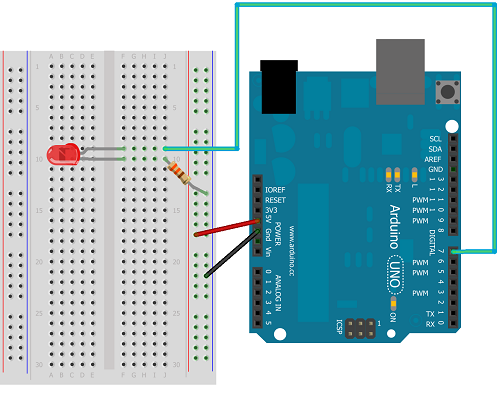
***ΠΡΟΓΡΑΜΜΑΤΙΖΟΥΜΕ ΜΕ ΤΟ ARDUINO- ΠΑΙΖΟΥΜΕ ΜΕ ΤΑ LEDS***

***Για την υλοποίηση όλων των εργασιών χρειάζονται :***

* Arduino kit
* 4 x LED
* 4 x  270 - 470 Ohm Αντίσταση (παθητική) ¼ watt
* Jumper – Διάφορα καλώδια σύνδεσης

**Κατασκευάζοντας πρακτικά το κύκλωμα με 1 LED**



**1. Το led είναι συνέχεια αναμμένο (κώδικας)**

int ledPin = 7;  
void **setup**()  
{  
 pinMode(ledPin, OUTPUT);   
}  
void **loop**()  
{  
 digitalWrite(ledPin, HIGH);   
}

ΕΞΗΓΟΥΜΕ ΤΟΝ ΚΩΔΙΚΑ

Η διαδικασία **setup** χρησιμοποιείται για τον κώδικα που πρόκειται να εκτελεστεί όταν το Arduino λάβει για πρώτη φορά ρεύμα (power). Επίσης βλέπουμε ότι στον κώδικα μας χρησιμοποιήσαμε μια εντολή **pinMode** για να ορίσουμε τον **ακροδέκτη 7** του Arduino σαν πόρτα-ακίδα **εξόδου** (**OUTPUT** ). Τέλος στην  διαδικασία βρόχου (**loop**), όπως υποδηλώνει και το όνομά της, εκτελείται το τμήμα κώδικα που αυτή εμπερικλείει «επανειλημμένα». Όταν το τσιπ Arduino εκτελέσει όλες τις οδηγίες στη διαδικασία βρόχου (**loop**), επιστρέφει στην πρώτη εντολή του βρόχου και ξεκινά πάλι. Στον κώδικα μας έχουμε χρησιμοποιήσει μια δήλωση **digitalWrite**  για να «φέρει» στον ακροδέκτη 7 την έξοδο μιας υψηλής τάσης (**HIGH**) και τελικώς την ενεργοποίηση (άναμμα) του LED. Είναι ξεκάθαρο ότι αυτό απαιτείται-χρειάζεται  να το κάνουμε μόνο μία φορά, αλλά επειδή πρόκειται για μία μόνο δήλωση, τίποτα δεν θα συμβεί εάν αυτό επαναλαμβάνεται.

**2.Τροποποιώντας τον κώδικα για να αναβοσβήνει το LED**

Ο παρακάτω κώδικας κάνει το συνδεδεμένο LED να αναβοσβήνει με σταθερό και ελεγχόμενο ρυθμό.

int ledPin = 7;  
void **setup**()  
{  
  pinMode(ledPin, OUTPUT);   
}  
void **loop**()  
{  
  digitalWrite(ledPin, HIGH);   
  delay(1000);  
  digitalWrite(ledPin, LOW);

  delay(1000);  
}

Η νέα δήλωση που παρουσιάζεται εδώ είναι η δήλωση καθυστέρησης (**delay).** Ο αριθμός στις αγκύλες είναι ο αριθμός των χιλιοστών του δευτερολέπτου που θέλουμε να περιμένει το Arduino πριν εκτελέσει την επόμενη εντολή. Εδώ έχουμε χρησιμοποιήσει 1000 ms, είναι δηλαδή μια καθυστέρηση ενός δευτερολέπτου.  Μπορείτε να αλλάξετε αυτόν τον αριθμό για να επιτύχετε ή επιβραδύνετε το αποτέλεσμα.

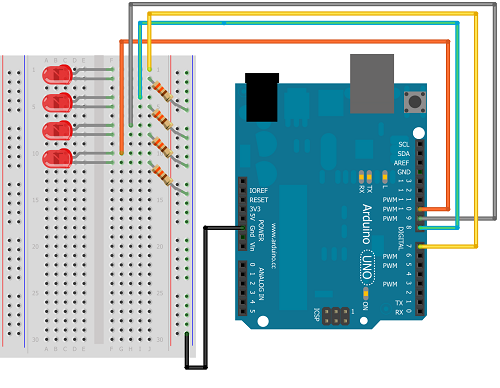
**3.Τροποποιώντας τον κώδικα για να αναβοσβήσει το LED συγκεκριμένες φορές**

Σε αυτό το παράδειγμα επιλέγουμε το LED να αναβοσβήσει μόνον 6 φορές για το σκοπό αυτό ο παρακάτω κώδικας προσθέτει μια μεταβλητή **d** (global variable **d)** για να παρακολουθεί πόσο  φορές το LED έχει ενεργοποιηθεί και απενεργοποιηθεί. Στην αρχή αυτή η μεταβλητή «τίθεται» με τιμή ίση με το 0 και παρακάτω στην αρχή του βρόχου (loop) αυτή η μεταβλητή αυξάνεται κατά 1 (d++) σε κάθε ένα κύκλο. Γίνεται έλεγχος και εάν δεν έχει υπερβεί το 6(d<=6 και η if), η ενδεικτική λυχνία συνεχίζει να αναβοσβήνει με τον ρυθμό που ορίζουν οι δηλώσεις **delay**. Ασφαλώς μόλις αυτή η μεταβλητή ξεπεράσει την οριζόμενη τιμή του 6 τότε το πρόγραμμα τερματίζεται.

int ledPin = 7;  
long d = 0;  
void **setup**()  
{  
  pinMode(ledPin, OUTPUT);   
}  
void **loop**()  
{  
  d++;  
  if (d<=6)  
  {  
    digitalWrite(ledPin, HIGH);   
    delay(1000);  
    digitalWrite(ledPin, LOW);   
    delay(1000);  
  }

**4.Επεκτείνοντας το κύκλωμα για 4 LED**

Τώρα στην παραπάνω κατασκευή ας τοποθετήσουμε ακόμη άλλα 3 LEDS -με τις αντίστοιχες αντιστάσεις τους- το ένα δίπλα στο άλλο σε σειρά.Τελικά η κατασκευή μας θα πρέπει να καταλήξει να μοιάζει με την παρακάτω εικόνα:



Εάν υλοποιήσετε αυτό το παραπάνω project, αυτό που βλέπετε στη φωτογραφία είναι όλο το κύκλωμα με τις περισσότερες καλωδιώσεις που θα μπορούσε να γίνει. **Βεβαιωθείτε ότι ο μακρύτερος αγωγός (ποδαράκι) που είναι και ο θετικός αγωγός του κάθε LED είναι εκείνος που τελικά συνδέεται απευθείας με τον εκάστοτε ακροδέκτη του Arduino, καθώς επίσης  και ότι οι αντιστάσεις συνδέουν τον βραχύτερο αγωγό (ποδαράκι) που είναι το (αρνητικό) στην κοινή θέση του breadboard, τώρα την θεωρούμε γείωση (GND).**

int ledPin[ ] = {7,8,9,10};  
void **setup**()  
{  
  for (int i =0;i<4;i++)  
  {  
    pinMode(ledPin[i], OUTPUT);  
  }  
}  
void **loop**()   
{  
  for (int i =0;i<4;i++)  
  {  
    digitalWrite(ledPin[i], HIGH);  
  }  
}

Παρατηρούμε ότι στην κορυφή του παραπάνω προγράμματος (κώδικα), **χρησιμοποιούμε έναν πίνακα** για να **αποθηκεύσουμε τη λίστα των ακίδων Arduino που έχουμε συνδέσει με τις λυχνίες LED**. Αμέσως μετά στη ρύθμιση, χρησιμοποιούμε ένα βρόχο (loop) για να ορίσουμε κάθε έναν από αυτούς τους ακροδέκτες-ακίδες να είναι ακίδες **εξόδου**. Επίσης μέσα σε αυτόν τον κύριο βρόχο, χρησιμοποιούμε ακόμη έναν άλλο βρόχο (loop) για να γυρίσουμε όλους αυτούς τους ακροδέκτες-ακίδες να γίνουν **HIGH**  (on).

Αντιγράψτε και εκτελέστε αυτόν τον κώδικα όπως είναι για να ελέγξετε ότι όλες οι λυχνίες LED ανάβουν. Στη συνέχεια, αλλάξτε το HIGH (υψηλό)  {digitalWrite(ledPin[i], HIGH);} σε LOW (χαμηλό) {digitalWrite(ledPin[i], LOW);} για να ελέγξετε επίσης ότι όλες πηγαίνουν στο off  (σβήνουν).

int ledPin[] = {7,8,9,10};  
void **setup**()  
{  
  for (int i =0;i<4;i++)  
  {  
    pinMode(ledPin[i], OUTPUT);  
  }  
}  
void **loop**()   
{  
  for (int i =0;i<4;i++)  
  {  
    digitalWrite(ledPin[i], LOW);  
  }  
}

Εάν κάποια από τις λυχνίες LED παρατηρήσετε ότι δεν ανάβει, τότε σαν πρώτο βήμα βεβαιωθείτε ότι έχετε  συνδέσει σωστά στο θετικό (μακρύτερο ποδαράκι) το καλώδιο που έρχεται από έναν ακροδέκτη (7,8,9,10) του Arduino και κατά δεύτερον ότι το κάθε LED διαθέτει την δική του παθητική αντιστάτη που συνδέει το αρνητικό (μικρότερο ποδαράκι) με το κοινό σημείο στο breadboard που εκεί καταλήγει και το καλώδιο από το GND του Arduino. Επίσης συχνά παρατηρείται να μην κάνει ηλεκτρική επαφή το ίδιο το breadboard καίτοι υπάρχει οπτικά μηχανική επαφή. (Κουνήστε λίγο τους ακροδέκτες που είναι καρφωμένοι σε αυτό ή αλλάξτε θέση). Εάν παρόλα αυτά κάποιο LED αρνείται να συνεργαστεί και έχετε κάνει όλα τα παραπάνω καθώς και  όλες οι συνδέσεις είστε βέβαιοι ότι είναι σωστές, δοκιμάστε να αντικαταστήσετε αυτή τη λυχνία LED με μια άλλη από το κιτ.

**5.Προγραμματίζοντας το Arduino να παρουσιάζει δυαδικούς αριθμούς**

Για να μπορέσουμε να κατανοήσουμε αυτό το βήμα θα πρέπει να έχουμε κάποιες θεωρητικές γνώσεις σχετικά με τους δυαδικούς αριθμούς. Μελετούμε τον παρακάτω πίνακα για τους πρώτους 16 δυαδικούς αριθμούς καθώς και την αντιστοιχία τους με το δεκαδικό αριθμητικό σύστημα και τα πράγματα θα γίνουν απλά.

**23 22 21 20**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Binary digit **3** | Binary digit **2** | Binary digit **1** | Binary digit **0** | Decimal |
| 0 | 0 | 0 | 0 | **0** |
| 0 | 0 | 0 | 1 | **1** |
| 0 | 0 | 1 | 0 | **2** |
| 0 | 0 | 1 | 1 | **3** |
| 0 | 1 | 0 | 0 | **4** |
| 0 | 1 | 0 | 1 | **5** |
| 0 | 1 | 1 | 0 | **6** |
| 0 | 1 | 1 | 1 | **7** |
| 1 | 0 | 0 | 0 | **8** |
| 1 | 0 | 0 | 1 | **9** |
| 1 | 0 | 1 | 0 | **10** |
| 1 | 0 | 1 | 1 | **11** |
| 1 | 1 | 0 | 0 | **12** |
| 1 | 1 | 0 | 1 | **13** |
| 1 | 1 | 1 | 0 | **14** |
| 1 | 1 | 1 | 1 | **15** |

Εδώ θα χρησιμοποιήσουμε τα ενδεικτικά LEDs με την εξής παρακάτω λογική:**Όταν το LED είναι αναμμένο (on) [δηλαδή παράγει φως]  θεωρούμε ότι τότε συμβολίζει το λογικό: 1**

**Όταν το LED είναι σβηστό (off) [δηλαδή δεν παράγει φως] θεωρούμε ότι τότε συμβολίζει το λογικό:  0**

**Με αυτή την παραδοχή θα μπορούσαμε με το κατάλληλο πρόγραμμα-κώδικα να αναπαραστήσουμε  σύμφωνα με τον παραπάνω πίνακα αριθμούς από το 0 έως το 15**

Για λόγους ευκολίας και απλότητας -από το όπως φαίνεται στην εικόνα της κατασκευής- περιστρέψτε και τοποθετήστε το breadboard 90° κατά τη φορά των δεικτών του ρολογιού και κατόπιν προσπαθήστε να συσχετίσετε το κύκλωμά σας με το τον παραπάνω δοσμένο πίνακα των τιμών δυαδικών θέσεων και την αντιστοιχία τους με το  δεκαδικό.

Τώρα πλέον η πρώτη λυχνία LED που βρίσκεται δεξιότερα στο breadboard είναι συνδεδεμένη με τον ακροδέκτη 7 και γίνεται η αναπαράσταση του δεξιού δυαδικού ψηφίου **(Binary digit 0)** (μονάδες).

Όταν αυτή η λυχνία LED είναι ενεργοποιημένη (αναμμένο LED), αυτό θα σημαίνει **1**. Προχωρώντας προς τα αριστερά, στο δίπλα ακριβώς LED έχουμε το **(Binary digit 1),** στο πάρα δίπλα έχουμε το  **(Binary digit 2)** και τελικώς στο δεξιότερο LED στην έχουμε το **(Binary digit 3)**

**Βασικά για τον πίνακα θα μπορούσαμε να πούμε ότι έχουμε από δεξιά προς τα αριστερά 1, 2, 4 & 8 για τις επικεφαλίδες της στήλης μας. Που δεν είναι τίποτε άλλο παρά οι δυνάμεις του 2**

**20 =1           21=2          22=4         23=8**

Στον παρακάτω κώδικα, έχουμε μια νέα διαδικασία για την εμφάνιση του δυαδικού αριθμού που θέλουμε να δείξουμε. **Οι αριθμοί στη γλώσσα προγραμματισμού αποθηκεύονται στην πραγματικότητα σε δυαδική μορφή ακόμη και όταν ορίζουμε τις τιμές τους χρησιμοποιώντας decimal.** Έτσι μπορούμε να εκμεταλλευτούμε αυτό το γεγονός για να αποφύγουμε να γράψουμε αρκετό κώδικα για να κάνουμε τη μετατροπή (αν και θα μπορούσαμε). Στον κώδικα αυτό, ένας βρόχος (loop) χρησιμοποιείται για να διαβάσει τις τιμές των πρώτων τεσσάρων bits του δεκαδικού αριθμού που αποστέλλεται στη διαδικασία.

Η **συνάρτηση** **bitRead**  **{ if (bitRead(numToShow, i)==1)}** που βλέπετε «λέει» αν υπάρχει 1 ή 0 σε μια δεδομένη τιμή θέσης. Εάν βρεθεί ένα 1, τότε ο ακροδέκτης-pin του Arduino μεταβαίνει σε κατάσταση HIGH δηλαδή ανάβει η λυχνία LED που είναι συνδεδεμένη στο αντίστοιχο pin, αν βρεθεί 0 τότε σβήνει η λυχνία LED που είναι συνδεδεμένη σε αυτό το pin για τη θέση αυτή.

Δοκιμάστε αυτόν τον κωδικό με όλους τους διαφορετικούς δεκαδικούς αριθμούς από το 0 έως και το 15. Μπορείτε επίσης να βάλετε τιμές έως και 255  **(28)**. Οι αριθμοί όμως σε αυτό το εύρος θα χρειαστούν 8 στήλες αξίας θέσης για να αναπαρασταθούν (άρα και ισάριθμα LEDs). Δεδομένου ότι ο κώδικάς μας αγνοεί τις τελευταίες 4 τιμές θέσης, οι λυχνίες LED θα είναι ενεργοποιημένες μόνο εάν ο αριθμός αυτός έχει 1 σε μία ή περισσότερες από τις πρώτες 4 στήλες αξίας θέσης.

Παρατηρήστε ότι **όλοι οι περιττοί αριθμοί** **θα έχουν την τιμή της πρώτης θέσης** άρα **θα ανάβει το αντίστοιχο LED**, ενώ **όλοι οι άρτιοι αριθμοί θα σημαίνει ότι δεν ανάβει το LED**.

int ledPin[] = {7,8,9,10};  
void **setup**()  
{  
  for (int i =0;i<4;i++)  
  {  
    pinMode(ledPin[i], OUTPUT);  
  }  
}  
void **loop**()   
{  
  displayBinary(2);  
}  
void displayBinary(byte numToShow)  
{  
  for (int i =0;i<4;i++)  
  {  
    if (bitRead(numToShow, i)==1)  
    {  
      digitalWrite(ledPin[i], HIGH);   
    }  
    else  
    {  
      digitalWrite(ledPin[i], LOW);   
    }  
  }  
}

**6.Προγραμματίζοντας το Arduino να μετράει προς τα πάνω**

Αυτός  ο κώδικας προγράμματος χρησιμοποιεί την διαδικασία του **displayBinary** που γράψαμε παραπάνω για να μετρήσει προς τα πάνω (από το 0 προς το 15) στο δυαδικό σύστημα.

Η χρονοκαθυστέρηση μεταξύ της εμφάνισης των αριθμών (600 ms) μας επιτρέπει να προλάβουμε να δούμε με σαφήνεια πως έγινε η αλλαγή για να εμφανισθεί ο επόμενος αριθμός στην ακολουθία.

int ledPin[] = {7,8,9,10};  
void **setup**()  
{  
  for (int i =0;i<4;i++)  
  {  
    pinMode(ledPin[i], OUTPUT);  
  }  
}  
void **loop**()   
{  
  for (byte counter =0;counter<=15; counter++)  
  {  
    displayBinary(counter);  
    delay(600);

  }  
}  
void displayBinary(byte numToShow)  
{  
  for (int i =0;i<4;i++)  
  {  
    if (bitRead(numToShow, i)==1)  
    {  
      digitalWrite(ledPin[i], HIGH);   
    }  
    else  
    {  
      digitalWrite(ledPin[i], LOW);   
    }  
  }  
}

**7.Προγραμματίζοντας το Arduino  - Μοτίβα**

Τώρα που μπορούμε να «παρουσιάσουμε» οποιονδήποτε αριθμό από το 0 έως το 15 σε δυαδική μορφή, θα μπορούσαμε να επωφεληθούμε από αυτό το γεγονός και να μετατρέψουμε οποιονδήποτε αριθμό, από τα φωτεινά LEDs χρησιμοποιώντας μόνο έναν αριθμό ως αναφορά. Δείτε τον παρακάτω κώδικα όπου μια σειρά αριθμών χρησιμοποιείται για να δημιουργήσει ένα τέτοιο φαινόμενο όπου το φως φαίνεται σαν να «αναπηδά» από το ένα LED στο άλλο.

int ledPin[] = {7,8,9,10};  
void **setup**()  
{  
  for (int i =0;i<4;i++)  
  {  
    pinMode(ledPin[i], OUTPUT);  
  }  
}  
void **loop**()   
{  
  byte nums[] = {0, 1, 3, 6, 4, 12, 8, 12, 4, 6, 3, 1, 0};  
  for (byte i = 0; i<13;i++)  
  {  
    displayBinary(nums[i]);  
    delay(25);  
  }   
}  
void displayBinary(byte numToShow)  
{  
  for (int i =0;i<4;i++)  
  {  
    if (bitRead(numToShow, i)==1)  
    {  
      digitalWrite(ledPin[i], HIGH);   
    }  
    else  
    {  
      digitalWrite(ledPin[i], LOW);   
    }  
  }  
}

**Δραστηριότητα: Οπτικός Κώδικας Mors με LED**

Χρησιμοποιώντας μόνο το πρώτο κύκλωμα με το ένα LED, γράψτε πρόγραμμα που να υλοποιεί τον οπτικό κώδικα Morse για τα 3 πρώτα γράμματα από το επώνυμό σας. Για το λόγο αυτό χρησιμοποιήστε ένα συνδυασμό μακρών και σύντομων καθυστερήσεων για να κάνετε το LED να αναβοσβήσει το επώνυμό σας. Βάλτε μια μεγαλύτερη καθυστέρηση στο τέλος του μηνύματος και στη συνέχεια αφήστε τον κώδικα να επαναλαμβάνει το επώνυμό σας ξανά και ξανά.

Επεξηγήσεις σχετικά με τον κώδικα Morse

**Βήματα και παραδοχές για την υλοποίηση της δραστηριότητας 2**

* Γνωρίζουμε ότι κάθε γράμμα της αλφαβήτου αποτελείται από ένα μονοσήμαντο συνδυασμό από ΤΕΛΕΙΕΣ και ΠΑΥΛΕΣ. Όπου παύλα (**dash**) θεωρούμε τον «*μεγάλο χρόνο*» που παραμένει αναμμένο το LED ενώ τελεία (**dot)** το «*μικρό χρόνο*» που παραμένει αναμμένο το LED.
* Ξεκινήστε επεξεργαζόμενοι το χρόνο που θα αφήσετε αναμμένη την ενδεικτική λυχνία LED για την τελεία (**dot)**. Αυτό είναι το μήκος μίας τελείας (**dot length**).
* Το μήκος της παύλας (**dash length**) πρέπει να είναι 3 φορές το μήκος της τελείας.
* Ανάμεσα σε κάθε τελεία ή παύλα πρέπει να υπάρχει ένα διάκενο για να ξεχωρίζουμε το ένα στοιχείο από το άλλο (δηλαδή διάκριση μεταξύ τελείας και παύλας) με ένα μήκος χρόνου (όπου η ενδεικτική λυχνία θα είναι σβηστή).
* Ανάμεσα σε κάθε πλήρες γράμμα, (δηλαδή διάκριση μεταξύ γραμμάτων)  πρέπει να υπάρχει κενό ίσο με το μήκος μιας παύλας (όπου η ενδεικτική λυχνία θα είναι σβηστή).
* Μεταξύ κάθε πλήρους λέξης, θα πρέπει να υπάρχει ένα κενό ίσο με το 7 φορές το μήκος της μιας τελείας (όπου η ενδεικτική λυχνία θα είναι σβηστή).

Παρακάτω βλέπετε την αντιστοιχία κάθε γράμματος με τον κώδικα Morse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Γράμμα** | **Κώδικας Morse** |  | **Γράμμα** | **Κώδικας Morse** |
| **A** | **· ─** |  | **Ν** | **─ ·** |
| **B** | **─ · · ·** |  | **Ξ** | **─ · · ─** |
| **Γ** | **─ ─ ·** |  | **Ο** | **─ ─ ─** |
| **Δ** | **─ · ·** |  | **Π** | **· ─ ─ ·** |
| **E** | **·** |  | **Ρ** | **· ─ ·** |
| **Ζ** | **─ ─ · ·** |  | **Σ** | **· · ·** |
| **Η** | **· · · ·** |  | **Τ** | **─** |
| **Θ** | **─ · ─ ·** |  | **Υ** | **─ · ─ ─** |
| **I** | **· ·** |  | **Φ** | **· · ─  ·** |
| **Κ** | **─ · ─** |  | **Χ** | **─ ─ ─ ─** |
| **Λ** | **· ─ · ·** |  | **Ψ** | **· · ─** |
| **Μ** | **─ ─** |  | **Ω** | **· ─ ─** |

### Απλοποίηση του κώδικα με χρήση συναρτήσεων

Μπορεί να γίνει ευκολότερα και με πιο ευανάγνωστο κώδικα χρησιμοποιώντας συναρτήσεις. Έστω ότι θέλουμε να μετατρέψουμε τον παρακάτω κώδικα σε συνάρτηση

  digitalWrite(pin, HIGH);

  delay(400); //led on signaling dot

  digitalWrite(pin, LOW);

  delay(400); // led off between dots and dashes

Τότε αντί για τον παραπάνω κώδικα θα γράψουμε dot();

και στο τέλος του κώδικα (μετά και το τελευταίο άγκιστρο) θα γράψουμε

void dot()

{

  digitalWrite(pin, HIGH);

  delay(400); //led on signaling dot

  digitalWrite(pin, LOW);

  delay(400); // led off between dots and dashes

}

Ομοίως μπορούμε να φτιάξουμε μια συνάρτηση για κάθε γράμμα