ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 6: C++ ΚΛΑΣΕΙΣ, ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ, ΠΟΛΥΜΟΡΦΙΣΜΟΣ

Κλάσεις

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Τάξεις / Κλάσεις

Ορισμός Δομών-Structure

- Δομές
  - Καθορίζουν τύπους που προκύπτουν από τη συνάθροιση μελών άλλων τύπων

```
struct Time {
        int hour;
        int minute;
        int second;
 };
```

- Ονοματολογία μελών σε δομή Structure
  - Στην ίδια **`struct`**: πρέπει να δίνονται μοναδικά ονόματα
  - Σε διαφορετικές **`structs`**: μπορούν τα ονόματα να διαμοιράζονται
- Ο ορισμός **`struct`** τελειώνει πάντα με ερωτηματικό

Ορισμός Δομών-Structure

- Αυτό-αναφερόμενη δομή-struct
  - Τα μέλη μιας Structure δε μπορεί να είναι εκφάνσεις (instance) εμφωλευμένης **`struct`**
  - Τα μέλη μιας Structure μπορούν να είναι δείκτης σε μια instance εμφωλευμένης **`struct`** (αυτό-αναφερόμενη structure)
    - Χρήσιμο για διασυνδεδεμένες λίστες, ουρές, στοίβες και δένδρα
- **`struct`**
  - Δημιουργεί έναν νέο τύπο δεδομένων που χρησιμοποιείται για να δηλώσει κανείς μεταβλητές
  - Δηλώνονται όπως και οι λοιπές μεταβλητές άλλων τύπων
  - Παραδείγματα:
    - **`Time timeObject;`**
    - **`Time timeArray[ 10 ];`**
    - **`Time *timePtr;`**
    - **`Time &timeRef = timeObject;`**

Προσπέλαση μελών Structure

- Τελεστές προσπέλασης μελών
  - Τελεστής τελεία (**.**) για μέλη δομής-structure και τάξης
  - Τελεστής βέλος (**->**) για μέλη δομής-structure και τάξης μέσω δείκτη σε αντικείμενο
  - Εκτύπωση μέλους **hour** του **timeObject**:

```
cout << timeObject.hour;
```

Ή

```
timePtr = &timeObject;
cout << timePtr->hour;
```

  - **timePtr->hour** είναι όμοιο με **(*timePtr).hour**
    - Οι παρενθέσεις απαιτούνται
      - ***** έχει μικρότερη προτεραιότητα έναντι του **.**

Τάξη - Class

- Τάξεις
  - Αντικείμενα Model
    - Χαρακτηριστικά - Attributes (μέλη δεδομένα)
    - Συμπεριφορές - Behaviors (μέλη συναρτήσεις)
  - Ορίζεται με τη λέξη-κλειδί **`class`**
  - Μέλη συναρτήσεις
    - Μέθοδοι
    - Καλούνται σε απάντηση μηνύματος
- Καθοριστές πρόσβασης σε μέλη
  - **`Δημόσια – public:`**
    - Πρόσβαση σε οποιοδήποτε αντικείμενο της τάξης εντός περιοχής -scope
  - **`Ιδιωτικό – private:`**
    - Πρόσβαση μόνο από μέλη συναρτήσεις της τάξης
  - **`Προστατευόμενο – protected:`**
    - Πρόσβαση από παράγωγες τάξεις

Τάξη - Class

- Συνάρτηση Constructor
  - Ειδικό μέλος συνάρτηση
    - Αρχικοποιεί τα δεδομένα
    - Έχει το ίδιο όνομα με την τάξη
  - Καλείται όταν παράγεται το αντικείμενο
  - Μπορεί να υπάρχουν πολλοί constructors
    - Υπερφόρτωση συναρτήσεων Function
    - Υπάρχει default, καλύπτεται με ορισμό χωρίς arguments
  - Δεν έχει/επιστρέφει κάποιο type (εξ ορισμού void)
  - Public ή private

Παράδειγμα

```
class Time {
  public:
    Time();
    void setTime( int, int, int );
  private:
    int hour;
    int mimute;
    int second;
};

void Time::setTime( int h, int m, int s ) {
  hour =   ( h >= 0 && h < 24 ) ? h : 0;
  minute = ( m >= 0 && m < 60 ) ? m : 0;
  Second = ( s >= 0 && s < 60 ) ? s : 0;
}

int main(){
  Time t;
  t.setTime(2,20,45)
}
```

Ομαδοποίηση των public / private μελών (δεν υποστηρίζεται στην JAVA)

Δυνατότητα υλοποίησης μεθόδων εκτός της κλάσης.

Χρήση τελεστή :: για να αναφερθούμε στην κλάση έξω από αυτήν.

Μέθοδος Καταστροφής (Destructor)

- Destructors
  - Έχουν το ίδιο όνομα με την τάξη
    - Προηγείται το σημάδι μαθηματικής άρνησης (tilde) (~)
  - Χωρίς παραμέτρους
  - Δε μπορεί να υπερφορτωθεί
  - Public

Πλεονεκτήματα

- Πλεονεκτήματα χρήσης τάξεων
  - Απλοποιούν τον προγραμματισμό
  - Διεπαφές
    - Κρύβουν την υλοποίηση
  - Επαναχρησιμοποίηση Κώδικα
    - Σύνθεση (συνάθροιση) - Composition (aggregation)
      - Αντικείμενα τάξης μπορούν να περιληφθούν ως μέλη άλλων τάξεων
    - Κληρονομικότητα
      - Νέες τάξεις προκύπτουν από παλαιές

Εμβέλεια τάξης και προσπέλαση μελών

- Εμβέλεια τάξης
  - Μέλη δεδομένα και συναρτήσεις
  - Εντός εμβέλειας
    - Μέλη τάξης
      - Άμεσα προσπελάσιμα από όλα τα μέλη συναρτήσεις
      - Αναφορά με το όνομα
  - Εκτός εμβέλειας τάξης
    - Αναφορά με handles
      - Όνομα αντικειμένου, αναφορά στο αντικείμενο, δείκτης στο αντικείμενο

Εμβέλεια τάξης και προσπέλαση μελών

- Εμβέλεια συνάρτησης
  - Μεταβλητές δηλώνονται σε συναρτήσεις μέλη
  - Είναι γνωστές μόνο στη συνάρτηση
  - Μεταβλητές με ίδιο όνομα με μεταβλητές εμβέλειας τάξης
    - Η εμβέλεια της μεταβλητής τάξης «κρύβεται»
      - Προσπέλαση με τελεστή καθορισμού εμβέλειας (`::`)

        `ClassName::classVariableName`
  - Οι μεταβλητές είναι γνωστές στις συναρτήσεις που ορίζονται
  - Οι μεταβλητές καταστρέφονται μετά την ολοκλήρωση της συνάρτησης

Εμβέλεια τάξης και προσπέλαση μελών

- Τελεστές για προσπέλαση μελών τάξης
  - Ίδια με των **`struct`**s
  - Επιλογή με τελεία (`.`)
    - Αντικείμενο
    - Αναφορά σε αντικείμενο
  - Επιλογή με βέλος (`->`)
    - Δείκτες

Παράδειγμα

```
Time t1;

Time *timePtr = &t1;

Time &timeRef = t1;

t1.hour = 8;
```

← Χρήση τελείας για την επιλογή μέλους δεδομένου από το αντικείμενο

```
timeRef.hour = 2;
```

← Χρήση τελείας για την επιλογή μέλους στην αναφορά timeRef στο αντικείμενο.

```
timePtr->hour = 3;
```

← Χρήση βέλους για το δείκτη timePtr στο αντικείμενο

Έλεγχος πρόσβασης στα μέλη

- Επίπεδα πρόσβασης
  - **`private`**
    - Default
    - Προσβάσιμο σε μέλη συναρτήσεις και **`friend`**s
  - **`public`**
    - Προσβάσιμο σε κάθε συνάρτηση που χειρίζεται αντικείμενο της τάξης
  - **`protected`**

Μέθοδοι Get/Set

- Συνάρτηση Set
  - Πραγματοποιεί ελέγχους εγκυρότητας προτού γίνουν μετατροπές σε δεδομένα **`private`**
  - Ειδοποιεί αν υπάρχουν λανθασμένες τιμές
- Συνάρτηση Get
  - Ελέγχει τη μορφή των δεδομένων που επιστρέφονται

Προσοχή: Επιστροφή αναφοράς σε δεδομένα private

- Αναφορά σε αντικείμενο
  - Alias for name of object
  - Lvalue
    - Μπορεί να λάβει τιμή σε ανάθεση
      - Αλλάζει το αρχικό αντικείμενο
- Επιστροφή αναφοράς
  - **`public`** συναρτήσεις μπορούν να επιστρέφουν μη-**`const`** αναφορές σε δεδομένα **`private`**
    - Κατά συνέπεια ο χρήστης μπορεί να τροποποιεί δεδομένα **`private`**

Default Ανάθεση

- Ανάθεση αντικειμένων
  - Τελεστής ανάθεσης (=)
    - Μπορεί να αναθέσει σε ένα αντικείμενο ένα άλλο ίδιου τύπου
    - Default ανάθεση
      - Κάθε δεξιό μέλος ανατίθεται ανεξάρτητα στο αριστερό μέλος
- Περνώντας και επιστρέφοντας αντικείμενα
  - Αντικείμενα περνούν ως παράμετροι συνάρτησης
  - Αντικείμενα επιστρέφονται από συναρτήσεις
  - Default πέρασμα με τιμή
    - Αντίγραφο του αντικείμενου που περνάει, επιστρέφεται
      - Αντίγραφο του constructor

        Αντίγραφο των αρχικών τιμών σε νέο αντικείμενο

const (Σταθερά)

- Η αρχή της ελάχιστης πρόσβασης
  - Επιτρέπουμε πρόσβαση για τροποποιήσεις μόνο στα απαραίτητα αντικείμενα
- **`const`**
  - Ορίζει αντικείμενο που δε τροποποιείται
  - Δίνει Compiler error

Χρήση Const

- **`const`** μέθοδοι
  - Οι μέθοδοι αντικειμένων **`const`** πρέπει να είναι και αυτές **`const`**
    - Δε μπορεί να τροποποιούν αντικείμενα
  - Ορίζουμε ως **`const`** σε
    - Πρωτότυπο
      - Μετά τη λίστα παραμέτρων
    - Δηλώσεις
      - Πριν την αρχή του αριστερού αγκίστρου

Αρχικοποίηση

- Αρχικοποίηση αντικειμένου
  - Αρχικοποίηση με member initializer syntax
    - Μπορεί να χρησιμοποιηθεί
      - Με όλα τα μέλη δεδομένων
    - Πρέπει να χρησιμοποιηθεί
      - **Για τα μέλη const**
      - Για όλες τις αναφορές μεταβλητών

Παράδειγμα

```
class myDate{
  public:
    myDate(int d, int m, int y) :day(d), month(m), year(y) { }
    ~myDate(){
       cout << "date object destroyed";
    }
    int getDay () const{     // Δεν τροποποιεί το αντικείμενο
      return day;
    }
    void setDay (int d){
      day = (d>0 && d<32)? d : 1;
    }
  private:
    int day;
    int month;
    const int year;
};
int main(){
  myDate d1(2,11,2001);
  //cout << d1.day; Δεν μπορούμε γιατί είναι private μέλος
  cout << d1.getDay();
  const myDate d2(4,12,2011);
  //d2.setDay(3); Compiler error γιατι η μέθοδος δεν είναι const
  int day2 = d2.getDay();
  // Μπορούμε γιατι η μέθοδος έχει δηλωθεί const
}
```

αρχικοποίηση

Η const μεταβλητή year δεν μπορεί να τροποποιηθεί για κανένα αντικείμενα. Παίρνει τιμή μόνο κατά την αρχικοποίηση με τον παραπάνω τρόπο.

Το const αντικείμενο d2 δεν μπορεί να τροποποιηθεί. Εκτός του δημιουργού, μπορούμε να καλέσουμε μόνο const μεθόδους της κλάσης

Σύνθεση/ Composition:
Αντικείμενα ως μέλη τάξης

- Σύνθεση/ Composition
  - Μία τάξη έχει αντικείμενα άλλης τάξης ως μέλη
- Κατασκευή αντικειμένων
  - Τα μέλη αντικείμενα δημιουργούνται με τη σειρά που δηλώνονται
    - Δεν ακολουθείται η σειρά του constructor
    - Δημιουργούνται πριν από τα αντικείμενα της τάξης που τα χρησιμοποιεί

Παράδειγμα - Σύνθεση

```
class myEvent{
  public:
    myEvent(myDate str, myDate endd): startdate(str),
                                      enddate(endd){};
    int getDuration() const;
  private:
    myDate startdate;
    myDate enddate;
};
```

friend Συναρτήσεις και Τάξεις

- **`friend`** συναρτήσεις
  - Ορίζονται εκτός εμβέλειας της τάξης
  - Έχουν πρόσβαση σε non-public members
- Δήλωση **`friend`**s
  - Συνάρτηση
    - Προηγείται το keyword **`friend`**
  - Όλες οι συναρτήσεις της τάξης **`classTwo`** ως **`friend`**s της τάξης **`classOne`**
    - Βάζουμε τη δήλωση της μορφής

      ```
      friend class classTwo;
      ```

      στον ορισμό της **`classOne`**

friend Συναρτήσεις και Τάξεις

- Ιδιότητες
  - Μπορεί να δοθεί όχι να ανακληθεί
    - τάξη **B** **friend** της τάξης **A**
      - Η τάξη **A** πρέπει να δηλώσει την τάξη **B** ως **friend**
  - Όχι συμμετρική
    - τάξη **B** **friend** της τάξης **A**
    - τάξη **A** όχι απαραίτητα **friend** της τάξης **B**
  - Όχι μεταβατική
    - τάξη **A** **friend** της **B**
    - τάξη **B** **friend** της **C**
    - τάξη **A** όχι απαραίτητα **friend** της **C**

Παράδειγμα

```
class myDate{
  friend int yearDifference(myDate, myDate);
  public:
    myDate(int d, int m, int y):day(d), month(m), year(y) {}
  private:
    int day; int month; int year;
};

int yearDifference(myDate dt1, myDate dt2){
  return dt2.year - dt1.year;
}

int main(){
  myDate d1(2,11,2001);
  myDate d2(3,4,2015);
  cout << yearDifference(d1,d2);
}
```

Η friend συνάρτηση δεν είναι μέλος της κλάσης.

- Δεν χρειάζεται ο τελεστής :: στην δήλωση
- Δεν καλείται από αντικείμενα της κλάσης

Απευθείας πρόσβαση στα private μέλη της friend κλάσης.

Χρήση του this

- **`this`**
  - Επιτρέπει στο αντικείμενο να έχει πρόσβαση στη δική του διεύθυνση
  - Ο τύπος του δείκτη **`this`** εξαρτάται από:
    - Τύπο του αντικειμένου
    - Αν η συνάρτηση είναι **`const`**
    - Για τις non-**`const`** συναρτήσεις **`Employee`**
      - **`this`** έχει τύπο **`Employee * const`**

        Constant δείκτη σε non-const **`Employee`** αντικείμενο
    - Για τις **`const`** συναρτήσεις **`Employee`**
      - **`this`** έχει τύπο **`const Employee * const`**

        Constant δείκτη σε constant **`Employee`** αντικείμενο

Χρήση του this

- Σειριακή κλήση συναρτήσεων
  - Πολλαπλές συναρτήσεις καλούνται με μία δήλωση
  - Η συνάρτηση επιστρέφει δείκτη αναφοράς στο ίδιο το αντικείμενο

```
{ return *this; }
```

  - Οι συναρτήσεις που δεν επιστρέφουν αναφορές πρέπει να κληθούν τελευταίες

Διαχείριση Δυναμικής Μνήμης

- Διαχείριση δυναμικής μνήμης
  - Ελέγχει τη διανομή μνήμης
  - Με χρήση των τελεστών **`new`** και **`delete`**
    - include standard header **`<new>`**

Διαχείριση Δυναμικής Μνήμης

- Έστω

```
Time *timePtr;
timePtr = new Time;
```

- **Τελεστής new**
  - Δημιουργεί αντικείμενα κατάλληλου μεγέθους για τον τύπο **Time**
    - Δίνει λάθος αν δεν υπάρχει χώρος στη μνήμη
  - Επιστρέφει δείκτη στον συγκεκριμένο τύπο
- Με αρχικοποίηση

```
double *ptr = new double( 3.14159 );
Time *timePtr = new Time( 12, 0, 0 );
```

- Δήλωση πίνακα

```
int *gradesArray = new int[ 10 ];
```

Διαχείριση Δυναμικής Μνήμης

- Έστω

```
delete timePtr;
```

- Τελεστής **`delete`**
  - Καλεί το destructor
  - Η μνήμη μπορεί να χρησιμοποιηθεί με άλλα αντικείμενα
- Deallocating arrays

```
delete [] gradesArray;
```

  - Απελευθερώνει το array στο οποίο δείχνει το **`gradesArray`**
  - Αν είναι δείκτης σε array αντικειμένων
    - Καλείται πρώτα ο destructor για κάθε αντικείμενο του array
    - Μετά απελευθερώνει τη μνήμη

Μεταβλητές Τάξης (Static)

- **`static`** μεταβλητή τάξης
  - Δεδομένα διαθέσιμα σε όλη την τάξη
    - Ιδιότητα της τάξης, όχι συγκεκριμένου αντικειμένου της τάξης
  - Αποδοτικό όταν απλά ένα αντίγραφο της τάξης είναι αρκετό
    - Μόνο η μεταβλητή **`static`** πρέπει να ενημερώνεται
  - Μπορεί να μοιάζει με global, αλλά έχει εμβέλεια στην τάξη
  - Αρχικοποιείται μια μόνο φορά
  - Υπάρχει ακόμη και χωρίς αντικείμενο

Παράδειγμα

```
class myDate{
  public:
    myDate(int d, int m, int y) :day(d), month(m), year(y) {
      if (y>maximumYear)
        maximumYear=y;
    }
    int static getMaxYear(){ return maximumYear ;}

  private:
    int day; int month; int year;
    static int maximumYear;
};

int myDate::maximumYear=0;

int main(){
  cout<< myDate::getMaxYear()<<endl;
  myDate d1(2,11,2001); myDate d2(3,4,2015);
  cout<< myDate::getMaxYear();
}
```

Static μέθοδος που επιστρέφει την static μεταβλητή

Αρχικοποίηση έξω από την κλάση.

Κλήση και χωρίς να υπάρχει ακόμα αντικείμενο.

Πρόσθετο Υλικό

- Μελετήστε και τα παραδείγματα από τα **Κεφάλαια 3, 9** 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

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

- Οι διαφάνειες βασίζονται στο βιβλίο «C++ How to Program, 8th Edition, Harvey M. Deitel, Paul J. Deitel, Prentice Hall.»