Templates

Εισαγωγή

- Templates
  - Templates Συναρτήσεων
    - Ορισμός μιας σειράς σχετιζόμενων συναρτήσεων (με υπερφόρτωση)
  - Templates Κλάσεων
    - 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 (internal linkage)

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_htp_9/code_examples/Code_Examples.zip