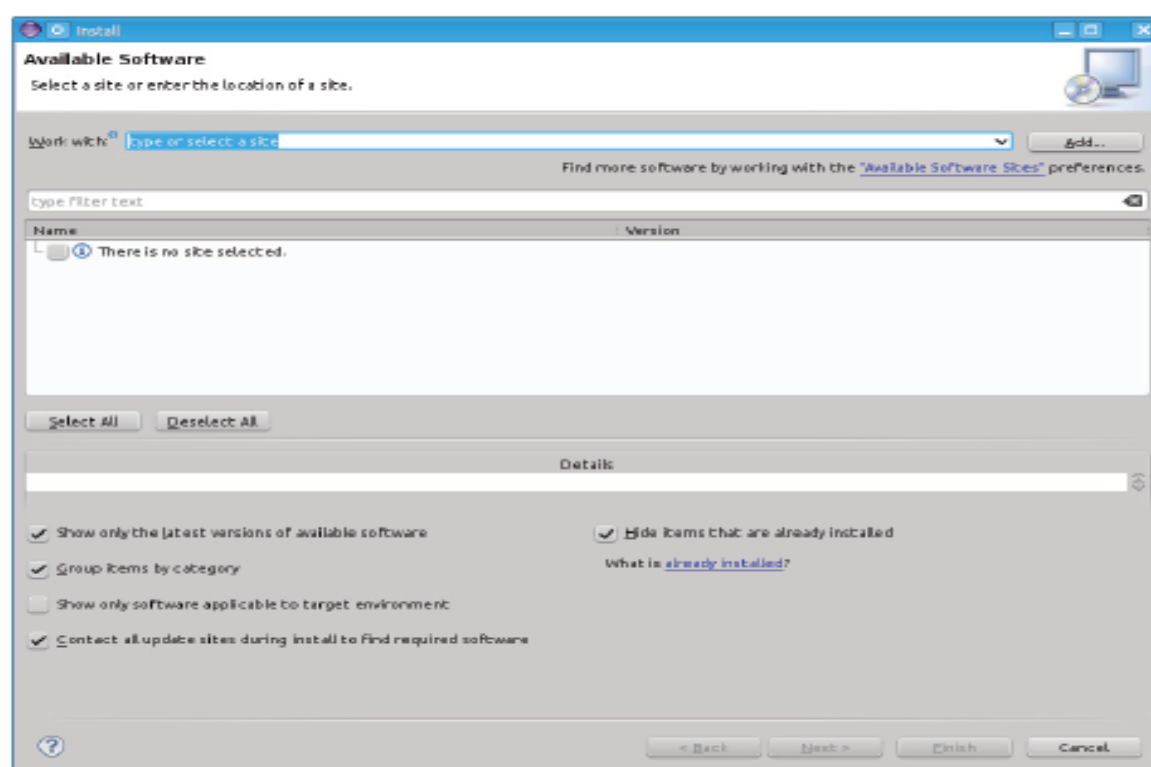


Es una pantalla de bienvenida que conduce a una serie de manuales y tutoriales. Como estos están principalmente orientados al trabajo con Java, no nos resultan de mucha utilidad ahora mismo. Lo mejor es que cerremos la pestaña titulada «Welcome» pulsando en el aspa que hay a su derecha. Encontrarás entonces un entorno de trabajo como este:

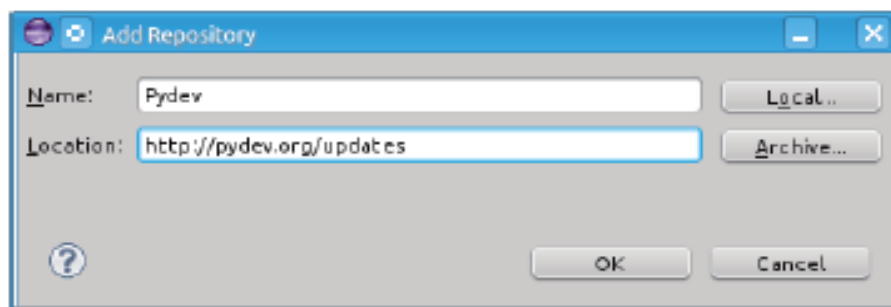
²La ruta del espacio de trabajo será diferente en función del sistema operativo con el que estés trabajando. Las imágenes han sido capturadas en un ordenador con sistema operativo Linux.



Hemos de instalar ahora el módulo Pydev, que adapta Eclipse al trabajo con Python. Necesitarás una conexión a Internet y haber instalado previamente Python 3.1 (o superior). Ve al menú *Help*→*Install New Software*. Aparecerá un cuadro de diálogo como este:

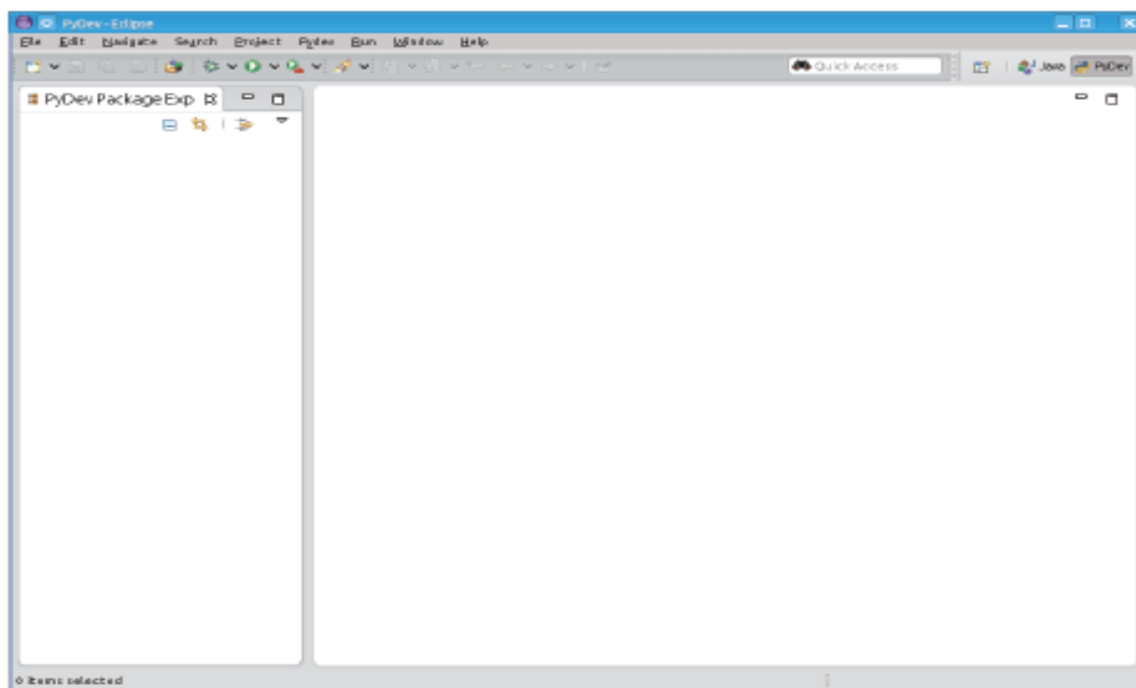


Pulsa el botón «Add». Aparecerá un nuevo cuadro de diálogo. En el campo «Name» escribe el texto *Pydev* y en el campo «Location» escribe la dirección <http://pydev.org/updates>:



Pulsa ahora el botón «Ok». En la nueva pantalla que aparece, marca la casilla «PyDev» y avanza dos pantallas más pulsando «Next». A continuación, acepta las condiciones que impone una licencia de uso, pulsa «Finish» e indica que confías en el certificado del sitio de instalación para que Eclipse proceda a instalar el componente Pydev. Por último, será necesario reiniciar Eclipse para completar el proceso de instalación.

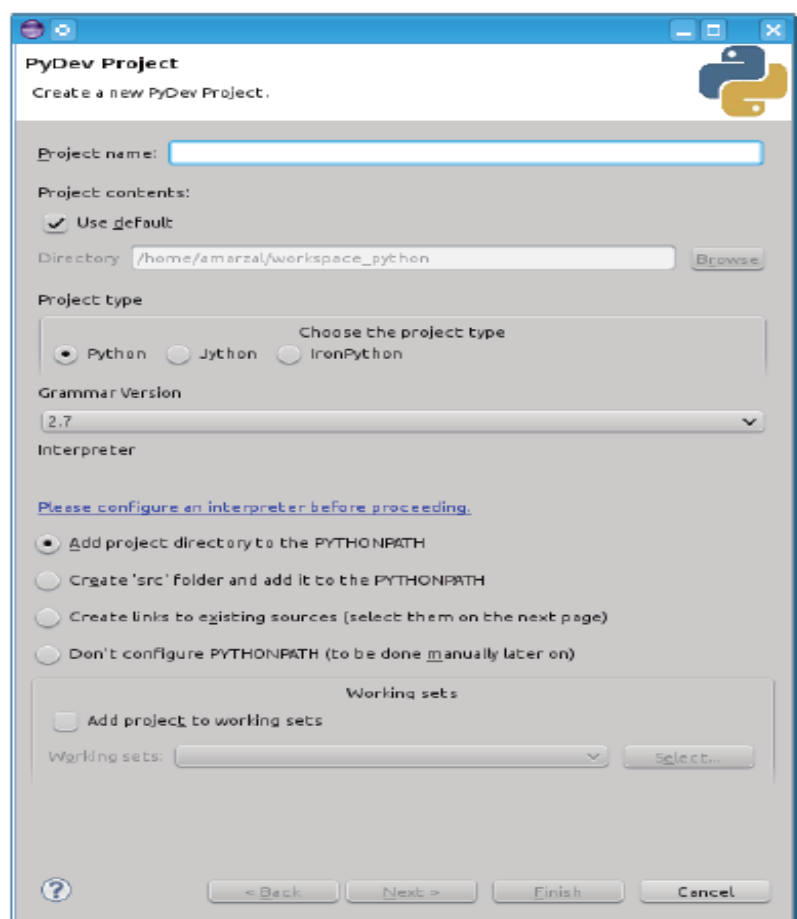
Solo nos queda un paso antes de crear un proyecto Pydev: seleccionar la perspectiva Pydev. Una perspectiva Eclipse es un conjunto de paneles dispuestos de una determinada forma que facilitan el desarrollo con un determinado lenguaje, tipo de ficheros o plataforma. Para elegir la perspectiva Pydev ve al menú *Window→Open Perspective→Other*, selecciona *Pydev* en el cuadro de diálogo que se habrá abierto y pulsa «OK». La apariencia de la ventana principal cambiará un poco:



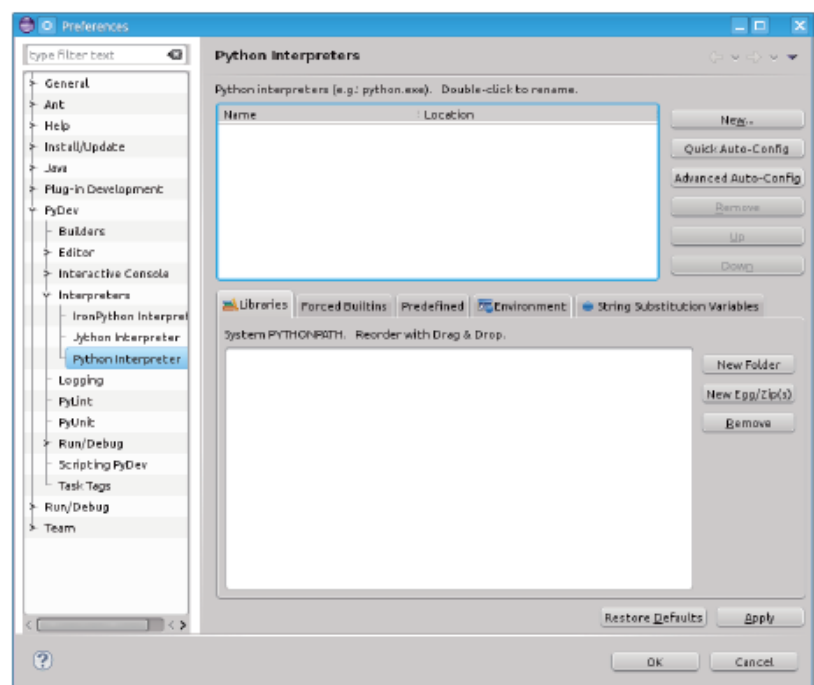
La ventana principal presenta dos regiones o vistas³ (del inglés «views»). En la izquierda encontrarás el «Pydev Package Explorer» y en él aparecerán los diferentes proyectos que crees y, dentro de cada proyecto, las carpetas y ficheros que lo formen. A su derecha hay un espacio para albergar los editores de texto con los que crearás y modificarás los programas.

Empecemos creando un proyecto al que llamaremos primeros_programas. Selecciona la opción de menú *File→New→Pydev Project*. Aparecerá un cuadro de diálogo como este:

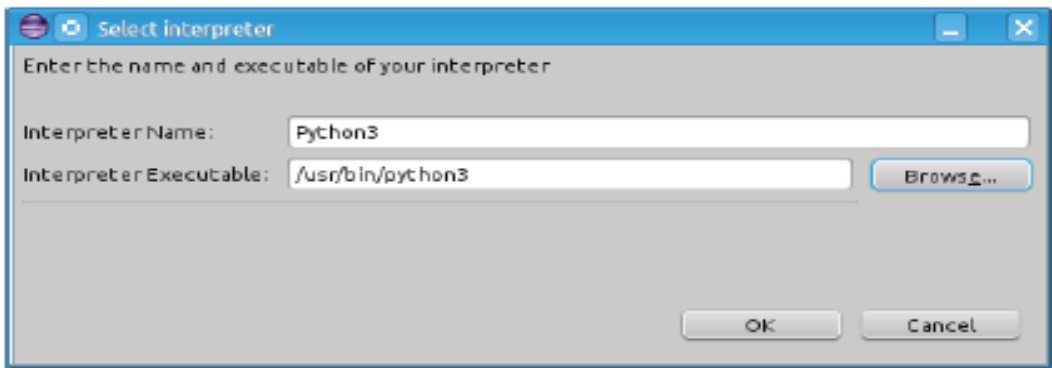
³Hay más vistas, que puedes activar mediante *Window→Show View*. Por ejemplo, la vista «Outline» permite mostrar una visión esquemática del programa cuyo editor tenga el foco y te resultará muy útil para navegar rápidamente por tus programas cuando estos tengan decenas (o centenares) de líneas de código.



El nombre del proyecto (campo «Project name») será `primeros_programas`. Dejaremos marcada la opción «Use default» para que Pydev cree la estructura del proyecto. El tipo de proyecto es «Python» y la versión de la gramática es 3.0 (aunque usemos Python 3.1). Y como este es nuestro primer proyecto, hemos de hacer un trabajo extra: dar de alta el intérprete Python que usaremos para ejecutar nuestro programa en el entorno. Para ello pinchamos en el enlace «Please configure an interpreter before proceeding» y seleccionamos «Manual Config». Esto nos llevará a un nuevo cuadro de diálogo:



Pulsa en el botón «New» que hay arriba a la derecha. Aparecerá un nuevo cuadro de diálogo titulado «Select interpreter». El primer campo debe contener «Python3»⁴. En un sistema Unix, como Linux, el segundo campo deberá contener una ruta similar a `/usr/bin/python3`. Si estás trabajando con Microsoft Windows, el segundo campo contendrá `C:\Python31\python.exe`⁵:

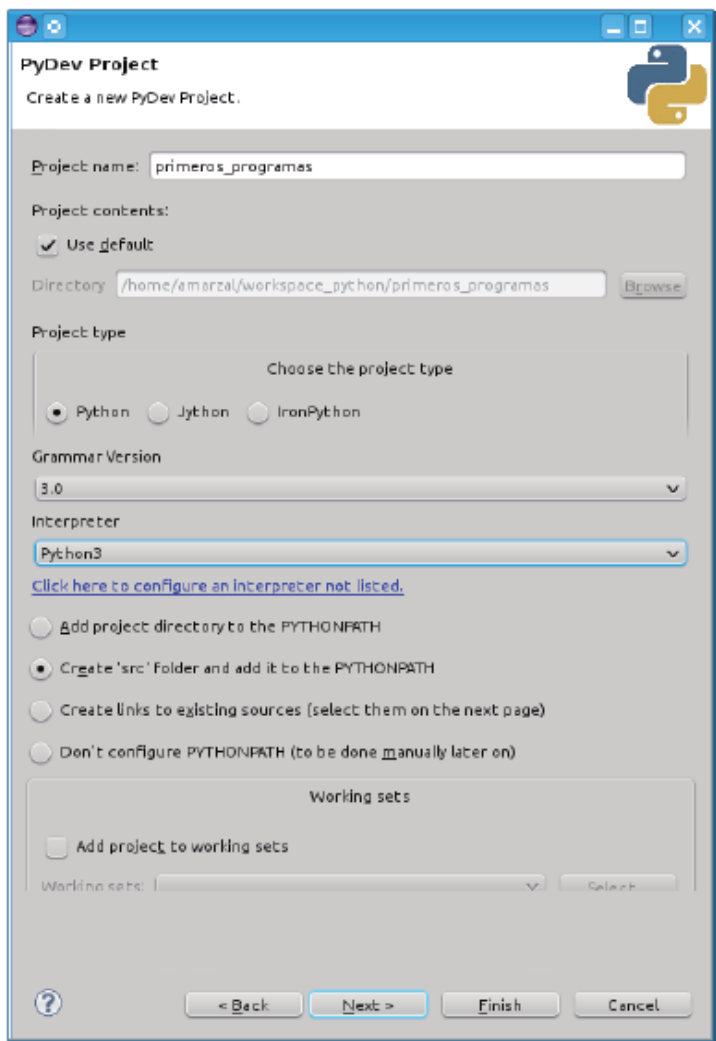


Seguimos. Pulsamos «OK» y aparece un nuevo cuadro de diálogo con una relación de rutas. Pulsamos nuevamente «OK». Regresamos así al cuadro de diálogo «Preferences» y en él pulsamos ahora «OK». Pydev se tomará su tiempo para configurar el sistema. (Recuerda que esto tan trabajoso solo lo hacemos una vez por espacio de trabajo, así que la pérdida de tiempo no se repetirá mucho). En el cuadro de diálogo aparece, bajo el desplegable para seleccionar la gramática, un nuevo desplegable para seleccionar un intérprete. Podemos dejarlo tal cual está, con la opción «Default», o seleccionar la opción «Python3» (pues en este momento solo hemos dado de alta un intérprete y es lo mismo seleccionar «Default» que «Python3», es decir, la etiqueta con la que hemos nombrado dicho intérprete). Es habitual que almacenemos nuestros

⁴Lo cierto es que da igual el texto del primer campo: no es más que una etiqueta. Eso sí: usemos un nombre sencillo que deje claro que usamos un intérprete Python de la versión 3.

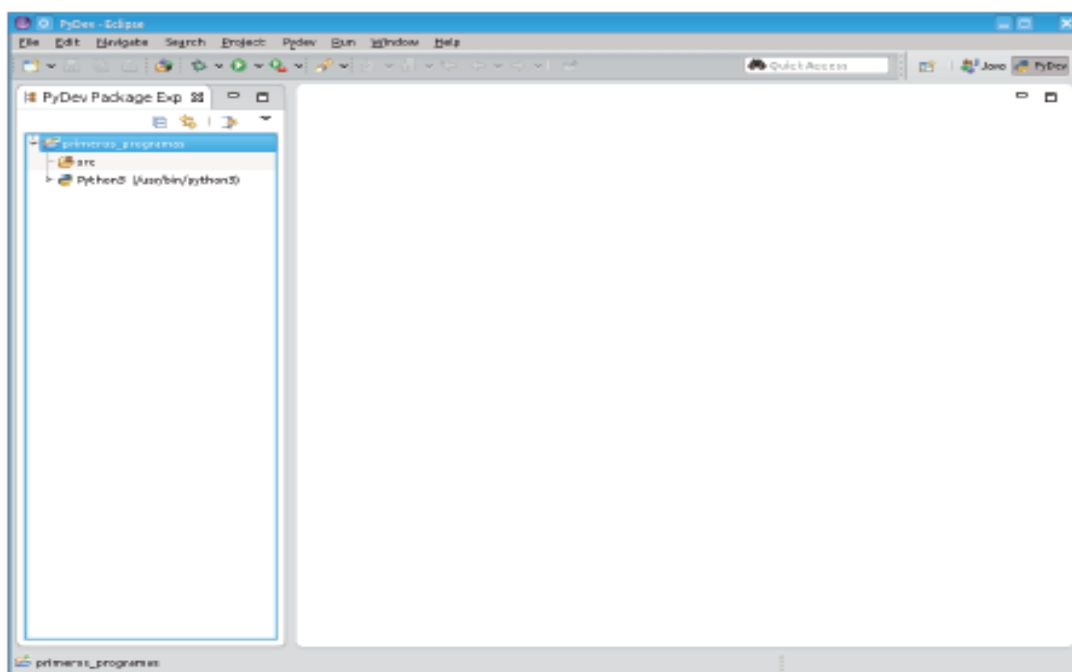
⁵Tanto en Unix como en Windows puede haber alguna diferencia con la ruta del intérprete. Todo depende del modo en que hiciste la instalación del paquete Python 3. Las rutas que indicamos corresponden a instalaciones estándar de Python 3.1.

programas en una carpeta llamada «src» (del inglés «source»). Para ello, marcaremos la opción «Create 'src' folder»:

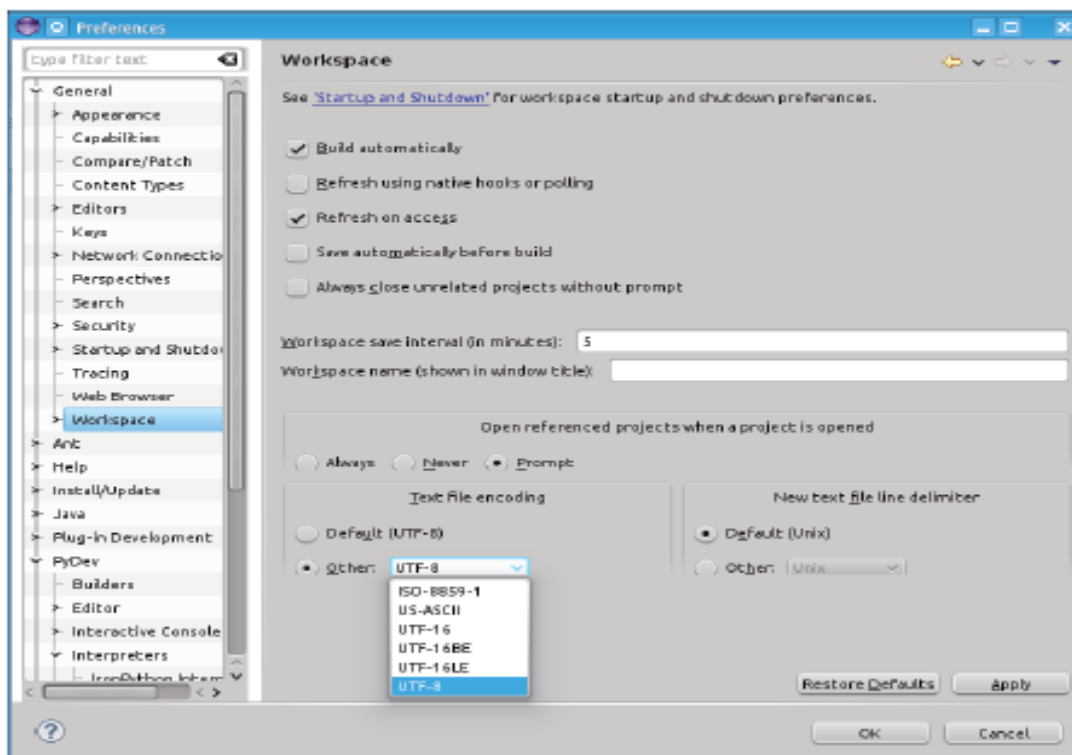


Pulsamos «Finish» y ya estamos listos para trabajar.

Fíjate en que en la vista del explorador de paquetes Python aparece una carpeta con el nombre `primeros_programas`. Si acercamos el ratón a ella comprobaremos que se trata de una carpeta desplegable. Al pulsar en el botón triangular que hay a su izquierda, se desplegará la carpeta y se mostrará su contenido:



Nos queda aún un pequeño detalle para tenerlo todo listo. Nos gustaría fijar una codificación de texto para todos los ficheros que más tarde no nos dé problemas con la representación de los caracteres acentuados o de la letra ñe. Podemos hacer esto fichero a fichero, pero es mucho mejor fijarlo en las preferencias del propio espacio de trabajo. Has de ir al menú *Window*→*Preferences*. Aparecerá un cuadro de diálogo complejo. Selecciona la opción *General*→*Workspace* en el árbol de menús que hay a mano izquierda. En el panel de la derecha aparecerá un cuadro de diálogo con un elemento titulado «Text file encoding». Ese elemento contiene dos botones de radio con las opciones «Default» y «Other». Si es necesario, selecciona la opción «Other» y, en el menú desplegable que se activa entonces, selecciona «UTF-8». Cierra finalmente el cuadro pulsando el botón «OK»:

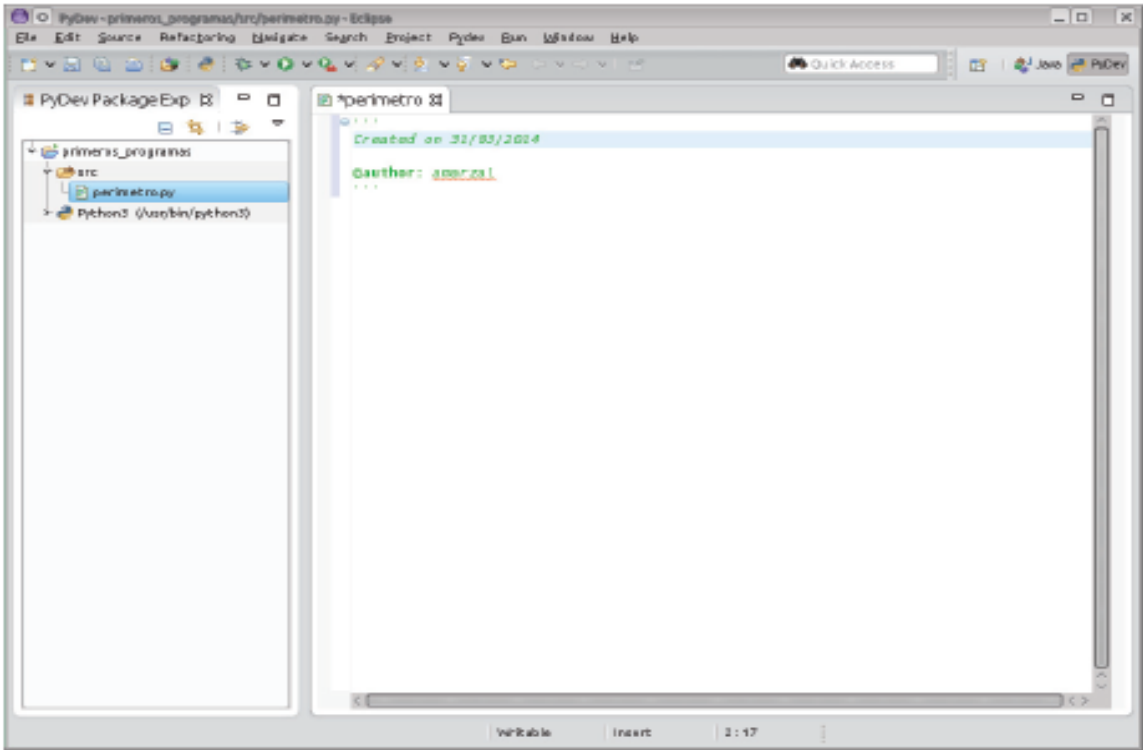


Con eso hemos seleccionado el juego de caracteres Unicode y nuestros ficheros podrán leerse igual en cualquier sistema moderno. Ya lo tenemos todo preparado para empezar a escribir nuestro primer programa.

3.1.2. Nuestro primer programa

Empezaremos por crear el fichero en el que escribiremos el programa. Pulsa el botón derecho en la carpeta «src» y en el menú contextual selecciona *New→Pydev module*. En el cuadro de diálogo no edites el primer campo, que dice `/primeros_programas/src`, ni el segundo, que está en blanco. En el tercero escribe `perimetro`⁶ y pulsa «OK». En el nuevo cuadro de diálogo que aparece, deja la selección de «Template» en la opción «Empty», tal cual está. (Seguramente estás ya abrumado por lo trabajoso que es configurar el entorno para crear un proyecto y trabajar en él. Tranquilo. Acabará siendo un proceso mecánico).

Acabamos de crear el fichero `perimetro.py` y se ha abierto un editor de texto para que podamos modificar su contenido. El fichero no está en blanco: contiene un texto que proviene de una plantilla. El texto contiene la fecha de creación y el nombre del autor:



Empezaremos por eliminar ese texto (ya veremos más adelante para qué podría servir) y escribir en su lugar este otro:

Expresión de Problemas y Algoritmos

El siguiente apunte se encuentra conformado por una recopilación de documentos de lectura sencilla, que permiten acercar a los alumnos a la correcta interpretación de los algoritmos básicos, con el fin de iniciarse en la programación de computadoras, de forma independiente al lenguaje a utilizar.

Se han respetado las fuentes en los documentos que se detallan, siempre que contengan la autoría del mismo. En los ejercicios se han utilizado variadas fuentes que han permitido nutrir a la guía de ejercicios de forma variada.

Anhelamos que ante la resolución de los ejercicios y con la explicación en clase logren un buen tránsito por la cátedra.

Expresión de Problemas y Algoritmos

Ejercicio 1

Ingresar por el teclado dos valores (números enteros) **A** y **B** ($B \neq 0$), efectuar las **4 operaciones básicas** y mostrar los resultados por la pantalla de las cuatro operaciones.

Ejercicio 2

Diseñar un algoritmo que lea e imprima una serie de números enteros distintos de cero y calcule la suma de los mismos. El algoritmo termina con el ingreso de 0, y debe mostrar la suma de los mismos.

Ejercicio 3

Diseñar un algoritmo que imprima y sume la serie de números 3,6,9,12...,99.

Ejercicio 4

Escribir un algoritmo que lea (se entiende que deben de ser ingresados) cuatro números y a continuación imprima el mayor de los cuatro.

Ejercicio 5

Diseñar un algoritmo que lea (se entiende que deben de ser ingresados) tres números y encuentre si uno de ellos es la suma de los otros dos (que los muestre al final mediante una leyenda).

Ejercicio 6

Escribir un algoritmo que calcule la superficie de un triángulo en función de la base y altura (se entiende que deben de ser ingresados). Mostrar su resultado.

Ejercicio 7

Calcular la longitud de la circunferencia y la superficie de un círculo, ingresando un número entero, que será el radio. Mostrar los resultados.

Ejercicio 8

Escribir un algoritmo que determine si el año es bisiesto o no. Se debe de tener en cuenta que, para saberlo, un año es bisiesto si es múltiplo entero de 4 (por ejemplo, 2020 lo fue y es un resultado entero, el 2021 no es un resultado entero).

Ingresar el año y mostrar la leyenda apropiada

Ejercicio 9

Diseñar un algoritmo que calcule el salario mensual de una persona, teniendo como dato el valor hora, la cantidad de horas semanales (tener presente que en el mes tenemos 4 semanas).

A ese valor, se le denomina, sueldo bruto. Al sueldo bruto, se le debe descontar el 11% por jubilación, el 3% por obra social y el 3% por contribución al sistema de salud de jubilaciones, éste nuevo valor es el sueldo neto, lo cual se debe mostrar al final de la operación.

Expresión de Problemas y Algoritmos

Ejercicio 10

Una empresa de software, ha decidido un aumento en sus salarios, en base a la siguiente escala:

Sueldo Bruto	Porcentaje de Aumento
0 – 150.000	20
150.001- 200.000	10
200.001- 250.000	5
250.001- 300.000	0

Diseñar un algoritmo que calcule el salario mensual de una persona, teniendo como dato el valor hora y la cantidad de horas semanales (tener presente que en el mes tenemos 4 semanas).

A ese valor, se le denomina, sueldo bruto. Al mismo aplicarle el porcentaje de aumento correspondiente (según la tabla anterior).

Al sueldo bruto, se le debe descontar el 11% por jubilación, el 3% por obra social y el 3% por contribución al sistema de salud de jubilaciones, éste nuevo valor es el sueldo neto, lo cual se debe mostrar al final de la operación.

Ejercicio 11

Diseñar un algoritmo, en el cual se ingresa la temperatura en grados Celsius, y un código (1 ó 2), si el código es 1, lo convierte en grados Fahrenheit [$F = (9/5) * C + 32$], en caso de ser el código 2, lo convierte en grados Kelvin [$K = C + 273$].

Se debe mostrar el resultado seleccionado.

Ejercicio 12

En una fábrica hay 4000 empleados distribuidos en 5 secciones, se quiere determinar cuántos empleados hay en cada sección.

El dato de entrada es el nombre de la persona y la sección.

Se debe mostrar como salida “Empleados sección 1:”. Así todas las secciones.

Ejercicio 13

Generar los números naturales del 1 al 200 e imprimir en forma secuencial aquellos mayores a 120.

Ejercicio 14

Ingresar un número entero y determinar si es positivo, negativo o cero, mostrar una leyenda en cada caso.

Ejercicio 15

Ingresar dos valores en distintas variables y determinar cuál de ellos es el mayor, colocar una leyenda que indique cuál es el mayor y qué valor tiene.

Expresión de Problemas y Algoritmos

Ejercicio 16

Ingresar un número entero y determinar si el mismo es menor 10, si está comprendido entre 10 y 100 o si el valor ingresado es mayor que 100.

En los tres casos se debe mostrar una leyenda y el valor ingresado.

Ejercicio 17

Se ingresan 100 números enteros no nulos (es decir que no llega el 0) y obtener el promedio de los positivos y la cantidad de negativos. Mostrar ambos resultados con una leyenda aclaratoria.

Ejercicio 18

Ingresar un solo par de valores **A** y **B** y efectuar las 4 operaciones básicas. Considerar que, si se intenta dividir por cero, en lugar del resultado, se debe imprimir el mensaje "**No es posible la división**".

Ejercicio 19

Realizar un algoritmo que permita ingresar la cantidad de productos comprados y el precio unitario de éstos, cuando la cantidad es igual a cero se termina el ingreso de datos. Se desea conocer el monto a abonar teniendo en cuenta que si la compra es superior a \$15000- se realiza un descuento del 12%. Mostrar el resultado final con una leyenda indicando el valor original, el descuento y el valor final.

Ejercicio 20

Desarrollar un algoritmo que permita ingresar 3 calificaciones de un alumno, calcule el promedio de las mismas e informe si está aprobado o no, según sea el promedio.

Promedio ≥ 7 , Aprobado. Promedio ≥ 4 y < 7 , Diciembre. Promedio < 4 , Marzo.

Ejercicio 21

En una estación de servicio se ingresa por día la cantidad de nafta súper y especial vendida en el corriente mes (30 días), como así también se dispone del total de ambas naftas del mes anterior. Ingresar todos estos datos e informar si hubo o no incremento en la venta y el porcentaje vendido de cada nafta en el mes respecto del mes anterior.

Ejercicio 22

Dada una serie de 16 números, todos distintos de cero, informar cuántos son pares y el promedio de los impares.

Ejercicio 23

Ingresar 25 números enteros e informar (mostrar con una leyenda):

- la suma de los mismos
- el promedio de los valores ingresados

Expresión de Problemas y Algoritmos

- el menor de todos estos números y en que ubicación se ingresó.

Ejercicio 24

En una oficina trabajan 25 empleados, de los cuales se ingresan el nombre y el sueldo. Incrementar en un 14% los sueldos menores a \$ 75.000- e informar:

- Nombre y nuevo sueldo de cada empleado
- Cantidad de empleados con sueldo inferior a \$ 75.000-
- Importe total de sueldos a abonar.

Ejercicio 25

En una empresa trabajan 250 empleados, de los cuales se ingresan el nombre, sueldo del convenio y cantidad de horas trabajadas.

El valor hora es el sueldo del convenio dividido 180 (porque se calcula sobre la base de 45 horas semanales).

Si la persona trabaja más de 180 horas mensuales, se considera cada una de éstas horas, una hora extra, y el valor de la misma es el 100% superior al valor de la hora normal.

Calcular el sueldo, teniendo presente en los casos de aquellos sueldos menores a \$ 75.000- donde se debe de incrementar en un 14%

Informar (mostrar al final):

- Nombre y nuevo sueldo de cada empleado
- Cantidad de empleados con sueldo inferior a \$ 75.000-
- Cantidad total de horas extras abonadas.
- Importe total de sueldos a abonar.

Ejercicio 26

Ingresar las medidas de los **3 lados de un triángulo** e informar de que tipo de triángulo se trata. Equilátero, Isósceles, Escaleno.

Informarlo con una leyenda

Ejercicio 27

Ingresar las medidas de los **3 lados de un triángulo rectángulo** e informar si se verifican sus catetos y la hipotenusa.

Informarlo con una leyenda

Ejercicio 28

Se ingresa el valor de la compra y el pago recibido (ambos valores enteros), debe de calcularse la cantidad y clase de cambio que debe regresarse.

Considerar que los billetes son \$1000, \$500, \$200, \$100, \$50, \$20, \$10.

Además de monedas de \$5, \$2, \$1

Expresión de Problemas y Algoritmos

Debe verificarse en primer lugar si el pago es suficiente, de no serlo, se debe imprimir una leyenda "Pago Insuficiente".

Si el pago excede el valor de la compra, debe especificarse el cambio a devolver y la cantidad de cada uno de los billetes o monedas, teniendo presente que existe la cantidad suficiente de cambio para abastecer a la demanda.

Ejercicio 29

En una fábrica de neumáticos, se ha registrado el número de máquina (del 1 al 4) y la medida que tiene cada uno (diámetro).

Además, para cada máquina se conoce el diámetro máximo y mínimo permitido en la fabricación.

Teniendo presente que se han de analizar 5000 neumáticos, dónde la información con la que se cuenta es:

Número de máquina, diámetro, diámetro máximo y mínimo permitido

Hallar para cada máquina que porcentaje de neumáticos fueron fabricados fuera de tolerancia respecto de la producción total de cada máquina.

Los resultados deben de estar acompañados por una leyenda para poder interpretar los mismos.

Ejercicio 30

Ingresar una serie indeterminada de números. Cuando se ingrese cero (0), se detendrá el ingreso de números y se deberá informar el promedio de todos los números ingresados y cuál fue el mayor valor de la serie.

Ejercicio 31

El dato de entrada de los alumnos es el número de DNI y 3 notas.

Hallar el promedio de éstas e informar (mostrar), para cada alumno, el número de DNI, el promedio obtenido y el comentario que corresponda según el resultado del promedio obtenido:

<u>Comentario</u>	<u>Promedio Obtenido</u>
Desaprobado:	<4
Aprobado:	≥ 4 y <6
Bueno:	≥ 6 y <8
Distinguido:	≥ 8 y <10
Sobresaliente:	10

El fin del ingreso de datos es cuando se ingresa el DNI=0.

Expresión de Problemas y Algoritmos

Ejercicio 32

Ingresar una serie indeterminada de números enteros. Cuando se ingrese cero (0), se detendrá el ingreso de números y se deberá informar el promedio de todos los números ingresados, cuál fue el mayor valor de la serie, el menor y la posición dónde se encuentra cada uno.

Ejercicio 33

Una empresa tiene 4 áreas de producción y por cada tarea realizada en cada pieza se registra la siguiente información:

Área Nro, Horas Empleadas

El fin del ingreso de datos se produce cuando ingresa el área Nro 5.

Leer sucesivamente los datos y determinar el total de horas empleado por cada área y que porcentaje representa cada área respecto al total de horas de producción.

Los valores de los respectivos resultados deberán estar acompañados por sus respectivas leyendas que indican que representa cada valor.

Ejercicio 34

En un curso de 45 alumnos, se han tomado tres exámenes parciales, y la información que llega es la siguiente:

Alumno Nota1 Nota2 Nota3 Nota4

Cuando se ingresa cada conjunto de datos, calcular el promedio y generar un listado que además del alumno, coloque el promedio con una leyenda que permita identificar si está aprobado o no (para aprobar la nota debe ser superior o igual a 6).

Además, al final se debe de informar la cantidad de alumnos aprobados y desaprobados

Ejercicio 35

Una fábrica de automóviles realizó una encuesta sobre los gustos de los compradores de acuerdo a las siguientes opciones.

Número de Opción:

1. Lujo
2. Maniobrabilidad
3. Rapidez
4. Economía
5. Otros motivos

Expresión de Problemas y Algoritmos

El ingreso de datos, se produce con la **Edad, Nro de Opción**; cuando se ingresa en el Nro de Opción el número 6, indica el fin del ingreso de datos.

Se necesita calcular:

- La cantidad de encuestados entre 18 y 30 años, inclusive, que prefieren la economía.
- La cantidad total de encuestados
- El porcentaje de encuestados mayores a 30 años que prefieren el lujo
- La cantidad de encuestados mayores a 40 años y menores a 55 años, que prefieren la rapidez.

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 36

Se han registrado tres temperaturas en el transcurso del mes con el siguiente formato:

Día, Temp1, Temp2, Temp3

Se necesita realizar un listado con la temperatura promedio de cada día. Asimismo, se necesita saber cuál ha sido la temperatura más baja y el día más caluroso (con mayor promedio). Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 37

Se ingresan 100 valores enteros no nulos y se necesita calcular:

- Si el valor es menor a 100, calcularle a cada uno un 10% y acumularlo en una variable
- Si el valor es mayor a 1000, calcular a cada uno un 20% y acumularlo en una variable
- Con otro valor, calcular a cada uno un 15% y acumularlo en una variable

Al finalizar el proceso se debe mostrar las tres variables calculadas

Ejercicio 38

En un colegio se necesita saber el promedio de altura de sus alumnos en cada turno, la información que llega es la siguiente:

Alumno/a, altura (en cmtrs), turno (1- Mañana / 2 – Tarde)

El fin del ingreso de datos se produce cuando llega el alumno=0.

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Expresión de Problemas y Algoritmos

Ejercicio 39

Se ingresa un par de valores enteros NumA y NumB, los cuales no son iguales y obtener:

- La cantidad de positivos
- El promedio de los negativos
- La suma de todos los valores de NumA positivos con valores NumB negativos.

El fin de ingreso de datos se produce cuando en el ingreso el NumA=0 y NumB=0.

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 40

Se ingresan 100 pares de valores enteros NumA y NumB, los cuales no son iguales y obtener:

- La suma de los valores menores de cada par
- El promedio de los negativos
- El máximo valor de NumA y su orden de ingreso
- El mínimo valor de NumB y su orden de ingreso

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 41

Se ingresan 100 números enteros no nulos y se necesita obtener:

- La suma de los números positivos múltiplos de 7.
- El promedio de los negativos ingresados en orden impar

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 42

Se ingresan un par de valores, que corresponden al código y un número entero. [cod ; num]

El fin de ingreso de datos se produce con el código 0. Únicamente existen el código 1, 2, 3.

Se pide calcular:

- El promedio de los valores ingresados con código 1.
- El máximo y el mínimo valor con su respectiva posición, con código 2.
- El promedio de los positivos y la cantidad de negativos con código 3.
- El porcentaje correspondiente a cada código respecto del total ingresado.

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 43

Se ingresan 100 pares de valores enteros NumA y NumB, los cuales no son iguales y obtener:

Expresión de Problemas y Algoritmos

- La suma de los mayores a -5
- El promedio de los múltiplos de 3 de NumB
- Imprimir los números comprendidos entre 100 y 200, inclusive, del NumA
- ¿Cuál es el porcentaje de los negativos menores a -10 del NumB?

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Ejercicio 44

Se ingresan números enteros no nulos (el ingreso del 0 representa el fin del ingreso de datos), de los cuales se debe calcular:

- El promedio de los negativos menores a -15 ingresados en orden impar.
- El menor de los positivos y su orden de ingreso
- El porcentaje de valores ingresados entre 25 y 50 respecto del total ingresado.

Los valores se deben mostrar conjuntamente con una leyenda para poder interpretarse correctamente.

Corte de Control

Ejercicio 45

Se cuenta con un conjunto de números positivos, ordenados en forma creciente. Cada número se repite una cantidad de veces indefinida y el último número del conjunto es 0 (cero). Se solicita realizar un algoritmo que, ingresando por teclado dichos números, informe:

- La cantidad de veces que se repite cada uno.
- Informe el número que más veces se repitió.

Ejercicio 46

Se cuenta con un conjunto de temperaturas tomadas en distintos días de un mismo mes (30 días), ordenadas cronológicamente. En cada día se tomaron diferentes cantidades de temperaturas. Se solicita realizar un algoritmo que, leyendo el día y la temperatura, muestre, para cada día, la temperatura máxima, la temperatura mínima y la temperatura promedio registradas.

Ejercicio 47

Se tiene la información de las ventas del mes efectuadas a los clientes diariamente, ordenado por el código de cliente, contando en el lote de datos la siguiente información: el código de cliente, el día y el importe vendido. El último dato del lote tiene cliente igual a cero. Se solicita un algoritmo para informar:

- El listado de las ventas a cada cliente, indicando, además, el total vendido al cliente.

Expresión de Problemas y Algoritmos

- Informar el cliente que más compras realizó.
- De cada cliente, informar el promedio de las compras que realizó.
- Informar el cliente que menor importe total compró.

Ejercicio 48

Una compañía aérea realiza viajes a Europa. Ingresan los vuelos diarios ordenados por destino, indicando el Número de Vuelo, Cantidad de Pasajeros por Vuelo y el Código de Destino (es una letra distinta para cada país)., Finaliza ingresando Código de Destino en blanco. Informar:

- Si cada avión viaja completo o con lugares libres, considerando que la capacidad de todos los aviones es de 150 pasajeros.
- El porcentaje de vuelos completos del día.
- La cantidad de pasajeros que viajaron a cada país.

Ejercicio 49

En una fábrica de zapatos, tienen varias máquinas que producen distintos modelos de zapatos, tiene el siguiente archivo de producción, previamente ordenado:

Tamaño/Cantidad

La finalización está determinada por el fin del archivo EOF.

Se debe tener en cuenta que pueden existir más de una cantidad por tamaño (varios registros que dependen de que máquina lo fabricó).

Se necesita (calcular y mostrar):

- El promedio general de cada tamaño.
- Cuantos tamaños tienen una cantidad total mayor a 300.
- Cuál ha sido el tamaño con menor fabricación.

Ejercicio 50

En un laboratorio de investigación se tiene el siguiente archivo, previamente ordenado:

Enfermedad/Cantidad de Bacterias

La finalización está determinada por EOF.

Se debe tener en cuenta que pueden existir más de un registro de bacterias por enfermedad.

Se necesita (calcular y mostrar):

- El promedio general de bacterias de cada enfermedad.
- Cuantas enfermedades tienen una cantidad acumulada de bacterias superior a 30.000.
- Cuál ha sido la enfermedad con menor cantidad de bacterias.

Expresión de Problemas y Algoritmos

Ejercicio 51

En un archivo vienen informados los datos de las notas que los alumnos obtienen en las materias que cursan, se encuentra ordenado por Nro Legajo- Apellido y Nombre del alumno. Asimismo, existen más de una nota por materia, por lo tanto, se desea saber:

- Cuál alumno resultó el de mayor promedio, y su promedio.
- La nota promedio de cada alumno en cada materia.
- El promedio general (de todos los alumnos y todas las materias).

Nro Legajo-Nombre y Apellido	Código Materia	Nota
------------------------------	----------------	------

Imprimir con una leyenda los valores solicitados.

La finalización está determinada por las siglas NN (Ningún Nombre).

VECTORES (Array/Arreglos)

Ejercicio 52

Generar un arreglo de 25 componentes en el cual el contenido del mismo contiene los primeros números naturales pares e imprimirlo.

Ejercicio 53

Ingresar un conjunto VEC1 de 50 números enteros. A partir de este conjunto generar otro VEC2 en el que cada elemento sea el doble del elemento homólogo de VEC1. Finalmente imprimir ambos vectores a razón de un valor de cada uno por renglón.

Ejercicio 54

Ingresar un conjunto de números enteros en un vector denominado VEC1 de 100 componentes. Si la suma de los componentes (el total del vector) resulta par, imprimir las de índice par, sino las de índice impar.

Ejercicio 55

Ingresar un conjunto de números enteros en un vector denominado VEC1, de 41 posiciones. Luego generar otro vector VEC2, donde se han de intercambiar los valores de las posiciones de VEC1 de forma equidistante.

Imprimir en paralelo ambos vectores.

Ejercicio 56

Ingresar un conjunto de números enteros en un vector de 50 posiciones, VEC1 y detectar el mayor y menor, con sus respectivas posiciones. Además, obtener el promedio del conjunto de valores ingresado.

Ordenamiento y Búsqueda

Una de las aplicaciones más frecuentes en los algoritmos es la ordenación.

Existen dos técnicas de ordenación fundamentales en gestión de datos: ordenación de listas y ordenación de archivos.

Los datos se pueden ordenar en orden ascendente o en orden descendente.

Cada recorrido de los datos durante el proceso de ordenación se conoce como pasada o iteración.

Los algoritmos de ordenación básicos son:

- Selección.
- Inserción.
- Burbuja.

Los algoritmos de ordenación más avanzados son:

- Shell.
- Quicksort.
- Heapsort (por montículos).
- Mergesort.

La eficiencia de los algoritmos de burbuja, inserción y selección es $O(n^2)$.

La eficiencia del algoritmo quicksort es $O(n \log n)$.

La búsqueda es el proceso de encontrar la posición de un elemento destino dentro de una lista.

Existen dos métodos básicos de búsqueda en arrays: **búsqueda secuencial y binaria**.

- La **búsqueda secuencial** se utiliza normalmente cuando el array no está ordenado. Comienza en el principio del array y busca hasta que se encuentra el dato buscado o se llega al final de la lista.
- Si un array está ordenado, se puede utilizar un algoritmo más eficiente denominado **búsqueda binaria**.
- La eficiencia de una búsqueda secuencial es $O(n)$.
- La eficiencia de una búsqueda binaria es $O(\log n)$.

Ejercicio 57

Ingresar un conjunto de números enteros en un vector (VEC1) de 30 posiciones y ordenar de forma descendente el mismo en un segundo vector VEC2 mediante el método de burbuja

Expresión de Problemas y Algoritmos

Mostrar en paralelo ambos vectores.

Ejercicio 58

Ingresar un conjunto de números enteros en un vector (VEC1) de 30 posiciones y ordenar de forma descendente el mismo en un segundo vector VEC2 mediante el método insert
Mostrar en paralelo ambos vectores.

Ejercicio 59

Ingresar un conjunto de números enteros en un vector (VEC1) de 30 posiciones y ordenar de forma descendente el mismo en un segundo vector VEC2 mediante el método Shell
Mostrar en paralelo ambos vectores.

Ejercicio 60

Hacer un algoritmo que permita ingresar n cantidad de trabajadores. (se debe de leer n y en base a dicho valor generar los vectores necesarios, que deben de estar enlazados por sus posiciones homologas, es decir iguales).

Su dato de entrada es: código, nombre, apellido, día, mes y año de nacimiento

Se debe imprimir una lista de los trabadores ordenados por fecha de nacimiento

Ejercicio 61

Generar un algoritmo que permita cargar los elementos de una factura (código, artículo, precio unitario y cantidad)

Los ítems máximos, conjunto de productos son 25, que es el máximo por factura.

Si se desean vender menos, se lo debe de informar para generar una correcta carga de datos.

Al momento de imprimir los mismos, deben de estar ordenados de forma decreciente por el monto de la venta, es decir precio unitario por cantidad.

Ejercicio 62

En un curso donde previamente se debe de cargar la cantidad límites de alumnos(n)

Generar un algoritmo que permita el ingreso del Nombre y Apellido del Alumno, su DNI, la Nota 1, la Nota 2 y la Nota 3.

A partir de ahí, generar un listado donde deben aparecer ordenados de forma decreciente según su promedio.

Ejercicio 63

Ingresar un conjunto de números enteros en un vector (VEC1) de 30 posiciones y ordenar de forma descendente el mismo en un segundo vector VEC2 mediante el método de burbuja
Mostrar en paralelo ambos vectores.

Y luego generar una búsqueda binaria de uno de los elementos.

Expresión de Problemas y Algoritmos

Ejercicio 64

Se desea eliminar todos los números duplicados de una lista o vector (array). Por ejemplo, si el array toma los valores:

4 7 11 4 9 5 11 7 3 5

ha de cambiarse a:

4 7 11 9 5 3

Escribir una función que elimine los elementos duplicados de un array.

Ejercicio 65

Escribir una función que elimine los elementos duplicados de un vector ordenado. ¿Cuál es la eficiencia de éste algoritmo? Compare la eficiencia del mismo con el del ejercicio 64.

Ejercicio 66

Un vector contiene los elementos mostrados a continuación. Los primeros dos elementos se han ordenado utilizando un algoritmo de inserción.

3 13 8 25 45 23 98 58

¿Cuál será el valor de los elementos del vector después de tres pasadas más del algoritmo?

Ejercicio 67

Dada la siguiente lista:

47 3 21 32 56 92

Después de dos pasadas de un algoritmo de ordenación, el array ha quedado dispuesto así:

3 21 47 32 56 92

¿Qué algoritmo de ordenación se está utilizando (selección, burbuja o inserción) ? Justifique la respuesta.

Ejercicio 68

Un array contiene los elementos indicados más abajo. Utilizando el algoritmo de ordenación Shell encuentre las pasadas y los intercambios que se realizan para su ordenación.

8 43 17 6 40 16 18 97 11 7

Ejercicio 69

Partiendo del mismo array que en el ejercicio 68, encuentre las particiones e intercambios que realiza el algoritmo de ordenación quicksort para su ordenación.

Ejercicio 70

Un array de registros se quiere ordenar según el campo clave fecha de nacimiento. Dicho campo consta de tres subcampos: día, mes y año de 2, 2 y 4 dígitos respectivamente. Adaptar el método de la burbuja a esta ordenación.

Expresión de Problemas y Algoritmos

Ejercicio 71

Un array contiene los elementos indicados a continuación. Utilizando el algoritmo de búsqueda binaria, trazar las etapas necesarias para encontrar el número 88.

8 13 17 26 44 56 88 97

Igual búsqueda, pero para el número 26.

Ejercicio 72

Escribir el algoritmo de ordenación correspondiente al método Shell para poner en orden alfabético una lista de n nombres.

Ejercicio 73

Escribir una función de búsqueda binaria aplicado a un array ordenado descendientemente.

Ejercicio 74

Supongamos que se tiene una secuencia de n números que deben ser clasificados:

1. Si se utilizar el método de Shell, ¿cuántas comparaciones y cuántos intercambios se requieren para clasificar la secuencia si:

- Ya está clasificado,
- Está en orden inverso.

2. Realizar los mismos cálculos si se utiliza el algoritmo quicksort.

MATRICES

Ejercicio 75

Realizar un proceso donde se genere una matriz, denominada MAT1, dónde los datos se han de almacenar dentro de una matriz compuesta por 10 filas y 8 columnas; y su contenido sean números enteros.

Se deben de ingresar los siguientes datos; Número $A \leq 10$; Número $B \leq 8$ y Número C (que representan la dirección del número a guardar en la matriz y un número entero distinto de Cero que representa C).

Sí el número C, es un cero representa el final del ingreso de datos. Pueden existir ingresos de datos que tienen la misma dirección, por lo tanto, se deben de acumular los valores ingresados.

A continuación, desarrollar un algoritmo que realice los siguientes procedimientos:

- Imprima la matriz MAT1.

Expresión de Problemas y Algoritmos

- Calcule e imprima el valor promedio de los componentes que conforman cada fila de la matriz en la columna 9.
- Genere e imprima un vector VECMAXCOL donde cada componente sea **el valor máximo de cada columna**.

Ejercicio 76

Se tiene un archivo formado con la siguiente información:

N° de Vendedor	Valor de la venta realizada
----------------	-----------------------------

Existen 10 vendedores, y se debe de tener en cuenta que pueden existir varias ventas del mismo vendedor, los datos ingresan respetando el formato establecido y pueden llegar en cualquier orden. El fin del ingreso de datos se da cuando llega el vendedor N° 11.

Se necesita realizar:

- Leer e imprimir el contenido del archivo a medida que se van leyendo.
- Hallar la cantidad total vendida por cada vendedor y emitir un listado ordenado en forma decreciente dónde figure el N° de vendedor y el total de su venta realizada.

Ejercicio 77

En una empresa de software se ha registrado la siguiente información por cada tarea realizada:

Categoría Empleado	Área	Horas Trabajadas
--------------------	------	------------------

Las categorías de los empleados están registradas según los siguientes códigos:

0. Jefe de Proyecto
1. Analista Funcional
2. Programador

Expresión de Problemas y Algoritmos

Dentro de la empresa hay 10 áreas distintas. El fin de ingreso de datos se produce cuando llega la categoría 3.

Los datos ingresan respetando el formato establecido y pueden llegar en cualquier orden (entiéndase cualquier número de fila y columna, dentro de los valores establecidos).

Hallar la cantidad de horas trabajadas en cada sección por cada categoría, imprimir dicha información. Además, se necesita conocer el total de horas por cada categoría.

Ejercicio 78

Una empresa tiene cuatro depósitos, donde se almacenan los artículos fabricados, los cuales son diez, que son vendidos por sus doce jefes de ventas.

Por cada venta realizada se registra la siguiente información:

N° de Depósito	N° de Artículo	N° de Vendedor	Valor de la Venta
----------------	----------------	----------------	-------------------

Pueden existir varias ventas del mismo depósito, del mismo artículo, del mismo vendedor, y los datos ingresan respetando el formato establecido y pueden llegar en cualquier orden.

El final de ingreso de datos se produce con el número de depósito 5.

Se pide:

- Cuál fue el valor de venta máximo, indicando el depósito, el artículo y el vendedor correspondiente.
- Realizar e imprimir una tabla con las ventas totales de cada depósito ordenada de forma decreciente, indicando el número de depósito y la venta total.

Solucionario de los Ejercicios de la Guía

¿Qué es un solucionario?

Son las respuestas a los ejercicios, pero NO a todos los ejercicios propuestos en la guía.

Los ejercicios resueltos son los IMPARES.

Se encuentran desarrollados en Lenguaje Python, por lo tanto, con escribirlos e interpretarlos, utilizando como referencia cada uno de los ejercicios podrán mejorar la calidad de desarrollo, tanto en la actual materia como en futuras.

Existen varias herramientas que permiten trabajar de forma sencilla con el lenguaje seleccionado, asimismo, la sintaxis, es decir la forma de escribir cada instrucción es muy similar a pseudocódigo.

Ejercicio 1

```
ValorA= int(input ('Ingrese el valor A: '))
ValorB= int(input ('Ingrese el valor B: '))

Suma= ValorA + ValorB
Resta= ValorA - ValorB
Multiplica= ValorA * ValorB
Divi= ValorA / ValorB

print ('El resultado de La Suma es :', Suma)
print ('El resultado de La Resta es :', Resta)
print ('El resultado de La Multiplicación es :', Multiplica)
print ('El resultado de La División es :', Divi)
```

Ejercicio 3

```
contador=3
suma=0

while contador < 100:
    print (contador)
    suma=suma + contador
    contador =contador +3

print ('La suma de La repetición es:', suma)
```

Expresión de Problemas y Algoritmos

Ejercicio 5

```

ValorA= int(input ('Ingrese el valor A:'))
ValorB= int(input ('Ingrese el valor B:'))
ValorC= int(input ('Ingrese el valor C:'))

if ValorA ==(ValorB+ValorC):
    print('El valor de A es La suma de B y C')
elif ValorB ==(ValorA+ValorC):
    print('El valor de B es La suma de A y C')
elif ValorC ==(ValorB+ValorA):
    print('El valor de C es La suma de B y A')
else:

    print('Ninguno cumple el enunciado')

```

Otra forma de resolverlo

```

ValorA= int(input ('Ingrese el valor A:'))
ValorB= int(input ('Ingrese el valor B:'))
ValorC= int(input ('Ingrese el valor C:'))

if ValorA ==(ValorB+ValorC):
    print('El valor de A es La suma de B y C')
else:
    if ValorB ==(ValorA+ValorC):
        print('El valor de B es La suma de A y C')
    else:
        if ValorC ==(ValorB+ValorA):
            print('El valor de C es La suma de B y A')
        else:

            print('Ninguno cumple el enunciado')

```

Ejercicio 7

```

ValorA= int(input ('Ingrese el valor del Radio:'))

Long_Circunfe=2*3.1415*ValorA
Sup_Circulo= 3.1415 * ValorA **2

print('El valor de La Longitud de La circunferencia es:', Long_Circunfe)

print('El valor de La superficie de La circunferencia es:', Sup_Circulo)

```


Expresión de Problemas y Algoritmos

Ejercicio 9

```

ValorHora= int(input ('Ingrese el valor de La Hora:'))
ValorCant_Hs= int(input ('Ingrese La cantidad de horas semanales:'))

Horas_Mes= ValorCant_Hs *4
Sueldo_Bruto= Horas_Mes * ValorHora
Jubi= Sueldo_Bruto * 0.11
ObraSocial= Sueldo_Bruto * 0.03
AporteSistemaSalud= Sueldo_Bruto * 0.03

Sueldo_Neto = Sueldo_Bruto - Jubi - ObraSocial - AporteSistemaSalud

print ('El Sueldo Bruto es:', Sueldo_Bruto)
print ('La Jubilación es:', Jubi)
print ('El Aporte a La Obra Social es:', ObraSocial)
print ('El Aporte al Sistema de Salud es:', AporteSistemaSalud)

print ('El Sueldo Neto es:', Sueldo_Neto)

```

Ejercicio 11

```

Codigo= int(input ('Ingrese el código 1- Fahrenheit // 2-Kelvin:'))
while Codigo == (1 or 2):
    Centi= int(input ('Ingrese La Temperatura en Centígrados:'))
    if Codigo == 1:
        Farh= (9//5)*Centi +32
        print('El valor de La temperatura en Fahrenheit es: ',Farh)
    else:
        Kelvin= Centi + 273
        print('El valor de La temperatura en Kelvin es: ',Kelvin)
    Codigo= int(input ('Ingrese el código 1- Fahrenheit // 2-Kelvin:'))

print('El valor del Código es Incorrecto')

```

Ejercicio 13

```

contador=120

while contador <= 199:
    contador =contador+1

    print ('Los mayores a 120 son:',contador)

```

Expresión de Problemas y Algoritmos

Ejercicio 15

```

ValorA= int(input ('Ingrese el valor A:'))
ValorB= int(input ('Ingrese el valor B:'))

if ValorA > ValorB:
    print('El valor de A:', ValorA, ', es Mayor al de B:', ValorB)
else:
    if ValorB > ValorA:
        print('El valor de B:', ValorB, ', es Mayor al de A:', ValorA)
    else:
        print('Ambos Valores son iguales')

```

Ejercicio 17

```

Contador=0
SumaPos=0
ContarPos=0
ContarNeg=0
DatoEntrada=0

DatoEntrada= int(input ('Ingrese un número Entero distinto de Cero:'))
while Contador < 100:
    while DatoEntrada == 0:
        print('El valor del Dato de Entrada es Incorrecto')
        DatoEntrada= int(input ('Ingrese un número Entero distinto de Cero:'))

    Contador=Contador +1
    if DatoEntrada > 0:
        ContarPos=ContarPos +1
        SumaPos=SumaPos + DatoEntrada
    else:
        ContarNeg= ContarNeg +1
    DatoEntrada= int(input ('Ingrese un número Entero distinto de Cero:'))

if ContarPos > 0:
    Promedio=SumaPos//ContarPos
    print('El Promedio de Los Positivos es: ',Promedio)
else:
    print('No ingresaron valores POSITIVOS')

print('La Cantidad de Negativos es:',ContarNeg)

```

Expresión de Problemas y Algoritmos

Ejercicio 19

```

ContadorProductos=0
Contador=0
Producto=0
Sub_Total=0
Descuento=0
Total=0
LeyendaProducto= ''
LeyendaValor= ''

print('La cantidad de productos debe ser mayor a 0')
ContadorProductos= int(input ('Ingrese La cantidad de productos:'))
while ContadorProductos!=0:
    while Contador < ContadorProductos:
        Contador=Contador +1
        LeyendaProducto = 'Ingrese La cantidad deL PRODUCTO ' + str(Contador) + ':'
        Producto= int(input (LeyendaProducto))
        LeyendaValor = 'EL PRECIO PRODUCTO ' + str(Contador) + ' es:'
        ValorProducto=int(input (LeyendaValor))
        Sub_Total = Sub_Total + Producto * ValorProducto

    if Sub_Total > 15000:
        Descuento= Sub_Total * 0.12
        Total= Sub_Total - Descuento
        print('EL TOTAL es:',Sub_Total)
        print('EL DESCUENTO es:',Descuento)
        print('EL Valor Final es:',Total)
    else:
        print('EL Valor Final es:',Sub_Total)
    ContadorProductos= int(input ('Ingrese La cantidad de productos (0 para
TERMINAR):'))
    Contador=0
    Sub_Total=0

```

Ejercicio 21

```

NaftaSuperAnterior=0
NaftaEspecialAnterior=0
NaftaSuperActual=0
NaftaEspecialActual=0
NumeroDia=1
Tot_Naf_Sup_Act=0
Tot_Naf_Esp_Act=0

```

Expresión de Problemas y Algoritmos

```

NaftaSuperAnterior= int(input ('Ingrese La cantidad de Nafta Súper del Mes Anterior:'))
NaftaEspecialAnterior= int(input ('Ingrese La cantidad de Nafta Especial del Mes
Anterior:'))

while NumeroDia >=1 and NumeroDia <=30:
    print('Carga de combustible del día:', str(NumeroDia))
    NaftaSuperActual= int(input ('Ingrese La cantidad de Nafta Súper del día:' +
str(NumeroDia) + ' :'))
    NaftaEspecialActual= int(input ('Ingrese La cantidad de Nafta Especial del día:' +
str(NumeroDia) + ' :'))
    Tot_Naf_Sup_Act=Tot_Naf_Sup_Act + NaftaSuperActual
    Tot_Naf_Esp_Act=Tot_Naf_Esp_Act + NaftaEspecialActual
    NumeroDia = NumeroDia +1

if Tot_Naf_Sup_Act > NaftaSuperAnterior :
    DifNSpos=Tot_Naf_Sup_Act - NaftaSuperAnterior
    print ('Se incremento La cantidad de Nafta súper vendida en:',DifNSpos)
    PorcNSpos= ((Tot_Naf_Sup_Act/NaftaSuperAnterior)-1)*100
    print ('El porcentaje de incremento La cantidad de Nafta súper vendida
es:',round(PorcNSpos,2))
else:
    DifNSneg=Tot_Naf_Sup_Act - NaftaSuperAnterior
    print ('Se dejó de vender La cantidad de Nafta súper en:',DifNSneg)
    PorcNSneg= (1 - (Tot_Naf_Sup_Act/NaftaSuperAnterior))*100
    print ('El porcentaje de perdida de La cantidad de Nafta súper vendida
es:',round(PorcNSneg,2))

if Tot_Naf_Esp_Act > NaftaEspecialAnterior :
    DifNEpos=Tot_Naf_Esp_Act - NaftaEspecialAnterior
    print ('Se incremento La cantidad de Nafta Especial vendida en:',DifNEpos)
    PorcNEpos= ((Tot_Naf_Esp_Act/NaftaEspecialAnterior)-1)*100
    print ('El porcentaje de incremento La cantidad de Nafta Especial vendida
es:',round(PorcNEpos,2))
else:
    DifNEneg=Tot_Naf_Esp_Act - NaftaEspecialAnterior
    print ('Se dejó de vender La cantidad de Nafta Especial en:',DifNEneg)
    PorcNEneg= (1 - (Tot_Naf_Esp_Act/NaftaEspecialAnterior))*100

    print ('El porcentaje de perdida de La cantidad de Nafta súper vendida
es:',round(PorcNEneg,2))

```

Expresión de Problemas y Algoritmos

Ejercicio 23

```

suma=0
dato=0
contar=1
menor=0
posicion=0

while contar <= 25:
    dato= int(input ('Ingrese un número entero:'))
    if contar == 1:
        posicion=1
        menor=dato
    if dato < menor:
        posicion=contar
        menor=dato
    suma= suma+ dato
    contar=contar + 1

promedio= suma/25
print('La suma de los valores ingresados es: ', suma)
print('El promedio es: ', promedio)
print('El Menor valor ingresado es: ', menor)
print('La posición de ingreso del menor valor es: ', posicion)

```

Ejercicio 25

```

contar=1
nombre= ' '
convenio=0
horastr=0
HsExtr=0
ValorHsConv=0
dife=0
sueldo=0
TotalHsExtr=0
TotalSueldos=0
SueldosMenores=0

while contar <=3:
    nombre=input('Ingrese el Nombre del Empleado:')
    convenio= int(input ('Ingrese el valor de sueldo por convenio:'))
    horastr=int(input ('Ingrese La cantidad de horas mensuales:'))
    ValorHsConv=convenio/180
    if horastr>180:
        dife=horastr-180
        sueldo = ValorHsConv*180 + 2*ValorHsConv*dife

```

Expresión de Problemas y Algoritmos

```

    TotalHsExtr= TotalHsExtr + dife
else:
    sueldo = ValorHsConv * horastr

if sueldo < 75000:
    sueldo=sueldo*1.14
    SueldosMenores=SueldosMenores + 1

TotalSueldos=TotalSueldos + sueldo
print('El empleado: ',nombre)
print('Tiene un sueldo de: ',round(sueldo,2) )
contar= contar+1

```

```

print ('La cantidad de empleados con sueldos menores a $75.000 son: ',SueldosMenores)
print ('La cantidad total de horas extras abonadas son: ', TotalHsExtr)
print ('La cantidad total de sueldos abonados son: ', round(TotalSueldos,2))

```

Ejercicio 27

```

from math import sqrt
catetoA=0
catetoB=0
Hipo=0
catetos=0

catetoA= int(input ('Ingrese el valor de un cateto:'))
catetoB= int(input ('Ingrese el valor de un cateto:'))
Hipo= int(input ('Ingrese el valor La Hipotenusa:'))

if Hipo < (catetoA or catetoB):
    print ('Ningún cateto puede ser mayor a La hipotenusa')
else:
    catetos=sqrt((catetoA**2)+(catetoB**2))
    if Hipo== catetos:
        print ('Se verifica el triangulo rectángulo')
    else:

        print ('NO es un triangulo rectángulo')

```

Expresión de Problemas y Algoritmos

Ejercicio 29

```

contar=1
NumMaq=0
Diametro=0
M1Dmax=0
M1Dmin=0
M2Dmax=0
M2Dmin=0
M3Dmax=0
M3Dmin=0
M4Dmax=0
M4Dmin=0
Total1=0
TotalOK1=0
Total2=0
TotalOK2=0
Total3=0
TotalOK3=0
Total4=0
TotalOK4=0
Porce1=0
Porce2=0
Porce3=0
Porce4=0

while contar < 5:
    if contar==1:
        M1Dmax= int(input ('Ingrese el diametro Máximo de La Máquina 1:'))
        M1Dmin= int(input ('Ingrese el diametro Mínimo de La Máquina 1:'))

    if contar==2:
        M2Dmax= int(input ('Ingrese el diametro Máximo de La Máquina 2:'))
        M2Dmin= int(input ('Ingrese el diametro Mínimo de La Máquina 2:'))

    if contar==3:
        M3Dmax= int(input ('Ingrese el diametro Máximo de La Máquina 3:'))
        M3Dmin= int(input ('Ingrese el diametro Mínimo de La Máquina 3:'))

    if contar==4:
        M4Dmax= int(input ('Ingrese el diametro Máximo de La Máquina 4:'))
        M4Dmin= int(input ('Ingrese el diametro Mínimo de La Máquina 4:'))

    contar=contar + 1

contar=1

```

Expresión de Problemas y Algoritmos

```

while contar <=5000:
    NumMaq=int(input('Ingrese el Número de Maquina:'))
    Diametro= int(input ('Ingrese el diametro de La Cubierta:'))
    if NumMaq==1:
        Total1=Total1+1
        if Diametro < M1Dmax:
            if Diametro > M1Dmin:
                TotalOK1=TotalOK1+1

    if NumMaq==2:
        Total2=Total2+1
        if Diametro < M2Dmax:
            if Diametro > M2Dmin:
                TotalOK2=TotalOK2+1

    if NumMaq==3:
        Total3=Total3+1
        if Diametro < M3Dmax:
            if Diametro > M3Dmin:
                TotalOK3=TotalOK3+1

    if NumMaq==4:
        Total4=Total4+1
        if Diametro < M4Dmax:
            if Diametro > M4Dmin:
                TotalOK4=TotalOK4+1
    contar=contar+1

Fuera1= Total1-TotalOK1
Fuera2= Total2-TotalOK2
Fuera3= Total3-TotalOK3
Fuera4= Total4-TotalOK4

if Total1==0:
    print('Sin Datos Maquina 1')
else:
    Porce1=(Fuera1/Total1)*100
    print ('El Porcentaje de falla de La Maquina 1 es:', round(Porce1,2))

if Total2==0:
    print('Sin Datos Maquina 2')
else:
    Porce2=(Fuera2/Total2)*100
    print ('El Porcentaje de falla de La Maquina 2 es:', round(Porce2,2))

if Total3==0:
    print('Sin Datos Maquina 3')
else:
    Porce3=(Fuera3/Total3)*100

```


Expresión de Problemas y Algoritmos

```
print ('El Porcentaje de falla de La Maquina 3 es:', round(Porce3,2))

if Total4==0:
    print('Sin Datos Maquina 4')
else:
    Porce4=(Fuera4/Total4)*100
    print ('El Porcentaje de falla de La Maquina 4 es:', round(Porce4,2))
```

Ejercicio 31

```
DNI=0
Nota1=0
Nota2=0
Nota3=0
Suma=0
Leyenda= ''

print('El DNI 0 termina el proceso')
DNI= int(input ('Ingrese el Número de Documento:'))

while DNI !=0:
    Nota1= int(input ('Ingrese La Nota 1:'))
    Nota2= int(input ('Ingrese La Nota 2:'))
    Nota3= int(input ('Ingrese La Nota 3:'))
    Suma= Nota1 + Nota2 + Nota3
    Promedio= Suma / 3
    if Promedio < 4:
        Leyenda='DESAPROBADO'
    if Promedio >=4 and Promedio < 6:
        Leyenda='APROBADO'
    if Promedio >=6 and Promedio < 8:
        Leyenda='BUENO'
    if Promedio >=8 and Promedio < 10:
        Leyenda='DISTINGUIDO'
    if Promedio == 10:
        Leyenda='SOBRESALIENTE'

    print('El alumno con DNI:',DNI,' obtuvo un Promedio de :',round(Promedio,2),' y
es:',Leyenda)
    DNI= int(input ('Ingrese el Número de Documento:'))
```

Expresión de Problemas y Algoritmos

Ejercicio 33

```

NumArea=0
HorasEmple=0
TotalHorasEmple=0
Total1=0
Total2=0
Total3=0
Total4=0
Porce1=0
Porce2=0
Porce3=0
Porce4=0

NumArea=int(input('Ingrese el Número de Area (Fin de Ingreso Area 5):'))

while NumArea !=5:
    HorasEmple= int(input ('Ingrese La cantidad de Horas Empleadas:'))
    TotalHorasEmple=TotalHorasEmple + HorasEmple
    if NumArea==1:
        Total1=Total1+HorasEmple

    if NumArea==2:
        Total2=Total2+HorasEmple

    if NumArea==3:
        Total3=Total3+HorasEmple

    if NumArea==4:
        Total4=Total4+HorasEmple

    NumArea=int(input('Ingrese el Número de Area (Fin de Ingreso Area 5):'))

if Total1==0:
    print('Sin Datos Área 1')
else:
    Porce1=(Total1/TotalHorasEmple)*100
    print ('El Porcentaje del Área 1 es:', round(Porce1,2))

if Total2==0:
    print('Sin Datos Área 2')
else:
    Porce2=(Total2/TotalHorasEmple)*100
    print ('El Porcentaje del Área 2 es:', round(Porce2,2))

if Total3==0:
    print('Sin Datos Área 3')

```

Expresión de Problemas y Algoritmos

```

else:
    Porce3=(Total3/TotalHorasEmple)*100
    print ('El Porcentaje del Área 3 es:', round(Porce3,2))

if Total4==0:
    print('Sin Datos Área 4')
else:
    Porce4=(Total4/TotalHorasEmple)*100
    print ('El Porcentaje del Área 4 es:', round(Porce4,2))

```

Ejercicio 35

```

NumOpcion=0
Edad=0
TotalEncuestados=0
Total1=0
Total2=0
Total3=0
Porce1=0

print('OPCIONES')
print('1-Lujo')
print('2-Maniobrabilidad')
print('3-Rapidez')
print('4-Economía')
print('5-Otros Motivos')
print('6-SALIR')
NumOpcion=int(input('Ingrese el Número de Opción:'))

while NumOpcion !=6:
    Edad= int(input ('Ingrese La EDAD del encuestado:'))
    TotalEncuestados=TotalEncuestados + 1
    if NumOpcion==4:
        if Edad >=18 and Edad <=30:
            Total1=Total1+1

    if NumOpcion==1:
        if Edad > 30:
            Total2=Total2+1

    if NumOpcion==3:
        if Edad > 40 and Edad < 55:
            Total3=Total3+1

    print('OPCIONES')
    print('1-Lujo')
    print('2-Maniobrabilidad')

```

Expresión de Problemas y Algoritmos

```

print('3-Rapidez')
print('4-Economía')
print('5-Otros Motivos')
print('6-SALIR')
NumOpcion=int(input('Ingrese el Número de Opción:'))

if TotalEncuestados==0:
    print('Sin Datos')
else:
    Porce1=(Total12/TotalEncuestados)*100
    print ('La cantidad entre 18 y 30 que prefieren Economía son:', round(Total1,2))
    print ('La Cantidad Total de encuestados es:', round(TotalEncuestados,2))
    print ('El Porcentaje de mayores a 30 que prefieren el Lujo son:', round(Porcel,2))
    print ('La Cantidad de encuestados mayores a 40 y menores a 55 que prefieren
rapidez:', round(Total3,2))

```

Ejercicio 37

```

DatoEntrada=0
Contar=1
Total1=0
Total2=0
Total3=0

DatoEntrada=int(input('Ingrese un Número entero:'))

while Contar<=5:
    while DatoEntrada==0:
        DatoEntrada=int(input('Ingrese un Número entero:'))

    Contar=Contar + 1
    if DatoEntrada<100:
        Total1=Total1+DatoEntrada * 0.1

    if DatoEntrada>1000:
        Total2=Total2+DatoEntrada * 0.2

    if DatoEntrada>=100 and DatoEntrada<=1000:
        Total3=Total3+DatoEntrada * 0.15

    DatoEntrada=int(input('Ingrese un Número entero:'))

print('Acumulados valores menores a 100:', round(Total1,2))
print('Acumulados valores mayores a 1000:', round(Total2,2))
print('Acumulados valores mayores/iguales a 100 y menores/iguales a 1000:',
round(Total3,2))

```

Expresión de Problemas y Algoritmos

Ejercicio 39

```

NumA=0
NumB=0
Positivos=0
Negativos=0
ContarNeg=0
suma=0
Promedio=0

NumA=int(input('Ingrese un Número entero A:'))
NumB=int(input('Ingrese un Número entero B:'))

while NumA == NumB:
    print('Los valores no deben de ser iguales')
    NumA=int(input('Ingrese un Número entero A:'))
    NumB=int(input('Ingrese un Número entero B:'))

while not(NumA == 0) or not(NumB == 0):
    if NumA > 0:
        Positivos=Positivos + 1
    else:
        Negativos=Negativos + NumA
        ContarNeg=ContarNeg + 1

    if NumB > 0:
        Positivos=Positivos + 1
    else:
        Negativos=Negativos + NumB
        ContarNeg=ContarNeg + 1

    if NumA > 0:
        if NumB < 0:
            suma= suma + NumA + NumB

    NumA=int(input('Ingrese un Número entero A:'))
    NumB=int(input('Ingrese un Número entero B:'))

print('La cantidad de positivos es: ',Positivos)
if ContarNeg==0:
    print('No ingresaron negativos')
else:
    Promedio=Negativos/ContarNeg
    print('El Promedio de Los negativos es: ', round(Promedio,2))
print('La suma de NumA positivos con NumB negativos es:', suma)

```

Expresión de Problemas y Algoritmos

Ejercicio 41

```

DatoEntrada=0
Contar=1
Resto=0
Resto2=0
Suma7=0
Negativos=0
ContarNeg=0
Promedio=0

DatoEntrada=int(input('Ingrese un Número entero no nulo:'))

while DatoEntrada == 0:
    print('Número entero no nulo')
    DatoEntrada=int(input('Ingrese un Número entero no nulo:'))

while Contar<=10:
    Resto = DatoEntrada % 7
    Resto2= Contar % 2
    if Resto == 0:
        Suma7=Suma7 + DatoEntrada

    if Resto2 == 1:
        if DatoEntrada <0:
            Negativos=Negativos + DatoEntrada
            ContarNeg=ContarNeg + 1
    Contar = Contar +1
    DatoEntrada=int(input('Ingrese un Número entero no nulo:'))
    while DatoEntrada == 0:
        print('Número entero no nulo')
        DatoEntrada=int(input('Ingrese un Número entero no nulo:'))

if ContarNeg==0:
    print('No ingresaron negativos')
else:
    Promedio=Negativos/ContarNeg
    print('El Promedio de Los negativos es :', round(Promedio,2))

print('La suma de Los positivos múltiplos de 7 son:',Suma7)

```

Expresión de Problemas y Algoritmos

Ejercicio 43

```

NumA=0
NumB=0
Contar=1
SumaMayores5=0
Suma3=0
Contar3=0
ContarNeg=0
Promedio=0

NumA=int(input('Ingrese un Número entero A:'))
NumB=int(input('Ingrese un Número entero B:'))

while NumA == NumB:
    print('Los valores no deben de ser iguales')
    NumA=int(input('Ingrese un Número entero A:'))
    NumB=int(input('Ingrese un Número entero B:'))

while Contar<=100:
    Resto3= NumB % 3
    if NumA > -5:
        SumaMayores5=SumaMayores5 + NumA

    if NumB > -5:
        SumaMayores5=SumaMayores5 + NumB

    if Resto3 ==0:
        Suma3= Suma3 + NumB
        Contar3=Contar3 +1

    if (NumA >= 100) and (NumA <=200):
        print('El valor de NumA entre 100 y 200 es:', NumA)

    if NumB < -10:
        ContarNeg = ContarNeg +1

    Contar=Contar + 1
    NumA=int(input('Ingrese un Número entero A:'))
    NumB=int(input('Ingrese un Número entero B:'))
    while NumA == NumB:
        print('Los valores no deben de ser iguales')
        NumA=int(input('Ingrese un Número entero A:'))
        NumB=int(input('Ingrese un Número entero B:'))

print('La suma de Los mayores a -5 es: ',SumaMayores5)

if Contar3==0:

```

Expresión de Problemas y Algoritmos

```

    print('No ingresaron múltiplos de 3 de NumB')
else:
    Promedio=Suma3/Contar3
    print('El Promedio de los múltiplos de 3 de NumB es :', round(Promedio,2))

print('El Porcentaje de los negativos menores a -10 de NumB es:',ContarNeg)

```

Ejercicio 45

```

Num=0
ContarN=0
AuxNum=0
ContarM=0
Mayor=0
NumeroMay=0

Num=int(input('Ingrese un Número entero:'))
while Num!=0:
    AuxNum=Num
    while AuxNum==Num:
        ContarN=ContarN + 1
        Num=int(input('Ingrese un Número entero:'))

    ContarM=ContarM + 1
    if ContarM==1:
        Mayor=ContarN
        NumeroMay=AuxNum

    if ContarN > Mayor:
        Mayor=ContarN
        NumeroMay=AuxNum

    print('El Número ', AuxNum, 'se repitió ',ContarN , 'veces')
    ContarN=0

print('El Número con más repeticiones es el', NumeroMay, 'y se repitió',Mayor , 'veces')

```


Expresión de Problemas y Algoritmos

Ejercicio 47

```

CodCliente=0
Dia=0
Importe=0
AuxCliente=0
TotalImporte=0
CantCompras=0
PromTotalImporte=0
PromTotalCantidad=0
Contar=0
MayCompras=0
MayCliente=0
MinImporte=0
MinCliente=0
Promedio=0

CodCliente=int(input('Ingrese el Código del cliente:'))
Dia=int(input('Ingrese el día:'))
Importe=int(input('Ingrese el Importe:'))

while CodCliente!=0:
    AuxCliente=CodCliente
    while AuxCliente==CodCliente:
        TotalImporte=TotalImporte + Importe
        CantCompras=CantCompras +1
        CodCliente=int(input('Ingrese el Código del cliente:'))
        Dia=int(input('Ingrese el día:'))
        Importe=int(input('Ingrese el Importe:'))

    print('El Cliente:',AuxCliente)
    print('Compró por un total de:',TotalImporte)

    PromTotalImporte= PromTotalImporte + TotalImporte
    PromTotalCantidad=PromTotalCantidad + CantCompras

    Contar=Contar+1

    if Contar==1:
        MayCompras=CantCompras
        MayCliente=AuxCliente
        MinImporte=TotalImporte
        MinCliente=AuxCliente

    if TotalImporte < MinImporte:
        MinImporte=TotalImporte
        MinCliente=AuxCliente

```

Expresión de Problemas y Algoritmos

```
if CantCompras > MayCompras:
    MayCompras=CantCompras
    MayCliente=AuxCliente
```

```
TotalImporte=0
CantCompras=0
```

```
Promedio=PromTotalImporte/PromTotalCantidad
print('El promedio de las compras es:', round(Promedio,2))
print('El cliente con menor acumulado de compras es:', round(MinCliente,2), 'con un valor de: ', round(MinImporte,2))
print('El cliente con mayor cantidad de compras es:', round(MayCliente,2), 'con una cantidad de compras de:', round(MayCompras,2))
```

Ejercicio 49

```
CodTamano=0
Cantidad=0
AuxTamano=0
ProduccionTamano=0
CantTamano=0
Promedio=0
Contar=0
May300=0
MenorFab=0
MenorCant=0
```

```
CodTamano=int(input('Ingrese el Tamaño del zapato:'))
Cantidad=int(input('Ingrese la cantidad producida del tamaño del zapato:'))
```

```
while CodTamano!=0:
    AuxTamano=CodTamano
    while AuxTamano==CodTamano:
        ProduccionTamano=ProduccionTamano + 1
        CantTamano=CantTamano + Cantidad
        CodTamano=int(input('Ingrese el Tamaño del zapato:'))
        Cantidad=int(input('Ingrese la cantidad producida del tamaño del zapato:'))
```

```
Promedio=CantTamano/ProduccionTamano
print('El promedio de las compras es:', round(Promedio,2))
```

```
if CantTamano > 300:
    May300 = May300 + 1
```

```
Contar=Contar+1
```

Expresión de Problemas y Algoritmos

```

if Contar==1:
    MenorFab=AuxTamano
    MenorCant=CantTamano

if CantTamano < MenorCant:
    MenorFab=AuxTamano
    MenorCant=CantTamano

CantTamano=0
ProduccionTamano=0

print('La cantidad de zapatos con una producción mayor a 300 unidades son:',
round(May300,2))
print('Los zapatos que menos se fabricaron fueron del tamaño:', round(MenorFab,2), 'con
una cantidad :', round(MenorCant,2))

```

Ejercicio 51

```

LegaApeNom= ''
CodMateria=0
Nota=0
CantNotaMateria=0
AcumNotaMateria=0
PromedioMat=0
AcuPromAlum=0
CantMateriasAlum=0
PromAlum=0
CantAlumn=0
MayPromGral=0
MayAlumn=''
AcuGral=0
PromGral=0

LegaApeNom=input('Ingrese el Legajo/Apellido/Nombre:')
CodMateria=int(input('Ingrese el Código de La Materia:'))
Nota=int(input('Ingrese La Nota de La Materia:'))

while LegaApeNom != 'NN':
    AuxApellido=LegaApeNom
    AuxCodMateria=CodMateria
    while AuxCodMateria==CodMateria:
        CantNotaMateria=CantNotaMateria + 1
        AcumNotaMateria= AcumNotaMateria + Nota
        LegaApeNom=input('Ingrese el Legajo/Apellido/Nombre:')
        CodMateria=int(input('Ingrese el Código de La Materia:'))
        Nota=int(input('Ingrese La Nota de La Materia:'))

```

Expresión de Problemas y Algoritmos

```

PromedioMat=AcumNotaMateria/CantNotaMateria
print('El promedio del alumno:',AuxApellido,' de la materia:',AuxCodMateria,' es:',
round(PromedioMat,2))

AcuPromAlum=AcuPromAlum + PromedioMat
CantMateriasAlum=CantMateriasAlum +1
CantNotaMateria=0
AcumNotaMateria=0
PromAlum=AcuPromAlum/CantMateriasAlum
CantAlumn=CantAlumn +1

if CantAlumn==1:
    MayPromGral=PromAlum
    MayAlumn=AuxApellido

if PromAlum > MayPromGral:
    MayPromGral=PromAlum
    MayAlumn=AuxApellido

AcuGral= AcuGral + PromAlum
AcuPromAlum=0
CantMateriasAlum=0

PromGral= AcuGral /CantAlumn
print('El mayor promedio es:',round(MayPromGral,2),' y corresponde al alumno:',MayAlumn)
print('El promedio General es de:',round(PromGral,2))

```

Ejercicio 53

```

VEC1= []
VEC2=[]
elemento=0
i=0
j=0

for i in range(0,50,1):
    elemento=int(input('Ingrese un número:'))
    VEC1.append(elemento)
    VEC2.append(elemento*2)

for j in range(0,50,1):
    print('El valor del Vector1 en la posición ',j,' es:',VEC1[j], ' y en el Vector 2, es:',VEC2[j])

```

Expresión de Problemas y Algoritmos

Ejercicio 55

```

VEC1= []
VEC2=[]
i=0
j=0

for i in range(0,41,1):
    elemento=int(input('Ingrese un número:'))
    VEC1.append(elemento)

for j in range(0,41,1):
    VEC2.append(VEC1[40-j])

j=0
for j in range(0,41,1):
    print('El valor del Vector1 en la posición ',j,'es:',VEC1[j], ' y en el Vector 2, es:',VEC2[j])

```

Ejercicio 57

```

VEC1= []
VEC2=[]
i=0
j=0
Aux=0

for i in range(0,29,1):
    elemento=int(input('Ingrese un número:'))
    VEC1.append(elemento)
    VEC2.append(elemento)

i=0
for i in range(0,28,1):
    for j in range(i+1,29,1):
        if VEC2[i] < VEC2[j]:
            Aux=VEC2[i]
            VEC2[i]=VEC2[j]
            VEC2[j]=Aux

j=0
for j in range(0,29,1):
    print('El valor del Vector1 en la posición ',j,'es:',VEC1[j], ' y en el Vector 2, es:',VEC2[j])

```

Expresión de Problemas y Algoritmos

Ejercicio 59

```

VEC1=[]
VEC2=[]
i=0
j=0
Aux=0

for i in range(0,30,1):
    elemento=int(input('Ingrese un número:'))
    VEC1.append(elemento)
    VEC2.append(elemento)

inc=int(len(VEC2)/2)

while (inc>0):
    for i in range(inc,len(VEC2)):
        j=i
        Aux=VEC2[i]
        while ((j>=inc) and (VEC2[j-inc]>Aux)):
            VEC2[j]=VEC2[j-inc]
            j=j-inc
        VEC2[j]=Aux

    if (inc==2):
        inc=1
    else:
        inc=int(inc/2)

j=0

for j in range(0,30,1):
    print('El valor del Vector1 en la posición ',j,' es:',VEC1[j], ' y en el Vector 2, es:',VEC2[j])

```

Ejercicio 61

```

limite=0
contar=0
codigo=0
articulo=''
preciouni=0
cant=0
valor=0
Vcod=[]

```

Expresión de Problemas y Algoritmos

```

Vart=[]
Vpreuni=[]
Vcant=[]
Vvalor=[]
AuxVcod=0
AuxVart=''
AuxVpreuni=0
AuxVcant=0
AuxVvalor=0

limite=int(input('Ingrese La cantidad de productos de La venta (Limite 25):'))

while (limite<0) or (limite>25):
    limite=int(input('Ingrese La cantidad de productos de La venta (Limite 25):'))

while contar<limite:
    codigo=int(input('Ingrese el código del producto:'))
    articulo=input('Ingrese nombre del artículo:')
    preciouni=int(input('Ingrese el precio unitario del producto:'))
    cant=int(input('Ingrese La cantidad del producto:'))
    valor=preciouni * cant
    Vcod.append(codigo)
    Vart.append(articulo)
    Vpreuni.append(preciouni)
    Vcant.append(cant)
    Vvalor.append(valor)
    contar=contar+1

I=0
J=0

for I in range(0,contar-1,1):
    for J in range(I+1,contar,1):
        if Vvalor[I] < Vvalor[J]:
            AuxVcod=Vcod[I]
            AuxVart=Vart[I]
            AuxVpreuni=Vpreuni[I]
            AuxVcant=Vcant[I]
            AuxVvalor=Vvalor[I]
            Vcod[I]=Vcod[J]
            Vart[I]=Vart[J]
            Vpreuni[I]=Vpreuni[J]
            Vcant[I]=Vcant[J]
            Vvalor[I]=Vvalor[J]
            Vcod[J]=AuxVcod
            Vart[J]=AuxVart
            Vpreuni[J]=AuxVpreuni

```

Expresión de Problemas y Algoritmos

```
Vcant[J]=AuxVcant
Vvalor[J]=AuxVvalor

j=0
for j in range(0,len(Vcod),1):

    print('Codigo: ',Vcod[j],' Artículos: ',Vart[j],' Precio Unitario: ',Vart[j],'
Cantidad:', Vcant[j],'Valor Total: ',Vvalor[j])
```

Ejercicio 63

```
VEC1=[]
VEC2=[]
i=0
j=0
Aux=0
paso=0
izq=0
der=0
centro=0

for i in range(0,30,1):
    elemento=int(input('Ingrese un número:'))
    VEC1.append(elemento)
    VEC2.append(elemento)

I=0
J=0

for I in range(0,30,1):
    for J in range(I+1,30,1):
        if VEC2[I] < VEC2[J]:
            Aux=VEC2[I]
            VEC2[I]=VEC2[J]
            VEC2[J]=Aux

J=0

for J in range(0,30,1):
    print('El valor del Vector1 en la posición ',J,'es:',VEC1[J],' y en el Vector 2,
es:',VEC2[J])
```


Expresión de Problemas y Algoritmos

```

busquedaelemento=int(input('Ingrese el número a buscar:'))

paso=0
izq=0
der=len(VEC2)-1

while izq<=der:
    paso=paso+1
    centro=(izq+der) // 2
    if VEC2[centro]== busquedaelemento:
        print('Valor encontrado en {} pasos, y su posición es {}'.format(paso,centro))
        break
    if VEC2[centro] < busquedaelemento:
        der=centro-1
    if VEC2[centro] > busquedaelemento:
        izq=centro + 1

if VEC2[centro]==busquedaelemento:
    print('El dato:', busquedaelemento, ' se encuentra en la posición:',centro)
else:
    print('Dato no encontrado')

```

Ejercicio 65

```

VEC1=[]
VEC2=[]
i=0
j=0
Aux=0
paso=0
izq=0
der=0
centro=0

for i in range(0,10,1):
    elemento=int(input('Ingrese un número:'))
    VEC1.append(elemento)
    VEC2.append(elemento)

I=0
J=0

for I in range(0,10,1):

```

Expresión de Problemas y Algoritmos

```
for J in range(I+1,10,1):
    if VEC2[I] < VEC2[J]:
        Aux=VEC2[I]
        VEC2[I]=VEC2[J]
        VEC2[J]=Aux
```

J=0

```
for J in range(0,10,1):
    print('El valor del Vector1 en la posición ',J,'es:',VEC1[J], ' y en el Vector 2, es:', VEC2[J])
```

I=0

```
while I< (len(VEC2)-1):
    if VEC2[I]==VEC2[I+1]:
        del VEC2[I+1]
    else:
        I=I+1
```

J=0

```
for J in range(0,len(VEC2),1):
    print('El valor del Vector2 en la posición ',J,'es:',VEC2[J])
```

Ejercicio 67

Es un ordenamiento por inserción.

```
#Función para mostrar estado de la lista
def mostrarLista(lista, lon):
    listaordenada=""
    for i in range(0,lon):
        listaordenada=listaordenada + str(lista[i])+" "
    print(listaordenada)

arreglo = [47,3,21,32,56,92];
#Recorrer el arreglo
for i in range(1,len(arreglo)):
    clave = arreglo[i]
    j = i-1
    #Comparar el valor seleccionado con todos los valores anteriores
    while (j>=0 and arreglo[j] > clave):
        #Insertar el valor donde corresponda
        arreglo[j+1] = arreglo[j]
        j = j-1
    arreglo[j+1] = clave
```

Expresión de Problemas y Algoritmos

```
mostrarLista(arreglo, len(arreglo))
```

```
mostrarLista(arreglo, len(arreglo))
```

Ejercicio 69

```
lista = [8,43,17,6,40,16,18,97,11,7]
```

```
pivote=0
```

```
menores=[]
```

```
mayores=[]
```

```
def particionado(lista):
```

```
    pivote=lista[0]
```

```
    menores=[]
```

```
    mayores=[]
```

```
    for i in range(1,len(lista)):
```

```
        if lista[i]<pivote:
```

```
            menores.append(lista[i])
```

```
        else:
```

```
            mayores.append(lista[i])
```

```
    print(menores)
```

```
    print(mayores)
```

```
    return menores, pivote, mayores
```

```
print(particionado(lista))
```

```
def quicksort(lista):
```

```
    if len(lista) <2:
```

```
        return lista
```

```
    else:
```

```
        menores,pivote,mayores=particionado(lista)
```

```
    # aquí se aplica recursividad
```

```
    return quicksort(menores) + [pivote] + quicksort(mayores)
```

```
print(quicksort(lista))
```

Expresión de Problemas y Algoritmos

Ejercicio 71

```
VEC2=[8,13,17,26,44,56,88,97]
```

```
busquedaelemento=int(input('Ingrese el número a buscar:'))
```

```
paso=0
```

```
izq=0
```

```
der=len(VEC2)-1
```

```
while izq<=der:
```

```
    paso=paso+1
```

```
    centro=(izq+der) // 2
```

```
    print('Posición:',centro, 'Su contenido:', VEC2[centro],
```

```
    'Paso:',paso, 'Izquierda:',izq, 'Derecha:',der)
```

```
    if VEC2[centro]== busquedaelemento:
```

```
        print('Valor encontrado en {} pasos, y su posición es {}'.format(paso,centro))
        break
```

```
    if VEC2[centro] > busquedaelemento:
```

```
        der=centro-1
```

```
    if VEC2[centro] < busquedaelemento:
```

```
        izq=centro + 1
```

```
if VEC2[centro]==busquedaelemento:
```

```
    print('El dato:', busquedaelemento, ' se encuentra en la posición:',centro)
```

```
else:
```

```
    print('Dato no encontrado')
```

Ejercicio 73

Respuesta: Ver Ejercicio 63

Ejercicio 75

```
MAT1=[]
```

```
i=0
```

```
NumA=0
```

```
NumB=0
```

Expresión de Problemas y Algoritmos

```

NumC=0
Fila=0
Colum=0
VECMAXCOL=[]

for i in range(0,11,1):
    a=[0] * 9
    MAT1.append(a)

for Fila in range(0,8,1):
    VECMAXCOL.append(Fila)

NumA=int(input('Ingrese el número de fila (un número entre 0 y 9):'))
NumB=int(input('Ingrese el número de columna (un número entre 0 y 7):'))
NumC=int(input('Ingrese el dato de entrada(si ingresa 0 termina el proceso):'))

while NumC!=0:

    while NumA<0 and NumA>14:
        NumA=int(input('Ingrese el número de fila (un número entre 0 y 9):'))

    while NumB<0 and NumB>9:
        NumB=int(input('Ingrese el número de columna (un número entre 0 y 7):'))

    MAT1[NumA][NumB]=NumC

    NumA=int(input('Ingrese el número de fila (un número entre 0 y 9):'))
    NumB=int(input('Ingrese el número de columna (un número entre 0 y 7):'))
    NumC=int(input('Ingrese el dato de entrada(si ingresa 0 termina el proceso):'))

i=0
for i in range(0,10,1):
    print('El contenido de la fila',i, ' es:',MAT1[i])

Fila=0
for Fila in range(0,10,1):
    for Colum in range(0,9,1):
        MAT1[Fila][8] = MAT1[Fila][8] + MAT1[Fila][Colum]

Fila=0
for Fila in range(0,10,1):
    MAT1[Fila][8]=MAT1[Fila][8] /8

```

Expresión de Problemas y Algoritmos

```
Fila=0
for Fila in range(0,10,1):
    print('El promedio de la fila',Fila,' es:',MAT1[Fila][8])

Colum=0
Fila=0

for Colum in range(0,8,1):
    MAT1[10][Colum]= MAT1[0][Colum]
    for Fila in range(0,10,1):
        if MAT1[Fila][Colum]>MAT1[10][Colum]:
            MAT1[10][Colum]= MAT1[Fila][Colum]

Fila=0

for Fila in range(0,8,1):
    VECMAXCOL[Fila]=MAT1[10][Fila]

Fila=0
for Fila in range(0,8,1):

    print('El mayor valor de la columna',Fila,' es:',VECMAXCOL[Fila])
```

Ejercicio 77

```
Empre1=[]
i=0
NumA=0
NumB=0
NumC=0
Fila=0
Colum=0

for i in range(0,3,1):
    a=[0] * 11
    Empre1.append(a)

print('El fin del ingreso de Datos se produce con la categoría 3')
NumA=int(input('Ingrese la Categoría del Empleado (0-Jefe, 1-Analista,2-Programador):'))
NumB=int(input('Ingrese el número Área (un número entre 0 y 9):'))
NumC=int(input('Ingrese la cantidad de Horas Trabajadas:'))
```

Expresión de Problemas y Algoritmos

```

while NumA!=3:

    while NumA<0 and NumA>4:
        NumA=int(input('Ingrese La Categoría del Empleado (0-Jefe, 1-Analista,2-Programador):'))

    while NumB<0 and NumB>9:
        NumB=int(input('Ingrese el número Área (un número entre 0 y 9):'))

    Empre1[NumA][NumB]=NumC

    NumA=int(input('Ingrese La Categoría del Empleado (0-Jefe, 1-Analista,2-Programador):'))
    NumB=int(input('Ingrese el número Área (un número entre 0 y 9):'))
    NumC=int(input('Ingrese La cantidad de Horas Trabajadas:'))

Colum=0
Fila=0
for Fila in range(0,3,1):
    for Colum in range(0,10,1):
        print('El Total de La Categoría ',Fila,' del Área:',Colum,'
es:',Empre1[Fila][Colum])

Fila=0
Colum=0
for Fila in range(0,3,1):
    for Colum in range(0,9,1):
        Empre1[Fila][10] = Empre1[Fila][10] + Empre1[Fila][Colum]

Fila=0

for Fila in range(0,3,1):

    print('El Total de La Categoría ',Fila,' es:', Empre1[Fila][10])

```