**RESUMEN C**

Características:

C es un lenguaje procedural, de estilo estructurado y sintaxis estricta. A diferencia de Python, es compilado. Es decir, que no es interpretado, no cuenta con un interprete para ir escribiendo el código y que lo evalúe.

Para crear un archivo, solo basta un identificador junto con la extensión “.c”

<identificador>.c.

La primera línea que debe/n aparecer es la carga de las librerías. Como principal aparece <stdio.h> que contiene funciones de entrada y salida.

Siempre nuestro archivo de C debe contener una función main, que es la que se ejecuta cuando se corre el programa. Esta función retorna un entero que representa el estado de retorno del programa. Suele devolver un 0 si todo salio correctamente, cualquier otro valor es señal de error. Si no hay un main en nuestro archivo al ejecutar salta un error.

Cada sentencia en C al final de línea lleva un “;”. También, los bloques estan determinados por “{ }”

La función printf se usa para imprimir en pantalla.

Para compilar, si nos encontramos en el directorio donde se encuentran nuestro archivo, solo basta ejecutar en consola en siguiente comando:

gcc <identificador.c> -Werror -o <identificador.exe>

“-Werror” nos permite no compilar si hay warnings, (un warning es un aviso de algo que no está bien) “-o” seguida del nombre del archivo de salida especifica el nombre del archivo ejecutable que se generará. En Windows se genera un ejecutable “.exe” y en Linux “.out”.

Tipos Basicos:

Los tipos básicos de C son:

* Char -> Carácter, char c = ‘h’;, se usan comillas simples.
* Short
* Int -> Entero
* Long
* Float -> Flotante.
* Double

Toda variable en C tiene un valor, pero si no se le asigna dicho valor, entonces se lo considera basura. Se le suele llamar basura al valor que se encuentra en el lugar de memoria que se le dio a la variable, pero que no le fue asignado nada.

Se pueden declarar constantes de la forma const int a = 1;. Este es un valor que no puede ser cambiado.

Operadores:

Como en Python se puede realizar operaciones al momento de asignar. Como, por ejemplo:

a += 5; que equivale a a = a + 5;

Tenemos los siguientes:

* = Asignación Básica.
* \*= Asigna Producto.
* /= Asigna División.
* += Asigna Suma.
* -= Asigna Resta.
* %= Asigna Modulo.

Se permiten también, este tipo de sentencias:

int a,b,c; a = b = c = 5;

Los operadores aritméticos son:

* + Suma.
* - Resta.
* - Negación.
* \* Multiplicación.
* / División.

Existen los operadores ++ y – que son equivalentes (hasta cierto punto) a sumarle 1 o restarle 1 a la variable.

Salida Estándar:

La entrada y salida en C es formateada, es decir, hay que explicitar como hay que mostrar o leer un dato.

Los formatos más comunes son:

* %d Entero.
* %f Flotante.
* %lf Double.
* %c Char.
* %s String.

Se necesitan tener la misma cantidad de formatos que de datos.

Operadores de comparación:

* < Menor.
* > Mayor.
* <= Menor igual.
* >= Mayor igual.
* == Igual.
* != Diferente.
* ! Negación.
* && AND.
* || OR.

Los datos que nos devuelven los operadores de comparación son 0 (falso) y 1(true).

Entrada estándar:

Para poder tomar el valor de una entrada estándar se utiliza la función scanf. Se usan formatos al igual que en la función printf.

La particularidad del scanf es que delante de la variable, se usa el símbolo &.

Sentencia IF:

En C la sentencia IF es de la siguiente forma:

if (condición1){

bloque si se cumple condición 1.

} else if (condición2){

bloque si s e cumple la condición 2.

} else {

bloque en caso contrario.

}

Bucle WHILE:

La sintaxis del bucle es:

while (condición){

sentencias.

}

Mientras la condición sea verdadera, se ejecutan las sentencias.

Funciones:

Sintaxis de una función:

tipo\_retorno <identificador función> (tipo\_1 <parámetro\_1,…,tipo\_n <parámetro\_n>){

sentencias.

}

Tipo\_retorno indica que tipo de valor devuelve la función (int, char, etc.) en esos casos habrá un return, si no devuelve nada, irá void.

Switch:

C cuenta con la estructura switch la cual compara el valor de una variable y, dependiendo su valor, ejecuta una sentencia especifica para el valor dado.

Su sintaxis es la siguiente:

switch(variable) {

case const1:

sentencias1

break;

…

case const\_n:

sentencias\_n

break;

default:

sentenciasDefault

break;

}

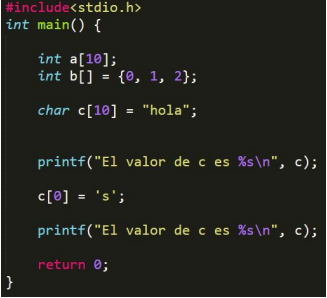
default se ejecuta si ninguna constante coincide con la variable.

Arrays:

Los arrays es un concepto parecido al de las listas en Python, pero con muchas limitaciones.

En C a los arrays se les debe indicar el tamaño, el cual, no puede modificarse.

También, son del mismo tipo de dato.



Ejemplo:



La primera y segunda declaran arrays de ints mientras que la tercera es un

array de chars.

Tanto a como c son arrays que están siendo inicializados con un tamaño de

10.

La notación para definir b indica, implícitamente, el tamaño que va a tener el

array. b se inicializa como un array de ints de tamaño 3 y con valores

iniciales 0, 1 y 2 en sus respectivas posiciones.

Strings:

Existen funciones especificas para usar con las strings, para eso se incluye la librería <string.h>

Las cadenas de caracteres estan definidas de la siguiente forma:

char c[8] = “hola”;



Donde el “\0” marca el fin de la cadena. De la posición 5 en adelante no sabemos que valores toma, entonces, es basura.

Cuando tenemos en la entrada del scanf, un array de char no suele ir el &.

Sentencia FOR:

El bucle FOR tiene, al igual que el bucle while, una condición lógica que, mientras sea verdadera, se ejecuta las sentencias.

Su sintaxis es la siguiente:

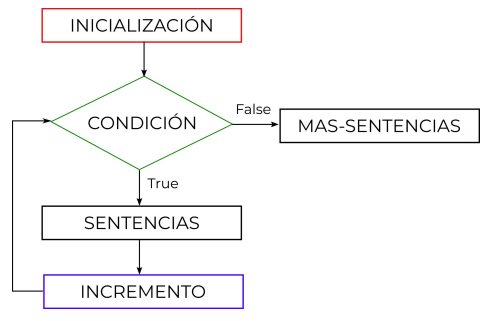
for(inicialización ; condición ; incremento){

sentencias a ejecutar si la condición es true-

}

El bloque inicialización corresponde a las sentencias que se ejecutan previo a verificar la condición del for por primera vez.

El bloque incremento se ejecuta cada iteración, luego de ejecutarse las sentencias y, previo a verificar la condición.



Punteros:

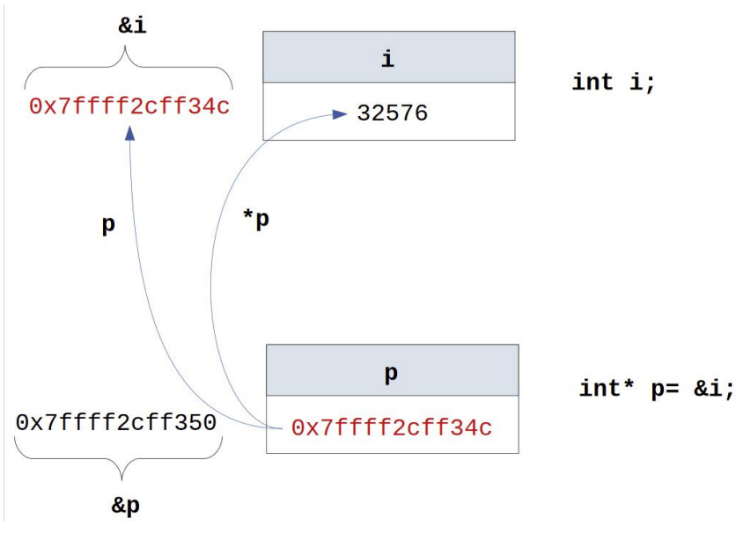
El & representa el lugar donde esta la variable en memoria. Para imprimir una dirección de memoria en el printf se tiene que usar el formato %p. Entonces, si queremos guardar una dirección de memoria en una variable, aparecen los punteros.

Si se quiere guardar una dirección de memoria en una variable, se indica de la siguiente manera.

int\* p

Se antepone el tipo de dato y luego se usa el “\*”.

También, se puede acceder al valor que esta en la direccion de memoria que apunta el puntero. \*p, de esta manera desreferenciamos.



Siento p un puntero a un tipo de dato T tenemos:

* p es tipo T\*.
* \*p es tipo T.
* &p es tipo T\*\*. Ya que estamos obteniendo la direccion donde esta el puntero en la memoria.

El motivo por el cual no se pone