Εργαστηριακές Ασκήσεις C++
΄Ετος: 2024-2025

Βασιλόπουλος Ιωάννης
Βογιατζάκη Ελένη
Καλογερόπουλος Νικήτας - Ρήγας
Κουτσομητρόπουλος Δημήτριος
Μακρής Χρήστος
Χατζηλυγερούδης Ιωάννης

Περιεχόμενα

1 Άσκηση 1

Σκοπός της άσκησης αυτής είναι:

α'. να δημιουργήσετε ένα νέο αρχείο C++,

β'. να εισάγετε κάποιον κώδικα, να τον μεταγλωττίσετε και να δείτε να τα αποτελέσματα με τηνχρήση του περιβάλλοντος Code::Blocks,

γ'. να εξοικειωθείτε με την είσοδο και έξοδο δεδομένων από και προς την οθόνη,

δ'. να εξοικειωθείτε με δομές επανάληψης όπως το for loop,

ε'. να πειραματιστείτε με την δημιουργία κλάσεων και την υπερφόρτωση τελεστών.

1) Να ανοίξετε το εργαλείο Code::Blocks, αφού δημιουργήσετε ένα νέο project (File→ New → Project…) να επιλέξετε την δημιουργία άδειου project (Empty Project) και να το αποθηκεύσετεστον επιθυμητό κατάλογο με το όνομα ''askisi1''.

Σημείωση 1: Προσέξτε να επιλέξετε τον κατάλληλο μεταγλωττιστή στο παράθυρο και για την διευκόλυνση σας να επιλέξετε την έκδοση mingw που περιλαμβάνει τους κατάλληλους μεταγλωττιστές και έτσι δε θα χρειαστεί να εγκαταστήσετε κάτι επιπρόσθετα.

Δημιουργήστε ένα νέο αρχείο, (*ενδεικτικά :File→ New → Empty File*) και αποθηκεύεστε τομε όνομα q1.cpp και να δοκιμάσετε τον παρακάτω κώδικα με εισόδους 5 και 15.

Σημείωση 2: Προσοχή στην κατάληξη του αρχείου διότι μερικές εκδόσεις του Code::Blocks αν δεν καθοριστεί το επίθεμά του αρχείου επιλέγεται αυτόματα το .c.

```
#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";
    }
    return 0;
}
```

Να αλλάξετε το παραπάνω πρόγραμμα ώστε εάν η είσοδος είναι μεγαλύτερη του 10 να εμφανίζεται μήνυμα "This is a big enough number"

2. Αφαιρέστε το αρχείο q1.cpp (*ενδεικτικά: Δεξί κλικ στο αρχείο → Remove file from project*) διότι δεν γίνεται να υπάρχουν πολλαπλές main συναρτήσεις στο ίδιο project και δημιουργήστε

ένα καινούριο αρχείο με όνομα q2.cpp με τον εξής κώδικα:

```
#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<<val1<<"\n"<<val2<<"\n"<<val3<<"\n";

return 0;
}
```

Να κάνετε τις απαραίτητες αλλαγές ώστε πριν τερματίσει να εμφανίζει τους αριθμούς που δόθηκαν ως είσοδο με ανάστροφη σειρά.

3. Μελετήστε τον παρακάτω κώδικα και εξηγείστε τι κάνει:

```
#include<iostream>
using namespace std;
int main(){
    cout<<"First set of numbers:"<<endl;
    int first[2][2], second[2][2];
    int i,j;
    for(i=0; i<2; i++){
        cout<<"Enter two integers: "<<i+1<<endl;
        for(j=0; j<2; j++){
            cin>>first[i][j];
        }
    }
    cout<<"\n\nSecond set of numbers:"<<endl;
    for(i=0; i<2; i++){
        cout<<"Enter two more integers: "<<i+1<<endl;
        for(j=0; j<2; j++){
            cin>>second[i][j];
        }
    }
    for(i=0; i<2; i++){
        for(j=0; j<2; j++){
            first[i][j]= first[i][j]+second[i][j];
        }
    }
    return 0;
}
```

Προσθέστε κατάλληλο for loop που να εμφανίζει το αποτέλεσμα στην οθόνη. Στη συνέχεια χρησιμοποιήστε 2 σταθερές που να καθορίζουν αντίστοιχα τις αντίστοιχες κοινές διαστάσεις των δύο πινάκων και κάντε τις κατάλληλες αλλαγές στα σημεία του κώδικα. Τέλος, αλλάξτε τον κώδικα

ώστε οι διαστάσεις των πινάκων να δίνονται από τον χρήστη μετά από κατάλληλο μήνυμα που θα εμφανίζεται στην οθόνη.

4. Α) Χρησιμοποιώντας τον παρακάτω ορισμό μίας κλάσης Kouti σε C++ γράψτε τον κώδικα της main() όπου:

```
class Kouti
{
public:
    double length;
    double breadth;
    double height;
};
```

i Θα δημιουργεί δύο αντικείμενα τύπου Kouti, τα KoutiA και KoutiB,

ii μία μεταβλητή τύπου double, ogkos, που θα εκφράζει τον όγκο και θα είναι αρχικοποιείται με 0.0,

iii Θα αρχικοποιεί τα αντικείμενα KoutiA και KoutiB με τις τιμές (2.0, 3.2, 6.0) και (2.5, 4.0, 5.0) αντίστοιχα,

iv Θα υπολογίζει και θα εμφανίζει τους όγκους των αντικειμένων στην μεταβλητή ogkos μαζί με το κατάλληλο μήνυμα.

Β) Μετατρέψτε τις ιδιότητες της κλάσης σε private και εφοδιάστε την κλάση με την κατάλληλη public μέθοδο calculateOgkos() που θα υπολογίζει τον όγκο κάθε αντικειμένου. Κάντε στην main() τις απαιτούμενες αλλαγές ώστε να λειτουργεί όπως στο παραπάνω ερώτημα.

5. Προσθέστε στην παραπάνω κλάση Kouti τον εξής κώδικα, ο οποίος πραγματοποιεί υπερφόρτωση του τελεστή +.

```
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;
    }
```

Να προσθέσετε στην Kouti μεθόδους setMikos(), setPlatos(), και setYpsos() και να τροποποιήσετε την main(), ώστε:

i να δημιουργεί ένα νέο αντικείμενο KoutiC με τύπο Kouti,

ii αφού εμφανίσει τον όγκο του πρώτου και του δεύτερου κουτιού να εφαρμόζει τον υπερφορτωμένο τελεστή στα δύο πρώτα κουτιά και να αποθηκεύει το αποτέλεσμα στο τρίτο κουτί - KoutiC,

iii να καλεί την calculateOgkos() επί του KoutiC και να εμφανίζει στην οθόνη το κατάλληλο μήνυμα

2 ΄Ασκηση 2

Σκοπός της άσκησης αυτής είναι:

α΄. να εξοικειωθείτε με την χρήση του vector της STL,

β΄. να πειραματιστείτε με την υλοποίηση ιεραρχίας κλάσεων μέσω κληρονομικότητας,

γ΄. να εξασκηθείτε στην χρήση προσδιοριστικών ορατότητας,

δ΄. να πειραματιστείτε με τον πολυμορφισμό και να αντιληφθείτε τις έννοιες early και late binding

1. Μελετήστε τον κώδικα που ακολουθεί και εξηγήστε την λειτουργία του. Μετατρέψτε τον ώστε στο διάνυσμα vec να τοποθετούνται τυχαίες τιμές χρησιμοποιώντας τη συνάρτηση rand().

```
#include <iostream>
#include <vector>
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(i);
    }
    cout << "extended vector size = " << vec.size() << endl;
    for(i = 0; i < vec.size(); i++)
    {
        cout << "Vector ["<<i<< "] = "<< vec[i] << endl;
    }
    return 0;
}
```

Να κάνετε τις απαραίτητες αλλαγές ώστε τα περιεχόμενα του διανύσματος να εμφανίζονται με την χρήση iterator όπως φαίνεται στις παρακάτω γραμμές κώδικα

```
vector<int>::iterator v = vec.begin();
  while( v != vec.end())
  {
      cout << "value of v = " << *v << endl;
      v++;
  }
```

Προσθέστε τώρα τον κατάλληλο κώδικα ώστε να μειωθεί το μέγεθος του vector σε 5 χρησιμοποιώντας την resize() και στη συνέχεια να εκτυπωθεί το μήνυμα "reduced vector size ="

ακολουθούμενο με το νέο μέγεθος του vector, εμφανίζοντας τα νέα περιεχόμενα του. Τι θα συμβεί στην περίπτωση που αντί της resize() κληθεί η resize(10,5);

2. Εκτελέστε Τον παρακάτω κώδικα και παρατηρήστε τον τρόπο που έχουν οριστεί οι συναρτήσεις getIpsos() και setIpsos(). Λειτουργούν σωστά; Πως μπορούμε να ορίζουμε και να χρησιμοποιούμε συναρτήσεις κλάσεων με αυτό τον τρόπο;

```
#include <iostream>
using namespace std;
class Kouti{
public:
    double mikos = 2.0;
    double platos;
    double ipsos;
    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.platos = 1.0;
    Small.ipsos = 2.0;
    ogkos = Small.platos * Small.ipsos * Small.mikos;
    cout << "Ogkos gia Kouti Small: " << ogkos <<endl;
    Big.mikos = 12.0;
    Big.platos = 13.0;
    Big.ipsos = 10.0;
    ogkos = Big.platos * Big.ipsos * Big.mikos;
    cout << "Ogkos gia Kouti Big: " << ogkos <<endl;
    return 0;
}
```

Να αλλάξετε το προσδιοριστικό πρόσβασης σε private πριν τον ορισμό των ιδιοτήτων της κλάσης Kouti, ώστε να τηρείται η αρχή της απόκρυψης των πληροφοριών. Εκτελέστε τον κώδικα. Τι πρόβλημα προκύπτει και γιατί; Κάντε τις απαραίτητες αλλαγές στην main, χρησιμοποιώντας

μεθόδους αντί των ιδιωτικών ιδιοτήτων ώστε ο κώδικας να λειτουργεί σωστά.

3. Μελετήστε τον παρακάτω κώδικα:

```
#include <iostream>
using namespace std;
class Polygon
{
protected:
    int width, height;
public:
    void set_values (int a, int b)
    {
        width=a;
        height=b;
    }
};
class Rectangle: public Polygon
{
public:
    int area ()
    {
        return width * height;
    }
};
int main ()
{
    Rectangle rect;
    rect.set_values (5,8);
    cout << "Emvadon orthogoniou: " << rect.area() << '\n';
    return 0;
}
```

Προσθέστε στην κατάλληλη θέση της ιεραρχίας και μια κλάση Triangle που θα αναπαριστά αντικείμενα τύπου τρίγωνα για τα οποία διατηρούμε δύο ιδιότητες (μήκος και πλάτος), μια μέθοδο set_values() με δύο ακέραια ορίσματα για το μήκος και το πλάτος και μια μέθοδο area() που θα επιστρέφει το εμβαδό υπολογίζοντας το κατά τα γνωστά, δηλ.(πλάτος*ύψος)/2. Στη συνέχεια προσθέστε στη main αντίστοιχες εντολές ώστε να δημιουργηθεί ένα αντικείμενο τρίγωνο με πλάτος 6 και μήκος 4 και να τυπωθεί στην οθόνη μήνυμα που θα εμφανίζει το εμβαδόν του.

4.Στον κώδικα που καταλήξατε στο ερώτημα 2 προσθέστε την κλάση PaintCost και κάντε τις απαραίτητες αλλαγές ώστε η κλάση Rectangle να είναι απόγονος και της PaintCost.

```
class PaintCost{
    public:
        int getCost(int area){
            return area * 70;
        }
};
```

Συμπληρώστε στη main κατάλληλες εντολές ώστε να εμφανίζεται μετά το μήνυμα για το εμβαδό του ορθογωνίου και το εξής "Synoliko kostos xrwmatos: "....."euro". Η κλάση Rectangle από πόσες κλάσεις κληρονομεί; Μπορεί κάτι τέτοιο να υλοποιηθεί σε Java; Εξηγήστε.

5. Ας υποθέσουμε ότι η μέθοδος area() πρέπει να έχει διαφορετική υλοποίηση στην Polygon από ότι στην Rectangle, όπως φαίνεται στον κώδικα που ακολουθεί:

```
#include <iostream>
using namespace std;
class Polygon {
protected:
    int width, height;
public:
    Polygon(int a=0, int b=0){
        width = a;
        height = b;
    }
    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;
}
```

Εκτελέστε τον παραπάνω κώδικα. Τί αποτέλεσμα προκύπτει; Είναι αυτό που θα περιμέναμε καλώντας την area() επί αντικειμένου τύπου Rectangle; Αναζητείστε πληροφορίες σχετικά με το μηχανισμό early binding που εφαρμόζει η C++ και εξηγήστε γιατί ο παραπάνω κώδικας λει-

τουργεί όπως παρατηρήσατε. Στο σημείο που ορίζεται η area() στην κλάση Polygon προσθέστε το χαρακτηριστικό virtual πριν το int area() υλοποιώντας με αυτό τον τρόπο το μηχανισμό late binding. Εκτελέστε ξανά τον κώδικα. Τί παρατηρείτε;