Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

Εργαστηριακή Άσκηση 1:

1)Περιπτώσεις:

3

12

Αποτελέσματα:

This value is too small.

```
#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
{
	cout <<"This is a big enough number! ";
}
return 0;
}
```

2)Περιπτώσεις:

3,4,5

Αποτελέσματα:

3,4,5

5,4,3

```
#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<<val1<<"\n"<<val2<<"\n"<<val3<<"\n";
system("PAUSE");
cout << val3 <<"\n"<<val2<<"\n"<<val2<<"\n";
return EXIT_SUCCESS;
}
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

3)Το πρόγραμμα δημιουργεί 2 δυσδιάστατους πίνακες first[2][2], second[2][2] και ορίζει 2 μετρητές i,j. Έπειτα περιμένει από τον χρήστη ακεραίους, τους οποίους τοποθετεί αρχικά στην πρώτη γραμμή του πίνακα first[2][2] και μετά στην δεύτερη. Όμοια εκτελείται η ίδια διαδικασία για τον πίνακα second[2][2]. Στο τέλος προσθέτει τα στοιχεία στις αντίστοιχες κοινές θέσεις των πινάκων και αποθηκεύει τα αθροίσματα στις αντίστοιχες κοινές θέσεις του πίνακα first[2][2].

```
#include<iostream>
using namespace std;

int main() {
	cout << "First set of numbers:" << endl;
	cout << "Insert the number of rows and the number of columns." << endl;
	int rows, columns;
	cin >> rows>>columns;
	int **first = new int*[columns];
	for (int counter = 0; counter < columns;counter++)
	{
		first[counter] = new int[rows];
	}
	int **second = new int*[columns];
	for (int counter = 0; counter < columns; counter++)
	{
		second[counter] = new int[rows];
	}
	int i, j;
	for (i = 0; i<rows; i++)
	{
		cout << "Enter two integers: " << i + 1 << endl;
		for (j = 0; j<columns; j++)
		{
			cin >> first[i][j];
		}
	}
	cout << "\n\n\nSecond set of numbers:" << endl;
	for (i = 0; i<rows; i++)
	{
		cout << "Enter two more integers: " << i + 1 << endl;
		for (j = 0; j<columns; j++)
		{
			cin >> second[i][j];
		}
	}
	for (i = 0; i<rows; i++)
	{
		for (j = 0; j<columns; j++)
		{
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

```
                        first[i][j] = first[i][j] + second[i][j];
                }
        }
        cout << "First:" << endl;
        for (i = 0; i<rows; i++)
        {
                for (j = 0; j<columns; j++)
                {
                        cout << first[i][j] << "\t";
                }
                cout << endl;
        }
        for (int counter = 0; counter < columns; counter++)
        {
                delete[] second[counter];
        }
        delete[] second;
        for (int counter = 0; counter < columns; counter++)
        {
                delete[] first[counter];
        }
        delete[] first;
        return 0;
}
```

4)

The volume of KoutiA is: 38.4

The volume of KoutiB is: 50

5)

The volume of KoutiA is: 38.4

The volume of KoutiB is: 50

The volume of KoutiC is: 369

```
#include<iostream>

using namespace std;

class Kouti
{
public:
        double calculateOgkos()
```

```
        {
                return  length*breadth*height;
        }
        void setMikos(double l)
        {
                length=l;
        }
        void setPlatos(double b)
        {
                breadth = b;
        }
        void setYpsos(double h)
        {
                height = h;
        }
        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;
        }
private:
        double length;
        double breadth;
        double height;
};
int main()
{
        double ogkos = 0.0;
        Kouti KoutiA;
        Kouti KoutiB;
        KoutiA.setMikos(2.0);
        KoutiA.setPlatos(3.2);
        KoutiA.setYpsos(6.0) ;
        KoutiB.setMikos(2.5);
        KoutiB.setYpsos(4.0);
        KoutiB.setPlatos(5.0);
        ogkos = KoutiA.calculateOgkos();
        cout << "The volume of KoutiA is: " << ogkos << endl;
        ogkos = KoutiB.calculateOgkos();
        cout << "The volume of KoutiB is: " << ogkos << endl;
        Kouti KoutiC;
        KoutiC = KoutiA + KoutiB;
        ogkos = KoutiC.calculateOgkos();
        cout << "The volume of KoutiC is: " << ogkos << endl;
}
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

Εργαστηριακή Άσκηση 2:

1)Το πρόγραμμα δημιουργεί έναν πίνακα τύπου vector.Εκτυπώνεται το αρχικό μέγεθος του vector, κατόπιν με ένα for-loop, γεμίζεται με τους αριθμούς 0-6 και εκτυπώνεται το καινούργιο μέγεθος του vector. Στο τέλος εκτυπώνονται τα κελιά και τα στοιχεία που περιέχουν.

```
#include <stdlib.h>
#include <iostream>
#include <vector>

using namespace std;

int main()
{
	vector<int> vec;
	int number;
	int i;
	cout << "vector size = " << vec.size() << endl;
	for (i = 0; i < 7; i++)
	{
		number = rand();
		vec.push_back(number);
	}
	cout << "extended vector size = " << vec.size() << endl;
	for (i = 0; i < vec.size(); i++)
	{
		cout << "Vector [" << i << "] = " << vec[i] << endl;
	}
	return 0;
}
```

```
#include <stdlib.h>
#include <iostream>
#include <vector>

using namespace std;

int main()
{
	vector<int> vec;
	int number;
	int i;
	cout << "vector size = " << vec.size() << endl;
	for (i = 0; i < 7; i++)
	{
		number = rand();
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

```
                vec.push_back(number);
        }
        cout << "extended 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 <stdlib.h>
#include <iostream>
#include <vector>

using namespace std;

int main()
{
        vector<int> vec;
        int number;
        int i;
        cout << "vector size = " << vec.size() << endl;
        for (i = 0; i < 7; i++)
        {
                number = rand();
                vec.push_back(number);
        }
        cout << "extended vector size = " << vec.size() << endl;
        vector<int>::iterator v = vec.begin();
        while (v != vec.end())
        {
                cout << "value of v = " << *v << endl;
                v++;
        }
        vec.resize(10,5);
        cout << "reduced vector size = " << vec.size() << endl;
        vector<int>::iterator v1 = vec.begin();
        while(v1 != vec.end())
        {
                cout << "value of v = " << *v1 << endl;
                v1++;
        }
        return 0;
}
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

Όταν καλείται η συνάρτηση resize(10,5) το μέγεθος του vector, αλλάζει σύμφωνα με το πρώτο όρισμα δηλαδή μεγαλώνει το μέγεθος σε 10 κελιά, ενώ το τα καινούργια κελιά γεμίζουν με το δεύτερο όρισμα δηλαδή τα νέα κελιά γεμίζουν με τον αριθμό 5.

2)Οι συναρτήσεις getIpsos() και setIpsos() λειτουργούν σωστά , καθώς έχουν οριστεί με τον τελεστή ανάλυσης εύρους, το σύμβολο ::. Ο τελεστής αυτός επιτρέπει την δημιουργία συναρτήσεων και εκτός του κώδικα μίας κλάσης, ο ορισμός των οποίων είναι ο ίδιος με τον ορισμό των άλλων συναρτήσεων. Η συναρτήσεις που δημιουργούνται με αυτόν τον τρόπο καλούνται όπως όλες οι συναρτήσεις κλάσεων χωρίς καμία διαφορά.

Το πρόβλημα που δημιουργείται είναι η αδυναμία πρόσβασης στις private μεταβλητές της κλάσης Kouti, λόγω της αλλαγής του ορισμού των ιδιοτήτων της κλάσης Kouti.

```
#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;
}
```

3)

```
#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;
```

```
        }
        void set_values(int a, int b)
        {
                width = a;
                height = b;
        }
};

class Triangle : public Rectangle
{
public:
        int area()
        {
                return (width * height) / 2.0;
        }
};

int main()
{
        Rectangle rect;
        rect.set_values(5, 8);
        cout << "Emvadon orthogoniou: " << rect.area() << '\n';
        Triangle tria;
        tria.set_values(6, 4);
        cout << "Emvadon trigonou: " << tria.area() << '\n';
        return 0;
}
```

4)

```
#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:
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

```
        int getCost(int area)
        {
                return area * 70;
        }
};

class Rectangle : public Polygon, public PaintCost
{
public:
        int area()
        {
                return width * height;
        }
        void set_values(int a, int b)
        {
                width = a;
                height = b;
        }
};

class Triangle : public Polygon
{
public:
        int area()
        {
                return (width * height) / 2.0;
        }
};

int main()
{
        Rectangle rect;
        rect.set_values(5, 8);
        cout << "Emvadon orthogoniou: " << rect.area() << '\n';
        cout << "Synoliko kostos xrwmatos: " << rect.getCost(rect.area()) << " euro." << endl;
        Triangle tria;
        tria.set_values(6, 4);
        cout << "Emvadon trigonou: " << tria.area() << '\n';
        return 0;
}
```

Η κλάση Rectangle κληρονομεί από 2 κλάσεις, τις κλάσεις Polygon και PaintCost. Η Java δεν επιτρέπει την λειτουργία της πολλαπλής κληρονομικότητας, αλλά μπορούμε να υλοποιήσουμε τη σχέση κληρονομικότητας αυτή, με την χρήση των interfaces.

5)

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

This is area as computed by the Polygon class.

Επειδή η C++ χρησιμοποιεί τον μηχανισμό early binding, η κλήση της συνάρτησης area() από αρχικοποιημένο δείκτη τύπου Polygon, οδηγεί στην κλήση της συνάρτησης area() που είναι ορισμένη στην κλάση Polygon. Ωστόσο, η αλλαγή του δείκτη, ώστε να 'δείχνει' σε αντικείμενο τύπου Rectangle, υποδεικνύει την κλήση της συνάρτησης area() που είναι ορισμένη στην κλάση Rectangle.

This is area as computed by the Rectangle class.

```
#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;
```

}

Λόγω της αλλαγής της συνάρτησης area() που είναι ορισμένη στην κλάση Polygon, σε virtual, ενεργοποιούμε τον μηχανισμό late binding για την συγκεκριμένη συνάρτηση, οπότε κατά την κλήση της συνάρτησης area(), μετά την αλλαγή του δείκτη τύπου Polygon σε τύπο Rectangle, καλείται η συνάρτηση area() που είναι ορισμένη στην κλάση Rectangle.

Άσκηση 1 του Σετ Ασκήσεων

α) Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class Undergrad.

Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class PhD.

Οι συναρτήσεις message() στις κλάσεις Undergrad και PhD επικαλύπτουν την αντίστοιχη συνάρτηση message() της υπερκλάση Student για τα αντικείμενα Ph1 και U1, ενώ καλείται η κληρονομημένη συνάρτηση message() της κλάσης Student για το αντικείμενο M1, λόγω απουσίας υλοποίησης της συνάρτησης message() στην κλάση MSc.

β) Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class Undergrad.

Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class PhD.

Hello, I am message() defined in superclass Student.

Αυτή την φορά κλήθηκε η συνάρτηση message() της υπερκλάση Student για το αντικείμενο S1, παρόλο που άλλαξε σε αντικείμενο τύπου undergrad, επειδή το αντικείμενο S1 είναι αντικείμενο τύπου student και όχι υποκλάσης του undergrad, γενικότερα, λόγω του μηχανισμού late binding.

γ) Hello, I am message() defined in superclass Student.

Hello, I am message() defined in superclass Student.

Hello, I am message() defined in superclass Student.

Εξαιτίας του μηχανισμού late binding αφού ο δείκτης ‘δείχνει’ σε αντικείμενο τύπου student, δεν λαμβάνονται υπόψη οι αλλαγές σε αυτόν και εκτελείται η συνάρτηση message() της κλάσης student για κάθε αντικείμενο. Προκειμένου να εμφανιστούν τα αναμενόμενα αποτελέσματα, απαιτείται ο

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

καθορισμός των συναρτήσεων message() κάθε κλάσης ως virtual ώστε να γίνεται χρήση του μηχανισμού late binding.

Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class Undergrad.

Hello, I am message() defined in class PhD.

δ)
```
#include <iostream>
#include <cstdlib>

using namespace std;

class student
{
private:
	string name;
	string surname;
	int Am;
public:
	virtual void message()
	{
		cout << "Hello, I am message() defined in superclass Student." << endl;
	}
};

class undergrad :public student
{
private:
	int entrance_order;
	int passed_courses;
public:
	virtual void message()
	{
		cout << "Hello, I am message() defined in class Undergrad." << endl;
	}
};
class MSc :public student
{
private:
	string dipl_dept;
	int dipl_grade;
	string thesis;
public:
	int get_dipl_grade()
```

```
        {
        }
};

class PhD :public student
{
private:
        string PhD_title;
        string professor;
        int start_month;
        int start_year;
public:
        virtual void message()
        {
                cout << "Hello, I am message() defined in class PhD." << endl;
        }
};

int main()
{
        int counter;
        student *pointer;
        student *array[4];
        student S1;
        undergrad U1;
        MSc M1;
        PhD Ph1;
        array[0]= &S1;
        array[1] = &U1;
        array[2] = &M1;
        array[3] = &Ph1;
        for (counter = 0; counter < 4; counter++)
        {
                array[counter]->message();
        }
        system("pause");
        return 0;
}
```

Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class Undergrad.

Hello, I am message() defined in superclass Student.

Hello, I am message() defined in class PhD.

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

Άσκηση 2 του Σετ Ασκήσεων

α)Ο κώδικας δίνει την δυνατότητα στον χρήστη να εισάγει το όνομα, την τιμή και το ‘score’ διαφόρων προϊόντων, ελέγχοντας το κλάσμα ‘score’ / τιμή και αν το κλάσμα αυτό είναι μεγαλύτερο από το αντίστοιχο κλάσμα του καλύτερου προϊόντος, το εισαγόμενο προϊόν γίνεται το καλύτερο προϊόν. Στην περίπτωση του πρώτου εισαγόμενου προϊόντος, το πρώτο προϊόν ανακηρύσσεται το καλύτερο. Η διαδικασία αυτή επαναλαμβάνεται μέχρι ο χρήστης να δηλώσει ότι δεν θέλει να εισάγει χαρακτηριστικά άλλου προϊόντος. Στο τέλος, εμφανίζονται τα χαρακτηριστικά του καλύτερου προϊόντος.

β)
```
#include <iostream>
#include <string>

using namespace std;

class Product
{
private:
	string name;
	double price;
	int score;
public:
	Product()//Constructor
	{
		name = "";
		price = 1.0;
		score = 0;
	}
	string getName()
	{
		return name;
	}
	double getPrice()
	{
		return price;
	}
	int getScore()
	{
		return score;
	}
	void read()
	{
		cout << "Please enter the model name: ";
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

```
		getline(cin, name);
		cout << "Please enter the price: ";
		cin >> price;
		cout << "Please enter the score: ";
		cin >> score;
		string remainder; /* read remainder of line */
		getline(cin, remainder);
	}
	void print() const
	{
		cout << name << endl
		<< " (Score: " << score << endl
		<< " Price: " << price << ")" << endl;
	}
	bool is_better_than(Product a)
	{
		if ((score / price) > (a.score / a.price))
		{
			return true;
		}
		return false;
	}
};

int main()
{
	Product best;
	bool more = true;
	while (more)
	{
		Product next;
		next.read();
		if (next.is_better_than(best))
		{
			best = next;
		}
		cout << "More data? (y/n) ";
		string answer;
		getline(cin, answer);
		if (answer != "y") more = false;
	}
	cout << "The best value is ";
	best.print();
	return 0;
}


γ)#include <iostream>
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

```
#include <string>

using namespace std;

class Product
{
private:
	string name;
	double price;
	int score;
public:
	Product()//Constructor
	{
		name = "";
		price = 1.0;
		score = 0;
	}
	string getName()
	{
		return name;
	}
	double getPrice()
	{
		return price;
	}
	int getScore()
	{
		return score;
	}
	void read()
	{
		cout << "Please enter the model name: ";
		getline(cin, name);
		cout << "Please enter the price: ";
		cin >> price;
		cout << "Please enter the score: ";
		cin >> score;
		string remainder; /* read remainder of line */
		getline(cin, remainder);
	}
	friend void print(Product a)
	{
		cout << a.getName() << endl
			<< " (Score: " << a.getScore() << endl
			<< " Price: " << a.getPrice() << ")" << endl;
	}
	bool is_better_than(Product a)
	{
```

```
                if ((score / price) > (a.score / a.price))
                {
                        return true;
                }
                return false;
        }
};

int main()
{
        Product best;
        bool more = true;
        while (more)
        {
                Product next;
                next.read();
                if (next.is_better_than(best))
                {
                        best = next;
                }
                cout << "More data? (y/n) ";
                string answer;
                getline(cin, answer);
                if (answer != "y") more = false;
        }
        cout << "The best value is ";
        print(best);
        return 0;
}
```

δ)

```
#include <iostream>
#include <string>

using namespace std;

class Product
{
private:
        string name;
        double price;
        int score;
public:
        Product()//Constructor
        {
                name = "";
                price = 1.0;
                score = 0;
```

```
	}
	string getName()
	{
		return name;
	}
	double getPrice()
	{
		return price;
	}
	int getScore()
	{
		return score;
	}
	void read()
	{
		cout << "Please enter the model name: ";
		getline(cin, name);
		cout << "Please enter the price: ";
		cin >> price;
		cout << "Please enter the score: ";
		cin >> score;
		string remainder; /* read remainder of line */
		getline(cin, remainder);
	}
	friend void print(Product a)
	{
		cout << a.getName() << endl
			<< " (Score: " << a.getScore() << endl
			<< " Price: " << a.getPrice() << ")" << endl;
	}
	bool operator >(const Product& p)
	{
		if ((score / price) > (p.score / p.price))
		{
			return true;
		}
		return false;
	}
};

int main()
{
	Product best;
	bool more = true;
	while (more)
	{
		Product next;
		next.read();
```

Αναφορά για το 3ο σετ εργαστηριακών ασκήσεων C++

```
                if (next > best)
                {
                        best = next;
                }
                cout << "More data? (y/n) ";
                string answer;
                getline(cin, answer);
                if (answer != "y") more = false;
        }
        cout << "The best value is ";
        print(best);
        return 0;
}
```