



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά
μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 7: C++ TEMPLATES, ΥΠΕΡΦΟΡΤΩΣΗ ΤΕΛΕΣΤΩΝ, ΕΞΑΙΡΕΣΕΙΣ

Templates

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Templates

Εισαγωγή

- Templates
 - Templates Συναρτήσεων
 - Ορισμός μιας σειράς σχετιζόμενων συναρτήσεων (με υπερφόρτωση)
 - Templates Τάξεων
 - Ορισμός μιας σειράς σχετικών κλάσεων



Function Templates

- Υπερφορτωμένες Συναρτήσεις (Overloaded)
 - Παρόμοιες λειτουργίες
 - Διαφορετικός τύπος δεδομένων
- Πρότυπες Συναρτήσεις (templates)
 - Ίδια ακριβώς λειτουργία
 - Διαφορετικός τύπος δεδομένων
 - Δήλωση μιας μόνο συνάρτησης template
 - Ο compiler παράγει ξεχωριστές συναρτήσεις
 - Type checking



Function Templates

- Ορισμός Function template
 - Ορισμός με την λέξη κλειδί **template**
 - Ο τύπος των παραμέτρων δηλώνεται μέσα σε brackets **< >**
 - Πριν από κάθε παράμετρο μπαίνει το: **class** ή **typename** (ισοδύναμα)

```
template< class T >
```

```
template< typename ElementType >
```

```
template< class BorderType, class FillType >
```
 - Μπορούμε να καθορίσουμε τον τύπο σε:
 - Ορίσματα συνάρτησης
 - Τύπος επιστρεφόμενης τιμής
 - Τοπικές Μεταβλητές μέσα στο σώμα της συνάρτησης



Παράδειγμα

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

```
template <class T>
class mypair {
    T a, b;
    public: mypair (T first, T second) {
        a=first;
        b=second;
    } T getmax ();
};
```

```
template <class T> T mypair<T>::getmax () {
    T retval;
    retval = a>b? a : b;
    return retval;
}
```

```
int main () {
    mypair <int> integerpair (100, 75);
    cout << integerpair.getmax();
    mypair <char> charpair ('g', 'c');
    cout << charpair.getmax();
    return 0;
}
```

Η κλάση mypair περιέχει δύο όμοιες μεταβλητές. Ο τύπος των μεταβλητών θα καθορίζεται κάθε φορά κατά την δημιουργία ενός αντικειμένου.

Ο τύπος μπορεί να είναι build in όπως στο παράδειγμα (int, char) αλλά και οποιασδήποτε κλάσης.

ΠΧ:

```
Dog dog1("Max")
```

```
Dog dog2("Wolfy");
```

```
mypair <Dog> charpair (dog1, dog2);
```

Προσοχή: στο συγκεκριμένο παράδειγμα θα πρέπει στην κλάση Dog να έχουμε κάνει υπερφόρτωση του τελεστή σύγκρισης >



Υλοποίηση Στοίβας με Template Κλάση

```
template< class T >
class Stack {
public:
    Stack( int = 10 );
    ~Stack() {
        delete [] stackPtr;
    }
    bool push( const T& );
    bool pop( T& );
    bool isEmpty() const { return top == -1; }
    bool isFull() const { return (top == size - 1); }
}

private:
    int size;
    int top;
    T *stackPtr; };
```

```
template< class T >
Stack< T >::Stack( int s ){
    size = s > 0 ? s : 10;
    top = -1;
    stackPtr = new T[ size ];
}
```

```
template< class T >
bool Stack< T >::push( const T &pushValue ){
    if ( !isFull() ) {
        stackPtr[ ++top ] = pushValue;
        return true;
    }
    return false;
}
```

```
template< class T >
bool Stack< T >::pop( T &popValue ){
    if ( !isEmpty() ) {
        popValue = stackPtr[ top-- ];
        return true;
    }
    return false;
}
```



Templates και static μέλη

- Απλή κλάση (όχι template class)
 - Τα **static** μέλη μοιράζονται από όλα τα αντικείμενα
- Class-template
 - Κάθε τύπος έχει δικά του αντίγραφα των **static** μεταβλητών
 - **static** μεταβλητές αρχικοποιούνται σε εμβέλεια αρχείου
 - Κάθε τύπος έχει δικά του αντίγραφα των **static** μεθόδων



Πρόσθετο Υλικό

- Μελετήστε και τα παραδείγματα από τα **Κεφάλαια 18** του βιβλίου:
«C++ How to Program, 9/e Paul & Harvey Deitel»
http://media.pearsoncmg.com/ph/esm/deitel/cpp_hpt_9/code_examples/Code_Examples.zip



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την 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.



Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1



Σημείωμα Αναφοράς

Copyright: Πανεπιστήμιον Πατρών, Ιωάννης Χατζηλυγερούδης, 2015.
«Οντοκεντρικός Προγραμματισμός». Έκδοση: 1.0.1 Πάτρα 2015. Διαθέσιμο
από τη δικτυακή διεύθυνση:

<https://eclass.upatras.gr/courses/CEID1105/>



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.



Σημείωμα Χρήσης Έργων Τρίτων

- Οι διαφάνειες βασίζονται στο βιβλίο «C++ How to Program, 8th Edition, Harvey M. Deitel, Paul J. Deitel, Prentice Hall.»

