Ciencias computacionales

Propedéutico: Programación y Estructura de Datos

Contents

1 Programación en C++

En el presente sub-tema se dará una breve introducción al lenguaje de programación C++, se definirán y explicarán las funciones estándar y la estructura para generar un programa.

1.1 Características de lenguaje

El lenguaje de programación C++ fue desarrollado por Bjarne Stroustrup en los Laboratorios Bell en la década de 1980. Este lenguaje es una extensión o mejora del lenguaje de programación C, destacando el soporte a la programación orientada a objetos (OOP, del inglés *Object-Oriented Programming*).

1.1.1 Estructura de un programa en C++

Un programa en C++ puede estructurarse de forma modular, dividiendo las partes esenciales del código en módulos para una mejor organización y comprensión de los programas; ver figura 1.1.1. Los módulos en nuestro programa pueden contener desde definiciones de funciones (prototipos) hasta clases con sus respectivas variables y funciones.

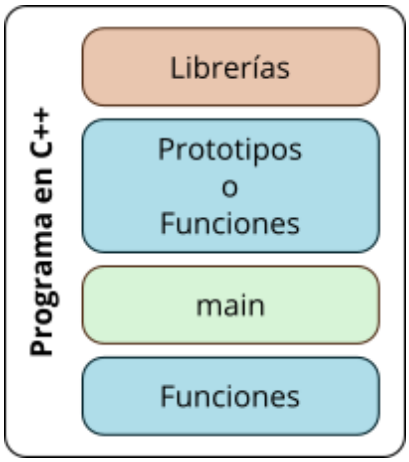

Figure 1: Estructura lógica de un programa en C++.

Para una mejor compresión de la estructura que debemos seguir para nuestros programas en C++, tomemos como ejemplo el código que se muestra en la tabla 1. Como se puede observar en la tabla, el código de ejemplo está estructurado con forme a la figura 1.1.1.

- Librerías (Líneas 1 a 2).
  - Se declaran el conjunto de liberías y archivos de cabecera a utilizar dentro de nuestro programa. Las librerías siempre deberán declararse comenzando por el caracter #, seguido de la palabra *include* y el nombre de la librería entre los signos <*nombre*>.
  - Al final de la declaración de las librerías es común declarar algún espacio de nombres. Los espacios de nombres sirven para agrupar conjuntos de funciones que comparten nombre con funciones ya existentes. El espacio de nombre sólo deberá usarse para evitar poner la referencia completa a un cierta función, en caso contrario podría provocar colisiones dentro de nuestro programa.
    - Ejemplo del espacio de nombres std.
      - Sin declaración previa del espacio de nombres: std::cout
      - Con declaración previa del espacio de nombres: cout
- Prototipos y funciones (Líneas 4 a la 9 y 19 a la 22).

–

- Main (Líneas 11 a la 17). Es la función principal de cada programa escrito en C++, la cuál es ejecutada antes que cualquier otra función. Dentro de cada programa en C++ sólo puede existir una sola función main, ya que cualquier otra declaración de está va a impedir que el compilador pueda procesar nuestro programa.

Librerías	1.	`#include <iostream>`
	2.	`using namespace std;`
	3.	
Funciones	4.	`void funcion1();`
	5.	
	6.	`void function2(void)`
	7.	`{`
	8.	`    \\Cuerpo de la función 2.`
	9.	`}`
	10.	
Main	11.	`int main()`
	12.	`{`
	13.	`    funcion1();`
	14.	`    function2();`
	15.	
	16.	`    return 0;`
	17.	`}`
	18.	
Funciones	19.	`void function1()`
	20.	`{`
	21.	`    \\Cuerpo de la función 1.`
	22.	`}`

Table 1: Ejemplo en código de un programa en C++

1.1.2 Tipos de datos y rangos de valores

Dentro del lenguaje C++ existen tipos de datos predefinidos, los tres principales tipos de datos en el lenguaje son:

- enteros,
- números de punto flotante (reales),
- caracteres:

Cada uno de estos tipos de datos tienen un rango el cuál significa el mínimo y máximo de datos que puede almacenar. A continuación se describen los tipos de datos antes mencionados.

Enteros: El tipo de datos entero es muy familiar y en el lenguaje es representado por la palabra reservada **int**. Los enteros son adecuados para aplicaciones que trabajan con datos numéricos enteros(ver tabla 2).

Caracteres: Es cualquier elemento de un conjunto de caracteres predefinidos. La mayoría de las computadoras utilizan el conjunto de caracteres ASCII, para hacer uso de este tipo de dato de utiliza la palabra reservada **char** (ver tabla 2).

Números depunto flotante: Se usan para para representar números reales que contiene un punto decimal tal como el 3.1416, o números muy grandes, tales como $1.85 * 10^9$. Para este tipo de dato se

pueden usar las palabras reservadas **float**, **double** y **long double**, dependiendo de la precisión de numero a manejar y el tamaño del mismo (ver tabla 3).

Tipo	Tamaño	Rango de valores
char	1 byte (8 bits)	-128 a +127 o de 0 a 255
unsigned char	1 byte (8 bits)	0 a 255
int	2 bytes (16 bits)	-32768 a +32767
unsigned int	2 bytes (16 bits)	0 a 65535
short	2 bytes (16 bits)	-32768 a +32767
unsigned short	2 bytes (16 bits)	0 a 65535
long	4 bytes (32 bits)	-2147483647 a 2147483647
unsigned long	4 bytes (32 bits)	0 to 4294967295

Table 2: Tipos de datos integrales

Tipo	Tamaño	Rango de valores	Exactitud
float	4 byte (32 bits)	-3.4E+38 a 1.2E-38	6 dígitos
double	8 bytes (62 bits)	-1.7E+308 a 2.3E-308	15 dígitos
long double	10 bytes (78 bits)	-1.1E+4932 a 3.4E-4932	19 dígitos

Table 3: Tipos de datos de punto flotante

1.1.3 Librerías estándar

Con al finalidad de mantener el lenguaje C++ lo más sencillo posible, existen funciones preprogramadas que se venden o se instalan junto con el compilador. Estas funciones se encuentran agrupadas en un conjunto de librerías de código, que constituyen a la llamada librería estándar del lenguaje.
En la tabla 4 se muestran las principales librerías que se ocupan dentro del lenguaje C++, mientras que en la tabla 5 se muestran librerías estandar del lenguaje C, mismas que pueden ser utilizadas en el lenguaje C++.

Contenedores			
bitset	deque	list	map
queue	set	stack	vector
Generales			
algorithm	functional	iteratorlocale	
memory	stdexcept	utility	
Manejo de caracteres			
string			
Flujos de entrada y salida			
fstream	ios	iostream	iosfwd
iomanip	istream	ostream	sstream
streambuf			
Numéricas			
complex			
Soporte del lenguaje			
exception	limits	new	typeinf

Table 4: Librerías estándar de C++

cassert	ctype	cerrno	cfloat
climits	cmath	csetjmp	csignal
cstdlib	cstddef	cstdarg	ctime
cstdio	cstring	cwchar	cwctype

Table 5: Librerías estándar del lenguaje C

1.1.4 Operadores

Los operadores son símbolos que tiene como función que el compilador realice una acción. Los operadores actúan sobre los operandos, mismos que son representados mediante expresiones. Esta sección tiene como finalidad explicar los diferentes tipos de operadores existentes los cuales se describen a continuación.

Operadores aritméticos: Los operadores aritméticos sirven para realizar operaciones aritméticas básicas (ver tabla 6). Estos operadores siguen las reglas algebraicas típicas de jerarquía o prioridad, las reglas antes mencionadas se describen en la tabla 7, dentro del lenguaje no existen operadores especiales para la potencia ni raíz cuadradas, debido a que no son consideradas operaciones aritméticas básicas, sin embargo se puede hacer uso de librerías para realizar dichas operaciones.

Operador	Expresión algebraica	Expresión en C
Suma	$f + 7$	`f + 7`
Substracción	$p - c$	`p - c`
Multiplicación	bm	`b * m`
División	$\frac{x}{y}$ $x \div y$	`x / y`
Módulo	$r \ mod \ s$	`r % s`

Table 6: Operadores aritméticos en C++

Operador(es)	Orden de cálculo (precedencia)
Paréntesis	Se caculan primero. Si los paréntesis están anidados,
Multiplicación, división y módulo	Se evaluán en segundo lugar. Si existen varias, se calcularán de izquierda a derecha.
Suma o resta	Se calculan al último. Si existen varios, serán evaluados de izquierda a derecha.

Table 7: Precedencia de operadores en C++

Operadores lógicos: Este tipo de operadores se utilizan cuando se es necesario combinar dos o más expresiones de relación los operadores lógicos existentes se muestran en la tabla 8. Estos operadores devuelven un valor booleano, es decir un verdadero o falso dependiendo del caso.

Operador	Ejemplo	Explicación
&&	a && b	Condición lógica AND. Ejemplo: SÍ a Y b entonces.
||	a || b	Condición lógica OR. Ejemplo: SÍ a O b entonces.
!	!a	Condición lógica NOT. Ejemplo: SÍ NO a entonces.

Table 8: Operadores lógicos en C++

Operadores binarios: Este tipo de operadores son muy parecidos a los operadores lógicos, sin embargo al realizar una operación binaria se afecto a los operandos bit a bit, es decir se aplica una operación lógica a cada uno de los bit que se encuentren en la expresión (ver tabla 9).

Operador	Ejemplo	Explicación
$\sim$	$\sim$a	Operación binaria para negación.
&	a & b	Operación binaria AND.
|	a | b	Operación binaria OR.
$\wedge$	a $\wedge$ b	Operación binaria XOR.
>>	a >> b	Corrimiento de bits a la derecha.
<<	a << b	Corrimiento de bits a la izquierda.

Table 9: Operadores binarios en C++

Operadores asignación: Dentro del lenguaje de programación C++ el símbolo de asignación se encuentra representado por el signo de igual (=). El operador = asigna el valor de la expresión que se encuentra del lado derecho a la variable situada de lado izquierdo tal y como se muestra en la tabla 10, en donde también se muestran ejemplos y descripciones de diferentes formas de utilizar el operador de asignación.
Un aspecto importante a mencionar es que, el operando que se encuentra del lazo izquierdo debe ser exclusivamente una variable, mientras que el de la derecha puede ser una constante, otra variable o alguna expresión.

Operador	Ejemplo	Explicación
=	a = b	Asigna el valor situado a la derecha a la variable situada a la izquierda.
+=	a += b	Al valor situado a la izquierda se le suma el valor de la derecha y el resultado es asignado a la variable situada a la izquierda.
-=	a -=b	Al valor situado a la izquierda se le resta el valor de la derecha y el resultado es asignado a la variable situada a la izquierda.
*=	a *= b	Al valor situado a la izquierda se multiplica por el valor de la derecha y el resultado es asignado a la variable situada a la izquierda.
/=	a /= b	Al valor situado a la izquierda es dividido por el valor de la derecha y el resultado es asignado a la variable situada a la izquierda.
% =	a % = b	Al valor situado a la izquierda es dividido por el valor de la derecha y el resultado es asignado a la variable situada a la izquierda.
>>=	a >>= b	Realiza un corrimiento de bits a la derecha al valor situado a la derecha y el resultado es asignado a la variable situada a la izquierda.
<<=	a <<= b	Realiza un corrimiento de bits a la izquierda al valor situado a la derecha y el resultado es asignado a la variable situada a la izquierda.
&=	a &=b	Realiza la operación lógica AND y asigna el resultado a la variable situada a la izquierda.
|=	a |= b	Realiza la operación lógica OR y asigna el resultado a la variable situada a la izquierda.
$\wedge$ =	a $\wedge$ = b	Realiza la operación lógica XOR y asigna el resultado a la variable situada a la izquierda.

Table 10: Operadores para asignación en C++

1.1.5 Palabras reservadas

Las palabras reservadas o también llamadas palabras clave son identificadores con un significado predefinido. Estas palabras no pueden ser empleadas como nombre de variables o de cualquier otra cosa, debido a que forman parte de la definición del propio lenguaje. Las palabras reservadas son utilizadas por el compilador para controlar alguna acción en el programa, algunas de las palabras reservadas se encuentran en la tabla 11.

asm	do	inline	short	typeid
auto	double	int	signed	typename
bool	dynamic_cast	long	sizeof	union
break	else	mutable	static	unsigned
case	enum	namespace	static_cast	using
catch	explicit	new	struct	virtual
char	extern	operator	switch	void
class	false	private	template	volatile
const	float	protected	this	wchar_t
const_cast	for	public	throw	while
continue	friend	register	true	
default	goto	reinterpret_cast	try	
delete	if	return	typedef	

Table 11: Palabras reservadas en C++

1.1.6 Entrada y salida estándar

Los programas que son creados interactúan con el exterior, a través de datos de entrada o datos de salida. Dentro del lenguaje C++ existe una librería para tener acceso a datos de entrada y salida estándar la cual lleva por nombre **iostream.h** la cual puede insertar o extraer datos utilizando los operadores de inserción $\ll$ y de extracción $\gg$ o utilizando funciones que se encuentran predefinidas en la librería antes mencionada. Esta librería definida para trabajar con diferentes dispositivos tanto de entrada como de salida.

Entrada: La entrada de datos hacia un programa pueden ser provenientes de diferentes fuentes tales como, teclado, archivos en disco, cámaras web, entre otras. Para poder obtener entradas del exterior hacia el programa en necesario utilizar la palabra reservada **cin** $\gg$ a la que se le denomina "entrada estándar".

Salida: La salida de datos de un programa puede ir dirigido a diversos dispositivos tales como: pantalla, impresora o archivos. Por defecto la palabra reservada para realizar está acción es **cout** $\ll$ a la también se le denomina como "salida estándar".

En la tabla 12 se muestra un ejemplo de código en C++ en donde se hace uso de las entradas y salidas estándar ya antes descritas.

```
1.  #include <iostream.h>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      int edad;
7.      cout << Hola!;
8.      cout << ¿Cúal es tu edad?;
9.      cin>> edad;
10.     return 0;
11.
12. }
```

Table 12: Ejemplo en código de un programa en C++ utilizando entrada y salida estándar

References

[DD13] P.J. Deitel and H.M. Deitel. *C++ how to Program.* Deitel, How to Program. Prentice Hall, 2013.

[KPP02] U. Kirch-Prinz and P. Prinz. *A Complete Guide to Programming in C++.* Cs1 - C++ Series. Jones and Bartlett Publishers, 2002.

[Rav11] D. Ravichandran. *Programming with C++.* McGraw-Hill Education (India) Pvt Limited, 2011.

[Str13] B. Stroustrup. *The C++ Programming Language.* Pearson Education, 2013.