Curso C

Repaso – cosas básicas

Concepto programa

- ¿Qué es un programa?
  - Secuencia ordenada de instrucciones con el fin de realizar una tarea
- Lenguaje ensamblador vs lenguaje alto nivel
  - Ensamblador : Más cercano a lenguaje maquina; nemotecnico; requiere conocimiento de la arquitectura;
  - Alto nivel: cercano a lenguaje humano; no requiere conocimiento arquitectura

Compilador e Interprete

- Compilador
  - Analiza el codigo esctrito y genera instrucciones en lenguaje maquina (programa objeto).
  - Posteriormente Linkado enlaza librerias y otros programas objeto y genera .exe
- Interprete
  - Analiza y ejecuta un programa sentencia a sentencia; no genera ejecutable
  - Menos eficiente, pero muy versátil
    - Java
    - C# (pseudo interpretado)

Bloques de un programa

- 1- Directivas pre-procesador
  - Includes, defines, pragmas,...
- 2- Bloque del programador
  - Comentario inicial indicando:
    - Nombre programa
    - autor
    - Descripcion
    - Variables globales y explicacion
    - Definicion de funciones
- 3- Funcion Principal (main)
- 4- Resto funciones

Tipos de datos: variables

- variable
  - TipoDato identificador[,identificador,...];
- Constantes
  - Valores numéricos
    - 23484 de tipo int
    - 253u indica unsigned
    - 746L indica long
    - 583UL indica usigned long
  - Octales y hexadecimales
    - 011 valor octal, equivale a 9 en base 10
    - 0xA hexadecimal
  - Científica: .874e-2 double= .00874
  - caracteres
  - const int mivar=5 enum dia{lunes=1,martes}

Tipos de Datos

Modificadores de Tipo

TIPO	Bits	RANGO
char	8	ASCII
unsigned char	8	0 .. 255
signed char	8	-128 .. 127
int	16	-32768 .. 32767
unsigned int	16	0 .. 65535
signed int	16	-32768 .. 32767
short int	8	-128 .. 127
unsigned short int	8	0 .. 255
signed short int	8	-128 .. 127
long int	32	-2147483648 .. 2147483649
signed long int	32	-2147483648 .. 2147483649
unsigned long int	32	0 .. 4294967296
float	32	6 dígitos precisión aprox.
double	64	12 dígitos precisión aprox.
long double	128	24 dígitos precisión aprox.

Depende de compilador, S.O, arquitectura. Int, long double

Tipos de datos: almacenamiento

- Modo de almacenamiento de variables:
  - auto-> defecto para variables declaradas en bloques {} -> son locales
  - extern-> variables globales. Son visibles en un fichero. Para hacerlas visibles en otros, declararlas en otros como extern. Inicializacion por defecto a 0 (?¿¿?)
  - static-> conservan su valor entre distintas ejecuciones dentro de un bloque
  - register-> recomendación a compilador almacenar dicha variable en registro CPU.

Tipos de datos: conversion

- Implicitas
  - Por ejemplo sumar variables distinto tipo (int + float), int se convierte a float para poder sumarse.
  - Asignar a variable resultado operaciones
    - double x= i*j+1 donde i,j son int
- Explicitas (casting)
  - K= (int) 1.7 + (int) masa;
  - Suele usarse en los valores de retorno de las funciones.
  - Importante en operaciones como division.

Operadores

- Aritmeticos
  - +, - , *, /, %
- Asignacion
  - =, +=, -=, *=, /=
    - Ejemplo: distancia+=1 -> distancia=distancia +1;
- Incrementales
  - ++, --
- Relacionales
  - ==, <=,>=,<,>,!=
- Lógicos
  - &&, || , !
    - Para evaluar expresiones

Operadores

- Operadores bits
  - & (and),| (or),^ (xor), ~ (c1), >>(desplz. Dcha), << (desplz. Izda)

```
#include <stdio.h>

int main()
        {
        printf( "El resultado de la operación 170 & 155 es: %i\n", 170 & 155 );
        }
```

```
170 -> 10101010
155 -> 10011011 &
138 -> 10001010
```

Eficiencia multiplicar, dividir

Operadores

- Otros
  - sizeof
  - [], ()
  - .
  - ->
  - & (variables)
  - * (variables)

Operadores: precedencia

Precedencia
() [] -> .
! ~ ++ -- (molde) * & sizeof (El * es el de puntero)
* / % (El * de aquí es el de multiplicación)
+ -
<< >>
< <= > >=
== !=
&
^
|
&&
||
?:
= += -= *= /=
,

Control Flujo Ejecucion

- Operador condicional
  - Expresion1? Expresion2: expresion3
- if (expresion){} else {}
- switch

```
switch ( variable )
        {
        case opción 1:
                código a ejecutar si la variable tiene el
                valor de la opción 1
                break;
        case opción 2:
                código a ejecutar si la variable tiene el
                valor de la opción 2
                break;
        default:
                código que se ejecuta si la variable tiene
                un valor distinto a los anteriores
                break;
        }
```

- goto (salta a una etiqueta)

Sólo usar en programas bajo nivel

Control Flujo Ejecucion

- Bucles
  - For

```
for( dar valores iniciales ; condiciones ; incrementos )
        {
        conjunto de intrucciones a ejecutar en el bucle
        }
```

    - Flexible: podemos obviar algunas secciones, varias inicializaciones, incrementos, condiciones (separadas por comas)
  - While

```
while ( condición )
        {
        bloque de instrucciones a ejecutar
        }
```

    - Mientras condicion cierta -> ejecuta

Control Flujo Ejecucion

- Bucles
  - do..while

```
do
        {
        instrucciones a ejecutar
        } while ( condición );
```

    - Se ejecuta al menos una vez

- break -> interrumpe ejecucion bucle y salta a siguiente instrucción fuera.
- continue-> fuerza siguiente iteracion bucle

Entrada/salida

- Printf -> imprime en la unidad de salida el texto, constantes y variables que se indiquen.
  - int printf(“cadena control”, argumento1, argumento2,...);
    - Ejemplo:
      - int i=5; float masa=3.2;
        printf(“resultado nº: %d, masa= %f \n”,i,masa);

Entrada/salida

- Printf (modificadores)

%c imprime un carácter con el código ASCII dado (con 66 imprime B, por ejemplo).

%s una cadena

%d un decimal

%u un decimal negativo o positivo sin signo

%o un integral en octal

%x un integral en hexadecimal

%e un número con punto flotante en notación científica

%f un número con punto flotante en notación decimal

%g puede ser %e ó %f, lo que decida printf que es lo mejor

%X lo mismo que %x pero con mayúsculas

%E como %e pero con 'E' mayúscula

%G lo mismo que %E cuando se usa notación científica

%p un "pointer"; imprime la locación en memoria de una variable en hexadecimal

%n imprime la cantidad de caracteres desplegados hasta el momento

Entrada/salida

- Printf (modificadores)

%c imprime un carácter con el código ASCII dado (con 66 imprime B, por ejemplo).

%s una cadena

%d un decimal

%u un decimal negativo o positivo sin signo

%o un integral en octal

%x un integral en hexadecimal

%e un número con punto flotante en notación científica

%f un número con punto flotante en notación decimal

%g puede ser %e ó %f, lo que decida printf que es lo mejor

%X lo mismo que %x pero con mayúsculas

%E como %e pero con 'E' mayúscula

%G lo mismo que %E cuando se usa notación científica

%p un "pointer"; imprime la locación en memoria de una variable en hexadecimal

%n imprime la cantidad de caracteres desplegados hasta el momento

Entrada/salida

- Printf (formato)

'0' (cero): Para quitar el padding de un número. El número es convertido a un número con la cantidad 0 necesarios a la derecha.

```
printf("%03d", 7); # imprime '007'
printf("%03d", 153); # imprime '153'
```

'-' (menos): Especifica que el valor será ajustado a la derecha, mientras que normalmente es hacia la izquierda.

```
printf("%5s", 'foo'); # imprime ' foo'
printf("%-5s", 'foo'); # imprime 'foo     '
```

' ' (un espacio): Especifica que un espacio debe de dejarse antes de un número positivo.

```
printf("% d", 7); # imprime ' 7'
printf("% d", -7); # imprime '-7'
```

'+' (signo de más): Especifica que este valor siempre debe de ponerse antes del valor.

```
printf("%+d", 7); # imprime '+7'
```

Entrada/salida

- Printf (formato)

. (un punto) El punto de precisión seguido de un número especifica la cantidad de números que se deben de imprimir para 'd', 'i', 'o', 'u', 'x', y expresiones 'X'. Para expresiones 'e', 'E', y 'F', es el número de dígitos que se deben de mostrar después del número decimal. Para 'g' y 'G' el número máximo de caracteres significativos. Para la expresión de cadena 's', es el máximo número de caracteres que se deben de desplegar.

```
printf("%.3d", 7); # imprime '007'
printf("%.2f", 3.66666); # imprime '3.66'
printf("%.3s", 'foobar'); # imprime 'foo'
```

Entrada/salida

- Gotoxy (int x, int y)
  - Posiciona el cursor en la pantalla. Posicion 1,1 es la esquina superior izquierda.
- Clrscr
  - Limpia/borra la pantalla
  - Ojo, estas dos sentencias solo para consola DOS

Entrada/salida

- Scanf
  - Funciona de forma similar que printf, pero para entrada de datos desde teclado

int scanf(“%x1, %x2”, &arg1, &arg2);

Donde %x1y%x2 son modificadores del tipo %d,%f,%s, etc

Entrada/salida

- putchar/getchar
  - Permiten escribir o leer un solo carácter
  - Ej. putchar(‘a’);
    - c= getchar()