UN PROGRAMA CON VARIAS FUNCIONES

Un programa un poco mas complicado

OBJETIVOS

.

- Aprender a operar con Arrays .
- Como partir un problema en funciones especialistas.
- Operar con enteros y Strings.
- Entrenar y desarrollar algunas habilidades claves para programar.

MATERIAL REQUERIDO.

	**Arduino Uno o similar**. Un PC con el entorno de Arduino correctamente instalado y configurado.

PLANTEANDO UN PROGRAMA UN POCO MÁS COMPLICADO.

Hemos visto ya como definir funciones. En esta sesión vamos a plantear un programa que acepte un número desde la consola y compruebe si es o no primo. Y en caso de no serlo, nos calcule cuales son los divisores primos de este.

Normalmente para resolver un problema complejo es buena política partirlo en otros problemas más pequeños que podamos resolver más fácilmente. En este caso vamos a plantear al menos 3 funciones:

- Primo() – Calcula si un número dado es primo, devuelve true y en caso contrario false.
- Divisores() – Para un número dado nos devuelve sus divisores primos.
- GetLine() – Vamos a definir una función genérica que recoja una cadena de texto de la puerta serie, para procesarla a posteriori. En este caso recogerá el número a probar.

La idea es, que nuestro programa empiece comprobando si un numero es primo. Si lo es, bien por el. SI no llamaremos a una función que calcule cuales son sus divisores. Y por ultimo necesitamos de algo que nos pueda dar un número desde la consola para probarlo y por eso escribimos otro programa que nos permita recibir cómodamente este numero de entrada.

Fijaros que casi sin haber escrito una linea de programa, ya he decidido como partirlo en bloques mas sencillos de manejar y programar. En otras palabras, he buscado una estrategia, de resolución

- *Esto es un poco trampa. Parece que se me acaba de ocurrir instantáneamente pero no (Aunque parezca increíble).*
- *Después de pensar un rato y pasar un rato mayor escribiendo y afinando el programa, da gusto presentarlo como si fuera fácil. Que lo es, pero lo que no ves, es la de horas que hay que estar haciendo pruebas hasta todo va encajando y queda afinado.*
- *Con tiempo y práctica (No, no mucha) iréis mejorando con rapidez, en vuestra habilidad de romper problemas en pedazos manejables, simplemente requiere tiempo y entrenamiento, y esto es algo que os será útil no solo para programar.*

OPERANDO CON ARRAYS.

Con la función Primo() que vimos en la sesión anterior, a medida que el tamaño del número a probar, crece, el tiempo que tarda en determinar si es primo también, ya que dividimos por todos los números que le preceden.

Una manera más eficaz de calcular si un número es primo, es dividirlo solo por los números primos menores que el. Pero para esto necesitaríamos un modo de archivar estos primos.

Podríamos ejecutar primero el programa Prog_8.3 para hallar los N primeros números primos, y si dispusiéramos de algún medio para guardarlos, tendríamos un sistema más eficaz para decidir si un número es o no primo.

Una manera de archivar estos números es definir un array.

Un array es simplemente una colección de elementos organizados como una matriz, y pueden definirse con varias dimensiones. Empecemos con un array de una sola dimensión. Para definirlo podemos optar por dos maneras:

```
int serie1 [ 5] ; //Creamos una colección de 5 enteros
int serie2[] = { 3,5,6,12, 23} ;
```

En el primer caso definimos un array de enteros, de una sola dimensión con 5 elementos, sin asignar valores de momento.

En el segundo caso asignamos un array de enteros a los valores que le pasamos entre llaves, sin especificar cuantos, porque le dejamos a C++ la tarea de contarlos. Decimos que definimos el array por enumeración.

Para asignar o leer los valores de un array se utiliza un índice entre corchetes. Veamos este programa **Descargar** :

```
int serie2[] = { 3,5,6,12, 23} ;            // Prog_9_1
void setup()
    {
        Serial.begin(9600) ;
    }
void loop()
    {
       for (int i=0 ; i<5 ; i++)
            Serial.println("Posicion " + String(i)+ ": "+ String(serie2[i])) ;
    }
```

El programa imprime el contenido del array recorriendo sus 5 posiciones.

- ***Atención**: la primera posición del un array es la 0 y la última el número de elementos – 1. Así serie2 [0] devuelve el primer elemento 3, y serie2[4] el último 23.*

Un error muy peligroso, y difícil de detectar sería algo así (Prog_9.2):

```
int serie2[] = { 3,5,6,12, 23} ;
    for (int i=0 ; i<99 ; i++)
         Serial.println("Posicion " + String(i)+ ": "+ String(serie2[i])) ;
```

Uno esperaría que C++ generase un error, ya que definimos un array de 5 elementos y hacemos referencia a 100, pero no. Nuevamente C++ nos

sorprende devolviendo correctamente los 5 primeros valores y luego sigue leyendo posiciones de memoria consecutivas tan tranquilo, como si tuvieran sentido.

- *C++ espera que seamos nosotros quienes controlemos esto, así que mucho cuidado*

Por último, mencionar que podemos manejar arrays de varias dimensiones:

```
Int Tablero[ 8, 8 ] ;
```

Imaginad que Tablero representa las posiciones de una partida de ajedrez y cada valor que contiene esa posición corresponde a una pieza que se encuentra en esa casilla.

AFINANDO LA FUNCIÓN PRIMO()

.

Si corremos el programa Prog_8.3 nos dará en el monitor una lista de primos hasta el 1024 (*o hasta el número que deseemos modificando el valor de máximo*) y seleccionándolos con el ratón podremos copiar esos valores y pegarlos en el IDE para crear un array con ellos (Prog_9.3):

```
int P[] =
{          2,          3,          5,          7,         11,         13,         17,         19,
          23,         29,         31,         37,         41,         43,         47,         53,
          59,         61,         67,         71,         73,         79,         83,         89,
          97,        101,        103,        107,        109,        113,        127,        131,
         137,        139,        149,        151,        157,        163,        167,        173,
         179,        181,        191,        193,        197,        199,        211,        223,
         227,        229,        233,        239,        241,        251,        257,        263,
         269,        271,        277,        281,        283,        293,        307,        311,
         313,        317,        331,        337,        347,        349,        353,        359,
         367,        373,        379,        383,        389,        397,        401,        409,
         419,        421,        431,        433,        439,        443,        449,        457,
         461,        463,        467,        479,        487,        491,        499,        503,
         509,        521,        523,        541,        547,        557,        563,        569,
         571,        577,        587,        593,        599,        601,        607,        613,
         617,        619,        631,        641,        643,        647,        653,        659,
         661,        673,        677,        683,        691,        701,        709,        719,
         727,        733,        739,        743,        751,        757,        761,        769,
         773,        787,        797,        809,        811,        821,        823,        827,
         829,        839,        853,        857,        859,        863,        877,        881,
         883,        887,        907,        911,        919,        929,        937,        941,
         947,        953,        967,        971,        977,        983,        991,        997,
        1009,       1013,       1019,       1021
 }  ;
```

Hemos definido un array enumerando sus elementos, entre llaves y separados por comas.

- *Es importante percatarse de que después de copiar y pegar las salida de Prog_8.3 hemos borrado la coma después del 1021, porque si no daría un error de sintaxis al definir el array.*
- *Obsérvese que hay un punto y coma después de la llave de cierre del array. Aunque está entre llaves es una instrucción de asignación y no una definición de función.*
- *Al definir el array por enumeración, si el número es alto podemos perder de vista cuantos elementos contiene. Si necesitamos calcular el número de miembros podemos utilizar la función sizeof():*

```
int size = sizeof(P) / sizeof(int);
```

- Donde P es nuestro array y dividimos por sizeof(int) porque definimos P como int. Y para este caso devuelve un valor de 172 elementos

Ahora bastaría dividir el número a probar por aquellos elementos del array P, menores que él:

```
bool Primo(int x)
```

```
{
    int index = 0 ;
    while ( P[index]  <  x)
        { if ( x  %  P[index++] == 0)
              return(false);
        }
    return(true);
}
```

Recorremos los valores almacenados en el array P[] mediante index al que vamos incrementando en cada iteración, hasta que nos toque probar un primo mayor que el valor que comprobamos.

FUNCIÓN DIVISORES ()

.

Esta función va a recorrer los elementos del array en tanto en cuanto sean menores (posibles divisores) que el número que probamos. Si encontramos divisores primos los guardamos en un array que hemos llamado Div[] al que le asignamos un máximo de 32 divisores:

```
int Div[32] ;
int Divisores(int x)
    {
        int index = 0 ;                     //Apunta a la posicion del array P[]
        int pos = 0 ;                       //Para apuntar al array de divisores Div[]
        while ( P[index] < x)
             {
                 int k = P[index++] ;
                 if ( x % k == 0)
                 Div[pos++]= k ;            //Guardamos el divisor en en el array Div[].
              }                             // para uso posterior
        return(pos);                        //Devolvemos el numero de divisores encontrado
    }
```

Cuando queramos imprimir los divisores, basta con recorrer Div[].

- *Es importante entender que tanto la función Primo() como Divisores() recorren el array de números primos hasta que sobrepasan el valor del numero a probar. Si el número que probamos es mayor que el máximo primo que contiene P[], Podemos obtener resultados extraños, ya que leeremos más elementos de los que hemos definido.*
- *Este método de buscar divisores es válido solo para números inferiores a 1024(o en su caso, al máximo número hasta el que hayamos calculado primos), porque un número no será primo si Primo() lo afirma, ya que encontrará divisores. Pero puede afirmar que un numero es primo erróneamente si sus divisores son superiores al máximo primo en P[].*

LA FUNCIÓN GETLINE()

.

Aunque ya comentamos que podemos usar una función parseInt () incluida en Arduino para recoger un valor del puerto serie, tiene el inconveniente de que si no recibe una entrada salta al cabo de un tiempo (muy escasito) y devuelve 0, por lo que tendríamos que controlar el valor devuelto para que no se repitiese continuamente.

Por eso vamos a escribir una función de uso general que nos permita recoger una cadena de texto de la puerta serie sin que salga hasta que reciba un String que vamos a hacer finalice en intro. De hecho ya vimos este programa, aunque no como función en la sesión Comunicación con el exterior

```
String GetLine()
    {
        String S = "" ;
        if (Serial.available())
```

```
        {
             char c = Serial.read(); ;
             while ( c != '\n')                          //Hasta que el character sea intro
                  {
                       S = S + c ;
                       delay(25) ;
                       c = Serial.read();
                  }
             return(S) ;
        }
}
```

Definimos Getline() de tipo String, porque queremos que nos devuelva un texto. Comprobamos que hay algo disponible en la puerta serie, y en caso afirmativo construimos un String S añadiéndole cada uno de los caracteres que leemos del puerto serie, hasta que encontremos un intro.

Al encontrar el intro, se cumple la condición de salida del while y termina la función devolviendo la cadena construida (sin el intro).

- *Normalmente convendrá comprobar si hay algo disponible en la puerta serie **antes** de llamar a GetLine(), y si es así, la comprobación que hace GetLine() de tener algo disponible en el Serial seria redundante.*
- *Pero si llamáramos a GetLine() sin comprobarlo y esta no lo controlase, quedaríamos atrapados en esta función hasta que alguien escribiera algo finalizado con intro para poder salir y podría no ser sencillo comprender el problema.*
- *Nuevamente hemos incluido un delay de 25 ms en el while para asegurarnos de que Arduino no puede volver a leer mas caracteres antes de que a la velocidad de 9600 bps haya llegado el próximo carácter. Si la velocidad de comunicación es de 115200 bits por segundo o más, se puede suprimir este retraso.*

EL PROGRAMA PRINCIPAL

.

Podemos ya escribir nuestra función principal loop(), que llame a las funciones que hemos definido a lo largo de esta sesión, para determinar si un numero que le pasamos por la puerta serie es primo o no y en caso negativo que nos muestre los divisores primos encontrados.

Podría ser algo así: **Calculo de numeros primos en arduino** :

```
void loop()
    {
        if (Serial.available())
            {
                String s = GetLine();
                int i = s.toInt() ;          //Como esperamos un numero, convertimos el texto a numero
                if ( Primo(i))
                     Serial.println(String(i) + " Es primo.");
                else
                     {
                        Serial.println(String(i) + " No es primo.");
                        Serial.println("Sus divisores son: ");
                         int j = Divisores(i);                //Recogemos el numero de divisores encontrados
                         for (int n =0 ; n<j ; n++)           //Imprimimos los divisors del Div[]
                              Serial.print(String(Div[n]) + ",\t");
                         Serial.println(" ");                 // Al acabar salta de linea
                     }
            }
    }
```

Empezamos comprobando si hay algo sin leer en la puerta serie y si es asi llamamos a GetLine() para que nos consiga lo que hay.

Como GetLine() nos devuelve un tipo String() usamos la función estándar de Arduino C++, s.toInt() que convierte el contenido String a tipo numérico int.

Después llamamos a Primo() para que compruebe este número. Si es primo, simplemente imprime un mensaje para confirmarlo. En caso contrario llamamos a Divisores() que busca y almacena en el array Div[] los divisores primos que encuentra.

Cuando divisores regresa, nos devuelve el número de divisores encontrados y podemos imprimirlos con un sencillo bucle for.

RESUMEN DE LA SESIÓN

.

- Hemos operado con arrays de una única dimensión tanto para leer su contenido como para modificarlo.
- Hemos utilizado el programa de primos como excusa para mostrar cómo se pueden resolver problemas complejos, dividiéndolos en otros más sencillos que podamos resolver con funciones simples.
- Esta es, sin duda, una de las habilidades clave para ser un programador competente, y como todo en la vida requiere practica e ir trabajando.
- También hemos definido una función de propósito general GetLine() que utilizaremos más veces en el futuro.

ANTERIOR **SIGUIENTE**

(14) COMMENTS

Jmcn *13 Jul 2015*

Reply

Primero quiero felicitarte por el estupendo tutorial que nos ofreces a los novatos, con contenidos progresivos y llenos de buen humor. Asi da gusto aprender. Muchas gracias.
Una duda:
Cuando imprimo el valor del calculo con la funcion sizeof (P)/sizeof (int), me resulta 172, no 72, como tu dices. Donde esta el error?.
Un saludo.

admin *13 Jul 2015*

Reply

Hola jm, tienes toda la razon y es un error mio al escribir el resultado. La idea es que diviendo el tamaño del array entre el tamaño de un int nos da el numero de elementos que son 172. Ya lo he corregido para futuras visitas, muchas gracias

jmcn *13 Jul 2015*

Reply

De cualquier manera, por aclarar mejor la función que realiza ” sizeof (P)”. Por qué hay que dividirlo por “sizeof (int)” para obtener el numero de elementos del array?. Por qué “sizeof (int)” es 2?. Si imprimo el valor de “sizeof (P)”, me entrega el valor doble (344).
Gracias

admin *14 Jul 2015*

Reply

sizeof() devuelve el tamaño en bytes de el argumento que le le pasas. Asi, sizeof(P) devuelve el tamaño en bytes del array P y sizeof(int) devuelve 2 porque en Arduino C++ un int es de 16 bits o 2 bytes, por eso el numero de elemntos en en el array es 344 / 2 = 172
Conviene destacar, ya que estamos, que sizeof(int) puede devolver otros valores dependiendo del entorno de trabajo. Asi en un PC devolvera 4 porque los enteros son de 4 bytes y en otros entornos los valores pueden cambiar. En cambio sizeof(P) / sizeof(int) siempre devolvera el numero de elementos de ese array , suponiendo que es un array of int, independientemente de la plataforma en que lo ejecutes

Jose Angel *04 Ene 2016*

Reply

Hola estimado!

Primero que nada quería felicitarte por los tutoriales expuestos, que me han realmente ayudado muchísimo. Gracias por todo el esfuerzo y trabajo que le han puesto y la rápida respuesta a los comentarios.
El segundo tema era para plantear algo. Antes que nada, entiendo bastante de programación pero el único lenguaje que he usado ha sido Java. Por eso quizás C++ se manejaba de manera distinta a lo que venía haciendo. He aquí mi duda correspondiente a la función/método -Primo()

```
bool Primo(int x)
{
int index = 0 ;
while ( P[index] < x)
{ if ( x % P[index++] == 0)
return(false);
}
return(true);
}
```

En el " if (x % P[index++] == 0) " . ¿No estaríamos salteando siempre el primer elemento del Array?. Ya que index comienza en 0 y la comparación la estaría haciendo con el elemento posterior al primero.

```
bool Primo(int x)
{
int index = 0 ;
while ( P[index] < x)
{ if ( x % P[index] == 0)
{ return(false);
index++;
}
}
return(true);
}
```

Es lo que le haría a la función. No sé si estoy o no errado. Saludos desde Argentina y nuevamente gracias!

Reply

admin *04 Ene 2016*

Hola Jose Angel, No conozco Java porque nunca le he dedicado tiempo y por eso no se si existen los ++, aunque es facil que si porque segun tengo entendido, Java es un derivado de C++ al que se le eliminan los punteros y se define un modelo de trabajo (Y seguro que mas cosas)

EN cuanto a tu pregunta el operador ++ se puede usar en dos modos: prefix y postfix. SI hacemos index++ en postfix, significa : usa index en esta operacion y al acabar incrementalo. Equivale a esto:
usa index en lo que sea
index = index +1

Si lo usaramos en modo prefix, si que seria el caso que planteas: ++index , ya que entonces seria: priomero incrementa y despues usalo en esta instruccion, ¿vale?

Las dos programas son exactamente iguales

Un saludo

Reply

Edwin *14 Ene 2016*

Admin, muy buen tutorial te felicito.

he corrido el programa 9-3 pero veo que nunca entrega mas de 2 divisores para los no primos a pesar que me he asegurado que está int Div[32] y no int Div[2] . De hecho he bajado tu programa literalmente y tiene ese inconveniente., es decir por ejemplo; el divisor para 8 es el numero 2 y que pasa con el 4?,. otro ejemplo: los divisores para 64 es 2 y que pasa con con 4, 8, 16, 32?. para 20 es 2 y 5 faltaría el 10.
Bueno hasta el momento no he podido deducir que ocurre.

Reply

admin *14 Ene 2016*

Hola Edwin. El programa 9_3 calcula los divisores primos de un numero que le das y por eso no considera el 4 o el 6, ya que los divisores primos son 2 y 3
EN cuanto al problema de que solo te devuelve dos divisores, es porque pruebas numeros relativamente pequeños. si pruebas con el numero 210, por poner un ejemplito, veras como te devuelve 2,3,5,7

Reply

Edwin *15 Ene 2016*

ok gracias, tenias razón.

Reply

Daniel Cano *16 Ene 2016*

Hola una pregunta, pero antes decirte que se agradecen mucho estos tutoriales, cada vez se me hace más fácil tanto leer como interpretar código. La cosa es que en las funciones Divisores(int x) y Primo(int x) no acabo de entender porque pones esa x, se supone que donde la x tendría que ir la información que va a entrar no? Quiero decir no devería ir (por ejemplo) “int i” ya que esta variante existe después??

admin *17 Ene 2016*

Hola Daniel, esa x con su definicion de int es la forma en que el compilador de C++ espera de que le informemos de que vamos a pasarle una variable a la funcion del tipo que le indicamos. Le damos un nombre como x, para poder indicar debajo como usar ese valor en la funcion

Reply

Damian *26 Mar 2016*

Hola que tal, otra vez gracias por el curso que es sin dudas el mejor que he visto! Le hago una consulta, así como intro es igual a '\n' o tabulador es igual a '\t' como puedo saber otras teclas y demás para seguir jugando con esto hasta machacarlo en mi cerebro? desde ya muchas gracias!

Reply

admin *28 Mar 2016*

Damian, me parece recordar que puedes buscar la lista. Escape codes en c++ y enseguida encontraras en google la lista de comandos disponibles

Reply

Reply

GIVE A REPLY

Conectado como Javier. ¿Quieres salir?

Message

Post comment

Sí, agrégame a tu lista de correos.

CATEGORIAS DE LOS PRODUCTOS

Selecciona una categoría