Introducción a la programación

Marduk Bolaños Puchet

Miércoles 28 de agosto de 2012

Parte I

Algoritmos y programación

Algoritmo

Un algoritmo es una secuencia **ordenada**, **finita** e **inequívoca** de pasos a seguir para resolver un determinado problema.

Ejemplos

Una receta de cocina, hacer una trenza, instalar un equipo, hacer un sandwich, etc.

Cómo hacer un taco

1. Cocinar la carne
2. Calentar la tortilla
3. Servir la carne en la tortilla
4. Agregar jugo de limón y salsa picante al gusto
5. Enrrollar la tortilla

Elementos de un algoritmo (computacional)

- Datos de entrada: ¿cuántos?, ¿de qué tipo son?, ¿en qué forma los recibe el algoritmo?
- Operaciones básicas: El algoritmo se basa en operaciones básicas y se asume que el ejecutante las entiende.
- Resultados: Se debe especificar qué resultados reportar y cómo. Además, si no se puede calcular el resultado debido a un error en los datos de entrada, se debe indicar.

Un buen algoritmo debe

- Producir el resultado correcto para cualquier conjunto de datos de entrada válido.
- Tener una ejecución eficiente, requiriendo la menor cantidad de pasos posible.
- Estar diseñado para que otros lo puedan entender y modificarlo para resolver otros problemas.

Programa

Conjunto de instrucciones que una computadora puede entender y ejecutar. Así se expresa un algoritmo computacional. Las instrucciones se escriben en un **lenguaje de programación**.

Lenguaje de programación

Conjunto de símbolos y reglas para combinar dichos símbolos, que se utiliza para escribir un programa. Tiene tres elementos:

- Léxico: Conjunto de símbolos permitidos (vocabulario).
- Sintaxis: Reglas que indican cómo realizar construcciones válidas con símbolos.
- Semántica: Reglas que permiten determinar el significado de cualquier construcción del lenguaje.

Lenguajes de programación

Se clasifican según su proximidad con el lenguaje de la máquina o con el lenguaje natural en:

- Lenguajes de bajo nivel: Lenguajes de máquina. Diferente para cada arquitectura.
- Lenguajes de nivel medio: Su vocabulario consiste de palabras cortas que hacen referencia a instrucciones de la máquina.
- Lenguajes de alto nivel: Su vocabulario es un subconjunto del inglés. Son los más cercanos al lenguaje natural.

Estos últimos se clasifican por el tipo de problema que permiten resolver:

- Aplicaciones científicas: Pascal, Fortran
- Procesamiento de datos: Cobol, SQL
- Inteligencia artificial: Lisp, Prolog

Traductores

Ensambladores

Un **ensamblador** es un programa que traduce las instrucciones en lenguaje ensamblador a lenguaje de máquina. Un grupo de instrucciones se pueden agrupar en una macroinstrucción (**macro**). Un **macroensamblador** es un programa que traduce las macros a lenguaje de máquina.

Compiladores

Un **compilador** es un programa que traduce un **programa fuente** escrito en un lenguaje de alto nivel en un **programa objeto** en lenguaje de máquina.

Intérpretes

Un **intérprete** es un programa que traduce programas escritos en un lenguaje de alto nivel a lenguaje de máquina. A diferencia de un compilador, un intérprete traduce una instrucción e inmediatamente la ejecuta. Así continúa hasta llegar al final del programa.

Paradigmas de programación

Un **paradigma de programación** es un estilo fundamental de programación. Un lenguaje de programación puede soportar distintos paradigmas.

- Programación imperativa: El programa define los comandos que la computadora debe ejecutar.
  - Programación procedural: El programa consiste de varios **procedimientos** o **subrutinas** o **funciones**
- Programación declarativa: Es un estilo no imperativo en el que los programas describen el resultado del cómputo, sin especificar los pasos que se deben llevar a cabo para obtener el resultado.
  - Programación funcional: Consiste en evaluar funciones matemáticas, cuyos argumentos pueden ser otras funciones.
  - Programación lógica: El programa consiste de enunciados de lógica y el programa busca la demostración de los mismos.

- Programación estructurada: Los programas se componen de estructuras jerárquicas que controlan el flujo del programa:
  - Sucesión: Enunciados que se ejecutan en cierto orden.
  - Selección: Uno o varios enunciados se ejecutan dependiendo del estado del programa.
  - Repetición: Un enunciado es ejecutado hasta que el programa llega a cierto estado o bien, se aplica una operación a cada elemento de una colección.
  - Programación orientada a objetos: Utiliza estructuras de datos llamadas **objetos** que encapsulan datos y métodos que actúan sobre ellos.
- Programación dirigida por eventos: El flujo del programa es determinado por acciones del usuario (hacer click, presionar una tecla) o por mensajes de otros programas.
- Metaprogramación: Los programas escriben programas o manipulan otros programas (incluyendo a sí mismos).

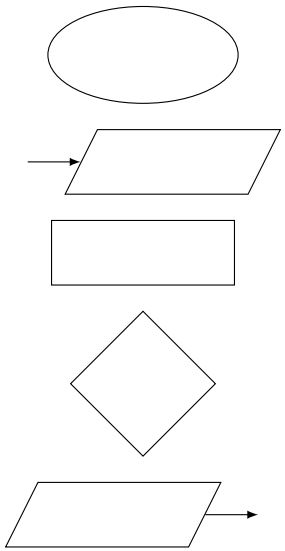
Parte II

Diseño de algoritmos y elementos básicos de un programa

Diagrama de flujo

Es la representación gráfica de un algoritmo. Utiliza símbolos, que representan distintos pasos del algoritmo, unidos por flechas (líneas de flujo) que indican el orden de ejecución de los pasos.

Los símbolos principales son:



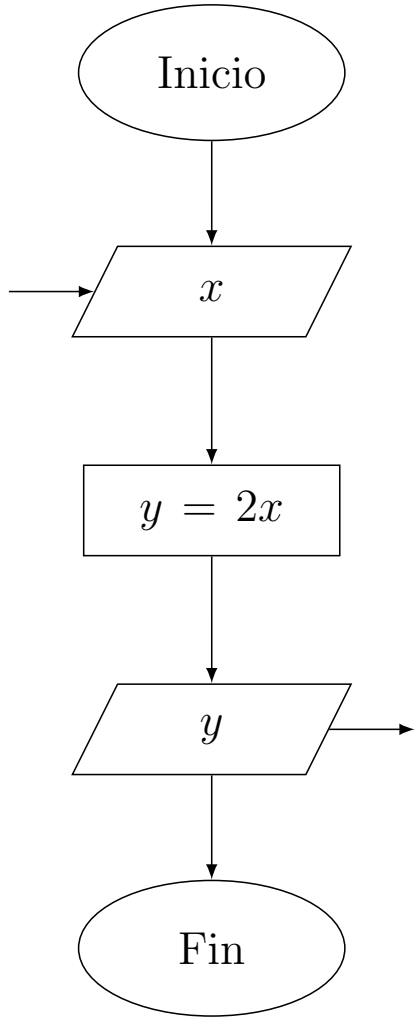
Inicio/Fin

Entrada

Proceso

Decisión

Salida



Datos y expresiones

Tipos de datos

- Números enteros: 0,-2,3
- Números reales: 0.43,-1.23
- Lógico: Verdadero o Falso
- Carácter: Alfabético en mayúscula o minúscula, numérico
- Cadena: Sucesión de caracteres encerrados entre comillas dobles

Constantes

Las constantes son elementos cuyo valor no cambia durante el desarrollo del algoritmo. Ejemplo: pi $= 3.14159$

Variables

Las variables son elementos cuyo valor puede cambiar durante el desarrollo del algoritmo. Se identifican por un **nombre** y un **tipo**. Ejemplo: int a

Expresiones

Una **expresión** es una combinación de operadores y operandos. Los operandos pueden ser constantes, variables u otras expresiones. Los operadores pueden ser aritméticos, lógicos, de bits, relacionales.

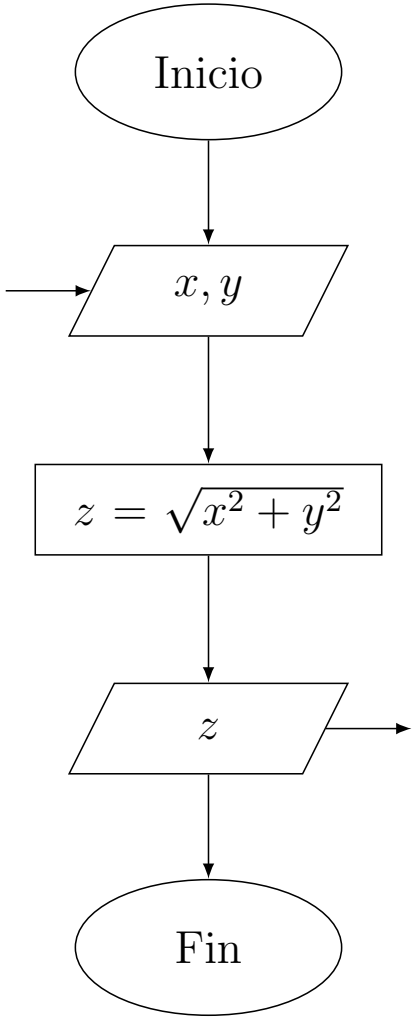
Operadores

Aritméticos	Prioridad	Relacionales	Lógicos
- (menos)	3	$>$	&& (and)
*	2	$<$	|| (or)
/	2	$=$	! (not)
+	1	$!=$	
- (resta)	1	$>=$	
		$<=$	
		$==$	

Ejemplo: $a + b \times c = a + (b \times c)$

Estructuras secuenciales

Es una secuencia lineal de acciones, que se ejecutan en el orden en el que se escribieron.

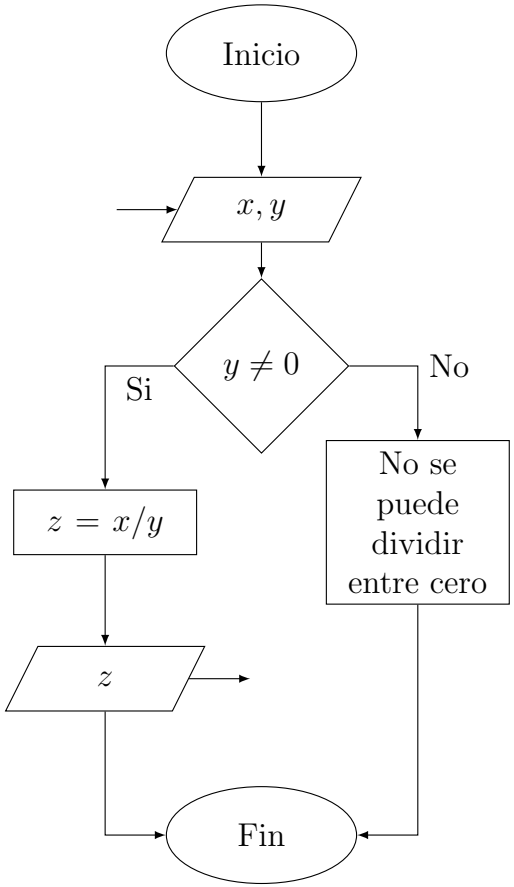


Estructuras selectivas

Permiten controlar la ejecución de acciones que requieren ciertas condiciones para su realización.

Se utilizan cuando:

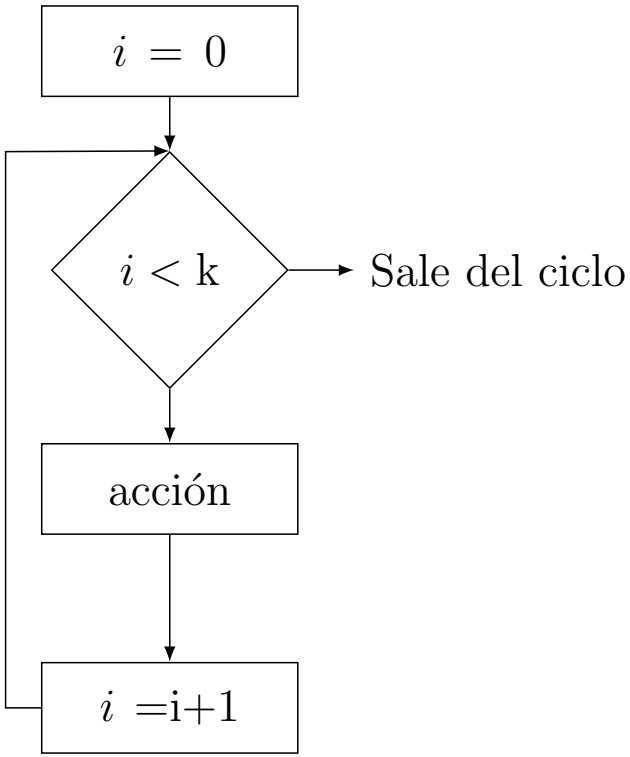
- se tienen acciones excluyentes
- es necesario elegir una acción dentro de un conjunto de acciones
- es necesario verificar que los datos son válidos para la aplicación en cuestión



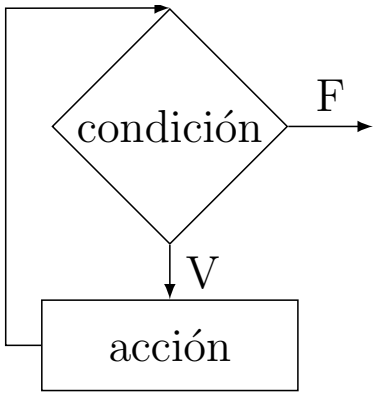
Estructuras de repetición

Iterar es repetir una tarea siempre y cuando la condición de término este bien definida. Existen tres clases de mecanismos de iteración:

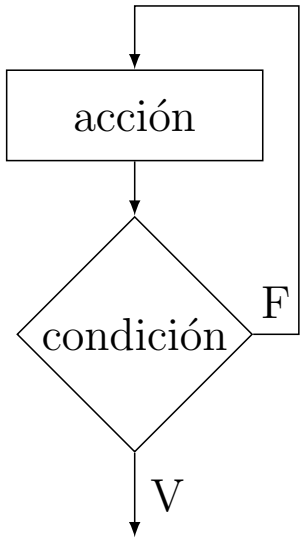
Para $i \in [0, k)$



Siempre que



Repetir hasta que



Pasos para programar

1. **Diseñar** el programa: Es la parte más importante, pues se definen los datos de entrada, el procesamiento de los datos y los datos de salida. Es recomendable hacer un bosquejo del programa con el mayor detalle posible.
2. **Escribir** el programa: Para esto se utiliza un editor (vi, emacs, etc)
3. **Compilar** el programa: Después de grabar el archivo con el código fuente, se utiliza un compilador para convertir el programa a código máquina. Aquellas instrucciones que el compilador no puede entender generar avisos preventivos o mensajes de error. Por lo general, esto ocurre debido a **errores de sintaxis**.
4. **Probar** el programa: Si todo se hizo correctamente, el programa correrá sin problemas. Sin embargo, puede haber dos tipos de errores: errores al momento de la ejecución y errores de lógica.

Parte III

Ejemplos

El primer algoritmo (Euclides, 400 a 300 a.C.)

El máximo común divisor (MCD) de dos números es el número más grande que los divide a ambos.

Sean $a, b \in \mathbb{Z}$. La división de a entre b da como resultado $a = bq + r$.

Si $r = 0$, entonces b es el MCD de ambos. De lo contrario, buscamos un número u que divida tanto a a como a b, es decir, $a = su$ y $b = tu$.
Notamos que u también divide a r, pues
$r = a - bq = su - qtu = (s - qt)u$.

De manera similar, un número v que divide a b y r, también divide a a pues, $a = bq + r = s'vq + t'v = (s'q + t')v$.

Por tanto, cualquier divisor común de a y b es también un divisor común de b y r. De esta manera podemos construir un algoritmo iterativo.

$$
\begin{array}{lll}
q_1 = \left\lfloor \frac{a}{b} \right\rfloor & a = b\,q_1 + r_1 & r_1 = a - b\,q_1 \\
q_2 = \left\lfloor \frac{b}{r_1} \right\rfloor & b = q_2\,r_1 + r_2 & r_2 = b - q_2\,r_1 \\
q_3 = \left\lfloor \frac{r_1}{r_2} \right\rfloor & r_1 = q_3\,r_2 + r_3 & r_3 = r_1 - q_3\,r_2 \\
q_4 = \left\lfloor \frac{r_2}{r_3} \right\rfloor & r_2 = q_4\,r_3 + r_4 & r_4 = r_2 - q_4\,r_3 \\
q_n = \left\lfloor \frac{r_{n-2}}{r_{n-1}} \right\rfloor & r_{n-2} = q_n\,r_{n-1} + r_n & r_n = r_{n-2} \\
 & & -q_n\,r_{n-1} \\
q_{n+1} = \left\lfloor \frac{r_{n-1}}{r_n} \right\rfloor & r_{n-1} = q_{n+1}\,r_n + 0 & r_n = r_{n-1}/q_{n+1}
\end{array}
$$

Algoritmo computacional para encontrar el MCD de dos números a y b

1. Dar los números a y b
2. Sean $x = a$, $y = b$, $u = a \bmod b$
3. Siempre que $u \neq 0$ se hace lo siguiente (de lo contrario ir al paso 4):
   - I $x = y$, $y = u$
   - II $u = x \bmod y$
   - III Volver al paso I.
4. $\mathrm{MCD}(a, b) = y$

Prueba de escritorio

En general

x	y	u
a	b	r_1
b	r_1	r_2
r_1	r_2	r_3
$\vdots$	$\vdots$	$\vdots$
r_{n-1}	r_n	0

Nótese que si $b > a$ entonces $r_1 = a$.

Un ejemplo numérico: ¿MCD(43,18)?

x	y	u
43	18	7
18	7	4
7	4	3
4	3	1
3	1	0

Los números de Fibonacci

Los números de Fibonacci pertenecen a la sucesión:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, $\cdots$

Por definición, los primeros dos números son 0 y 1 y cada número subsecuente es la suma de los dos anteriores. De esta manera, cualquier elemento de la sucesión está dado por:

$F_n = F_{n-1} + F_{n-2}$

Números consecutivos de la sucesión de Fibonacci aparecen frecuentemente en la naturaleza:

- El crecimiento de las hojas en un tallo
- Las frutillas de la piña
- Las florecillas que componen un girasol

Algoritmo computacional para calcular F_k

1. Dar el número k
2. Valores iniciales: $x = 0$, $y = 1$, $z = 0 + 1$
3. Contador: $j = 1$
4. Siempre que $j < k$ (de lo contrario ir al paso 5):
   - I $x = y$, $y = z$
   - II $z = x + y$
   - III $j = j + 1$
   - IV Volver al paso I
5. $F_k = z$

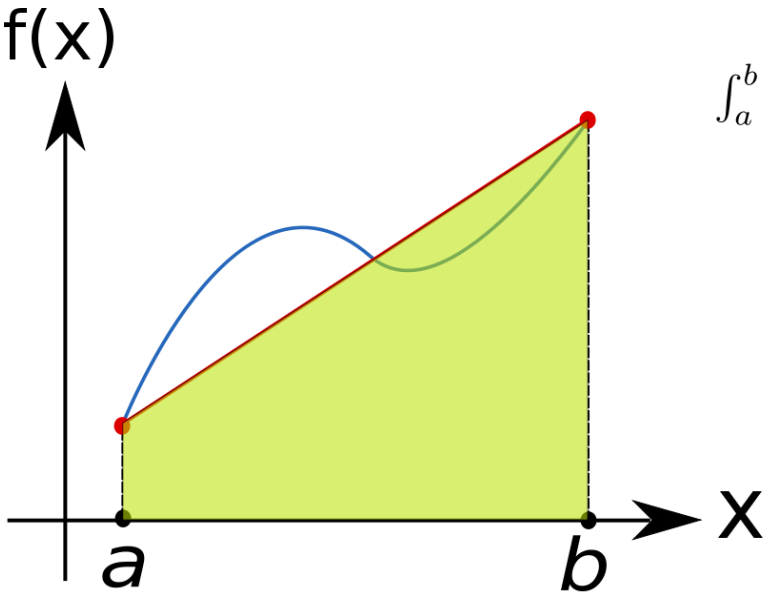
Prueba de escritorio

x	y	z	j
0	1	1	1
1	1	2	2
1	2	3	3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
55	89	144	12
$\vdots$	$\vdots$	$\vdots$	$\vdots$

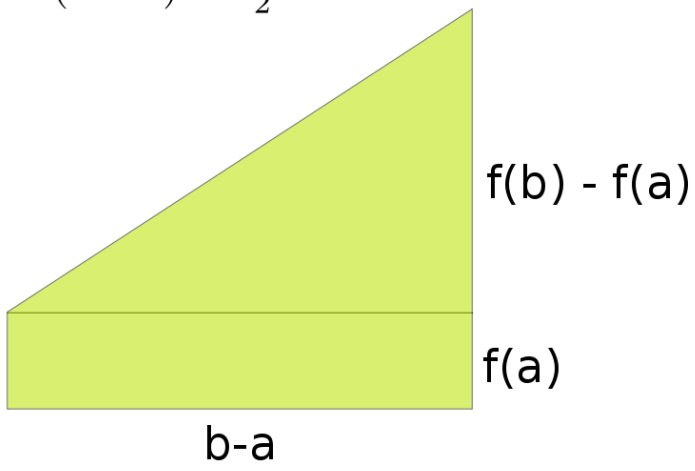
Integración numérica: La regla del trapecio

En física, a menudo es necesario calcular la integral de una función de una variable independiente. Por ejemplo, para encontrar la trayectoria de un objeto como función del tiempo.

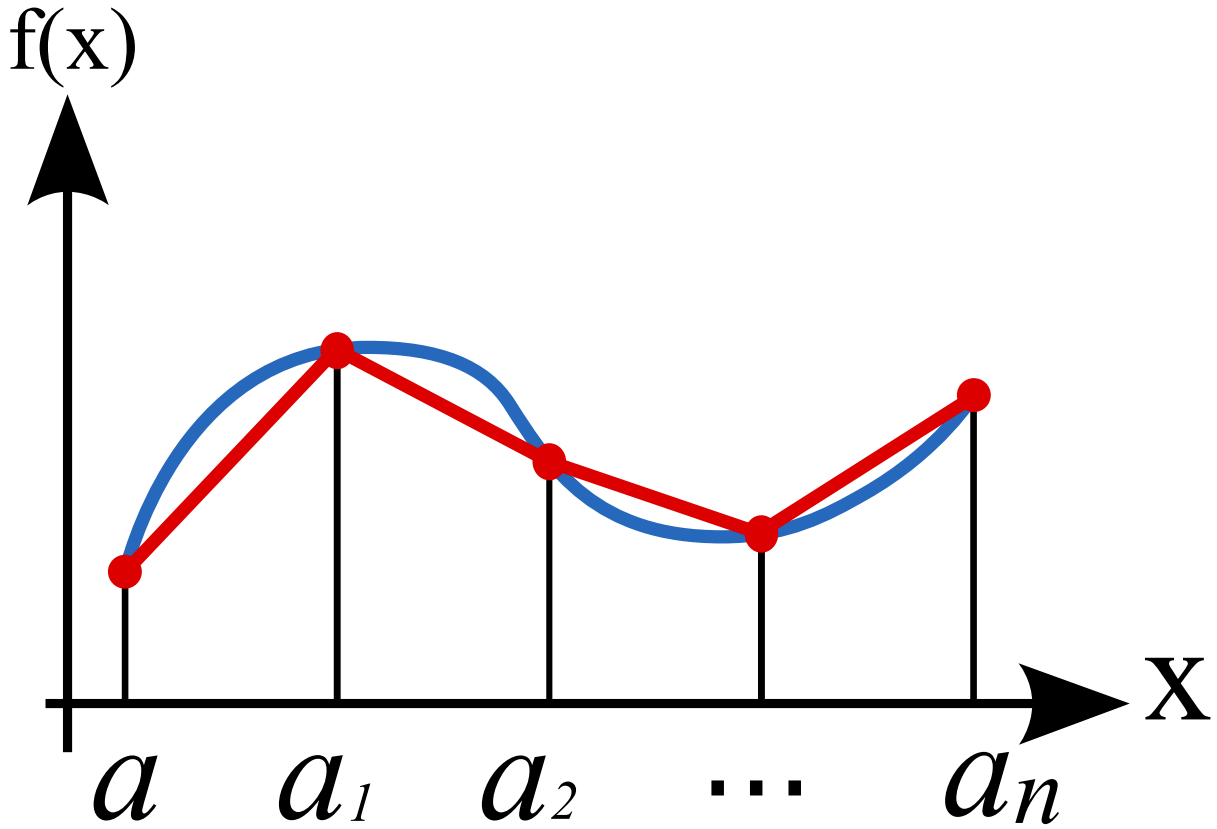
En la mayoría de los casos, la integral no se puede resolver en términos de una función conocida y es necesario evaluarla numéricamente. Uno de los métodos numéricos de integración más sencillos es la regla del trapecio.



$$\int_a^b f(x)dx \approx (b-a)\frac{f(b)+f(a)}{2}$$



Esta aproximación es muy burda. Obtenemos una mejor aproximación si dividimos el intervalo (a, b) en n pedazos y para cada subintervalo de longitud $\frac{b-a}{n}$ utilizamos la regla del trapecio.



Algoritmo computacional para calcular la integral de una función de una variable independiente en un intervalo (a, b) utilizando la regla del trapecio

1. Dar la función f(x)
2. Dar los extremos del intervalo a y b
3. Dar el número n de particiones del intervalo (a, b)
4. Contador: $j = 0$; Integral: $g = 0$; Intervalo: $i = (b - a)/n$; $y = a$
5. Siempre que $j < n$ (de lo contrario ir al paso 6):
   - I $z = y + i$
   - II $g = g + i[f(y) + f(z)]/2$
   - III $y = z$
   - IV $j = j + 1$
   - V Ir al paso I
6. La integral de $f(x)$ en el intervalo (a, b) es g