

**Ανοικτά Ακαδημαϊκά Μαθήματα**

**Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας**

**Αντικειμενοστρεφής Προγραμματισμός (Ε)**

**Ενότητα 8:** Templates

Κλειώ Σγουροπούλου

Τμήμα Μηχανικών Πληροφορικής ΤΕ

|  |  |
| --- | --- |
| Το περιεχόμενο του μαθήματος διατίθεται με άδεια Creative Commons εκτός και αν αναφέρεται διαφορετικά | Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους. |

# Πίνακας Περιεχομένων

[Πίνακας Περιεχομένων 2](#_Toc402945298)

[Αντικειμενοστρεφής Προγραμματισμός- Εργαστήριο 8 3](#_Toc402945299)

[Templates 3](#_Toc402945300)

[**Άσκηση 1** 4](#_Toc402945301)

[**Άσκηση 2** 4](file:///%5C%5Cgus%5Copencourses%5C%CE%91_%CE%9C%CE%91%CE%98%CE%97%CE%9C%CE%91%CE%A4%CE%91%5C%CE%A5%CE%9B%CE%99%CE%9A%CE%9F%5C%CE%A3%CE%A4%CE%95%CE%A6%5C%CE%A3%CE%93%CE%9F%CE%A5%CE%A1%CE%9F%CE%A0%CE%9F%CE%A5%CE%9B%CE%9F%CE%A5%20%CE%9A%CE%9B%CE%95%CE%99%CE%A9%5C%CE%91%CE%9D%CE%A4%CE%99%CE%9A%CE%95%CE%99%CE%9C%CE%95%CE%9D%CE%9F%CE%A3%CE%A4%CE%A1%CE%95%CE%A6%CE%97%CE%A3%20%CE%A0%CE%A1%CE%9F%CE%93%CE%A1%CE%91%CE%9C%CE%9C%CE%91%CE%A4%CE%99%CE%A3%CE%9C%CE%9F%CE%A3%20-%20%CE%95%5C%CE%95%CE%9A%CE%A0%CE%91%CE%99%CE%94%CE%95%CE%A5%CE%A4%CE%99%CE%9A%CE%9F%20%CE%A5%CE%9B%CE%99%CE%9A%CE%9F%5C%CE%95%CE%A0%CE%95%CE%9E%5C%CE%93%CE%99%CE%91%20%CE%91%CE%9D%CE%95%CE%92%CE%91%CE%A3%CE%9C%CE%91%5C8_%CE%95%CF%81%CE%B3%CE%B1%CF%83%CF%84%CE%AE%CF%81%CE%B9%CE%BF%28%CE%95%CE%B1%CF%81%CE%B9%CE%BD%CF%8C_2013%29.docx#_Toc402945302)

[Function templates με πολλαπλές παραμέτρους και πολλαπλά template argument 4](#_Toc402945303)

[Class templates 5](#_Toc402945304)

[**Άσκηση 3** 6](#_Toc402945305)

# Αντικειμενοστρεφής Προγραμματισμός- Εργαστήριο 8

## Templates

Τα templates, που διακρίνονται σε function templates και class templates, μας επιτρέπουν να ορίσουμε συναρτήσεις και κλάσεις γενικής χρήσης, τις οποίες θα μπορούμε να χρησιμοποιήσουμε ώστε να χειρίζονται διαφορετικούς τύπους δεδομένων κάθε φορά, με τον ίδιο τρόπο. Για παράδειγμα, μπορούμε να ορίσουμε ένα function template μιας συνάρτησης abs (absolute), το οποίο θα μπορεί να χρησιμοποιηθεί να δώσει την απόλυτη τιμή κάθε είδους μεταβλητής, είτε είναι int, float, κλπ:

#include <iostream> using namespace std;

template <class T> //function template T abs(T n)

{

return (n < 0) ? -n : n;

}

int main()

{

int int1 = 5;

int int2 = -6;

long lon1 = 70000L;

long lon2 = -80000L;

double dub1 = 9.95;

double dub2 = -10.15;

//calls instantiate functions

cout << “\nabs(“ << int1 << ”)=” << abs(int1); //abs(int)

cout << “\nabs(“ << int2 << ”)=” << abs(int2); //abs(int)

cout << “\nabs(“ << lon1 << ”)=” << abs(lon1); //abs(long)

cout << “\nabs(“ << lon2 << ”)=” << abs(lon2); //abs(long)

cout << “\nabs(“ << dub1 << ”)=” << abs(dub1); //abs(double) cout << “\nabs(“ << dub2 << ”)=” << abs(dub2); //abs(double) cout << endl;

return 0;

}

Κάθε φορά που ο compiler συναντά την συνάρτηση abs( ), π.χ. την abs(dub1), δημιουργεί (instantiates) την συνάρτηση που αντιστοιχεί στον τύπο της μεταβλητής που δώσαμε ως παράμετρο (εδώ: double).

### **Άσκηση 1**

Να υλοποιήσετε ένα template συνάρτησης swap(a, b), το οποίο θα αντιμεταθέτει τις τιμές των a και b ως εξής:

int a = 10; int b = 5; swap(……);

cout << a; //να εμφανίζει πλέον 5 και όχι 10

Να υλοποιήσετε την συνάρτηση με χρήση: α) pointers, β) references

### **Άσκηση 2**

Να υλοποιήσετε ένα template συνάρτησης add(a, b) η οποία θα προσθέτει δύο αριθμούς:

int a = 10; int b = 5;

int c = add(a, b);

int c = add(a, b);

Ερώτηση: Μπορείτε να χρησιμοποιήσετε την add( ) για να προσθέσετε δύο αντικείμενα μιας κλάσης (όχι primitive τύπους, όπως int, double κλπ.)? Τι χρειάζεται για να γίνει αυτό;

## Function templates με πολλαπλές παραμέτρους και πολλαπλά template argument

Μια συνάρτηση μπορεί να πάρει πολλαπλές παραμέτρους ως template. Επίσης, μπορεί να δεχτεί ως παράμετρο, δύο ή και περισσότερα templates, όπως φαίνεται παρακάτω:

#include <iostream> using namespace std;

//function returns index number of item, or -1 if not found template <class atype, class btype>

btype find(atype\* array, atype value, btype size)

{

for(btype j=0; j<size; j++) //note use of btype if(array[j]==value)

return j;

return static\_cast<btype>(-1);

|  |  |
| --- | --- |
| } |  |
| char chrArr[] = char ch = 5; int intArr[] = | {1,{1, | 3,3, | 5,5, | 9,9, | 11,11, | 13};13}; | //array//value to find |
| int in = 6; |  |  |  |  |  |  |  |

long lonArr[] = {1L, 3L, 5L, 9L, 11L, 13L};

long lo = 11L;

double dubArr[] = {1.0, 3.0, 5.0, 9.0, 11.0, 13.0};

double db = 4.0;

int main()

{

cout << “\n 5 in chrArray: index=” << find(chrArr, ch, 6);

cout << “\n 6 in intArray: index=” << find(intArr, in, 6);

cout << “\n11 in lonArray: index=” << find(lonArr, lo, 6);

cout << “\n 4 in dubArray: index=” << find(dubArr, db, 6); cout << endl;

return 0;

}

## Class templates

Τα class templates μας επιτρέπουν αντίστοιχα να ορίσουμε μια γενικής χρήσης κλάση η οποία θα μπορεί να χρησιμοποιηθεί ανάλογα με την περίσταση, για διαφορετικά είδη member variables. Κλασσικό παράδειγμα αποτελούν τα class templates για γενικής χρήσης κατασκευές, όπως είναι οι ουρές, οι συνδεδεμένες λίστες, οι στοίβες κ.α., που μπορούν να αποθηκεύσουν κάθε είδους μεταβλητή, είτε πρόκειται για απλή μεταβλητή (int, char, κλπ.) είτε για κλάση ορισμένη από τον χρήστη.

#include <iostream> using namespace std; const int MAX = 100;

template <class Type> class Stack

{

private:

Type st[MAX]; //stack: array of any type

int top; //number of top of stack

public:

Stack(); //constructor

void push(Type var); //put number on stack Type pop(); //take number off stack

};

template<class Type>

Stack<Type>::Stack() //constructor

{

top = -1;

}

template<class Type>

void Stack<Type>::push(Type var) //put number on stack

{

st[++top] = var;

}

template<class Type>

Type Stack<Type>::pop() //take number off stack

{

return st[top--];

}

int main()

{

Stack<float> s1; //s1 is object of class Stack<float>

s1.push(1111.1F); //push 3 floats, pop 3 floats s1.push(2222.2F);

s1.push(3333.3F);

cout << "1: " << s1.pop() << endl;

cout << "2: " << s1.pop() << endl; cout << "3: " << s1.pop() << endl;

Stack<long> s2; //s2 is object of class Stack<long> s2.push(123123123L); //push 3 longs, pop 3 longs

s2.push(234234234L);

s2.push(345345345L);

cout << "1: " << s2.pop() << endl; cout << "2: " << s2.pop() << endl; cout << "3: " << s2.pop() << endl; return 0;

}

### **Άσκηση 3**

Να ορίσετε μια κλάση Point για την αποθήκευση σημείων στον χάρτη (με συντεταγμένες x και y), και

να την υλοποιήσετε ως class template, ώστε να χρησιμοποιούμε integers, doubles κλπ. για τις συντεταγμένες. Να υλοποιήσετε επίσης:

α) τους απαραίτητους constructors της κλάσης β) μια συνάρτηση display

γ) να υπερφορτώσετε τον operator+ για την προσθήκη δύο σημείων (προσθέτουμε απλά τις αντίστοιχες συντεταγμένες)

Στην main( ) να δημιουργήσετε δύο αντικείμενα τύπου Point των οποίων οι συντεταγμένες θα είναι double, να τα προσθέσετε μεταξύ τους και να αποθηκεύσετε το αποτέλεσμα σε τρίτο (μηδενικό) Point.

|  |
| --- |
| **Ανοικτά Ακαδημαϊκά Μαθήματα****Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας** |
| **Τέλος Ενότητας** |
| **Χρηματοδότηση*** Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
* Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Αθήνας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
* Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

 |

**Σημειώματα**

**Σημείωμα Αναφοράς**

Copyright ΤΕΙ Αθήνας, Κλειώ Σγουροπούλου, 2014. Κλειώ Σγουροπούλου. «Αντικειμενοστρεφής Προγραμματισμός». Ενότητα 8: «Templates». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: [ocp.teiath.gr](https://ocp.teiath.gr/).

**Σημείωμα Αδειοδότησης**

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

* που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
* που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
* που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

**Διατήρηση Σημειωμάτων**

* Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
* Το Σημείωμα Αναφοράς
* Το Σημείωμα Αδειοδότησης
* Τη δήλωση Διατήρησης Σημειωμάτων
* Το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους.