Η ΓΛΩΣΣΑ C++

Μάθημα 7:

Φιλικές Κλάσεις και Συναρτήσεις

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Θεωρία

1. Φιλικές Κλάσεις

1. Γενικά

Μπορούμε να ορίσουμε μία κλάση Α να είναι **φιλική** σε μία κλάση Β

- Με τον τρόπο αυτό τα αντικείμενα της κλάσης Α έχουν πρόσβαση στα ιδιωτικά στοιχεία των αντικειμένων της κλάσης Β.

- Ο ορισμός γίνεται ως εξής:

```
class A {
  ....
};

class B {
   public:
      friend class A;
}
```

- Πρακτικά δηλώνουμε σε μια κλάση, άλλες κλάσεις που θα έχουν πρόσβαση στα ιδιωτικά της μέλη.
- Στον ορισμό του παραδείγματος, τα αντικείμενα της κλάσης Α, θα έχουν πρόσβαση στα ιδιωτικά στοιχεία της κλάσης Β.

Α. Θεωρία

1. Φιλικές Κλάσεις

2. Ένα απλό παράδειγμα

- Το ακόλουθο απλό παράδειγμα φιλικών κλάσεων έχει και θεολογικές ανησυχίες…

```
class man
{
  public:
    friend class god;
    man(int in_sins);
    int get_sins() const;
  private:
    int sins;
};

man::man(int in_sins)
{
   sins = in_sins;
}
int man::get_sins() const
{
   return sins;
}
```

```
class god
{
  public:
    void forgive(man &ob);
};

void god::forgive(man &ob)
{
   ob.sins = 0;
}
```

```
int main()
{
   man Euthypro(100);
   god Apollo;

   cout<<"Sins="<<Euthypro.get_sins()<<endl;
   Apollo.forgive(Euthypro);
   cout<<"Sins="<<Euthypro.get_sins();

   return 0;
}
```

Το πρόγραμμα είναι το: «cpp7.friend_class_ex1.cpp»

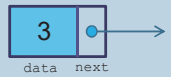
Α. Θεωρία
1. Φιλικές Κλάσεις
2. Ένα απλό παράδειγμα

- Παρατηρήσεις:
  - Μία «φιλία» έχει κατεύθυνση
    - Π.χ. η κλάση god είναι φιλική στην κλάση man
      - αλλά δεν ισχύει το αντίθετο
    - Άρα τοποθετούμε σε μία κλάση, μία άλλη κλάση που είναι φιλική σε αυτήν.
  - Οι «φιλίες» πρέπει να έχουν περιορισμένη χρήση.
    - Π.χ. αν όλοι είναι φίλοι με όλους, τότε είναι (σχεδόν) σαν να έχουμε δηλώσει όλα τα μέλη των κλάσεων να είναι δημόσια.

Α. Θεωρία
1. Φιλικές Κλάσεις
3. Ένα σύνθετο παράδειγμα (Συνδεδεμένη Λίστα)

- Βλέπουμε και ένα παράδειγμα που οι φιλικές κλάσεις είναι ιδιαίτερα χρήσιμες.
- Στο μάθημα «Δομές Δεδομένων σε C – Μάθημα 4: Απλά Συνδεδεμένη Λίστα» είδαμε την αντίστοιχη δομή δεδομένων που αποτελείται (με όρους C++) από δύο κλάσεις:
  - Τον κόμβο (μέλος: data, next)
  - Την λίστα (μέλος: head και μεθόδους τις ενέργειες επί της λίστας)
- Η κλάση λίστα θα ορίζεται φιλική στον κόμβο, ώστε να έχει πρόσβαση στα ιδιωτικά της μέλη.
- Ο ορισμός του κόμβου θα είναι:

```
class node
{
  public:
    friend class linked_list;
  private:
    int data;
    node *next;
};
```

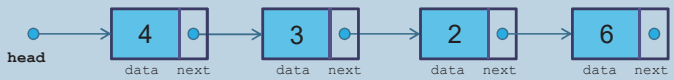


  - όπου ορίζεται ότι η συνδεδεμένη λίστα θα είναι φιλική σε αυτήν.

Α. Θεωρία
1. Φιλικές Κλάσεις
3. Ένα σύνθετο παράδειγμα (Συνδεδεμένη Λίστα)

- Ο ορισμός ενός (υποσυνόλου) της συνδεδεμένης λίστας θα είναι:

```
class linked_list
{
  public:
    linked_list();
    bool insert_start(int in_data);
    void print();
  private:
    node *head;
};
```



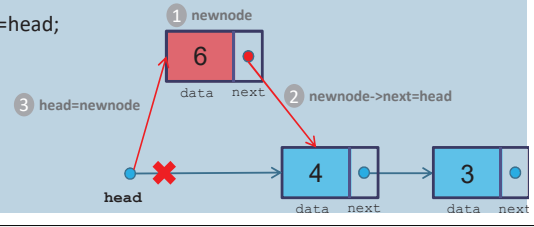
- Με αυτές τις συναρτήσεις μπορούμε να κάνουμε
  - απλές εισαγωγές στην αρχή της λίστας
  - και να κάνουμε επίσης και μία εκτύπωση των στοιχείων της λίστας
- Οι συναρτήσεις αυτές, θα έχουν πρόσβαση στα ιδιωτικά μέλη των κόμβων.

Α. Θεωρία
1. Φιλικές Κλάσεις
3. Ένα σύνθετο παράδειγμα (Συνδεδεμένη Λίστα)

- Ο ορισμός των μεθόδων είναι:

```
linked_list::linked_list()
{
   head = NULL;
}

void linked_list::print()
{
  node *current;

  current=head;
  while(current!=NULL)
  {
    cout<<current->data<<" ";
    current=current->next;
  }
}
```

```
bool linked_list::insert_start(int x)
{
  node *newnode;

  newnode = new node;
  if (!newnode)
  {
    cout<<"Error allocating memory";
    return false;
  }
  newnode->data=x;

  newnode->next=head;
  head=newnode;
  return true;
}
```



Α. Θεωρία
1. Φιλικές Κλάσεις
3. Ένα σύνθετο παράδειγμα (Συνδεδεμένη Λίστα)

- Βλέπουμε ότι ο αντικειμενοστραφής τρόπος σκέψης ταιριάζει πολύ περισσότερο με τις δομές δεδομένων.
  - και προσφέρει σημαντικά πλεονεκτήματα έναντι του διαδικαστικού προγραμματισμού:
    - όπως η απόκρυψη πληροφορίας,
    - και η δημιουργία μιας πιο φιλικής προγραμματιστικής διεπαφής.
- Έτσι κάποιος τώρα μπορεί να χρησιμοποιήσει την κλάση μας

```
int main()
{
   linked_list list;

   list.insert_start(5);
   list.insert_start(4);
   list.insert_start(3);
   list.print();

   return 0;
}
```

  - Με πιο εύκολο τρόπο από τον αντίστοιχο κώδικα στη C

Το πρόγραμμα είναι το: «cpp7.friend_class_linked_list.cpp»

Α. Θεωρία
2. Φιλικές Συναρτήσεις
1. Φιλικές συναρτήσεις σε κλάση

Μπορούμε να ορίσουμε μία συνάρτηση f να είναι **φιλική** σε μία κλάση A
- Με τον τρόπο αυτό η συνάρτηση f «βλέπει» τα ιδιωτικά μέλη των αντικειμένων της κλάσης.

- Ο ορισμός γίνεται ως εξής:

```
class A {
  public:
     friend type f (..);
  ...
};

type f (...)
{
 ...
}
```

  - Πρακτικά δηλώνουμε ότι η συνάρτηση (που δεν ανήκει στην κλάση) έχει πρόσβαση στα ιδιωτικά μέλη της κλάσης.

Α. Θεωρία
2. Φιλικές Συναρτήσεις
1. Φιλικές συναρτήσεις σε κλάση

- Βλέπουμε και ένα παράδειγμα βγαλμένο από τη ζωή:

```
class man
{
  public:
    friend void good_weather(man &ob);
    man(int in_mood);
    void report_mood();
  private:
    int mood;
};

void good_weather(man &ob);
```

```
man::man(int in_mood)
{
   mood = in_mood;
}

void man::report_mood()
{
   if (mood<10)
      cout<<"I am ok..."<<endl;
   else
      cout<<"I feel good!"<<endl;
}

void good_weather(man &ob)
{
   ob.mood +=10;
}
```

```
int main()
{
   man bob(5);

   bob.report_mood();
   good_weather(bob);
   bob.report_mood();

   return 0;
}
```

Το πρόγραμμα είναι το: «cpp7.friend_class_linked_list.cpp»

Α. Θεωρία
2. Φιλικές Συναρτήσεις
2. Φιλικές μέθοδοι σε κλάση

Μία **μέθοδος κλάσης** (ως συνάρτηση και η ίδια) μπορεί να είναι **φιλική** σε μία κλάση
- Έτσι, σε αντίθεση με τις φιλικές κλάσεις, όπου όλες οι μέθοδοι της φιλικής κλάσης έχουν πρόσβαση στα ιδιωτικά μέλη,
- Περιορίζουμε την πρόσβαση της φιλικής κλάσης μόνο στην μέθοδο που έχει δηλωθεί φιλική.

- Ο ορισμός γίνεται ως εξής:

```
class A {
  ....
  some_method(...);
};

class B {
   public:
      friend A::some_method;
}
```

  - Πρακτικά δηλώνουμε ότι κάποια μέθοδος άλλης κλάσης θα έχει πρόσβαση στα ιδιωτικά μέλη της κλάσης.

Α. Θεωρία
2. Φιλικές Συναρτήσεις
2. Φιλικές μέθοδοι σε κλάση

- Βλέπουμε και ένα παράδειγμα βγαλμένο από τη ζωή΄πάλι:

```
class man; //forward declaration

class state
{
  public:
    void withdraw(man &ob);
};

class man
{
  public:
    friend void state::withdraw(man &ob);
    man(int in_money);
    int get_money() const;
  private:
    int money;
};
```

```
void state::withdraw(man &ob)
{
  ob.money=0;
}

man::man(int in_money)
{
  money = in_money;
}

int man::get_money() const
{
  return money;
}
```

```
int main()
{
   man Papadakis(1500);
   state Greece;

   cout<<"Papadakis' Money="
      <<Papadakis.get_money()
      <<endl;

   Greece.withdraw(Papadakis);

   cout<<"Papadakis' Money="
      <<Papadakis.get_money();

   return 0;
}
```

Το πρόγραμμα είναι το: «cpp7.friend_method.cpp»

Α. Θεωρία
3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)
1. Γενικά

- Οι φιλικές συναρτήσεις μπορούν να χρησιμοποιηθούν για την υπερφόρτωση τελεστών σε δύο χρήσιμες περιπτώσεις:
  - Όταν θέλουμε να είναι αριστερό μέλος σε διθέσιο τελεστή απλή μεταβλητή και δεξί μέλος αντικείμενο.
    - Ώστε π.χ. να γράφουμε εντολές σαν την εξής:

      ```
      5+ob
      ```

  - Όταν θέλουμε να υπερφορτώσουμε τον τελεστή <<
    - Ώστε να γράφουμε εντολές σαν την εξής:

      ```
      cout<<ob;
      ```

- Αλλά επίσης ισχύει ότι αρκετές από τις υπερφορτώσεις που κάναμε στο προηγούμενο μάθημα μπορούν να γίνουν και μέσω φιλικών κλάσεων.

Α. Θεωρία
3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)
2. Άριστερό μέλος που είναι απλή μεταβλητή.

Η υπερφόρτωση του διθέσιου αριθμητίκού τελεστή + με αριστερό μέλος μεταβλητή γίνεται
- Γράφοντας ως φιλική μέθοδο στην κλάση την:
  - **friend class_name operator+ (int left, class_name &right);**

π.χ. στην κλάση complex ενσωματώνουμε στην κλάση την δήλωση:
- complex operator+(int left, complex &right);

Ενώ στο σώμα της μεθόδου:
- και δεδομένου ότι η πρόσθεση γίνεται μεταξύ αντικειμένου και ακεραίου(π.χ. **left+right**)
- Η μέθοδος καλείται με ορίσματα τα left και right
- Είναι σαν να κανουμε την κλήση **operator+(left, right)**
- και επιστρέφει ένα καινούργιο αντικείμενο, το οποίο είναι το αποτέλεσμα της πράξης.

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
complex operator+(int left, complex &right)
{
   complex result;

   result.real = left+right.real;
   result.imag = left+right.imag;
   return result;
}
```


Α. Θεωρία
3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)
2. Άριστερό μέλος που είναι απλή μεταβλητή

Επεκτείνουμε την κλάση complex του προηγούμενου μαθήματος ως εξής:

```
/*cpp7.friend_operator_plus.cpp Υπερφόρτωση του + με
φιλική συνάρτηση */

#include <iostream>
using namespace std;

class complex {
  public:
    ...
    friend complex operator+(int left, complex &right);
  private:
    double real;
    double imag;
};
```

```
int main()
{
   complex a(1.0,1.0);
   complex b(2.0,3.0);

   complex c;

   c = 5 + a;

   cout<<c.get_real()<<" "<<c.get_imag();

   return 0;
}
```

```
...
complex operator+(int left, complex &right)
{
   complex result;

   result.real = left+right.real;
   result.imag = left+right.imag;
   return result;
}
```

Α. Θεωρία

3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)

2. Άριστερό μέλος που είναι απλή μεταβλητή

Μέσω φιλικών κλάσεων μπορούμε να κατασκευάσουμε και όλους τους διθέσιους τελεστές (εκτός του =, που πρέπει να είναι μέλος της κλάσης) που είδαμε στο προηγούμενο μάθημα.

- Π.χ. η δήλωση της προηγούμενου μαθήματος της υπερφόρτωσης του +:

```
class_name operator+ (int right);
```

- μπορεί να γίνει (εντελώς ισοδύναμα) μέσω φιλικών συναρτήσεων:

```
friend class_name operator+ (class_name &left, int right);
```

- Θα επιλέξουμε ο τελεστής να είναι μέλος της κλάσης, σε όλες τις περιπτώσεις που δεν έχει αριστερό μέλος απλή μεταβλητή.
  - Είναι πιο φυσικός στη χρήση και πιο συνηθισμένος στην πράξη.

Α. Θεωρία

3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)

2. Υπερφόρτωση του <<

- Πολύ χρήσιμη υπερφόρτωση είναι του τελεστή <<
- Ώστε να μπορούμε να κάνουμε εκτυπώσεις όπως:

```
cout<<ob;
```

- Η πλήρης κατανόηση του πως ακριβώς γίνεται αυτή η υπερφόρτωση απαιτεί γνώσεις των ρευμάτων εισόδου/εξόδου που θα μελετήσουμε σε επόμενο μάθημα.
  - Για την ώρα θα το μελετήσουμε εμπειρικά.
- Το cout, όπως έχουμε πει, είναι ένα αντικείμενο το οποίο διαχείριζεται την έξοδο στην οθόνη.
  - Και όπως κάθε αντικείμενο, έτσι και αυτό είναι στιγμιότυπο μίας κλάσης
    - Συγκεκριμένα της ostream που διαχειρίζεται ρεύματα εξόδου.

Α. Θεωρία

3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)

2. Υπερφόρτωση του <<

Η υπερφόρτωση του << με δεξί μέλος αντικείμενο της κλάσης γίνεται
- Γράφοντας ως φιλική μέθοδο στην κλάση την:
  - **friend ostream &operator<<(ostream &left, const class_name &right);**

π.χ. στην κλάση μιγαδικών ενσωματώνουμε τη δήλωση
- friend ostream &operator<<(ostream &left, const complex &right);

Ενώ στο σώμα της μεθόδου:
- Η εκτύπωση γίνεται π.χ. με την εντολή: **cout<<right**
- Η μέθοδος καλείται με ορίσματα τα cout και ob
- Είναι σαν να κανουμε την κλήση **operator<<(cout, ob)**
- και πρέπει να επιστρέφει αναφορά στο αντικείμενο εκτύπωσης ώστε να είναι εφικτές διαδοχικές εκτυπώσεις..

Στο παράδειγμα το σώμα της συνάρτησης είναι:

```
friend ostream &operator<<(ostream &left, const complex &right)
{
  left<<"("<<right.real<<", "<<right.imag<<")";

  return left;
}
```


Α. Θεωρία

3. Υπερφόρτωση Τελεστών (με φιλικές συναρτήσεις)

2. Υπερφόρτωση του <<

Επεκτείνουμε την κλάση complex του προηγούμενου μαθήματος ως εξής:

```
/*cpp7.friend_operator_print.cpp Υπερφόρτωση του << με
φιλική συνάρτηση */

#include <iostream>
using namespace std;

class complex {
 public:
   ...
   friend ostream &operator<<(ostream &left,
                              const complex &right);
 private:
   double real;
   double imag;
};
```

```
int main()
{
  complex a(1.0,1.0);

  cout<<a;

  return 0;
}
```

```
...
ostream &operator<<(ostream &left,
                    const complex &right)
{
  left<<"("<<right.real<<", "<<right.imag<<")";

  return left;
}
```

Α. Θεωρία
4. Σύνοψη για τις υπερφορτώσεις

- Για την υπερφόρτωση των:
  - Αριθμητικών Τελεστών (+,-,*,/,%) προτιμάμε την υπερφόρτωση ως μέλος της κλάσης
    - Εκτός και αν έχει στο αριστερό μέρος στοιχειώδη τύπο δεδομένων οπότε απαιτείται υπερφόρτωση με φιλική συνάρτηση
  - Μονοθέσιων Τελεστών (++, --) απαιτείται η υπερφόρτωση ως μέλος της κλάσης
  - Ισότητας (=) απαιτείται η υπερφόρτωση ως μέλους κλάσης
  - Πίνακα ([]) απαιτείται η υπερφόρτωση ως μέλους κλάσης
  - Εκτύπωσης (<<) απαιτείται η υπερφόρτωση με φιλική συνάρτηση.

- Επίσης ας γνωρίζουμε (αν φανεί χρήσιμο κάποτε) ότι μπορούμε να υπερφορτώσουμε:
  - Τους αριθμητικούς τελεστές +=,-=, *=, /=, %=
  - Τους bitwise: &, |, ^, <<, >>, ~
  - Τους σχεσιακούς τελεστές: ==, !=, <, >, <=, >=
  - Τους λογικούς τελεστές ||, && και !
  - Τους new και delete
  - ->, ->*, , , *, &, ()

Β. Ασκήσεις
Άσκηση 1: Κλάση ARRAY

Υπερφορτώστε τον τελεστή << της κλάσης ARRAY (Μάθημα 6, Άσκηση 2) ώστε να τυπώνει εύσχημα τα περιεχόμενα του πίνακα

Β. Ασκήσεις
Άσκηση 2.1: Κλάση STRING

Επεκτείνουμε τη κλάση STRING (Μάθημα 6, Άσκηση 1.3) με υπερφόρτωση τελεστών:

- Υπερφορτώστε τον τελεστή << ώστε να τυπώνει τη συμβολοσειρά.

Β. Ασκήσεις
Άσκηση 2.2: Κλάση STRING

Επεκτείνουμε τη κλάση STRING (Μάθημα 6, Άσκηση 1.2) με υπερφόρτωση τελεστών:

- Υπερφορτώστε τον τελεστή >> ώστε να διαβάζει μία λέξη από την είσοδο (αντίστοιχα με την ostream για την έξοδο, χρησιμοποιήστε την istream, αντικείμενο της οποία είναι το cin)

Β. Ασκήσεις

Άσκηση 3: Φιλική Συνάρτηση προς δύο κλάσεις

Μία συνάρτηση μπορεί να είναι φιλική προς δύο (ή περισσότερες) κλάσεις.

- Ορίστε μία κλάση καρέκλα με ιδιωτικά μέλη:
  - το χρώμα της (συμβολοσειρα) και το ύψος της (ακέραιος)
  - να έχει κατασκευαστή
  - να έχει υπερφόρτωση του <<
- Ορίστε μία κλάση τραπέζι με ιδιωτικά μέλη:
  - το χρώμα της (συμβολοσειρα) το μήκος και το πλάτος του (ακέραιοι)
  - να έχει κατασκευαστή
  - να έχει υπερφόρτωση του <<
- Κατασκευάστε μία συνάρτηση shrink, η οποία να παίρνει ως ορίσματα ένα τραπέζι και μία καρέκλα
  - και να μειώνει τα μέλη διαστάσεών τους κατά 10%

Κατασκευάστε και μία main που να αναδεικνύει τα παραπάνω.

Β. Ασκήσεις

Άσκηση 4.1: Κλάση «Σημείο»

Ορίστε μια κλάση σημείο (point):

- Έχει ως μέλη τις συντεταγμένες του στο 2Δ χώρο (x,y)
- Έχει κατασκευαστή. Οι συντεταγμένες πρέπει να είναι από 0 έως 59.
  - Αν εισαχθεί μη έγκυρη τιμή να εμφανίζεται μήνυμα λάθους και να τίθεται η λάθος συντεταγμένη ίση με 0
- Έχει accessors για τις συντεταγμένες του
- Έχει υπερφόρτωση του << ώστε να τυπώνει ένα σημείο ως (x,y)

Ελέγξτε την κλάση σας, με μία κατάλληλη συνάρτηση main.

Β. Ασκήσεις

Άσκηση 4.2: Κλάση «Πλαίσιο»

Θεωρώντας την κονσόλα σαν ένα NxN πλαίσιο, μπορούμε να απεικονίσουμε σε αυτήν γεωμετρικά σχήματα:

- Ορίστε μία κλάση board η οποία:
  - Έχει ως μέλος τη διάσταση N
  - Έναν πίνακα χαρακτήρων NxN, του οποίου κάθε θέση θα «χρωματίζεται» με κάποιον χαρακτήρα
  - να έχει κατασκευαστή (δέχεται το N και δεσμεύει δυναμικά το χώρο μνήμης). Αρχικοποιεί τους χαρακτήρες σε τελείες (.)
  - να έχει καταστροφέα
  - να έχει κατασκευαστή αντιγράφου
  - να έχει υπερφόρτωση του =
  - να έχει accessors για το (i,j) στοιχείο του πίνακα.
  - να υπερφορτώνει τον τελεστή << και να τυπώνει το πλαίσιο στην οθόνη.

Ελέγξτε την κλάση σας, με μία κατάλληλη συνάρτηση main.

Β. Ασκήσεις

Άσκηση 4.3: Κλάση «Πλαίσιο» - συνέχεια

Επεκτείνετε την κλάση board

- ώστε να έχει μία μέθοδο insert_point
  - η οποία παίρνει ως όρισμα ένα σημείο (αντικείμενο point) και να ενημερώνει το πλαίσιο.
    - θέτοντας αντίστοιχο χαρακτήρα ίσο με το χαρακτήρα 'O'
  - Αν το σημείο είναι εκτός του πλαισίου να μην το τυπώνει.
- να έχει μία μέθοδο clear:
  - η οποία θα σβήνει το πλαίσιο (σβήσιμο των σημείων).

Ορίστε τη συνάρτηση main ώστε επαναληπτικά να δίνει τέσσερις επιλογές στο χρήστη:

- Να εισάγει ένα καινούργιο σημείο
- Να εκτυπώσει το πλαίσιο
- Να καθαρίσει το πλάισιο
- Να γίνει έξοδος από το πρόγραμμα