Informacion sobre C y Java

Vazquez Rocha Jorge Ivan

3 de febrero de 2020

1. Lenguaje C

1.1. Palabras reservadas

En C, como en cualquier otro lenguaje, existen una serie de palabras clave *(key-words)* que el usuario no puede utilizar como identificadores (nombres de variables y/o de funciones). Estas palabras sirven para indicar al computador que realice una tarea muy determinada (desde evaluar una comparación, hasta definir el tipo de una variable) y tienen un especial significado para el compilador.

A continuación se presenta la lista de 32 palabras clave del ANSII C:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
co	if	static	while

1.2. Tipos de datos básicos o primitivos

Existen Básicamente dos tipos de datos numéricos en ANSII C, Números Enteros y Números Reales o de Punto Flotante, cada uno de ellos subdividido en otros tipos de datos.

Adicionalmente, el tipo de dato char se usa para almacenar caracteres. Sólo uno de los 256 caracteres posibles puede almacenarse en un char a la vez.

	Tipo	Tamaño (bits)	Rango
Enteros	unsigned char	8	0 a 255
	char	8	-128 a 127
	enum	16	-32,768 a 32,767
	unsigned int	16	0 a 65,535
	short int	16	-32,768 a 32,767
	int	16	-32,768 a 32,767
	unsigned long	32	0 a 4,294,967,295
	long	32	-2,147,483,648 a 2,147,483,647

	Tipo	Tamaño (bits)	Rango
Reales	float	32	3.4×10^{-38} a 3.4×10^{38}
	double	64	1.7×10^{-308} a 1.7×10^{308}
	long double	80	3.4×10^{-4932} a 1.1×10^{4932}

1.3. Operadores

A continuación se presenta una tabla con información respecto a los operadores aritméticos:

Operador	Significado	Tipo de Operandos	Tipo de Resultado
+	Suma	Entero o Real	Entero o Real
-	Resta	Entero o Real	Entero o Real
*	Multiplicación	Entero o Real	Entero o Real
/	División	Entero o Real	Entero o Real
%	Residuo de la División Entera	Entero	Entero

Las expresiones con más de dos operandos requieren de reglas matemáticas que aseguren su correcta interpretación.
Por Ejemplo: $3 \cdot 4+5$ Puede ser interpretada como: $(3 \cdot 4)+5 = 17$ ó bien $3 \cdot (4+5) = 27$.
Para evitar confusión, se siguen las siguientes normas de prioridad:

- Operaciones entre paréntesis, se evalúan primero los más internos.
- Operadores ($\cdot$, /).
- Operador (%).
- Operadores (+ , −).

- En caso de operadores de igual prioridad se evalúa la expresión de izquierda a derecha.

Ejemplo: $2 \cdot 2 + 3 \cdot 3 + 6/2\,\%2 = (2 \cdot 2) + (3 \cdot 3) + ((6/2)\,\%2)$

2. Formatos de la funcion *printf()*

La función *printf()* imprime en la unidad de salida (el monitor, por defecto), el texto, y las constantes y variables que se indiquen. La forma general de esta función se puede estudiar viendo su prototipo:

int *printf*(cadena de control, tipo arg1, tipo arg2, ...)

Explicación: La función *printf()* imprime el texto contenido en **cadena de control** junto con el valor de los otros argumentos, de acuerdo con los formatos incluidos en **cadena de control**.
Los puntos suspensivos (...) indican que puede haber un número variable de argumentos.
Cada formato comienza con el carácter (%) y termina con un carácter de conversión.
Considérese el ejemplo siguiente;

```
int i;
double tiempo;
float masa;

printf(Resultado n °: %d En el instante %lf la masa vale %f n, i, tiempo, masa);
```

En el se imprimen 3 variables (i, tiempo y masa) con los formatos (%d, %lf y %f), correspondientes a los tipos (int, double y float), respectivamente.
La **cadena de control** se imprime con el valor de cada variable intercalado en el lugar del formato correspondiente.
Lo importante es considerar que debe haber correspondencia uno a uno (el 1º con el 1º, el 2º con el 2º, etc.) entre los formatos que aparecen en la cadena de control y los otros argumentos (constantes, variables o expresiones).
Entre el carácter % y el carácter de conversión puede haber, por el siguiente orden, uno o varios de los elementos que a continuacin se indican:

- Un número entero positivo, que indica la anchura mínima del campo en caracteres.
- Un signo (-), que indica alineamiento por la izquierda (el defecto es por la

derecha).
- Un punto (.) que separa la anchura de la precisión.
- Un número entero positivo, la precisión, que es el nº máximo de caracteres a imprimir en un string, el nº de decimales de un float o double, o las cifras mínimas de un int o long.
-Un cualificador: una (h) para short o una (l) para long y double

Los caracteres de conversión más usuales se muestran a continuación:

Carácter	Tipo de argumento	Carácter	Tipo de argumento
d, i	int decimal	o	octal unsigned
u	int unsigned	x, X	hex. unsigned
c	Char	s	Cadena de char
f	float notacin decimal	e, g	float not. científica o breve
p	puntero (void *)		

3. Lenguaje Java

3.1. Palabras reservadas

En el lenguaje de programación Java, una palabra clave o palabra reservada tiene un significado predefinido en el lenguaje; debido a esto, los programadores no pueden usar palabras clave como nombres para variables, métodos, clases o como cualquier otro identificador. Debido a sus funciones especiales en el lenguaje, la mayoría de los entornos de desarrollo integrados para Java utilizan el resaltado de sintaxis para mostrar palabras clave en un color diferente para una fácil identificación.[1]

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

[1] https://www.ciberaula.com/cursos/java/palabras_reservadas_java.php

3.2. Tipos de datos básicos o primitivos

Como ya hemos comentado Java es un lenguaje de tipado estático. Es decir, se define el tipo de dato de la variable a la hora de definir esta. Es por ello que todas las variables tendrán un tipo de dato asignado.

El lenguaje Java da de base una serie de tipos de datos primitivos:

byte
short
int
long
float
double
boolean
char

byte
Representa un tipo de dato de 8 bits con signo. De tal manera que puede almacenar los valores numúricos de -128 a 127 (ambos inclusive).

short
Representa un tipo de dato de 16 bits con signo. De esta manera almacena valores numéricos de -32.768 a 32.767.

int
Es un tipo de dato de 32 bits con signo para almacenar valores numéricos. Cuyo valor mínimo es -231 y el valor máximo 231-1.

long
Es un tipo de dato de 64 bits con signo que almacena valores numéricos entre -263 a 263-1.

float
Es un tipo dato para almacenar números en coma flotante con precisión simple de 32 bits.

double
Es un tipo de dato para almacenar números en coma flotante con doble precisión de 64 bits.

boolean
Sirve para definir tipos de datos booleanos. Es decir, aquellos que tienen un valor de true o false. Ocupa 1 bit de información.

char
Es un tipo de datos que representa a un carécter Unicode sencillo de 16 bits.

3.3. Operadores

3.4. Formatos de la clase Formatter”