UTN
INSPT

MODULO 0

Resolución de problemas

Breve descripción

Este documento detalla qué es la resolución de problemas y las metodologías para utilizar en la asignatura.

Versión 1.1

Martin Jerman
Martin.jerman@inspt.utn.edu.ar

Introducción

En todo ámbito de la vida uno se encuentra con problemas que tiene que resolver. De hecho, un problema lleva a otro con lo que podríamos decir que "la vida es una sucesión de problemas a resolver". En las siguientes líneas desarrollaremos los pasos para la resolución de problemas y algunos tips.

¿Qué entendemos por "resolución de problemas"?

La resolución de problemas es la fase que supone la conclusión de un proceso más amplio que tiene como pasos previos la identificación del problema y su modelado. Ahora bien, ¿qué entendemos por "problema"? Por problema se entiende un asunto del que se espera una solución que dista de ser obvia a partir del planteamiento inicial.

¿Qué es un problema formal?

Un problema existe cuando se satisfacen las siguientes 4 condiciones:

- Una situación inicial dada, claramente definida;
- Una situación final deseada, claramente definida;
- Un conjunto de recursos claramente definido que ayude a llegar a la situación final;
- Un compromiso o "involucramiento" del individuo por llegar a dicha situación final.

Lo cual, en síntesis, un problema es una situación a resolver con un objetivo no alcanzable inmediatamente y que tiene 4 componentes:

- Datos y suposiciones;
- Objetivos, metas, resultados;
- Recursos (inteligencia creativa, herramientas físicas o cognitivas, conocimiento acumulado educación, entrenamiento y tiempo);
- Deseo o necesidad de resolverlo.

Los problemas se pueden clasificar en dos tipos diferentes para su resolución:

- Problemas mal o poco definidos: son aquellos que no tienen objetivos claros o caminos evidentes de solución.
- Problemas bien definidos: tienen objetivos específicos y caminos de solución claramente definidos.

En el caso de la programación, ambos tipos de problemas son factibles. Dentro del ámbito académico los enunciados de problemas tienden a ser lo mas definidos y claros posible para evitar malas interpretaciones y forzar al estudiante a aplicar determinado conocimiento. Sin embargo, en la vida real del programador, los problemas se presentan como poco definidos.

Por otra parte, dado un problema (sea del tipo que sea) puede tener más de una solución o forma de resolución. Dado que un mismo problema puede ser resuelto por distintas personas, cada persona ideará su solución que seguramente diferirá de otras personas. Claro que una solución puede ser mas eficiente que otra o más avanzada, pero mientras el problema se resuelva será una solución valida.

Pasos para la resolución de problemas

La resolución de un problema se puede dividir en los siguientes pasos:

1. **Análisis del problema:** primer paso para encontrar la solución a un problema es el análisis de este. ¿Qué se necesita o qué se pide?
2. **Estudio de su solución:** Se debe examinar cuidadosamente el problema a fin de obtener una idea clara sobre lo que se solicita y determinar los datos necesarios para conseguirlo, así como también las relaciones entre esos datos.

Un algoritmo es un conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.

3. **Diseño del Algoritmo:** Un algoritmo puede ser definido como la secuencia ordenada de pasos, sin ambigüedades, que conducen a la resolución de un problema dado y expresado en lenguaje natural, por ejemplo, el castellano. Todo algoritmo debe ser:
    a. Preciso: Indicando el orden de realización de cada uno de los pasos.
    b. Definido: Si se sigue el algoritmo varias veces proporcionándole (consistente) los mismos datos, se deben obtener siempre los mismos resultados.
    c. Finito: Al seguir el algoritmo, este debe terminar en algún momento, es decir tener un número finito de pasos.

Pongamos un ejemplo: Un mesero novato de la cafetería Tortoni necesita preparar un café con leche y no sabe cómo. ¿Podríamos enseñarle? Apliquemos los pasos anteriores:

1. **Paso 1 (análisis):** ayudado por la pregunta "qué hay que hacer?" y descartando el contexto del problema, tenemos el objetivo del problema: Preparar un café con leche.
2. **Paso 2 (estudio):** Para poder preparar el café con leche, estudiamos el caso utilizando el contexto del problema. Estando en la cafetería necesitaremos una taza vacía, conseguir el agua caliente, el café y la leche y mezclarlos de alguna manera.
3. **Paso 3 (algoritmo):** Ahora que sabemos qué hacer y una idea de cómo, pasamos a diseñar el algoritmo, que es una sucesión de pasos:
    a. Tomar una taza limpia de la lacena
    b. Con una cuchara de café, agregar 2 cucharadas de café instantáneo
    c. Con la cuchara de la azucarera, agregar 2 cucharadas de azúcar
    d. Verter un chorro de agua caliente en la taza
    e. Revolver la mezcla durante unos segundos
    f. Agregar el resto del agua caliente hasta un 80% de la taza
    g. Agregar la leche caliente hasta llenar la taza
    h. Servir el café preparado al comensal

Como se ve en este ejemplo, se paso por los tres pasos. Suena algo obvio lo que se hizo dada la trivialidad del problema, pero sirve para ilustrar el procedimiento. Notar que en el paso 1 se identifica el problema y cual es el objetivo; en el paso 2 se esboza la resolución y en el tercero la resolución.

Habrán notado que el paso 3 es muy detallado. El nivel de detalle dependerá de los requerimientos del problema, pero siempre mientras mas detalle haya, mejor. Llevando esto al mundo de la programación, verán que el nivel de detalle dependerá de la complejidad del problema y de las herramientas que se dispongan. De momento quedémonos con **tener en cuenta los detalles**.

Por otra parte, si se presta atencion, el algoritmo descrito en el paso 3 tiene una forma particular. Los pasos A, B, C y D agregan ingredientes a la taza. Estos ingredientes son la **entrada** de nuestro algoritmo para poder preparar el café.Luego procesamos esos ingredientes en la taza durante el paso E, F y G. Esta es la etapa del **proceso**. Finalmente tras terminar la preparacion, se entrega la **salida** del algoritmo.

Con esto vemos que todo algoritmo tiene 3 partes fundamentales:

- Entrada: son las lineas donde se ingresan los datos
- Proceso: son las lineas que procesan los datos para obtener resultados
- Salida: son las lineas que muestran los resultados al usuario o a otro proceso

Para identificar qué será la entrada y qué será la salida, podemos ayudarnos con algunas preguntas.

Entrada	**Salida**
¿Qué datos son de entrada?	¿Cuáles son los datos de salida?
¿Cuántos datos se introducirán?	¿Cuántos datos de salida se producirán?
¿Cuántos son datos de entrada válidos?	¿Qué formato y precisión tendrán los resultados?

Veremos durante todo el curso que este patron de Entrada-Proceso-Salida se respeta en casi todos los casos.

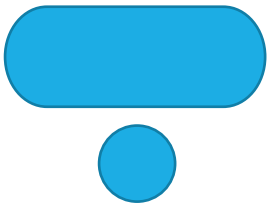
Elementos De Programación

Existen dos herramientas de programación usadas como Lenguajes algorítmicos, para representar un algoritmo: Diagramas de Flujo y Pseudocódigo. El lenguaje algorítmico debe ser independiente de cualquier lenguaje de programación particular, pero fácilmente traducible a cada uno de ellos. Alcanzar estos objetivos conducirá al empleo de métodos normalizados para la representación de algoritmos.

Diagramas De Flujo

También conocidos como flowchart, en estos se utilizan símbolos estándar, en el que cada paso para la elaboración del programa se representa con el símbolo y orden adecuados, unidos o conectados por flechas, también llamadas líneas de flujo, esto porque indican el sentido en el que se mueve el proceso. En resumen, el diagrama de flujo es un medio de presentación visual y gráfica del flujo de datos a través de un algoritmo.

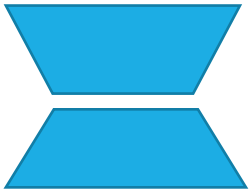
Bloques terminales



Bloques de inicio o fin de programa: Indican los límites del procedimiento considerado como principal. Generalmente se trata de un programa completo o de un módulo funcionalmente autónomo.

Bloques de acción

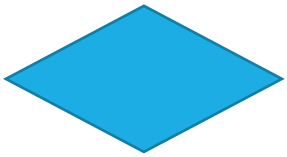
Bloque de acción simple: Representa una acción sencilla que puede ser considerada como única y que generalmente se codifica con una sola instrucción. Por ejemplo: incrementar contador, ubicar cursor, abrir archivo, etc.



Bloque de entrada/salida: Representa una acción simple de entrada o salida de datos, generalmente desde o hacia un dispositivo periférico como el teclado, la pantalla o el disco. Por ejemplo: ingresar valor, leer registro, mostrar resultado, etc.

Bloque de procedimiento: Representa un conjunto de acciones que se consideran juntas, sin analizar su detalle. Este grupo de acciones se describe generalmente como procedimiento en otra parte del diagrama. Por ejemplo: buscar elemento, ordenar conjunto, procesar dato, etc.

Bloques de decisión

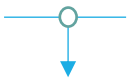


Bloque de decisión simple: Representa la acción de analizar el valor de verdad de una condición, que sólo puede ser verdadera o falsa (selección simple). Según el resultado de esta evaluación se sigue uno u otro curso de acción. Por lo tanto, de un bloque de decisión simple siempre salen exactamente dos flujos, uno por V (sí) y otro por F (no).

Bloque de decisión múltiple: Representa la acción de analizar el valor de una variable, que puede tomar uno entre una serie de valores conocidos (selección múltiple). Según el resultado de esta evaluación, se sigue uno entre varios cursos de acción. Por lo tanto, de un bloque de decisión múltiple siempre salen varios flujos, uno por cada valor esperado de la variable analizada.

Flujos y conectores

Flecha o flujo: Indica la secuencia en que se van ejecutando las acciones al pasar de un bloque a otro.

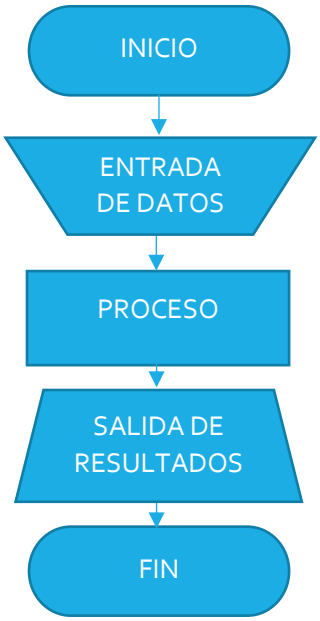


Conector: Indica la convergencia de dos o más flujos. En la práctica determina el comienzo o el fin de una estructura.

Veamos a continuación un conjunto de reglas o normas que nos permiten construir un diagrama de flujo:

1. Todo diagrama de flujo debe tener un inicio y un fin.
2. Las líneas utilizadas para indicar la dirección del diagrama deben ser rectas, horizontales o verticales, nunca se deben cruzar entre sí.
3. No debe haber líneas sin conexión a los demás elementos del diagrama de flujo.
4. Un diagrama de flujo se debe construir de arriba hacia abajo y de requerirse de izquierda a derecha.
5. La notación o símbolos utilizados en el diagrama de flujo son independientes del lenguaje de programación utilizado para la elaboración del programa o aplicación.
6. No puede llegar más de una línea de conexión a un símbolo.

Un ejemplo de diagrama de flujo:



Pseudocódigo

La formalización de las acciones se realiza a través de la estructura del algoritmo en pseudocódigo. El pseudocódigo se considera una herramienta para el diseño que permite obtener una solución mediante aproximaciones sucesivas. Se denomina notación de pseudocódigo a aquella que permite describir la solución de un problema en forma de algoritmo dirigido al computador utilizando palabras y frases del lenguaje natural sujetas a determinadas reglas.

Por ejemplo, sumar valores ingresados por teclado:

```
comienza
int a, b, c
escribe "De un valor entero"
lee a
escribe "de otro valor entero"
lee b
c ← a + b
escribe "La suma de"
escribe a
escribe "con"
escribe b
escribe "es"
escribe c
termina
```