Capitulo 1

1.1 Hola Mundo y manejo de print

*#include* <stdio.h>

/\*

    Curso C

\n sirve para inmdicar una nueva linea

printf es una función de biblioteca que escribe la salida, en

este caso la cadena de caracteres que se encuentra entre comillas.

Nótese que \n representa un solo carácter. U na secuencia de escape como \n

proporciona un mecanismo general y extensible para representar caracteres invisibles o difíciles de escribir. Entre otros que C proporciona están \t para tabulación, \b para retroceso, \" para comillas, y \ \ para la diagonal invertida. Hay una

lista completa en la sección 2.3.

\*/

main(){

*printf*( "hola, mundo\n");

}

1.2 Conversion Fahrenheit a Celsius, manejo de variables punto (flotante e int) e iteraciones

/\*

    Curso C

    conversion de fahrenheint a celcius

    cap 1, var, expre e iteraciones

\*/

*#include* <stdio.h>

// print la tabla de F-C para fahr=0,20 .., 300

main(){

    // Definicion

*int* fahr,celsius;

*int* lower,upper,step;

    // inicializador

    lower=0;

    upper=300;

    step=20;

     // %d -> SIRVE PARA REPRESENTAR VARIABLES

    fahr = lower ;

*while*(fahr<= upper)

    {

        /\*  %d -> SIRVE PARA REPRESENTAR VARIABLES

            \t%d -> SIRVE PARA TABULAR CADA RESULTADO

            Este ejemplo también muestra un poco más acerca de cómo funciona printf.

            En realidad, printf es una función de propósito general para dar form ato de salida, que se describirá con detalle en

            el capítulo 7. Su primer argumento es una cadena de caracteres que serán impresos, con cada % indicando en donde uno de

            los otros (segundo, tercero, ...) argum entos va a ser sustituido, y en qué forma

            será impreso. Por ejemplo, % d especifica un argum ento entero, de modo que la

            proposición

            printf(''%d\t%d\n", fahr, celsius);

            hace que los valores de los dos enteros fah

        \*/

        celsius=5\*(fahr-32)/9;

*printf*("%d\t%d\n",fahr,celsius);

        fahr=fahr+step;

    }

}

Esto es muy semejante a lo anterior, excepto que fahr y celsius están declarados como float, y la fórmula de conversión está escrita en una forma más natural.

No pudimos utilizar 5/9 en la versión anterior debido a que la división entera lo

truncaría a cero. Sin embargo, un punto decimal en una constante indica que ésta

es de punto flotante, por lo que 5.0/9.0 no se trunca debido a que es una relación

de dos valores de punto flotante.

Si un operador aritmético tiene operandos enteros, se ejecuta una operación

entera. Si un operador numérico tiene un operando de punto flotante y otro entero, este último será convertido

a punto flotante antes de hacer la operación. Si

se hubiera escrito fahr - 32, el 32 sería convertido automáticam ente a punto

flotante. Escribir constantes de punto flotante con puntos decimales explícitos,

aun cuando tengan valores enteros, destaca su naturaleza de punto flotante para

los lectores humanos.

/\*

    Curso C

    conversion de fahrenheint a celcius

    cap 1, var, expre e iteraciones

    gcc tu\_archivo.c -o nombre\_ejecutable

    ./nombre\_ejecutable

\*/

*#include* <stdio.h>

// print la tabla de F-C para fahr=0,20 .., 300

main(){

    // Definicion

*float* fahr,celsius;

*int* lower,upper,step;

    // inicializador

    lower=0;

    upper=300;

    step=20;

     // %d -> SIRVE PARA REPRESENTAR VARIABLES

    fahr = lower ;

*while*(fahr<= upper)

    {

        \*/

        celsius=5.0\*(fahr-32.0)/9.0;

*printf*("%3.0f\t%6.1f\n",fahr,celsius);

        fahr=fahr+step;

    }

}

%d escribe como entero decimal

%6 d escribe como entero decimal, por lo menos con 6 caracteres de amplitud

%f escribe como punto flotante

%6f escribe como punto flotante, por lo menos con 6 caracteres de amplitud

%.2 f escribe como punto flotante, con 2 caracteres después del punto decimal

%6 .2 f escribe como punto flotante, por lo menos con 6 caracteres de ancho y 2 después del punto decimal

Entre otros, printf también reconoce %o para octal, % x para hexadecimal, % c para carácter, °/os para cadena de caracteres y % % para % en sí.

Ejercicio 1-3. M odifique el program a de conversión de tem peraturas de modo que escriba un encabezado sobre la tabla. □ Ejercicio 1-4. Escriba un program a que imprima la tabla correspondiente Celsius a Fahrenheit.

/\*

    Curso C

    conversion de fahrenheint a celcius

    cap 1, var, expre e iteraciones

\*/

*#include* <stdio.h>

// print la tabla de F-C para fahr=0,20 .., 300

main(){

    // Definicion

*float* fahr,celsius;

*int* lower,upper,step;

    // inicializador

    lower=0;

    upper=300;

    step=20;

     // %d -> SIRVE PARA REPRESENTAR VARIABLES

    fahr = lower ;

*printf*("celsius   fahr\n");

*while*(celsius <= upper)

    {

        /\*  %d -> SIRVE PARA REPRESENTAR VARIABLES

            \t%d -> SIRVE PARA TABULAR CADA RESULTADO

        \*/

        fahr=((9.0/5.0)\*celsius)+32.0;

*printf*("%3.0f\t%6.2f\n",celsius,fahr);

        celsius=celsius+step;

    }

}

1.3 La proposición for

*#include* <stdio.h>

/\*Imprime la tabla Fahrenheit-Celcius\*/

main()

{

*int* fahr;

*printf*("Fahr Cel\n");

*for* (fahr = 0;fahr <= 300; fahr= fahr + 20)

*printf*("%3d%6.1f\n", fahr,(5.0/9.0)\*(fahr-32));

}

La proposición for es un ciclo, una forma generalizada del while. Si se compara con el while anterior, su operación debe ser clara. Dentro de los paréntesis existen tres secciones, separadas por punto y coma. La primera, la inicialización se ejecuta una vez, antes de entrar propiam ente al ciclo. La segunda sección es la condición o prueba que controla el ciclo: fahr < = 300 Esta condición se evalúa; si es verdadera, el cuerpo del ciclo (en este caso un simple printf) se ejecuta. Después el incremento de avance fahr = fahr + 2 0 se ejecuta y la condición se vuelve a evaluar. El ciclo term ina si la condición se hace falsa. Tal como con el while, el cuerpo del ciclo puede ser una proposición sencilla o un grupo de proposiciones encerradas entre llaves. La inicialización, la condición y el incremento pueden ser cualquier expresión. La selección entre while y for es arbitraria, y se basa en aquello que parezca más claro. El for es por lo general apropiado para ciclos en los que la inicialización y el incremento son proposiciones sencillas y lógicamente relacionadas, puesto que es más compacto que el while y mantiene reunidas en un lugar a las proposiciones que controlan al ciclo.

1.4 Constantes simbólicasComo observación, es importante no dejar en el código números mágico como 300 y 20 en un programa, ya que estos proporcionan poca información a quien tenga que leer el programa y son difíciles de modificar , para eliminar esta mala practica se aconseja el uso de #define

#define nombre texto de reemplazo

A partir de esto, cualquier ocurrencia de nombre (que no esté entre comillas ni como parte de otro nombre) se sustituirá por el texto de reemplazo correspondiente. El nombre tiene la misma form a que un nombre de variable: una secuencia de letras y dígitos que comienza con una letra. El texto de reemplazo puede ser cualquier secuencia de caracteres; no está limitado a números.

Las cantidades LOWER, UPPER y STEP son constantes simbólicas, no variables, por lo que no aparecen entre las declaraciones. Los nombres de constantes simbólicas, por convención, se escriben con letras mayúsculas, de modo que se puedan distinguir fácilmente de los nombres de variables escritos con minúsculas. Nótese que no hay punto y coma al final de una línea # d efin e.