Αντικειμενοστρεφής Προγραμματισμός

Εργαστήριο 2

Κατασκευή και καταστροφή αντικειμένων

Όταν δημιουργούμε ένα αντικείμενο είναι πολλές φορές απαραίτητο να δώσουμε κάποιες αρχικές τιμές στις μεταβλητές του. Ένας τρόπος για να γίνει αυτό είναι δημιουργώντας μια συνάρτηση που θα δώσει τιμές σε όλες τις παραμέτρους του αντικειμένου. Παράδειγμα μιας τέτοιας συνάρτησης είναι η συνάρτηση set_values της κλάσης Movie στην 3η άσκηση του 1ου Εργαστηρίου.

Ο πιο σωστός όμως τρόπος είναι η αρχικοποίηση του αντικειμένου μέσω μιας ειδικής συνάρτησης που ονομάζεται **κατασκευαστής ή συνάρτηση αρχικών συνθηκών (construction)**. Ο κατασκευαστής αποτελεί μέλος της κλάσης που ορίζει το αντικείμενο, έχει *το ίδιο όνομα με την κλάση* και εκτελείται αυτόματα με τη διακήρυξη του αντικειμένου. Δεν έχει τύπο επιστρεφόμενης τιμής και μπορεί να υπερφορτωθεί.

Σε αντίθεση με τις συναρτήσεις αρχικών συνθηκών οι συναρτήσεις **τελικών συνθηκών, καταστροφείς (destructors)**, εκτελούνται με την καταστροφή του αντικειμένου. Με το όρο «καταστροφή» εννοούμε τη στιγμή κατά την οποία η μνήμη που δεσμεύτηκε για την καταχώρηση των μεταβλητών του αντικειμένου αποδεσμεύεται από το πρόγραμμα (π.χ. με την έξοδο από μια συνάρτηση όπου το αντικείμενο ορίστηκε ως τοπική μεταβλητή). Οι καταστροφείς έχουν και αυτοί το ίδιο όνομα με την κλάση, ενώ πριν από το όνομα τίθεται ο χαρακτήρας ~. Ένα παράδειγμα χρήσης ενός καταστροφέα είναι όταν ένα αντικείμενο θα πρέπει να απελευθερώσει τη μνήμη που προηγούμενα είχε δεσμεύσει με μια συνάρτηση δυναμικής κατανομής μνήμης.

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

class Student
{
public:
	string name;
	int code;

	Student();
	Student(string _name, int _code);
	Student(const Student&);
	~Student();
	void display();
};

Student::Student()
```

```
{
	name = "N/A"; //not available
	code = 0;
}

Student::Student(string _name, int _code)
{
	name = _name;
	code = _code;
}

//copy constructor
Student::Student(const Student& copy)
{
	name = copy.name;
	code = copy.code;
}

//destructor
Student::~Student()
{
	cout << "destructor called" << endl;
}

void Student::display()
{
	cout << "Name: " << name << ", code: " << code << endl;
}

int main()
{
	Student s1;
	Student s2("John Doe", 2323);
	Student s3(s2); //copy constructor invoked

	s3.name = "Bruce Lee";
	Student s4 = s3; //copy constructor invoked

	s1.display();
	s2.display();
	s3.display();
	s4.display();
	system("pause");
	return 0;
}
```

Στο προηγούμενο παράδειγμα υπάρχουν τρεις κατασκευαστές:

- ο default κατασκευαστής που δίνει default αρχικές τιμές σε όλα τα αντικείμενα που δημιουργούνται από αυτόν,
- ο γενικός κατασκευαστής (general constructor) που μπορεί να δώσει την αρχική κατάσταση που θέλουμε εμείς, και
- ο κατασκευαστής αντιγραφής (copy constructor) που δημιουργεί ένα αντικείμενο ως αντίγραφο ενός άλλου.

Επίσης, ορίζεται και ο destructor της κλάσης Student, ο οποίος όμως στην συγκεκριμένη περίπτωση δεν επιτελεί κάποια σημαντική λειτουργία. Ο καταστροφέας μια κλάσης μας είναι απαραίτητος όμως όταν η κλάση περιέχει pointers, ή αν κατά την διάρκεια ζωής του αντικειμένου χρησιμοποιούνται αρχεία ή ανοίγονται συνδέσεις με βάσεις δεδομένων.

Επισημαίνεται τέλος ότι μπορούμε να συμπυκνώσουμε τον γενικό και τον default κατασκευαστή σε έναν, όπως παρακάτω. Αν στην κλήση του κατασκευαστή δεν δοθούν παράμετροι, τότε χρησιμοποιούνται οι default τιμές.

```
Student::Student(string _name = "N/A", int _code=0)
{
	name = _name;
	code = _code;
}
```

Γενικότερα οι βασικοί τύποι των κατασκευαστών που παρέχει η γλώσσα C++ είναι:

Τύπος	Γενική Μορφή
Default constructor	X(void) X(a1=d1, a2=d2, …)
General constructor	X(a1, a2, a3, …)
Copy constructor	X(X&, a2=d2, …) X(const X&, a2=d2, …)

Οι default και copy constructors μας παρέχονται αυτόματα από την C++ και γενικά δεν είναι απαραίτητο να τους υλοποιήσουμε εκτός και αν πρέπει να κάνουν κάτι ιδιαίτερο.

Άσκηση 1:
Να ορίσετε μια κλάση Patient η οποία θα πρέπει να περιγράφει τα στοιχεία ενός ασθενούς σε ένα νοσοκομείο. Τα στοιχεία που μας ενδιαφέρουν είναι τα εξής:

- Ονοματεπώνυμο
- Έτος γέννησης
- Διεύθυνση
- Κλινική
- Δωμάτιο

Να υλοποιήσετε:

1. τους απαραίτητους constructors (γενικό, default και copy)
2. μια συνάρτηση display

και να δημιουργήσετε αντικείμενα μέσα στην main, ώστε να δοκιμάσετε τις παραπάνω συναρτήσεις.

Αντικείμενα μέσα σε Αντικείμενα

Μια κλάση μπορεί να περιλαμβάνει και αντικείμενα εκτός από απλές μεταβλητές. Το έχουμε δει ήδη, καθώς οι κλάσεις μας ως τώρα περιέχουν αντικείμενα της κλάσης string.
Μπορούμε να φτιάξουμε τις δικές μας κλάσεις και να συμπεριλάβουμε αντικείμενά τους σε άλλες κλάσεις.

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

```
class Thesis
{
public:
	string title;
	string supervisor;
	double grade;

	Thesis(string _title, string _supervisor, double _grade)
	{
		title = _title;
		supervisor = _supervisor;
		grade = _grade;
	}

	Thesis()
	{
		title = "N/A";
		supervisor = "N/A";
		grade = 0;
	}

	void display()
	{
		cout << "Title: " << title << ", supervisor: " << supervisor <<
endl;
		cout << "Grade: " << grade << endl;
	}

};

class Student
{
public:
	string name;
	int code;
	Thesis th;

	Student();
	Student(string _name, int _code, Thesis _th);
	void display();
};

Student::Student()
{
	name = "N/A"; //not available
	code = 0;
}

Student::Student(string _name, int _code, Thesis _th)
{
	name = _name;
	code = _code;
	th = _th;
}

void Student::display()
{
	cout << "Name: " << name << ", code: " << code << endl;
	cout << "Thesis info: " << endl;
	th.display();
```

```
}

int main()
{
	Thesis temp_thesis("A study on C++", "Prof. H. Jones", 9.5);
	Student s1;
	Student s2("John Doe", 2323, temp_thesis);

	s1.display();
	s2.display();
	system("pause");
	return 0;
}
```

Άσκηση 2:
Το πεδίο «Διεύθυνση» στην κλάση Patient έχει οριστεί ως string. Θα θέλαμε να έχουμε ξεχωριστά την πληροφορία για την οδό, τον αριθμό, τον ΤΚ, την πόλη και την χώρα του ασθενούς ώστε να μπορούμε να επεξεργαζόμαστε τα δεδομένα και ενδεχομένως να βγάζουμε στατιστικά πιο εύκολα.

Για τον λόγο αυτό θα πρέπει:

- Να φτιάξουμε μια νέα κλάση Address που θα περιέχει αυτά τα στοιχεία. Μην ξεχάσετε και τους constructors.
- Να αντικαταστήσουμε στην κλάση Patient το string της διεύθυνσης με ένα object τύπου Address
- Στην συνάρτηση main να δημιουργήσετε ένα αντικείμενο τύπου Patient

Πίνακας Αντικειμένων

Οι πίνακες της C++, εκτός από απλές μεταβλητές μπορούν να περιέχουν και αντικείμενα, π.χ.

```
Student s[10]; //πίνακας σπουδαστών
```

Αν τοποθετήσετε μηνύματα (μέσω cout) στους κατασκευαστές της κλάσης Student θα παρατηρήσετε ότι κατά την έναρξη του προγράμματος ***καλείται 10 φορές ο default constructor της Student***. Άρα και τα 10 αντικείμενα του πίνακα αρχικοποιούνται αυτόματα.

```
int main()
{
	Student s[10]; //default constructor called 10 times
	s[0].name = "John Doe";
	s[0].code = 2323;
	s[0].th.title = "A study on C++";
	s[0].th.supervisor = "Prof. H. Jones";
	s[0].th.grade = 9.5;

	s[0].display();
	s[1].display(); //prints default info
	s[9].display(); //prints default info
	system("pause");
	return 0;
}
```

Άσκηση 3:
Στην συνάρτηση main να ορίσετε ένα πίνακα που θα περιέχει 10 ασθενείς. Στην συνέχεια να εκτελέσετε ένα loop στο οποίο:

- Ο χρήστης της εφαρμογής θα πληκτρολογεί τα στοιχεία ενός ασθενούς (και την διεύθυνσή του)
- Τα στοιχεία αυτά θα καταγράφονται σε ένα αντικείμενο μέσα στον πίνακα
- Το loop θα τελειώνει όταν ο χρήστης το επιλέξει (π.χ. αν πατήσει '0')
- Τέλος, θα τρέχει ένα άλλο loop στο οποίο θα εμφανίζονται τα στοιχεία όλων των ασθενών που καταγράφηκαν στην λίστα (Θα πρέπει απλά να χρησιμοποιήσετε την συνάρτηση display)

Παράρτημα

Εναλλακτικός τρόπος χρήσης του αντικειμένου Thesis. Εδώ δεν περνάμε ένα αντικείμενο τύπου Thesis ως παράμετρο στον constructor της Student αλλά περνάμε τις παραμέτρους title, supervisor και grade ξεχωριστά.

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

class Thesis
{
public:
	string title;
	string supervisor;
	double grade;

	Thesis(string _title, string _supervisor, double _grade)
	{
		title = _title;
		supervisor = _supervisor;
		grade = _grade;
	}

	Thesis()
	{
		title = "N/A";
		supervisor = "N/A";
		grade = 0;
	}

	void display()
	{
		cout << "Title: " << title << ", supervisor: " << supervisor <<
endl;
		cout << "Grade: " << grade << endl;
	}

};

class Student
{
```

```
public:
	string name;
	int code;
	Thesis th;

	Student();
	Student(string _name, int _code, string _title, string _supervisor, double
_grade);
	void display();
};

Student::Student()
{
	name = "N/A"; //not available
	code = 0;
	th.title = "N/A";
	th.supervisor = "N/A";
	th.grade = 0;
}

Student::Student(string _name, int _code, string _title, string _supervisor,
double _grade)
{
	name = _name;
	code = _code;
	th.title = _title;
	th.supervisor = _supervisor;
	th.grade = _grade;
}

void Student::display()
{
	cout << "Name: " << name << ", code: " << code << endl;
	cout << "Thesis info: " << endl;
	th.display();
}

int main()
{
	Student s1;
	Student s2("John Doe", 2323, "A study on C++", "Prof. H. Jones", 9.5);

	s1.display();
	s2.display();
	system("pause");
	return 0;
}
```