PROGRAMACIÓN IMPERATIVA CON LENGUAJE C

Omar Iván Trejos Buriticá

ECOE EDICIONES

CONTENIDO

INTRODUCCIÓN

¿Cómo enseñar programación de computadores? Esta ha sido una pregunta que, de manera recurrente, me he planteado a lo largo de los más de veinte años que tengo de experiencia docente. He recurrido a muchos libros que me guíen e indiquen la mejor forma de hacerlo. En ese camino, me he encontrado con muchos libros cuyo contenido versa sobre programación de computadores, pero ninguno de ellos me orientó hacia la forma de hacerlo.

Con el tiempo me arriesgué a pensar que la respuesta a mis necesidades podría estar en mí mismo y fue entonces cuando decidí estudiar un Doctorado en Ciencias de la Educación. Todas mis expectativas fueron colmadas mientras iba preparando una forma de enseñar programación de computadores, la gran respuesta que había inspirado no sólo mis estudios sino también mis inquietudes investigativas.

Este libro es una propuesta para enseñar programación de computadores basado en un lenguaje bastante popular conocido como el lenguaje C. Intento poner esta propuesta a consideración de docentes, estudiantes, profesionales y programadores en general que quieran aprender a programar y que encuentren la respuesta en sus páginas.

Este libro es producto de muchas investigaciones. Entre estas podemos contar el proceso que me llevó a concluir con éxito mi tesis doctoral que, precisamente, se fundamentó en el tema de la enseñanza en Ingeniería. Mi intención es que usted,

querido lector, aproveche estos contenidos al máximo, le saque todo el provecho posible y aprenda a programar bajo la combinación de dos paradigmas. Estos le ilustrarán cómo hacer que un computador haga por usted diferentes procesos aprovechando su alta velocidad de proceso. Reciba mi bienvenida a este libro y aprovéchelo al máximo.

Para mí, la respuesta ha sido encontrada a través de una dosificación del conocimiento en un formato de lecciones. Estas están pensadas para que usted las lea, realmente las lea y las disfrute pues, por ser tan cortas, podrá asimilarlas de una forma más simple. Espero que usted también encuentre en este libro esas respuestas que le permitan decir: "Aprendí a programar".

Omar Iván Trejos Buriticá, PhD
omartrejos@utp.edu.co

¿Cómo usar este libro?

En primera instancia, le recomiendo que lo lea pausadamente. No se afane en avanzar, la apropiación y asimilación de los conceptos pertinentes a la programación de computadores implica tiempo pues no son exactamente los mismos conceptos que subyacen a la lógica natural y deliberativa que tenemos los seres humanos.

Lea y vuelva a leer, piense en lo que ha leído y, ante todo, realice (o, por lo menos, intente realizar) los ejercicios propuestos. Pregunte cuando tenga dudas y propóngase terminar los ejercicios siempre. No los deje a medio camino. Verá que cada vez que usted lleve un ejercicio hasta el final notará cómo su lógica humana se amplía para acudir a la lógica computacional en los casos donde corresponda.

Para el profesor

Utilice este libro al ritmo que sus estudiantes le permitan hacerlo, aprovechando que está diseñado en forma de lecciones. Cada lección proporciona el material para una sesión de clase. Usted y yo sabemos que lo más importante no es avanzar en un contenido, sino que los estudiantes realmente aprendan lo que se puede asimilar de ese eje temático. Sea muy paciente con la asimilación de la lógica de programación, de la programación y de las minucias de los lenguajes de programación. Estas tres instancias deben ser trasegadas por sus estudiantes para alcanzar el nivel de excelencia que usted ya ha adquirido en materia de programación.

Para el estudiante

Siga el ritmo que le indique el profesor. Revise los ejercicios resueltos, resuelva los ejercicios propuestos y, siempre sin excepción, pregunte. No se quede con dudas. Por simples que le parezcan, recuerde que cuando se trata de lógica, las dudas son también lógicas, pero si no se resuelven simplemente van quedando lagunas cuya resolución posteriormente puede llegar a ser más compleja. Comparta las soluciones encontradas con sus compañeros. Es muy enriquecedor alimentarse de la lógica de los demás y aportarles lo que, desde nuestra lógica, se pueda aportar. La forma excelsa para aprender a nadar es nadando. Asimismo, el camino óptimo para asimilar la lógica de programación es practicando, practicando y practicando.

LECCIÓN 1

LOS PRIMEROS PASOS

1.1. ¿Qué es programar?

Primero quiero compartir con usted lo que significa programar. Formalmente programar se puede definir como la capacidad que tiene una persona para articular instrucciones de un lenguaje de programación y lograr que un computador trabaje por sí mismo de forma más rápida y confiable.

Informalmente, podríamos decir que programar es jugar un poco con la tecnología; es lograr que toda la capacidad de procesamiento de un computador (o de un dispositivo similar) se ponga a nuestro servicio; es lograr que el computador resuelva ciertos problemas por nosotros con las instrucciones que le hayamos ingresado y que funcione del modo en que nosotros queremos.

Programar reúne cinco elementos que lo hacen posible: paciencia, conocimiento, experiencia, talento y arte. La paciencia es necesaria dado que la construcción de programas para computadores no siempre es tan simple como esperamos que sea. Normalmente, son nuestras necesidades las que pueden tener un nivel de dificultad mayor cuando esperamos que un computador las resuelva.

El conocimiento es necesario pues se requiere una adecuada apropiación de la teoría de la programación, su metodología, algún lenguaje de programación y algo del modelo computacional en el cual dicho lenguaje funciona. La experiencia es necesaria debido a que muchos de los problemas que resultan cuando uno está programando se pueden resolver fácilmente cuando se cuenta con más horas de práctica.

El talento es inherente a cualquier actividad humana en donde primen las capacidades intelectuales. Aun teniendo la misma formación, los mismos profesores, los mismos libros y la misma experiencia, es muy posible que dos programadores sean muy diferentes y uno de ellos sea más talentoso, de manera natural, que el otro.

Finalmente, teniendo en cuenta lo dicho, podríamos decir que programar se convierte en un arte debido a que el ingenio y la creatividad cobran especial importancia y son relevantes para la construcción de programas eficientes, útiles y funcionales.

En este sentido, se puede definir la programación como el área que estudia el arte de programar. Esta constituye el área formal en la cual se estudian las bases matemáticas, metodológicas, conceptuales y tecnológicas para llegar a programar con suficiencia, eficiencia y pureza en el código.

1.2. Paradigma de programación

Usted encontrará un término que será utilizado con frecuencia en este y en otros libros de programación de computadores: paradigma de programación. Bueno, voy a descomponer el término en dos partes. Primero, podemos decir que el término paradigma corresponde a un modelo o conjunto de esquemas que permiten tener una determinada óptica para resolver un problema el cual es aceptado por una comunidad específica. Un paradigma es una manera de pensar y resolver una determinada situación. Es la forma como se articulan matemáticas, ciencia y tecnologías para llegar a un tipo específico de solución. Es el fundamento que subyace a toda forma de expresión tecnológica. Por ejemplo, la búsqueda de información hace treinta años se hacía por medio de la consulta bibliográfica en bibliotecas. En nuestros días, esa misma búsqueda de información se hace a través de los computadores y del Internet. El paradigma de hace treinta años era la consulta bibliográfica, hoy es es la consulta por Internet.

Como ya tenemos resuelto el primer término y el segundo fue trabajado previamente, entonces ya podemos decir qué es un paradigma de programación. Estamos hablando de un modelo que proporciona los elementos matemáticos, conceptuales y técnicos necesarios para poder aprovechar las potencialidades de un lenguaje de programación en la solución de un problema específico que puede ser resuelto a través de la tecnología.

1.3. Lenguaje de programación

Hasta el momento ha visto el de término lenguaje de programación. Pues bien, sepa que este concepto se trata de un conjunto de instrucciones entendibles y ejecutables por un computador, que tiene una sintaxis propia y que, normalmente, cuenta con un entorno y unas reglas de desarrollo.

La sintaxis de un lenguaje de *programasion* es equivalente a la ortografía del idioma español. Supongo que estará sorprendido de que en este párrafo la palabra programación esté escrita con la letra s; no fue un error involuntario sino totalmente consciente. Era para que usted recordara plenamente qué es la sintaxis.

Sabemos que todas las palabras terminadas en −ción se escriben con c, a excepción de pensión, misión, lesión, pasión, torsión, fusión, fisión, presión e impresión. Esa es una regla ortográfica de muchas que tiene nuestro idioma español.

De la misma manera, todo lenguaje de programación tiene una cantidad, comparativamente pequeña, de reglas sintácticas necesarias para que el computador (a través de un elemento llamado compilador) entienda las instrucciones y las ejecuta apropiadamente. Un compilador es un programa que permite verificar el cumplimiento de las reglas sintácticas de un lenguaje de programación.

1.4. Aprender a programar

¿Por qué aprender a programar? Le daré primero la razón que me nace del corazón, la que siento en el alma. Considero que es necesario porque es un juego apasionante, retador y muy satisfactorio. Pero existen otras razones menos pasionales y más profesionales que le pueden dar una idea clara de las razones por las cuales aprender a programar: (a) porque permite conocer muy bien las potencialidades tecnológicas de los computadores, (b) porque a través de la programación usted puede lograr realizar tareas que, de otras formas, serían muy dispendiosas, (c) porque la programación permite entender hasta donde la tecnología nos puede ayudar y a partir de dónde no nos puede ayudar, (d) porque es la espina dorsal de la carrera más popular en América Latina: la ingeniería de sistemas, (e) porque, con el avance de la tecnología de los computadores, la programación cada día tiene nuevos retos, nuevas formas de expresión y, sobre todo, nuevas aplicaciones. Podrían existir muchas otras razones que justifiquen las razones por las cuales aprender a programar (y yo tendría muchísimas más). Por ahora, considere que aprender a programar puede convertirse, con el tiempo, en una necesidad profesional. Considerando todo lo anterior, me gustaría invitarlo a que se adelante en el tiempo para que cuando aparezca esta necesidad usted ya la haya resuelto. La aparición, cada vez mayor, de asignaturas de programación en diferentes programas profesionales así lo confirma.

El paradigma de programación que conoceremos es el paradigma imperativo, un paradigma bastante depurado pues es el que más años ha durado en el mundo de la tecnología. Esto quiere decir que no tendremos que hacer muchísimos esfuerzos para entender algunos conceptos porque, de alguna manera, son muy cercanos a los conceptos propios de nuestra lógica natural.

Este paradigma se puede utilizar cómodamente a la luz de varios lenguajes de programación. En este libro utilizaremos un lenguaje de programación llamado

lenguaje C en el entorno DevC++. A pesar de ser un lenguaje híbrido (permite la utilización de varios paradigmas de programación) vamos a estudiar su arista imperativa. Por esta razón, muchos de los conceptos que usted está a punto de estudiar corresponden a dicha arista dado que este lenguaje puede ir mucho más allá.

1.5. Lenguaje C

En este libro es posible que utilicemos el nombre lenguaje C o DevC++ indistintamente. Esto se debe a que C es el nombre del lenguaje y DevC++ el entorno con el cual vamos a trabajar. Este lenguaje fue desarrollado por Dennis Ritchie y Brian Kernigham en la década de los ochenta e introducido al mundo académico a través de una serie de artículos que destacaban la necesidad de tener un lenguaje que fuera más flexible, abierto y sencillo que los lenguajes de programación que existían en ese momento en el mercado.

Estamos hablando de un lenguaje de programación sencillo en el cual, acorde con la filosofía del paradigma imperativo, se privilegia la instrucción y el concepto de estados como la base para la construcción de cualquier programa. Para que tenga una idea más clara es necesario que sepa que así como la célula es la base de todo ser vivo y la familia la base de la sociedad, la instrucción y el concepto de estados son la base del paradigma funcional y, específicamente, de los programas construidos en Scheme. El paradigma imperativo y la programación imperativa derivan su nombre del concepto de instrucción, su núcleo de trabajo.

El código de un programa es el conjunto de instrucciones que escribimos para que el programa logre un determinado objetivo. Normalmente, el programa es antecedido por el algoritmo, es decir, el conjunto de pasos lógicos que, ajustados a un paradigma de programación, permiten alcanzar un objetivo. El programa es equivalente al algoritmo, pero escrito en los términos de un lenguaje de programación.

Hagamos una analogía muy sencilla. Imagínese una receta compuesta de ciertos ingredientes y considere la mejor forma de combinarlos para obtener un determinado plato. Pues bien, ingredientes y preparación constituyen lo que podríamos llamar el programa. Este no solo muestra los elementos necesarios para la preparación, sino también el procedimiento para combinarlos y obtener un plato deseado. En uno de los siguientes capítulos explicaremos con más detalle alguno de los términos que se acaban de utilizar. En aquél capítulo hablaremos de la dimensión metodológica necesaria para la construcción de un programa. Este capítulo le permitirá conocer lo que hay que hacer para iniciar un programa.

1.6. Ejercicios propuestos

Sería muy difícil dejar esta primera aproximación a la programación sin plantear unas preguntas que, preferiblemente, deberá responder en sus propios términos:

a. ¿Qué es programar?
b. ¿Qué es un programa?
c. ¿Qué es un paradigma de programación?
d. ¿Qué es un lenguaje de programación?
e. ¿Qué se necesita para estudiar programación?

LECCIÓN 2

EL PRIMER PROGRAMA

2.1. Concepto de función

La base de la programación imperativa es la instrucción y la base del paradigma funcional, la función. Una instrucción es una sentencia que requiere de la memoria, como puente de paso, para una ejecución existosa. Por otro lado, la función se refiere al conjunto de instrucciones que permiten lograr un pequeño objetivo. El presente libro de programación imperativa está basado en los elementos de juicio que proporciona la programación funcional, la cual considera a la función como su núcleo principal de trabajo. Más adelante se verán, a fondo, las ventajas y características de la función como fundamento para el desarrollo de un programa. Una función es, por lo tanto, equivalente a un "pequeño programa" que permite lograr, a su vez, un "pequeño objetivo". Una función debe lograr un objetivo específico.

2.2. Primer programa

En esta sección vamos a construir una función que permita recibir un número entero como parámetro y retornar su último dígito. Para hacerlo, lo primero que debemos hacer para escribir la función en lenguaje C, es escribir el tipo de dato que se va a retornar. Como el último dígito de un número entero es, a su vez, un número entero (pues sería un número comprendido entre 0 y 9), entonces debe escribirse la palabra reservada *int* que le indica al lenguaje C que la función va a retornar un valor entero.

Lo segundo que hacemos es ponerle un nombre a la función. Ese nombre debe ser tan claro que fácilmente haga referencia a lo que hace la función. En este caso, como el enunciado consiste en "construir una función que permita recibir un número entero como parámetro y retornar su último dígito", entonces la función se llamará *ultimodigito* (este nombre es suficientemente claro). Lo tercero que se le escribe a una función son los parámetros que necesita, es decir, los ingredientes necesarios para que la función funcione.

Si quisiéramos cocinar un plato de arroz con pollo, salta a la vista que requeriremos arroz y pollo. Despues, al tenerlos y mezclarlos, ya tendríamos, en su mínima expresión, un arroz con pollo. Los condimentos agregados responden al gusto de quien cocine, pero no son absolutamente necesarios. El enunciado hace referencia a que "…permita recibir un número entero como parámetro…". Eso significa que el nombre de una variable entera se encerrará entre paréntesis indicando que en la función *ultimodigito* está su parámetro, es decir, el ingrediente que se requiere para que funcione.

Lo que tendríamos hasta el momento sería lo siguiente:

```
int ultimodigito (int num)
```

Esto le indica al compilador de lenguaje C que se está construyendo una función que retorna un valor entero, que se llama *ultimodigito* y que para que funcione, requiere de un valor entero que se almacenará en la variable *num*. Esta línea se conoce como el prototipo de la función.

A continuación, construimos el cuerpo de la función que, por razones de sintaxis del lenguaje C, se encierra entre llaves { }. El cuerpo de esta función podrá ser el siguiente:

```
{       int Ud;                 // Línea 1
        Ud = num % 10;          // Línea 2
        return (Ud);    }       // Línea 3
```

En la línea 1 aparece el inicio de la función con la llave que abre ({) y también se declara una variable llamada *Ud* (necesaria para dar entender que es un último dígito). Esta variable será la que almacenará el último dígito del número recibido como parámetro. La línea 2 almacena en la variable *Ud*, el resultado de obtener el residuo de la división del contenido de la variable *num* entre 10.

Finalmente, la línea 3 retorna el valor almacenado en la variable *Ud* y encontramos allí el fin del cuerpo de la función con la llave que cierra (}). Nótese que al final lógico de cada instrucción se escribe un signo de punto y coma (;). De la misma manera, cuando se escriben dos signos / / seguidos, se le indica el lenguaje C que lo que sigue a continuación es un comentario, es decir, un texto explicativo del código. De acuerdo a esto, la función completa es la siguiente:

```
int ultimodigito (int num)
{   int Ud;                    // Variable local
    Ud = num % 10;             // Se obtiene último dígito
    return (Ud);     }         // Retorna el último dígito
```

Esta es la primera función escrita en lenguaje C que, para poderla usar, necesita que alguien la active. Ese alguien es otra función. En este caso, la función que nombrará a *ultimodigito* es la función principal. Estamos hablando de un tipo de función especial que tiene un nombre específico y unas características concretas también.

Dado que es una función, debe tener primero, al igual que la función *ultimodigito*, un tipo de dato a retornar. En la función principal, el tipo de dato siempre es entero y se identifica con la palabra reservada *int*. Lo siguiente es el nombre de la función. La función principal siempre tiene un mismo nombre: *main*( ). Los paréntesis indican que es una función que, por ahora, no requiere parámetros, es decir, ingredientes para que funcione. De esta forma, el prototipo de la función principal es el siguiente:

```
int main( )
```

Ahora bien, el cuerpo de esta función podría ser el siguiente:

```
{                                       // Línea 1
    int n;                              // Línea 2
    cout<<"Digite un número entero…>";  // Línea 3
    cin>>n;                             // Línea 4
    cout<<"Su último dígito es ";       // Línea 5
cout<<ultimodigito(n);                  // Línea 6
}                                       // Línea 7
```

- La línea 1 indica, con la llave que abre ({), el comienzo del cuerpo de la función principal.
- En la línea 2 se declara una variable local (una variable que solo se reconoce dentro de la función principal). Su nombre es n.
- La línea 3 indica que se despliega un aviso en la pantalla solicitando un número entero.
- La línea 4 hace la lectura de un dato entero y lo almacena en la variable *n*, la cual se declaró de manera local. La línea 5 despliega en pantalla otro título que dice *"Su último dígito es"*.
- En la línea 6 se muestra lo que retorne de la función *ultimodigito*, a la cual se le envía como parámetro el valor contenido en la variable n. Este valor se guarda en la variable *num*, que es el parámetro de la función *ultimodigito*, y dentro de esa función, realiza todas las operaciones con el contenido de la variable *num* como si fuera el contenido de la variable *n* original.

- Por último, en la línea 7 aparece el final de la función principal.

De acuerdo a lo expuesto, solo faltaría adicionar dos líneas. En la primera línea del programa se debe escribir:

#include <iostream.h>

Esto permite que las instrucciones *cin* y *cout* funcionen adecuadamente dado que *iostream,h* es una librería en donde se encuentran funciones y objetos que facilitan la lectura y escritura de datos. Las librerías son conjuntos de funciones que están disponibles en los programas, cuando se les incluye. Estas están pensadas con el fin de realizar operaciones que, de otra forma, deberían implementarse de nuevo completamente. La otra línea que se necesita adicionar es la siguiente:

using namespace std;

Esta línea permite entender que las funciones *cin* y *cout* se relacionan con los dispositivos estándar: teclado y pantalla respectivamente. De no incluir esta línea, se necesitaría hacer una implementación diferente al momento de querer usar *cin* y *cout* para leer y escribir datos.

2.3. Código completo

A continuación, se reproduce lo que podría ser el programa completo debidamente comentado:

/* CONSTRUIR UN PROGRAMA QUE PERMITA LEER UN ENTERO Y QUE MUESTRE SU ÚLTIMO DÍGITO */

```
// Cabecera
#include <iostream.h>                    // Librerías
using namespace std;                     // Entrada y salída estándar

// Función que obtiene el último dígito de un número
int ultimodigito (int num)
{
        int Ud;                          // Variable local
        Ud = num % 10;                   // Se obtiene último dígito
        return (Ud);                     // Retorna el último dígito
}
// Función principal
int main( )
{       // Inicio function ppal
        int n;                           // Variable local
```

```
  cout<<"Digite un número entero…>";        // Título
  cin>>n;                                   // Leer y almacenar un entero
  cout<<"Su último dígito es ";             // Título
  cout<<ultimodigito(n);                    // Muestre último dígito
}                                           // Fin función ppal
```

Este es tan solo el primer programa completo. Este tiene una cabecera que incluye el llamado a una librería conocida como *iostream,h.* Así mismo, tiene también una función llamada *ultimodigito* y tiene una función principal llamada *main( ).* Es importante recordar que la función principal siempre se llamará *main( )*.

2.4. Ejercicios propuestos

1. Construir un programa que permita leer un número entero y mostrar la mitad de su último dígito.
2. Construir un programa que permita leer un número entero y mostrar los dos últimos dígitos.
3. Construir un programa que permita leer dos enteros y mostrar la suma del último dígito de cada uno de los números leídos.

PROGRAMACIÓN IMPERATIVA CON LENGUAJE C

La obra ofrece, a quienes se inician en la programación de computadores, una guía organizada desde la metodología hasta la teoría y práctica que requieren el conocimiento de matrices y funciones para resolver un problema de manera sencilla, explicando los elementos básicos que intervienen en la programación de computadores.

Los capítulos del libro siguen la línea invisible determinada por los conceptos propios de la programación de computadores, desde la forma de plantear una solución para un problema computable hasta el aprovechamiento de matrices y funciones para simplificarlo. Se explican conceptos teóricos al margen de cualquier lenguaje de programación, acudiendo tanto a conceptos comunes de estos como a los diferentes paradigmas de la programación tradicional y moderna.

El libro está dirigido a estudiantes y docentes de las áreas de Informática, Programación, Ingeniería de Sistemas, y en general a todos los interesados en la programación de computadores.

Colección: Ingeniería y salud en el trabajo
Área: Informática

Incluye

- Ejemplos explicados detalladamente a la luz de cada concepto estudiado.
- Ejercicios propuestos al finalizar cada capítulo, con diferentes niveles de exigencia.
- Conceptos comunes a los paradigmas de la programación moderna.

Omar Iván Trejos Buriticá

Ingeniero de sistemas de la Universidad INCCA de Colombia. Especialista en Instrumentación física y Master en Comunicación educativa de la Universidad Tecnológica de Pereira. PhD en Ciencias de la educación (RUDECOLOMBIA – UTP). Ha sido docente, desde 1996, de la Facultad de Ingenierías, Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira, donde se ha desempeñado, además, como Decano de la Facultad y Director del programa de Ingeniería de Sistemas y Computación.

ECOE EDICIONES

www.ecoeediciones.com

ISBN 978-958-771-543-9

9 789587 715439

e-ISBN 978-958-771-544-6