Η ΓΛΩΣΣΑ C++

Μάθημα 13:

Πρότυπα Κλάσεων

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Θεωρία

1. Πρότυπα Κλάσεων

1. Η ανάγκη για πρότυπα κλάσεων

- Έχουμε δει ότι μία κλάση ορίζει μια έννοια.
- Σε πολλές περιπτώσεις όμως αναγκαζόμαστε να έχουμε υλοποιήσεις της ίδιας έννοιας πολλές φορές.
  - Π.χ. μπορούμε να ορίσουμε μία κλάση που ορίζει μία στοίβα ακεραίων
    - Αλλά θα χρειαστεί να ορίσουμε άλλη κλάση για να ορίσουμε μια στοίβα πραγματικών.

- Για τον λόγο αυτό ορίζουμε τα **πρότυπα κλάσεων**
  - Και θα μπορούμε να ορίζουμε π.χ μια κλάση - στοίβα που περιέχει δεδομένα του τύπου T
  - Και έπειτα στο πρόγραμμα μας θα λέμε, ότι θέλουμε ένα αντικείμενο όπου στη θέση του T έχουμε ακέραιο
    - ή πραγματικό
    - ή συμβολοσειρά
    - ή οποιονδήποτε άλλο τύπο δεδομένων

Α. Θεωρία

1. Πρότυπα Κλάσεων

2. Ορισμός Προτύπου

- Ορίζουμε ότι μία κλάση είναι πρότυπο ως εξής:

```
template <typename T>
class class_name {
  …
};
```

  - όπου T είναι ο «γενικός» τύπος δεδομένων που χρησιμοποιεί η κλάση.

- Έπειτα στο σώμα της κλάσης χρησιμοποιούμε το T σαν να ήταν οποιοσδήποτε τύπος δεδομένων, π.χ. μπορούμε να ορίσουμε ένα μέλος που να είναι τύπου δεδομένων T:

```
template <typename T>
class class_name {
 public:
    T var_name;
  …
};
```

Παρατήρηση:

- Μπορούμε να βλέπουμε το <T> σαν παραμετροποίηση της κλάσης, σαν να κατασκευάζουμε μια κλάση που «παίρνει όρισμα» τον τύπο δεδομένων T που θα χρησιμοποιήσεί.
- Αντί για τη λέξη κλειδί typename μέσα στα <> μπορούμε να χρησιμοποιήσουμε και τη λέξη class

Α. Θεωρία

1. Πρότυπα Κλάσεων

2. Ορισμός Προτύπου

- Για να δηλώσουμε μια μέθοδο έξω από την κλάση χρησιμοποιούμε το εξής συντακτικό:

```
template <typename T>
class class_name {
  …
  ret_type method(args);
  …
};

template <typename T>
ret_type class_name<T>::method(args)
{ … }
```

  - δηλαδή το όνομα της κλάσης είναι class_name<T>

Παρατήρηση:

- Ένα template μπορεί να έχει πολλούς παραμετρικούς τύπους δεδομένων.
- πχ. για 2 ΤΔ, απλά επεκτείνουμε στο συντακτικό των ορισμάτων <typename T1, typename T2>

Α. Θεωρία

1. Πρότυπα Κλάσεων

3. Στιγμιότυπο (instance) της πρότυπης κλάσης

- Στο πρόγραμμα μας μπορούμε να χρησιμοποιήσουμε την πρότυπη κλάση για να ορίσουμε αντικείμενα, αρκεί να δώσουμε τιμή στην παράμετρο του τύπου T:
- Π.χ. για να ορίσουμε ένα αντικείμενο της προηγούμενης κλάσης όπου το T είναι ακέραιος, γράφουμε:

```
class_name<int> ob_name;
```

- ενώ για να ορίσουμε ένα αντικείμενο όπου το T είναι πραγματικός, γράφουμε:

```
class_name<double> ob_name;
```

- δηλαδή ενώ ξέραμε:
  - ότι μία κλάση ορίζει έναν τύπο δεδομένων
    - π.χ. η κλάση «σκύλος» ορίζει έναν τύπο δεδομένων
  - ένα πρότυπο ορίζει τύπο δεδομένων, μόνο εφόσον έχουμε αρχικοποιήσει τον τύπο δεδομένων που παίρνει σαν όρισμα:
    - π.χ. μία λίστα πραγματικών ορίζει τον τύπο δεδομένων list<double>
      - ενώ μία λίστα ακεραίων ορίζει τον τύπο δεδομένων list<int>
  - λέμε τότε ότι έχουμε ένα στιγμιότυπο της πρότυπης κλάσης.

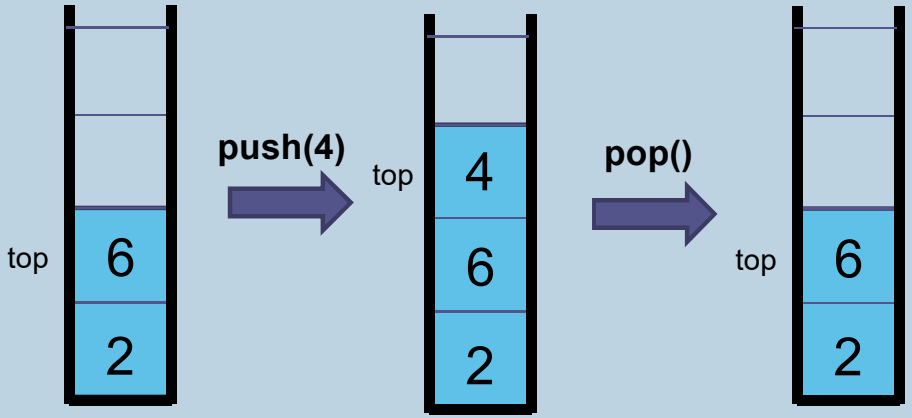
Α. Θεωρία

1. Πρότυπα Κλάσεων

4. Παράδειγμα: «Template Στοίβα»

- Θα κατασκευάσουμε μία στοίβα η οποία να αποθηκεύει στοιχεία οποιουδήποτε τύπου δεδομένων.
  - (βλ. και «Δομές Δεδομένων σε C - Μάθημα 2– Στοίβα»)
- Ξεκινάμε με τη δήλωση της κλάσης ως πρότυπης και των βασικών πράξεών της:

```
#define STACK_SIZE 100

template <typename T>
class stack {
 public:
   stack();
   bool empty();
   bool full();
   bool push(T elem);
   bool pop(T *elem);
 private:
   T array[STACK_SIZE];
   int top;
};
```



- Σκεφτόμαστε ότι δουλεύουμε με μία στοίβα ακεραίων και όπου θα βάζαμε int, βάζουμε T.

Α. Θεωρία

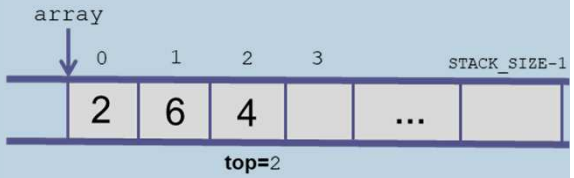
1. Πρότυπα Κλάσεων

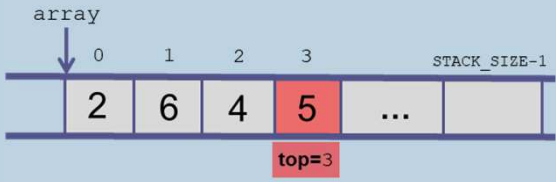
4. Παράδειγμα: «Template Στοίβα»

- Συνεχίζουμε με τις δηλώσεις των μεθόδων:

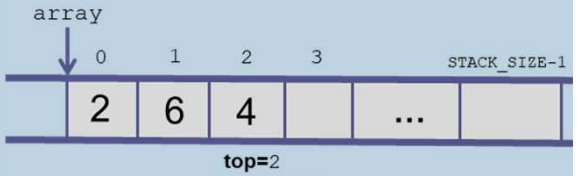
```
template <typename T>
stack<T>::stack()
{
   top=-1;
}

template <typename T>
bool stack<T>::empty()
{
   return top==-1;
}

template <typename T>
bool stack<T>::full()
{
   return top==STACK_SIZE-1;
}
```

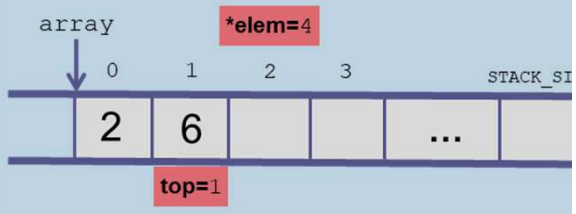
```
template <typename T>
bool stack<T>::push(T elem)
{
   if (full())
      return false;

   top++;
   array[top]=elem;
   return true;
}
```






```
template <typename T>
bool stack<T>::pop(T *elem)
{
   if (empty())
      return false;

   *elem = array[top];
   top--;
   return true;
}
```





Α. Θεωρία

1. Πρότυπα Κλάσεων

4. Παράδειγμα: «Template Στοίβα»

- Και πλέον έχουμε μία στοίβα την οποία μπορούμε να χρησιμοποιήσουμε με οποιονδήποτε τύπο δεδομένων:

```
int main()
{
   /* integer stack */
   stack<int> st;
   int temp;

   st.push(2);
   st.push(5);

   st.pop(&temp);
   cout<<temp<<" ";
   st.pop(&temp);
   cout<<temp<<" ";
```

```
   /* string stack */
   stack<string> st2;
   string temp2;

   st2.push("Hello");
   st2.push("World");

   st2.pop(&temp2);
   cout<<endl<<temp2<<" ";
   st2.pop(&temp2);
   cout<<temp2<<" ";

   return 0;
}
```

Ολοκληρωμένο το πρόγραμμα είναι το: «cpp13.template.cpp»

Α. Θεωρία

1. Πρότυπα Κλάσεων

5. Παράμετροι που δεν είναι τύποι σε πρότυπα κλάσεων

- Μπορούμε να ορίσουμε και παραμέτρους που δεν είναι τύποι, σε πρότυπα κλάσεων αν αυτό είναι χρήσιμο.
- Π.χ. στην κλάση στοίβα μπορούμε να προσθέσουμε μια ακέραια παράμετρο που να ορίζει το μέγεθος του πίνακα που χρησιμοποιείται:

```
template <typename T, int STACK_SIZE>
class stack {
  public:
    stack();
    bool empty();
    bool full();
    bool push(T elem);
    bool pop(T *elem);
  private:
    T array[STACK_SIZE];
    int top;
};
```

Α. Θεωρία

1. Πρότυπα Κλάσεων

5. Παράμετροι που δεν είναι τύποι σε πρότυπα κλάσεων

- ενώ ο ορισμός των μεθόδων θα ενσωματώνει (όπως είναι αναμενόμενο) όλες τις παραμέτρους:
- π.χ.:

```
template <typename T, int STACK_SIZE>
bool stack<T, STACK_SIZE>::empty()
{
   return top==-1;
}
```

Ολοκληρωμένο το πρόγραμμα είναι το: «cpp13.template_non_type_param.cpp»

Α. Θεωρία

1. Πρότυπα Κλάσεων

6. Ποιος είναι ο τύπος δεδομένων;

- Προσοχή!
  - Ξέρουμε ότι μία κλάση ορίζει έναν τύπο δεδομένων
    - με τον οποίο μπορούμε να κατασκευάσουμε αντικείμενα, να ορίσουμε δείκτες που να δείχνουν σε αντικείμενα του τύπου, κ.λπ.
  - Μία κλάση – πρότυπο δεν ορίζει κάποιον τύπο δεδομένων.
  - Η κλάση κατασκευάζεται όταν ορίζονται οι παραμέτροι της κατά το χρόνο μεταγλώττισης.
- Έτσι π.χ. δεν μπορούμε να ορίσουμε κάτι του τύπου:

```
stack *p;
```

  - (δεν υπάρχει τύπος δεδομένων stack)

- Μπορούμε να ορίσουμε μόνο αν συνοδεύεται από τα ορίσματα που έχουμε θέσει στις παραμέτρους του προτύπου
- Π.χ. για την πρώτη στοίβα μπορούμε να γράψουμε:

```
stack<int> *p;
```

- ή π.χ. να κατασκευάσουμε μια συνάρτηση με ορίσμα τον νέο τύπο δεδομένων:

```
void fun(stack<double> &ob) { … }
```

Α. Θεωρία

1. Πρότυπα Κλάσεων

7. Στατικά Μέλη σε πρότυπες κλάσεις

- Εφόσον δεν υπάρχει στην ουσία πρότυπη κλάση
  - αλλά πρότυπη κλάση της οποίας έχουν οριστεί οι παραμέτροι
- Αν ορίσουμε κάποιο στατικό μέλος:
  - Αυτό θα είναι ξεχωριστό για κάθε τύπο που θα ορίσουμε στο πρότυπο.

- Π.χ. αν στην πρότυπη κλάση «στοίβα»
  - Χρησιμοποιούμε στο πρόγραμμά μας στοίβες ακεραίων και στοίβες πραγματικών
    - τότε οι στοίβες ακεραίων θα έχουν το δικό τους στατικό μέλος
    - και οι στοίβες πραγματικών θα έχουν το δικό τους στατικό μέλος

Α. Θεωρία

1. Πρότυπα Κλάσεων

7. Στατικά Μέλη σε πρότυπες κλάσεις

- Επεκτείνουμε το αρχικό πρόγραμμα ώστε να περιλαμβάνει και ένα στατικό μέλος:

```
template <typename T>
class stack {
 public:
   static int ob_cnt;
   stack();
   bool empty();
   bool full();
   bool push(T elem);
   bool pop(T *elem);
 private:
   T array[STACK_SIZE];
   int top;
};

template <typename T>
int stack<T>::ob_cnt = 0;
```

```
int main()
{
   /* integer stack */
   stack<int> st;
   stack<int> st2;
   stack<string> st3;

   cout<<stack<int>::ob_cnt<<endl;
   cout<<stack<string>::ob_cnt;

   return 0;
}

template <typename T>
stack<T>::stack()
{
   top=-1;
   ob_cnt++;
}
```

Ολοκληρωμένο το πρόγραμμα είναι το: «cpp13.template_with_static.cpp»

Α. Θεωρία

2. Πρότυπα Συναρτήσεων

1. Ορισμός Πρότυπης Συνάρτησης

- Αντίστοιχα με τα πρότυπα κλάσεων μπορούμε να ορίσουμε και πρότυπα συναρτήσεων
- Ο ορισμός είναι αντίστοιχος:

```
template <typename T>
… func(…)
```

- και χρησιμοποιούμε το T όπου το χρειαζόμαστε στην συνάρτησή μας (παράμετρος, επιστροφή ή στο σώμα της συνάρτησης).

- Η κλήση της συνάρτησης ορίζει και τον τύπο δεδομένων για το T, π.χ.:

```
func<int>(args);
```

Α. Θεωρία

2. Πρότυπα Συναρτήσεων

2. Παράδειγμα

- Βλέπουμε και το κλάσικο παράδειγμα (swap) με χρήση πρότυπης συνάρτησης:

```
template <typename T>
void myswap(T &a, T &b)
{
   T temp;
   temp=a;
   a=b;
   b=temp;
}
```

```
int main()
{
   int a=2, b=3;

   cout<<"a="<<a<<" b="<<b<<endl;

   myswap<int>(a,b);

   cout<<"a="<<a<<" b="<<b<<endl;
   return 0;
}
```

Ολοκληρωμένο το πρόγραμμα είναι το: «cpp13.swap.cpp»

Α. Θεωρία

2. Πρότυπα Συναρτήσεων

3. Φιλικές Συναρτήσεις σε Πρότυπα Κλάσεων

- Αντίστοιχα μπορούμε να ορίσουμε μια συνάρτηση που να είναι φιλική σε μία κλάση – πρότυπο.
- π.χ. αν θέλουμε να ορίσουμε την υπερφόρτωση του τελεστή << στην κλάση πρότυπο στοίβας κάνουμε τα εξής:

```
template <typename T>
class stack {
 public:
   …
   template <typename U>friend ostream &operator<<(ostream &left, const stack<U> &right);
   …
};
```

```
template <typename T>
ostream &operator<<(ostream &left, const stack<T> &right)
{
   for (int i=0; i<=right.top; i++)
      left<<right.array[i]<<" ";

   return left;
}
```

Ολοκληρωμένο το πρόγραμμα είναι το: «cpp13.template_with_friend.cpp»

Β. Ασκήσεις

Άσκηση 1: Κλάση ARRAY

Τροποποιήστε την κλάση ARRAY του μαθήματος 7 (εφαρμογή 1) ώστε να είναι παραμετρική.

Β. Ασκήσεις

Άσκηση 2: Απλά Συνδεδεμένη Λίστα

Κατασκευάστε μία κλάση – πρότυπο η οποία να υλοποιεί την δομή δεδομένων απλά συνδεδεμένη λίστα (Για τις βασικές πράξεις και την λειτουργικότητα της βλέπε «Δομές Δεδομένων σε C – Μάθημα 4 – Απλά Συνδεδεμένη Λίστα» και επεκτείνετε την αρχική κατασκευή που είχαμε κάνει στο «Μάθημα 7 – Φιλικές Κλάσεις και Συναρτήσεις»).