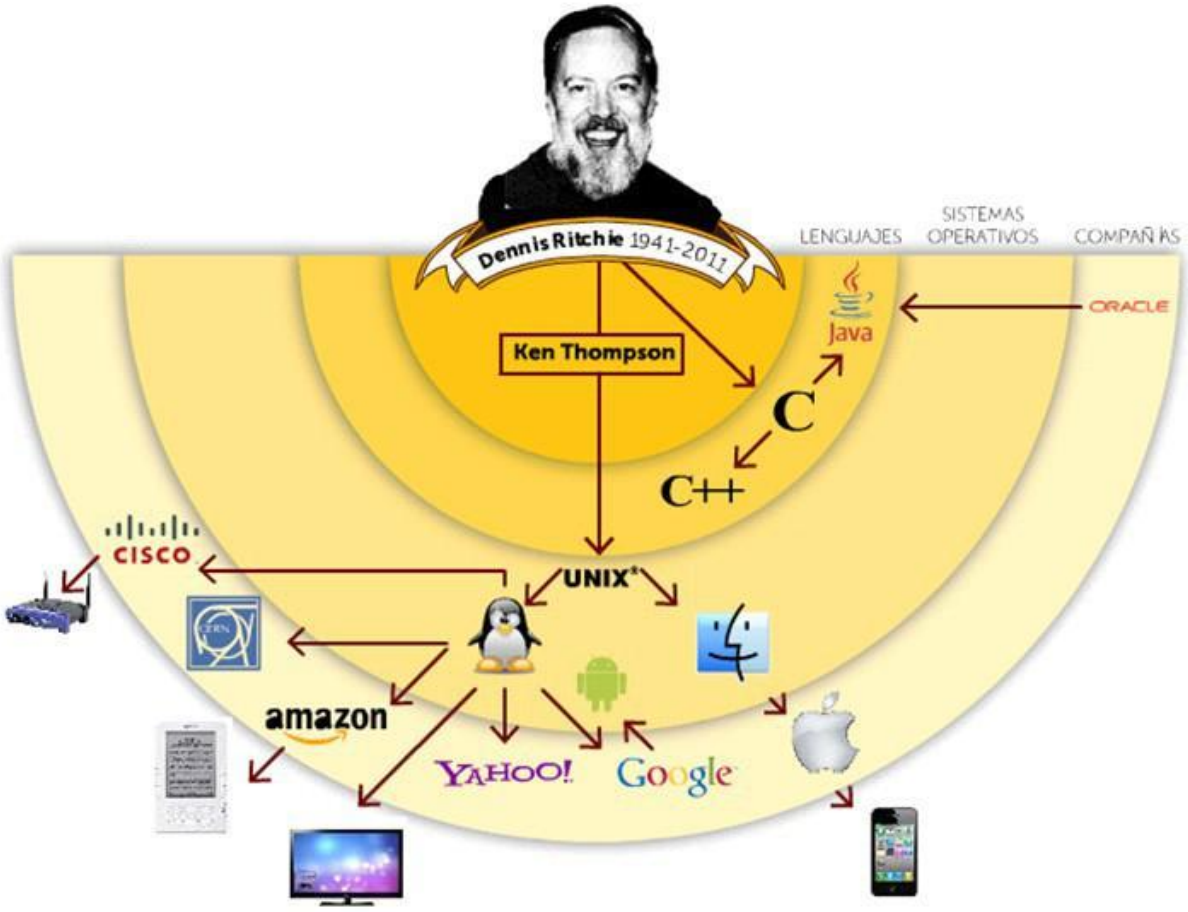
Introducción al lenguaje C.

Laboratorio de Computación I
Tecnicatura Superior en Programación.
UTN-FRRQ

Tabla de Contenido

Introducción

El lenguaje de programación C fue creado por Brian Kernighan y Dennis Ritchie a mediados de los años 70. La primera implementación del mismo la realizó Dennis Ritchie sobre un computador DEC PDP-11 con sistema operativo UNIX. C es el resultado de un proceso de desarrollo que comenzó con un lenguaje anterior, el BCPL, el cual influyó en el desarrollo por parte de Ken Thompson de un lenguaje llamado B, el cual es el antecedente directo del lenguaje C.



El lenguaje C es un lenguaje para programadores en el sentido de que proporciona una gran flexibilidad de programación y una muy baja comprobación de incorrecciones, de forma que el lenguaje deja bajo la responsabilidad del programador acciones que otros lenguajes realizan por sí mismos. Así, por ejemplo, C no comprueba que el índice de referencia de un vector (llamado array en la literatura informática) no sobrepase el tamaño del mismo; que no se escriba en zonas de memoria que no pertenecen al área de datos del programa, etc.

El lenguaje C es un lenguaje estructurado, en el mismo sentido que lo son otros lenguajes de programación tales como el lenguaje Pascal, el Ada o el Modula-2\. Además, el lenguaje C no es rígido en la comprobación de tipos de datos, permitiendo fácilmente la conversión entre diferentes tipos de datos y la asignación entre tipos de datos diferentes, por ejemplo, la expresión siguiente es válida en C:

```
float a; /*Declaro una variable para números reales*/
int b; /*Declaro otra variable para número enteros*/
b=a; /*Asigno a la variable para entera el número real*/
```

Todo programa de C consta, básicamente, de un conjunto de funciones, y una función llamada *main*, la cual es la primera que se ejecuta al comenzar el programa, llamándose desde ella al resto de funciones que compongan nuestro programa.

Desde su creación, surgieron distintas versiones de C, que incluían unas u otras características, palabras reservadas, etc. Este hecho provocó la necesidad de unificar el lenguaje C, y es por ello que surgió un standard de C, llamado ANSI- C, que declara una serie de características, etc., que debe cumplir todo lenguaje C. Por ello, y dado que todo programa que se desarrolle siguiendo el standard ANSI de C será fácilmente portable de un modelo de ordenador a otro modelo de ordenador, y de igual forma de un modelo de compilador a otro, en estos apuntes explicaremos lenguaje C basado en el standard ANSI-C.

El lenguaje C posee un número reducido de palabras reservadas (tan solo 32) que define el standard ANSI-C. Estas palabras reservadas pueden verse en la tabla siguiente:

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

Tabla 1 Palabras reservadas del lenguaje C

1.1 Variables, constantes y operadores

1.1.1 Nombres de identificadores

Se conocen como identificadores a los nombres de variables, funciones y etiquetas definidos por el usuario. La longitud de un identificador en C puede tener entre 1 y 32 caracteres. El primer caracter debe ser una letra o un guion bajo (_) y los siguientes caracteres pueden ser letras, números o guion bajo.

Se debe tener en cuenta que los nombres de identificadores tienen en cuenta las mayúsculas y minúsculas. De esta forma si tenemos 3 variables como min, Min y MIN se las trata como variables distintas.

Por otro lado, es importante destacar 2 puntos importantes para hacer más fácil la interpretación de un programa.
Los nombres de las variables se escriben en minúsculas o con la primera letra en mayúsculas.

Los nombres de las constantes definidas por medio de la directiva del compilador #define se escriben en mayúsculas. De esta forma cuando alguien lee el código fuente de un programa detecta fácilmente cuales son las constantes y cuales las variables.

1.1.2 Tipos de datos

En C existen 5 tipos de datos definidos. En la tabla siguiente:

Tipo de dato	Bits	Rango (con signo)	Rango (sin signo)
char	8	-128 a +127	0 a 255
int (1)	16	-32768 a +32767	0 a 65535
long (2)	32	−2147483647 a +2147483647	0 a 4294967295
float	32	$-3.2x10^{+38}$ a $+3.2x10^{+38}$	
double	64	$-1.7x10^{+308}$ a $+1.7x10^{+308}$	
void	0	sin valor	sin valor

Tabla 2 Tipos de datos del lenguaje C

(1) El tipo *int* debe tener 16 bits **como mínimo**, comúnmente en sistemas operativos para PC regulares, tiene 32 bits

(2) El tipo *long* debe tener 32 bits **como mínimo**, comúnmente en sistemas operativos para PC regulares, tiene 32 bits

Una variable del tipo *char* se utiliza normalmente para almacenar valores definidos en la tabla ASCII o para almacenar cualquier número que se encuentre en el rango mostrado en la tabla. Por ejemplo, para la edad de una persona puede tranquilamente utilizarse una variable del tipo *char*.

Las variables de tipo *int* se utilizan para guardar números enteros y los *float* y *double* se utilizan para números reales.

Ahora pensemos que uno de los datos a ingresar en nuestro programa sea el número de DNI de una persona, ¿Qué tipo de dato utilizará para la variable?

Un número de DNI podría ser 18.458.321 podemos ver que es un número entero por lo tanto en un primer momento podemos pensar en guardarlo en una variable del tipo *int*, pero el valor máximo de un entero sin signo es 65535 con lo cual se ve claramente que el DNI no se puede guardar en una variable tipo *int*.

La edad de una persona siempre es positiva y normalmente menor a 150 años, entonces se elige un dato del tipo *int* donde solo se utiliza menos del 0.5% del rango de un entero.

Para solucionar este tipo de inconvenientes existen los modificadores del tipo de dato:

signed	
unsigned	Aplicable a *char, int* y *long*
short	Aplicable a *int*
long	Aplicable a *int* y a *long*

Tabla 3 Modificadores de tipos de datos del lenguaje C

Por defecto, es decir mientras no se declare explícitamente, las variables son *signed*.

El modificador *unsigned* se utiliza cuando los valores que va a adoptar la variable no llevan signo. Por ejemplo, una variable del tipo *unsigned char* tendrá un rango desde 0 a 255, totalmente apto para guardar una edad.

El modificador *long* (normalmente utilizado con el tipo int) extiende el rango a 32 o 64 bits. Un tipo de dato *long int* es apto para guardar el número de DNI.

1.1.3 Constantes

Las constantes se refieren a valores fijos que no pueden ser cambiados durante el transcurso del programa.

Normalmente en un programa, ya sea por practicidad o por una cuestión de claridad del código fuente, se suelen definir constantes por medio de la directiva define.

De acuerdo al tipo de variable los valores de las constantes se expresan de diferentes formas

Para el tipo *char* la constante se escribe con apóstrofes por ejemplo 'a', '\n','3'

Para el tipo *int* será 123, 5, -2000

Para el tipo *float* y *double* será 125.3, 3.14, -0.05

Para las cadenas de caracteres el texto comienza y termina con comillas (") por ejemplo "hola", "esto es una prueba"

1.1.4 Operadores aritméticos

Un operador es un símbolo que le dice al compilador que realice determinadas operaciones aritméticas o lógicas.

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resto de la división
++	Incremento
--	Decremento

Tabla 4 Operadores aritméticos

1.1.5 Operadores relacionales y lógicos

Operador	Significado
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
==	Igual
i=	Distinto

Tabla 5 Operadores relacionales

Operador	Significado
&&	AND
| |	OR
i	NOT

Tabla 6 Operadores lógicos

1.1.6 Operadores a nivel de bit

Operador	Significado
&	AND
|	OR
^	OR Exclusive
~	Complemento a 1
>>	Desplazamiento a la derecha
<<	Desplazamiento a la izquierda

Tabla 7 Operadores a nivel de bit

7.1 Forma general de un programa

En un programa se pueden encontrar secciones determinadas y un orden para las mismas. La forma que se detalla en la tabla es la más general, esto no quiere decir que todos los programas deben contener cada una de las secciones detalladas, pero si las contiene, debería estar en el lugar que corresponde.

Por ejemplo, la línea ***#include <stdio.h>*** se escribe en todos los programas por el simple hecho de que en algún momento se utiliza la función ***printf*** o ***scanf***.

Tanto ***#include*** como ***#define*** son directivas del compilador que no hacen al lenguaje en sí, pero que son necesarias y deben estar ubicadas en el lugar adecuado.

Si bien las directivas *include* pueden no estar al principio del programa y los *define* no hace falta que se encuentren inmediatamente después de los *include*, es conveniente para que se mantenga un orden en la escritura de todos los programas. El orden propuesto para cada una de las secciones del programa es el que se detalla a continuación.

- Archivos de cabecera	#include <stdio.h>
- Declaración de constantes y macros	#define MAXNOM 20 #define MAXDIR 30 #define MAXPERS 100

- Declaración de estructuras	struct gente {
	char nombre [MAXNOM];
	char dir [MAXDIR];
	int edad;
	};

- Declaración de prototipos de funciones	void CargaGente (struct gente *); int ValidaEdad(void);
- Declaración de variables globales	int flag;

- Desarrollo de la función principal	

void main(void)
 	
{
 	
variables locales de main
 	
------- 	
 	
------ 	
 	
}
	void main(void)
	
	{
	
	struct gente agenda [MAXPERS];
	
	int i, cant;
	
	---------- 	
	
	}

- Desarrollo de otras funciones	

int f1 (char x)
 	
{
 	
Variables locales de f1
 	
}
void f2 (void)
 	
{
 	
Variables locales de f2
 	
}
	void cargaGente (char * p)
	
	{
	
	int i, cant;
	
	}
	int validaEdad(void)
	
	{
	
	int edad;
	
	}

Tabla 8 Partes de un programa