Στην τρίτη άσκηση κινήθηκα ως εξής:

Αφού συμπεριέλαβα τις απαραίτητες βιβλιοθήκες, έκανα define στο πλήθος των παικτών, στις γραμμές και στήλες και επομένως και στον υπολογισμό του πλήθους καρτών, καθώς αφορούν στοιχεία που μπορεί να αλλάξουν στο μέλλον και έτσι διευκολύνεται ο κώδικάς μου σε τυχόν αλλαγές.

Μετά όρισα με typedef δύο τύπους struct, έναν για τον παίκτη και έναν για την κάρτα με τα αντίστοιχα πεδία όπως αυτά ζητούνται από το παιχνίδι.

Ακολούθησε η δήλωση των συναρτήσεων που θα χρησηποποιήσω στη main.  
 Κατά τη δήλωση των μεταβλητών δήλωσα και δύο struct paiktes και kartes ως μονοδιάστατους πίνακες. Ενώ στον ενδεικτικό πίνακα στο τέλος της εκφώνησης προτείνεται ως γνωστικό αντικείμενο οι πολυδιάστατοι πίνακες, έκρινα πως το συγκεκριμένο πρόγραμμα μπορεί να λειτουργήσει εξίσου και με μονοδιάστατους πίνακες. Η χρήση δισδιάστατου πίνακα ανεβάζει την πολυπλοκότητα της επίλυσης, δίχως –κατά τη γνώμη μου–να προσφέρει κάτι περισσότερο στο πρόγραμμα. Από τη στιγμή που δεν είναι ρητή απαίτηση από την εκφώνηση , επέλεξα την λύση μέσω μονοδιάστατων πινάκων.

Ακολούθως έγινε κλήση της σχετικής συνάρτησης εισαγωγής των παικτών. Χρησημοποιώντας με αναφορά ένα δείκτη p σε struct paiktis\_t με τη χρήση strcpy γράφω σε κάθε θέση του πίνακα στο πεδίο p.name το αντίστοιχο όνομα παίκτη. Επίσης βάζω σε κάθε πεδίο p.score το 0 ασφαλώς και το αντίστοιχο id χαρακτήρα στο πεδίο p.id .

Για να τυπώσω μια δωδεκάδα τυχαίων καρτών καλώ τις εξής συναρτήσεις:

srand(time(NULL)) Έτοιμη συνάρτηση στη βιβλιοθήκη time.h ώστε να δίνει τυχαίους αριθμούς κάθε φορά που καλείται η rand

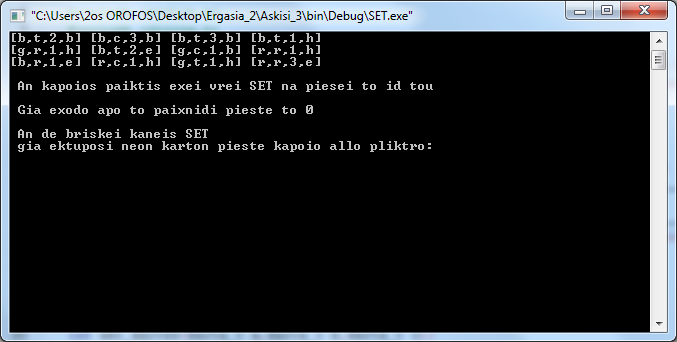
tuhaies\_kartes(kartes) Για να βγουν τυχαίες και διαφορετικές κάρτες. Η συνάρτηση αυτή καλεί 2 ακόμη συναρτήσεις-τις getNextCard και idies\_kartes.

Η getNextCard (με όρισμα με αναφορά ενός struct k τύπου karta\_t)λειτουργεί ως εξής: Καταχωρεί στη μεταβλητή r το υπόλοιπο της διαίρεσης με το 3(modulo (% )3) των τυχαίων αριθμών που μας δίνει η έτοιμη συνάρτηση rand, ώστε η r να λαμβάνει με τυχαίο τρόπο 3 τιμές μόνο (0,1,2). Με τη χρήση switch καταχωρούμε στην ιδιότητα του χρώματος με αναφορά στο πεδίο (\*k).colour-γι’αυτό και βάζουμε βελάκι - το αντίστοιχο χρώμα για κάθε τιμή που μπορεί να λάβει η r. Ομοίως και με τις άλλες ιδιότητες. Στο πλήθος είναι πιο απλό. Προσαυξάνουμε +1 το αποτέλεσμα της rand και το καταχωρούμε στο k->number, ώστε να λαμβάνει πλέον τις τιμές 1 , 2 ή 3 (αντί 0, 1 ή 2).

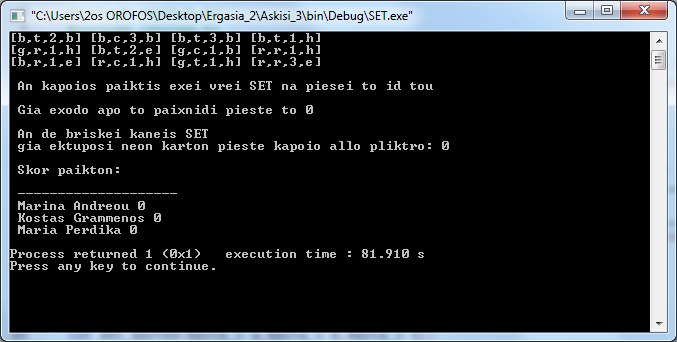
Η idies\_kartes(με ορίσματα 2 κάρτες τύπου karta\_t) είναι τύπου int που μας επιστρέφει την τιμή 1 όταν ταυτόχρονα όλες οι ιδιότητες των δύο καρτών είναι ίδιες. Σε αντίθετη περίπτωση η συνάρτηση επιστρέφει την τιμή 0

Επιστρέφουμε στη συνάρτηση tuhaies\_kartes. Με μια for διατρέχει το πλήθος καρτών και γεμίζει κάθε θέση με μια κάρτα κάνοντας κλήση της getNextCard(με όρισμα τη διεύθυνση της i θέσης του struct k. Μετά με δύο εμφωλευμένες for η εσωτερική for ελέγχει για κάθε θέση i όλες τις προηγούμενες x εκδοθείσες κάρτες για ισότητα με την κλήση της συνάρτησης idies\_kartes. Αν είναι ίσες εκδίδει νέα κάρτα με κλήση της getNextCard και ο x αρχικοποιείται σε κάθε γύρο στην τιμή -1, έτσι ώστε με το βήμα αύξησης x++ να λαμβάνει την τιμή 0 και να επανεκινάει η for τον έλεγχο για ίδιες κάρτες από την πρώτη κάρτα μέχρι την τρέχουσα(δηλαδή για όλες τις κάρτες που έχουν ήδη εκδοθεί πριν από την τρέχουσα κάρτα i).Αν οι κάρτες δεν είναι ίδιες τότε βγαίνει από τον εσωτερικό βρόχο for και συνεχίζει η εξωτερική for για την επόμενη κάρτα και ούτω καθεξής, μέχρι το πλήθος καρτών.

ektuposi\_karton(kartes) Τυπώνει τις εκδοθείσες κάρτες με τη βοήθεια μιας for διατρέχοντας το πλήθος καρτών και με μια printf με ορίσματα τα 4 πεδία των ιδιοτήτων του struct k τύπου karta\_t. Με χρήση μιας if( i%4==3 ) αλλάζουμε γραμμή κάθε 4 στοιχεία για να εμφανιστούν οι κάρτες σε πλέγμα 3 γραμμές Χ 4 στήλες.



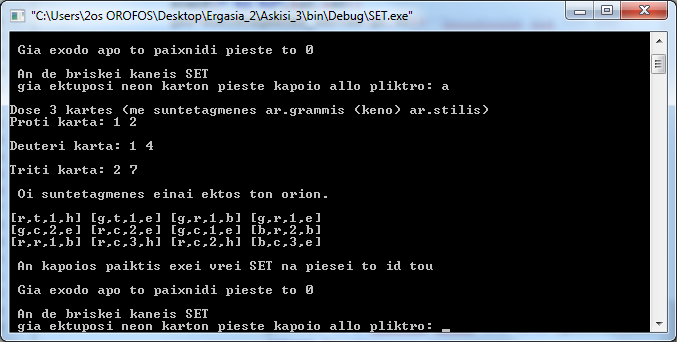
Μετά ξεκινάει το loop που τερματίζει μόλις κάποιος παίκτης πατήσει το 0. Αυτό περιγράφεται στο if που ακολουθεί με την κλήση της συνάρτησης ektuposi\_skore, οπου με μια for διατρέχουμε το πλήθος παικτών και με μια printf τυπώνουμε τα πεδία p.name και p.score για κάθε παίκτη.



Όποιος παίκτης πατήσει το id του γίνεται ενεργός. Αυτό γίνεται ως εξής: Αρχικοποιούμε τη βοηθητική μεταβλητή energos σε μια τιμή που αποκλείεται να λάβει μετά τους ελέγχους που θα πραγματοποιήσoyme(επιλέγω το -999 !!!) Μετά διατρέχουμε το πλήθος παικτών και ελέγχουμε αν το δοθέν id ισούται με κάποιο που περιλαμβάνεται στο πεδίο του struct paiktes.id . Αν ισχύει αυτό καταχωρούμε τη θέση p στη μεταβλητή energos.Αν στο τέλος της for η μεταβλητή energos έχει αλλάξει τιμή(σε σχέση με την αρχικοποίηση στο -999), σημαίνει πως βρέθηκε υπαρκτός παίκτης.Πλέον ζητούμε από τον παίκτη να μας δώσει τις τρεις κάρτες που θεωρεί ότι σχηματίζουν ΣΕΤ, δίνοντάς μας συντεταγμένες γραμμής και στήλης για κάθε κάρτα.

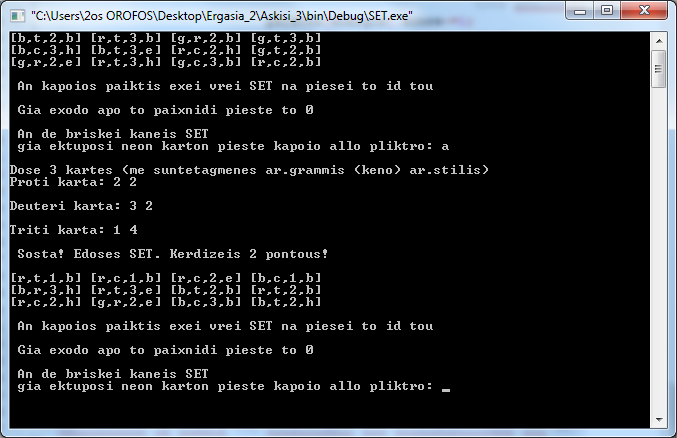
Με κλήση της συνάρτησης suntetagmenes\_kartas μετασχηματίζουμε τις δοθείσες συντεταγμένες στην αντίστοιχη θέση του μονοδιάστατου πίνακά μας ως εξής: Πολλαπλασιάζουμε τη (γραμμή-1) που μας έδωσε ο παίκτης με το 4(καθώς κάθε γραμμή έχει 4 στοιχεία) και προσθέτουμε τη (στήλη-1) και καταχωρούμε το αποτέλεσμα σε μια μεταβλητή gs, η οποία είναι και το return της int συνάρτησής μας.[Το -1 γίνεται γιατί ο παίκτης δε βλέπει τη γραμμή(και στήλη) 0 του πίνακα].

Στη συνέχεια κάνουμε έναν έλεγχο αν κάποια από τις συντεταγμένες είναι εκτός ορίων, ενημερώνουμε σχετικά και τυπώνουμε πάλι τις κάρτες(με κλήση ektuposi\_kartas).

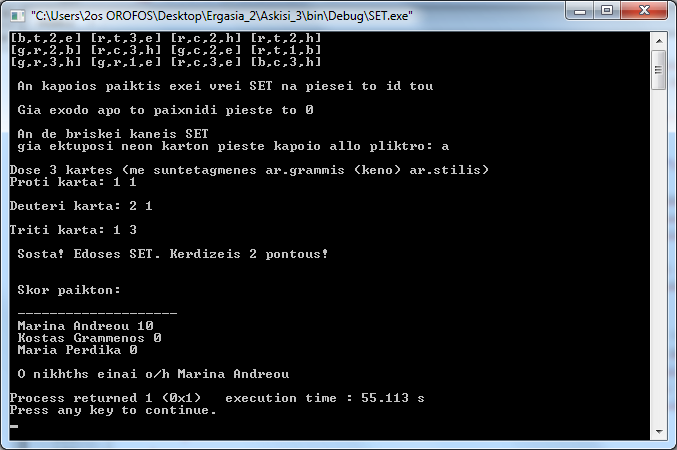


Αν είναι εντός ορίων οι συντεταγμένες καλούμε τη συνάρτηση set\_kartes(με ορίσματα 3 κάρτες με τις συντεταγμένες που δόθηκαν απο τον παίκτη-όπως αυτές μετασχηματίστηκαν μετά την κλήση της συνάρτησης suntetagmenes\_kartas). Αυτή η συνάρτηση δουλεύει ως εξής: Με εμφωλευμένα if ελέγχουμε αν για ΚΑΘΕ ιδιότητα των τριών επιλεγμένων καρτών ισχύει ΕΙΤΕ ισότητα, ΕΙΤΕ μη ισότητα ΚΑΙ για τις 3 κάρτες.Αν ισχύει αυτό η int συνάρτηση μας επιστρέφει 1, αλλιώς μας γυρνάει 0. Οπότε αν ισχύει η συνθήκη και το ΣΕΤ είναι σωστό, τότε το πρόγραμμα ενημερώνει με αντίστοιχο μήνυμα και προσθέτει στο πεδίο paiktes[energos].score 2 πόντους. Αν το σκορ του ενεργού παίκτη φτάσει ή ξεπεράσει το 10 τότε το παιχνίδι καλεί τη συνάρτηση εκτύπωσης σκορ, ανακοινώνε το όνομα του νικητή,διαβάζοντας το πεδίο paiktes[energos].name και κλείνει (return 1).Αν δεν υπάρχει ακόμη νικητής,τότε καλεί την tuhaies\_kartes για να επιλέξει νέες κάρτες προς εκτύπωση. Αν ο παίκτης απαντήσει λάθος το πρόγραμμα ελέγχει το σκορ του αν είναι μεγαλύτερο του 0 και τότε αφαιρεί έναν πόντο από το πεδίο paiktes[energos].score και ενημερώνει με αντίστοιχο μήνυμα. Αν ο παίκτης που απάντησε λάθος έχει ήδη μηδενικό σκορ, τότε το πρόγραμμα δεν αφαιρεί πόντο από τον παίκτη, απλά με μήνυμα ενημερώνει πως παραμένει με μηδενικό σκορ. Σε κάθε περίπτωση(είτε σωστής είτε λάθος απάντησης) το πρόγραμμα εκτυπώνει κάρτες. Απλά στην περίπτωση σωστής απάντησης έχει ήδη προηγουμένως κληθεί η τυχαίες κάρτες και οι κάρτες θα είναι νέες. Ενώ στην περίπτωση της λάθος απάντησης, το πρόγραμμα απλώς επανεκτυπώνει τις ίδιες κάρτες.

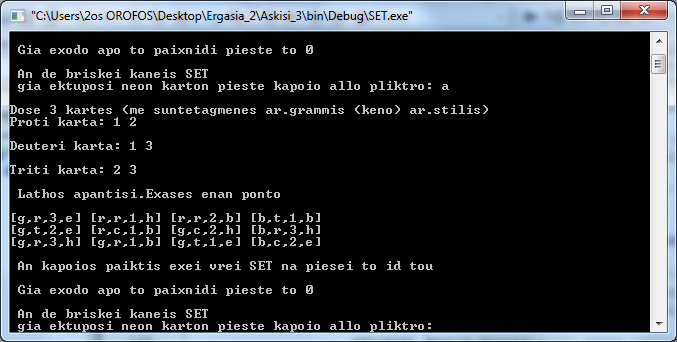
Περίπτωση εύρεσης ΣΕΤ από παίκτη:



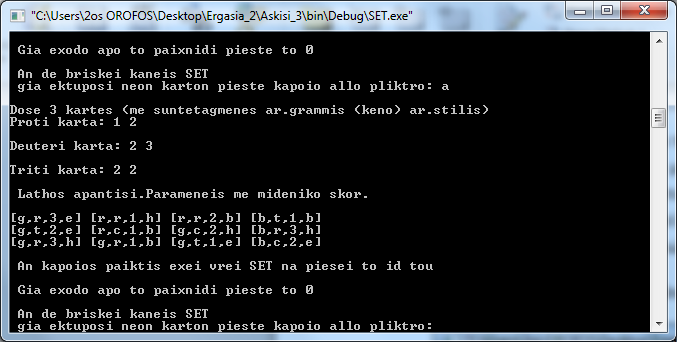
Περίπτωση σωστής απάντησης και εύρεσης νικητή (σκορ τουλάχιστον 10 πόντους)



Περίπτωση λάθος απάντησης παίκτη με σκορ μεγαλύτερο από το μηδέν:



Περίπτωση λάθος απάντησης παίκτη με μηδενικό σκορ:



Πλέον είμαστε στην περίπτωση όπου η μεταβλητή energos δεν έχει αλλάξει τιμή και παραμένει στην αρχικοποίηση που της δώσαμε(τιμή -999). Αυτό σημαίνει πως το id που δόθηκε δεν είναι κάποιο από αυτά που περιλαμβάνονται στο πεδίο paiktes.id, άρα κανείς δε βρίσκει ΣΕΤ και το παιχνίδι ανακοινώνει την έκδοση νέων καρτών με σχετικό μήνυμα και καλεί τις συναρτήσει tuhaies\_kartes και ektuposi\_karton(πχ. παρακάτω πατήθηκε το v):

