Capítulo 1
Resolución de Problemas

Objetivos

La resolución de problemas, utilizando como herramienta una computadora, requiere contar con la capacidad de expresión suficiente como para indicar a la máquina lo que debe llevarse a cabo.

Se comenzará resolviendo situaciones del mundo real tratando de utilizar determinados elementos que caracterizan a una secuencia de órdenes, que como veremos, se pueden representar en una computadora.
El tema central de este capítulo es la definición del concepto de problema, algoritmo y los elementos que lo componen.

Temas a tratar

- Introducción.
- Etapas en la resolución de problemas con computadora.
- Algoritmo.
- Pre y Postcondiciones de un algoritmo.
- Elementos que componen un algoritmo: Secuencia de Acciones, Selección, Repetición e Iteración.
- Importancia de la indentación en las estructuras de control.
- Conclusiones.

1.1 Introducción

La Informática es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.

La palabra ciencia se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas. En este sentido la Informática se vincula especialmente con la Matemática.

Si se busca en el diccionario una definición en la palabra ***problema (****que proviene del griego y significa "lanzar hacia adelante")* podrá hallarse alguna de las siguientes, aunque existen muchas otras:

- Cuestión o proposición dudosa, que se trata de aclarar o resolver.
- Obstáculo arrojado ante la inteligencia para ser superado
- Enunciado encaminado a averiguar el modo de obtener un resultado cuando se conocen ciertos datos.

Todos vivimos resolviendo problemas: desde el más básico de asegurar la cotidiana subsistencia, común a todos los seres vivos, hasta los más complejos desafíos planteados por la ciencia y la tecnología. La importancia de la actividad de resolución de problemas es evidente; en definitiva, todo el progreso científico y tecnológico, el bienestar y hasta la supervivencia de la especie humana dependen de esta habilidad.

En los problemas no es siempre evidente el camino a seguir; incluso puede haber varios; y desde luego no está codificado y enseñado previamente. Hay que apelar a conocimientos dispersos, y no siempre de matemáticas; hay que relacionar saberes procedentes de campos diferentes, hay que poner a punto relaciones nuevas.
Pero además tiene que ser una cuestión que nos interese, que nos provoque las ganas de resolverla, una tarea a la que estemos dispuestos a dedicarle tiempo y esfuerzos. Como consecuencia de todo ello, una vez resuelta nos proporciona una sensación considerable de placer. E incluso, sin haber acabado el proceso, sin haber logrado la solución, también en el proceso de búsqueda, en los avances que vamos realizando, encontraremos una componente placentera.

Pautas a seguir en la resolución de un Problema:

George Pólya (1887 - 1985) formuló cuatro etapas esenciales para la resolución de un problema, que constituyen el punto de arranque de todos los estudios posteriores:

1. **COMPRENDER EL PROBLEMA**. Parece, a veces, innecesaria, sobre todo en contextos escolares; **pero es de una importancia capital**, sobre todo cuando los problemas a resolver no son de formulación estrictamente matemática. Es más, es la tarea más difícil, por ejemplo, cuando se ha de hacer un tratamiento informático: entender cuál es el problema que tenemos que abordar, dados los diferentes lenguajes que hablan el demandante y el informático.
   - Se debe leer el enunciado despacio.
   - ¿Cuáles son los datos? (lo que conocemos)
   - ¿Cuáles son las incógnitas? (lo que buscamos)
   - Hay que tratar de encontrar la relación entre los datos y las incógnitas.
   - Si se puede, se debe hacer un esquema o dibujo de la situación.
2. **TRAZAR UN PLAN PARA RESOLVERLO**. Hay que plantearla de una manera flexible y alejada del mecanicismo.
   - ¿Este problema es parecido a otros que ya conocemos?
   - ¿Se puede plantear el problema de otra forma?
   - Imaginar un problema parecido pero más sencillo.
   - Suponer que el problema ya está resuelto; ¿cómo se relaciona la situación de llegada con la de partida?
   - ¿Se utilizan todos los datos cuando se hace el plan?
3. **EJECUTAR EL PLAN**. Tener en cuenta que el pensamiento no es lineal, que hay saltos continuos entre el diseño del plan y su puesta en práctica.
   - Al ejecutar el plan se debe comprobar cada uno de los pasos.
   - ¿Se puede ver claramente que cada paso es correcto?
   - Antes de hacer algo se debe pensar: ¿qué se consigue con esto?
   - Se debe acompañar cada operación matemática de una explicación contando lo que se hace y para qué se hace.

- Cuando se tropieza con alguna dificultad que nos deja bloqueados, se debe volver al principio, reordenar las ideas y probar de nuevo.

4. **COMPROBAR LOS RESULTADOS**. Es la más importante en la vida diaria, porque supone la confrontación con contexto del resultado obtenido por el modelo del problema que hemos realizado, y su contraste con la realidad que queríamos resolver.

- Leer de nuevo el enunciado y comprobar que lo que se pedía es lo que se ha averiguado.
- Debemos fijarnos en la solución. ¿Parece lógicamente posible?
- ¿Se puede comprobar la solución?
- ¿Hay algún otro modo de resolver el problema?
- ¿Se puede hallar alguna otra solución?
- Se debe utilizar el resultado obtenido y el proceso seguido para formular y plantear nuevos problemas.

Si se piensa en la forma en que una persona indica a otra como resolver un problema, se verá que habitualmente se utiliza un lenguaje común y corriente para realizar la explicación, quizá entremezclado con algunas palabras técnicas. Es importante que la comunicación dada por el experto pueda ser entendida por el novato. Los que tienen cierta experiencia al respecto saben que es difícil transmitir el mensaje y por desgracia, con mucha frecuencia se malinterpretan las instrucciones y por lo tanto se ejecuta incorrectamente la solución obteniéndose errores.

Cuando intentamos resolver un problema utilizando una computadora veremos que las indicaciones no pueden ser ambiguas. Ante cada orden resulta fundamental tener una única interpretación de lo que hay que realizar. Una máquina no posee la capacidad de entendimiento y decisión del ser humano para resolver situaciones no previstas. Si al dar una orden a la computadora se produce una situación no contemplada, será necesario abortar esa tarea y recomenzar todo el procedimiento nuevamente.

Además, para poder indicar a la computadora las órdenes que debe realizar es necesario previamente entender exactamente lo que se quiere hacer. Es fundamental conocer con qué información se cuenta y qué tipo de transformación se quiere hacer sobre ella.

A continuación se analizarán en forma general las distintas etapas que deben seguirse para poder llegar a resolver un problema utilizando una computadora como herramienta.

1.2 Etapas en la resolución de problemas utilizando una computadora

La resolución de problemas utilizando como herramienta una computadora no se resume únicamente en la escritura de un programa, sino que se trata de una tarea más compleja. El proceso abarca todos los aspectos que van desde interpretar las necesidades del usuario hasta verificar que la respuesta brindada es correcta.
Las etapas son las siguientes:

Análisis del problema

En esta primera etapa, se analiza el problema en su contexto del mundo real. Deben obtenerse los requerimientos del usuario. El resultado de este análisis es un modelo preciso del ambiente del problema y del objetivo a resolver. Dos componentes importantes de este modelo son los datos a utilizar y las transformaciones de los mismos que llevan al objetivo.

Diseño de una solución
La resolución de un problema suele ser una tarea muy compleja para ser analizada como un todo. Una técnica de diseño en la resolución de problemas consiste en la identificación de las partes (subproblemas) que componen el problema y la manera en que se relacionan. Cada uno de estos subproblemas debe tener un objetivo específico, es decir, debe resolver una parte del problema original. La integración de las soluciones de los subproblemas es lo que permitirá obtener la solución buscada.

Especificación de algoritmos
La solución de cada subproblema debe ser especificada a través de un **algoritmo**. Esta etapa busca obtener la secuencia de pasos a seguir para resolver el problema. La elección del algoritmo adecuado es fundamental para garantizar la eficiencia de la solución.

Escritura de programas
Un algoritmo es una especificación simbólica que debe convertirse en un programa real sobre un lenguaje de programación concreto. A su vez, un programa escrito en un lenguaje de programación determinado (ej: Pascal, Ada, etc) es traducido automáticamente al lenguaje de máquina de la computadora que lo va a ejecutar. Esta traducción, denominada compilación, permite detectar y corregir los errores sintácticos que se cometan en la escritura del programa.

Verificación
Una vez que se tiene un programa escrito en un lenguaje de programación se debe verificar que su ejecución produce el resultado deseado, utilizando datos representativos del problema real. Sería deseable poder afirmar que el programa cumple con los objetivos para los cuales fue creado, más allá de los datos particulares de una ejecución. Sin embargo, en los casos reales es muy difícil realizar una verificación exhaustiva de todas las posibles condiciones de ejecución de un sistema de software. La facilidad de verificación y la depuración de errores de funcionamiento del programa conducen a una mejor calidad del sistema y es un objetivo central de la Ingeniería de Software.

En cada una de las etapas vistas se pueden detectar errores lo cual llevará a rever aspectos de la solución analizados previamente.

Dada la sencillez de los problemas a resolver en este curso, la primera etapa correspondiente al análisis del problema, sólo se verá reflejada en la interpretación del enunciado a resolver. Sin embargo, a lo largo de la carrera se presentaran diferentes asignaturas que permitirán familiarizar al alumno con las técnicas necesarias para hacer frente a problemas de gran envergadura.

Con respecto a la segunda etapa, se pospondrá el análisis de este tema hasta el capítulo 5, ya que se comenzará a trabajar con problemas simples que no necesitan ser descompuestos en otros más elementales.

Por lo tanto, a continuación se trabajará sobre el concepto de algoritmo como forma de especificar soluciones concretas para la computadora.

1.3 Algoritmo

La palabra algoritmo deriva del nombre de un matemático árabe del siglo IX, llamado Al-Khuwarizmi, quien estaba interesado en resolver ciertos problemas de aritmética y describió varios métodos para resolverlos. Estos métodos fueron presentados como una lista de instrucciones específicas (como una receta de cocina) y su nombre se utiliza para referirse a dichos métodos.

Un algoritmo es, en forma intuitiva, una receta, un conjunto de instrucciones o de especificaciones sobre un proceso para hacer algo. Ese algo generalmente es la solución de un problema de algún tipo. Se espera que un algoritmo tenga varias propiedades. La primera es que un algoritmo no debe ser ambiguo, o sea, que si se trabaja dentro de cierto marco o contexto, cada instrucción del algoritmo debe significar sólo una cosa.

Se presentan a continuación algunos ejemplos:

Ejemplo 1.1:
Problema : Indique la manera de salar una masa.
Malo : Ponerle algo de sal a la masa
Bueno: Agregarle una cucharadita de sal a la masa.

Ejemplo 1.2:
Problema: Determinar si el número 7317 es primo.
Malo: Divida el 7317 entre sus anteriores buscando aquellos que lo dividan exactamente. **Bueno**: Divida el numero 7317 entre cada uno de los números 2, 3, 4, ..., 7315, 7316. Si una de las divisiones es exacta, la respuesta es no. Si no es así, la respuesta es sí.

Esta es una solución no ambigua para este problema. Existen otros algoritmos mucho más eficaces para dicho problema, pero esta es una de las soluciones correctas.

Ejemplo 1.3:
Problema: Determinar la suma de todos los números enteros.

En este caso no se puede determinar un algoritmo para resolver este problema. Un algoritmo debe alcanzar la solución en un tiempo finito, situación que no se cumplirá en el ejemplo ya que los números enteros son infinitos.

Además de no ser ambiguo, un algoritmo debe detenerse. Se supone también que cuando se detiene, debe informar de alguna manera, su resultado. Es bastante factible escribir un conjunto de instrucciones que no incluyan una terminación y por lo tanto dicho conjunto de

instrucciones no conformarían un algoritmo.

Ejemplo 1.4:
Problema: Volcar un montículo de arena en una zanja.

Algoritmo: Tome una pala. Mientras haya arena en el montículo cargue la pala con arena y vuélquela en la zanja. Dejar la pala.

Este algoritmo es muy simple y no ambiguo. Se está seguro que en algún momento parará, aunque no se sabe cuántas paladas se requerirán.

Resumiendo, un algoritmo puede definirse como una secuencia ordenada de pasos elementales, exenta de ambigüedades, que lleva a la solución de un problema dado en un tiempo finito.

Para comprender totalmente la definición anterior falta clarificar que se entiende por "paso elemental".

Ejemplo 1.5:
Escriba un algoritmo que permita preparar una tortilla de papas de tres huevos.
El enunciado anterior basta para que un cocinero experto lo resuelva sin mayor nivel de detalle, pero si este no es el caso, se deben describir los pasos necesarios para realizar la preparación. Esta descripción puede ser:

Mezclar papas cocidas, huevos y una pizca de sal en un recipiente
Freír

Esto podría resolver el problema, si el procesador o ejecutor del mismo no fuera una persona que da sus primeros pasos en tareas culinarias, ya que el nivel de detalle del algoritmo presupone muchas cosas.

Si este problema debe resolverlo una persona que no sabe cocinar, se debe detallar, cada uno de los pasos mencionados, pues estos no son lo bastante simples para un principiante.

De esta forma, el primer paso puede descomponerse en:

Pelar las papas
Cortarlas en cuadraditos
Cocinar las papas
Batir los huevos en un recipiente
Agregar las papas al recipiente y echar una pizca de sal al mismo

El segundo paso (freír) puede descomponerse en los siguientes tres:

Calentar el aceite en la sartén
Verter el contenido del recipiente en la sartén

Dorar la tortilla de ambos lados

Nótese además que si la tortilla va a ser realizada por un niño, algunas tareas (por ejemplo batir los huevos) pueden necesitar una mejor especificación.

El ejemplo anterior sólo pretende mostrar que la lista de pasos elementales que compongan nuestro algoritmo depende de quién sea el encargado de ejecutarlo (Comunicación entre el experto y el novato)

Si en particular, el problema va a ser resuelto utilizando una computadora, el conjunto de pasos elementales conocidos es muy reducido, lo que implica un alto grado de detalle para los algoritmos.
Se considera entonces como un paso elemental aquel que no puede volver a ser dividido en otros más simples. De ahora en adelante se utiliza la palabra instrucción como sinónimo de paso elemental

Un aspecto importante a discutir es el detalle que debe llevar el algoritmo. Esto no debe confundirse con el concepto anterior de paso elemental. En ocasiones, no se trata de descomponer una orden en acciones más simples sino que se busca analizar cuáles son las **órdenes relevantes** para el problema. Esto resulta difícil de cuantificar cuando las soluciones son expresadas en lenguaje natural. Analice el siguiente ejemplo:
Se considera entonces como un paso elemental aquel que no puede volver a ser dividido en otros más simples. De ahora en adelante se utiliza la palabra instrucción como sinónimo de paso elemental
Ejemplo 1.6:
Desarrolle un algoritmo que describa la manera en que Ud. se levanta todas las mañanas para ir al trabajo.

Salir de la cama
Quitarse el pijama Ducharse
Vestirse
Desayunar
Arrancar el auto para ir al trabajo

Nótese que se ha llegado a la solución del problema en seis pasos, y no se resaltan aspectos como: colocarse los zapatos después de salir de la cama, o abrir la llave de la ducha antes de ducharse. Estos aspectos han sido descartados, pues no tienen mayor trascendencia. En otras palabras se sobreentienden o se suponen. A nadie se le ocurriría ir a trabajar descalzo.

En cambio existen aspectos que no pueden obviarse o suponerse porque el algoritmo perdería lógica. El tercer paso, “vestirse", no puede ser omitido. Puede discutirse si requiere un mayor nivel de detalle o no, pero no puede ser eliminado del algoritmo.

Un buen desarrollador de algoritmos deberá reconocer esos aspectos importantes y tratar de simplificar al mínimo su especificación de manera de seguir resolviendo el problema con la menor

cantidad de órdenes posibles.

1.4 Pre y Postcondiciones de un algoritmo

Precondición es la información que se conoce como verdadera antes de comenzar el algoritmo.

En el ejemplo 1.1:
Problema: Indique la manera de salar una masa.
Algoritmo: Agregarle una cucharadita de sal a la masa.

Se supone que se dispone de todos los elementos para llevar a cabo esta tarea. Por lo tanto, como precondición puede afirmarse que se cuenta con la cucharita, la sal y la masa.

Precondición es la información que se conoce como verdadera antes de comenzar el algoritmo.

Postcondición es la información que se conoce como verdadera al concluir el algoritmo si se cumple adecuadamente el requerimiento pedido.

En el ejemplo 1.2:
Problema: Determinar si el numero 7317 es primo.
Algoritmo: Divida el numero 7317 entre cada uno de los números 2, 3, 4, ..., 7315, 7316. Si una de las divisiones es exacta, la respuesta es no. Si no es así, la respuesta es sí.
La postcondición es que se ha podido determinar si el numero 7317 es primo o no.

En el ejemplo 1.4:
Problema: Volcar un montículo de arena en una zanja.
Algoritmo: Tome una pala. Mientras haya arena en el montículo cargue la pala con arena y vuélquela en la zanja. Dejar la pala.

- ¿Cuáles serían las precondiciones y las postcondiciones del algoritmo?

La precondición es que se cuenta con la pala, la arena, un montículo con arena y está ubicado cerca de la zanja que debe llenar.
La postcondición es que el montículo quedó vació al terminar el algoritmo.

1.5 Elementos que componen un algoritmo

1.5.1 Secuencia de Acciones

Una secuencia de acciones está formada por una serie de instrucciones que se ejecutan una a continuación de otra.

Esto se muestra gráficamente en la figura 1.1

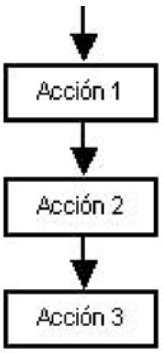

Figura 1.1: Secuencia

Ejemplo 1.7: Escriba un algoritmo que permita cambiar una lámpara quemada.

Colocar la escalera debajo de la lámpara quemada
Tomar una lámpara nueva de la misma potencia que la anterior
Subir por la escalera con la nueva lámpara hasta alcanzar la lámpara a sustituir
Desenroscar la lámpara quemada
Enroscar la nueva lámpara hasta que quede apretada la nueva lámpara
Bajar de la escalera con lámpara quemada
Tirar la lámpara a la basura

Ejemplo 1.8: Escriba un algoritmo que permita a un robot subir 8 escalones

Levantar Pie Izquierdo
Subir un escalón
Levantar Pie Derecho
Subir un escalón
Levantar Pie Izquierdo
Subir un escalón
Levantar Pie Derecho
Subir un escalón
Levantar Pie Izquierdo
Subir un escalón
Levantar Pie Derecho
Subir un escalón
Levantar Pie Izquierdo
Subir escalón
Levantar Pie Derecho
Subir un escalón

Se denomina flujo de control de un algoritmo al orden en el cual deben ejecutarse pasos individuales

Hasta ahora se ha trabajado con flujo de control secuencial, o sea, la ejecución uno a uno de pasos, desde el primero hasta el último.

Las estructuras de control son construcciones algorítmicas que alteran directamente el flujo de control secuencial del algoritmo.

Con ellas es posible seleccionar un determinado sentido de acción entre un par de alternativas específicas o repetir automáticamente un grupo de instrucciones.

A continuación se presentan las estructuras de control necesarias para la resolución de problemas más complejos.

1.5.2 Selección

La escritura de soluciones a través de una secuencia de órdenes requiere conocer a priori las diferentes alternativas que se presentarán en la resolución del problema. Lamentablemente, es imposible contar con esta información antes de comenzar la ejecución de la secuencia de acciones.

Por ejemplo, que ocurriría si en el ejemplo 1.7 al querer sacar la lámpara quemada, el portalámpara se rompe. Esto implica que el resto de las acciones no podrán llevarse a cabo por lo que el algoritmo deberá ser interrumpido. Si se desea que esto no ocurra, el algoritmo deberá contemplar esta situación. Nótese que el estado del portalámpara es desconocido al iniciar el proceso y sólo es detectado al intentar sacar la lámpara quemada. Por lo que el solo uso de la secuencia es insuficiente para expresar esta solución.

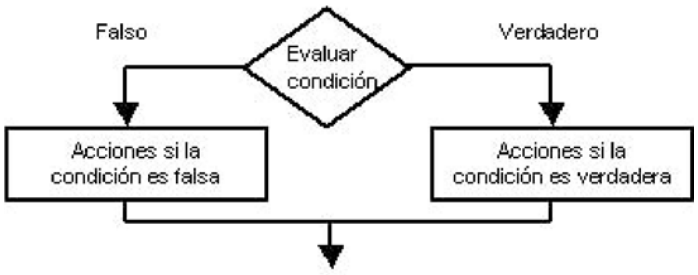

Figura 1.2: Estructura Si-Entonces-Sino

A través de la selección se incorpora, a la especificación del algoritmo, la capacidad de decisión. De esta forma será posible seleccionar una de dos alternativas de acción posibles durante la ejecución del algoritmo.

Por lo tanto, el algoritmo debe considerar las dos alternativas, es decir, qué hacer en cada uno de los casos. La selección se notará de la siguiente forma:

si (condición)
acción o acciones a realizar si la condición es verdadera (1)

sino
acción o acciones a realizar si la condición es falsa (2)

donde **condición** es una expresión que al ser evaluada puede tomar solamente uno de dos valores posibles: **verdadero o falso.**

El esquema anterior representa que en caso de que la condición a evaluar resulte verdadera se ejecutarán las acciones de (1) y NO se ejecutarán las de (2). En caso contrario, es decir si la condición resulta ser falsa, solo se ejecutarán las acciones de (2).

En la figura 1.2 se grafica la selección utilizando un rombo para representar la decisión y un rectángulo para representar un bloque de acciones secuenciales.

Analice el siguiente ejemplo:

Ejemplo 1.9: Su amigo le ha pedido que le compre $1 de caramelos en el kiosco. De ser posible, prefiere que sean de menta pero si no hay, le da igual que sean de cualquier otro tipo. Escriba un algoritmo que represente esta situación.

Ir al kiosco
si (hay caramelos de menta)
Llevar caramelos de menta (1)
sino
Llevar de cualquier otro tipo (2)
Pagar 1 peso

Los aspectos más importantes son:

Figura 1.3: Estructura Si-Entonces

- No es posible saber si en el kiosco hay o no hay caramelos de menta ANTES de llegar al kiosco por lo que no puede utilizarse únicamente una secuencia de acciones para resolver este problema.
- La condición "hay caramelos de menta" sólo admite dos respuestas posibles: hay o no hay; es decir, verdadero o falso respectivamente.
- Si se ejecuta la instrucción marcada con (1), NO se ejecutará la acción (2) y viceversa.
- Independientemente del tipo de caramelos que haya comprado, siempre se pagará $1. Esta acción es independiente del tipo de caramelos que haya llevado.

En algunos casos puede no haber una acción específica a realizar si la condición es falsa. En ese caso se utilizará la siguiente notación:

si (condición)
acción o acciones a realizar en caso de que la condición sea verdadera.

(Esto se muestra gráficamente en la figura 1.3.)

Ejemplos 1.9: Su amigo se ha puesto un poco más exigente y ahora le ha pedido que le compre $1 de caramelos de menta en el kiosco. Si no consigue caramelos de menta, no debe comprar nada.

Escriba un algoritmo que represente esta situación.

Ir al kiosco
si (hay caramelos de menta)
Pedir caramelos de menta correspondientes a $1
Pagar $1

Con este último algoritmo, a diferencia del ejemplo 1.8, si la condición "hay caramelos de menta" resulta ser falsa, no se realizará ninguna acción.

1.5.3 Repetición

Un componente esencial de los algoritmos es la repetición. La computadora, a diferencia de los humanos, posee una alta velocidad de procesamiento. A través de ella, es posible ejecutar, de manera repetitiva, algunos pasos elementales de un algoritmo. Esto puede considerarse una extensión natural de la secuencia.

La repetición es la estructura de control que permite al algoritmo ejecutar un conjunto de instrucciones un número de veces fijo y conocido de antemano.

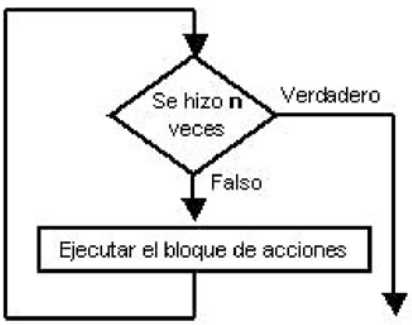

Figura 1.4: Estructura repetitiva

La notación a utilizar es la siguiente:

repetir N
    Acción o acciones a realizar N veces.

y se muestra gráficamente en la figura 1.4.

Se analizan a continuación algunos algoritmos que tienen repeticiones.

Ejemplo 1.11: Escriba un algoritmo que permita poner 4 litros de agua en un balde utilizando un vaso de 50 cc.

Se observa que hay dos pasos básicos: llenar el vaso con agua y vaciarlo en el balde. Para completar los cuatro litros es necesario repetir estas dos operaciones ochenta veces. Suponga que se dispone de un vaso, un balde y una canilla para cargar el vaso con agua.

Tomar el vaso y el balde
repetir 80
    Llenar el vaso de agua.
    Vaciar el vaso en el balde.
Dejar el vaso y el balde.

Nótese que, la instrucción "Dejar el vaso y el balde" no pertenece a la repetición. Esto queda indicado por la sangría o indentación utilizada para cada instrucción. Por lo tanto, se repetirán 80 veces las instrucciones de "Llenar el vaso de agua" y "Vaciar el vaso en el balde".

El ejemplo 1.8, que inicialmente se presento como un ejemplo de secuencia, puede escribirse utilizando una repetición de la siguiente forma:

Ejemplo 1.12: Escriba un algoritmo que permita a un robot subir 8 escalones.

repetir 4
    LevantaPieIzquierdo
    Subir un escalón.
    LevantaPieDerecho
    Subir un escalón

Este algoritmo realiza exactamente las mismas acciones que el algoritmo del ejemplo 1.8. Las ventajas de utilizar la repetición en lugar de la secuencia son: la reducción de la longitud del código y la facilidad de lectura.

Ejemplo 1.13: Juan y su amigo quieren correr una carrera dando la vuelta a la manzana. Considerando que Juan vive en una esquina, escriba el algoritmo correspondiente.

repetir 4
Correr una cuadra
Doblar a la derecha

1.5.4 Iteración

Existen situaciones en las que **se desconoce el número de veces** que debe repetirse un conjunto de acciones. Por ejemplo, si se quiere llenar una zanja con arena utilizando una pala, será difícil indicar exactamente cuántas paladas de arena serán necesarias para realizar esta tarea. Sin embargo, se trata claramente de un proceso iterativo que consiste en cargar la pala y vaciarla en la zanja.

Por lo tanto, dentro de una iteración, además de una serie de pasos elementales que se repiten; es necesario contar con un mecanismo que lo detenga.

La iteración es una estructura de control que permite al algoritmo ejecutar en forma repetitiva un conjunto de acciones utilizando una condición para indicar su finalización.

El esquema iterativo es de la forma:

Figura 1.5: Estructura iterativa

mientras (condición)
Acción o acciones a realizar en caso de que la condición sea verdadera.

Las acciones contenidas en la iteración serán ejecutadas mientras la condición sea verdadera. Es importante notar que, la primera vez, antes de ejecutar alguna de las acciones de la iteración, lo primero que se realiza es la evaluación de la condición. Sólo luego de comprobar que es verdadera se procede a ejecutar el conjunto de acciones pertenecientes al mientras.

Si inicialmente la condición resultara falsa, el contenido del mientras no se ejecutará ni siquiera una sola vez.

Este funcionamiento se muestra gráficamente en la figura 1.5.

Es importante que las acciones realizadas en el interior de la iteración modifiquen el valor de verdad de la condición a fin de garantizar que la iteración terminará en algún momento.

Analicemos el siguiente ejemplo:

Ejemplo 1.14: Escriba un algoritmo que permita volcar un montículo de arena en una zanja utilizando una pala.

```
Tomar la pala.
Ubicarse frente a la zanja.
mientras (no esté vacío el montículo de arena)
        cargar la pala con arena
        volcar la arena en la zanja
Dejar la pala.
```

La iteración indica que, mientras no se vacíe el montículo, se seguirá incorporando arena en la zanja. Cuando el montículo esté vacío, la condición será falsa y la iteración terminará. Es importante destacar, que si el montículo inicialmente estaba vacío, ninguna palada de arena será tomada del montículo ni incorporada a la zanja. Es decir, la condición se verifica ANTES de comenzar la iteración. Además no se sabe cuántas palas deberán cargarse para llenar el montículo.

La iteración es un proceso fundamental en los algoritmos, y se debe ser capaz de pensar en términos de ciclos de iteración para poder construir los algoritmos.

1.6 Importancia de la identación en las estructuras de control

Las instrucciones que pertenecen a una estructura de control deben tener una sangría mayor que la utilizada para escribir el comienzo de la estructura. De esta forma, podrá identificarse donde termina el conjunto de instrucciones involucradas. A esta sangría se la denomina identación.

Este concepto se aplica a las tres estructuras de control vistas previamente: selección, repetición e iteración.

El siguiente ejemplo muestra el uso de la identación en la selección:

Ejemplo 1.15: Suponga que se planea una salida con amigos. La salida depende del clima: si llueve vos y tus amigos irán al cine a ver la película elegida, por el contrario si no llueve irán de pesca. Luego de realizar el paseo se juntarán a comentar la experiencia vivida. Escriba el algoritmo que resuelva esta situación.

```
Juntarse en una casa con el grupo de amigos
Mirar el estado del tiempo.
si (llueve)                                (1)
        elegir película
        ir al cine
```

```
sino
        preparar el equipo de pesca
        ir a la laguna a pescar
Volver a la casa a comentar sobre el paseo     (2)
```

Como puede apreciarse, las acciones que deben ser realizadas cuando la condición es verdadera se encuentran desplazadas un poco más a la derecha que el resto de la estructura. Algo similar ocurre con las acciones a realizar cuando la condición es falsa. De esta forma puede diferenciarse lo que pertenece a la selección del resto de las instrucciones.

En el ejemplo anterior, la instrucción "Volver a la casa a comentar sobre el paseo" se realiza siempre sin importar si llovió o no. Esto se debe a que no pertenece a la selección. Esto queda de manifiesto al darle a las instrucciones (1) y (2) la misma indentación.

Ejemplo 1.16: Ud. desea ordenar una caja con 54 fotografías viejas de manera que todas queden al derecho; esto es, en la orientación correcta y la imagen boca arriba. Las fotografías ordenadas se irán guardando en el álbum familiar. Escriba el algoritmo que le permita resolver este problema.

```
Tomar la caja de fotos y un álbum vacío.
repetir 54
        Tomar una fotografía.
        si (la foto está boca abajo)
                dar vuelta la foto
        si (la foto no está en la orientación correcta)
                girar la foto para que quede en la orientación correcta
        guardar la fotografía en el álbum
guardar el álbum
```

Según la identación utilizada, la repetición contiene a la acción de "Tomar una fotografía", ambas selecciones y la instrucción "guardar la fotografía en el álbum". Las instrucciones "Tomar la caja de fotos y el álbum" y "Guardar el álbum" no pertenecen a la repetición.

Ejemplo 1.17: Ud. se dispone a tomar una taza de café con leche pero previamente debe endulzarlo utilizando azúcar en sobrecitos. Escriba un algoritmo que resuelva este problema.

```
Tomar la taza de café con leche.
mientras (no esté lo suficientemente dulce el café)
        Tomar un sobre de azúcar.
        Vaciar el contenido del sobre en la taza.
        Mezclar para que el azúcar se disuelva.
Tomar el café con  leche.
```

Note que en este último ejemplo no se conoce de antemano la cantidad de sobrecitos de azúcar necesarios para endulzar el contenido de la taza. Además, la condición se evalúa antes de agregar el primer sobre. Según indentación utilizada, la iteración incluye tres instrucciones. La acción "Tomar el café con leche" se ejecutará sólo cuando la iteración haya terminado, es decir,

cuando la condición sea falsa.

1.7 Conclusiones

El uso de algoritmos permite expresar, de una forma clara, la manera en que un problema debe ser resuelto. Los elementos que lo componen son característicos de la con computadora.

La ejercitación es la única herramienta para poder comprender y descubrir la verdadera potencialidad de las estructuras de control. Resulta fundamental alcanzar un total entendimiento del funcionamiento de estas estructuras para poder lograr expresar soluciones más complejas que los ejemplos aquí planteados.

Nota: para realizar este apunte se utilizó como bibliografía básica el material del curso de Ingreso a la Facultad de Informática de la UNLP. En el mismo se han realizado algunos agregados y modificaciones.