Η ΓΛΩΣΣΑ C++

Μάθημα 8:

Στατικά Μέλη Κλάσεων

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Θεωρία

1. Στατικές μεταβλητές – μέλη κλάσης

1. Δήλωση και αρχικοποίηση

- **Στατική Μεταβλητή-Μέλος Κλάσης:**
  - Είναι μια μεταβλητη που είναι κοινή για όλα τα αντικείμενα μιας κλάσης
    - (σε αντίθεση με τα κανονικά μέλη που κάθε αντικείμενο έχει τα δικά του μέλη)

- Μια στατική μεταβλητή δηλώνεται εκτός της κλάσης, ως εξής:

```
data_type class_name::var_name = init_value;
```

  - (πρέπει να γίνει και η αρχικοποίηση της μεταβλητής)
- πρέπει να δηλωθεί στην κλάση με τη λέξη-κλειδί static, π.χ.:

```
class class_name {
  public:
     static data_type var_name;
  …
};
```

- και μπορούμε να έχουμε πρόσβαση (εκτός της κλάσης) σε αυτήν ως εξής:

```
class_name::var_name
```

Παρατήρηση:

- Με τις στατικές μεταβλητές απεικονίζουμε πληροφορίες που αφορούν την κλάση.

Α. Θεωρία

1. Στατικές μεταβλητές – μέλη κλάσης

2. Παράδειγμα

- Με τις στατικές μεταβλητές ορίζουμε συνήθως:
  - Μετρητές που μετράνε π.χ. πόσα αντικείμενα της κλάσης είναι ενεργά
  - Σταθερές (κοινές για όλα τα μέλη της κλάσης)
    - π.χ. το μέγιστο μήκος λέξης που επιτρέπεται στην κλάση STRING

- Βλέπουμε και ένα παράδειγμα:
  - Κατασκευάζουμε έναν πίνακα δεικτών σε αντικείμενα orc
  - Τα orcs ζωντανεύουν και ψοφάνε τυχαία.
  - Μετράμε (μέσω στατικής μεταβλητής) πόσα orcs είναι ζωντανά κάθε στιγμή.

Α. Θεωρία

1. Στατικές μεταβλητές – μέλη κλάσης

2. Παράδειγμα

```
/* cpp8.static_members.cpp */

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

#define N 5

class orc {
  public:
    static int alive_orcs;
    orc();
    ~orc();
  private:
    int h;
};

int orc::alive_orcs = 0;
```

```
orc::orc()
{
   alive_orcs++;
}

orc::~orc()
{
   alive_orcs--;
}
```

Α. Θεωρία

1. Στατικές μεταβλητές – μέλη κλάσης

2. Παράδειγμα

```
int main()
{
   orc *arr[N];
   int i, pos;

   for (i=0; i<N; i++)
      arr[i]=NULL;

   for (i=0; i<100; i++)
   {
      pos = rand()%N;
```

```
if (arr[pos]==NULL)
      {
         arr[pos]=new orc;
         if (!arr[pos]) exit(0);
      }
      else
      {
         delete arr[pos];
         arr[pos]=NULL;
      }


      if (i%10==0)
         cout<<"Step "<<i<<": Alive orcs = "<<orc::alive_orcs<<endl;
   }
   return 0;
}
```

Α. Θεωρία

2. Στατικές μέθοδοι κλάσης

1. Δήλωση και σκοπός

- **Στατική Μέθοδος Κλάσης:**
  - Είναι μια μέθοδος που είναι κοινή για όλα τα αντικείμενα μιας κλάσης

- πρέπει να δηλωθεί στην κλάση με τη λέξη-κλειδί static, π.χ.:

```
class class_name {
  public:
     static data_type func_name(params);
  …
};
```

  - Το σώμα της δηλώνεται ως συνήθως εκτός της κλάσης

- και μπορούμε να την καλέσουμε (εκτός της κλάσης) σε αυτήν ως εξής:

```
class_name::func_name(args)
```

Α. Θεωρία

2. Στατικές μέθοδοι κλάσης

1. Δήλωση και σκοπός

- Σημαντικό!
  - Ο κώδικας μίας στατικής μεθόδου αποθηκεύεται σε ξεχωριστό μέρος που αφορά την κλάση.
  - Έτσι έχει πρόσβαση μόνο στα static μέλη της κλάσης
    - και όχι στα συνηθισμένα μέλη
  - Όπως επίσης δεν έχει πρόσβαση στον δείκτη this (αφού δεν έχει πρόσβαση σε κάποιο συγκεκριμένο αντικείμενο)

- Συνήθως χρησιμοποιείται ώστε:
  - Να τεθεί η στατική μεταβλητή ως ιδιωτικό μέλος της κλάσης
    - και έπειτα να έχουμε πρόσβαση στη μεταβλητή μέσω της στατικής μεθόδου.
  - ή για να καθορίσει μια γενική συμπεριφορά της κλάσης

- Στο παράδειγμα που ακολουθεί επεκτείνουμε την κλάση orc ώστε να έχει:
  - μία στατική ιδιωτική μεταβλητή που καθορίζει αν τα orcs είναι σε frenzy mode
  - μία στατική δημόσια μέθοδο για να τροποποιείται η τιμή αυτής της μεταβλητής

Α. Θεωρία

2. Στατικές μέθοδοι κλάσης

2. Παράδειγμα

```
/* cpp8.static_methods.cpp */

#include <iostream>
using namespace std;

class orc {
  public:
    static int alive_orcs;
    static bool frenzy_mode;
    static void toggle_frenzy_mode();
    orc();
    ~orc();
    friend ostream &operator<<(ostream &left, const orc &right);
  private:
    int h;
};

int orc::alive_orcs = 0;
bool orc::frenzy_mode = false;
```

Α. Θεωρία

2. Στατικές μέθοδοι κλάσης

2. Παράδειγμα

```
int main()
{
   orc orc1,orc2;

   cout<<"NORMAL MODE"<<endl;
   cout<<"Orc1: "<<orc1<<endl;
   cout<<"Orc2: "<<orc2<<endl;

   orc::toggle_frenzy_mode();
   cout<<"FRENZY MODE"<<endl;
   cout<<"Orc1: "<<orc1<<endl;
   cout<<"Orc2: "<<orc2<<endl;

   return 0;
}
```

```
orc::orc()
{
   alive_orcs++;
}

orc::~orc()
{
   alive_orcs--;
}
```

Α. Θεωρία

2. Στατικές μέθοδοι κλάσης

2. Παράδειγμα

```
void orc::toggle_frenzy_mode()
{
   if (frenzy_mode)
      frenzy_mode=false;
   else
      frenzy_mode=true;
}

ostream &operator<<(ostream &left, const orc &right)
{
   if (orc::frenzy_mode)
      cout<<"AARFFGHHRHHHHR";
   else
      cout<<"ougba";
}
```

Α. Θεωρία

3. Στατικές τοπικές μεταβλητές

1. Δήλωση και Αρχικοποιήση

- Οι στατικές τοπικές μεταβλητές συμπεριφέρονται όπως στη C.
- (Ακολουθεί υπενθύμιση από το Μάθημα 14 της C)

- Σε κάποιες εξειδικευμένες εφαρμογές, δεν θέλουμε κάθε φορά που τελειώνει μια συνάρτηση να καταστρέφει μια συγκεκριμένη τοπική μεταβλητή της, αλλά να διατηρηθεί η τιμή της και την επόμενη φορά που θα κάνουμε κλήση της συνάρτησης.
- Η C μας δίνει αυτήν την ευκολία, δηλώνοντας την μεταβλητή με την λέξη κλειδί static πριν από την δήλωση της, π.χ.:

```
static int m;
```

- Συνεπώς μία στατική τοπική μεταβλητή είναι μία τοπική μεταβλητή που δεν καταστρέφεται όταν ολοκληρώνεται η κλήση της συνάρτησης, αλλά διατηρεί την τιμή της για την επόμενη φορά που θα γίνει κλήση της συνάρτησης.

  - Μια στατική τοπική μεταβλητή πρέπει να έχει πάντα και αρχικοποίηση κατά την δήλωση που θα εκτελεστεί μόνο την πρώτη φορά που θα εκτελεστεί η συνάρτηση.

Α. Θεωρία

3. Στατικές τοπικές μεταβλητές

2. Παράδειγμα

```
/* cpp8.static_local_variables.cpp */
#include <iostream>
using namespace std;

void func();

int main()
{
    int i;

    for (i=0; i<10; i++)
        func();

    return 0;
}
```

```
void func()
{
   int y=0;
   static int x=0;

   x=x+1;
   y=y+1;
   cout<<endl<<"x="<<x<<", y="<<y;
}
```

Α. Θεωρία
4. Χρόνος Ζωής Στατικών Μεταβλητών

- Οι στατικές μεταβλητές
  - κατασκευάζονται και δεσμεύουν χώρο πριν από την έναρξη της main
  - και καταστρέφονται μετά την έναρξη της main
- σε αντίθεση με:
  - τις συνήθεις μεταβλητές (καλούνται και αυτόματες) που έχουν χρόνο ζωής όσο και το μπλοκ κώδικα στο οποίο ανήκουν
    - π.χ. μία μεταβλητή που ορίζεται σε μια συνάρτηση δημιουργείται όταν ξεκινά η συνάρτηση
    - και καταστρέφεται όταν τελειώσει η συνάρτηση
  - και τις δυναμικές (που δεσμεύονται με τη new)
    - Δημιουργούνται όταν κάνουμε new
    - Καταστρέφονται όταν κάνουμε delete

- Βλέπουμε και ένα παράδειγμα στην επόμενη διαφάνεια.

Α. Θεωρία
4. Χρόνος Ζωής Στατικών Μεταβλητών

```
/* cpp8.lifespan.cpp */
#include <iostream>
using namespace std;

class life {
public:
   int x;
   life(int in_x);
   ~life();
};
```

```
life::life(int in_x)
{
   x = in_x;
   cout<<"Constructing "<<x<<endl;
}

life::~life()
{
   cout<<"Destructing "<<x<<endl;
}
```

```
int main()
{
   static life ob1(1);
   cout<<"Start Main"<<endl;
  {
     life ob2(2);
   }

   life *ob3 = new life(3);
   delete ob3;
   cout<<"End Main"<<endl;

   return 0;
}
```

Β. Ασκήσεις

Άσκηση 1: Κλάση με πολλαπλούς constructors

Επεκτείνετε την κλάση ARRAY (Άσκηση 1 – Μάθημα 7) με μία στατική μεταβλητή MAX_LENGTH που να αρχικοποιείται σε 1000:

- Διορθώστε τον constructor ώστε να δεσμεύει χώρο μόνο όταν το μέγεθος είναι <= 1000

Β. Ασκήσεις

Άσκηση 2: Αριθμοί Fibonacci

- Κατασκευάστε μία συνάρτηση με όνομα fibonacci η οποία παίρνει όρισμα έναν ακέραιο n και επιστρέφει τον αριθμό Fibonacci που υπολογίζεται σύμφωνα με τον τύπο:

$$f_n = \begin{cases} 1, & n = 1 \ \text{ή} \ n = 2 \\ f_{n-1} + f_{n-2}, & n > 2 \end{cases}$$

- Κατασκευάστε μία main που να υπολογίζει το 15ο αριθμό fibonacci
- Επεκτείνετε τη συνάρτηση με μία στατική τοπική μεταβλητή που υπολογίζει το πλήθος των αναδρομικών κλήσεων που παραγματοποιήθηκαν.