115 ejercicios resueltos de Programación C++

Desde www.ra-ma.es podrá descargar material adicional.

JOFEBEUS
IRYOPOGU

Ra-Ma®
edU

115 ejercicios resueltos de programación C++

Jorge Fernando Betancourt Uscátegui
Irma Yolanda Polanco Guzmán

Ra-Ma®
edU®
Conocimiento a su alcance
Bogotá - México, D.F.

Betancourt Uscátegui, Jorge Fernando, *et al.*

115 ejercicios resueltos de programación C++/ Jorge Fernando Betancourt Uscátegui e Irma Yolanda Polanco Guzmán --. Bogotá: Ediciones de la U, 2021
518 p. ; 24 cm
ISBN 978-958-792-297-4 e-ISBN 978-958-792-298-1
1. Programación 2. Estructuras secuenciales 3. Cadenas 4. Almacenamiento Tít. 621.39 ed.

Edición original publicada por © Editorial Ra-ma (España)
Edición autorizada a Ediciones de la U para Colombia

Área: Informática
Primera edición: Bogotá, Colombia, septiembre de 2021
ISBN. 978-958-792-297-4

Ediciones de la U es una empresa editorial que, con una visión moderna y estratégica de las tecnologías, desarrolla, promueve, distribuye y comercializa contenidos, herramientas de formación, libros técnicos y profesionales, e-books, e-learning o aprendizaje en línea, realizados por autores con amplia experiencia en las diferentes áreas profesionales e investigativas, para brindar a nuestros usuarios soluciones útiles y prácticas que contribuyan al dominio de sus campos de trabajo y a su mejor desempeño en un mundo global, cambiante y cada vez más competitivo.

Coordinación editorial: Adriana Gutiérrez M.
Carátula: Ediciones de la U
Impresión: DGP Editores SAS
Calle 63 #70D-34, Pbx (57+1) 3203510

Impreso y hecho en Colombia
Printed and made in Colombia

A nuestros padres por sus enseñanzas,
por los valores impartidos y por su
apoyo constante para la superación
tanto personal como profesional
A todos los interesados en acrecentar
su conocimiento en el campo de la
programación y la lógica en la solución
de problemas mediante el lenguaje C++

ÍNDICE

INTRODUCCIÓN

El lenguaje de programación C++ se originó alrededor de la década de los 80 como una extensión del lenguaje C, en términos generales un programa en C++ consta de dos partes. La primera incluye las directivas del preprocesador en la que se especifican las librerías empleadas, la definición de variables globales y constantes y la segunda que corresponde al cuerpo del programa que se identifica con la función main y las llaves de inicio { y de fin } del bloque de instrucciones.

A continuación se describe el contenido de cada uno de los diferentes capítulos, donde al final de los mismos se presentan los ejercicios resueltos. En el capítulo 1 se introduce al lenguaje de programación C++, describe las principales características y generalidades, los elementos básicos de un programa como son las instrucciones o sentencias, los tipos de datos, las bibliotecas o librerías, la definición de constantes simbólicas y literales, el manejo y especificación de comentarios, los tipos de operadores empleados, la definición de variables y sus elementos, las sentencias de entrada y salida, de asignación, palabras reservadas, las sentencias de escape y la tabla ASCII. Además se describe las fases de ejecución de un programa, los tipos de errores y se presenta un ejemplo de un programa.

El capítulo 2 titulado Estructuras secuenciales, comienza con la definición del concepto de sentencia, la descripción de estructura secuencial como el bloque de sentencias ordenadas en forma lógica.

Las estructuras selectivas se describen en el capítulo 3. La toma de decisiones con base en la evaluación de condiciones se delimita mediante la estructura if, estructura if – else o cuando se requiera evaluar diferentes rangos se emplean las estructuras if anidadas.

En el capítulo 4 se describe otra estructura selectiva como lo es la estructura switch. La estructura selectiva switch es útil cuando se conoce con certeza el resultado de una expresión y de acuerdo con el mismo se ejecutan diversas acciones.

Las estructuras repetitivas, iterativas o cíclicas se describen en el capítulo 5. Es claro que cada estructura presenta sus características y se aplican en determinadas condiciones, la estructura ***for*** (para), es útil cuando se conoce el número de iteraciones proporcionadas por la variable de control del ciclo desde un valor inicial hasta un valor final con un paso que puede ser incremento o decremento. La estructura ***while*** (mientras), ejecuta el conjunto de sentencias delimitadas entre las llaves { y }, siempre que se cumpla la condición que controla el ciclo. La estructura ***do – while*** (haga – mientras), es similar a la estructura ***while*** salvo que la condición que controla el ciclo se evalúa al final del bloque, lo que implica que se ejecuta al menos una vez.

El capítulo 6 describe los ***arreglos unidimensionales*** o ***vectores***, es claro que en muchos programas se requiere almacenar un conjunto de datos para luego procesarlos e imprimir los resultados de los cálculos efectuados, se comienza con la definición de vector, la declaración de acuerdo con el tipo de dato, las diferentes formas de inicializar los elementos. El procesamiento de los elementos implica aplicar las estructuras repetitivas o iterativas, de las cuales se emplea la estructura ***for*** por su característica y facilidad para las iteraciones requeridas.

El manejo de ***tablas*** o ***matrices*** mediante ***arreglos bidimensionales*** se describe en el capítulo 7, se comienza con la definición del concepto de arreglo bidimensional, seguido de la forma de declarar las matrices según el tipo de dato, las diversas maneras de inicializar los elementos de la tabla, el procesamiento de los elementos de la matriz se ejecuta mediante estructuras repetitivas for para controlar las filas y para el control de las columnas de la tabla.

El capítulo 8, describe las ***funciones de usuario***, se inicia con la declaración de las variables globales y variables locales, para continuar con la definición de los conceptos de procedimiento y de funciones de usuario. Las funciones de usuario al igual que los procedimientos pueden requerir o no el paso de parámetros para su ejecución, por lo que se describen las dos modalidades básicas: paso de parámetros por valor y paso de parámetros por referencia. La diferencia sustancial entre los procedimientos y las funciones de usuario es que los primeros no retornan valor alguno, por lo cual los resultados de la ejecución de las diversas sentencias se realizan directamente en el procedimiento, en tanto que las funciones de usuario retornan valores de distinto tipo como: int (entero), float (de punto flotante), string (cadena) o double (real).

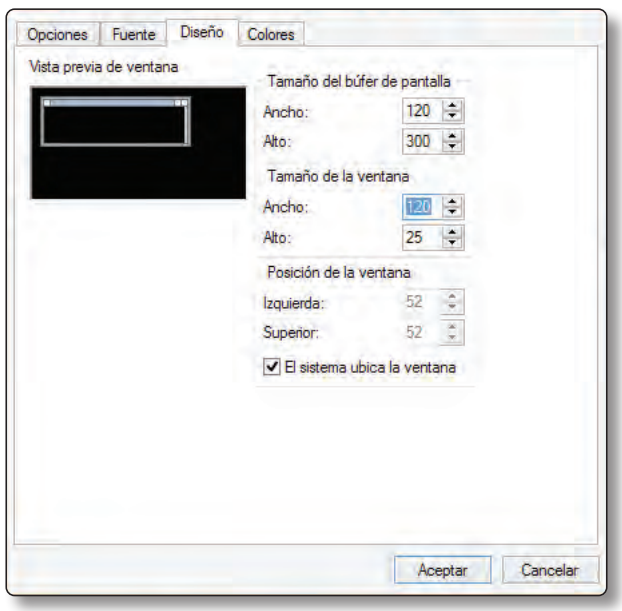
La ***recursividad*** se presenta en el capítulo 9, donde se describe el concepto, las características, los distintos tipos y las ventajas y desventajas de aplicar la recursividad en la solución de problemas relacionados.

En el capítulo 10 se ilustra el ***manejo de cadenas***, se define el concepto de cadena de caracteres, los tipos de datos string, la entrada y salida de cadenas de caracteres y las operaciones predefinidas.

Los ***registros o estructuras*** se ilustran en el capítulo 11, se describe el concepto de registro o estructura, las operaciones completas de asignación, paso como parámetros a procedimientos, estructuras como valor de retorno de funciones. Los miembros como estructuras y las operaciones correspondientes.

El capítulo 12 presenta el ***almacenamiento en memoria secundaria*** mediante el manejo de los ***ficheros o archivos***, se describe los archivos de texto y los binarios, los flujos de entrada y salida asociados a los archivos, la entrada de datos desde archivos de texto y la manipulación de los archivos (visualización, consulta, eliminación y adición de registros)

Por último, el capítulo 13 relacionado con la ***programación orientada a objetos POO***, describe el concepto de clases y sus características, los atributos y métodos, la instanciación, el tratamiento de las clases tanto en línea como a nivel modular, el manejo del concepto de herencia, de polimorfismo y de herencia múltiple.



Para obtener los resultados acorde con las imágenes indicadas en cada ejercicio, es conveniente ***configurar*** el ***tamaño*** de la ***ventana de salida***, para ello, se da clic en el ***icono*** de la ***parte superior izquierda*** de la ***ventana*** y luego en la opción ***propiedades*** de la ***ventana emergente***, se da clic en la pestaña ***Diseño*** y se ***configura*** el ***ancho y el alto de la ventana de salida***, lo recomendable para el ancho es 120 como se indica en la ***figura relacionada***.

1

GENERALIDADES DEL LENGUAJE DE PROGRAMACIÓN C++

1.1 CARACTERÍSTICAS

Creado a mediados de la década de los 80 por Bjarne Strousstrup, C++ es un lenguaje de programación considerado como una extensión del programa C, se caracteriza por abarcar tres aspectos de la programación[1]. (Olivares 2008)

1. Programación estructurada (paradigma de la programación basado en utilizar funciones o subrutinas y sólo tres estructuras de control: secuencia, selección o condicional e iteración o ciclo o bucle)[2]. (Covantec 2018).

2. Programación genérica (Se puede sintetizar en una palabra: **generalización**. Significa que, en la medida de lo posible, los algoritmos deben ser parametrizados al máximo y expresados de la forma más independiente posible de detalles concretos, permitiendo así que puedan servir para la mayor variedad posible de tipos y estructuras de datos)[3]. (Zator 2016)

3. Programación Orientada a Objetos (Se define como un paradigma de la programación, una manera de programar específica, donde se organiza el

1 *https://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf*

2 *https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion5/programacion_estructurada.html*

3 *https://www.zator.com/Cpp/E4_12.htm*

código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones)[4]. (Desarrolloweb 2019).

1.2 ELEMENTOS BÁSICOS DE UN PROGRAMA

En todo programa se distinguen diferentes elementos del lenguaje C++, entre los que se destacan[5]. (Benjumea, V y Roldán, M 2017).

1.2.1 Instrucciones

Son cada una de las sentencias del programa que ejecutan diversas acciones, como la captura de datos, el procesamiento de datos y la salida de datos y resultados. Todas las instrucciones en C++ deben terminar con el símbolo punto y coma (;). Ejemplo de sentencias o instrucciones son:

```
system("cls");                                  //Comando de limpieza de la pantalla
cout << "\n\tDígite el radio del círculo";     //Comando de salida (mensaje de texto)
cin >> ra;                                      //Comando de entrada (captura de
datos)
area = 3.1416 * pow(ra, 2);                     //Comando de procesamiento de datos
```

1.2.2 Datos y tipos

Los programas en C++ pueden procesar diferentes tipos de datos: numéricos, de carácter, booleanos y de cadena, tal como se describen en la tabla 1.

4 *https://desarrolloweb.com/articulos/499.php*

5 *http://www.lcc.uma.es/~vicente/docencia/cppdoc/programacion_cxx.pdf*

Tipo	Rango	Tamaño	Ejemplos
char	-128 a 127	1 Byte	$, ‘a’
string		28 Bytes	“Programa_C++”
bool	True, False	1 Byte	-12 > 12
short	-32768 a 32767	2 Bytes	-1545, 25749
int	-2147483648 a 2147483647	4 Bytes	-1048576, 1245689
long	-2147483648 a 2147483647	4 Bytes	-34502397, 123456
long long	-9223372036854775808 a 9223372036854775807	8 Bytes	-12547389545, 234589735712
unsigned char	0 a 255	1 Byte	227
unsigned short	0 a 65535	2 Bytes	35789
unsigned	0 a 4294967295	4 Bytes	315479234
unsigned long	0 a 4294967295	4 Bytes	40123579
unsigned long long	0 a 18446744073709551615	8 Bytes	112245467832
float	-1.17549435E-38 a 3.40282347E+38	4 Bytes	6.023E23
double	2.2250738585072014E-308 a 1.7976931348623157E+308	8 Bytes	3.141516, 2.7182
long double	3.36210314311209350626E-4932 a 1.18973149535723176502E+4932	12 Byte	-54628935E-150, 12357689342E1250

Tabla 1. Tipos de datos en C++. Fuente. Adaptado de[6] (Benjumea, V y Roldán, M 2017).

1.2.3 Bibliotecas

Una de las principales características del lenguaje de programación C++ es el amplio número de bibliotecas que provee y que se invocan cuando se requiera. En la estructura de un programa, las primeras líneas incluyen las bibliotecas necesarias para el correcto funcionamiento, por tal razón, se escriben tantas líneas ***#include*** como se requiera. Por ejemplo, la línea de código ***#include <iostream>,*** incluye la biblioteca ***iostream*** requerida cuando se establecen operaciones de ***entrada (lectura) o de salida (escritura) de datos***. Para emplear la biblioteca ***iostream*** es necesario utilizar el ***espacio de nombres std*** mediante la instrucción ***using namespace std;.***

6 *http://www.lcc.uma.es/~vicente/docencia/cppdoc/programacion_cxx.pdf*

1.2.4 Constantes simbólicas y constantes literales

El cálculo del área de un círculo se determina mediante la expresión $\pi * r^2$ donde π es el número pi cuyo valor es 3.1416 aproximadamente y r es el radio del círculo. Un fragmento de un programa en C++ que realiza el cálculo del área del círculo puede ser:

```
cout << "\n\tDígite el radio del círculo";
cin >> ra;
area = 3.1416 * pow(ra, 2);
```

Se observa que en el cálculo del área se asigna a la variable ***area*** el producto de multiplicar ***3.1416*** por el resultado de elevar el ***radio (ra) al cuadrado***. En dicho caso se emplea la ***constante 3.1416 de forma literal***.

Una variante del fragmento del programa anterior para el cálculo del área del círculo puede ser:

```
#define pi 3.1416
using namespace std;
int main()
{
   system("cls");            //Comando de limpieza de la pantalla
   system("color 70");       //Comando de configuración de color
   setlocale(LC_ALL, "");
   float ra, area, area1;
   cout << "\n\tLenguaje de programación C++";
   cout << "\n\tDígite el radio del círculo";
   cin >> ra;
   area = pi * pow(ra, 2);
   system("pause");             //Comando de pausa
}
```

En este caso el cálculo del área se realiza mediante la asignación a la variable ***area*** del producto de ***pi*** por el resultado de elevar el ***radio (ra) al cuadrado***, previamente se ha definido el nombre ***pi*** con el valor ***3.1416*** mediante la declaración ***#define pi 3.1416***. Por lo cual, se ha empleado una ***constante simbólica*** en el cálculo del área del círculo.

1.2.5 Identificadores

El identificador se define como el nombre que se asigna a cada elemento que se introduce en un programa en C++ que permite hacer referencia al mismo. El lenguaje C++ diferencia entre minúsculas y mayúsculas, en la siguiente sentencia se definen tres indicadores ra, area y area1 que corresponden a variables de tipo float.

```
float ra, area, area1;
```

1.2.6 Comentarios

En el lenguaje C++, los comentarios permiten hacer anotaciones o documentar partes o secciones de un programa, los comentarios pueden ser definidos en una línea empleando el símbolo `//` para indicar el comienzo o en un bloque para lo cual se requiere los símbolos `/*` al comienzo y `*/` al final del bloque. Las siguientes líneas incluyen comentarios de cada tipo.

```
// Programa que ordena los elementos de un vector de forma ascendente
/* Programa que dado un número ingresado por teclado, retorna su raíz cuadrada,
(emplee la función sqrt) y el cuadrado (emplee la función pow) */
```

1.2.7 Delimitadores

En un programa C++, es usual emplear símbolos como: ( ) { } , ; < > [ ] para especificar el comienzo o fin de una entidad, estos símbolos empleados se conocen como delimitadores. Por ejemplo, el símbolo ; se emplea para indicar el final de una instrucción o sentencia, los símbolos { y } se utilizan para delimitar el comienzo y fin de una función, procedimiento o de un bloque de sentencias.

1.2.8 Operadores

Son símbolos que se utilizan o emplean con significado propio para indicar un tipo de operación determinada. En la tabla 2 se describe el significado de los principales operadores matemáticos, relacionales, lógicos, de asignación e incrementales empleados en el lenguaje de programación C++.

Operadores matemáticos			
Símbolo	**Operación**	**Ejemplo**	**Resultado**
+	Suma	-8 + 23	15
-	Resta	32 - 23	9
*	Multiplicación	12 * 5	60
/	División	34 / 5	6.8
%	Módulo o residuo	34 % 5	4
Operadores relacionales			
<	Menor que	(-20 + 12) < 4 + 2 * 3	- 8 < 10
>	Mayor que	15 > 10 – 3 * (6 – 2 * 4)	15 > 16
<=	Menor o igual	3 / 4 <= 2 * (5 – 3 * 3) / 4	3 / 4 <= - 2
>=	Mayor o igual	25 >= - 2 * (7 – 3 * 5)	25 >= 16
==	Igual	2 + 5 * 4 == 66 – 4 * (6 + 5)	22 == 22
!=	Diferente	(2 – 4 * 6) != - 25 + 3	- 22 != - 22
Operadores lógicos			
&& and	Y lógica	(2 > 3 * 2) && (15 < 20)	False && True (False)
|| or	O lógico	(4 != 3 * 5) || (3 > 4 * 3)	True || False (True)
!	Negación lógica	! (12 – 3 * 2 == 5 * 3 - 9)	! True (False)
Operadores de asignación			
=	Igual	are = 2 * 3.1416 * 10	are = 62.832
+=	Más igual	a = 2; a += 5	a = 7
-=	Menos igual	a = 5; a -= 3	a = 2
*=	Por igual	a = 12; a *= 3	a = 36
/=	Dividido igual	a = 32; a /= 4	a = 8
Operadores incrementales y decrementales unitarios			
++	Incremento	x = 4; x++	x = 5
--	decremento	x = 4; x--	x = 3

Tabla 2. Tipos principales de operadores. Fuente. Elaboración propia.

1.2.9 Variables

En todo programa los datos iniciales y cálculos intermedios deben ser almacenados en memoria en variables de algún tipo, a diferencia de otros programas, en C++ las variables deben ser declaradas antes de utilizarse. Las variables se caracterizan por tener: un nombre que la identifica, un contenido, ser de algún tipo y la dirección en memoria.

Ejemplo 1

En la ejecución del siguiente fragmento de código, sí el usuario digita 10, los elementos de la variable ra son:

```
float ra;
cout << "\n\tDígite el radio del círculo";
cin >> ra;
cout << "\t" << &ra;
```

Nombre	ra
Contenido	10
Tipo de variable	float (real de punto flotante)
Dirección de memoria	001EFA98

Ejemplo 2

En la ejecución del siguiente fragmento de código, sí el usuario digita Fernando, los elementos de la variable nom son:

```
string nom;
cout << "\n\tDigite el nombre del usuario:\t";
cin >> nom;
cout << "\t" << &nom;
```

Nombre	nom
Contenido	Fernando
Tipo de variable	string (cadena)
Dirección de memoria	0108F960

Ejemplo 3

En la ejecución del siguiente fragmento de código, sí el usuario digita 25, los elementos de la variable edad son:

```
int edad;
cout << "\n\tDigite la edad del usuario:\t";
cin >> edad;
cout << "\t" << &edad;
```

Nombre	edad
Contenido	25
Tipo de variable	entero
Dirección de memoria	0108F954

Ejemplo 4

En la ejecución del siguiente fragmento de código, sí el usuario digita M, los elementos de la variable sex son:

```
char sex;
char* pchar;
pchar = &sex;
cout << "\n\tDigite el género del usuario:\t";
cin >> sex;
cout << "\n\tContenido " << sex;
cout << "\n\tMemoria:  " << &pchar;
```

Nombre	sex
Contenido	M
Tipo de variable	char (carácter)
Dirección de memoria	00D9F83C

1.2.10 Entrada / Salida

Por lo general un programa requerirá captar datos de entrada, procesarlos y mostrar los resultados. En el lenguaje C++ se emplea el flujo ***cin*** y el operador de extracción >> para la entrada de datos y ***cout*** con el operador de inserción << para la salida de datos. El siguiente fragmento de un programa ilustra los flujos delimitados.

```
cout << "\n\tDigite el nombre del usuario:\t";
cin >> nom;
cout << "\n\tEl nombre del usuario es:\t" << nom;
cout << "\n\tLa dirección de memoria de la variable nom es:\t" << &nom;
```

La primera línea muestra por pantalla el mensaje "Digite el nombre del usuario:", la segunda permite capturar lo que digita el usuario en la variable nom, las siguientes dos líneas muestran por pantalla el dato digitado por el usuario y la dirección de memoria de la variable nom.

1.2.11 Asignación

El operador de asignación = empleado en una instrucción o sentencia de asignación, permite adjudicar lo que se encuentra a la derecha del operador a la variable ubicada a la izquierda del mismo. A la derecha del operador puede ser un valor constante o el resultado de una expresión aritmética. En el siguiente fragmento de un programa se ilustra diferentes instrucciones de asignación.

```
float ang, anr, fusen, fucos, futan;
anr = ang * 3.1416 / 180;
fusen = sin(anr);
fucos = cos(anr),
futan = tan(anr);
```

En la primera línea se declaran las variables ang, anr, fusen, fucos, futan de tipo float, en la segunda línea se asigna el resultado de operar ang * 3.1416 / 180 a la variable anr, en la tercera línea se asigna el resultado de aplicar la función ***sin(anr)*** a la variable ***fusen***, en la siguiente de aplicar la función ***cos(anr)*** a la variable ***fucos*** y en la última de aplicar la función ***tan(anr)*** a la variable ***futan***.

1.2.12 Palabras reservadas

El lenguaje C++ emplea diversas palabras que tienen significado especial y que obviamente no pueden ser empleadas con otro sentido al crear un programa. Tales palabras hacen referencia a funciones propias del lenguaje, a estructuras o declaraciones de bibliotecas. Algunas de las palabras reservadas se ilustran en la tabla 3.

Palabras reservadas en el lenguaje C++				
main	*cin*	*cout*	*return*	*while*
for	*default*	*sin*	*cos*	*tan*
switch	*string*	*iostrem*	*double*	*float*
rand	*system*	*pause*	*pow*	*fixed*
setprecision	*case*	*break*	*class*	*struct*
private	*public*	*include*	*define*	*char*
int	*boolean*	*long*	*void*	*using*
color	*endl*	*time*	*null*	*and*
do	*namespace*	*sleep*	*srand*	*iomanip*

Tabla 3. Algunas palabras reservadas en el lenguaje C++.
Fuente. Adaptado de[7] (Benjumea, V y Roldán, M 2017).

7 *http://www.lcc.uma.es/~vicente/docencia/cppdoc/programacion_cxx.pdf*

1.2.13 Secuencias de escape

Son caracteres de control que normalmente se emplean para especificar ciertos caracteres especiales dentro de las cadenas de texto. En el lenguaje de programación C++ las secuencias de escape se resumen en la tabla 4, para su ejecución se requiere utilizar el carácter (\) barra invertida y por lo general una letra.

Secuencia de escape	Descripción	Representación
\’	Comilla simple	Byte 0x27
\”	Comilla doble	Byte 0x22
\?	Signo interrogación	Byte 0x3F
\\	Barra invertida	Byte 0x5C
\0	Carácter nulo	Byte 0x00
\a	Pitido	Byte 0x07
\b	Retorno	Byte 0x08
\f	Nueva página	Byte 0x0C
\n	Nueva línea	Byte 0x0A
\r	Retorno de carro	Byte 0x0D
\t	Tabulación horizontal	Byte 0x09
\v	Tabulación vertical	Byte 0x0B
\nnn	Valor en formato octal	Byte nnn
\Xnn	Valor en formato hexadecimal	Byte nn
\Unnnn	Valor en formato Unicode Puede dar como resultado varios caracteres	Código U+nnnn
\Unnnnnnnn	Valor en formato Unicode Puede dar como resultado varios caracteres	Código U+nnnnnnnn

Tabla 4. Secuencias de escape o caracteres de control. Fuente. Obtenido de[8] (cppreference.com 2018).

1.2.14 Tabla ASCII

ASCII es un acrónimo de la expresión inglesa ***American Standard Code for Information Interchange*** que traducido equivale a ***Código Estándar para el Intercambio de Información***. Corresponde a un patrón de codificación numérico empleado para representar 128 caracteres diferentes utilizando una escala decimal del 0 al 127, se requiere para ello **7 bits** (2^7 = 128 permutaciones). Generalmente el código ***ASCII*** se amplifica a **8 bits** (**1 byte**) añadiendo un bit de control, llamado bit de paridad.

8 *https://es.cppreference.com/w/cpp/language/escape*

En la tabla 5, se describe cada uno de los 128 códigos ***ASCII*** codificados en los sistemas numéricos: ***decimal*** (base 10), ***octal*** (base 8), ***hexadecimal*** (base 16) y el carácter (CH) que representa.

dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch
0	0	00	NUL (null)	32	40	20	(space)	64	100	40	@	96	140	60	`
1	1	01	SOH (start of header)	33	41	21	!	65	101	41	A	97	141	61	a
2	2	02	STX (start of text)	34	42	22	“	66	102	42	B	98	142	62	b
3	3	03	ETX (end of text)	35	43	23	#	67	103	43	C	99	143	63	c
4	4	04	EOT (end of transmission)	36	44	24	$	68	104	44	D	100	144	64	d
5	5	05	ENQ (enquiry)	37	45	25	%	69	105	45	E	101	145	65	e
6	6	06	ACK (acknowledge)	38	46	26	&	70	106	46	F	102	146	66	f
7	7	07	BEL (bell)	39	47	27	‘	71	107	47	G	103	147	67	g
8	10	08	BS (backspace)	40	50	28	(	72	110	48	H	104	150	68	h
9	11	09	HT (horizontal tab)	41	51	29	)	73	111	49	I	105	151	69	i
10	12	0a	LF (line feed - new line)	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	0b	VT (vertical tab)	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	0c	FF (form feed - new page)	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	0d	CR (carriage return)	45	55	2d	-	77	115	4d	M	109	155	6d	m
14	16	0e	SO (shift out)	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	0f	SI (shift in)	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	DLE (data link escape)	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	DC1 (device control 1)	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	DC2 (device control 2)	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	DC3 (device control 3)	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	DC4 (device control 4)	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	NAK (negative acknowledge)	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	SYN (synchronous idle)	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	ETB (end of transmission block)	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	CAN (cancel)	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	EM (end of medium)	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1a	SUB (substitute)	58	72	3a	:	90	132	5a	Z	122	172	7a	z
27	33	1b	ESC (escape)	59	73	3b	;	91	133	5b	[	123	173	7b	{
28	34	1c	FS (file separator)	60	74	3c	<	92	134	5c	\	124	174	7c	|
29	35	1d	GS (group separator)	61	75	3d	=	93	135	5d	]	125	175	7d	}
30	36	1e	RS (record separator)	62	76	3e	>	94	136	5e	^	126	176	7e	~
31	37	1f	US (unit separator)	63	77	3f	?	95	137	5f	_	127	177	7f	DEL (delete)

Tabla 5. Tabla ASCII. Fuente. Obtenido de[9] (cppreference.com 2018).

9 *https://es.cppreference.com/w/cpp/language/escape*

1.3 FASES DE LA EJECUCIÓN DE UN PROGRAMA

La conversión de un algoritmo en un programa ejecutable requiere en general de distintas fases, en la tabla 5 se describen algunas de las principales características de las fases.

Fase	Descripción	Archivo de salida	Archivo incorporado
Edición	Se convierte el algoritmo en instrucciones del lenguaje C++ empleando un editor de texto para obtener el código fuente	Archivo de extensión .cpp (código fuente)	
Preproceso	Facilita procesar el código fuente e incluir los archivos de cabecera que provean al compilador la información requerida de las funciones de biblioteca		Archivos de cabecera extensión .h
Compilación	Traduce el código fuente modificado a código de máquina o código objeto	Archivo de extensión .obj (código objeto)	
Enlace	Requiere de un programa llamado enlazador o linkador que une el código objeto de las funciones de biblioteca utilizadas con el código objeto del programa que las utiliza	Archivo de extensión .exe (Archivo ejecutable)	Bibliotecas empleadas
Ejecución	Mediante el archivo .exe generado se pone en marcha el programa		

Tabla 6. Fases en la ejecución de un programa en C++. Fuente. Adaptado de[10] (Carlospes s.f).

1.4 TIPOS DE ERRORES EN LA EJECUCIÓN DE UN PROGRAMA

Es posible que cuando se compile un programa, aparezcan errores que deben ser identificados y corregidos para la correcta ejecución del programa, la ***depuración*** es el proceso mediante el cual se detectan y corrigen los errores resultantes de la fase de compilación en la ejecución de un programa. Los errores pueden ser de diferentes tipos.

1.4.1 Errores de tiempo de compilación

Son errores detectados por el preprocesador, el analizador sintáctico y el propio compilador. Los comunes son errores de sintaxis como se observa en el siguiente fragmento de código.

10 *http://www.carlospes.com/curso_de_lenguaje_c/01_02_fases.php*

```
int edad;
cout << "\n\tDigite la edad del usuario:\t"
cin >> edades;
```

En la segunda línea se presenta el error de falta del carácter ***; (punto y coma)*** que indica el final de la sentencia o instrucción, en la tercera línea el error obedece a que se intenta leer en la variable edades que no está declarada, ya que la variable declarada es edad.

1.4.2 Errores de tiempo de enlazado

Son detectados por el enlazador, ejemplo de este tipo de errores es el llamado a una función cuya definición o estructura no aparece por ningún lado, como se observa en el siguiente fragmento de código.

```
#include <iostream>
#include <string>
#include <windows.h>
#include <time.h>
#include <locale.h>
using namespace std;
void captura();
void procesa();
int main()
{
   system("cls");                    //Comando de limpieza de la pantalla
   system("color 70");               //Comando de configuración de color
   setlocale(LC_ALL, "");
   string nom;
   int edad;
   char sex;
   cout << endl;
   cout << "\t";
   cout << "\n\tLenguaje de programación C++";
   captura();
   system("pause");
   return 0;
}
```

En el fragmento de código se observa que se declara la función ***captura()*** de tipo ***vacío (void)*** que no retorna valor alguno ni requiere parámetros, luego se hace el llamado a la función pero falta la definición de la estructura de la misma. De igual forma se declara la función ***procesa()*** que no retorna valor alguno porque es de tipo ***void***, ni requiere parámetros para su ejecución, pero no se realiza el llamado ni está definida su estructura.

1.4.3 Errores de tiempo de ejecución (runtime)

Son aquellos errores que se producen cuando se ejecuta o corre el programa, son más difíciles de diagnosticar en especial en programas o aplicaciones relativamente grandes. Por ejemplo error de tiempo de ejecución es "un servicio de red no disponible" o "memoria insuficiente".

1.5 EJEMPLO DE UN PROGRAMA

El código de programa que se presenta en la tabla 6, permite al usuario ingresar un número para adivinar un número generado de forma aleatoria simulando el lanzamiento de un dado numerado del uno (1) al seis (6). Los elementos del programa incluyen:

- *Bloque de comentarios*. Las primeras cinco líneas delimitadas por los caracteres /* que determinan el inicio del bloque y */ para indicar el fin del bloque.
- *Directivas del preprocesador*. Líneas de la seis a la diez en la que se definen las bibliotecas utilizadas ***iostream*** (uso del flujo de entrada y de salida), ***string*** (definir variables tipo cadena), ***locale.h*** (uso de caracteres especiales como acentos) y ***time.h*** (biblioteca que contiene funciones para manipular y formatear la fecha y hora del sistema).
- *Prototipo función de usuario*. Se declara la función de usuario ***procesar()*** que no retorna valor ya que es de tipo ***vacío (void)*** y tampoco requiere parámetros para su ejecución.
- *Función principal*. Función ***main*** de tipo entero cuyo bloque de instrucciones está delimitado por los caracteres **{** de inicio y **}** de finalización, contiene un bucle determinado por la sentencia repetitiva **while** que le brinda al usuario la oportunidad de volver a procesar o jugar, además de la llamada a la función de usuario ***procesar***.

- *Estructura de la función de usuario procesar*. La función de usuario procesar es de tipo ***vacío (void)*** por lo que no retorna valor ni requiere parámetros para su ejecución, la función captura el valor ingresado por el usuario validándolo en el rango de 1 a 6, lo compara con el número generado de forma aleatoria y determina sí es igual, en cuyo caso muestra el mensaje "Felicidades, adivinó el número" y muestra el número, en caso contrario le indica al usuario que "No adivinó el número".

Línea	Instrucción / Sentencia	Descripción
1	`/* Programa que simula el juego de adivinar un número en el`	Bloque de comentarios delimitados por los caracteres /* inicio y */ final
2	`lanzamiento de un dado numerado del 1 al 6`	
3	`Autores:  JFBU / IYPG`	
4	`Año:  2020`	
5	`*/`	
6	`#include <iostream>`	Directivas del preprocesador (Bibliotecas y nombre de espacio estándar)
7	`#include <string>`	
8	`#include <locale.h>`	
9	`#include <time.h>`	
10	`using namespace std;`	
11	`void procesar();`	Prototipo función
12	`int main()`	Función principal
13	`{`	
14	`    setlocale(LC_ALL, "");`	
15	`    string seguir = "S";`	
16	`    srand(time(NULL));`	
17	`    while (seguir == "S" or seguir == "s")`	
18	`    {`	
19	`        system("cls");`	
20	`        system("color 70");`	
21	`        procesar();`	Llamada función
22	`        cout << "\n\tDesea continuar jugando S o N\t";`	
23	`        cin >> seguir;`	
24	`    }`	
25	`    cout << endl << endl;`	
26	`    system("pause");`	
27	`    return 0;`	
28	`}`	

Estructura de la función de usuario procesar

```
29      void procesar()
30      {
31          int nc, nu;
32          nc = rand() % 6 + 1;
33          cout << "\n\tDigite el número entre 1 y 6\t";
34          cin >> nu;
35          while (nu < 1 or nu > 6)
36          {
37              cout << "\n\tDato errado, digite el número entre 1 y
    6\t";
38              cin >> nu;
39          }
40          if (nc == nu)
41          {
42              cout << "\n\tFelicidades, adivinó el número:\t" <<
    nc << endl;
43          }
44          else
45              cout << "\n\tNo adivinó el número:\t" << endl;
46      }
```

Tabla 7. Ejemplo de un programa en C++. Fuente. Elaboración propia.