Αντικειμενοστραφής Προγραμματισμός

Μέρος 2ο: Κατασκευαστές και άλλα χρήσιμα εργαλεία

Εξάμηνο Σπουδών: 6ο

Κωδικός Μαθήματος: 647

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

Περιεχόμενα

- Κατασκευαστές
  - Ορισμός
  - Κλήση
- Άλλα εργαλεία ανάπτυξης
  - Ο τροποποιητής παραμέτρων `const`
  - Inline συναρτήσεις
  - Στατικά μέλη

Μέρος 1°:
Κατασκευαστές

Κατασκευαστές

- Αρχικοποιούν αντικείμενα
  - Συγκεκριμένα, αρχικοποιούν ορισμένες ή όλες τις μεταβλητές μέλη
  - Πιθανόν εκτελούν και άλλες ενέργειες
- Αποτελούν ειδική περίπτωση συναρτήσεων μελών
  - Καλούνται αυτόματα όταν δηλώνονται αντικείμενα
- Πολύ χρήσιμο εργαλείο
  - Ουσιαστικό μέρος του αντικειμενοστραφούς προγραμματισμού

Δήλωση κατασκευαστών

- Οι κατασκευαστές δηλώνονται όπως κάθε άλλη συνάρτηση μέλος
- Πρέπει να έχουν το ίδιο όνομα με αυτό της κλάσης (στο παράδειγμα `DayOfYear`)
  - Δεν επιστρέφουν τιμή (ούτε `void`)
  - Βρίσκονται στο δημόσιο τμήμα
  - Εάν ήταν στο ιδιωτικό, δεν θα μπορούσαν να κληθούν κατά τη δήλωση των αντικειμένων

```
class DayOfYear
{
public:
    DayOfYear(int monthValue, int dayValue);  //Initializes the month and day to arguments.
    DayOfYear(int monthValue);      //Initializes the date to the first of the given month.
    DayOfYear();                    //Initializes the date to January 1.
    // ...
private:
    int month;
    int day;
};
```

Κλήση κατασκευαστών

- Κατά τη δήλωση, δημιουργούνται τα αντικείμενα
  - Με κλήση του κατασκευαστή
  - Οι τιμές στις παρενθέσεις μεταβιβάζονται ως ορίσματα στον κατασκευαστή
  - Στο παράδειγμα, οι μεταβλητές μέλη `month` και `day` αρχικοποιούνται ως εξής:

`date1.day` $\rightarrow$ 21
`date1.month` $\rightarrow$ 2

`date2.day` $\rightarrow$ 5
`date2.month` $\rightarrow$ 5

`date3.day` $\rightarrow$ 1
`date3.month` $\rightarrow$ 1

```
int main( )
{
    DayOfYear date1(2, 21), date2(5,5), date3;  //Declare three objects of type DayOfYear
    cout << "Initialized dates:\n";             //using different constructors
    date1.output(); cout << endl;
    date2.output(); cout << endl;
    date3.output(); cout << endl;
}
```

Μια σημαντική σημείωση

- Θεωρήστε τον παρακάτω κώδικα:

```
DayOfYear date1, date2;
date1.DayOfYear(7, 4); // ILLEGAL!
date2.DayOfYear(5, 5); // ILLEGAL!
```

- Φαινομενικά θα ήταν αποδεκτός
  - καθώς οι κατασκευαστές αποτελούν ειδική περίπτωση συναρτήσεων μελών,
  - αλλά δεν μπορούμε να τους καλέσουμε όπως άλλες συναρτήσεις μέλη

- Οι κατασκευαστές καλούνται αυτόματα κατά τη δήλωση των αντικειμένων

Υλοποίηση κατασκευαστή

- Η υλοποίηση ενός κατασκευαστή γίνεται όμοια με αυτή κάθε άλλης συνάρτησης μέλους
  - Παρατηρείστε ότι υπάρχει το ίδιο όνομα γύρω από τον τελεστή εμβέλειας : :
  - Προσδιορίζεται έτσι με σαφήνεια ότι πρόκειται για κατασκευαστή
  - Δεν υπάρχει τύπος επιστροφής (ακριβώς όπως και στην δήλωση)
- Δύο κατασκευαστές δεν μπορούν να έχουν την ίδια υπογραφή (υπερφόρτωση κατασκευαστών)

```
//Initializes the date to the first of the given month.
DayOfYear::DayOfYear(int monthValue) {
    month = monthValue;
    day = 1;
}
//Initializes the month and day to arguments.
DayOfYear::DayOfYear(int monthValue, int dayValue) {
    month = monthValue;
    day = dayValue ;
}
```

Εναλλακτική υλοποίηση

- Οι υλοποιήσεις του προηγούμενου παραδείγματος είναι ισοδύναμες με τις παρακάτω
  - Εδώ γίνεται χρήση του λεγόμενου “τμήματος αρχικοποίησης” του κατασκευαστή
  - Το σώμα κάθε κατασκευαστή έμεινε άδειο
  - Προτιμώμενη προσέγγιση

```
//Initializes the date to the first of the given month.
DayOfYear::DayOfYear(int monthValue) : month(monthValue), day(1)
{
    /*Body intentionally empty.*/
}
//Initializes the month and day to arguments.
DayOfYear::DayOfYear(int monthValue, int dayValue) : month(monthValue), day(dayValue)
{
    /*Body intentionally empty.*/
}
```

Ο προκαθορισμένος κατασκευαστής

- Ορίζεται ως ο κατασκευαστής χωρίς ορίσματα (θα πρέπει πάντα να υλοποιείται)
- Δημιουργείται αυτόματα; Και ναι και όχι
  - Εάν η κλάση δεν ορίζει κάποιον κατασκευαστή $\rightarrow$ Ναι
  - Εάν η κλάση ορίζει έστω και έναν κατασκευαστή $\rightarrow$ Όχι
- Εάν δεν υπάρχει, τότε μια δήλωση όπως η παρακάτω δεν είναι δυνατή:
  - `MyClass myObject;`

```
//Implementation of the default constructor.
DayOfYear::DayOfYear() {
    month = 1;
    day = 1;
}
```

Ένας επιπρόσθετος λόγος για την ύπαρξη κατασκευαστή

- Ο κατασκευαστής μπορεί να χρησιμοποιηθεί και για την επικύρωση των δοθέντων τιμών
  - Να βεβαιώσει δηλαδή ότι μόνο ορθές τιμές εκχωρούνται στις ιδιωτικές μεταβλητές μέλη
  - Όχι μόνο για την αρχικοποίηση των μεταβλητών μελών
  - Ουσιαστική αρχή του αντικειμενοστραφούς προγραμματισμού

```
//Initializes the month and day to arguments.
DayOfYear::DayOfYear(int monthValue, int dayValue) : month(monthValue), day(dayValue)
{
    if ((month < 1) || (month > 12)) {
        cout << "Illegal month value!"; exit(1);
    }
    if ((day < 1) || (day > 31)) {
        cout << "Illegal day value!"; exit(1);
    }
}
```

Hands-on ενότητα 1

Υλοποίηση της κλάσης DayOfYear με κατασκευαστές

Ρητές κλήσεις κατασκευαστών

- Ένας κατασκευαστής μπορεί να κληθεί ξανά και μετά τη δήλωση του αντικειμένου
  - Όπως γίνεται δηλαδή και με τις υπόλοιπες συναρτήσεις μέλη
  - Η μέθοδος αυτή είναι χρήσιμη π.χ. για τον (επαν-)ορισμό των μεταβλητών μελών
  - Ωστόσο, είναι αρκετά διαφορετική από την τυπική κλήση συνάρτησης μέλους.
- Μια τέτοια κλήση επιστρέφει “ανώνυμο αντικείμενο”, το οποίο μπορεί εν συνεχεία να ανατεθεί

```
int main()
{
  //Default constructor call
  DayOfYear holiday(17, 11);
  holiday.output();

  //An explicit constructor call
  holiday = DayOfYear(25, 12);
  holiday.output();
}
```

Τύποι μεταβλητών μελών μιας κλάσης

- Οι μεταβλητές μέλη μιας κλάσης μπορούν να είναι οποιουδήποτε τύπου
  - δηλαδή μπορεί να είναι αντικείμενα άλλων κλάσεων
- Αυτό ορίζει την σχέση “έχει” μεταξύ των αντίστοιχων αντικειμένων
  - π.χ. κάθε αντικείμενο τύπου LineSegment έχει δύο μεταβλητές μέλη τύπου Point.
- Όταν οι μεταβλητές μέλη είναι αντικείμενα, η αρχικοποίηση τους στον κατασκευαστή απαιτεί ειδική σύνταξη

```
class Point {
public:
   Point();
   Point(int xCoord, yCoord);
   int getX() const;
   int getY() const;
private:
   int x, y;
};
```

```
class LineSegment {
public:
    LineSegment(int x1, int y1, int x2, int y2);
private:
    Point startPoint, endPoint;
};

LineSegment::LineSegment(int x1, int y1, int x2, int y2)
  : startPoint(x1, y1), endPoint(x2, y2) {}
```

Hands-on ενότητα 2

Υλοποίηση της κλάσης Holiday

Αντικείμενα ως παράμετροι συναρτήσεων

- Οι παράμετροι συναρτήσεων μπορεί να είναι:
  - Με τιμή: Απαιτείται αντιγραφή $\rightarrow$ Επιβάρυνση
  - Με αναφορά: Δεν απαιτεί αντιγραφή $\rightarrow$ Πιο αποτελεσματική μέθοδος
- Η διαφορά τους είναι αμελητέα για απλούς τύπους
- Υπάρχει σαφές πλεονέκτημα για παραμέτρους αντικείμενα
  - Η κλήση με αναφορά πλεονεκτεί
  - Ειδικά για αντικείμενα που ενθυλακώνουν “μεγάλα” δεδομένα

```
//Welcoming a customer to his/her back account
void welcome(BankAccount& yourAccount);

void welcome(BankAccount& yourAccount) {
    cout << "Welcome to our bank.\n"
         << "The status of your account is:\n";
    yourAccount.output();
}
```

Μέρος 2°:
Άλλα εργαλείας ανάπτυξης

Ο τροποποιητής παραμέτρων `const`

- Η τοποθέτηση της λέξης-κλειδί `const` πριν τον τύπο της παραμέτρου
  - Καθιστά την παράμετρο “μόνο για ανάγνωση”
  - Κάθε προσπάθεια τροποποίησης της οδηγεί σε σφάλμα μεταγλώττισης
- Εάν δεν υπάρχει λόγος για τροποποίηση της παραμέτρου από τη συνάρτηση
  - προτείνεται η “προστασία” της παραμέτρου με τη χρήση του τροποποιητή `const`
- Καλή πρακτική: Να γίνεται αυτό όπου είναι εφικτό

```
//Welcoming a customer to his/her back account
void welcome(const BankAccount& yourAccount);

void welcome(const BankAccount& yourAccount) {
    cout << "Welcome to our bank.\n"
         << "The status of your account is:\n";
    yourAccount.output();
}
```

Const συναρτήσεις μέλη

- Αν μια συνάρτηση μέλος οριστεί `const` δεν επιτρέπεται να τροποποιήσει τις τιμές των μεταβλητών μελών της κλάσης
- Αντικείμενα που έχουν δηλωθεί `const` μπορούν να καλέσουν μόνο `const` συναρτήσεις μέλη
- Καλή πρακτική: Συναρτήσεις μέλη που δεν τροποποιούν μεταβλητές μέλη θα πρέπει να δηλώνονται `const`
- Η λέξη-κλειδί `const` χρησιμοποιείται στη δήλωση όσο και στην υλοποίηση της συνάρτησης

```
class BankAccount {
private:
    int accountEuros; //The account balance
    int accountCents; //euros,cents
public:
    double getBalance() const;
};
double BankAccount::getBalance() const {
    return (accountEuros + accountCents*0.01);
}
```

Inline συναρτήσεις

- Για συναρτήσεις που δεν είναι μέλη:
  - Χρησιμοποιήστε τη λέξη-κλειδί `inline` στην υλοποίηση τους
  - Δεν απαιτείται δήλωση της συνάρτησης
- Ο κώδικας της συνάρτησης εισάγεται στο σημείο της κλήσης από τον μεταγλωττιστή
  - Μειώνεται η επιβάρυνση από την κλήση (δεν γίνεται κλήση)
  - Αποδοτικότερη προσέγγιση αν η υλοποίηση είναι μικρή

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

//An inline function to compute the cube of an integer
inline int cube(int s) { return s * s * s; }

int main() {
    cout << "The cube of 3 is: " << cube(3) << endl;
    return 0;
}
```

Inline συναρτήσεις

- Για συναρτήσεις μέλη:
  - Τοποθετήστε την υλοποίηση (κώδικα) της συνάρτησης στον ορισμό της στην κλάση
  - Αυτό καθιστά αυτόματα τη συνάρτηση μέλος inline
- Ο κώδικας της συνάρτησης εισάγεται στο σημείο της κλήσης από τον μεταγλωττιστή
  - Μειώνεται η επιβάρυνση από την κλήση (δεν γίνεται κλήση)
  - Αποδοτικότερη προσέγγιση αν η υλοποίηση είναι μικρή

```
class BankAccount {
private:
    int accountEuros; //The account balance
    int accountCents; //euros,cents
public:
    double getBalance() const {
        return (accountEuros + accountCents*0.01);
    }
};
```

Ανακατεύθυνση κατασκευαστή

- Η C++11 επιτρέπει σε έναν κατασκευαστή να καλέσει έναν άλλο
- Στο παράδειγμα:
  - Ο προκαθορισμένος κατασκευαστής καλεί έναν άλλον κατασκευαστή για την αρχικοποίηση των μεταβλητών μελών
  - Αυτό γίνεται κάνοντας κλήση στον κατάλληλο κατασκευαστή στο τμήμα αρχικοποίησης

```
class BankAccount {
private:
    int accountEuros; //The account balance
    int accountCents; //euros,cents
public:
    BankAccount();
    BankAccount(int euros, int cents);
};
BankAccount::BankAccount() : BankAccount(0, 0) {}
BankAccount::BankAccount(int euros, int cents) : accountEuros(euros), accountCents(cents) {}
```

Στατικές μεταβλητές μέλη

- Ορίζονται εισάγοντας τη λέξη-κλειδί `static` πριν τον τύπο τους
- Αν μια μεταβλητή μέλος είναι στατική, τότε όλα τα αντικείμενα της κλάσης “μοιράζονται” ένα αντίγραφο της
- Αν ένα αντικείμενο αλλάξει την τιμή μιας στατικής μεταβλητής μέλους, η αλλαγή γίνεται σε όλα τα αντικείμενα της κλάσης
- Χρήσιμες για την “παρακολούθηση”
  - του πλήθους των αντικειμένων που έχουν δημιουργηθεί μια δεδομένη στιγμή
  - του πλήθους των κλήσεων μιας συνάρτησης μέλους (πόσες φορές έχει κληθεί)

Στατικές συναρτήσεις μέλη

- Συναρτήσεις μέλη μπορεί να είναι στατικές
  - εάν δεν έχουν πρόσβαση σε μη στατικές μεταβλητές μέλη του αντικειμένου
  - και “πρέπει” να είναι μέλη της κλάσης

- Τέτοιες συναρτήσεις μπορούν να κληθούν:
  - Είτε από αντικείμενα:
    π.χ., `myObject.getTurn();`
  - Είτε “από την κλάση”:
    π.χ., `Server::getTurn();`

- Μπορούν να έχουν πρόσβαση σε στατικές μεταβλητές μέλη ή να καλέσουν άλλες στατικές συναρτήσεις.

Hands-on ενότητα 3

Υλοποίηση της κλάσης Server

Σύνοψη

- Κατασκευαστές: Αρχικοποιούν τις μεταβλητές μέλη της κλάσης
  - Καλούνται όταν δηλώνονται αντικείμενα
  - Έχουν το ίδιο όνομα με αυτό της κλάσης

- Ο προκαθορισμένος κατασκευαστής δεν έχει παραμέτρους
  - Πρέπει πάντα να ορίζεται

- Οι μεταβλητές μέλη μιας κλάσης μπορεί να είναι αντικείμενα άλλων κλάσεων
  - Η αρχικοποίηση τους γίνεται στο τμήμα αρχικοποίησης

Σύνοψη

- Σταθερές παράμετροι κλήσης με αναφορά
  - Πιο αποδοτικές από τις κλήσεις με τιμή

- `Inline` υλοποιήσεις σύντομων συναρτήσεων
  - Μπορούν να βελτιώσουν την αποτελεσματικότητα του κώδικα

- Στατικές μεταβλητές μέλη
  - Κοινή χρήση από όλα τα αντικείμενα μιας κλάσης