El lenguaje C

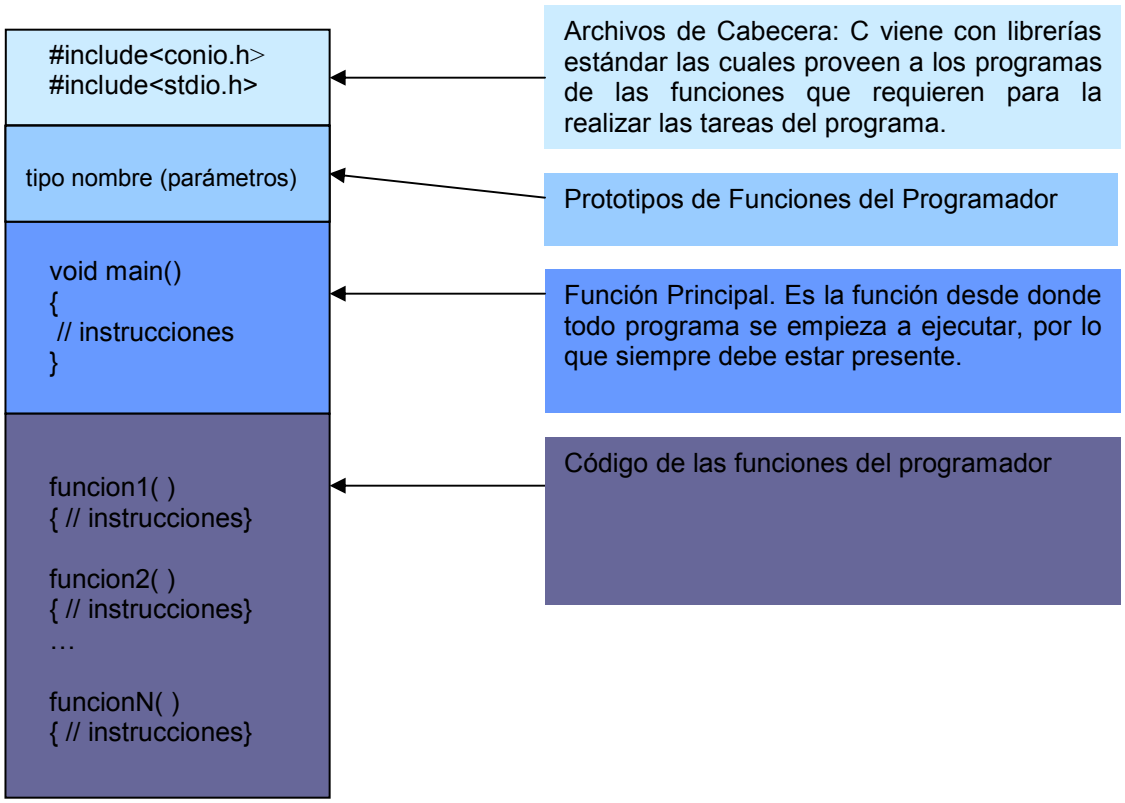
El lenguaje C es uno de los lenguajes de programación estructurada más utilizados en nuestros días. El principal componente estructural de C es la función. En C, las funciones son bloques en los cuales ocurren las actividades de un programa. Esto nos permite separar los programas en tareas, lo que nos conduce a la programación modular.

Otra forma de estructurar en C es usando bloques de códigos. Un bloque de código es un grupo de instrucciones conectadas lógicamente el cual es tratado como una unidad. Un bloque está construido por un conjunto de instrucciones colocadas entre llaves **{ }**. Por ejemplo:

```
if (a>3)
 { printf("hola");       //  bloque
   printf("%d",a);        //    de
   getch();              //código
 }
```

Partes de un programa en C.

La estructura básica de un programa en C, es la siguiente:



Identificadores

Un identificador es el nombre que se le da a una variable, constante, función, etiqueta u otro objeto utilizado en C. Un identificador en C puede variar entre 1 y 32 caracteres. El primer carácter debe ser una letra o un subrayado (_) seguido de una secuencia de caracteres siendo letras, numero o subrayados.

Ejemplo de identificadores:

correcto	incorrecto
Mexicali	3f32
cont	pot*10
sueldo	Fact!
f_32	Num 3
Fila12s	Dia..9

Tipos de datos

Existen 5 tipos de datos importantes en C:

Tipo	Descripción	Tamaño en bytes	Rango
char	carácter	1 byte	0 al 255
int	Números enteros	2 bytes	-32768 al 32767
float	Números Reales	4 bytes	3.4E-38 al 3.4E+38
double	Números Reales	8 bytes	1.7E-308 al 1.7E+308
void	Nulo	0 bytes	sin valores

Ejemplos de datos que se guardan en las variables:

char : 'a', 'B', '7', '*', '/n';
int: 89, 0, -78
float: 1.67, 9.0, -89.67E-9;
double: 789.97E300, 1.5

Existen igualmente algunos modificadores que en unión con los tipos de datos básicos aumentan la cantidad de ellos, estos son: signed, unsigned, long y short.

Tipo	Rango	Tamaño en bytes
unsigned char	0 al 255	1
signed char	-128 al 127	1
unsigned int	0 al 65535	2
signed int	-32768 al 32767	2
short int	-32768 al 32767	2
unsigned short int	0 al 65535	2
signed short int	-32768 al 32767	2
long int	-2147483648 al 2147483647	4
unsigned long int	0 al 4294967295	4
signed long int	-2147483648 al 2147483647	4
long double	1.7E-308 al 1.7E+308	8

El lenguaje C tiene 43 palabras reservadas que en combinación con la sintaxis formal de C forman el Lenguaje de programación C. Lista que a continuación se muestra:

auto	enum	short	volatile	_ss
double	register	unsigned	do	cdecl
int	typedef	continue	if	far
struct	char	for	static	huge
break	extern	signed	while	interrupt
else	return	void	asm	near
long	union	default	_cs	pascal
switch	const	goto	_ds	
case	float	sizeof	_es	

En C las mayúsculas y minúsculas se consideran diferentes, esto es, case es diferente de Case y CASE Las palabras reservadas no puede ser utilizadas como nombres de variable o nombre de función.

Declaración de las variables

Todas las variables deben ser declaradas antes de ser utilizadas de la siguiente manera:

<tipo de dato> <nombre de variable>;

ejemplo: int numero;

donde int es el tipo de dato y numero es el nombre que se le dio a la variable que guarda un dato de tipo entero. Toda variable local debe ser declarada dentro de la función en la cual va a operar y son declaradas siempre inmediatamente después de una llave que abre '{'.

Instrucciones en C

La instrucción básica en C es la asignación. Tiene la forma:

variable = expresión;

en donde la expresión puede ser desde una constante hasta operaciones mas complejas.

ejemplos:
numero = 38
numero = b* 24
numero = pow(5, 4);

Operadores Aritméticos

+	suma
*	multiplicación
-	resta
/	división
%	división modular
--	decremento en una unidad
++	incremento en una unidad

Jerarquía de los operadores aritméticos

El orden en se realizan las operaciones aritméticas o jerarquía de los operadores aritméticos es:

( )	mayor jerarquía
++, --	
*, /, %	
+, -	menor jerarquía

Por ejemplo en la siguiente ecuación la expresión se evaluará de la forma siguiente:

```
a = 5 + 4 / 2 - 3 % 5;
```

primero se realiza la operación 4 / 2, después 3 % 5 y por último se realiza 5 + resultado de la división 4/2 - resultado de la división modular 3%5 :

```
a = 5 + 2 - 3 % 5
a = 5 + 2 - 3
a = 7 - 3
a = 4
```

Ahora bien también se utilizan los paréntesis como símbolos de agrupación y los cuales tienen la mayor jerarquía, esto es, primero se realiza lo que se encuentre encerrado entre ellos y si existen paréntesis dentro de paréntesis primero se realizan los internos y posteriormente los externos

```
a = ( 5 + 4 ) / 3;   primero se realiza la suma y después la división.
```

Salida y entrada de datos

Las operaciones de entrada y salida de datos se realizan mediante funciones, funciones que deben ser llamadas de alguna parte, la cual es conocida como librerías o bibliotecas: en C las funciones de salida y entrada son: printf(), para salida y scanf(), para entrada. Ambas están definidas en la librería stdio.h, por lo que hay que incluirla en el programa.

Salida de Datos

El comando más utilizado para salida de datos en modo texto es printf. printf tiene varias formas de uso:

1) Solo mostrar texto. En este caso la sintaxis es la siguiente:

```
printf ("Mensaje que desea enviar a la pantalla");
```

2) Si el mensaje incluye valores provenientes de variables o expresiones, entonces es necesario usar uno o más especificadores de formato. La sintaxis es

```
printf("Mensaje %<especificador>", variable o expresión);
```

donde <especificador> se sustituye por un especificador de formato apropiado para el tipo de variable que se desee incluir en el mensaje.

La siguiente tabla muestra los especificadores de formato más comunes:

Especificador	Descripción
%c	un solo carácter
%d	un valor entero decimal
%f	un valor fraccionario
%e	un valor fraccionario expresado en notación científica con e minúscula.
%E	un valor fraccionario expresado en notación científica con E mayúscula.
%g	escoge entre %e y %f el que sea mas corto
%G	escoge entre %e y %f el que sea mas corto en mayúscula
%o	un número en octal (base ocho)
%s	una cadena de caracteres
%u	un entero decimal sin signo
%x	un número en hexadecimal (base 16) con las letras en minúsculas.
%X	un número en hexadecimal (base 16) con la letras en mayúsculas

Ejemplos:

impresión de un mensaje: printf("Como te llamas?:");

impresión del contenido de la variable float resultado: printf("El resultado es: %f" , r);

impresión de una constante carácter: printf("El resultado es: %c", 164);

impresión de una variable carácter: printf("El resultado es: %c", letra);

impresión de una expresión (operación): printf("El resultado es: %d", 5*3);

Existen algunos caracteres que no pueden mandarse a pantalla directamente, para esto se usa la secuencia escape

Secuencia	Descripción
\b	Backscape.
\n	brinco de línea.
\r	Retorno de carro
\t	realiza una tabulación
\"	Imprime comillas
\'	Imprime el apostrófe
\\	Imprime diagonal
\a	bocina
\f	Avance de página

Por ejemplo: printf("la dirección es http:\\\\fcqi.uabc.mx\\docentes\\palacios");

Con la línea anterior lo que se observa en la pantalla es:

http:\\fcqi.uabc.mx\docentes\palacios

Entrada de datos

La captura de datos desde teclado se efectúa con el comando **scanf,** el cual tiene el siguiente formato:

scanf("Especificador de formato", &variable);

el & (ampersand) representa la dirección en memoria de la variable por lo que siempre debe incluirse, excepto cuando se va a capturar una variable de tipo caracteres. Ejemplo:

```
scanf("%d",&alumnos);
scanf("%s", nombre);
```

Especificador	Tipo de Dato
%c	char
%d	int
%f	float
%ld	long int
%lf	double

de igual manera que el printf(), debe haber correspondencia de tipo y cantidad entre las variables y los comandos de formato y las variables.

Nota: dentro del formato de entrada NO se pueden incluir mensajes.....solo comandos de formato, si se requiere enviar mensajes se tiene que utilizar el printf para enviar dichos mensajes.