|  |
| --- |
| giorgos-1001@hotmail.com |
| C++ |
| 3 Set C++ 2020 |

|  |
| --- |
| Γιώργος Ντάκος: \\\Α.Μ: 1059569 /// Έτος: Γ΄  24/5/2020 |

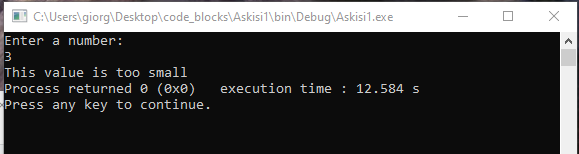
**ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ(C++)**

**Άσκηση 1**

**Ερώτημα 1:**

Γράφουμε το πρόγραμμα μας και μετά το κάνουμε compile και περνούμε τα έξεις αποτελέσματα για κάθε περίπτωση:

Για value=3:

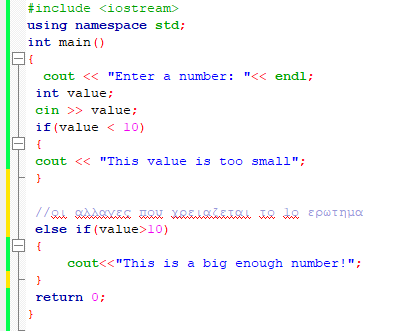


Για value=12:



Στην αρχή του προγράμματος έχουμε την καθιερωμένη βιβλιοθήκη <iostream> που έχει τα καθιερωμένα αντικείμενα και πράξεις iostream. Στην συνέχεια με την using namespace std κάνουμε ορατά τα ονόματα του std χωρίς το πρόθεμα std::. Έπειτα ορίζουμε μια συνάρτηση την main χωρίς κάποιο όρισμα και η οποία κάνει τα εξής: Πρώτα πολλά γράφεται στο ρεύμα εξόδου cout το μήνυμα που βρίσκεται μέσα στις αγκύλες χάρης τα << και μετρά με την endl ορίζουμε το τέλος της γραμμής οπότε πάμε σε καινούργια. Μετά ορίζουμε μια μεταβλητή τύπου int και με το ρεύμα εισόδου cin γράφουμε την τιμή που θα πάρει η value από το command line. Στην συνέχεια ακολουθεί μια συνθήκη if που αν η μεταβλητή value>10 τότε τυπώνεται το αντίστοιχο μήνυμα. Στο τέλος του προγράμματος επιστρέφουμε την τιμη 0.

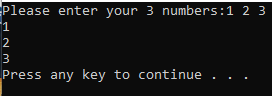
Για να εμφανίζει το μήνυμα «This is a big enough number!» όταν θα ισχύει η συνθήκη value > 10, κάνουμε τις παρακάτω αλλαγές που φαίνεται στην εικόνα :



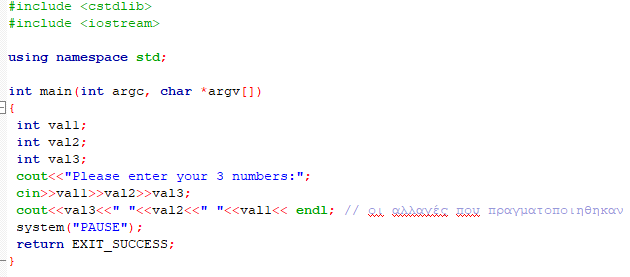
**Ερώτημα 2:**

Στον κώδικα αυτό περιλαμβάνουμε μια νέα βιβλιοθήκη την <cstdlib> η οποία περιέχει συναρτήσεις κατανομής μνήμης της C.Στην συνέχεια ορίζουμε μια συνάρτηση main με 2 μεταβλητές την μια τύπου int και την άλλη τύπου πίνακα δεικτών χαρακτήρων .Αυτό που κάνει το πρόγραμμα είναι να δέχεται 3 ακέραιους αριθμούς και να μας τους τυπώνει στην συνέχεια με την σειρά που τους βάλαμε τον έναν κάτω από τον άλλο(αυτό γιατί μετά από κάθε τιμή που τυπώνει δέχεται το όρισμα \n που δηλώνει νέα γραμμή ).Τέλος με system(“PAUSE”) σταματά το περιβάλλον command line που τρέχει το πρόγραμμα(αυτή η εντολή ισχύει για windows) και με την σταθερά return EXIT\_SUCCES επιστρέφουμε την επιτυχή εκτέλεση του προγράμματος.

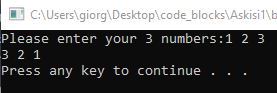
Τα αποτελέσματα:



Για να εμφανίζει τους αριθμούς που δοθήκαν με αναστροφή σειρά πριν τερματίσει το πρόγραμμα κάνουμε τις παρακάτω αλλαγές που φαίνεται στην εικόνα:



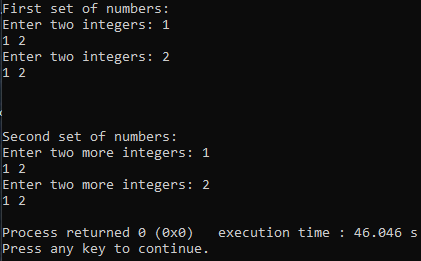
Τα αποτελέσματα:



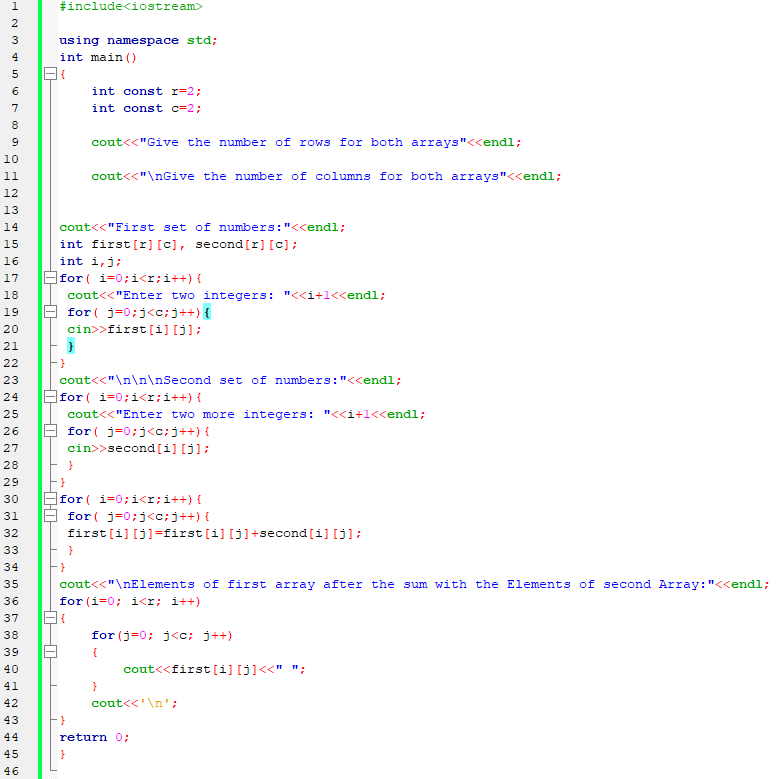
**Ερώτημα 3:**

Το συγκεκριμένο πρόγραμμα έχει ως ορίσματα 2 δισδιάστατους πίνακες που ο καθένας έχει από 2 γραμμές και 2 στήλες οπού δέχονται ακέραιους αριθμούς και άλλες 2 μεταβλητές. Στην συνέχεια έχει 2 for loop οπού το ένα είναι εμφωλευμένο μέσα στο άλλο οπότε το μήνυμα να εισάγουμε 2 αριθμούς θα τυπωθεί 2 φορές και επίσης με το 2ο for θα οπού κρατάει τις στήλες του πίνακα ενώ το 1ο for τις γραμμές θα συμπληρώσουμε όλα τα στοιχεία του πίνακα. Αυτά ισχύουν για τον πίνακα first.Tα ιδιά συμβαίνουν και για τον πίνακα second. Επίσης κάθε φορά το πρόγραμμα μας δείχνει σε ποια γραμμή του πίνακα βρισκόμαστε. Στο τέλος προσθέτει τα στοιχεία του 1ου πίνακα με τα στοιχεία του 2ου πίνακα απλά δεν τυπώνονται τα αποτελέσματα.

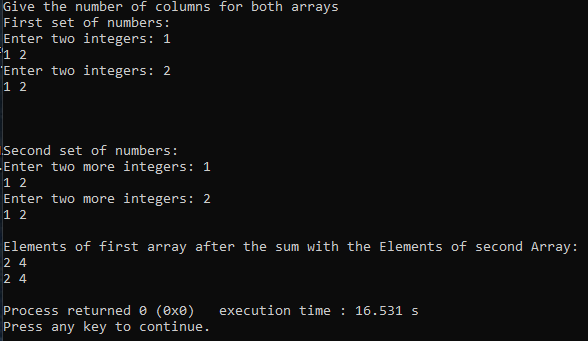
Τα αποτελέσματα:



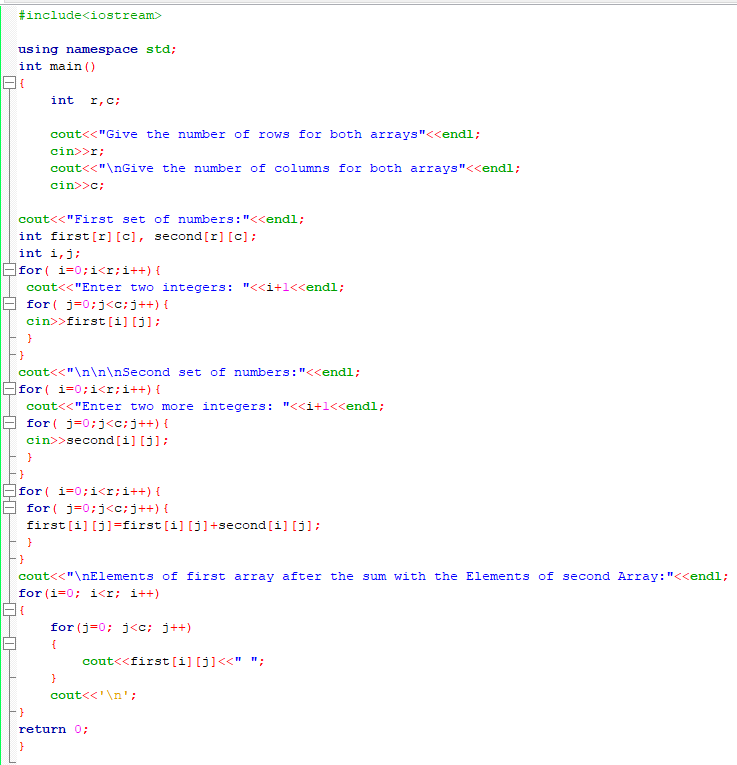
Για να πάρουμε τα δεδομένα του πίνακα 1 μετά την πρόσθεση των στοιχείων του με τα στοιχεία του δευτέρου πίνακα δημιουργούμε ένα εμφωλευμένο for οπού το πρώτο for θα είναι για τις γραμμές του πίνακα και το δεύτερο for για τις στήλες του πίνακα οπού στο δεύτερο for θα έχουμε την εκτύπωση των στοιχείων του πίνακα. Επίσης για να έχουμε δυο κοινές μεταβλητές για τους πίνακες που θα καθορίζουν τα μεγέθη των πινάκων , χρησιμοποιούμε την const μετά τον τύπο δεομένων της κάθε σταθεράς . Παρακάτω φαίνεται ο κώδικας με τις αλλαγές:



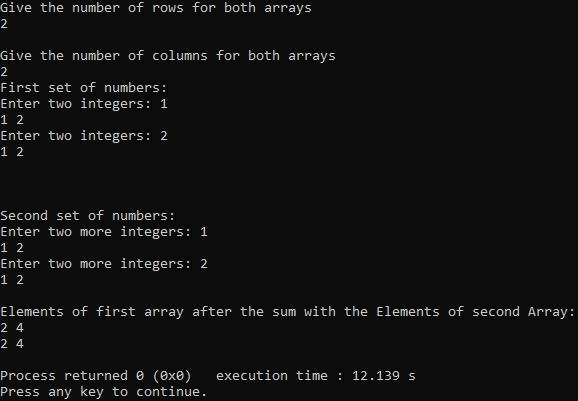
Τα αποτελέσματα:



Για να δώσει ο χρήστης τις διαστάσεις των πινάκων κάνουμε τις εξής αλλαγές που φαίνονται παρακάτω:



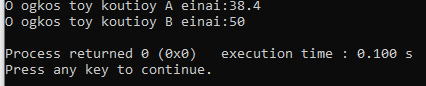
Τα αποτελέσματα:



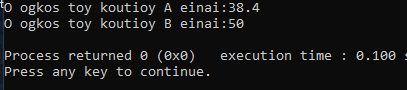
**Ερώτημα 4:**

Δημιουργούμε την κλάση όπως μας ζητάει η άσκηση. Στην συνέχεια δηλώνουμε στην main μια μεταβλητή ogkos τύπου double η οποία θα μας παρέχει τον όγκο του κάθε αντικειμένου .Έπειτα δημιουργούμε τα αντικείμενα τα οποία είναι τύπου Kouti και μετά αρχικοποιούμε τις μεταβλητές των αντικειμένων που έχουν κληρονομήσει από την κλάση. Στο τέλος υπολογίζουμε τον όγκο ξεχωριστά για κάθε αντικείμενο και διατυπώνουμε τα ανάλογα μηνύματα.

Τα αποτελέσματα φαίνονται παρακάτω:



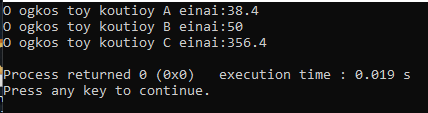
Τώρα για την δεύτερη περίπτωση αφού έχουμε private τις μεταβλητές θα φτιάξουμε μια public μέθοδο τύπου double την calculateOgkos() με 3 ορίσματα τύπου double.Έπειτα θα κάνουμε τις private μεταβλητές να ισούνται με αυτές των ορισμάτων της μεθόδου και στο τέλος της μεθόδου επιστρέφουμε τον όγκο του αντικειμένου. Μέσα στην main δημιουργούμε τα αντικείμενα KoutiA και KoutiB.Έπειτα με σωστά μηνύματα εξόδου και τις σωστές τιμές στα ορίσματα της calculateOgkos() ξεχωριστά για κάθε αντικείμενο παίρνουμε τα παρακάτω αποτελέσματα:



**Ερώτημα 5:**

Προσθέτουμε την υπερφόρτωση του τελεστή + στο public κομμάτι της κλάσης το οποίο έχει ως όρισμα ένα σταθερό δείκτη τύπου Kouti. Μετά δημιουργούμε ένα αντικείμενο kouti μέσα στην υπερφόρτωση και αρχικοποιούμε τις τιμές του αντικειμένου με τον εξής τρόπο: Το αντικείμενο κουτί καλεί την κάθε private μεταβλητή μέσω της «.» και τα βάζει να ισούται το καθένα με την πρόσθεση του αντίστοιχου μεγέθους της private με το μέγεθος που δείχνει ο δείκτης b.Με αυτό τρόπο επιτρέπουμε την πρόσθεση αντικειμένων. Μετά βγάζουμε τα ορίσματα από την calculateOgkos() και αφήνουμε μόνο να μας επιστρέφει τον όγκο του αντικειμένου. Τα μεγέθη τώρα θα τα ορίζουμε μέσα από τις 3 καινούργιες μεθόδους, setMikos(), setPlatos(), setYpsos(), οπού κάθε μια έχει το δικό της όρισμα το οποίο το βάζει να ισούται με αντίστοιχο το μέγεθος. Μέσα στην main κάνουμε της απαραίτητες αλλαγές και καλούμε τις 3 παραπάνω μεθόδους με τα καταλληλά ορίσματα και για τα 2 αντικείμενα KoutiA και KoutiB. Μετά δημιουργούμε ένα 3ο αντικείμενο KoutiC του οποίου τα μεγέθη θα ισούνται με την πρόσθεση των μεγεθών των 2 άλλων αντικείμενων δηλαδή θα γράψουμε την εντολή KoutiC=KoutiA+KoutiB.Στο τέλος βάζουμε ακόμα μια cout που θα μας τυπώνει τον όγκο του κουτιού KoutiC.

Τα αποτελέσματα:

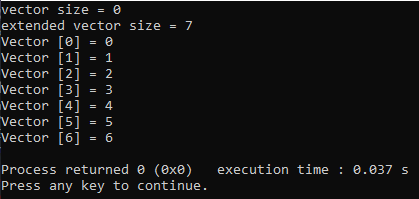


**Άσκηση 2**

**Ερώτημα 1:**

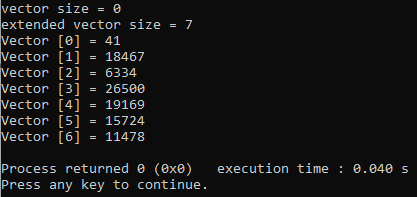
Πρώτα πολλά εκτελούμε τον κώδικα σύμφωνα με αυτά που μας ζητάει η άσκηση και συμπεραίνουμε τα εξής: Η βιβλιοθήκη vector είναι μια καθιερωμένη βιβλιοθήκη προτύπων(STL) η οποία δημιουργεί μια δομή δεδομένων με τις ιδιότητες ενός δυναμικού πίνακα. Με την συνάρτηση vec.size() μας επιστρέφει τον αριθμό των στοιχείων που υπάρχουν στο διάνυσμα. Έτσι στην αρχή η παραπάνω συνάρτηση που την καλούμε μας επιστρέφει 0.Μετα με μια for εισάγουμε τα στοιχεία μας μέσα στο διάνυσμα μέσω της συνάρτησης push.back() οπού εισάγει ένα στοιχείο στο τέλος της δομής και εκχωρεί αυτόματα μνήμη αν αυτό είναι αναγκαίο. Οπότε μετά το for ξανά καλούμε την συνάρτηση vec.size() η οποία περιμένουμε να μας επιστρέψει άλλο μέγεθος αφού δώσαμε στοιχεία στο διάνυσμα μας μέσω της for.Τέλος πάλι με ένα for εκτυπώνουμε τα περιεχόμενα της κάθε θέσης του διανύσματος.

Τα αποτελέσματά:



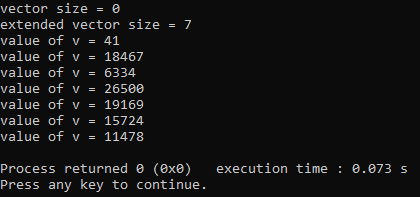
Tώρα για να παίρνει τυχαίους αριθμούς το διάνυσμα μας απλά στην συνάρτηση push.back() αντί να βάλουμε για όρισμα το counter της for θα βάλουμε την συνάρτηση rand() δηλαδή θα έχουμε την εντολή vec.push\_back(rand()).

Τα αποτελέσματα:



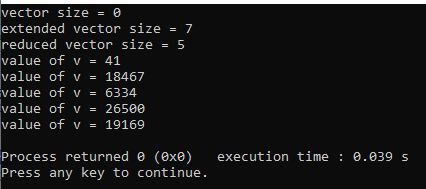
Στην συνέχεια κάνοντας τις αλλαγές που μας ζητάει η άσκηση συμπεραίνουμε τα εξής: Με την χρήση του iterator ορίζουμε έναν δείκτη v τύπου int που δείχνει στο πρώτο στοιχείο του διανύσματος(begin()) και μετά με μια while όσο ο iterator είναι διάφορος του τέλους του τελευταίου στοιχείου του διανύσματος μας τυπώνει τα περιεχόμενα της θέσης στην οποία βρίσκεται ο δείκτης v.

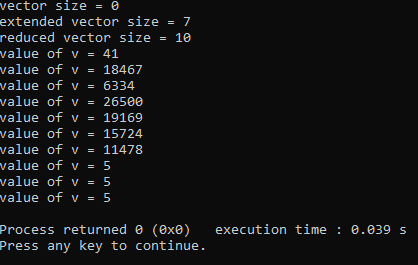
Τα αποτελέσματα:



Κάνοντας πάλι τώρα τις αλλαγές που μας ζητάει η άσκηση έχουμε τα εξής: Με την resize() αλλάζουμε το μέγεθος του διανύσματος οπότε το καλούμε με το τρόπο αυτό:vec.resize(5) και μετά με την χρήση πάλι της size() τυπώνουμε το νέο μέγεθος του διανύσματος. Τώρα αν την καλέσουμε με αυτό τον τρόπο(resize(10,5)) γίνεται το εξής: Το πρώτο όρισμα είναι αυτό που θα αλλάξει το μέγεθος το διανύσματος , το δεύτερο όρισμα της resize αυτό που μας λέει είναι ότι αν δημιουργηθούν παραπάνω θέσεις στο διάνυσμα από πριν τότε θα πάρουν την τιμή του ορίσματος αυτού:

Τα αποτέλεσμα και για τις 2 περιπτώσεις:

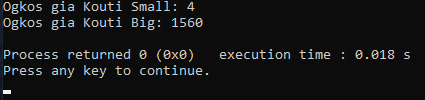




**Ερώτημα 2:**

Σύμφωνα με τα ζητούμενα της άσκησης παρατηρούμε ότι οι setIpsos και getIpsos δεν δημιουργούν κάποιο πρόβλημα με τον τρόπο που έχουν δημιουργηθεί στον κώδικα και αυτό γιατί τις έχουμε ορίσει μέσα στην κλάση και δημιουργούμε το σώμα τους καλώντας την κλάση με την μέθοδο έξω από το ορισμό της κλάσης.

Τα αποτελέσματα:

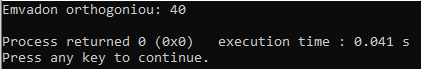


Τώρα αν κάνουμε τις αλλαγές που μας ζητάει η άσκηση μας εμφανίζει κάποια errors. Αυτό συμβαίνει διότι από την στιγμή που έχουμε δηλώσει ως private τις μεταβλητές της κλάσης δεν μπορούμε να τις καλέσουμε μέσα στην main αφού είναι προσπελάσιμες μόνο από την κλάση Kouti. Οπότε αυτό που κάνουμε είναι να ορίσουμε κάποιες συναρτήσεις οι οποίες θα είναι public οπότε θα είναι προσπελάσιμες και από την main μέσα από τις οποίες θα θέτουμε τις τιμές των μεταβλητών του κουτιού αλλά και να μας επιστρέφονται αυτές οι τιμές και έτσι θα πραγματοποιήσουμε τον σκοπό της άσκησης. Ο κωδικός με τις αλλαγές θα παρατεθεί στο παράρτημα Α. Τα αποτελέσματα παραμένουν τα ίδια.

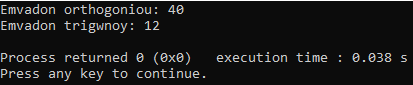
**Ερώτημα 3:**

Βλέπουμε ότι αυτό που κάνει ο κώδικας είναι το εξής. Έχουμε ορίσει μια κλάση Polygon με protected 2 μεταβλητές οπότε αυτές είναι προσπελάσιμες από τις υποκλάσεις της Polygon αλλά και από την ίδια την κλάση. Επίσης έχει μια μέθοδο public από την οποία μπορούμε και δίνουμε τιμές στις μεταβλητές της Polygon. Μετά έχουμε μια υποκλάση της κλάσης Polygon η οποία έχει μια public μέθοδο που μας επιστρέφει τον εμβαδόν ενός ορθογωνίου.

Τα αποτελέσματα:



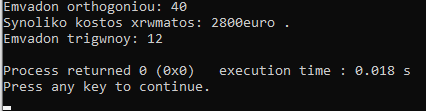
Φτιάχνουμε την κλάση Triangle που κληρονομεί από την Polygon και δημιουργούμε μια μέθοδο area() που θα μας επιστρέφει το εμβαδόν του τρίγωνου. Τέλος κάνουμε και απαραίτητες αλλαγές στην main και έχουμε τα εξης αποτελέσματα:



**Ερώτημα 4:**

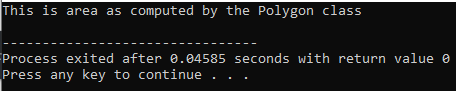
Κάνουμε τις αλλαγές που μας ζητάει η άσκηση και συμπεραίνουμε τα εξής: Η κλάση Rectangle κληρονομεί και από την κλάση Polygon αλλά και από την PaintCost κάτι το οποίο δεν γίνεται στην java γιατί από όσο ξέρουμε μια subclass κληρονομεί μόνο από μια υπερκλαση.

Τα αποτελέσματα:



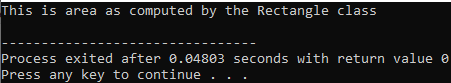
**Ερώτημα 5:**

Εκτελούμε τον κώδικα σύμφωνα με την άσκηση και παρατηρούμε ότι παίρνουμε κάποια μη αναμενόμενα αποτελέσματα δηλαδή ενώ εμείς περιμέναμε να πάρουμε το εμβαδόν του αντικειμένου rec παίρνουμε κάτι τελείως διαφορετικό που φαίνεται παρακάτω:



Αυτό συμβαίνει για το λόγο ότι έχουμε ένα αντικείμενο polygon δείκτη το οποίο δείχνει στο rec και στην συνάρτηση area() όμως επειδή έχουν κοινό όνομα με την συνάρτηση εμβαδού στην Rectangle και την Polygon ο compiler κάνει κλήση της συνάρτησης της Polygon.Αυτό λέγεται early binding(πολυμορφισμός χρόνου μεταγλώττισης).Όπως υποδηλώνει το όνομα, o μεταγλωττιστής συσχετίζει απευθείας μια διεύθυνση στην κλήση συνάρτησης. Αντικαθιστά την κλήση με μια οδηγία γλώσσας μηχανής που λέει στο κεντρικό πλαίσιο να μεταβεί στη διεύθυνση της λειτουργίας.

Τώρα βάζοντας την εντολή virtual(εικονική) στην συνάρτηση area() της Polygon παρατηρούμε ότι έχουμε διαφορετικά αποτελέσματα τα οποία φαίνονται παρακάτω:



Με αυτό τον τρόπο υλοποιήσαμε έναν μηχανισμό late binding δηλαδή ο μεταγλώττισης προσθέτει κώδικα που προσδιορίζει το είδος του αντικειμένου κατά το χρόνο εκτέλεσης και στη συνέχεια ταιριάζει με την κλήση με τον σωστό ορισμό λειτουργίας .Αυτό έγινε δηλώνοντας τη area() ως εικονική συνάρτηση. Τώρα οι εικονικές συναρτήσεις μας επιτρέπουν να δημιουργήσουμε μια λίστα με δείκτες κλάσης βάσης και μεθόδους κλήσεων οποιασδήποτε από τις παραγόμενες τάξεις χωρίς καν να γνωρίσουμε το είδος του παραγομένου αντικειμένου κλάσης.

**C++ SET 3:**

**Άσκηση 1**

**Ερώτημα 1:**

Κανουμε τις αλλαγες που μας ζηταει(ολοκληρος ο κωδικας θα παρατεθει στο τελος).

**Ερώτημα 2:**

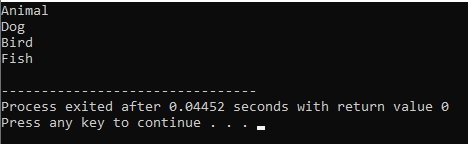
Έχοντας δημιουργήσει ως protected τον κατασκευαστή της κλάσης Animal και οι τρεις κολάσεις Bird, Fish, Dog κληρονομούν από την Animal τότε ως protected ο κατασκευαστής Αnimal με τα ορίσματα μπορεί να κληρονομηθεί και από τις υπόλοιπες υποκλίσεις και να αρχικοποιήσουν τις ανάλογες τιμές τους. Τώρα αν κάνουμε τον κατασκευαστή Animal private θα έχουμε πρόβλημαδιότι τότε ο κατασκευαστής θα είναι προσπελάσιμος μόνο από την ιδιά του την κλάση και όχι από τις υποκλάσεις του.

**Ερώτημα 3:**

Δημιουργούμε την main όπως μας ζητάει το ερώτημα και μεταγλωττίζουμε.

**Ερώτημα 4:**

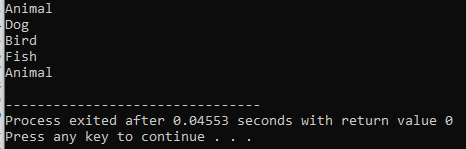
Εκτελούμε τον κώδικα και παίρνουμε τα εξής αποτελέσματα:



Όπως βλέπουμε η getClass() μας έχει επιστρέψει κάθε φορά και διαφορετική κλάση. Αυτό έγινε διότι κάθε αντικείμενο είναι διαφορετικής κλάσης και καλεί την getClass της δική της κλάσης δηλαδή το αντικείμενο a που είναι τύπου Animal θα καλέσει την getClass() της Animal ενώ το d που είναι τύπου Dog θα καλέσει την getClass() της Dog.Το ίδιο ισχύει και για τα αντικείμενα b, f.

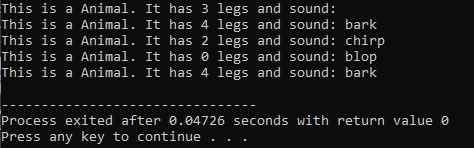
**Ερώτημα 5:**

Κάνοντας τις αλλαγές που μας ζητάει το ερώτημα παρατηρούμε ότι αντί να μας επιστρέψει την getClass της κλάσης Dog μας επιστρέφει την getClass της Animal.Αυτό έγινε διότι το αντικείμενο a όταν το εξισώνουμε με το αντικείμενο d αυτό που γίνεται είναι τα ορίσματα του d να μεταφερθούν στο a και όχι να αλλάξει η κλάση στην οποία ανήκει το αντικείμενο a. Οπότε και γι’ αυτό πήραμε τα παρακάτω αποτελέσματα:



**Ερώτημα 6:**

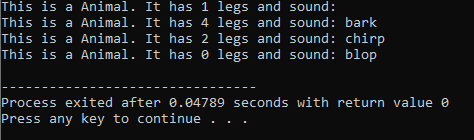
Κάνοντας τις απαραίτητες αλλαγές και εκτελώντας τον κώδικα παίρνουμε τα παρακάτω αποτελέσματα:



Παρατηρούμε πρώτα πολλά ότι το αντικείμενο a πηρέ κάποια ορίσματα δηλαδή δεν περιμέναμε να μας επιστρέψει πόδια και ήχο. Αυτό έγινε διότι από το public κατασκευαστή Animal όταν καλούμε την printInfo απλά δίνει κάποιες default τιμές στο αντικείμενο μας. Μετά ήταν αναμενόμενο για κάθε αντικείμενο η printfInfo() να μας επιστρέψει τις ανάλογες τιμές των legs και sound.Από την άλλη βέβαια περιμέναμε για κάθε αντικείμενο να πάρουμε και την αντίστοιχη κλάση αντί αυτού μας επιστρέφει μόνο την κλάση Animal.Αυτό γίνεται διότι η printInfo() είναι ορισμένη μέσα στην Animal οπότε η τιμή που θα επιστρέψει για την getClass θα είναι αυτή που είναι ορισμένη μέσα στην Animal.Τώρα στην τελευταία γραμμή των αποτελεσμάτων αυτό που παρατηρούμε είναι ότι το αντικείμενο a πήρε τις τιμές των ορισμάτων του αντικείμενου d και αυτό χάρης στο τελεστή < = > και γι’ αυτό παίρνουμε τα παραπάνω αποτελέσματα.

**Ερώτημα 7:**

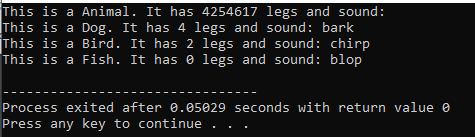
Κάνουμε τις αλλαγές που μας ζητάει η άσκηση και παίρνουμε τα εξής αποτελέσματα:



Όπως βλέπουμε πάλι παίρνουμε κάποια αναμενόμενα αλλά και κάποια μη αναμενόμενα αποτελέσματα. H πρώτη γραμμή των αποτελεσμάτων είναι συνέπεια του ότι έχουμε μια μεταβλητή a προς αντικείμενο τύπου Animal οπού του έχουμε αναθέσει να δείχνει στο αντικείμενο aa τύπου Animal το οποίο από default παίρνει στα τις τιμές που βλέπουμε. Τώρα για τις υπόλοιπες 3 γραμμές ισχύει το ίδιο δηλαδή κάθε φορά ο δείκτης δείχνει απλά σε άλλο αντικείμενο άλλης κλάσης στα οποία έχουμε εμείς αρχικοποιήσει τις τιμές legs και sounds στον κατασκευαστή της κάθε κλάσης. Τώρα ο λόγος που δεν παίρνουμε την αντίστοιχη κλάση για τα αντικείμενα είναι ότι έχουμε κάνει early binding οπότε ο compiler παίρνει την τιμή της διεύθυνσης που ανήκει στην getClass της κλάσης Animal.

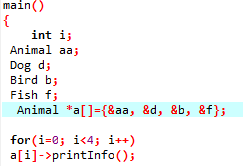
**Ερώτημα 8:**

Η αλλαγή που πρέπει να γίνει γενικά είναι να υλοποιήσουμε τον μηχανισμό late binding.Αυτό θα το επιτύχουμε δηλώντας την getClass() στην Animal ως virtual(εικονική). Κάνοντας την παραπάνω αλλαγή λύνουμε το πρόβλημα μας και περνούμε τα παρακάτω αποτελέσματα:

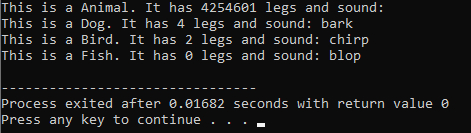


**Ερώτημα 9:**

Κάνουμε τις αλλαγές που μας ζητάει δημιουργώντας έναν πίνακα με δείκτες τύπου Animal στον οποίο αναθέτουμε τα αντικείμενα της κάθε κλάσης. Έπειτα με for διατρέχουμε τον πίνακα η οποία για κάθε στοιχείο του πίνακα καλεί την printInfo().

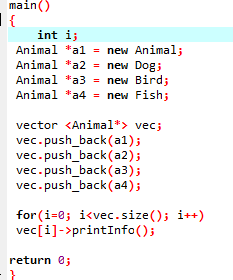


Τα αποτελέσματα:



**Ερώτημα 10:**

Χρησιμοποιούμε τα vectors και ο κώδικας που συμπληρώσαμε είναι ο παρακάτω:



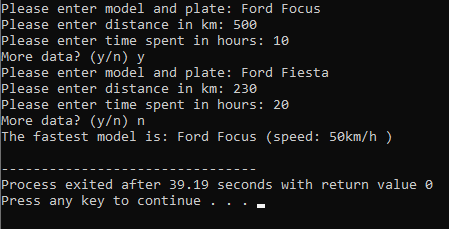
Γενικά τα vectors είναι πιο βολικά από τους πίνακες διότι έχουμε περισσότερες δυνατότητες όπως δεν χρειάζεται να ξέρουμε το μέγεθος τους ενώ για τους πίνακες πρέπει έχουμε άφθονες συναρτήσεις για την προσπέλαση/εισαγωγή/διαγραφή στοιχείων ενός vector ενώ στους πίνακες όχι και επίσης τα vectors μπορούν δεχθούν μια δυναμική χρήση στους κώδικες ενώ οι πίνακες όχι. Βέβαια στην συγκεκριμένη περίπτωση είναι προτιμότεροι οι πίνακες διότι ξέρουμε το μέγεθος του πίνακα και επίσης γλιτώνουμε γραμμές κώδικα.

**Άσκηση 2**

**Ερώτημα 1:**

Η λειτουργία του κώδικα αυτή είναι η εξής. Ζητάει από τον χρήστη να του δώσει ένα μοντέλο αυτοκίνητου, τα χιλιόμετρα, και τον χρόνο τον οποίο χρειάζεται να φτάσει αυτά τα χιλιόμετρα. Στην συνέχεια σε ρωτάει αν θέλουμε να βάλουμε και αλλά μοντέλα με τα χαρακτηριστικά τους. Αν δώσουμε την απάντηση y επαναλαμβάνεται η διαδικασία και μέσα σε ένα if αν το πηλίκο των χιλιομέτρων προς τον χρόνο είναι μεγαλύτερο από το προηγούμενο πηλίκο τότε παίρνουμε ως τιμές αυτές που κάνουν το πηλίκο μεγαλύτερο. Τώρα αν δώσουμε ως απάντηση n τότε βγαίνουμε από τον βρόγχο while και μας τυπώνει τα αποτελέσματα για το ποιο είναι το πιο γρήγορο αμάξι.

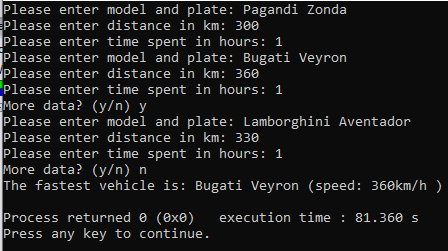
Τα αποτελέσματα :



**Ερώτημα 2:**

Για να αποκτήσει το πρόγραμμα έναν αντικειμενοστραφή ύφος θα δημιουργήσουμε μια κλάση Vehicle που θα παίρνει ως private τα 3 ορίσματα που το χαρακτηρίζουν(μοντέλο, χιλιόμετρα, ώρες).Στην συνέχεια στο πεδίο public θα έχει την μέθοδο input η οποία θα είναι αυτή στην οποία θα δίνω τιμές για τα ορίσματα. Έπειτα θα έχουμε μια bool τύπου μέθοδο που θα συγκρίνει 2 αντικείμενα της κλάσης και ικανοποιείται η συνθήκη της if τότε θα μας επιστρέφει true αλλιώς false.Τέλος έχουμε την μέθοδο show που μας τυπώνει τα αποτελέσματα για το πιο γρήγορο αμάξι.

Τα αποτελέσματα:



**Ερώτημα 3:**

Δηλώνοντας ως friend μια μέθοδο αυτό που καταφέρνουμε είναι να έχουμε μια ειδική επιχορήγηση για πρόσβαση σε ιδιωτικά και προστατευόμενα μέλη δηλαδή να έχουμε μια μέθοδο άλλης τάξης ή μια global function. Τα αποτελέσματα είναι ιδιά με τα παραπάνω.

**Ερώτημα 4:**

Οι αλλαγές του κώδικα θα παρατεθούν στο παράρτημα στο τέλος της αναφοράς. Τα αποτελέσματα είναι ιδιά με αυτά του 2ου ερωτήματος.

**ΠΑΡΑΡΤΗΜΑ-Α-ΚΩΔΙΚΕΣ**

**ΤΟΥ ΦΥΛΛΑΔΙΟΥ**

#include <iostream>

using namespace std;

int main()

{

cout << "Enter a number: " << endl;

int value;

cin >> value;

if(value < 10)

{

cout << "This value is too small";

}

else if(value > 10)

{

cout << "This is a big enough number!";

}

return 0;

}

#include <cstdlib>

#include <iostream>

using namespace std;

int main(int argc, char \*argv[])

{

int val1;

int val2;

int val3;

cout<<"Please enter your 3 numbers:";

cin>>val1>>val2>>val3;

cout<<val3<<" "<<val2<<" "<<val1<<"\n";

system("PAUSE");

return EXIT\_SUCCESS;

}

#include <iostream>

using namespace std;

class Kouti

{

private:

double length;

double breadth;

double height;

public:

Kouti operator+(const Kouti& b)

{

Kouti kouti;

kouti.length = this->length + b.length;

kouti.breadth = this->breadth + b.breadth;

kouti.height = this->height + b.height;

return kouti;

}

setMikos(double m)

{

length=m;

}

setIpsos(double y)

{

height=y;

}

setPlatos(double p)

{

breadth=p;

}

double calculateOgkos()

{

return length\*height\*breadth;

}

};

main()

{

Kouti KoutiA;

Kouti KoutiB;

Kouti KoutiC;

KoutiA.setMikos(2.0);

KoutiA.setPlatos(3.2);

KoutiA.setIpsos(6.0);

KoutiB.setMikos(2.5);

KoutiB.setPlatos(4.0);

KoutiB.setIpsos(5.0);

cout<<"O ogkos toy antikeimenoy Kouti A einai: "<<KoutiA.calculateOgkos()<<endl;

cout<<"O ogkos toy antikeimenoy Kouti B einai: "<<KoutiB.calculateOgkos()<<endl;

KoutiC=KoutiA+KoutiB;

cout<<"O ogkos toy antikeimenoy Kouti B einai: "<<KoutiC.calculateOgkos()<<endl;

}

#include <iostream>

#include <vector>

#include <stdlib.h>

using namespace std;

int main()

{

vector <int> vec;

int i;

cout << "vector size = " << vec.size() << endl;

for(i = 0; i < 7; i++){

vec.push\_back(rand());

}

cout << "extended vector size = " << vec.size() << endl;

vec.resize(10,5);

cout << "reduced vector size = " << vec.size() << endl;

vector<int>::iterator v = vec.begin();

while( v != vec.end()) {

cout << "value of v = " << \*v << endl;

v++;

}

return 0;

}

#include <iostream>

using namespace std;

class Kouti

{

private:

double mikos = 2.0;

double platos;

double ipsos;

public:

void setMikos( double mik ) {

mikos = mik;

};

void setPlatos( double pla ) {

platos = pla;

};

double getPlatos (void){

return platos;

};

double getMikos (void) {

return mikos;

};

void setIpsos( double ips );

double getIpsos( void );

};

double Kouti::getIpsos( void )

{

return ipsos;

}

void Kouti::setIpsos ( double ips )

{

ipsos = ips;

}

int main( )

{

Kouti Small;

Kouti Big;

double ogkos=0.0;

Small.setPlatos(1.0);

Small.setIpsos(2.0);

ogkos = Small.getPlatos() \* Small.getIpsos() \* Small.getMikos();

cout << "Ogkos gia Kouti Small: " << ogkos <<endl;

Big.setMikos(12.0);

Big.setPlatos(13.0);

Big.setIpsos(10.0);

ogkos = Big.getPlatos() \* Big.getIpsos() \* Big.getMikos();

cout << "Ogkos gia Kouti Big: " << ogkos <<endl;

return 0;

}

#include <iostream>

using namespace std;

class Polygon {

protected:

int width, height;

public:

void set\_values (int a, int b)

{ width=a; height=b;}

};

class PaintCost

{

public:

int getCost(int area)

{

return area \* 70;

}

};

class Rectangle: public Polygon, public PaintCost{

public:

int area ()

{ return width \* height; }

};

class Triangle: public Polygon{

public:

int area()

{

return width\*height/2;

}

};

int main () {

Rectangle rect;

Triangle trian;

rect.set\_values(5,8);

trian.set\_values(6,4);

cout << "Emvadon orthogoniou: " << rect.area() << '\n';

cout << "Synoliko kostos xrwmatos: " << rect.getCost(rect.area()) <<" euro ."<< '\n';

cout << "Emvadon trigwnoy: " << trian.area() << '\n';

return 0;

}

#include <iostream>

using namespace std;

class Polygon {

protected:

int width, height;

public:

Polygon( int a=0, int b=0)

{

width = a;

height = b;

}

virtual int area()

{

cout << "This is area as computed by the Polygon class" <<endl;

return 0;

}

};

class Rectangle: public Polygon{

public:

Rectangle( int a=0, int b=0):Polygon(a, b) { }

int area ()

{

cout << "This is area as computed by the Rectangle class" <<endl;

return (width \* height);

}

};

int main( )

{

Polygon \*polygon;

Rectangle rec(10,7);

polygon = &rec;

polygon->area();

return 0;

}

**ΠΑΡΑΡΤΗΜΑ-Β-ΚΩΔΙΚΕΣ**

**ΑΣΚΗΣΕΩΝ ΤΟΥ ΣΕΤ**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Animal //οριζουμε την κλαση animal

{

private://τα ορισματα αυτα ειναι προσπελασιμα μονο απο την κλαση Animal

int legs;

string sound;

protected://ο κατασκευαστης που βρισκεται στο πεδιο protected ειναι προσπελασιμο μονο απο την Animal και τις υποκλασεις της

Animal(int l, string s)//ο κατασκευαστης της κλασης

{

legs=l;//η legs παιρνει την τιμη που θα δωσουμε στο οριμσα l σε μια αρχικοποιησει ενος αντικειμενου

sound=s;//η sound παιρνει την τιμη που θα δωσουμε στο οριμσα s σε μια αρχικοποιησει ενος αντικειμενου

}

public://αυτο το πεδιο ειναι προσπελασιμο απο οποιαδηποτε μεθοδο και κλαση

Animal()//ενας κατασκευαστης τον οποιο μπορουμε να καλεσουμε σε οποιαδηποτε κλαση η μεθοδο χωρις ορισματα βεβαια

{

}

virtual string getClass()//εικονικη string μεθοδος που μας επιστρεφει την κλαση του αντικειμενου.Ετσι επιτυγχανουμε μηχανισμο late binding

{

return "Animal";

}

void printInfo()//Μεθοδο που μου τυπωνει το νουμερο των ποδιων και τον ηχο του καθε αντικειμενου

{

cout <<"This is a "<<getClass()<<". It has "<<legs<<" legs and sound: "<<sound<<"\n";

}

int getLegs()//επιστρεφει τον αριθμο ποδιων

{

return legs;

}

string getSound()//επιστρεφει τον ηχο καθε ζωου

{

return sound;

}

};

//κλαση Bird που ειναι υποκλαση της Animal

class Bird:public Animal{

public:

//Ο κατασκευαστης της Bird καλει τον κατασκευαστη της animal ωστε να δωσει τα καταλληλα ορισματα

Bird(): Animal(2, "chirp")

{

}

string getClass()//επιστρεφει την κλαση του ζωου

{

return "Bird";

}

};

//κλαση Dog που ειναι υποκλαση της Animal

class Dog:public Animal{

public:

//Ο κατασκευαστης της Dog καλει τον κατασκευαστη της Animal ωστε να δωσει τα καταλληλα ορισματα

Dog(): Animal(4, "bark")

{

}

string getClass()//επιστρεφει την κλαση του ζωου

{

return "Dog";

}

};

//κλαση Fish που ειναι υποκλαση της Animal

class Fish:public Animal{

public:

//Ο κατασκευαστης της Fish καλει τον κατασκευαστη της Animal ωστε να δωσει τα καταλληλα ορισματα

Fish(): Animal(0, "blop")

{

}

string getClass()//επιστρεφει την κλαση του ζωου

{

return "Fish";

}

};

main()

{

int i;

Animal \*a1 = new Animal;//δημιουργουμε εναν δεικτη τυπου Animal στον οποιο αναθετουμε ενα αντικειμενο τυπου Animal

Animal \*a2 = new Dog;//δημιουργουμε εναν δεικτη τυπου Animal στον οποιο αναθετουμε ενα αντικειμενο τυπου Dog

Animal \*a3 = new Bird;//δημιουργουμε εναν δεικτη τυπου Animal στον οποιο αναθετουμε ενα αντικειμενο τυπου Bird

Animal \*a4 = new Fish;//δημιουργουμε εναν δεικτη τυπου Animal στον οποιο αναθετουμε ενα αντικειμενο τυπου Fish

vector <Animal\*> vec;//δημιουργουμε ενα διανυσμα τυπου Animal που θα περιεχει του δεικτες τυπου Animal

vec.push\_back(a1);//εισερχεται το 1ο στοιχειο μεσα στο διανυσμα

vec.push\_back(a2);//εισερχεται το 2ο στοιχειο μεσα στο διανυσμα

vec.push\_back(a3);//εισερχεται το 3ο στοιχειο μεσα στο διανυσμα

vec.push\_back(a4);//εισερχεται το 4ο στοιχειο μεσα στο διανυσμα

//Με μια for Διατρεχουμε το διανυσμα vec ωστε να προσπελασουμε τα στοιχεια του

for(i=0; i<vec.size(); i++)

vec[i]->printInfo();//για καθε στοιχειο που παιρνουμε του λεμε να καλεσει την printfInfo()

return 0;

}

#include <iostream>

#include <string>

using namespace std;

class Vehicle{//δημιουργουμε την κλαση Vehicle

//δημιουργουμε το πεδιο private με τις παρακατω μεταβλητες οι οποιες ειναι προσπελασιμες μονο απο την Vehicle

private:

string model="";

int km=1;

double hours=1.0;

//Αυτο το πεδιο ειναι προσπελασιμο απο οποιαδηποτε κλαση ή μεθοδο

public:

void input()//μεθοδο απο την οποια ο χρηστης δινει τιμες στα ορισματα του αντικειμενου τυπου Vehicle

{

string remainder;

cout << "Please enter model and plate: ";

getline(cin, model);

cout << "Please enter distance in km: ";

cin >> km;

cout << "Please enter time spent in hours: ";

cin >> hours;

getline(cin, remainder);

}

bool isFasterThan(Vehicle a)//Μια boolean μεθοδο με ενα ορισμα τυπου Vehicle που αν ισχυει η συνθηκη της if επιστρεφει true αλλιως false

{

if(a.km/a.hours < km/hours)

{

return true;

}

else

{

return false;

}

}

void const show()//μια σταθερα συναρτηση που μας τυπωνει τα αποτελεσματα για το πιο γρηγορο αντικειμενο.

{

cout<<model<< " (speed: " << km/hours << "km/h )\n";

}

};

int main()

{

Vehicle v1;//δημιουργουμε ενα αντικειμενο τυπου Vehicle

v1.input();//καλουμε την input επι το αντικειμενο v1

bool more = true;

while (more)//μια συνθηκη που οσο το more ειναι true διατρεχουμε τον κωδικα μεσα σε αυτην

{

Vehicle v2;//δημιουργουμε ενα αντικειμενο τυπου Vehicle

v2.input();//καλουμε την input επι το αντικειμενο v2

if (v2.isFasterThan(v1)) v1 = v2;//συγκρινουμε τα 2 αντικειμενα που δημιουργησαμε για να δουμε ποιο ειναι πιο γρηγορο

cout << "More data? (y/n) ";//Το προγραμμα μας ρωταει αν θελουμε να δωσουμε και αλλα δεδομενα

string answer;

getline(cin, answer);//παιρνει την γραμμη της απαντησης μας

if (answer != "y") more = false;//Αν ειναι y τοτε συνεχιζουμε να δινουμε δεδομενα αλλιως αν ειναι n βγαινουμε απο την συνθηκη while

}

cout << "The fastest vehicle is: ";

v1.show();//καλουμε την show επι του v1 το οποιο εχει ως ορισματα αυτα του πιο γρηγορου αυτοκινητου

return 0;

}

#include <iostream>

#include <string>

using namespace std;

//Για περισσοτερα σχολια στα πεδια που δεν αναφερονται ανατρεξτε τον αμεσως πιο πανω κωδικα

class Vehicle{

private:

string model;

int km;

double hours;

public:

bool operator > (const Vehicle& b)//Υπερφορτωνουμε τον τελεστη > για να μπορουμε να τον χρησιμοποιησουμε να συγκρινουμε 2 αντικειμενα.Δηλαδη θα κανει την δουλεια της isFasterThan.Την εχουμε δηλωσει ως Boolean γιατι μας επιστρεφει Boolean τιμες

{

if(this->km/this->hours > b.km/b.hours )//βαζουμε βελος με this για να παιρνουμε τις τορινες τιμες του κάθε αντικειμενου

{

return true;

}

else

{

return false;

}

}

void input()

{

string remainder;

cout << "Please enter model and plate: ";

getline(cin, model);

cout << "Please enter distance in km: ";

cin >> km;

cout << "Please enter time spent in hours: ";

cin >> hours;

getline(cin, remainder);

}

friend void show(Vehicle a);//Με αυτό τον τροπο καταφερνουμε να καλουμε την show με ειδικο τροπο αλλα και να την γραψουμε ως μια global μεθοδο

};

void show(Vehicle a)//Εδώ φτιαχνουμε το σωμα της show

{

cout<<a.model<< " (speed: " << a.km/a.hours << "km/h )\n";

}

int main()

{

Vehicle v1;

v1.input();

bool more = true;

while (more)

{

Vehicle v2;

v2.input();

if(v2>v1) v1 = v2;//Συγκρινουμε τα δυο αντικειμενα και αν το v2 είναι πιο γρηγορο από το v1 τοτε το v1 παιρνει τις τιμες του v2

cout << "More data? (y/n) ";

string answer;

getline(cin, answer);

if (answer != "y") more = false;

}

cout << "The fastest vehicle is: ";

show(v1);//Εδώ βλεπουμε ότι καλουμε την μεθοδο show με έναν διαφορετικο τροπο.

return 0;

}