Δρ. Γ. Σ. Τσελίκης

C++ Από τη Θεωρία στην Εφαρμογή

| Περιέχει **500+** λυμένες ασκήσεις
και παραδείγματα κλιμακούμενης δυσκολίας

Ενδεικτικές Ασκήσεις Βιβλίου

Ε.1 Σε ένα τεστ 50 ερωτήσεων πολλαπλής επιλογής η πρώτη απάντηση παίρνει τρεις βαθμούς, η δεύτερη έναν βαθμό και η τρίτη δύο βαθμούς. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τις 50 απαντήσεις ενός εξεταζόμενου και να εμφανίζει τους συνολικούς του βαθμούς με χρήση της εντολής **switch**. Αν ο εξεταζόμενος επιλέξει τρεις φορές διαδοχικά την πρώτη απάντηση το πρόγραμμα να εμφανίζει τους συνολικούς του βαθμούς και να τερματίζει. Θεωρήστε ότι ο εξεταζόμενος δίνει έγκυρες απαντήσεις με αριθμούς στο [1, 3]. Ο παρακάτω κώδικας δεν είναι σωστός. Μπορείτε να βρείτε το γιατί;

```
#include <iostream>
int main()
{
	int i, ans, first_ans, points;

	points = first_ans = 0;
	for(i = 0; i <= 50; i++)
	{
		std::cout << "Enter answer [1-3]: ";
		std::cin >> ans;

		switch(ans)
		{
			case 1:
				first_ans++;
				if(first_ans == 3)
				{
					std::cout << "Candidate gets " << points << " points in total\n";
					return 0; // Τερματισμός προγράμματος.
				}
				points += 3;
			break;

			case 2:
				points += 1;
			break;

			case 3:
				points += 2;
			break;
		}
	}
	std::cout << "Candidate gets " << points << " points in total\n";
	return 0;
}
```

Ε.2 Να γραφεί ένα πρόγραμμα το οποίο να εμφανίζει, αν υπάρχουν, τις ακέραιες λύσεις της εξίσωσης: `3x + 7y – 5z = 10`, όπου τα `x`, `y`, `z` είναι ακέραιοι στο [-30, 30]. Επίσης, αν υπάρχουν λύσεις, να εμφανίζει ξεχωριστά στο τέλος και αυτή στην οποία το άθροισμα των `x`, `y`, `z` έχει τη μικρότερη τιμή.

Ε.3 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει ακέραιες τιμές και να τις αποθηκεύει σε ένα `vector` αντικείμενο. Αν ο χρήστης εισάγει την τιμή `-1`, η εισαγωγή των ακεραίων να τερματίζει. Στη συνέχεια, το πρόγραμμα να ελέγχει αν το διάνυσμα είναι συμμετρικό, δηλαδή, αν το πρώτο του στοιχείο είναι ίσο με το τελευταίο, το δεύτερο με το προτελευταίο, κ.ο.κ. Αν δεν είναι συμμετρικό, το πρόγραμμα να ελέγχει αν υπάρχει κάποια τιμή που να επαναλαμβάνεται και, αν ναι, να τερματίζει άμεσα. Αλλιώς, αν όλες οι τιμές είναι διαφορετικές να εμφανίζεται ανάλογο μήνυμα.

Ε.4 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει ακεραίους και να τους αποθηκεύει σε έναν τετραγωνικό πίνακα (π.χ. `3×3`). Στη συνέχεια, να ελέγχει αν ο πίνακας αποτελεί μαγικό τετράγωνο, δηλαδή αν το άθροισμα της κάθε γραμμής, στήλης και διαγωνίου του είναι το ίδιο.

Ε.5 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <iostream>
int main()
{
	int *ptr1, *ptr2, i = 10, j = 20;

	ptr1 = &i;
	ptr2 = &j;

	ptr2 = ptr1;
	*ptr1 = *ptr1 + *ptr2;
	*ptr2 *= 2;
	std::cout << *ptr1 + *ptr2 << '\n';
	return 0;
}
```

Ε.6 Να συμπληρώσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τον δείκτη `p2` για να διαβάζετε συνεχώς βαθμούς φοιτητών, τον δείκτη `p1` για να εμφανίσετε πόσοι φοιτητές πήραν βαθμό στο [5, 10] και τον δείκτη `p3` για να εμφανίσετε τον μεγαλύτερο βαθμό σε αυτό το διάστημα. Αν ο χρήστης εισάγει την τιμή `-1`, η εισαγωγή των βαθμών να τερματίζεται. Οι μεταβλητές `sum`, `max` και `grade` να χρησιμοποιηθούν μόνο μία φορά.

```
#include <iostream>
int main()
{
	int *p1, sum;
	float *p2, *p3, grade, max;
```

```
        ...
}
```

Ε.7 Να συμπληρώσετε το παρακάτω πρόγραμμα ώστε να διαβάζει δύο ακεραίους και να εμφανίζει το άθροισμα των ακεραίων που μεσολαβούν μεταξύ τους, χρησιμοποιώντας τους δείκτες `p1`, `p2` και `p3`. Για παράδειγμα, αν ο χρήστης εισάγει τους ακεραίους `6` και `10`, το πρόγραμμα να εμφανίζει `24 (7+8+9)`. Το πρόγραμμα να υποχρεώνει τον χρήστη να εισάγει τιμές μικρότερες από `100` και ο πρώτος ακέραιος να είναι μικρότερος από τον δεύτερο. Οι μεταβλητές `i`, `j` και `sum` να χρησιμοποιηθούν μόνο μία φορά.

```
#include <iostream>
int main()
{
        int *p1, *p2, *p3, i, j, sum;
        ...
}
```

Ε.8 Να συμπληρώσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τους δείκτες `p1` και `p2` για να διαβάσετε και να αποθηκεύσετε στον πίνακα `arr` τους ακεραίους κωδικούς 100 προϊόντων, με τον περιορισμό ένας κωδικός να αποθηκεύεται στον πίνακα μόνο αν δεν έχει ήδη αποθηκευτεί. Αν ο χρήστης εισάγει την τιμή `-1`, η εισαγωγή των κωδικών να τερματίζει. Το πρόγραμμα να εμφανίζει τους κωδικούς πριν τερματίσει. Η διαχείριση του πίνακα να γίνει με σημειογραφία δείκτη και η μεταβλητή `arr` να χρησιμοποιηθεί μέχρι 5 φορές.

```
#include <iostream>
int main()
{
        const int SIZE = 100;
        int *p1, *p2, arr[SIZE];
        ...
}
```

Για την εισαγωγή των κωδικών, το πρόγραμμα να εμφανίζει μηνύματα με την παρακάτω μορφή:

```
Enter code_1:
Enter code_2:
...
```

Ε.9 Να γραφεί ένα πρόγραμμα το οποίο να προσομοιώνει το δημοφιλές παιχνίδι της «κρεμάλας». Το πρόγραμμα να διαβάζει μία λέξη μέχρι 30 χαρακτήρες, η οποία θα αποτελεί τη μυστική λέξη. Στη συνέχεια, το πρόγραμμα να διαβάζει επαναληπτικά κάποιο γράμμα από τον παίκτη. Αν το γράμμα δεν περιέχεται στη μυστική λέξη, το πρόγραμμα να ενημερώνει τον παίκτη. Αν περιέχεται, να εμφανίζει τα γράμματα που έχουν βρεθεί και στα υπόλοιπα να βάζει τον χαρακτήρα της υπογράμμισης *(underscore)*. Το παιχνίδι ολοκληρώνεται όταν βρεθεί η λέξη ή αν ο παίκτης έχει κάνει

7 λανθασμένες προβλέψεις. Θεωρήστε ότι δεν επιτρέπεται ο χρήστης να εισάγει το ίδιο γράμμα πάνω από μία φορά.

Ε.10 Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει συνεχώς αλφαριθμητικά και να αποθηκεύει σε ένα `vector` που περιέχει `string` αντικείμενα αυτά που αρχίζουν με `'a'` και τελειώνουν σε `'s'`. Στη συνέχεια, το πρόγραμμα να διαβάζει ένα αλφαριθμητικό και να εμφανίζει πόσες φορές έχει αποθηκευτεί. Αν ο χρήστης εισάγει το `end`, η εισαγωγή των αλφαριθμητικών να τερματίζει.

Ε.11 Να συμπληρώσετε τα κενά στο πρόγραμμα (δεν επιτρέπεται να προσθέσετε άλλες εντολές ή μεταβλητές) ώστε η τιμή του `a` μετά την κλήση της `f()` να γίνεται ίση με τετράγωνο της τιμής που θα εισάγει ο χρήστης (π.χ. αν ο χρήστης εισάγει 4, το πρόγραμμα να εμφανίζει 16).

```
#include <iostream>

void f(____);

int main()
{
        int a;

        std::cin >> a;

        ____;  /* Συμπληρώστε την κλήση της συνάρτησης. */

        std::cout << a << '\n';
        return 0;
}

void f(____)
{
        ____;  /* Συμπληρώστε το σώμα της συνάρτησης. */
}
```

Ε.12 Δημιουργήστε μία **void** συνάρτηση που να δέχεται σαν παραμέτρους ένα `vector` αντικείμενο που περιέχει τις τιμές των εμπορευμάτων σε ένα κατάστημα και να επιστρέφει, μέσω παραμέτρων, την ελάχιστη τιμή, τη μέγιστη και τον μέσο όρο των τιμών. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει συνεχώς τιμές εμπορευμάτων και να τις αποθηκεύει σε ένα `vector` αντικείμενο. Αν ο χρήστης εισάγει αρνητική τιμή ή μηδέν, η εισαγωγή των τιμών να τερματίζει. Το πρόγραμμα να εμφανίζει την ελάχιστη τιμή, τη μέγιστη και τον μέσο όρο των τιμών με χρήση της συνάρτησης.

Ε.13 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <iostream>

void swap(const char *s1, const char *s2);
```

```
int main()
{
	const char *p[] = {"Shadow", "Play"};

	swap(p[0], p[1]);
	std::cout << p[0] << ' ' << p[1] << '\n';
	return 0;
}

void swap(const char *s1, const char *s2)
{
	const char *p;

	p = s1;
	s1 = s2;
	s2 = p;
}
```

Ε.14 Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους δύο `vector` αντικείμενα που περιέχουν ονόματα και να επιστρέφει ένα `vector` αντικείμενο που να αποτελεί την τομή τους, δηλαδή, να περιέχει τα κοινά ονόματα. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τον αριθμό των ονομάτων, να δημιουργεί δύο `vector` αντικείμενα, να διαβάζει τα ονόματα και να τα καταχωρεί σε αυτά. Μετά, να καλεί την συνάρτηση και να εμφανίζει τα κοινά ονόματα. Για τα ονόματα, χρησιμοποιήστε τον τύπο `string`.

Ε.15 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <iostream>

void f(int num);

int main()
{
	f(3);
	return 0;
}

void f(int num)
{
	int i;

	if(num >= 1)
	{
		std::cout << '\n';
		for(i = 0; i < num; i++)
			std::cout << "One ";
		f(num/2);
		std::cout << "\n***";
		f(num/2);
	}
}
```

Ε.16 Να ορίσετε τη δομή `Product` με πεδία: όνομα προϊόντος, κωδικός και τιμή. Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους ένα `vector` τέτοιων δομών και τον κωδικό ενός προϊόντος. Η συνάρτηση να ελέγχει αν υπάρχει κάποιο στοιχείο του διανύσματος που να έχει τον ίδιο κωδικό και, αν υπάρχει, να επιστρέφει έναν δείκτη σε αυτό το στοιχείο, αλλιώς να επιστρέφει **`nullptr`**. Να γραφεί ένα πρόγραμμα που να χρησιμοποιεί αυτή τη δομή για να εισάγει ο χρήστης τα στοιχεία προϊόντων και να τα αποθηκεύσει σε ένα `vector` διάνυσμα. Η εισαγωγή των προϊόντων να τερματίζει αν ο χρήστης εισάγει αρνητική τιμή ή μηδέν. Στη συνέχεια, να διαβάζει τον κωδικό ενός προϊόντος, να καλεί τη συνάρτηση και, αν το προϊόν υπάρχει, να εμφανίζει τα υπόλοιπα στοιχεία του.

Ε.17 Να ορίσετε τη δομή `Student` με πεδία: όνομα φοιτητή και βαθμός. Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους δύο αναφορές σε `string` αντικείμενα και να επιστρέφει μία αναφορά στο μεγαλύτερο. Υπερφορτώστε την συνάρτηση με μία άλλη συνάρτηση, η οποία να δέχεται σαν παραμέτρους δύο αναφορές σε δομές τύπου `Student` και να επιστρέφει το όνομα του φοιτητή με το μεγαλύτερο βαθμό. Χειριστείτε την περίπτωση που είναι ίδιοι. Να γραφεί ένα πρόγραμμα το οποίο να ελέγχει την λειτουργία των δύο συναρτήσεων.

Ε.18 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <iostream>

double* test(double *p1, double *p2);
double& test(double& r1, double& r2);

int main()
{
	double a = 1.2, b = 3.4;

	*test(&a, &b) = 5.6;
	std::cout << a << ' ' << b << '\n';

	test(a, b) = 7.8;
	std::cout << a << ' ' << b << '\n';
	return 0;
}

double* test(double *p1, double *p2)
{
	if(*p1 < *p2)
		return p1;
	else
		return p2;
}

double& test(double& r1, double& r2)
{
	if(r1 < r2)
		return r1;
```

```
        else
                return r2;
}
```

Ε.19 Δημιουργήστε μία πρότυπη συνάρτηση που να δέχεται σαν παραμέτρο ένα vector αντικείμενο με αριθμητικά στοιχεία γενικού τύπου και να επιστρέφει τον μέσο όρο τους. Υπερφορτώστε την συνάρτηση με μία άλλη πρότυπη συνάρτηση με τύπο επιστροφής **void**, η οποία να δέχεται δύο αριθμητικές παραμέτρους που μπορεί να έχουν διαφορετικούς τύπους και να επιστρέφει μέσω δείκτη τον μέσο όρο τους. Να γραφεί ένα πρόγραμμα το οποίο να ελέγχει την λειτουργία των δύο συναρτήσεων.

Ε.20 Υπάρχουν λάθη στο παρακάτω πρόγραμμα; Αν ναι, να τα διορθώσετε και να βρείτε την έξοδο του προγράμματος.

```
#include <iostream>
#include <vector>

class A
{
public:
        ~A() {std::cout << "~A\n";}
};

class B
{
public:
        ~B() {std::cout << "~B\n";}
};

class C
{
public:
        std::vector<A> v;
        B *b;
        C& push(A *p) {v.push_back(*p); return *this;}
};

int main()
{
        C c;
        A *p = new A;
        std::cout << &c.push(p) << '\n';
        delete p;
        return 0;
}
```

Ε.21 Συμπληρώστε το δημόσιο τμήμα της παρακάτω κλάσης και δημιουργήστε ένα πρόγραμμα, το οποίο να διαβάζει και να αποθηκεύει τα στοιχεία υπαλλήλων σε ένα vector αντικείμενο που θα περιέχει στοιχεία τύπου Employee. Αν ο χρήστης εισάγει αρνητική τιμή για μισθό, η εισαγωγή των στοιχείων να τερματίζει. Στη συνέχεια, το πρόγραμμα να διαβάζει ένα ποσό και να εμφανίζει τα στοιχεία των

υπαλλήλων που έχουν μεγαλύτερο μισθό από αυτόν. Επίσης, το πρόγραμμα να εμφανίζει τα στοιχεία του υπαλλήλου με τον μεγαλύτερο μισθό. Θεωρήστε ότι μόνο ένας υπάλληλος έχει τον μεγαλύτερο μισθό. Υπάρχει ο περιορισμός να μην ορίσετε κατασκευαστή στην κλάση.

```
class Employee
{
private:
	string name;
	float sal;
public:
	...
};
```

Ε.22 Συμπληρώστε το δημόσιο τμήμα της παρακάτω κλάσης και δημιουργήστε ένα πρόγραμμα, το οποίο να διαβάζει και να αποθηκεύει τα στοιχεία φοιτητών σε ένα `vector` αντικείμενο που θα περιέχει στοιχεία τύπου `Student`. Αν ο χρήστης εισάγει αρνητική τιμή για βαθμό, η εισαγωγή των στοιχείων να τερματίζει. Στη συνέχεια, το πρόγραμμα να καλεί μία συνάρτηση `sort()` που να παίρνει σαν όρισμα το διάνυσμα και να ταξινομεί τα στοιχεία του κατά αύξουσα σειρά βαθμού. Για την ταξινόμηση χρησιμοποιήστε τον αλγόριθμο επιλογής. Επίσης, η κλάση να περιέχει μία συνάρτηση `show()`, την οποία το πρόγραμμα να καλεί για να εμφανίσει τα στοιχεία των φοιτητών. Υπάρχει ο περιορισμός να ορίσετε μόνο τρεις συναρτήσεις συνολικά.

```
class Student
{
private:
	string name;
	float grd;
public:
	...
};
```

Ε.23 Να ορίσετε την κλάση `Time` με ιδιωτικά πεδία τα ώρες (π.χ. `hrs`), λεπτά (π.χ. `mins`) και δευτερόλεπτα (π.χ. `secs`). Να προσθέσετε κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
int main()
{
	Time t1, t2(23, 59, 59); /* Υπάρχει ο περιορισμός η κλάση να έχει έναν κατασκευαστή. Τα πεδία του t1 να αρχικοποιούνται με 0. Επίσης, υπάρχει ο περιορισμός οι παράμετροι του κατασκευαστή να έχουν τα ίδια ονόματα με τα ιδιωτικά πεδία. */
	++t2; /* Αυξάνεται ο αριθμός των δευτερολέπτων κατά ένα. Αν η ώρα είναι 23:59:59, όπως στην περίπτωση του t2, η νέα ώρα να γίνει 0:0:0. */
	cout << (t1 == t2) << '\n'; /* Να συγκρίνονται οι δύο ώρες και αν είναι ίδιες, το πρόγραμμα να εμφανίζει 1 αλλιώς 0. Για παράδειγμα, με αυτές τις τιμές των t1 και t2 το πρόγραμμα εμφανίζει 1. */
	cout << t1; /* Να εμφανίζεται η ώρα στη μορφή h:m:s. Για παράδειγμα, 0:0:0. */
```

```
	return 0;
}
```

Ε.24 Θεωρήστε την παρακάτω κλάση Student. Προσθέστε όποια συνάρτηση ή πεδίο κρίνετε σκόπιμο ώστε να λειτουργεί το παρακάτω πρόγραμμα. Να μη δηλώσετε καμία συνάρτηση εμβόλιμη και να καταργήσετε την πράξη της αντιγραφής, δηλαδή, να μην επιτρέπεται η εντολή s1 = s2. Επίσης, υπάρχει ο περιορισμός συνολικά στο πρόγραμμα να ορίσετε μέχρι τέσσερις συναρτήσεις (εκτός της main()).

```
class Student
{
private:
	int code; // Αριθμός μητρώου.
	int courses; // Αριθμός μαθημάτων.
	float *grd; /* Να δεσμευτεί μνήμη για την αποθήκευση των βαθμών
του φοιτητή σε courses μαθήματα. */
	...
public:
	...
};

int main()
{
	int i;
	Student s1, s2;

	cin >> s1 >> s2; /* Ο χρήστης να εισάγει τιμές για τα πεδία code
και courses των δύο αντικειμένων. Μετά, το πρόγραμμα να διαβάζει τους
βαθμούς κάθε φοιτητή και να τα αποθηκεύει στη μνήμη που δείχνει το
αντίστοιχο grd μέλος. */
	i = (s1 > s2); /* Η συνάρτηση υπερφόρτωσης να συγκρίνει τον
αριθμό των επιτυχόντων μαθημάτων και να επιστρέφει 1 αν ο πρώτος
φοιτητής έχει περάσει περισσότερα μαθήματα, 2 αν ο δεύτερος έχει
περισσότερα μαθήματα και 3 αν έχουν περάσει τον ίδιο αριθμό μαθημάτων.
Ένα μάθημα θεωρείται περασμένο αν ο βαθμός σε αυτό είναι >= 5. */
	if(...) /* Ελέγξτε την τιμή επιστροφής του i σύμφωνα με τα
παραπάνω. Συμπληρώστε τον κώδικα, ώστε το πρόγραμμα να εμφανίζει τον
αριθμό μητρώου του φοιτητή που έχει περάσει τα περισσότερα μαθήματα ή
Same αν έχουν περάσει τον ίδιο αριθμό. */
	return 0;
}
```

Ε.25 Θεωρήστε τις παρακάτω κλάσεις Product και Employee:

```
class Product
{
	private:
		int code;
		float prc; // Τιμή προϊόντος.
	public:
		...
};
```

```
class Employee
{
	private:
		string name;
		float sal; // Μισθός υπαλλήλου.
	public:
		...
};
```

Να ορίσετε μία πρότυπη συνάρτηση `sort()` που να δέχεται σαν όρισμα ένα `vector` αντικείμενο (π.χ. `v`), το οποίο να περιέχει `Product` ή `Employee` στοιχεία και να τα ταξινομεί. Αν περιέχει `Product` στοιχεία, η ταξινόμηση να γίνεται κατά αύξουσα σειρά των τιμών των προϊόντων. Αν περιέχει `Employee` στοιχεία, η ταξινόμηση να γίνεται κατά αλφαβητική σειρά των ονομάτων. Για την ταξινόμηση χρησιμοποιήστε τον αλγόριθμο επιλογής. Κάθε κλάση να περιέχει έναν **`inline`** κατασκευαστή, μία **`inline`** συνάρτηση `show()` για την εμφάνιση των πεδίων της, και την απαραίτητη συνάρτηση για την σύγκριση των στοιχείων του `v`. Να γραφεί ένα πρόγραμμα το οποίο να δηλώνει δύο `vector` αντικείμενα με `Product` και `Employee` στοιχεία, να καλεί την `sort()` και να εμφανίζει τα στοιχεία τους.

Ε.26 Προσθέστε στην κλάση κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
class T
{
private:
	string str;
	int key;
public:
	void show() const {cout << str << ' ' << key;} /* Δεν επιτρέπεται να αλλάξετε το σώμα της. */
	...
};

void f(T t) // Δεν επιτρέπεται να αλλάξετε το σώμα της.
{
	t.show();
}

int main()
{
	T t;

	t["abc"] = 10; /* Το str να γίνει ίσο με abc και το key ίσο με 10. */
	f(t); /* Προσοχή, το πρόγραμμα πρέπει πρώτα να εμφανίζει Yes και μετά abc και 10. */
	return 0;
}
```

Ε.27 Προσθέστε στην κλάση κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
class T
{
private:
	char *s;
	...
};

int main()
{
	T t1("abc"), t2("def"), t3("gg"), t4("hh"); /* Το κάθε αλφαριθμητικό να αντιγράφεται στο αντίστοιχο πεδίο s. */
	t4 = t3 = t1+t2; /* Η ένωση των αλφαριθμητικών που είναι αποθηκευμένα στα t1 και t2, δηλαδή το abcdef, πρέπει να αποθηκευτεί στο πεδίο s των t3 και t4. */
	T t5(t4);
	t5[2] = 'x'; /* Το αποθηκευμένο αλφαριθμητικό πρέπει να γίνει abxdef. */
	return 0;
}
```

Ε.28 Ποια είναι τα λάθη στο παρακάτω πρόγραμμα;

```
#include <iostream>

class A
{
private:
	int p;
public:
	A(int a) {p = a;}
};

class B : public A
{
public:
	B(int v) {std::cout << v << '\n';}
	void show(int v) const {std::cout << p+v << '\n';}
};

int main()
{
	B b(10);
	A& r = b;

	r.show(20);
	return 0;
}
```

Ε.29 Να ορίσετε την κλάση A, από την A να παράγεται με δημόσια πρόσβαση η B, και από την B να παράγεται με δημόσια πρόσβαση η C. Κάθε κλάση να περιέχει ένα ιδιωτικό μέλος τύπου string (π.χ. name) και έναν κατασκευαστή, ο οποίος να έχει

άδειο σώμα. Προσθέστε στις κλάσεις κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
int main()
{
	C c("Last", "Sec", "First"); /* Το Last να αποθηκεύεται στο
πεδίο name της C κλάσης, το Sec στο πεδίο name της B και το First στο
πεδίο name της A. */
	A *p = &c;
	p->show(); /* Το πρόγραμμα να εμφανίζει την τιμή του πεδίου name
της C, δηλαδή, Last. */
	return 0;
}
```

Ε.30 Να ορίσετε την κλάση `Product` με προστατευμένο πεδίο τον κωδικό του προϊόντος (π.χ. `code`). Από την κλάση `Product` να παράγεται με δημόσια πρόσβαση η κλάση `Book` με ιδιωτικό πεδίο τον τίτλο του βιβλίου (π.χ. `title`). Κάθε κλάση να περιέχει την δική της έκδοση της `show()`, η οποία να εμφανίζει την τιμή του μέλους της. Να ορίσετε την συνάρτηση `f()` η οποία να δέχεται σαν ορίσματα δύο δείκτες σε αντικείμενα και να εμφανίζει τις τιμές των μελών τους. Προσθέστε στις κλάσεις κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
int main()
{
	Product prod(100); // Ο κωδικός γίνεται 100.
	Book b("C++", 200); /* Ο κωδικός γίνεται 200 και ο τίτλος
γίνεται C++. Ο κατασκευαστής της Book να καλεί τον κατασκευαστή της
Product. */
	... /* Δηλώστε ένα δείκτη που να δείχνει στο prod, έναν άλλο που
να δείχνει στο b και μετά καλέστε την f(). */
	return 0;
}

void f(____ *p1, ____ *p2) /* Βρείτε τον τύπο των δεικτών, πρέπει να
είναι ο ίδιος. */
{
	p1->show(); /* Να εμφανίζεται ο κωδικός του Product
αντικειμένου, δηλαδή, 100. */
	p2->show(); /* Να εμφανίζεται ο κωδικός και ο τίτλος του Book
αντικειμένου, δηλαδή, 200 και C++. */
}
```

Ε.31 Ποια είναι η έξοδος του παρακάτω προγράμματος;

```
#include <iostream>
using std::cout;

class A
{
private:
	int k;
public:
	A() {k = 1;}
```

```
        void f() const {cout << k;}
        virtual void v() const {cout << " A.v() ";}
};

class B : public A
{
private:
        int k;
public:
        B() {k = 2;}
        void f() const {cout << k;}
        virtual void v() const override {cout << " B.v() ";}
};

void t1(A a)
{
        a.f();
        a.v();
}

void t2(A& r)
{
        r.f();
        r.v();
}

int main()
{
        B b;
        t1(b);
        t2(b);
        return 0;
}
```

Ε.32 Να ορίσετε την κλάση `Fruit` με ιδιωτικά πεδία το όνομα του φρούτου και τον αριθμό των θερμίδων του (π.χ. `cal`). Η κλάση να περιέχει κατάλληλο κατασκευαστή που να αρχικοποιεί τα πεδία της. Από την κλάση `Fruit` να παράγεται με δημόσια πρόσβαση η κλάση `Apple` με ιδιωτικό πεδίο τον αριθμό των αντιοξειδωτικών που περιέχει (π.χ. `antiox`). Η κλάση να περιέχει κατασκευαστή που να αρχικοποιεί το πεδίο της και ο οποίος να καλεί τον κατασκευαστή της `Fruit`. Να ορίσετε την συνάρτηση `f()` η οποία να δέχεται σαν όρισμα μία αναφορά σε ένα αντικείμενο και μία ακέραια τιμή και να τα προσθέτει. Να γραφεί ένα πρόγραμμα σύμφωνα με τα παρακάτω σχόλια.

```
int main()
{
        /* Το πρόγραμμα να δημιουργεί ένα Fruit αντικείμενο (π.χ. fr)
και ένα Apple αντικείμενο (π.χ. ap). Μετά, να καλεί την f() δύο φορές.
Την πρώτη με όρισμα το fr, και την δεύτερη με όρισμα το ap. */
}

void f(____ &p, int num) // Βρείτε τον τύπο της αναφοράς.
{
```

```
        cout << p+num << '\n'; /* Με την p+num, αν ο τύπος του
αντικειμένου είναι Fruit, το num να προστίθεται στην τιμή του πεδίου
cal και αυτή η νέα τιμή να εμφανίζεται, ενώ, αν είναι Apple, το num να
προστίθεται στην τιμή του πεδίου antiox και αυτή η νέα τιμή να
εμφανίζεται. */
}
```

Ε.33 Ποια είναι τα λάθη στο παρακάτω πρόγραμμα;

```
#include <iostream>

class A
{
public:
        int p;
        A(int a) {p = a;}
};

class B
{
public:
        int p;
        B() {p = 20;}
        void show(const char str[]) const {std::cout << str << '\n';}
};

class C : public A, public B
{
private:
        int v;
public:
        C() : v(10) {}
        void show() const {std::cout << v << '\n';}
};

int main()
{
        C c;
        c.p = 10;
        c.show("test");
        return 0;
}
```

Ε.34 Να ορίσετε την αφηρημένη κλάση με το όνομα `Person`, η οποία να περιέχει σαν ιδιωτικά πεδία το όνομα (τύπου `string`) και τον κωδικό (ακέραιος). Να ορίσετε την κλάση `Student` με δημόσια πρόσβαση από την `Person` με ιδιωτικό πεδίο τους βαθμούς του σε πέντε μαθήματα (π.χ. `grd[5]`). Επίσης, να ορίσετε την κλάση `Employee` με δημόσια πρόσβαση από την `Person` με ιδιωτικό πεδίο τις αμοιβές του σε τρία έργα που συμμετέχει (π.χ. `fee[3]`). Προσθέστε στις κλάσεις κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
int main()
{
```

```
        Person *p;
        Student s; /* Το name να αρχικοποιείται με None και το code με
την τιμή -1. */
        Employee e; /* Το name να αρχικοποιείται με None και το code με
την τιμή -1. */
        p = &s;
        (*p)[3] = 5; // Η τιμή του s.grd[3] να γίνει 5.
        p = &e;
        (*p)[1] = 100; // Η τιμή του e.fee[1] να γίνει 100.
        return 0;
}
```

Ε.35 Να ορίσετε την κλάση `Shape` με ιδιωτικά πεδία τις συντεταγμένες του κέντρου `x` και `y` και την γνήσια εικονική συνάρτηση `area()` για τον υπολογισμό του εμβαδού. Να ορίσετε την κλάση `Shape2D` με δημόσια πρόσβαση από την `Shape` με ιδιωτικό πεδίο το όνομα του σχήματος. Να ορίσετε την κλάση `Rect` με δημόσια πρόσβαση από την `Shape2D` με ιδιωτικά πεδία το μήκος και ύψος του ορθογωνίου. Να ορίσετε την κλάση `Shape3D` με δημόσια πρόσβαση από την `Shape` με ιδιωτικό πεδίο το χρώμα του σχήματος και την γνήσια εικονική συνάρτηση `vol()` για τον υπολογισμό του όγκου. Να ορίσετε την κλάση `Sphere` με δημόσια πρόσβαση από την `Shape3D` με ιδιωτικό πεδίο την ακτίνα της σφαίρας. Οι κλάσεις `Rect` και `Sphere` να μην είναι αφηρημένες. Προσθέστε στις κλάσεις κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
int main()
{
        Sphere sph(1, 2, "Blue", 3); /* Το x πεδίο του Shape υπο-
αντικειμένου να γίνει 1, το y πεδίο 2, το χρώμα του Shape3D υπο-
αντικειμένου να γίνει Blue και η ακτίνα της sph να γίνει 3. */
        Rect r(4, 5, "Rect", 6, 7); /* Το x πεδίο του Shape υπο-
αντικειμένου να γίνει 4, το y πεδίο 5, το όνομα του Shape2D υπο-
αντικειμένου να γίνει Rect, το μήκος του r να γίνει 6 και το ύψος του
7. */
        Shape *p[2] = {&sph, &r};
        p[0]->show(); /* Το πρόγραμμα να εμφανίζει τις τιμές του κέντρου
x και y, καθώς και το εμβαδό και τον όγκο της σφαίρας. */
        p[1]->show(); /* Το πρόγραμμα να εμφανίζει τις τιμές του κέντρου
x και y, καθώς και το εμβαδό του τετραγώνου. */
        return 0;
}
```

Ε.36 Να ορίσετε την κλάση `Account` με ιδιωτικά πεδία το όνομα του δικαιούχου (π.χ. `name`) και το ποσό του λογαριασμού (π.χ. `bal`). Η κλάση να περιέχει την δημόσια συνάρτηση `create_acnt()` η οποία να καλείται μετά τη δημιουργία κάποιου `Account` αντικειμένου με ορίσματα το όνομα του δικαιούχου και το αρχικό ποσό κατάθεσης. Το όνομα πρέπει να περιέχει μόνο χαρακτήρες. Αν δεν περιέχει ή αν η τιμή του ποσού είναι αρνητική, η `create_acnt()` να δημιουργεί μία εξαίρεση με τύπο αντικείμενο της παρακάτω κλάσης `Err_Rpt` και αντίστοιχο διαγνωστικό μήνυμα. Αν οι τιμές των ορισμάτων είναι έγκυρες να αποθηκεύονται στα μέλη `name`

και bal, αντίστοιχα. Το πρόγραμμα να διαβάζει συνεχώς όνομα και ποσό κατάθεσης μέχρι να είναι έγκυρα, δηλαδή, η create_acnt() να μην δημιουργήσει εξαίρεση. Στη συνέχεια, το πρόγραμμα να διαβάζει ένα ποσό και να καλεί τις συναρτήσεις deposit() και withdraw(). Αυτές να αποτελούν δημόσιες συναρτήσεις της Account και να δέχονται σαν όρισμα το ποσό που θα προστεθεί ή θα αφαιρεθεί από το ποσό του λογαριασμού, αντίστοιχα. Αν το ποσό δεν είναι έγκυρο, οι συναρτήσεις να δημιουργούν μία εξαίρεση με τύπο Err_Rpt. Αν είναι έγκυρο, η τιμή του bal να ενημερώνεται ανάλογα. Τέλος, το πρόγραμμα να καλεί μία δημόσια συνάρτηση show(), η οποία να εμφανίζει τις τιμές των name και bal.

```
class Err_Rpt
{
public:
        string msg;
        Err_Rpt(const char *m) : msg(m) {}
};
```

Ε.37 Υπάρχουν λάθη στο παρακάτω πρόγραμμα; Αν όχι, τι θα εμφανίσει;

```
#include <iostream>

class A
{
public:
        void p() const {std::cout << "p\n";}
};

template <typename T> class B
{
private:
        T x;
public:
        int k;
        B(T y) : x(y), k(2) {x.p();}
};

void f(B& r)
{
        std::cout << r.k;
}

int main()
{
        A a;
        B<A*> b(&a);
        f(b);
        return 0;
}
```

Ε.38 Να ορίσετε την πρότυπη κλάση Circle<T> με ιδιωτικό πεδίο την ακτίνα (π.χ. rad) του κύκλου. Ο τύπος της rad να είναι T. Προσθέστε στην κλάση κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
int main()
{
	Circle<int> c1(5), c2; /* Ο τύπος της rad γίνεται int και η τιμή της ίση με την τιμή του ορίσματος. Η προκαθορισμένη αρχική τιμή είναι 0. */
	c2+3; // Το rad αυξάνεται κατά 3.
	Circle<int> c3 = c1+c2; /* Το c3.rad να γίνει ίσο με το άθροισμα των c1.rad και c2.rad. */
	Circle<double> c4(1.5);
	c4+1.2; // Το rad να αυξάνεται κατά 1.2.
	cout << c3 << c4 << '\n'; /* Το πρόγραμμα να εμφανίζει την τιμή των c3.rad και c4.rad, δηλαδή, 8 και 2.7. */
	return 0;
}
```

Ε.39 Να ορίσετε την πρότυπη κλάση `Map_Key<T1, T2>` με δύο ιδιωτικά πεδία τύπου `T1` και `T2`. Προσθέστε στην κλάση κατάλληλες συναρτήσεις ώστε να λειτουργεί το παρακάτω πρόγραμμα.

```
template <typename T1, typename T2> class Map_Key
{
private:
	T1 key;
	T2 val;
public:
	____ check(____); /* Ορίστε την check() η οποία να δέχεται σαν παράμετρο μία τιμή (π.χ. k) και να ελέγχει αν η τιμή του key είναι ίδια με του k. */
	____ get(____); /* Ορίστε την get() η οποία να επιστρέφει την τιμή val. */
	// Προσθέστε όποια άλλη συνάρτηση κρίνετε σκόπιμο.
};

int main()
{
	Map_Key<string, int> mk_1("One", 1), mk_2("Two", 2);
	Map_Key<string, int> mk_3 = mk_1 + mk_2;
	/* Δηλώστε ένα vector (π.χ. v) με στοιχεία Map_Key και αποθηκεύστε τα mk_1, mk_2, και mk_3 σε αυτό. Μετά, διαβάστε ένα αλφαριθμητικό και ελέγξτε αν είναι αποθηκευμένο στο v. Αν είναι, το πρόγραμμα να εμφανίζει την τιμή του val. Για παράδειγμα, αν ο χρήστης εισάγει το One, το πρόγραμμα να εμφανίζει 1. */
	...
}
```

Ε.40 Να ορίσετε την κλάση `Country` με ιδιωτικά πεδία: όνομα, πρωτεύουσα και πληθυσμός. Θεωρήστε ότι κάθε γραμμή του αρχείου κειμένου `country.txt` περιέχει τα στοιχεία χωρών με τη μορφή:

```
όνομα  πρωτεύουσα  πληθυσμός
```

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει το αρχείο και να χρησιμοποιεί την κλάση για να αποθηκεύσει τα στοιχεία των χωρών σε ένα `vector` με `Country`

στοιχεία. Στη συνέχεια, να διαβάζει έναν αριθμό και να εμφανίζει τα στοιχεία των χωρών με μεγαλύτερο πληθυσμό από αυτόν τον αριθμό. Προσθέστε στην κλάση όποια συνάρτηση θεωρείτε σκόπιμο.

Ε.41 Υποθέστε ότι το δυαδικό αρχείο `test.bin` περιέχει τους βαθμούς ενός φοιτητή. Το πλήθος των βαθμών είναι αποθηκευμένο στην αρχή του αρχείου. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει τους βαθμούς από το αρχείο (χρησιμοποιήστε τον τύπο **`float`**) και να τους αποθηκεύει σε μία δυναμικά δεσμευμένη μήμη. Στη συνέχεια, να διαβάζει έναν αριθμό και να εμφανίζει τους βαθμούς με μεγαλύτερη τιμή από αυτόν.

Ενδεικτικές Απαντήσεις

Ε.1 *Απάντηση:*

Ένα λάθος είναι ότι η εντολή `points += 3;` πρέπει να πάει πριν το **`if`**, ώστε να προστεθούν οι βαθμοί της τρίτης διαδοχικής απάντησης πριν το πρόγραμμα τερματίσει με την **`return`**. Ένα δεύτερο λάθος είναι ότι δεν μηδενίζεται η `first_ans` στην περίπτωση που ο εξεταζόμενος επιλέξει τη δεύτερη ή την τρίτη απάντηση, ώστε να αρχίσει η μέτρηση των διαδοχικών πρώτων απαντήσεων από το μηδέν. Άρα, σε αυτές τις δύο **`case`** περιπτώσεις πρέπει να προστεθεί η εντολή `first_ans = 0`. Ένα τρίτο λάθος είναι ότι ο αριθμός των επαναλήψεων δεν είναι 50 αλλά 51. Απλά, διαγράφουμε το = και γράφουμε `i < 50` .

Ε.2 *Απάντηση:*

```
#include <iostream>
int main()
{
	int x, y, z, sum, min_sum, min_x, min_y, min_z, flag;

	flag = 1;
	min_sum = 91; /* Θέτουμε μία μέγιστη τιμή σύμφωνα με το διάστημα
των λύσεων που καθορίζεται στην εκφώνηση. */
	for(x = -30; x <= 30; x++)
		for(y = -30; y <= 30; y++)
			for(z = -30; z <= 30; z++)
				if(3*x + 7*y - 5*z == 10)
				{
					flag = 0;
					std::cout << "Solution: " << x << ' '
<< y << ' ' << z << ' ' << '\n';
					sum = x+y+z;
					if(sum < min_sum)
					{
						min_x = x;
						min_y = y;
						min_z = z;
						min_sum = sum;
					}
				}
	if(flag)
		std::cout << "No solution\n";
	else
		std::cout << "Values: " << min_x << ' ' << min_y << ' ' <<
min_z << ' ' << '\n';
	return 0;
}
```

Ε.3 *Απάντηση:*

```
#include <iostream>
#include <vector>
using std::cout;
using std::cin;
using std::vector;

int main()
{
	bool flag;
	int i, j, size;
	vector<int> v;

	while(1)
	{
		cout << "Enter number: ";
		cin >> i;
		if(i == -1)
			break;
		v.push_back(i);
	}
	size = v.size();
	flag = 1;
	for(i = 0; i < size/2 ; i++)
		if(v[i] != v[size-1-i])
		{
			flag = 0;
			break; /* Αφού διαπιστώσαμε ότι το διάνυσμα δεν
είναι συμμετρικό, ο βρόχος τερματίζεται. */
		}
	if(flag)
		cout << "Symmetric\n";
	else
	{
		cout << "Not symmetric with ";
		for(i = 0; i < size; i++)
		{
			for(j = i+1; j < size; j++) /* Αυτός ο βρόχος
ελέγχει αν το v[i] υπάρχει στον πίνακα. */
			{
				if(v[i] == v[j])
				{
					cout << "same values\n";
					return 0; /* Αφού διαπιστώσαμε ότι δύο
στοιχεία έχουν την ίδια τιμή, το πρόγραμμα τερματίζεται άμεσα. */
				}
			}
		}
		cout << "different values\n";
	}
	return 0;
}
```

Σχόλια: Αν κάποιος αναρωτηθεί, τι γίνεται αν ο αριθμός των στοιχείων είναι περιττός, δεν υπάρχει πρόβλημα με τον έλεγχο της συμμετρίας, αφού το μεσαίο στοιχείο δεν συγκρίνεται με κάποιο άλλο στοιχείο.

Ε.4 *Απάντηση:*

```
#include <iostream>
using std::cout;
using std::cin;

int main()
{
	const int ROWS = 3;
	const int COLS = 3;
	int i, j, sum, tmp, arr[ROWS][COLS];

	sum = tmp = 0;
	for(i = 0; i < ROWS; i++)
	{
		// Βρίσκουμε τα αθροίσματα των στοιχείων των διαγωνίων.
		for(j = 0; j < COLS; j++)
		{
			cout << "Enter element [" << i << "]" << "[" << j
<< "]: ";
			cin >> arr[i][j];
			if(i == j)
				sum += arr[i][j];
			if(i+j  ==  ROWS-1)  /* Ελέγχουμε  αν  το  στοιχείο
βρίσκεται στη δευτερεύουσα διαγώνιο. */
				tmp += arr[i][j];
		}
	}
	if(sum != tmp)
	{
		cout << "Not magic square -> Sum_main_diag: " << sum << "
Sum_sec_diag: " << tmp << '\n';
		return 0;
	}
	for(i = 0; i < ROWS; i++)
	{
		/* Αρχικοποίηση της μεταβλητής που υπολογίζει το άθροισμα
των στοιχείων της κάθε γραμμής. */
		tmp = 0;
		for(j = 0; j < COLS; j++)
			tmp += arr[i][j];

		if(sum != tmp)
		{
			cout << "Not magic square -> Sum_row_" << i+1 << ":
" << tmp << " Sum_diag: " << sum << '\n';
			return 0;
		}
	}
	for(i = 0; i < COLS; i++)
	{
```

```
                tmp = 0; /* Αρχικοποίηση της μεταβλητής που υπολογίζει το
άθροισμα των στοιχείων της κάθε στήλης. */
                for(j = 0; j < ROWS; j++)
                        tmp += arr[j][i];

                if(sum != tmp)
                {
                        cout << "Not magic square -> Sum_col_" << i+1 << ":
" << tmp << " Sum_diag: " << sum << '\n';
                        return 0;
                }
        }
        cout << "Magic square !!!\n";
        return 0;
}
```

E.5 *Απάντηση:*

Απάντηση: Με την εντολή `ptr2 = ptr1;` ο `ptr2` δείχνει εκεί που δείχνει ο `ptr1`, δηλαδή, στη διεύθυνση του `i`. Άρα, το `*ptr2` είναι ίσο με την τιμή του `i`. Αφού και οι δύο δείκτες δείχνουν στη διεύθυνση του `i`, η εντολή `*ptr1 = *ptr1 + *ptr2;` ισοδυναμεί με `i = i+i = 10+10 = 20`. Παρόμοια, η εντολή `*ptr2 *= 2;` αναλύεται σε `*ptr2 = 2*(*ptr2);` που ισοδυναμεί με `i = 2*i = 2*20 = 40`. Τελικά, το πρόγραμμα εμφανίζει την τιμή της έκφρασης `*ptr1 + *ptr2 = i+i = 40+40 = 80`.

E.6 *Απάντηση:*

```
#include <iostream>
int main()
{
        int *p1, sum;
        float *p2, *p3, grade, max;

        p1 = &sum;
        *p1 = 0;

        p3 = &max;
        *p3 = 0;

        p2 = &grade;
        while(1)
        {
                std::cout << "Enter grade: ";
                std::cin >> *p2;

                if(*p2 == -1)
                        break;
                if(*p2 >=5 && *p2 <= 10)
                {
                        (*p1)++;
                        if(*p2 > *p3)
                                *p3 = *p2;
                }
```

```
        }
        std::cout << *p1 << " students passed (max:" << *p3 << ")\n";
        return 0;
}
```

E.7 *Απάντηση:*

```
#include <iostream>
int main()
{
        int *p1, *p2, *p3, i, j, sum;

        p1 = &i;
        p2 = &j;
        p3 = &sum;
        *p3 = 0;
        do
        {
                std::cout << "Enter two numbers (a < b < 100): ";
                std::cin >> *p1 >> *p2;
        } while(*p1 >= *p2 || *p2 > 100);

        (*p1)++;
        while(*p1 < *p2)
        {
                *p3 += *p1;
                (*p1)++;
        }
        std::cout << "Sum: " << *p3 << '\n';
        return 0;
}
```

E.8 *Απάντηση:*

```
#include <iostream>
using std::cout;
using std::cin;

int main()
{
        const int SIZE = 100;
        int *p1, *p2, arr[SIZE];

        p1 = arr;
        while(p1 < arr+SIZE)
        {
                cout << "Enter code_" << p1-arr+1 << ": ";
                cin >> *p1;
                if(*p1 == -1)
                        break;

                for(p2 = arr; p2 < p1; p2++) /* Ξεκινώντας από την αρχή
του πίνακα, ελέγχουμε αν ο κωδικός έχει ήδη αποθηκευτεί. */
                {
                        if(*p1 == *p2)
```

```
                        {
                                cout << "Error: Code " << *p1 << " exists "
<< '\n';
                                break;
                        }
                }
                /* Αν ο κωδικός δεν υπάρχει στον πίνακα αυξάνουμε τον
δείκτη. */
                if(p2 == p1)
                        p1++;
        }
        // Εμφάνιση κωδικών.
        for(p2 = arr; p2 < p1; p2++)
                cout << "C:" << *p2 << '\n';
        return 0;
}
```

E.9 *Απάντηση:*

```
#include <iostream>
#include <cstring>
using std::cout;
using std::cin;

int main()
{
        const int ERROR_TRIES = 7;
        char ch, secret[30], hide[30] = {0};
        int i, found, len, error, correct;

        cout << "Enter secret word: ";
        cin.getline(secret, sizeof(secret));
        len = strlen(secret);

        for(i = 0; i < len; i++)
                hide[i] = '_';

        error = correct = 0;
        while(error < ERROR_TRIES)
        {
                cout << "Enter character: ";
                ch = cin.get();
                found = 0;
                for(i = 0; i < len; i++)
                {
                        if(secret[i] == ch)
                        {
                                hide[i] = ch;
                                found = 1;
                                correct++;
                                if(correct == len)
                                {
                                        cout << "Secret word is found !!!\n";
                                        return 0;
                                }
                        }
```

```
                }
                if(found == 0)
                {
                        error++;
                        cout << "Error, " << ch << " does not exist. You've
got " << ERROR_TRIES - error << " more attempts\n";
                }
                else
                        cout << hide << '\n';
                cin.get(); /* Εξάγουμε το '\n', από την προηγούμενη get().
*/
        }
        cout << "Sorry, the secret word was " << secret << '\n';
        return 0;
}
```

E.10 *Απάντηση:*

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

int main()
{
        int i, cnt, size;
        string s;
        vector<string> v_str;

        while(1)
        {
                cout << "Enter text: ";
                getline(cin, s);
                if(s == "end")
                        break;
                size = s.size();
                if(s[0] == 'a' && s[size-1] == 's')
                        v_str.push_back(s);
        }
        size = v_str.size();
        if(size == 0)
        {
                cout << "None string is stored\n"; /* Δεν έχει νόημα να
συνεχίσουμε. */
                return 0;
        }
        cout << "Enter text: ";
        getline(cin, s);
        cnt = 0;
        for(i = 0; i < size; i++)
        {
                if(v_str[i] == s)
                        cnt++;
        }
        cout << s << " is stored " << cnt << " times\n";
```

```
	return 0;
}
```

Ε.11 *Απάντηση:*

Αφού θέλουμε η `f()` να αλλάζει την τιμή του a, πρέπει να της μεταβιβάσουμε την διεύθυνση του a. Αφού το a είναι ακέραιος, η παράμετρος της `f()` είναι δείκτης σε ακέραιο (π.χ. `int *p`). Στο δεύτερο κενό, για την κλήση της `f()` γράφουμε `f(&a)`. Τέλος, στο σώμα της `f()` χρησιμοποιούμε τον δείκτη και γράφουμε `*p = *p * *p;`.

Ε.12 *Απάντηση:*

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

void stat(const vector<float>& v, float *min, float *max, float *avg);

int main()
{
	int i;
	float min, max, avg, val;
	vector<float> prc_v;

	while(1)
	{
		cout << "Enter price: ";
		cin >> val;
		if(val <= 0)
			break;
		prc_v.push_back(val);
	}
	if(prc_v.size() == 0) /* Σημαίνει ότι δεν αποθηκεύτηκε καμία
τιμή. */
		return 0;
	stat(prc_v, &min, &max, &avg);
	cout << "Max=" << max << " Min=" << min << " Avg=" << avg <<
'\n';
	return 0;
}

void stat(const vector<float>& v, float *min, float *max, float *avg)
{
	int i, size;
	float sum;

	sum = *min = *max = v[0];
	size = v.size();
	for(i = 1; i < size; i++)
	{
		if(v[i] > *max)
			*max = v[i];
```

```
            if(v[i] < *min)
                  *min = v[i];
            sum += v[i];
      }
      *avg = sum/size;
}
```

Σχόλια: Αφού ο τύπος επιστροφής της συνάρτησης είναι **void** χρησιμοποιούμε παραμέτρους δείκτες, ώστε να μπορεί η συνάρτηση να επιστρέψει τις επιθυμητές τιμές.

E.13 *Απάντηση:*

Η `swap()` δέχεται σαν ορίσματα τις τιμές των δεικτών, όχι τις διευθύνσεις τους. Άρα, η αντιμετάθεση δεν έχει καμία επίδραση και το πρόγραμμα εμφανίζει `Shadow Play`. Αυτό που θέλω να κάνετε είναι να τροποποιήσετε το πρόγραμμα, ώστε να εμφανίζει `Play Shadow`. Για να δούμε, μπορείτε; Υπόδειξη, αλλάξτε τον τύπο των παραμέτρων σε... βρείτε το.

E.14 *Απάντηση:*

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

vector<string>     find_same(const     vector<string>&     v1,     const
vector<string>& v2);

int main()
{
      int i, num;

      cout << "Enter number of names: ";
      cin >> num;
      vector<string> v1(num), v2(num);
      for(i = 0; i < num; i++)
      {
            cout << "Enter name_" << i+1 << ": ";
            cin >> v1[i];
            cout << "Enter name_" << i+1 << ": ";
            cin >> v2[i];
      }
      vector<string> v3 = find_same(v1, v2);
      num = v3.size();
      if(num == 0)
            cout << "No common names\n";
      else
      {
            cout << "\nCommon names\n";
            for(i = 0; i < num; i++)
                  cout << v3[i] << '\n';
```

```
                /* Θα μπορούσαμε να γράψουμε:
                for(auto& s : v3)
                        cout << s << '\n'; */
        }
        return 0;
}

vector<string>      find_same(const      vector<string>&      v1,      const
vector<string>& v2)
{
        int i, j, num;
        vector<string> v;
        num = v1.size();
        for(i = 0; i < num; i++)
        {
                for(j = 0; j < num; j++)
                {
                        if(v1[i] == v2[j])
                        {
                                v.push_back(v1[i]);
                                break;   /* Τερματισμός του εσωτερικού for
και συνεχίζουμε με τον έλεγχο του επόμενου στοιχείου. */
                        }
                }
        }
        return v;
}
```

Ε.15 *Απάντηση:*

Με την πρώτη κλήση της `f()` το πρόγραμμα εμφανίζει τρία `One`. Μετά, καλείται πάλι η `f()` με όρισμα `1`, ενώ η `cout` και η επόμενη κλήση της `f()` με όρισμα `1` θα γίνουν όταν τερματιστεί η προηγούμενη κλήση. Άρα, το πρόγραμμα εμφανίζει ένα `One`, καλείται η `f()` με όρισμα `0`, η οποία επιστρέφει. Τώρα, καλείται η `cout` που δεν είχε εκτελεστεί προηγουμένως και μετά πάλι η `f()` με όρισμα `0`. Στη συνέχεια, καλούνται οι `cout` και η `f()` με όρισμα `1`. Το πρόγραμμα εμφανίζει ένα `One` και, όπως πριν, καλείται η `f()` με όρισμα `0`, η οποία επιστρέφει. Μετά, καλείται η `cout` και μετά πάλι η `f()` με όρισμα `0`. Άρα, τελικά το πρόγραμμα εμφανίζει:

```
One One One
One
***
***
One
***
```

Ε.16 *Απάντηση:*

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

```
struct Product
{
	string name;
	int code;
	float prc;
};

const Product *find_prod(const vector<Product>& v, int code);

int main()
{
	const Product *ptr;
	int code;
	float prc;
	Product tmp;
	vector<Product> prod;

	while(1)
	{
		cout << "\nPrice: ";
		cin >> prc;
		if(prc <= 0)
			break;

		tmp.prc = prc;
		cin.get();
		cout << "Name: ";
		getline(cin, tmp.name);

		cout << "Code: ";
		cin >> tmp.code;
		prod.push_back(tmp);
	}
	if(prod.size() == 0)
	{
		cout << "No product is stored\n";
		return 0;
	}
	cout << "\nEnter code to search: ";
	cin >> code;

	ptr = find_prod(prod, code);
	if(ptr == nullptr)
		cout << "\nNo product with code = " << code << '\n';
	else
		cout << "\nN:" << ptr->name << " C:" << code << " P:" <<
ptr->prc << '\n';
	return 0;
}

const Product *find_prod(const vector<Product>& v, int code)
{
	for(int i = 0; i < v.size(); i++)
	{
		if(v[i].code == code)
```

```
                    return &v[i]; /* Αν βρεθεί ο κωδικός του προϊόντος,
η συνάρτηση τερματίζει άμεσα και επιστρέφει τη διεύθυνση της δομής. */
        }
        return nullptr; /* Αν φτάσουμε σε αυτό το σημείο σημαίνει ότι ο
κωδικός του προϊόντος δεν βρέθηκε, οπότε η συνάρτηση επιστρέφει
nullptr. */
}
```

E.17 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

struct Student
{
        string name;
        float grd;
};

const string& cmp(const string& a, const string& b);
string cmp(const Student& a, const Student& b);

int main()
{
        string s1 = "alpha", s2 = "beta";
        Student st1 = {"N.Smith", 6.5}, st2 = {"P.Krow", 5.2};

        string tmp = cmp(s1, s2);
        cout << tmp << '\n';

        tmp = cmp(st1, st2);
        if(tmp == "")
                cout << "Same grades\n";
        else
                cout << tmp << '\n';
        return 0;
}

const string& cmp(const string& a, const string& b)
{
        if(a > b)
                return a;
        else
                return b;
}

string cmp(const Student& a, const Student& b)
{
        if(a.grd > b.grd)
                return a.name;
        else if(a.grd < b.grd)
                return b.name;
        else
```

```
                return "";
}
```

Ε.18 *Απάντηση:*

Απάντηση: Η πρώτη έκδοση της `test()` επιστρέφει τον δείκτη στην παράμετρο με τη μικρότερη τιμή, ενώ η δεύτερη την αναφορά. Στην πρώτη κλήση έχουμε `p1 = &a`, άρα `*p1 = a = 1.2`. Παρομοίως, `p2 = &b`, άρα `*p2 = b = 3.4`. Επομένως, η `test()` επιστρέφει τον δείκτη `p1`. Αφού η `test()` επιστρέφει τον δείκτη στο `a`, η εντολή `*test(&a, &b) = 5.6;` κάνει το `a` ίσο με `5.6` και το πρόγραμμα εμφανίζει: `5.6 3.4`

Στη δεύτερη κλήση της `test()`, αφού `r1 = a = 5.6` και `r2 = b = 3.4`, η `test()` επιστρέφει αναφορά στο `r2`. Αφού η `test()` επιστρέφει αναφορά στο `b`, το `b` γίνεται ίσο με `7.8` και το πρόγραμμα εμφανίζει: `5.6 7.8`.

Ε.19 *Απάντηση:*

```
#include <iostream>
#include <vector>
using std::vector;
using std::cout;

template <typename T> double avg(const vector<T>& v);
template <typename T1, typename T2> void avg(T1 t1, T2 t2, double *p);

int main()
{
        double i;
        vector<int> v = {1, 2, 4};

        avg(1, 2, &i);
        std::cout << "Avg_1:" << avg(v) << '\n';
        std::cout << "Avg_2:" << i << '\n';
        return 0;
}

template <typename T> double avg(const vector<T>& v)
{
        T sum;
        int i, size;

        size = v.size();
        sum = 0;
        for(i = 0; i < size; i++)
                sum += v[i];
        return (double)sum/size;
}

template <typename T1, typename T2> void avg(T1 t1, T2 t2, double *p)
{
        *p = (t1+t2)/2.0;
```

```
}
```

Ε.20 *Απάντηση:*

Είναι λάθος αν κάποια κλάση περιέχει μόνο αποδομητή; Όχι βέβαια. Όταν δηλώνεται το p, ο εξ'ορισμού κατασκευαστή δημιουργεί ένα A αντικείμενο. Είναι σωστή η κλήση της push(); Ναι, γιατί να μην είναι; Όταν καλείται, η push_back() αποθηκεύει ένα αντίγραφο του αντικειμένου που μεταβιβάζεται στο c.v. Η push() επιστρέφει το c αντικείμενο και έτσι το πρόγραμμα εμφανίζει τη διεύθυνση μνήμης του. Μετά, αποδεσμεύεται η μνήμη για τον δείκτη p, καλείται ο αποδομητής της A και το πρόγραμμα εμφανίζει ~A. Τέλος, καλείται ο εξ'ορισμού αποδομητής της C. Θα κληθεί ο αποδομητής της B; Όχι βέβαια, δεν έχει δημιουργηθεί B αντικείμενο, ο b είναι ένας απλός δείκτης. Προσοχή τώρα, όταν καταστραφεί το v, θα κληθεί ο αποδομητής για κάθε A αντικείμενο που περιέχει. Επειδή περιέχει μόνο ένα, το πρόγραμμα εμφανίζει ~A. Συνοπτικά, το πρόγραμμα δεν έχει κανένα λάθος και εμφανίζει πρώτα τη διεύθυνση μνήμης του c, και μετά δύο φορές ~A.

Ε.21 *Απάντηση:*

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

class Employee
{
private:
	string name;
	float sal;
public:
	void set(const string& n, float s);
	float get() const;
	void show() const;
};

void Employee::set(const string& n, float s)
{
	name = n;
	sal = s;
}

float Employee::get() const
{
	return sal;
}

void Employee::show() const
{
	cout << "N:" << name << " S:" << sal << '\n';
}
```

```
int main()
{
	int i, pos;
	float base, sal, max_sal;
	string name;
	Employee tmp;
	vector<Employee> empl;

	while(1)
	{
		cout << "\nName: ";
		getline(cin, name);

		cout << "Salary: ";
		cin >> sal;
		if(sal == -1)
			break;

		tmp.set(name, sal);
		empl.push_back(tmp);
		cin.get(); // Απομακρύνουμε τον χαρακτήρα νέας γραμμής.
	}
	cout << "\nEnter base: ";
	cin >> base;
	max_sal = -1;
	for(i = 0; i < empl.size(); i++)
	{
		sal = empl[i].get();
		if(sal > base)
			empl[i].show();
		if(sal > max_sal)
		{
			pos = i;
			max_sal = sal;
		}
	}
	if(max_sal != -1)
		empl[pos].show();
	return 0;
}
```

Σχόλια: Χρειαζόμαστε `set/get` συναρτήσεις για την πρόσβαση στα ιδιωτικά μέλη και μία `show()` για την εμφάνιση των δεδομένων.

Ε.22 *Απάντηση:*

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

class Student
{
private:
	string name;
```

```
        float grd;
public:
        Student(const string& n = "", float g = 0);
        friend void sort(vector<Student>& v);
        void show() const;
};

Student::Student(const string& n, float g)
{
        name = n;
        grd = g;
}

void Student::show() const
{
        cout << "N:" << name << " G:" << grd << '\n';
}

void sort(vector<Student>& v)
{
        int i, j, size;
        Student tmp;

        size = v.size();
        for(i = 0; i < size-1; i++)
        {
                for(j = i+1; j < size; j++)
                {
                        if(v[i].grd > v[j].grd)
                        {
                                tmp = v[i];
                                v[i] = v[j];
                                v[j] = tmp;
                        }
                }
        }
}

int main()
{
        int i;
        float grd;
        string name;
        vector<Student> st;

        while(1)
        {
                cout << "\nName: ";
                getline(cin, name);

                cout << "Grade: ";
                cin >> grd;
                if(grd == -1)
                        break;

                Student tmp(name, grd);
```

```
                st.push_back(tmp);
                cin.get();
        }
        sort(st);
        for(i = 0; i < st.size(); i++)
                st[i].show();
        return 0;
}
```

Σχόλια: Χρειαζόμαστε έναν κατασκευαστή για τη δημιουργία των αντικειμένων. Αφού υπάρχει ο περιορισμός για τρεις συναρτήσεις, η συνάρτηση ταξινόμησης πρέπει να δηλωθεί σαν φιλική, ώστε να έχει πρόσβαση στα ιδιωτικά μέλη.

E.23 ***Απάντηση:***

```
#include <iostream>
using std::cout;
using std::ostream;

class Time
{
private:
        int hrs;
        int mins;
        int secs;
public:
        Time(int hrs=0, int mins=0, int secs=0);
        bool operator==(const Time& t) const;
        void operator++();
        friend ostream& operator<<(ostream& out, const Time& t);
};

Time::Time(int hrs, int mins, int secs)
{
        this->hrs = hrs; /* Επειδή υπάρχει ο περιορισμός για ίδια
ονόματα χρησιμοποιούμε τον δείκτη this. */
        this->mins = mins;
        this->secs = secs;
}

bool Time::operator==(const Time& t) const
{
        if((hrs == t.hrs) && (mins == t.mins) && (secs == t.secs))
                return true;
        else
                return false;
}

void Time::operator++()
{
        secs++;
        if(secs == 60)
        {
                secs = 0;
```

```
                mins++;
                if(mins == 60)
                {
                        mins = 0;
                        hrs++;
                        if(hrs == 24)
                                hrs = 0;
                }
        }
}

ostream& operator<<(ostream& out, const Time& t)
{
        out << t.hrs << ':' << t.mins << ':' << t.secs << '\n';
        return out;
}
```

Ε.24 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::cin;
using std::istream;

class Student
{
private:
        int code;
        int courses;
        float *grd;
        int suc; /* Προστέθηκε αυτό το πεδίο για τον αριθμό των
επιτυχόντων μαθημάτων. */
public:
        ~Student();
        void show() const;
        int operator>(const Student& s) const;
        friend istream& operator>>(istream& in, Student& s);
        Student& operator=(const Student&) = delete; /* Κατάργηση
αντιγραφής. */
};

Student::~Student()
{
        delete[] grd;
}

int Student::operator>(const Student& s) const
{
        if(suc > s.suc)
                return 1;
        else if(suc < s.suc)
                return 2;
        else
                return 3;
```

```
}

void Student::show() const
{
	cout << "\nCode:" << code << '\n';
}


istream& operator>>(istream& in, Student& s)
{
	int i;

	cout << "Enter code: ";
	in >> s.code;
	cout << "Enter number of courses: ";
	in >> s.courses;
	s.grd = new float[s.courses];
	s.suc = 0;

	for(i = 0; i < s.courses; i++)
	{
		cout << "Enter grade: ";
		in >> s.grd[i];
		if(s.grd[i] >= 5)
			s.suc++;
	}
	return in;
}

int main()
{
	int i;
	Student s1, s2;

	cin >> s1 >> s2;
	i = (s1 > s2);
	if(i == 1)
		s1.show();
	else if(i == 2)
		s2.show();
	else
		cout << "\nSame number of passed courses\n";
	return 0;
}
```

Ε.25 *Απάντηση:*

```
#include <iostream>
#include <string>
#include <vector>
using std::cout;
using std::string;
using std::vector;

template <typename T> void sort(vector<T>& v);
```

```
class Product
{
	private:
		int code;
		float prc;
	public:
		Product(int c = 0, float p = 0) {code = c; prc = p;}
		bool operator>(const Product& p) const;
		void show() const {cout << "C:" << code << " P:" << prc <<
'\n';}
};

bool Product::operator>(const Product& p) const
{
	if(prc > p.prc)
		return true;
	return false;
}

class Employee
{
	private:
		string name;
		float sal;
	public:
		Employee(const string& n = "", float s = 0) {name = n; sal
= s;}
		bool operator>(const Employee& e) const;
		void show() const {cout << "N:" << name << " S:" << sal <<
'\n';}
};

bool Employee::operator>(const Employee& e) const
{
	if(name > e.name)
		return true;
	return false;
}

int main()
{
	int i;
	vector<Product> v_prd = {Product(10, 3.5), Product(20, 1.5),
Product(30, 2.5)};
	vector<Employee> v_emp = {Employee("Mike", 100),
Employee("John", 200), Employee("Nick", 300)};

	sort(v_prd);
	sort(v_emp);
	for(i = 0; i < v_prd.size(); i++)
		v_prd[i].show();
	for(i = 0; i < v_emp.size(); i++)
		v_emp[i].show();
	return 0;
}
```

```
template <typename T> void sort(vector<T>& v)
{
	int i, j, size;
	T tmp; // Θα κληθεί ο κατασκευαστής της κάθε κλάσης.

	size = v.size();
	for(i = 0; i < size-1; i++)
	{
		for(j = i+1; j < size; j++)
		{
			if(v[i] > v[j]) /* Για την σύγκριση των στοιχείων
καλείται η συνάρτηση υπερφόρτωσης του > της κάθε κλάσης. */
			{
				tmp = v[i];
				v[i] = v[j];
				v[j] = tmp;
			}
		}
	}
}
```

Ε.26 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

class T
{
private:
	string str;
	int key;
public:
	void show() const {cout << str << ' ' << key;}
	T() {}
	T(const T& t);
	int& operator[](string s);
};

T::T(const T& t)
{
	cout << "Yes\n";
	str = t.str;
	key = t.key;
}

int& T::operator[](string s)
{
	str = s;
	return key;
}
```

Σχόλια: Αφού δεν επιτρέπεται να αλλάξουμε το σώμα των συναρτήσεων, καταλαβαίνουμε ότι πρέπει να προστεθεί ο κατασκευαστής αντιγράφου, ο οποίος θα κληθεί με την κλήση της `f()`.

Ε.27 *Απάντηση:*

```
#include <iostream>
#include <cstring>
#include <cstdlib>
using std::cout;
using std::ostream;

class T
{
private:
	char *s;
public:
	explicit T(const char str[]);
	T() {s = nullptr;}
	~T();
	T(const T& t);
	T operator+(const T& t);
	T& operator=(const T& t);
	char& operator[](int i);
};

T::T(const char str[])
{
	s = new char[strlen(str)+1];
	strcpy(s, str);
}

T::~T()
{
	delete[] s;
}

T T::operator+(const T& t) /* Υπερφόρτωση του τελεστή +. Θα κληθεί με την εντολή t1+t2; Το s είναι το πεδίο του t1, ενώ το t αναφέρεται στο t2. */
{
	T tmp;

	tmp.s = new char[strlen(s)+strlen(t.s)+1];
	tmp.s[0] = '\0';
	strcpy(tmp.s, s);
	strcat(tmp.s, t.s);
	return tmp;
}

T& T::operator=(const T& t) /* Υπερφόρτωση του τελεστή =. Πρώτα Θα κληθεί με την εντολή t3 = t1+t2; Το s είναι το πεδίο του t3, ενώ το t αναφέρεται στο αντικείμενο που επιστρέφει η συνάρτηση operator+. */
{
```

```
        if(this == &t) /* Έλεγχος αυτοανάθεσης για τη γενική περίπτωση,
για το συγκεκριμένο πρόγραμμα δεν χρειάζεται. */
                return *this;

        delete[] s;
        s = new char[strlen(t.s)+1];
        strcpy(s, t.s);
        return *this;
}

T::T(const T& t) /* Ο κατασκευαστής αντιγράφου θα κληθεί με την εντολή
T t5(t4); Το s είναι το πεδίο του t5, ενώ το t αναφέρεται στο t4. */
{
        s = new char[strlen(t.s)+1];
        strcpy(s, t.s);
}

char& T::operator[](int i)
{
        if(i < 0 || i >= strlen(s)) /* Έλεγχος ορίων για τη γενική
περίπτωση, για το συγκεκριμένο πρόγραμμα δεν χρειάζεται. */
                exit(EXIT_FAILURE);
        return s[i];
}
```

Σχόλια: Ας δούμε με ποια λογική προστέθηκαν οι παραπάνω συναρτήσεις. Αφού το πεδίο s είναι δείκτης, ο κατασκευαστής πρέπει να δεσμεύσει μνήμη για την αντιγραφή του αλφαριθμητικού. Άρα, πρέπει να προστεθεί αποδομητής για την αποδέσμευση της μνήμης. Για να επιτρέπεται η πρόσθεση δύο `T` αντικειμένων, πρέπει να οριστεί συνάρτηση υπερφόρτωσης του τελεστή +. Η συνάρτηση επιστρέφει αντικείμενο, ώστε να μπορεί να εκχωρηθεί η τιμή επιστροφής της σε ένα άλλο αντικείμενο. Επειδή η κλάση `T` περιέχει δείκτη πρέπει να προσθέσουμε συνάρτηση υπερφόρτωσης του τελεστή = ώστε να επιτευχθεί βαθιά αντιγραφή. Για τον ίδιο λόγο προσθέτουμε και τον κατασκευαστή αντιγράφου για να κληθεί όταν δημιουργείται το `t4`. Για να είναι έγκυρη η εντολή `t5[2] = 'x';` πρέπει να υπερφορτωθεί ο τελεστής `[]` και για να μπορεί να αλλάξει η τιμή του στοιχείου σε αυτή τη θέση πρέπει να επιστρέφεται αναφορά στο στοιχείο.

Ε.28 *Απάντηση:*

Το πρώτο λάθος είναι στον ορισμό της `show()`. Αφού το `p` είναι ιδιωτικό μέλος, η `show()` δεν έχει άμεση πρόσβαση σε αυτό. Το δεύτερο λάθος είναι στη δημιουργία του `b`. Αφού ο κατασκευαστής της `B` δεν καλεί κατασκευαστή της `A` δεν μπορεί να δημιουργηθεί το `A` αντικείμενο. Για παράδειγμα, αν γράφαμε `B(int v):A(v)` ή αν είχαμε ορίσει τον εξ'ορισμού κατασκευαστή στην `A` δεν θα υπήρχε πρόβλημα. Το `r` μπορεί να αναφερθεί σε `B` αντικείμενο, αλλά επειδή έχει δηλωθεί σαν αναφορά στην κλάση `A` και η `A` δεν περιέχει `show()` είναι λάθος να την καλέσουμε.

Ε.29 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

class A
{
private:
	string name;
public:
	A(const string& s) : name(s) {}
	virtual void show() const {cout << name << '\n';}
};

class B : public A
{
private:
	string name;
public:
	B(const string& s1, const string& s2) : A(s2), name(s1) {}
	virtual void show() const {cout << name << '\n';}
};

class C : public B
{
private:
	string name;
public:
	C(const string& s1, const string& s2, const string& s3) : B(s2,
s3), name(s1) {}
	virtual void show() const {cout << name << '\n';}
};
```

Σχόλια: Ο κατασκευαστής της παράγωγης κλάσης καλεί τον κατασκευαστή της βασικής του. Αφού υπάρχει ο περιορισμός να έχει άδειο σώμα, η αρχικοποίηση του κάθε πεδίου name γίνεται στη λίστα αρχικοποίησης. Μετά, αφού βλέπουμε ότι ενώ ο τύπος του p είναι δείκτης στην A καλείται η show() της C, καταλαβαίνουμε ότι η show() που θα ορίσουμε πρέπει να είναι εικονική.

E.30 ***Απάντηση:***

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

class Product
{
protected:
	int code;
public:
	Product(int c);
```

```
        virtual  ~Product()  {};  /* Δεν χρειάζεται, απλά για να
συνηθίζετε. */
        virtual void show() const;
};

class Book : public Product
{
private:
        string title;
public:
        Book(string t, int c);
        virtual void show() const override;
};

void f(const Product *p1, const Product *p2);

Product::Product(int c)
{
        code = c;
}

void Product::show() const
{
        cout << "C:" << code << '\n';
}

Book::Book(string t, int c) : Product(c)
{
        title = t;
}

void Book::show() const
{
        cout << "C:" << code << " T:" << title << '\n'; /* Αφού το code
είναι προστατευμένο μέλος μπορούμε να το προσπελάσουμε. */
}

int main()
{
        Product prod(100);
        Book b("C++", 200);

        Product *p1 = &prod;
        Product *p2 = &b;
        f(p1, p2);
        return 0;
}

void f(const Product *p1, const Product *p2)
{
        p1->show();
        p2->show();
}
```

Σχόλια: Αφού η άσκηση μας περιορίζει ότι οι δύο δείκτες στην `f()` πρέπει να είναι του ίδιου τύπου και επίσης βλέπουμε ότι οι δύο κλήσεις στην `show()` στην `f()` εμφανίζουν τα μέλη των `Product` και `Book` αντικειμένων καταλαβαίνουμε ότι η `show()` πρέπει να δηλωθεί εικονική. Ο τύπος θα πρέπει να είναι δείκτης στη βασική κλάση, ώστε να μπορεί να μεταβιβαστεί δείκτης στην παράγωγη κλάση (π.χ. `p2`) και να λειτουργήσει ο μηχανισμός των εικονικών συναρτήσεων. Αν η `show()` δεν είχε δηλωθεί σαν εικονική, το πρόγραμμα θα καλούσε δύο φορές την `show()` της κλάσης `Product` και θα εμφάνιζε `100` και `200`.

E.31 *Απάντηση:*

Όταν καλείται η `t1()` μόνο το `A` τμήμα του αντικειμένου `b` θα μεταβιβαστεί στη συνάρτηση και θα αντιγραφεί στο `a`. Άρα, η `f()` θα εμφανίσει `1` και μετά το πρόγραμμα θα εμφανίσει `A.v()`. Για να το ξεκαθαρίσω, το πεδίο `k` της `A` δεν έχει καμία σχέση με το `k` της `B`. Όταν καλείται η `t2()`, το `r` αναφέρεται στο `b` που είναι αντικείμενο της παράγωγης κλάσης. Επειδή η `f()` δεν είναι εικονική θα κληθεί η `A.f()` και το πρόγραμμα θα εμφανίσει `1`. Αντίθετα, επειδή η `v()` είναι εικονική θα κληθεί η `B.v()`. Τελικά, το πρόγραμμα εμφανίζει: `1 A.v() 1 B.v()`

E.32 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

class Fruit
{
private:
	string name;
	int cal;
public:
	Fruit(const string& s, int c) : name(s), cal(c) {}
	virtual ~Fruit() {};
	virtual int operator+(int num);
};

class Apple : public Fruit
{
private:
	int antiox;
public:
	Apple(int ant, const string& name, int cal) : Fruit(name, cal),
antiox(ant) {}
	virtual int operator+(int num) override;
};

int Fruit::operator+(int num)
{
```

```
        cal += num;
        return cal;
}

void f(Fruit& p, int num);

int Apple::operator+(int num)
{
        antiox += num;
        return antiox;
}

int main()
{
        Fruit fr("F1", 10);
        Apple ap(20, "F2", 30);

        f(fr, 5); // Το πρόγραμμα εμφανίζει 15.
        f(ap, 5); // Το πρόγραμμα εμφανίζει 25.
        return 0;
}

void f(Fruit& p, int num)
{
        cout << p+num << '\n';
}
```

Σχόλια: Αφού βλέπουμε στην `f()` ότι υπάρχει πρόσθεση αντικειμένου με ακέραιο καταλαβαίνουμε ότι ο τελεστής + πρέπει να υπερφορτωθεί. Για να μπορεί να καλείται η **`operator`**`+()` ανάλογα με τον τύπο του αντικειμένου που αναφέρεται η `p`, η συνάρτηση πρέπει να δηλωθεί σαν εικονική στη βασική κλάση. Αν δεν είχε δηλωθεί εικονική, το πρόγραμμα θα καλούσε την **`operator`**`+()` της κλάσης `Fruit` στην κλήση `f(ap, 5)`. Ο τύπος της αναφοράς στην `f()` είναι στη βασική κλάση, ώστε να μπορεί να μεταβιβαστεί αναφορά σε παράγωγη κλάση (π.χ. `ap`) και να λειτουργήσει ο μηχανισμός των εικονικών συναρτήσεων.

Ε.33 *Απάντηση:*

Το πρώτο λάθος είναι όταν δημιουργείται το `c`. Το λάθος είναι ότι αφού δεν ορίζεται εξ'ορισμού κατασκευαστής στην `A` δεν μπορεί να κατασκευαστεί το `A` υπο-αντικείμενο. Αν καλούσαμε τον υπάρχοντα κατασκευαστή δεν θα υπήρχε πρόβλημα. Για παράδειγμα, `C() : A(5), v(10) {}`. Το δεύτερο λάθος είναι η πρόσβαση στο `p`. Αφού και οι δύο κλάσεις `A` και `B` περιέχουν μέλος με το ίδιο όνομα `p`, ο μεταγλωττιστής δεν γνωρίζει ποιο από τα δύο να προσπελάσει. Για παράδειγμα, αν γράφαμε `c.A::p = 10;` δεν θα υπήρχε πρόβλημα. Το τρίτο λάθος είναι η κλήση της `show()`. Η `show()` της κλάσης `C` κρύβει την `show()` της κλάσης `B`. Άρα, αν θέλουμε να καλέσουμε τη `show()` της κλάσης `C` πρέπει να γράψουμε `c.show()`, ενώ για τη `show()` της κλάσης `B` πρέπει να γράψουμε `c.B::show("test");`

Ε.34 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::string;

class Person
{
private:
	string name;
	int code;
public:
	Person(const string& n = "None", int c = -1) : name(n), code(c)
{}
	virtual ~Person() {}
	virtual float& operator[](int i) = 0;
};

class Student : public Person
{
private:
	float grd[5];
public:
	virtual float& operator[](int i);
};

class Employee : public Person
{
private:
	float fee[3];
public:
	virtual float& operator[](int i);
};

float& Student::operator[](int i)
{
	return grd[i];
}

float& Employee::operator[](int i)
{
	return fee[i];
}
```

Σχόλια: Εξετάζουμε την `main()` για να δούμε με ποιο τρόπο να υλοποιήσουμε τις κλάσεις. Αρχικά, από τα σχόλια του κώδικα στη δημιουργία των αντικειμένων καταλαβαίνουμε ότι πρέπει να ορίσουμε έναν εξ'ορισμού κατασκευαστή στην `Person` με προκαθορισμένες τιμές. Μετά, βλέπουμε ότι αφού ο `p` δείχνει στο `s`, το `*p` αντιστοιχεί στο `s`. Επομένως, η εντολή `(*p)[3] = 5;` είναι ισοδύναμη με `s[3] = 5;`. Για να επιτραπεί αυτή η εντολή συμπεραίνουμε ότι η κλάση `Student` πρέπει να υπερφορτώνει τον τελεστή `[]`. Επίσης, η συνάρτηση υπερφόρτωσης **`operator`**`[]` πρέπει να δέχεται σαν όρισμα μία ακέραια παράμετρο και να επιστρέφει αναφορά στο αντίστοιχο στοιχείο του πίνακα, ώστε να μπορεί να τροποποιηθεί η τιμή του.

Παρόμοια, με την εντολή `(*p)[1] = 100;` που είναι ισοδύναμη με `e[1] = 100;` συμπεραίνουμε ότι ο τελεστής `[]` πρέπει να υπερφορτωθεί και στην κλάση `Employee`. Για να μπορεί να καλείται η **`operator[]`** ανάλογα με τον τύπο του αντικειμένου που δείχνει ο δείκτης `p`, η συνάρτηση πρέπει να δηλωθεί σαν εικονική στη βασική κλάση. Επειδή η εκφώνηση μας υποχρεώνει η βασική κλάση να είναι αφηρημένη δηλώνουμε τη συνάρτηση σαν γνήσια εικονική.

Ε.35 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

class Shape
{
private:
	float cntr_x;
	float cntr_y;
public:
	Shape(float x, float y) : cntr_x(x), cntr_y(y) {}
	virtual ~Shape() {}
	virtual void show() const {cout << "Cx:" << cntr_x << ' ' <<
"Cy:" << cntr_y << '\n';}
	virtual void area() const = 0;
};

class Shape2D : public Shape
{
private:
	string name;
public:
	Shape2D(float x, float y, const string& n) : Shape(x, y),
name(n) {}
};

class Rect : public Shape2D
{
private:
	float len;
	float hght;
public:
	Rect(float x, float y, const string& n, float l, float h) :
Shape2D(x, y, n), len(l), hght(h) {}
	virtual void area() const {cout << "\nArea(R):" << len*hght <<
'\n';}
	virtual void show() const;
};

void Rect::show() const
{
	area();
	Shape::show();
```

```
}

class Shape3D : public Shape
{
private:
	string col;
public:
	Shape3D(float x, float y, const string& n) : Shape(x, y), col(n)
{}
	virtual void vol() const = 0;
};

class Sphere : public Shape3D
{
private:
	float rad;
public:
	Sphere(float x, float y, const string& n, float r) : Shape3D(x,
y, n), rad(r) {}
	virtual void area() const {cout << "Area(S):" << 4*3.14*rad*rad
<< '\n';}
	virtual void vol() const {cout << "Vol(S):" <<
(4/3.0)*3.14*rad*rad*rad << '\n';}
	virtual void show() const;
};

void Sphere::show() const
{
	area();
	vol();
	Shape::show();
}
```

Σχόλια: Αφού η εκφώνηση μας λέει ότι οι Rect και Sphere δεν είναι αφηρημένες καταλαβαίνουμε ότι πρέπει να υλοποιούν τις γνήσιες εικονικές συναρτήσεις που δηλώνονται στις βασικές κλάσεις τους. Αφού ο p είναι πίνακας δεικτών στην Shape και πρέπει να εμφανιστεί πληροφορία που περιέχεται σε παράγωγες κλάσεις καταλαβαίνουμε ότι η show() που θα ορίσουμε στην Shape πρέπει να είναι εικονική. Για να εμφανιστεί η πληροφορία, ορίζουμε show() στις Rect και Sphere και στο σώμα τους καλούμε την show() της Shape.

Ε.36 *Απάντηση:*

```
#include <iostream>
#include <string>
using std::cout;
using std::cin;
using std::string;

class Err_Rpt
{
public:
	string msg;
```

```
        Err_Rpt(const char *m) : msg(m) {}
};

class Account
{
private:
        string name;
        double bal;
public:
        void create_acnt(const string& n, double amnt);
        void deposit(double amnt);
        void withdraw(double amnt);
        void show() const {cout << "N:" << name << ' ' << "B:" << bal <<
'\n';}
};

void Account::create_acnt(const string& n, double amnt)
{
        int i;

        for(i = 0; i < n.size(); i++)
        {
                if((n[i] >= 'a' && n[i] <= 'z') ||
                        (n[i] >= 'A' && n[i] <= 'Z'))
                        continue;
                else
                        throw  Err_Rpt("Error:  Name  must  contain  letters
only\n");
        }
        if(amnt < 0)
                throw Err_Rpt("Error: Not acceptable amount\n");
        name = n;
        bal = amnt;
}

void Account::deposit(double amnt)
{
        if(amnt < 0)
                throw    Err_Rpt("Error:    Not    acceptable    amount    to
deposit\n");
        bal += amnt;
}

void Account::withdraw(double amnt)
{
        if(amnt < 0 || amnt > bal)
                throw    Err_Rpt("Error:    Not    acceptable    amount    to
withdraw\n");
        bal -= amnt;
}

int main()
{
        double amnt;
        string name;
        Account acnt;
```

```
        while(1)
        {
                try
                {
                        cout << "Enter initial amount: ";
                        cin >> amnt;
                        cin.get();

                        cout << "Enter name: ";
                        getline(cin, name);
                        acnt.create_acnt(name, amnt);
                        break;
                }
                catch(const Err_Rpt& err)
                {
                        cout << err.msg;
                }
        }
        try
        {
                cout << "Enter amount to deposit: ";
                cin >> amnt;
                acnt.deposit(amnt);

                cout << "Enter amount to withdraw: ";
                cin >> amnt;
                acnt.withdraw(amnt);
        }
        catch(const Err_Rpt& err)
        {
                cout << err.msg;
                return 0; // Τερματισμός προγράμματος.
        }
        acnt.show();
        return 0;
}
```

E.37 ***Απάντηση:***

Ναι, υπάρχουν λάθη. Το πρώτο λάθος είναι ο ορισμός της `f()`. Δεν υπάρχει κλάση `B`. Ο προγραμματιστής πρέπει να καθορίσει την έκδοση της κλάσης στην οποία αναφέρεται η `r` (π.χ. `B<A*>`). Το δεύτερο λάθος είναι στη δήλωση του `b`. Αφού ο τύπος `T` αντικαθίσταται με `A*`, η έκδοση της κλάσης είναι:

```
class B<A*>
{
private:
        A* x;
public:
        int k;
        B(A* y) : x(y), k(2) {x.p();}
};
```

Ας δούμε τον κατασκευαστή. Η αρχικοποίηση `x(y)` είναι σωστή, το `x` θα γίνει ίσο με `y`, δηλαδή `x = &a;` Όμως, αφού το `x` είναι δείκτης θα έπρεπε να γράψουμε `x->p()` και όχι `x.p()`.

Ε.38 *Απάντηση:*

```
#include <iostream>
using std::cout;
using std::ostream;

template <typename T> class Circle
{
private:
	T rad;
public:
	Circle<T>(T r = 0) {rad = r;}
	void operator+(T i) {rad += i;}
	Circle<T> operator+(const Circle<T>& c) const;
	template <typename V> friend ostream& operator<<(ostream& out,
const Circle<V>& c);
};

template <typename T> Circle<T> Circle<T>::operator+(const Circle<T>&
c) const
{
	Circle<T> tmp;
	tmp.rad = rad + c.rad;
	return tmp;
}

template <typename V> ostream& operator<<(ostream& out, const
Circle<V>& c)
{
	out << c.rad << '\n';
	return out;
}
```

Σχόλια: Από την `main()` καταλαβαίνουμε ότι πρέπει να ορίσουμε συναρτήσεις υπερφόρτωσης τελεστών. Επίσης, ορίζουμε μία πρότυπη φιλική συνάρτηση που να μπορεί να χρησιμοποιηθεί από όλες τις εκδόσεις της κλάσης.

Ε.39 *Απάντηση:*

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

template <typename T1, typename T2> class Map_Key
{
private:
	T1 key;
	T2 val;
public:
```

```
        Map_Key() {};
        Map_Key(const T1& k, const T2& v) : key(k), val(v) {};
        bool check(const T1& k) const {return (key == k) ? true :
false;}
        T2 get() const {return val;}
        Map_Key<T1, T2> operator+(const Map_Key& mk) const;
};

template <typename T1, typename T2>
Map_Key<T1, T2> Map_Key<T1, T2>::operator+(const Map_Key& mk) const
{
        Map_Key<T1, T2> tmp;
        tmp.key = key + mk.key;
        tmp.val = val + mk.val;
        return tmp;
}

int main()
{
        int i;
        string s;
        Map_Key<string, int> mk_1("One", 1), mk_2("Two", 2);
        Map_Key<string, int> mk_3 = mk_1 + mk_2;
        vector<Map_Key<string, int>> v;

        v.push_back(mk_1);
        v.push_back(mk_2);
        v.push_back(mk_3);
        cout << "Enter string: ";
        cin >> s;
        for(i = 0; i < v.size(); i++)
        {
                if(v[i].check(s))
                {
                        cout << "Value:" << v[i].get() << '\n';
                        return 0;
                }
        }
        cout << "Not found\n";
        return 0;
}
```

Σχόλια: Δοκιμάσαμε την λειτουργία του προγράμματος με ζευγάρια τύπων `string` και **`int`**. Δοκιμάστε το και με άλλους τύπους.

Ε.40 *Απάντηση:*

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <vector>
using namespace std;

class Country
```

```
{
private:
	string name;
	string capital;
	int pop;
public:
	Country(const string& n = "", const string& c = "", int p = 0) :
name(n), capital(c), pop(p) {}
	void set(const string& n, const string& c, int p);
	int get_pop() const {return pop;}
	void show() const;
};

void Country::set(const string& n, const string& c, int p)
{
	name = n;
	capital = c;
	pop = p;
}

void Country::show() const
{
	cout << name << '\t' << capital << '\t' << pop << '\n';
}

int main()
{
	int i, pop;
	string name, capital;
	Country tmp;
	vector<Country> cntr;

	ifstream fin("country.txt");
	if(fin.is_open() == false)
	{
		cout << "Error: is_open() failed\n";
		exit(EXIT_FAILURE);
	}
	while(1)
	{
		fin >> name >> capital >> pop;
		if(!fin)
			break;
		tmp.set(name, capital, pop);
		cntr.push_back(tmp); // Αποθήκευση της χώρας.
	}
	cout << "Enter population: ";
	cin >> pop;
	for(i = 0; i < cntr.size(); i++)
		if(cntr[i].get_pop() >= pop)
			cntr[i].show();

	fin.close();
	return 0;
}
```

Ε.41 *Απάντηση:*

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

int main()
{
	int i, grd_num;
	float *grd_arr, grd;

	ifstream fin("test.bin", ios_base::in | ios_base::binary);
	if(fin.is_open() == false)
	{
		cout << "Error: is_open() failed\n";
		exit(EXIT_FAILURE);
	}
	fin.read((char*)&grd_num, sizeof(int));
	if(!fin)
	{
		fin.close();
		cout << "Error: read() failed\n";
		exit(EXIT_FAILURE);
	}
	grd_arr = new float[grd_num]; /* Δέσμευση μνήμης για την
αποθήκευση των βαθμών. */
	fin.read((char*)grd_arr, grd_num * sizeof(float)); /* Διάβασμα
των βαθμών και έλεγχος ότι δεν συνέβη κάποιο λάθος. */
	if(fin)
	{
		cout << "Enter grade: ";
		cin >> grd;
		for(i = 0; i < grd_num; i++)
			if(grd_arr[i] > grd)
				cout << grd_arr[i] << '\n';
	}
	else
		cout << "Error: read() failed to read grades\n";

	delete[] grd_arr;
	fin.close();
	return 0;
}
```