Introducción a la Programación

Algoritmos y Estructuras de Datos I

Primer cuatrimestre de 2023

Solucionando problemas con una computadora

Introducción a la Programación - AED I

Objetivo: Aprender a programar en lenguajes funcionales y en lenguajes imperativos.

- Especificar problemas.
  - Describirlos en un lenguaje semiformal.
- Pensar algoritmos para resolver los problemas.
  - En esta materia nos concentramos en programas para tratamiento de secuencias principalmente.
- Empezar a Razonar acerca de estos algoritmos y programas.
  - Veremos conceptos de testing.
  - Veremos nociones de complejidad.

IP - AED I: Régimen de aprobación

- Con nota numérica
  - 1 parcial integrador con su recuperatorio (al final de la cursada)
  - 1 trabajo práctico grupal (4 integrantes)
- Aprobado / No aprobado
  - 3 trabajos prácticos individuales
  - Consistirán en programar ejercicios y subirlos a una plataforma (CMS)
  - TP 1: funcional (no lo tienen que entrgar los estudiantes que aprobaron el Taller de Álgebra I)
  - TP 2 y TP 3: python
- Régimen de promoción
  - Todas las instancias de evaluación deben estar aprobadas
  - Si el promedio de notas general es mayor o igual a 8: promoción directa
  - Si el promedio de notas general está entre 6 y 8: coloquio (sólo válido en el cuatrimestre en curso)
  - Si el promedio de notas general está entre 4 y 6: final escrito (tradicional)

¿Qué es una computadora?

- Una **computadora** es una máquina que procesa información automáticamente de acuerdo con un programa almacenado.

  1. Es una máquina.
  2. Su función es procesar información, y estos términos deben entenderse en sentido amplio.
  3. El procesamiento se realiza en forma automática.
  4. El procesamiento se realiza siguiendo un programa.
  5. Este programa está almacenado en una memoria interna de la misma computadora.

¿Qué es un algoritmo?

- Un **algoritmo** es la descripción de los pasos precisos para resolver un problema a partir de datos de entrada adecuados.

  1. Es la descripción de los pasos a dar.
  2. Especifica una sucesión de pasos primitivos.
  3. El objetivo es resolver un problema.
  4. Un algoritmo típicamente trabaja a partir de datos de entrada.

Ejemplo: Un Algoritmo

- **Problema:** Encontrar todos los números primos menores que un número natural dado n
- **Algoritmo:** Criba de Eratóstenes (276 AC - 194 AC)
  [1.] Escriba todos los números naturales desde 2 hasta a n
  [2.] Para $i \in \mathbb{Z}$ desde 2 hasta $\lfloor \sqrt{n} \rfloor$
  Si i no ha sido marcado,
  Entonces Para $j \in \mathbb{Z}$ desde i hasta $\lfloor \frac{n}{i} \rfloor$ haga lo siguiente:
  Si no ha sido marcado, marcar el número $i \times j$

¿Qué es un programa?

- Un **programa** es la descripción de un algoritmo en un lenguaje de programación.

    1. Corresponde a la implementación concreta del algoritmo para ser ejecutado en una computadora.
    2. Se describe en un lenguaje de programación.

Ejemplo: Un Programa (en Haskell)

Implementación de la Criba de Eratóstenes en el lenguaje de programación Haskell

```
erastotenes ::  Int -> [Int]
erastotenes n = erastotenes_aux [x|x <- [2..n]] 0

erastotenes_aux :: [Int] -> Int -> [Int]
erastotenes_aux lista n
  | n == length lista-1 = lista
  | otherwise = erastotenes_aux lista_filtrada (n+1)
    where lista_filtrada = [x|x <-lista ,(x `mod` lista!!n)/=0 ||
                                          x==lista!!n]
```

Especificación, algoritmo, programa

1. **Especificación:** descripción del problema a resolver.
   - ¿Qué problema tenemos?
   - Habitualmente, dada en lenguaje formal.
   - Es un contrato que da las propiedades de los datos de entrada y las propiedades de la solución.

2. **Algoritmo:** descripción de la solución escrita para humanos.
   - ¿Cómo resolvemos el problema?

3. **Programa:** descripción de la solución para ser ejecutada en una computadora.
   - También, ¿cómo resolvemos el problema?
   - Pero descripto en un lenguaje de programación.

Problema, especificación, algoritmo, programa

Dado un problema a resolver (de la vida real), queremos:

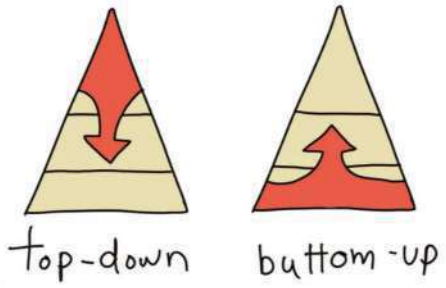
- Poder describir de una manera clara y unívoca (especificación)
  - Esta descripción debería poder ser validada contra el problema real
- Poder diseñar una solución acorde a dicha especificación
  - Este diseño debería poder ser verificado con respecto a la especificación
- Poder implementar un programa acorde a dicho diseño
  - Este programa debería poder ser verificado con respecto a su especificación y su diseño
  - Este programa debería ser la solución al problema planteado

También hablaremos de cómo encarar problemas...

O partir el problema en problemas más chicos...

Los conceptos de modularización y encapsulamiento siempre estarán relacionados con los principios de diseño de software. La estrategia se puede resumir en:

- Descomponer un problema grande en problemas más pequeños (y sencillos)
- Componerlos y obtener la solución al problema original
- Estrategias Top Down versus Bottom Up



Diferenciaremos el QUÉ del CÓMO

- Dado un problema, será importante describirlo sin ambigüedades.
- Una buena descripción no debería condicionarse con sus posibles soluciones.
- Saber que dado un problema, hay muchas formas de describirlo y a su vez, muchas formas de solucionar... y todas pueden ser válidas!

Especificación de problemas

- Una especificación es un contrato que define qué se debe resolver y qué propiedades debe tener la solución.

    1. Define el qué y no el cómo.

- La especificación de un problema incluye un conjunto de parámetros: datos de entrada cuyos valores serán conocidos recién al ejecutar el programa.

- Además de cumplir un rol "contractual", la especificación del problema es insumo para las actividades de ...

    1. testing,
    2. verificación formal de corrección,
    3. derivación formal (construir un programa a partir de la especificación).

Lenguaje naturales y lenguajes formales

Lenguajes naturales

- Idiomas (castellano)
- Mucho poder expresivo (modos verbales –potencial, imperativo–, tiempos verbales –pasado, presente, futuro—, metáforas, etc.)
- Con un plus (conocimiento del contexto, suposiciones, etc)
- No se usan para especificar porque pueden ser ambiguos, y no tienen un cálculo formal.

Lenguajes formales

- Sintaxis sencilla
- Limitan lo que se puede expresar
- Explicitan las suposiciones
- Relación formal entre lo escrito (sintaxis) y su significado (semántica)
- Tienen cálculo para transformar expresiones válidas en otras válidas

Lenguajes formales. Ejemplos

Aritmética. Es un lenguaje formal para los números y sus operaciones. Tiene un cálculo asociado.

Lógicas proposicional, de primer órden, modales, etc.

Lenguajes de especificación de programas

Contratos

- Una especificación es un contrato entre el programador de una función y el usuario de esa función.
- **Ejemplo**: calcular la raíz cuadrada de un número real.
- ¿Cómo es la especificación (informalmente, por ahora) de este problema?
- Para hacer el cálculo, el programa debe recibir un número no negativo.
  - Obligación del usuario: no puede proveer números negativos.
  - Derecho del programador: puede suponer que el argumento recibido no es negativo.
- El resultado va a ser la raíz cuadrada del número recibido.
  - Obligación del programador: debe calcular la raíz, siempre y cuando haya recibido un número no negativo
  - Derecho del usuario: puede suponer que el resultado va a ser correcto

Partes de una especificación (contrato)

1. Encabezado
2. Precondiciones o cláusulas "requiere"
   - Condición sobre los argumentos, que el programador da por cierta.
   - Especifica lo que requiere la función para hacer su tarea.
   - Por ejemplo: "el valor de entrada es un real no negativo"
3. Postcondiciones o cláusulas "asegura"
   - Condiciones sobre el resultado, que deben ser cumplidas por el programador siempre y cuando el usuario haya cumplido las precondiciones.
   - Especifica lo que la función asegura que se va a cumplir después de llamarla (si se cumplía la precondición).
   - Por ejemplo: "la salida es la raíz cuadrada del valor de entrada"

Parámetros y tipos de datos

- La especificación de un problema incluye un conjunto de parámetros: datos de entrada cuyos valores serán conocidos recién al ejecutar el programa.
- Cada parámetro tiene un tipo de datos.
  - **Tipo de datos:** Conjunto de valores provisto de ciertas operaciones para trabajar con estos valores.
- Ejemplo 1: parámetros de tipo *fecha*
  - valores: ternas de números enteros
  - operaciones: comparación, obtener el año, ...
- Ejemplo 2: parámetros de tipo *dinero*
  - valores: números reales con dos decimales
  - operaciones: suma, resta, ...

¿Por qué escribir la especificación del problema?

- Nos ayuda a entender mejor el problema
- Nos ayuda a construir el programa
  - Derivación (Automática) de Programas
- Nos ayuda a prevenir errores en el programa
  - Testing
  - Verificación (Automática) de Programas