Introducción a la Programación y Computación 1 Sección E

Ing. MSc. Neftalí Calderón

Algorítmo

Es un conjunto de instrucciones o reglas bien definidas, ordenadas y finitas que permiten realizar una actividad mediante pasos sucesivos.

Un algoritmo puede ser expresado en lenguaje natural o pseudocódigo, sin embargo con el lenguaje natural las descripciones tienden a ser ambiguas y extensas

obtener los valores a operar
realizar el cálculo algebraico
desplegar el resultado en pantalla

Niveles de algorítmo

Descripción de alto nivel: se identifica el problema, se selecciona un método matemático y se explica el algoritmo de manera verbal, Ej. “Un estudiante va a la Universidad”

Descripción formal: se usa pseudocódigo para describir la secuencia de pasos que encuentran la solución.

Implementación: se muestra el algoritmo expresado en un lenguaje de programación específico.

```
System.out.println(nota);
cout<<nota;
escribir(nota);
```

Métodos algoritmicos

Los métodos algorítmicos se pueden implementar en las computadoras, en especial las operaciones lógicas y matemáticas.

Los algoritmos se pueden expresar mediante formulas, instrucciones, diagramas de flujo y pseudocódigos. La definición de un algoritmo debe describir tres partes: Entrada, proceso y salida.

Métodos heurísticos

La heurística es la capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines.

La capacidad heurística es un rasgo característico de los humanos, desde cuyo punto de vista puede describirse como el arte y la ciencia del descubrimiento y de la invención o de resolver problemas mediante la creatividad o pensamiento lateral o divergente.

Un método algorítmico es una prescripción efectuada paso a paso para alcanzar un objetivo en particular.

Un método heurístico en cambio, constituye solo una buena apuesta, es un procedimiento que creamos y que nos ofrece una probabilidad razonable de solución (uso de reglas), estos procedimientos sacrifican exactitud por tiempo.

Paradígmas de programación

Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis.

Cada nuevo paradigma responde a una necesidad real de afrontar problemas.

Un paradigma de programación es una forma de conceptualizar cómo deben estructurarse y organizarse las tareas que se lleva a cabo en un programa.

```
10  leer(dato1)
20  if (dato1 > 10) begin
30      dato1 = dato1 + 15;
40  end
50
60  goto 10
70
```

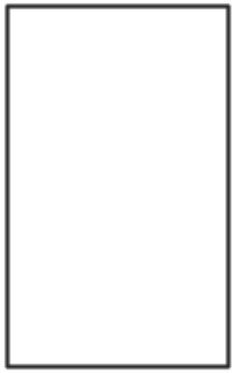
Programación imperativa: el paradigma de programación clásico (cambios de estado)

Entre los paradigmas de programación de software, la programación imperativa (del latín imperare, ordenar) se considera el paradigma clásico. Los primeros lenguajes de programación y, por extensión, también los primeros programas informáticos, se basaban completamente en este enfoque, que prevé una secuencia regularizada de órdenes o instrucciones determinadas.

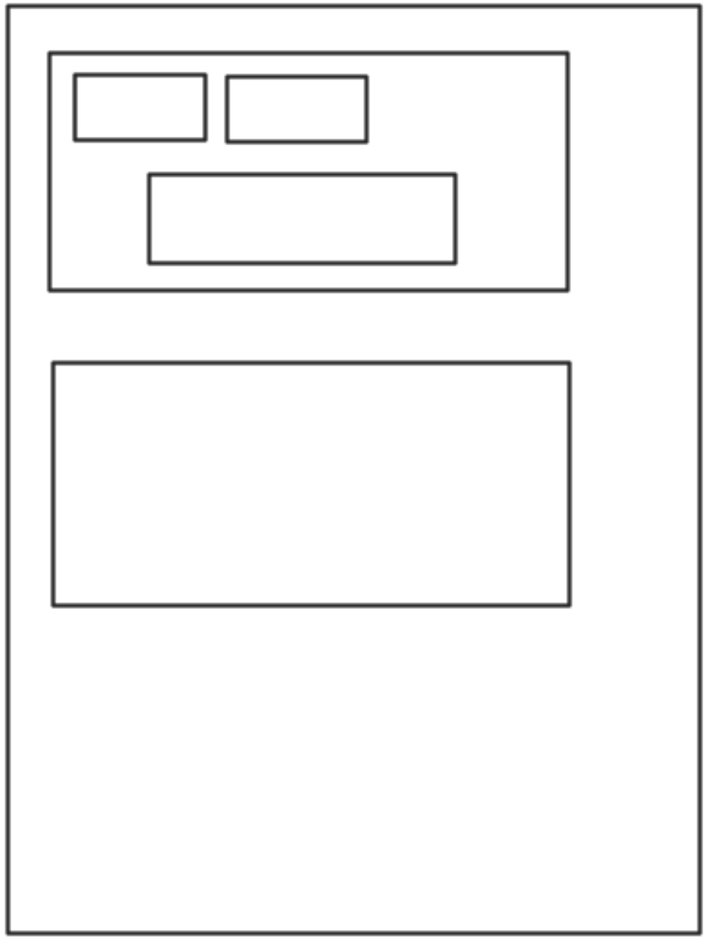
Este paradigma de programación es la base, por ejemplo, de los veteranos lenguajes Pascal y C, así como de todos los lenguajes ensambladores, entre otros. En la programación imperativa, el centro de atención radica, entre otras cosas, en trabajar lo más cerca posible del sistema. Como consecuencia, el código de programación resultante es fácil de comprender y, a la vez, muy abarcable.

Programación estructurada, procedimental, modular, POO

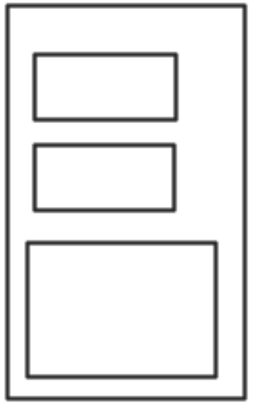
Estructuras de control
- sentencias condicionales
- ciclos



Modular
- Dividir en módulos



Procedimental
- Funciones y procedimientos



ERP
- Facturación
- Inventario
- Contabilidad
- CxC
- CxP
- Bancos

Programación declarativa: paradigmas de software del pasado más reciente (uso de predicados)

De forma paralela a la evolución continuada del hardware y el software, con el enfoque declarativo se desarrolló un paradigma alternativo para la programación de código. El principio fundamental de la programación declarativa radica en la descripción del resultado final que se busca.

Por lo tanto, en primera línea se encuentra el “qué” del resultado y no el “cómo” de los pasos que llevan a la solución, como es el caso en la programación imperativa.

Esto provoca que el código de la programación declarativa sea más difícil de comprender debido al alto grado de abstracción, aunque resulta muy corto y preciso.

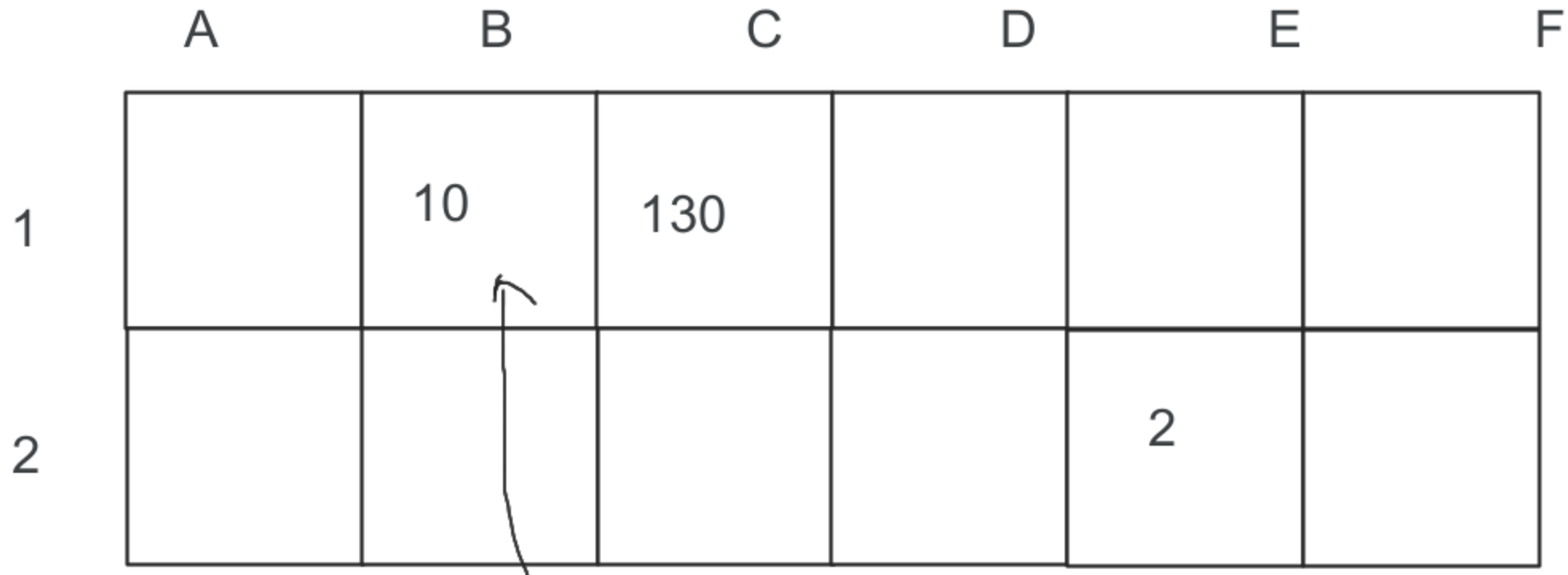
Variable

Una variable es un nombre asociado a un elemento de datos que está situado en posiciones contiguas de la memoria principal, y su valor puede cambiar durante la ejecución de un programa.

Es la forma más simple de almacenamiento, representando una zona de memoria donde se almacena un elemento de datos.

Asignación de memoria

Una variable es la marcadora de una posición en la memoria de la computadora. Cuando se crea una nueva variable en un programa, este asigna la cantidad de memoria en función del tipo de datos de la variable.



int base

int altura

```
int base = 0;
int altura = 2;
int resultado;

resultado = base * altura;
imprimir(resultado);
```

Declaración de variable

Pueden empezar con letras, y puede tener números, letras y el símbolo _.
Con estos límites:

- no puede tener espacios.
- no empezar con un número.
- no puede ser una palabra reservada. Por ejemplo, if, for, while…

Algunos lenguajes pueden permitir empezar con carácter especial, como $.

int a, b, c, e..... aa, bb,

Alcance

A través del alcance se determina hasta dónde se puede leer o cambiar el valor de una variable.

Las variables globales son aquellas que se pueden usar a lo largo de todo el programa. Es decir, su alcance es la aplicación completa.

Las variables locales solo se pueden usar en la función o procedimiento donde se declararon, o también en cualquier otra función que sea llamada por esa función.

```
main() {

string usuario = '';

  int suma(int dato1, dato2) {
     int resultado = 0;
     resultado = dato1 + dato2;
     return(resultado);

    int aaa() {
       resultado
    }
  }

  int multiplicacion(int dato1, dato2) {
     ...
     ....
      ...
  }

}
```