**Daniel Celis**

**Practica 2.**

**21.** En el stack en diferentes zonas se escriben la dirección a la que debe retornar cada que hace saltos en memoria, y en otra zona se guardan las variables con las que se hace dicho control.

**22.** if (age > 18 && cupo>0) // se hará una sentencia si la edad ingresada es mayor de 18 años y hay por lo menos un cupo.

if (!age < 18) // se hará una sentencia si la edad ingresada es mayor de 18 años.

if (permiso = 1 || age > 18) // se hará una sentencia si se tiene permiso (binario) o si la edad ingresada es mayor de 18 años.

**23. a)** La función atoi convierte un string a una variable tipo int, esto lo hace descartando los espacios en blanco e interpretando los valores de la cadena convirtiéndolos a un número tipo int. Si la secuencia de caracteres interpretada no es un número valido o el string esta vacio se retorna un cero.

**b)** El operador % es el modulo de la división, se usa para guardar una variable con valores entre un rango. Recordar que:

x = 7 % 5; // x now contains 2

**c)** MyData está en la memoria..

**d)** char str[]es un string sin tamaño definido, el compilador cuenta el número de elementos y crea un arreglo de tamaño apropiado para contenerlo en memoria.

**24.** Una función puede llamarse con diferentes tipos de parámetros usando la función random.

- random(max): genera un valor aleatorio en un rango entre 0 y max.

- random(min, max): genera un valor aleatorio en un rango entre min y max.

**25.** Randomseed() es una función que pide como parámetro un valor que será el valor semilla en la función random() debido a que esta es una función pseudo-aleatoria que siempre parte de una semilla, cambiarla hará mas aleatoria la función random(). En el código ejemplo se entra como semilla un valor análogo lo que da 256 posibilidades de semilla.

**26.** serial.print(val): esta función imprime el valor que se ingrese como parámetro ya sea números caracteres, o cadenas de caracteres.

serial.print(val, format): esta función imprime el valor que se ingrese como parámetro en el formato que se defina como segundo parámetro, ya sea decimal, octal, hexadecimal, binario. Para números de punto flotante como formato se puede definir cuantos decimales desea mostrar.

**27.** Print: Imprime los datos que pasan por el puerto de comunicaciones en caracteres ASCII legibles, sea cual sea el formato que se seleccione.

Write: Imprime los datos binarios que pasan por el puerto de comunicaciones sin traducirlos a ningún código, como series de bits.

**28.**  Escribe en el puerto el número 1250 pero como el numero tiene 2bytes, se trunca por lo cual solo aparece en pantalla E2 ya que son los bit de menor peso.

**29.** La transmisión asincrónica es aquella en la que los bits de datos de un carácter se transfieren de manera independiente en el tiempo con respecto a otro carácter y no hay sincronización con el receptor por lo cual este no sabe cuándo debe leer el carácter enviado, por tanto debe usarse un bit de inicio y un bit de paro para que el receptor identifique que se ha recibido un carácter.

**30.** Serial.avaible() lo que hace es obtener la información que esta almacenada en el puerto serial disponible para lectura. Esta información se almacena en un buffer de recepción con un tamaño de 64 bytes. Al poner esta función en un if => if(Serial.avaible()>0) lo que hacemos es poder hacer una sentencia en el caso de que haya información en dicho buffer.

**31**. myData contiene la información ingresada por consola, esta información es un año el cual se verifica si es bisiesto.

**32**. La función ReadLine() funciona entrando a un while que se mantiene desde que haya información en el buffer, desde que esto ocurra se llama la función Serial.readBytesUntil() y su resultado es una posición de la variable STR y dicho resultado es lo que devuelve la función.

**33.** Cuando aumentamos el intervalo de espera, el año ingresado no corresponde con el que realmente aparece.

**34**.