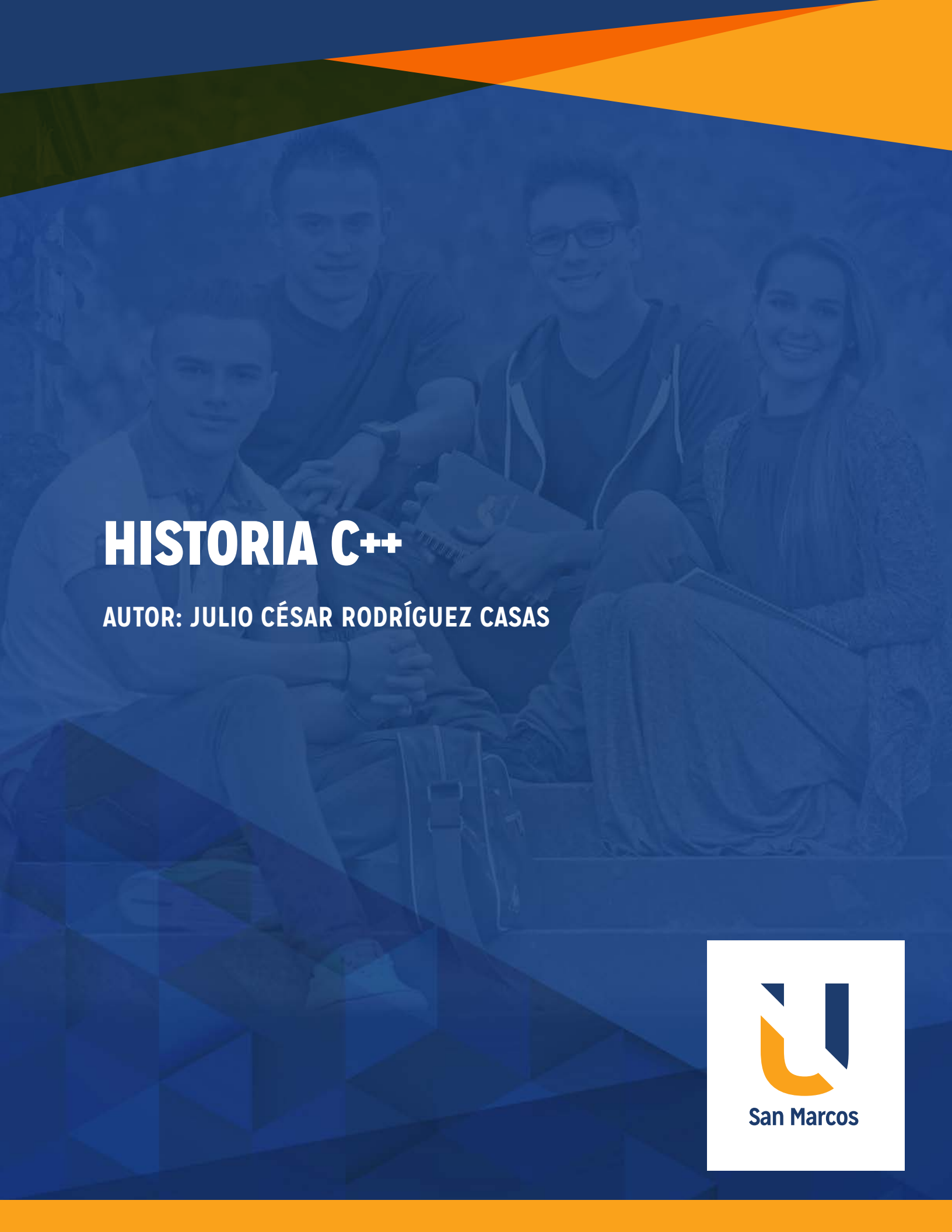

HISTORIA C++

AUTOR: JULIO CÉSAR RODRÍGUEZ CASAS

San Marcos

ÍNDICE

INTRODUCCIÓN

En el presente referente abordaremos los temas relacionados con el lenguaje C++, partiendo desde una breve reseña acerca de la historia del lenguaje C++, teniendo en cuenta los conceptos básicos de C++; así mismo, veremos cómo se puede instalar el compilador del lenguaje C++ con el propósito de desarrollar programas y compilarlos y así mismo corregir los posibles errores de sintaxis que se puedan presentar.

De igual forma, se ofrece la opción de ejecutar los programas online, con una URL para compilar, sin la necesidad de tener instalado el compilador en el computador.

Adicionalmente se realiza la explicación línea por línea del primer programa en C++ para entender cada una de las instrucciones e irse familiarizando con el código.

Para empezar, los invitamos a descargar y tener a mano las siguientes lecturas:

Lectura recomendada

Para reforzar los temas que veremos más adelante se invita al estudiante realizar la lectura complementaria Programación en C++/Lo más básico.

De igual forma se facilita el Manual de programación de C++.

¡Comencemos!

Las principales herramientas necesarias para escribir un programa en C++ son las siguientes:

Un compilador de C++.

Paciencia.

- Nociones sobre programación.
- Un editor cualquiera de texto.

Programación básica
Lenguaje C++

HISTORIA C++

Inicialmente el lenguaje C se realizó en los laboratorios Bell de AT&T en los años 1969 y 1973, en un comienzo se le nombró como "C", la mayoría de sus características se tomaron de un lenguaje llamado "B".

¡Importante!

En 1983 se formó el comité ANSI (Instituto Nacional Americano de Estándares) con el objeto de crear un estándar. Este proceso tuvo una duración de aproximadamente de seis años y a principios de los años 90 es estándar fue reconocido como ISO (Organización Internacional de Estándares) así mismo se comienza a comercializar con el nombre ANSI C.

En forma paralela en 1980 aparece el lenguaje C++, en los laboratorios Bell de AT&T, con el señor Bjarne Strouptrup quien realiza el diseño del lenguaje C++ con el fin de adicionar nuevas características al lenguaje C, como son clases y funciones virtuales tomadas de SIMULA 67, tipos genéricos y expresiones tomadas de lenguaje ADA. Así mismo de ALGOL 68 se tomo la declaración de variables en cualquier parte del programa.

C++ comenzó su proyecto de estandarización ante el comité ANSI y su primera referencia es The Annotated C++ Reference Manual [Ellis 89]. En diciembre de 1989 se reunió el comité X3J16 del ANSI por iniciativa de Hewlett Packard.

Con base en el auge y éxito del lenguaje en el año 1990, se reunieron las organizaciones ISO y ANSI para concretar y definir un estándar con el fin de formalizar el lenguaje. En 1998 termina con la aprobación del ANSI C++.

En junio de 1991, a la estandarización de ANSI se unió ISO (International Organization for Standardization) con su propio comité (ISO-WG-21), creando un esfuerzo común ANSI/ISO para desarrollar un estándar para C++.).

C es un lenguaje de propósito general que se puede utilizar para escribir cualquier tipo de programa, pero su éxito y popularidad está especialmente relacionado con el sistema operativo UNIX. (Fue desarrollado como lenguaje de programación de sistemas, es decir, un lenguaje de programación para escribir **sistemas operativos** y utilidades (programas) del sistema.)

Sistemas operativos
Los sistemas operativos son los programas que gestionan (administran) los recursos de la computadora. Ejemplos bien conocidos de sistemas operativos además de UNIX son MS/DOS, OS/2, MVS, Linux, Windows 95/98, Windows NT, Windows 2000, OS Mac, etc.

La especificación formal del lenguaje C es un documento escrito por Ritchie, titulado *The C Reference Manual*. En 1997, Ritchie y Brian Kernighan, ampliaron ese documento y publicaron un libro referencia del lenguaje *The C Programming Language* también conocido por el K&R.

En la actualidad el lenguaje C++ se ha vuelto muy popular y en se utiliza mucho en las universidades, además es un lenguaje orientado a objetos, de igual forma se puede utilizar como un lenguaje estructurado como su antecesor el lenguaje C y si se quiere trabajar con algoritmos y estructuras de datos.

Instrucción

A este punto, le invitamos a consultar el capítulo 1 "Introducción a la ciencia de la computación y a la programación" de Joyanes, específicamente de la página 38 a 43 para ahondar en la evolución de C++.

Conceptos básicos del lenguaje C++

Funciones

El lenguaje C++ está basado en el concepto de funciones en donde cada una tiene un nombre y una lista de argumentos. En general se puede dar a una función el nombre que se quiera con excepción de MAIN que se reserva para la función que inicia la ejecución del programa.

Una función es un proceso con entradas y salidas bien definidas. Su implementación práctica también debe ir encaminada a la realización de bloques bien definidos en cuanto al proceso que realicen y los insumos y consumos que requiera. Las funciones reciben datos a través de una lista, y devuelve determinada información de un tipo específico. A la lista de datos que la función recibe, se le conoce con el nombre de lista de parámetros.

Las funciones son una de las herramientas más útiles en la programación porque permiten encerrar código con un nombre e invocarlo a través de él. Con esto los programadores evitan repetir el código cada vez que sea necesario en un programa, además de que, es posible ocultar código de forma que el usuario de la función no tenga que conocer los detalles del cómo se hacen las cosas.

Biblioteca Estándar de C++

En C++, la biblioteca estándar es una colección de Clases y funciones, escritas en el núcleo del lenguaje. La biblioteca estándar proporciona varios contenedores genéricos, funciones para utilizar y manipular esos contenedores, funciones objeto, cadenas y flujos genéricos (incluyendo E/S interactiva y de archivos) y soporte para la mayoría de las características del lenguaje. La biblioteca estándar de C++ también incorpora la ISO C90 biblioteca estándar de C. Las características de la biblioteca estándar están declaradas en el espacio de nombres (namespace) std.

La Standard Template Library es un subconjunto de la biblioteca estándar de C++ que contiene los contenedores, algoritmos, iteradores, funciones objeto, etc.; aunque algunas personas utilizan el término STL indistintamente con la biblioteca estándar de C++.

Para profundizar en el tema de programación, se recomienda realizar la lectura y la actividad de las lecturas complementarias; para que pueda instalar y compilar los ejercicios de lenguaje C++.

Otra alternativa para compilar los programas a través de la web en línea es JDoogle. Se recomienda habilitar el modo interactivo.

Conceptos básicos de un programa en C++

Analicemos el siguiente código:

```
#include <iostream>
using namespace std;

int main() {

// aquí escribiremos el código del programa . . .
cout << "Hola Areandina"; << endln;
}
```

Si ejecutamos el programa anterior, no presenta ningún error y nos mostrará el mensaje Hola Areandina.

Estructura del código fuente

Todo el código que escribamos irá siempre dentro de las llaves que tenemos en la función int main () { ...(código)... }.

Vamos ahora a analizar por encima el programa anterior:

Línea 1: #include <iostream>

Las líneas que empiezan por el carácter almohadilla (#), son leídas por lo que se conoce como el "preprocesador". estas son líneas especiales que se ponen antes del código del programa en sí, También se llaman "cabeceras". En este caso #include <iostream>, encarga al preprocesador que se incluya una sección de código C++ estándar que permite realizar operaciones estándar de entrada y salida de datos. Las secciones que se incluyan mediante la almohadilla (#) se llaman también bibliotecas o librerías.

Línea 2: using namespace std;

Video

Antes de continuar, veamos el video sobre Librerías e instrucciones básicas.

Esta línea declara un "namespace" o espacio de nombres. En concreto el namespace "std" o estándar. Dentro del namespace std están declarados todos los elementos de las bibliotecas estándar. Mediante esta línea podemos usar los elementos de la biblioteca estándar sin necesidad de tener que declararlos cada vez que se usan.

Línea 3:

Esta es una línea en blanco, no es imprescindible ponerla, pero ayuda a una mayor claridad en la comprensión del código para el programador. Veremos más adelante el tratamiento de líneas en blanco y espacios en la programación.

Líneas 4 y 7: int main() {

}

Esta estructura es la función principal que contiene el código (main significa "principal"). Lo que se escriba dentro de las llaves de esta función es el código fuente en sí, es decir el programa que se ejecutará. Es posible que pueda haber instrucciones fuera de esta función, pero siempre tendrán que estar dirigidas desde esta función.

Más adelante entraremos a explicar más en profundidad lo que son las funciones, las librerías y otros conceptos que aquí hemos dicho de pasada. De momento hay que saber que para programar en C++ lo primero que debemos hacer es escribir la plantilla que damos aquí. En nuestro ejemplo, dentro de la función int main() { } hemos puesto las siguientes líneas:

Línea 5: Este es un comentario no lo interpreta el compilador.

// aquí escribiremos el código del programa ...

Dentro de las llaves de la función int main() escribiremos el código propiamente dicho, es decir, lo que queremos que se ejecute en el programa.

En este caso hemos puesto un comentario. La doble//barra del principio de línea indica que este es un comentario introducido por el programador. Este no tiene ningún efecto en la programación. Se usan para introducir explicaciones u observacio-

nes. son útiles para documentar el programa e indicar cómo funciona.

Línea 6: cout << "Hola Areandina" << endl;

Esta línea es una sentencia C ++. una sentencia es una instrucción que produce algún efecto o cambia algo en el programa. Las sentencias se ejecutan en el mismo orden en que aparecen. A las sentencias también se les llama declaraciones.

¡Importante!

La sentencia o declaración en un lenguaje de programación, es el equivalente a la frase en el lenguaje humano.

Esta sentencia tiene varias partes:

- En primer lugar, cout, que identifica el dispositivo de salida de caracteres estándar (por lo general la pantalla del ordenador).
- En segundo lugar, el operador de inserción (<<), que indica que lo que sigue debe insertarse (mostrarse en pantalla).
- Por último, una frase entre comillas ("Hola Areandina"), que es el contenido insertado.

Estas dos últimas partes se repiten para volver a insertar otro elemento, es decir repetimos el operador de inserción (**<<**) y después indicamos lo que debe insertarse, en este caso es un salto de línea que se indica mediante la instrucción endl.

Observamos que la declaración termina con un punto y coma **(;)**. Este carácter marca el final de la declaración. Todas las declaraciones o sentencias en C ++ deben terminar con un punto y coma.

Comentarios

Línea 5: Este es un comentario no lo interpreta el compilador.

// aquí escribiremos el código del programa ...

Como hemos visto en el código anterior, los comentarios son anotaciones que hace el programador para sí mismo. El comentario no tiene ningún efecto en la programación, por lo tanto, y a efectos de programación, es ignorado, y el compilador no lo lee.

Sin embargo, puede resultar útil para el programador, como nota aclaratoria que indica lo que se está haciendo.

De una línea

Hay dos tipos de comentarios, el primero, que ya hemos visto, es el comentario de una sola línea.

Para insertarlo escribimos una doble barra, y todo lo que se escriba después de ella en la misma línea es un comentario.

Por ejemplo:

```
// Este es un comentario de una línea.
```

El comentario de una línea no tiene por qué empezar al principio de línea, sino que puede ir en la misma línea después de una declaración, por ejemplo:

```
cout << "Hola Areandina" << endl;// Hola mundo en pantalla.
```

De varias líneas

El segundo tipo de comentario es el de varias líneas, este empieza por los caracteres /*, (barra, asterisco), escribimos después el comentario en una o varias líneas, y para indicar que ha acabado escribimos después los caracteres */ (asterisco, barra). por ejemplo:

```
/* abrimos el comentario.
Sigue el comentario en esta línea...
otra línea más del comentario...
...cerramos el comentario */
```

Todo lo que escribamos entre los signos /* y */, será un comentario de varias líneas y será ignorado por el compilador.

Espacios en blanco

Al escribir el código escribimos una serie de palabras clave, declaraciones, etc.

Como hemos visto en los códigos que hemos hecho hasta ahora, se pueden poner líneas en blanco, más de un espacio entre palabras, o tabulaciones, sin que esto afecte al código.

De hecho, podríamos poner todo el código seguido, si exceptuamos algunas instrucciones que deben ir en una sola línea. Estas son: las que empiezan por #include, y los comentarios de una sola línea. Por ejemplo, podríamos poner el programa del "Hola Areandina" así:

```
//Hola mundo ...
#include <iostream>
using namespace std; int main()
{cout << "Hola Areandina" << endl;}
```

Sin embargo, para una mayor claridad en el código, se suelen utilizar líneas en blanco, tabulaciones, y otros espacios en blanco. De esta manera indicamos qué elementos son principales, y cuales dependen de otros, y los cambios entre las distintas partes del programa. Por ejemplo, el código anterior, bien organizado, quedará así:

```
//Hola mundo
#include <iostream>
using namespace std;

int main() {
  cout << "Hola Areandina" << endl;
}
```

Estructura de un programa C++

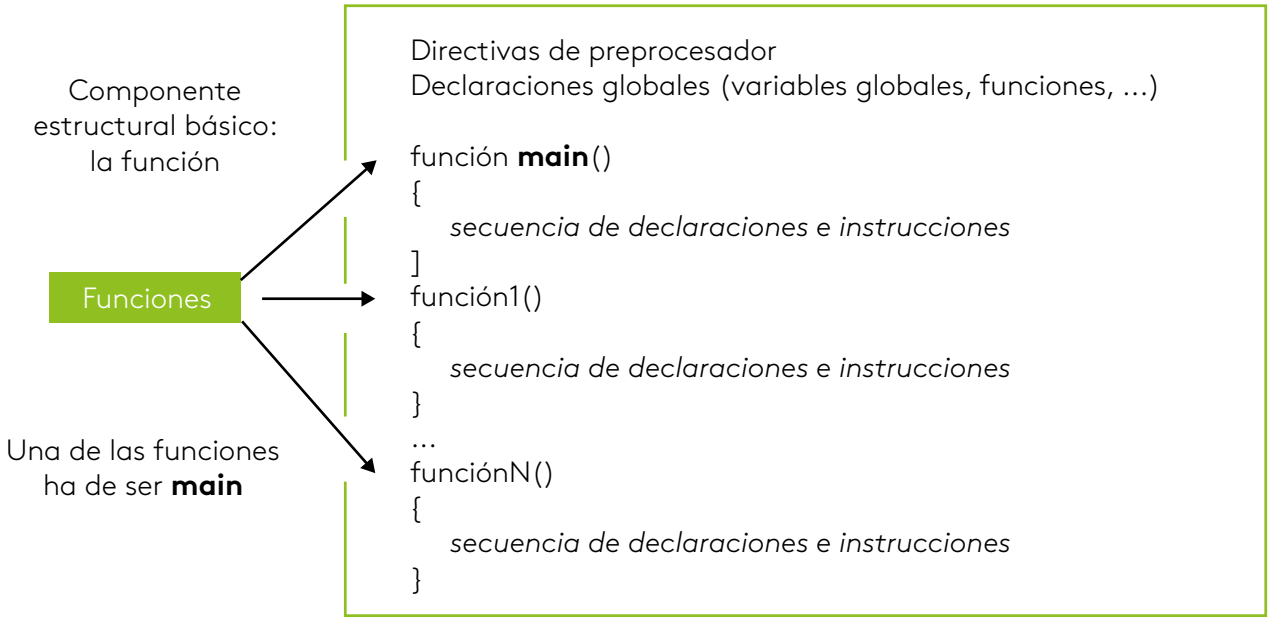

Figura 1. Estructura básica de un programa en lenguaje C++
Tomado de: https://www.yumpu.com/es/document/view/14837159/1-estructura-basica-de-un-programa-c-

return 0;

Todas las funciones deben retornar un valor. Y la función main(), retornará el valor cero (0) si no hubo ningún error. Las funciones deberán retornar el valor especificado al declararlas "**int main ()**" significa que la función **main** devolverá un número entero (a los números enteros se les denomina **int** por "integer" que significa entero en inglés).

Lectura recomendada

Para profundizar en el tema se invita al estudiante a realizar la lectura complementaria acerca de la salida de datos.

Video

Se invita al estudiante a que vea el video acerca de cómo se realiza un programa con base en el algoritmo.

Identificadores y palabras reservadas en el lenguaje C++

Las palabras reservadas son identificadores predefinidos que tienen significados especiales y no pueden usarse como identificadores creados por el usuario en los programas.

Las palabras reservadas de C++ pueden agruparse en tres (3) grupos.

- **Primer grupo:** contiene las palabras de C y que C++ como evolución de C también contiene:

auto	const	double	Break	switch
short	struct	unsigned	Signed	goto
else	For	long	Enum	char
void	case	default	volatile	static
register	sizeof	typedef	Return	
do	extern	If	Int	
union	while	float	continue	

Tabla 1. Primer grupo de palabras reservadas para C++
Fuente: propia

- **Segundo grupo:** contiene palabras que no provienen de C y que, por tanto, solo utiliza C++:

Asm	namespace	try	explicit
new	typeid	false	template
typename	friend	this	const_cast
public	virtual	mutable	true
dynamic_cast	reinterpret_cast	bool	inline
static_cast	catch	operator	wchar_t
class	private	using	
throw	delete	protected	

Tabla 2. Segundo grupo de palabras reservadas para C++
Fuente: propia

- **Tercer grupo:** las siguientes palabras reservadas se han añadido como alternativas para algunos operadores de C++ y hacen los programas más legibles y fáciles de escribir:

not_eq	or_eq	xor_eq
and_eq bitor	not	or
xor		

Tabla 3. Tercer grupo de palabras reservadas para C++
Fuente: propia

Tipos de datos y sus formatos

Los datos en C++

El primer objetivo de un computador es el manejo de información a partir de datos. Estos datos pueden ser costos, calificaciones, temperaturas, presupuestos, datos personales, signos vitales, velocidad, entre otros.

¡Importante!

Los distintos tipos de datos son representados en la memoria del computador de acuerdo con el tipo y el lenguaje de programación que use. Los números enteros en C++, por ejemplo, miden 16 bits o 2 bytes (bit=dígito binario).

El mínimo número que se puede escribir en 16 bits equivale a 16 ceros (0) que al ser convertidos a decimal representan precisamente el valor 0 (cero). El máximo número que se puede escribir en 16 bits son 16 unos que representan el número 65535 decimal. Teniendo en cuenta que hablamos de números sin signo.

La siguiente tabla indica los tipos de datos simple de C++.

TIPO	EJEMPLO	BYTES	RANGO
char	'C'	1	0 a 255
short	-15	2	-128 a 127
int	1024	2	-32768 a 32767
unsigned int	42345	2	0 a 65535
long	262144	4	-2147483648 a 2147483637
float	10.45	4	3.4*(10 e -38) a 3.4*(10 e 38)
double	0.000000000045	8	1.7*(10 e -308) a 1.7*(10 e 308)
long double	1,00E-08	8	Igual que double

Tabla 4. Tipos de datos simple de C++
Fuente: propia

Tipos de datos

Los tipos de datos que maneja C++ son:

- **Enteros:** dentro de los enteros están los tipos: short, int, long, los cuales varían en rango de acuerdo al compilador que se utilice, siendo long rango mayor y short el de menor.

- **Flotantes:** dentro de los flotantes C++ tiene los tipos: float, double y long double donde al igual que los enteros varía el rango de cada uno de acuerdo al compilador que se utilice. De igual forma el float es el de menor rango siendo long double el de rango mayor

- **Caracteres:** se utiliza el tipo char. Para representar un carácter en C++ se utilizan comillas.

  Ejemplos: 'a', 'b' , '05'

  Para representar una cadena de caracteres se utilizan las comillas.

  Ejemplo: "Areandina", "INGENIERÍA DE SISTEMAS"

Declaración de variables

Las variables son elementos clave en todo lenguaje de programación. Se deben declarar diciendo el tipo de dato, el nombre y si es necesario se inicializa a un valor. El compilador separa un espacio en la memoria física del computador para manipular la variable declarada por el usuario

¡Importante!

La declaración de una variable es un estatuto que proporciona información de la variable al compilador de C++.

La sintaxis para la declaración de una variable es:

tipo variable

- tipo: es el nombre de un tipo de dato conocido por C++.
- variable: es un identificador (nombre) válido en C++.

Ejemplo: Declaración de una variable de tipo entero llamada x e inicializada en 100;

```
int  x = 100 ;
```

Veamos un ejemplo de declaración de constante y variables:

```
#include <iostream>
int main( ) {
 const float pi=3.141592;
int radio=5;
float area;
área = pi * radio * radio;
cout << "El área del circulo es: "<< área
<< endl;
return 0; }
```

Expresiones lógicas

Las expresiones lógicas son todas aquellas expresiones que obtienen como resultado verdadero o falso.

Estos operadores unen estas expresiones devolviendo también verdadero o falso. Son:

Operador	Significado
&&	Y (AND)
||	O (OR)
!	NO (NOT)

Tabla 5. Expresiones lógicas
Fuente: propia

Por ejemplo: (18>6) **&&** (20<30) devuelve verdadero (1) ya que la primera expresión (18>6) es verdadera y la segunda (20<30) también.

El operador **Y (&&)** devuelve verdadero cuando las dos expresiones son verdaderas.

El operador **O (||)** devuelve verdadero cuando cualquiera de las dos es verdadero.

Finalmente, el operador NO (!) invierte la lógica de la expresión que le sigue; si la expresión siguiente es verdadera devuelve falso y viceversa.

Por ejemplo !(18>15) devuelve falso (0).

Operadores aritméticos

Los operadores aritméticos se usan para realizar cálculos de aritmética de números reales y de aritmética de punteros. C++ dispone de los siguientes:

Operación	Operador aritmético	Operador en C++
Suma	+	+
Resta	-	-
Multiplicación	X	*
División		/
Módulo	Mod	%

Tabla 6. Operadores aritméticos
Fuente: propia

Video

Antes de continuar, les invitamos a ver el video sobre operadores aritméticos y lógicos.

Operaciones relacionales

Los **operadores relacionales** son símbolos que se usan para comparar dos valores. Si el resultado de la comparación es correcto la expresión considerada es verdadera, en caso contrario es falsa.

Operador	Operación	Ejemplo	Resultado
=	Igual que	hola' = 'lola'	FALSO
<>	Diferente a	a' <> 'b'	VERDADERO
<	Menor que	7 < 15	VERDADERO
>	Mayor que	22 > 11	VERDADERO
<=	Menor o igual que	15 <= 22	VERDADERO
>=	Mayor o igual que	35 >= 20	VERDADERO

Tabla 7. Operadores relacionales
Fuente: propia

Funciones de Entrada y Salida

La mayoría de los programas en C++ incluyen el archivo de encabezado **<iostream>,** el cual contiene la información básica requerida para todas las operaciones de entrada y salida (E/S) de flujo.

Cuando usamos la instrucción:

```
Cout <<"Mensaje a la pantalla"<< endl;
```

Estamos enviando una cadena de caracteres ("Mensaje a la pantalla") al dispositivo de salida estándar (la pantalla). Luego, el manipulador de flujo endl da el efecto de la secuencia de escape '\n', nueva línea.

Se invita al estudiante para que pruebe el siguiente programa:

```
#include <iostream>
int main() {
 cout <<"Hola estudiante Areandina" <<endl;
 cout << 2 + 2 <<endl;//imprime un entero
cout << 9/2 <<endl;//imprime un flotante
cout<<(int) (3.141592+2)<<endl;//imprime un entero
return 0;
}
```

La instrucción **cout <<** puede imprimir tanto números enteros como flotantes sin necesidad de decirle específicamente el tipo de datos del que se trata, pero, por supuesto que al enviarle una cadena de caracteres esta debe de estar entre comillas.

¡Importante!

La interacción con el usuario es algo muy importante en la programación, imaginemos que en este preciso momento y con los conocimientos que tenemos hasta ahora, necesitamos hacer un programa que calcule la distancia a la que caerá un proyectil lanzado a determinada velocidad y ángulo, o simplemente un programa que calcule las raíces de una ecuación cuadrática. Sería muy molesto estar cambiando los valores de las variables directamente en el código para cada caso que queramos calcular. Por eso debemos ver cuanto antes la forma de leer datos desde el teclado.

La principal función para leer desde el teclado es **cin >>**, pero es mejor ver un ejemplo para tener la idea clara.

```
#include <iostream>
using namespace std;
int main() {
   int numero;
   char car;
   float otroNum;
    cout << "Escribe un numero:"<<endl;
    cin >>numero;
    cout <<"\nEl número que ingresó es: «<<numero<<endl;
    cout<<"Digite una letra"<<endl;
    cin>>car;
    cout<<"\nLa letra que ingresó es: "<<car<<endl;
    cout<<"Digite un número de tipo flotante"<<endl;
     cin>>otroNum;
      cout<<"\El número que digitó es: "<<otroNum;
return 0;
}
```

En resumen, **cin** es el flujo de entrada asociado al teclado, **cout** es el flujo de salida estándar asociado a la pantalla, y existe otro, que, aunque a lo largo de este trabajo casi no lo utilicemos, es necesario nombrarlo, cerrar, que es el flujo de error estándar asociado a la pantalla.

¡Importante!

Los operadores << , >> son operadores de inserción y extracción de flujo respectivamente, y no deben confundirse con los de desplazamiento de bits. Estos operadores son muy eficaces porque no es necesario especificar formatos para presentación o lectura, ellos los presentan en función al tipo de datos de la variable. Aunque en ocasiones podría necesitar de nuestra ayuda para obtener los resultados específicos que queremos, y para eso están los modificadores de formato.

Menús, funciones y arreglos

Estructuras de control anidadas

En lenguajes de programación, las **estructuras de control** permiten modificar el flujo de ejecución de las instrucciones de un programa. Con las **estructuras de control** se puede de acuerdo con una condición, ejecutar un grupo u otro de sentencias (If-Then-Else) Si – Entonces – de lo contrario.

Video

Veamos un video sobre estructura Condicional If-Else (Si - De lo contrario)

En nuestra vida cotidiana, todos tenemos una lógica a seguir, continuamente tomamos decisiones, y estas decisiones repercuten en nuestra acción siguiente. Por ejemplo, supongamos el caso de un estudiante de nivel preparatoria que cursa el sexto semestre, él está pensando en presentar el examen para la universidad, sin embargo, sus calificaciones no le han dado mucho aliento últimamente, y está en riesgo de tener que repetir ese semestre, si ocurre eso, el resultado que tenga en el examen no importará. Lo que vaya a pasar marcará el camino a seguir en su vida.

```
#include <iostream>
using namespace std;
int main(){
   float  nota;
   cout<<"Cuál fue tu nota de Algoritmos y Programación?" << endl;
   cin>> nota;

   if (nota >= 3.0)
     cout << "Felicitaciones Aprobó";
   else
     cout << "Lo lamento, debe repetir la metería";

   cin.ignore();
   cin.get();
   return 0;

}
```

Instrucción switch, case y break

Sentencia switch

Se trata de una sentencia que permite construir alternativas múltiples, cuando la estructura condicional no resulta muy cómoda de utilizar. Cuando el programa encuentra un case que es igual al valor de la expresión se ejecutan todos los ***case*** siguientes. Por eso se utiliza la sentencia break para hacer que el programa abandone el bloque switch.

Veamos la sintaxis:

Switch

```
#include <iostream>
using namespace std;
int main() {
int dia;
cout<<"Digite un número de la semana: ";
cin >> dia;
switch(dia)
{ case 1:
  cout <<"Lunes " <<endl;
  break;
  case 2:
  cout <<"Martes" <<endl;
  break;
  case 3:
  cout <<"Miércoles" <<endl;
  break;
  case 4:
  cout <<"Jueves " <<endl;
  break;
  case 5:
  cout <<"viernes" <<endl;
  break;
  case 6:
  cout <<"Sábado" <<endl;
  break;
  case 7:
  cout <<"Domingo" <<endl;
  break;
  default:
  cout<<"Dia de la semana no corresponde"<<endl;
  break;
  }
   return 0;
  }
```

While

Los ciclos while son también una estructura cíclica, que nos permite ejecutar una o varias líneas de código de manera repetitiva sin necesidad de tener un valor inicial e incluso a veces sin siquiera conocer cuándo se va a dar el valor final que esperamos, los ciclos while, no dependen directamente de valores numéricos, sino de valores booleanos, es decir su ejecución depende del valor de verdad de una condición dada, verdadera o falso, nada más. De este modo los ciclos while, son mucho más efectivos para condiciones indeterminadas, que no conocemos cuándo se van a dar a diferencia de los ciclos for, con los cuales se debe tener claro un principio, un final y un tamaño de paso.

```
#include “iostream”
using namespace std;
int main(){
int i = 0;
while(i <= 10)
{
cout << “Soy la variable i, mi valor en esta iteración es: “ << i <<endl;
i++;
}
   return 0;
}
```

Funciones

Definición de una función

Cuando tratamos de resolver un problema, resulta muy útil utilizar la filosofía de “divide y vencerás”. Esta estrategia consiste en dividir nuestro problema en otros más sencillos.

¡Importante!

Cuando realizamos un programa, por ejemplo, en el que se repetirán varias instrucciones, pero con distintos valores que definan los resultados, podemos construir el programa con base en funciones.

Una función es un bloque de instrucciones a las que se les asigna un nombre. Entonces, cada que necesitemos que se ejecuten esa serie de instrucciones, haremos una invocación a la función.

Prototipos de funciones

El prototipo de una función se refiere a la información contenida en la declaración de una función. Una función debe de estar definida o al menos declarada antes de hacer uso de ella.

Cuando se declara una función debe de especificarse el tipo de dato que va a devolver, el nombre de la función, y los parámetros. La siguiente declaración: **int suma(int** a, **int** b);

Especifica una función que devuelve un tipo de dato entero, tiene por nombre suma, y al invocarla, recibe 2 parámetros de tipo entero.

Esta declaración debe de escribirse antes de la función **main**, y su definición puede escribirse después de esta.

Por ejemplo:

```
#include "iostream"
using namespace std;

  int suma(int x, int y);

  int main(){
     int numero1,numero2;
     cout << "CALCULAR LA SUMA DE 2 NUMEROS"<<endl;
     cout << "Ingrese el primer dato:  " <<endl;
     cin >> numero1;
     cout << "Ingrese el segundo dato:  " <<endl;
     cin >> numero2;
     cout << "La suma es: "<< suma(numero1,numero2) <<endl;

     return 0; }
     int suma(int a, int b)
     {
        return a+b;
```

Instrucción

Para finalizar, le invitamos a realizar la actividad de repaso 1, con el propósito de poner en práctica los conceptos y procedimientos que se enunciaron en el referente de pensamiento.

San Marcos

www.usanmarcos.ac.cr

San José, Costa Rica