Universidad Autónoma del Estado de México
Centro Universitario Valle de Chalco

Paradigmas de programación y conceptos básicos

Dra. María de Lourdes López García

Lenguaje de Programación Estructurado
Unidad de aprendizaje I
Ingeniero en Computación

Propósito de la Unidad de Aprendizaje

Esta unidad de aprendizaje tiene la finalidad de proporcionar y desarrollar en el alumno las habilidades que requiere para la codificación de programas en un lenguaje de programación estructurado, cubriendo las necesidades de programación y codificación para el desarrollo de sistemas que un profesional en el área de la computación pueda tener.

Introducción

La computadora es una máquina que recibe datos, los procesa y reporta los resultados.

1. Dispositivos de entrada: son todos aquellos que permiten la comunicación del usuario hacia la computadora, como los teclados, lápices ópticos, escáneres y ratones.

2. Dispositivos de salida: son todos aquellos que permite la comunicación de la computadora hacia el usuario, como las pantallas, impresoras y las bocinas.

Introducción

La computadora es una máquina que recibe datos, los procesa y reporta los resultados.

3. Hardware: es la parte palpable de la computadora como la tajeta madre, procesador, memorias, disco duro, etc.

4. **Software**: es la parte intangible, es decir, todos los programas que ayudan al control y uso de la computadora.

5. Usuario

Software

El software es un conjunto de funciones y procedimientos para que la computadora lleve a cabo tareas específicas y que permiten el funcionamiento del hardware. Puede clasificarse en:

- Software de sistemas.
- Software de uso general.
- Software de uso específico.

Lenguaje de programación

1. Un lenguaje (en la informática) es un conjunto de signos y reglas que permite la comunicación con una computadora. (RAE)

2. Un lenguaje de programación es un conjunto de reglas sintácticas y semánticas que permiten la comunicación con una computadora.

Lenguaje de programación

1. Un lenguaje de bajo nivel permite controlar el hardware de la computadora de manera directa.
   - Lenguaje máquina.
   - Lenguaje ensamblador.

2. Un lenguaje de alto nivel permite la interpretación en un lenguaje natural (entendible por los humanos) a un lenguaje máquina a través de algoritmos.
   - C, Java, C++, C#

Algoritmo

1. Es la expresión de cómo calcular el valor de una función dada una lista cualquiera de parámetros de la misma. Antonio Mechén (2011) Diseño de programas. Alfaomega.

2. Es un conjunto finito de operaciones a realizar para resolver un determinado problema.

3. Es un conjunto **finito** de pasos, **precisos** y **ordenados** para resolver un problema. Jiménez, Jiménez y Alvarado (2014) Fundamentos de programación. Alfaomega.

Caraterística de un algoritmo

1. *Finito*: debe tener un número razonable de pasos para resolver el problema, (inicio y fin).

2. *Preciso*: cada instrucción debe ser clara y precisa.

3. *Ordenado*: las instrucciones deben organizarse de manera lógica y ordenada.

Modos de representar un algoritmo

- Narrativo o descriptivo.
- Pseudocódigo.
- Diagrama de flujo.

Problema 1

Desarrolle un algoritmo que lea un número entero y verifique si es par o impar.

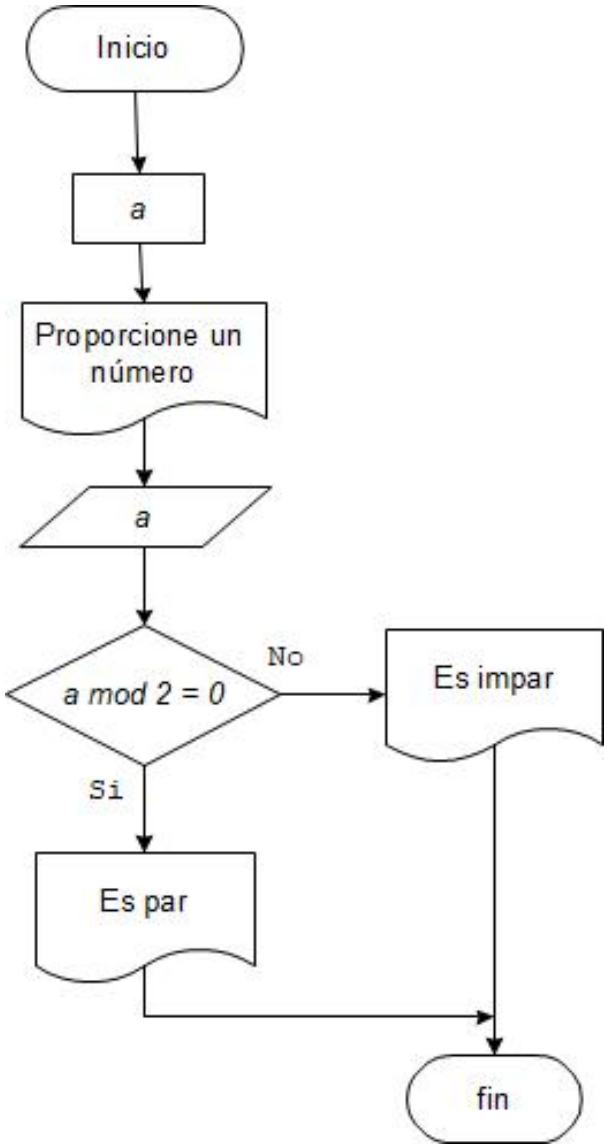
Problema 1: algoritmo narrativo

1. Inicio
2. Se declara un entero *a*.
3. Se solicita al usuario se proporcione un número y se lee *a*.
4. Se calcula el módulo 2 de *a*.
5. Si el resultado es 1 entonces se imprime que es par.
6. Sino entonces se imprime el resultado es impar.
7. Fin.

Problema 1: algoritmo en pseudocódigo

```
1.  Inicio
2.  Declarar el entero a
3.  Imprimir Proporcione un número
4.  Leer a
5.  Si a modulo 2 = 0 entonces
5.1 Imprimir Es par
5.2 Sino
5.2.1 Imprimir Es impar
5.3 fin-Si
6.  Fin.
```

Problema 1: diagrama de flujo



Problema 2

Desarrolle un algoritmo que lea números enteros y los sume hasta que reciba un número negativo.

Problema 2: algoritmo narrativo

1. Inicio
2. Se declaran dos números enteros *x* y *total* y se asigna el valor a cero a ambos.
3. Se solicita al usuario se proporcione el número que desea y se lee *x*.
4. Se verifica si el número proporcionado es mayor a cero.
5. Mientras sea *x* mayor a cero entonces se suma a *total* y vuelve a leer *x*.
6. Si es menor a cero imprime el valor de *total*.
7. Fin.

Problema 2: algoritmo en pseudocódigo

1. Inicio
2. Declarar $x = 0$, $total = 0$ enteros
3. Mientras $x >= 0$ realiza

3.1 $total = total + x$

3.2 Imprimir *Proporcione el valor de* x

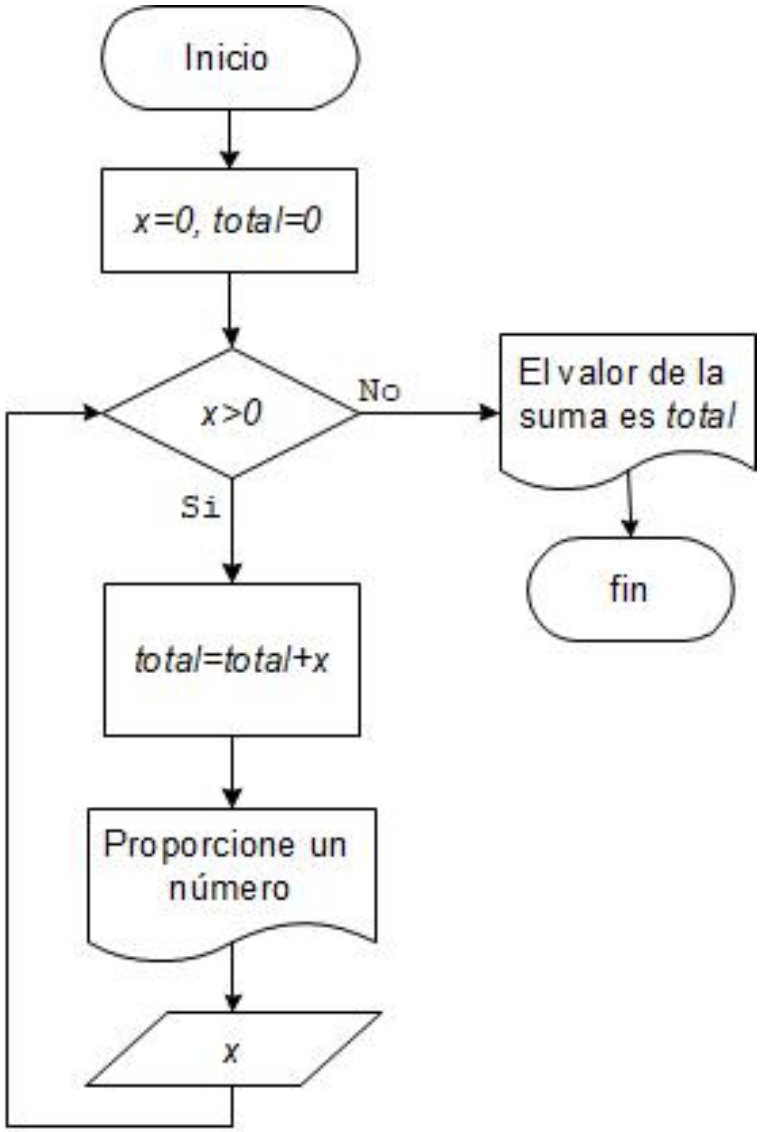
3.3 Leer x

3.4 fin-mientras

4 Imprimir $total$

5. Fin.

Problema 2: diagrama de flujo



Programa

Un programa es un conjunto de instrucciones basadas en un lenguaje de programación que una computadora ejecuta para realizar una acción específica y debe ser conciso, claro, eficiente, eficaz y portable.

- Programa fuente (hecho por el programador)
- Programa objeto (generado por el compilador del lenguaje)

Ejemplo de un programa fuente

```
1  void parImpar()
2  {
3     int a;
4
5     printf("Proporcione un número\n");
6     scanf("%d",&a);
7
8     if (a%2 ==0)
9        printf("Es par");
10    else
11       printf("Es Impar");
12 }
```

Código en Lenguaje C del problema 1.

Ejemplo de un programa fuente

```c
void suma()
{
  int x=0,total=0;

  while (x>=0)
  {
    total=total+x;
    printf("Proporcione un número\n");
    scanf("%d",&x);
  }

  printf("La suma es %d: "total);
  }
```

Código en Lenguaje C del problema 2.

Paradigmas de programación

Los tipos de programación más comunes son los siguientes:

- Imperativo o procedimental: se le indica a la computadora cada paso a realizar, claramente definido.

- Lógico: se basa en el concepto de proposición y predicado, mediante la aplicación de hipótesis, reglas de inferencia, tautologías y teoremas.

Paradigmas de programación

- Estructurado: se basa en el uso de tres instrucciones básicas (secuencia, condicional e interación).
- Modular: consiste en dividir un programa grande en módulos o subprogramas más pequeños, con el fin de hacerlo más entendible y manejable.

Paradigmas de programación

- Orientado a objetos: se basa en expresar el problema como objetos definidos de cierta clase y desarrollando acciones a través de sus métodos.

- En la nube: proporciona un servicio de computación a través de Internet, comprando tiempo máquina sin necesidad de instalación de los compiladores.

Lenguaje de programación estructurado

Resuelve los problemas presentados a través de algoritmos que usen las siguientes instrucciones:

- Instrucciones de secuencia: operaciones aritméticas, lógicas y relacionales.
- Intrucciones de condición: condiciones simples, compuestas y anidadas.
- Instrucciones de repetición: ciclos determinados y no determinados.

Instrucciones de secuencia

Con ellas se realizan las operaciones básicas

- Operaciones aritméticas: suma, resta, multiplicación y división.
- Operaciones lógicas: and, or y not.
- Operaciones relacionales: $<, \leq, >, \geq, \neq, =$

Instrucciones de secuencia

Ejemplos, para el lenguaje de programación C:

- Operaciones aritméticas: $a + b$, $a - b$, $a * b$, a/b.
- Operaciones lógicas: $a\delta\delta b$, $a||b$, $!a$
- Operaciones relacionales:
  $a < b, a \leq b, a > b, a \geq b, a! = b, a == b$

Instrucciones de condición

Con ellas se realizan preguntas con respuestas binarias de 0 o 1, verdadero o falso, si o no.

- Condiciones simples: `if`
- Condiciones compuestas: `if-else`
- Condiciones anidadas: `if (if-else) else (if)`
- Condiciones múltiples: `switch-case`

Ejemplo de un programa fuente

```
//if simple
if (a==b)
      printf("Son iguales");

//if-else
if (m>10 && m<20)
      printf("El rango es correcto");
else
      printf("El número está fuera de rango");

//if anidado
if (a>b && a>c)
   { printf("El mayor es a");
     if (b<c)
         printf("El menor es b");
     else
         printf("El menor es c");
   }
```

Código en Lenguaje C para las instrucciones de condición 1/2.

Ejemplo de un programa fuente

```
switch (opcion)
{
   case 0: r=a+b;
        break;
   case 1: r=a-b;
        break;
   case 2: r=a*b;
        break;
   case 3: r=a/b;
        break;
   default: printf("Opción inválida");
        break;
}
```

Código en Lenguaje C para las instrucciones de condición 2/2.

Instrucciones de repetición

Con ellas se realizan ciclos hasta que la condición de paro se cumpla.

- Ciclo determinado (se sabe cuántas veces va a repetirse): **`for`**
- Ciclos indeterminado (sabe cuándo terminar pero no cuántas veces se va a repetir)
  - Primero preguta y luego realiza: **`while`**
  - Primero realiza y luego pregunta: **`do-while`**

Ejemplo de un programa fuente

```c
// ejemplo for
for (i=0;i<10; i++)
     printf("%d ,i);

//ejemplo while
i=10;
while (i!=0)
{ scanf("%d", &i)
}

//ejemplo do-while
do
{ printf("Elije la opción");
  printf("0. Leer ");
  printf("1. Escribir ");
  printf("3. Salir ");
  scanf("%d", &x);
}while(x!=3);
```

Código en Lenguaje C para las instrucciones de repetición.

Comentarios finales

- El uso de la tecnología permite tener mayor eficiencia y precisión en muchas de las aplicaciones.

- En el ámbito computacional, desarrollar soluciones de acuerdo al problema establecido permite el uso del paradigma de programación más adecuado.

Comentarios finales

- El diseño de un algoritmo que resuelve un problema determinado debe ser independiente del lenguaje en el que se programa, así permitirá ser implementado en diferentes paradigmas y en diferentes lenguajes de programación.

- La programación estructurada permite resolver problemas usando las instrucciones básicas de secuencia, condición y repetición, sin la necesidad de utilizar herramientas más elaboradas.

Bibliografía

1. José Jiménez Murillo, Eréndira Jiménez Hernández y Laura Alvarado Zamora (2014) Fundamentos de programación. Editorial Alfaomega.

2. Pablo Augusto Sznajdleder (2012) Algoritmos a fondo con implementaciones en C y JAVA. Editorial Alfaomega.

3. Antonio Mechén Peñuela (2011) Diseño de programas. Editorial Alfaomega.

4. José Jiménez Murillo, (2009) Matemática para la computación. Editorial Alfaomega.