UNIVERSIDAD NACIONAL DE HURLINGHAM

Universidad Nacional de San Martín

Tokens of C - Keywords, Data-Types, Variables, Constants, Operators, Identifiers

Leandro Luciano Gagliardi
lgagliardi@unsam.edu.ar

Caracteres especiales

character	Name	character	Name
+	plus sign	-	minus sign(hyphen)
*	asterisk	%	percent sign
\	backward slash	/	forward slash
<	less than sign	=	equal to sign
>	greater than sign	_	underscore
(	left parenthesis	)	right parenthesis
{	left braces	}	right braces
[	left bracket	]	right bracket
,	comma	.	period
‘	single quotes	“	double quotes
:	colon	;	semicolon
?	question mark	!	exclamation sign
&	ampersand	|	vertical bar
~	tilde sign	^	caret sign
#	hash		

Secuencias de escape / Caracteres de ejecución

Escape sequence	Meaning	ASCII Value	Purpose
\b	backspace	008	Moves the cursor to the previous position of the current line
\a	alert	007	Produces an audible or visible alert
\r	carriage return	013	Moves the cursor to beginning of the current line.
\n	newline	010	Moves the cursor to the beginning of the next line
\f	form feed	012	Moves the cursor to the initial position of the next logical page.
\0	null character	000	Used for termination of character string
\v	vertical tab	011	Moves the cursor to next vertical tab position
\t	Horizontal tab	009	Moves the cursor to the next horizontal tab position.
\\	backslash	092	Presents a character with backslash (\ )

Delimitadores y palabras reservadas

:	Dos puntos	utilizados para etiquetas
;	Punto y coma	fin de la declaración
()	Paréntesis	utilizados en expresiones
[]	Corchetes	utilizados para arrays
{}	Llaves	bloques de código
#	Numeral	directivas de preprocesador
,	Coma	delimitador de variables

auto	**break**	**case**
const	**continue**	**default**
double	**else**	**enum**
float	**for**	**goto**
int	**long**	**register**
short	**signed**	**sizeof**
struct	**switch**	**typedef**
unsigned	**void**	**volatile**

Identificadores

Los identificadores son palabras definidas por el usuario y se utilizan para dar nombres a entidades como variables, arrays, funciones, estructuras, etc. Las reglas para nombrar identificadores se dan a continuación:

- El nombre debe constar únicamente de letras mayúsculas y minúsculas, dígitos y el guión bajo (_).
- El primer caracter debe ser una letra o un guión bajo.
- El nombre no debe ser una palabra reservada.
- Las letras mayúsculas y minúsculas se consideran diferentes.
- El nombre de un identificador puede ser arbitrariamente largo. Sin embargo, algunas implementaciones de C reconocen únicamente los primeros ocho caracteres, aunque los compiladores estándar ANSI reconocen 31 caracteres.

Los identificadores generalmente reciben nombres significativos. Algunos ejemplos:

Valor a net_pay recl _data MARKS

Algunos ejemplos de nombres de identificadores no válidos son:

5bc int rec# avg no

Tipos de datos y calificadores

Hay cuatro **tipos de datos** fundamentales en C, que son **int**, **char**, **float** y **double**. **char** se usa para almacenar cualquier caracter individual, **int** se usa para almacenar un valor entero, **float** se usa para almacenar un número de punto flotante de precisión simple y **double** se usa para almacenar un número de punto flotante de precisión doble.
Hay dos tipos de calificadores de tipo:

- Calificadores de tamaño: **short**, **long**
- Calificadores de signo: **signed**, **unsigned.**

Los calificadores con signo y sin signo se pueden aplicar a los tipos char y entero. El rango de valores para los tipos de datos con signo es menor que el del tipo sin signo.

Para saber el tamaño de un tipo de dato podemos usar la función **sizeof()**.

Tipos de datos y calificadores

El tamaño y el rango de los diferentes tipos de datos en una máquina de 16 bits se muestran en la siguiente tabla. El tamaño y el rango pueden variar en máquinas con diferentes tamaños de palabra.

Basic data types	Data types with type qualifiers	Size(bytes)	Range
char	signed char	1	-128 to 127
	unsigned char	1	0 to 255
int	int or signed int	2	-32768 to 32767
	unsigned int	2	0 to 65535
	short int or signed short int	1	-128 to 127
	unsigned short int	1	0 to 255
	long int or signed long int	4	-2147483648 to 2147483647
	unsigned long int	4	0 to 4294967295
float	float	4	3.4E-38 to 3.4E+38
double	double	8	1.7E-308 to 1.7E+308
	long double	10	3.4E-4932 to 1.1E+4932

Uso de sizeof()

```c
#include <stdio.h>

int main(){

        printf("Size of char:%u\n", sizeof(char));
        printf("Size of unsigned char:%u\n", sizeof(unsigned char));
        printf("Size of signed char:%u\n\n", sizeof(signed char));

        printf("Size of int:%u\n", sizeof(int));

        printf("Size of unsigned int:%u\n", sizeof(unsigned int));
        printf("Size of signed int:%u\n", sizeof(signed int));

        printf("Size of long int:%u\n", sizeof(long int));
        printf("Size of short int:%u\n", sizeof(short int));

        printf("Size of unsigned long int:%u\n", sizeof(unsigned long int));
        printf("Size of unsigned short int:%u\n\n", sizeof(unsigned short int));

        printf("Size of float:%u\n", sizeof(float));
        printf("Size of double:%u\n", sizeof(double));

        printf("Size of long double:%u\n\n", sizeof(long double));

        return 0;
}
```

```
C Programming -> ./a.out
Size of char:1
Size of unsigned char:1
Size of signed char:1

Size of int:4
Size of unsigned int:4
Size of signed int:4
Size of long int:8
Size of short int:2
Size of unsigned long int:8
Size of unsigned short int:2

Size of float:4
Size of double:8
Size of long double:16

C Programming -> vim datatypes.c
```

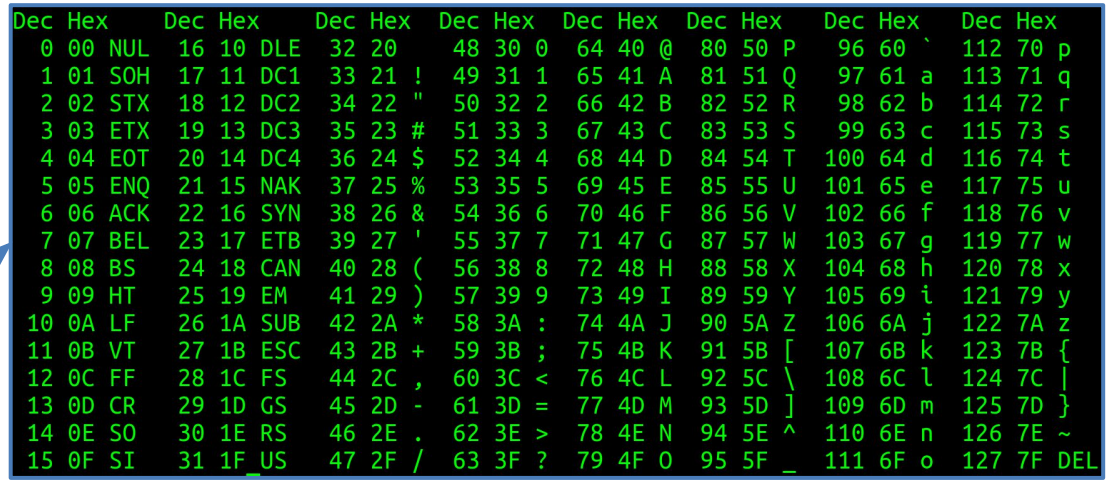
Macros

Enteros: 0, 1, 123, 0123, 064, 0b1001, 0xA024, …
Punto flotante: 0.6, 5.3, 4000.0, 2.5e9, 7.6e-6, -6.3E5, …
Caracter: ‘9’, ‘D’, ‘$’, ‘ ’, ‘#’, …
String: “Leandro”, “ ”, “593”, “A”, …

```
#define MAX 100
#define PI 3.14159625
#define CH 'y'
#define NAME "Leandro"
```

Tabla de caracteres ASCII:

$ sudo apt install ascii
$ ascii



Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex	
0	00	NUL	16	10	DLE	32	20		48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	01	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	EOT	20	14	DC4	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	BS	24	18	CAN	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	HT	25	19	EM	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	0D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

Variables, constantes y expresiones

datatype variablename;

```
int x, x1, x2 = 10;
float salary, salary1 = 0, salary2;
char grade;
long y;
short z;
```

x+y	operación aritmética
a=b+c	utiliza dos operadores (=) y (+)
a>b	expresión relacional
a==b	expresión lógica
func(a, b)	llamada a función

Si queremos definir constantes, basta con usar la palabra reservada **const** antes del tipo de dato:

```
const int c1, c2 = 10;
```

Este dato no puede ser modificado.

Operadores

Operadores aritméticos unarios: +x, -y, …
Operadores aritméticos binarios: a+b, c-d, e*f, g/h, i%j, …
Operador de asignación: k = a+b, a = b = c = d, …
Operadores compuestos: x -= 5, y *= 5, z /= 5, k %= 5 + x, a *= b + 5, …
Operadores de incremento y decremento: ++x, y++, --x, y--, …

```c
#include <stdio.h>

int main(){
        int x = 10;
        printf("Post-increment:\nx = %d\n", x++);
        printf("Pre-increment:\nx = %d\n", ++x);

        return 0;
}
```

```
C Programming -> vim post_pre.c
C Programming -> gcc post_pre.c
C Programming -> ./a.out
Post-increment:
x = 10
Pre-increment:
x = 12
```

Operadores

Operadores relacionales:
Supongamos que a = 9 y b = 5

Expresión	Relación	Valor de la expresión
a < b	falso	0
a <= b	falso	0
a == b	falso	0
a != b	verdadero	1
a > b	verdadero	1
a >= b	verdadero	1
a == 0	falso	0
b != 0	verdadero	1
a > 8	verdadero	1
2 > 4	falso	0

Operadores

Operador & (and):

condición 1	condición 2	resultado
falso	falso	falso
falso	verdadero	falso
verdadero	falso	falso
verdadero	verdadero	verdadero

Operador || (or):

condición 1	condición 2	resultado
falso	falso	falso
falso	verdadero	verdadero
verdadero	falso	verdadero
verdadero	verdadero	verdadero

Operadores

Operador ! (not):

condición	**resultado**
falso	verdadero
verdadero	falso

Operadores

Operador ? (ternario):

Supongamos a = 5 y b = 8:

x = a < b? a : b;

Esto significa que si **a** efectivamente es menor que **b**, **x** va a adquirir el valor de **a**. De lo contrario, adquirirá el valor de **b**.

Operador coma (,):

a = a + b, b = c * d, e = f / g;

Consideremos la siguiente expresión:

sum = (a = 3, b = 2 * a, a + b);

Operadores

```c
#include <stdio.h>

int main(){
        int sum, a, b;
        sum = (a = 3, b = 2 * a, a + b);
        printf("sum = %d\n", sum);

        return 0;
}
```

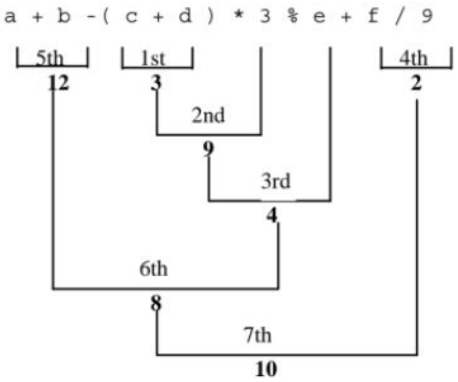
Operadores

```
C Programming -> gcc comma.c
C Programming -> ./a.out
sum = 9
```

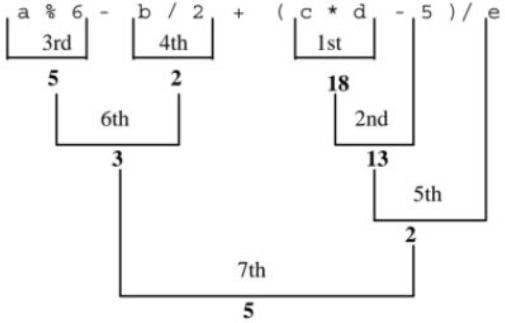
Operadores bit a bit (bitwise): &, |, ^, <<, >>, ~

Operator	Description	Precedence level	Associativity
( ) [ ] → .	Function call Array subscript Arrow operator Dot operator	1	Left to Right
+ - ++ - - ! ~ * & (datatype) sizeof	Unary plus Unary minus Increment Decrement Logical NOT One's complement Indirection Address Type cast Size in bytes	2	*Right to Left*
* / %	Multiplication Division Modulus	3	Left to Right
+ -	Addition Subtraction	4	Left to Right
<< >>	Left shift Right shift	5	Left to Right
< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	6	Left to Right
= = !=	Equal to Not equal to	7	Left to Right
&	Bitwise AND	8	Left to Right
^	Bitwise XOR	9	Left to Right
|	Bitwise OR	10	Left to Right
&&	Logical AND	11	Left to Right
||	Logical OR	12	Left to Right
? :	Conditional operator	13	*Right to Left*
= *= /= %= += - = &= ^= |= <<= >>=	Assignment operators	14	*Right to Left*
,	Comma operator	15	Left to Right

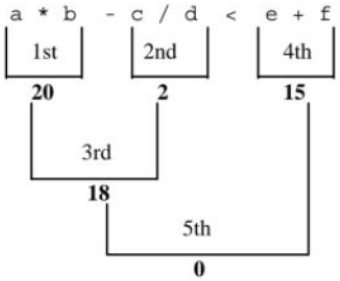
a=8, b=4, c=2, d=1, e=5, f=20
a+b-(c+d)*3%e+f/9



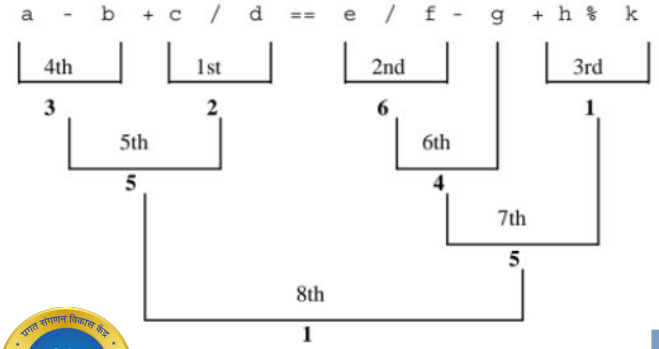
a=17, b=5, c=6, d=3, e=5
a%6-b/2+(c*d-5)/e



a=4, b=5, c=6, d=3, e=5, f=10
a*b-c/d<e+f



a=8, b=5, c=8, d=3, e=65, f=10, g=2, h=5, k=2
a-b+c/d==e/f-g+h%k



1.
```
int main(void)
{
        int a=-3;
        a = -a-a+!a;
        printf("%d\n",a);
        return 0;
}
```

2.
```
int main(void)
{
        int a=2,b=1,c,d;
        c = a<b;
        d = (a>b) && (c<b);
        printf("c=%d, d=%d\n",c,d);
        return 0;
}
```

3.
```
int main(void)
{
        int a=9,b=15,c=16,d=12,e,f;
        e = !(a<b || b<c);
        f = (a>b) ?  a-b: b-a;
        printf("e=%d, f=%d\n",e,f);
        return 0;
}
```

4.
```
int main(void)
{
        int a=5;
        a=6;
        a=a+5*a;
        printf("a=%d\n",a);
        return 0;
}
```

18.

```
int main(void)
{
    int a=10;
    a=a++;
    a = a++ * a--;
    printf("%d\n",a);
    printf("%d\n",a++ * a++);
    return 0;
}
```

19.

```
int main(void)
{
    int a=2, b=2, x, y ;
    x = 4*(++a * 2 + 3);
    y = 4*(b++ * 2 + 3 );
    printf("a=%d, b=%d, x=%d, y=%d\n",a,b,x,y);
    return 0;
}
```

Comentarios

```c
/* Este es un
 * programa de prueba */

// También se puede comentar usando barras dobles.

// En esta sección se encuentrn las directivas del
// preprocesador.
#include <stdio.h>
#define LOOPS   1000000

// Acá podemos declarar variables globales:
long int i;

// También podemos declarar funciones:
void forfunc(void);

int main(){
        long int count = 0;     // Esta es una variable local.
        forfunc();              // Esta es una sentencia y la llamada a la función.
        return 0;
}

// Esta es la definición de la función:
void forfunc(void){
        long int forCount = 0;  // Variable local.
        for(i = 0; i < LOOPS; i++);     // Sentencias
                forCount++;
}
```

UNIVERSIDAD NACIONAL DE HURLINGHAM

Universidad Nacional de San Martín

Extra Slides

Leandro Gagliardi
lgagliardi@unsam.edu.ar