Teodoro Luciano Córdova Neri

Lenguaje de programación estructurada

y sus aplicaciones en Borland C++ 5.02

Universidad Nacional de Ingeniería
Editorial Universitaria

Rector **Dr. Ing. Aurelio Padilla Ríos**
Primer Vicerrector **Geol. José S. Martínez Talledo**
Segundo Vicerrector **Msc. Ing. Walter Zaldívar Álvarez**

Primera edición, mayo de 2012

Lenguaje de programación estructurada y sus aplicaciones en Borland C++ 5.0.2
Impreso en el Perú / Printed in Peru

Universidad Nacional de Ingeniería
Editorial Universitaria

Av. Túpac Amaru 210, Rímac – Lima
Pabellón Central / Sótano
Telfs. 4814196 / 4811070 anexo 215
Correo-e: eduni@uni.edu.pe
Jefe EDUNI: Prof. Álvaro Montaño Freire
Coordinador Editorial: Nilton Zelada Minaya

Impreso en la Imprenta de la Editorial Universitaria de la
Universidad Nacional de Ingeniería

ISBN 978-612-4072-23-9

Hecho el Depósito Legal en la Biblioteca Nacional del Perú
Nº 2011-13203

Palabras liminares

Me complace felicitar a los docentes de nuestra Universidad ganadores del II Concurso para la Publicación de Libros de Texto convocado por el Rectorado y realizado en cada una de las Facultades. Una de las políticas medulares del Rectorado es la permanente mejora en la calidad académica, y en ese sentido nos interesa que cada docente tenga la oportunidad de convertir su labor cotidiana de enseñanza en textos para uso de los estudiantes universitarios de todo el país.

Los autores han hecho un meritorio esfuerzo para organizar los temas de sus exposiciones, realizando investigaciones y consultando fuentes peruanas y extranjeras, así como recogiendo el fruto del diálogo con sus colegas y los propios estudiantes. Asimismo, se han esmerado en presentar sus cursos de manera que facilita el acceso por parte de los interesados.

La publicación de textos académicos es una de las obligaciones de toda universidad y uno de los índices que se toma en cuenta para la evaluación de la calidad académica. Por ende, seguiremos apoyando la publicación de libros y revistas a través de nuestra Editorial Universitaria, cuya meta es formar parte del liderazgo peruano en la industria editorial dedicada a ingeniería, ciencia y arquitectura.

Es responsabilidad de la Universidad Nacional de Ingeniería aportar al Perú un liderazgo de base tecnológica que trabaje en estrecha asociación con las autoridades gubernamentales, los dirigentes empresariales y la sociedad civil en su conjunto, lo cual requiere de una política editorial y de publicaciones que estamos impulsando.

Dr. Ing. Aurelio Padilla Ríos
Rector

A mis padres Luis y Gliceria, gracias por todo.

A María, mi compañera de siempre.

A mis tres hijos, frutos del amor y compromiso con la vida...

Reconocimiento y gratitud a todos mis maestros por su importante apoyo Profesional.
Y en especial al Decano FIIS Mg. Luis Acuña P. y al Ing. Carlos Chafloque E. por su apoyo

Índice

Presentación

En el milenio pasado, como en el actual, se exige constantemente a la universidad elevar su calidad académica, especialmente la formación profesional que está relacionada directamente con la formación del docente universitario.

En general, el paradigma de la docencia universitaria está ligado a dos variables: el dominio y experiencia de la especialidad profesional y el conocimiento y praxis del proceso de enseñanza - aprendizaje en sus asignaturas correspondientes. Dentro de este contexto, felicito a las autoridades de la Universidad y al ente normativo de promocionar actividades estratégicas en el sector académico, tales como la publicación de textos universitarios, actividad estratégica que beneficia a la universidad como al docente, debido que en conjunto se está cumpliendo con el perfil de la universidad: Ciencia Tecnología y Sociedad (CTS).

Respecto al docente, estamos en la responsabilidad de proporcionar un conjunto de enfoques didácticos relacionados con nuestro rol universitario, donde el proceso de enseñanza - aprendizaje, métodos, técnicas y evaluación, engloben las TICS buscando mejoras continuas. En este sentido, como docente de la asignatura "Lenguaje de Programación Estructurada", asignatura obligatoria que forma parte del plan curricular de vuestra Facultad, presento el texto denominado ***"Aplicaciones en Borland C++ 5.02 – Programacion Estructurada"***, organizado y diseñado con base en la experiencia de más de 20 años comprometido con el concepto de programación en lenguajes estructurados.

Esperando que la lectura, análisis y reflexión de la antología que presento sirva para acrecentar el espíritu de renovación institucional, de innovación profesional continua y que obviamente trascenderá al mejor aprendizaje y alta calidad académica de nuestros alumnos, cuando además de la enseñanza presencial, también se dispongan de herramientas didácticas: libros, clases on line, etc.

El autor

Introducción

El comité para el estándar ANSI C se formó en el año 1983 con el objetivo de crear un **lenguaje uniforme** a partir del **Lenguaje de Programación C** original, desarrollado por Kernighan y Ritchie en 1972, en la ATT. Respecto a C++ comenzó a desarrollarse en 1980 por B. Stroustrup. Al comienzo era una extensión del lenguaje C, que fue denominada ***C with classes***. Este nuevo lenguaje comenzó a ser utilizado fuera de la AT&T en 1983. Ante la gran difusión y éxito que iba obteniendo en el mundo de los programadores, la AT&T comenzó a estandarizarlo internamente en 1987. En 1989 se formó un comité ANSI para estandarizarlo a nivel internacional.

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores permite ocupar el primer puesto como **herramienta de desarrollo de aplicaciones**, pues mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, conciso y eficiencia. Además, ha eliminado muchas de las dificultades y limitaciones que tiene C original. La evolución de C++ ha continuado con la aparición de ***Java***, un lenguaje creado simplificando algunas partes de C++ y añadiendo otras, que se utiliza en la presente década para realizar aplicaciones en internet.

El C++ se presenta como:

1. **Lenguaje de programación procedural** (orientado a algoritmos) y por cumplir con las normas de poseer las tres estructuras de control (secuenciales, condicionales/múltiples y repetitivas).

2. **Lenguaje orientado a objetos (*objectoriented programming*, *OOP*)**. Como lenguaje procesal se asemeja al C y es compatible con él. Este nivel de programación admite una filosofía completamente diferente, que exige del programador un completo cambio de mentalidad. Las características propias de la ***programación orientada a objetos***, este tema es el más fascinante, pues C++ cambia radicalmente su pensamiento de programación.

Cabe notar que un programa fuente diseñado a nivel C++, no compila en C, pero lo contrario sí se cumple. Además de las estructuras del control, también disponen de estructuras de datos en forma legible al usuario, tales como: listas o

vectores, tablas o matrices, etc. Asimismo, los registros (struct), uniones, archivos (FILE), programación dinámica (punteros), procesamiento de cadenas basadas fuertemente en funciones (strcmp, struppr, etc.).

El presente texto está basado en experiencias obtenidas por más de década y media, dictando asignaturas en las escuelas de Ingeniería de Sistemas, Computación e Informática de las diferentes universidad del Perú (Universidad Nacional de Ingeniería - Facultad de Ingeniería Industrial y de Sistemas), Universidad de San Martín, Universidad Antúnez de Mayolo, Universidad Tecnológica del Perú, Universidad San Martín de Porres, etc.).

Me es grato agradecer las sugerencias de colegas que, en su debida oportunidad, aportaron con sus opiniones para la mejora del presente libro.

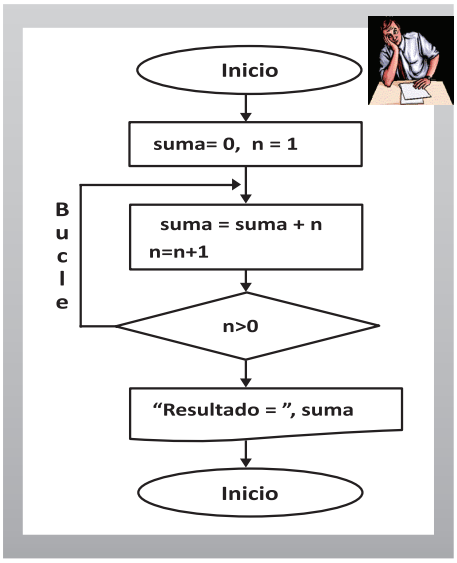
El autor

CAPÍTULO 1

Generalidades

OBJETIVOS

- Conocer la biblioteca standard de Borland C++ 5.02.
- Dar fundamentos para el estudio de otros lenguajes de programación en general y aplicarlos para java, etc.
- Comprender la programación de nivel estructurada.
- Comprender las técnicas básicas para resolver problemas.
- Desarrollar programas usando las técnicas de tipo top-down (de arriba abajo)
- Usar los operadores lógicos, relacionales, funciones resto y parte entera.
- Usar los tipos de datos enteros, reales, cadenas, booleanos.



1.2 INTRODUCCIÓN A LA PROGRAMACIÓN

1.2.1 Definiciones

Computadora. Una computadora (PC) se puede definir como una máquina electrónica digital capaz de procesar información y producir datos de salida, para lo cual requiere de datos de entrada. El término digital se debe al hecho de que la información almacenada y procesada por la computadora está representada mediante códigos numéricos binarios formados por ceros y unos (0 y 1) conocidos como bits , los mismos que con 8 bits, se forma un byte (una palabra).

Debo aclarar, que para los informáticos hay una diferencia entre ***datos*** e ***información***.

Dato. Es la representación de algún hecho, concepto o entidad real.

Información. Es el resultado del procesamiento de los datos.

Observación. Para este curso no haremos distinción entre dato e información sino que hablaremos de datos de entrada y datos de salida.

Proceso de información en una computadora

Una computadora está compuesta por dos elementos fundamentales: **hardware** y **software**.

Hardware. Está constituido por la parte física de la computadora. Es aquello que podemos ver y tocar. Está formado por el monitor, el teclado, el mouse, la unidad del sistema, la impresora, etc.

Software. Es la parte lógica de la computadora y está formado por el conjunto de programas que controlan el funcionamiento de la computadora.

Organización física de una computadora

El software es el conjunto de datos y programas que usa la computadora y se guardan en algún dispositivo del hardware, por ejemplo, un disco duro.

Programa. Es un conjunto detallado de instrucciones que instruyen al procesador para realizar determinados procesos. Los datos pueden ser cualquier información que necesite el programa: caracteres, números, imágenes, etc.

Algoritmo. Es un conjunto de reglas o procedimientos que permiten obtener un resultado determinado a partir de ciertas reglas definidas por el programador.

1.3 Características de un algoritmo de computador

Ser algoritmo: tiene que consistir en una secuencia de instrucciones claras y finitas.

Ser correcto: el algoritmo ha de resolver el problema planteado en todas sus facetas.

Ser legible: el usuario o usuarios deben comprender la sintaxis.

Ser eficiente: es relativa, porque depende de la máquinas en la que lo ejecutemos. Existen ejemplos de algoritmos eficientes que ocupan demasiado espacio para ser aplicados sin almacenamiento secundario lento, lo cual puede anular la eficiencia.

1.4 Lenguajes de programación

Sirven para editar programas (instrucciones). Las instrucciones escritas en la mayoría de los lenguajes de programación no son entendibles directamente

por el procesador, sino que requieren de pasos intermedios de traducción e interpretación para convertir.

Estas instrucciones al lenguaje del procesador, conocido como lenguaje máquina. Podemos citar como lenguajes de programación a Pascal, borlandc++ 5.02, etc.

1.5 Lenguajes de máquina

Permiten escribir instrucciones directamente entendibles por el procesador. Una instrucción máquina consiste en una secuencia de dígitos binarios (0 y 1) en la memoria principal, que le indica al procesador qué operación máquina debe realizar. Una colección de instrucciones máquina en la memoria principal se denomina **programa** en lenguaje máquina.

1.6 Lenguajes de bajo nivel

Los lenguajes de bajo nivel representan un paso hacia la humanización de los lenguajes de programación, son más fáciles que los lenguajes máquina pero, al igual que ellos, son dependientes de la máquina. Los lenguajes de bajo nivel son lenguajes simbólicos, siendo el más importante el lenguaje ensamblador.

1.7 Lenguajes de alto nivel

Los lenguajes de alto nivel son lenguajes humanizados en los que las instrucciones se escriben utilizando frases del inglés cotidiano (o una mezcla de inglés y otro idioma) y contienen notaciones matemáticas de uso común, facilitando así el aprendizaje del lenguaje y la escritura de programas.

1.8 Programa ejecutable

Un programa ejecutable tiene, normalmente, millones de instrucciones y no necesitan del lenguaje de programación para ejecutarse, pues solo basta ejecutar desde el sistema operativo (DOS).

1.9 Compilador

Es una herramienta básica en el mantenimiento y mejora del sistema operativo. Por ello este producto está en constante evolución, ya que de su buen rendimiento depende en parte el del sistema.

1.10 Palabras reservadas

Llamadas también palabras clave, son aquellas que tienen significado especial para el compilador y se utilizan para declarar variables, estructuras, clases, hacer operaciones, definir estructuras de control, etc.

Ejemplos:

break	do	enum	int	typedef
bool	double	for	private	while
case	else	if	sizeof	

1.11 Identificadores

Son secuencias de caracteres que representan a las variables, constantes, tipos, funciones, clases y etiquetas en un programa. En C++, un identificador válido debe cumplir con lo siguiente:

Reglas

13.1. Debe estar formado solamente por letras mayúsculas (de la "A" a la "Z"), o letras minúsculas (de la "a" a la "z") Adicionalmente puede combinarse con el carácter subrayado (_).

Ejemplo : pc, prom_pc, cont, _nombre.

13.2. Usar dígitos del 0 al 9. Debe comenzar con una letra o letras (nunca con números).

Ejemplo: pc_01, prom_pc, cont, Lab1, etc.

13.3. Puede comenzar con un subrayado, pero las palabras que comienzan con dos subrayados son reservadas para el uso interno del compilador.

13.4. No debe contener espacios en blanco, use el subrayado (_) en vez de blancos.

13.5. No usar palabras reservadas del compilador.

Ejemplos de identificadores no válidos:

3pc:	comienza con un número
nombre apellidos:	contiene un espacio en blanco
_i_y:	comienza con dos subrayados
x*y:	contiene un carácter no válido (*)
do:	es una palabra reservada

Nota: C++ es sensible al uso de las mayúsculas y minúsculas. Así, **A** es diferente de **a**.

1.12 Comentarios

Son explicaciones literales que sirven para aumentar la legibilidad de los programas. Estas son ignoradas totalmente por el compilador. En C++, se puede colocar comentarios en cualquier parte del programa donde sea posible, de dos maneras:

14.1 Mediante dos barras inclinadas (//), para comentarios de una línea.

14.2 Por los delimitadores (/*..*/) para comentarios de una o más líneas.

Ejemplo:// comenta respecto a una linea:

/* Comenta respecto a una línea o varias líneas:

1.13 Tipos de datos

En la siguiente tabla se ilustra los diferentes tipos de datos:

Tipo		Tamaño	Rango
Lógico	bool	1 bit	true o false
Enteros	unsigned char	8 bits	0 .. 255
	char	8 bits	–128 .. 127
	enum	16 bits	–32 768 .. 32 767
	unsigned int	16 bits	0 .. 65 535
	short int	16 bits	–32 768 .. 32 767
	int	16 bits	–32 768 .. 32 767
	unsigned long	32 bits	0 .. 4 294 967 295
	long	32 bits	–2 147 483 648 .. 2 147 483 647
Reales	float	32 bits	±3.4E–38 .. ±3.4E+38
	double	64 bits	±1.7E–308 .. ±1.7E+308
	long double	80 bits	±3.4E–4932 .. ±1.1E+4932

1.14 Carácter (char)

Las constantes que representan a caracteres o acciones especiales. Ejemplos: ‘s’, ‘A’, ‘-’,‘|’

‘\a’	alerta (bip sonoro)
‘\b’	retroceso (backspace)
‘\f’	salto de página en la impresora (formfeed)
‘\n’	nueva línea (newline)
‘\r’	retorno de carro (carriagereturn)
‘\t’	tabulación horizontal
‘\v’	tabulación vertical
‘\\’	barra invertida (backslash)

Constantes cadena (char * o char[])

Son secuencias de caracteres encerrados entre comillas:

"Lenguajes de Programacion", "Borland C++ 5.0.2"

También se pueden incluir secuencias de escape:

"\tEstudiar, es una etapa de sabidurias \n"

1.15 Declaración de constantes simbólicas

Mediante una directiva #define:

Sintaxis:

#define nombre valor

Ejemplos:
define pi 3.141
define linea "----------"

Puede hacer que una constante tome un valor en función de otras:
define pi 3.141,
define doble_pi 2*pi

Mediante tipos enumerados:

El tipo **enum** es útil cuando se crean constantes simbólicas con valores enteros.

Ejemplo:
enumdias = {lunes=1,martes,miercoles,jueves,viernes,sabado,domingo}

Además, se puede alterar el orden consecutivo, por ejemplo:

enum colores = {blanco,amarillo,rojo,azul,anterior=64 }

Variables.- Una variable es un identificador válido cuyo valor puede cambiar temporalmente durante la ejecución de un programa. Las variables deben ser declaradas antes de ser utilizadas.

Sintaxis:

Tipo_de_dato nombre_de_variable;

en donde tipo_de_dato es cualquier tipo válido en C++.

Ejemplos
float prom; // prom es un datos de tipo real.

Operadores.- Tienen una gran utilidad durante la programación, considerando su prioridad de evaluación de operadores y su significado. En la siguiente gráfica, se ilustra la sintaxis y el uso respectivo.

Categoría	Operador	Significado	Ejemplo
Negación	!	Negación lógica (NOT)	!(n<11) //si n es menor que 11,devuelve falso (0)
Contador Incremento	++	Preincremento o posincremento	n++ usa n y luego incrementa en 1 ++n // incrementa n en 1 y luego lo usa n
Contador Disminuir	--	Predecremento o posdecremento	--n// n disminuye en 1
Multiplicar	*	Multiplicación	a * b //Multiplica a por b.
Dividir	/	División	a/b // divide a entre b
Resto	%	Módulo (resto)	a%b // devuelve el módulo de la división entera de a y b (que deben ser enteros)
Operadores relacionales	<	Menor que	a < b // devuelve verdadero si a es menor que b. los otros son similares
	<=	Menor o igual que	a <= b
	>	Mayor que	c > d
	>=	Mayor o igual que	d >= e
Igualdad	==	Igualdad	5 == 3 // devuelve falso
	!=	Desigualdad	a!=13
y	&&	y Lógico (AND)	((5<13)&&(5>=4)) //Devuelve verdadero
O	||	oLógico (OR)	(mivar=="hola") || (3<5) //Devuelve verdadero
Condicional	?:	Condicional abreviado	(a<b) ? a++:a-- //Si a<b, a se aumenta en 1 Sino se disminuye en 1
Asignación	=	Asignación	x = 5 //A x se le asigna el valor entero 5 (x ←5)
Expresiones:	*=	Asignar producto	m*=3 //Equivale a m = m * 3
	/=	Asignar división	e/=d //Equivale a e = e / d
	%=	Asignar módulo	t%=2//Equivale a t = t % 2
	+=	Asignar suma	g+=7 //Equivale a g = g + 7

Estructura básica de un programa

// directivas o librerías
include<iostream.h>

```
# define pi 3.14
// Prototipo de funciones
tipo_datonomb_ funcion (lista_argumentos);
     ..................................................
//  Declaracion de variables globales
        inta,b,c;
// Declaración y definición de funciones: Implementación

tipo_datonomb_funcion(lista__de_argumentos)

{ Declaraciones locales
                    <Instrucciones >;
return(expresión_de_tipo_dato);
      }

tipo_nfuncion_n(lista_de_argumentos)
          { <ídem caso de la funcion1>;
          }
/

/Programa principal
voidmain()
 { Declaraciones sobre variables locales al proceso principal
          <Instrucciones>;
                    Nomb_funcion(lista_de_argumentos);
                    ---------------------------------------------
                    funcion_n(lista_de_argumentos);
                    return(0);
          }
```

Como se observa el esquema más empleado para un programa en C++ se compone básicamente por:

1. Directivas
2. Declaración y definición de funciones secundarias definidas por el usuario
3. Una función principal main()

Notas:

1. La función principal o main() regresa un valor 0 al sistema operativo para indicar que el proceso terminó exitosamente, cuando ocurre algún tipo de error regresará algún otro valor entero. Si no se requiere que main devuelva algún valor, se escribe **void**main() y se omite **return**0;
2. Si desea retornar un valor entero, entonces defina el tipo de dato delante de la función respectiva.

```
intmain()
{ return 0;
}
```

Las demás funciones, por lo general son llamadas a ejecución dentro del ámbito de nuestra función principal y por consiguiente, ahí pueden regresar valores.

Observación:
Durante el diseño de los programas solo especificaré una librería de entrada y de salida:

 # include<iostream.h>

Las demás librerías el usuario definirá según su requerimiento.