Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Αντικειμενοστρεφής Προγραμματισμός (C++)

Ενότητα 1: Κλάσεις

Καθηγήτρια Κλειώ Σγουροπούλου

Κλάσεις 1/4

- Η κλάση είναι μια λογική οντότητα οργάνωσης **δεδομένων** και **λειτουργιών** στην ίδια δομή. Η δήλωσή της γίνεται με τη λέξη κλειδί **class**, του οποίου η λειτουργικότητα είναι παρόμοια με αυτή του **struct**, με τη διαφορά ότι στην **struct** δεν μας δίνεται η δυνατότητα ορισμού λειτουργιών, παρά μόνο δεδομένων.

```
class class_name
{
   permission_label_1:
     member1;
   permission_label_2:
     member2;
   ...
} object_name;
```

Κλάσεις 2/4

- Οι λέξεις-κλειδιά ελέγχου πρόσβασης (permission labels), είναι
  - **private:**
  - **public:**
  - **protected:**
- Αναφέρονται στο επίπεδο πρόσβασιμότητας που ορίζουμε για κάθε μέλος της κλάσης:
  - τα **ιδιωτικά** (**private**) **μέλη** είναι προβάσιμα μόνο από άλλα μέλη στην ίδια κλάση ή από τις φιλικές (friend) κλάσεις.
  - Τα **προστατευόμενα** (**protected**) **μέλη** είναι προσβάσιμα από μέλη της ίδιας κλάσεις και των φιλικών κλάσεων. Επιπλέον δε, είναι προσβάσιμα από τις παράγωγες (derived) κλάσεις.
  - Τα **δημόσια** (**public**) **μέλη** είναι προσβάσιμα από οπουδήποτε η κλάση είναι ορατή.

Κλάσεις 3/4

```
#include <iostream.h>

class CRectangle {
    int x, y;
  public:
    void set_values (int,int);
    int area (void) {return (x*y);}
};

void CRectangle::set_values (int a, int b) {
  x = a;
  y = b;
}

int main () {
  CRectangle rect;
  rect.set_values (3,4);
  cout << "area: " << rect.area();
}
```

Κλάσεις 4/4

- Αν πριν τη δήλωση ενός μέλους δεν αφέρεται **ρητά** η προσβασιμότητα τότε θεωρείται ότι είναι **private**
- Ο τελεστής εμβέλειας :: χρησιμοποιείται για τη δήλωση μελών έξω από την κλάση. Στην περίπτωση που μια συνάρτηση οριστεί εξ' ολοκλήρου μέσα στην κλάση, τότε θεωρείται αυτόματα **inline.** Αν οριστεί εξωτερικά, θεωρείται κανονική συνάρτηση.
- Η δήλωση μεταβλητών της κλάσης (**στιγμιότυπα** ή **αντικείμενα**) γίνεται με το όνομα της κλάσης ώς τύπου δεδομένων
  - `CRectangle rect;`
- Η προσπέλαση των μελών μιας κλάσης για ένα συγκεκριμένο αντικείμενο γίνεται μέσω του τελεστή .
  - `rect.set_values (3,4);`

Κατασκευαστές 1/2

- **Κατασκευαστές**
  - Κατά τη διεργασία δημιουργίας τους, τα αντικείμενα είναι απαραίτητο να αρχικοποιούν τις μεταβλητές μέλη τους ή να αναθέτουν δυναμική μνήμη ώστε να είναι πλήρως λειτουργικά και να αποφεύγεται η επιστροφή απροσδιόριστων τιμών κατά την εκτέλεση σχετικών λειτουργιών.
  - Για το σκοπό αυτό η κλάση περιέχει μια ειδική συνάρτηση: τον **κατασκευαστή** (**constructor**). Ο κατασκευαστής δηλώνεται ως μια συνάρτηση-μέλος, η οποία **έχει ακριβώς το ίδιο όνομα με την κλάση.** Ο κατασκευαστής καλείται αυτόματα κάθε φορά που δημιουργείται ένα νέο στιγμιότυπο της κλάσης
  - Οι κατασκευαστές δεν επιστρέφουν ποτέ τιμή.

Κατασκευαστές 2/2

```
#include <iostream.h>
class CRectangle {
    int width, height;
  public:
    CRectangle (int,int);
    int area (void) {return (width*height);}
};

CRectangle::CRectangle (int a, int b) {
  width = a;
  height = b;
}

int main () {
  CRectangle rect (3,4);
  CRectangle rectb (5,6);
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
}
```

Καταστροφείς 1/3

- Καταστροφείς
  - Ο **καταστροφέας** (**destructor**) επιτελεί την αντίστροφη λειτουργία από αυτή του κατασκευαστή. Καλείται αυτόματα όταν ένα αντικείμενο εξέρχεται από τη μνήμη, είτε διότι σταμάτησε να υφίσταται λόγος ύπαρξής του (π.χ. Ήταν δηλωμένο ως τοπικό αντικείμενο σε μια συνάρτηση της οποίας η εκτέλεση ολοκληρώθηκε), είτε επειδή είναι ένα δυναμικά ορισμένο αντικείμενο το οποίο αποδεσμεύεται μέσω του τελεστή **delete.**
  - Ο καταστροφέας δηλώνεται ως μια συνάρτηση-μέλος, η οποία **έχει ακριβώς το ίδιο όνομα με την κλάση με μία τίλντα (~) μπροστά.**
  - Οι καταστροφείς δεν επιστρέφουν ποτέ τιμή.

Καταστροφείς 2/3

```
#include <iostream.h>

class CRectangle {
    int *width, *height;
  public:
    CRectangle (int,int);
    ~CRectangle ();
    int area (void) {return (*width * *height);}
};

CRectangle::CRectangle (int a, int b) {
  width = new int;
  height = new int;
  *width = a;
  *height = b;
}
```

Καταστροφείς 3/3

```
CRectangle::~CRectangle () {
  delete width;
  delete height;
}

int main () {
  CRectangle rect (3,4), rectb (5,6);
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
  return 0;
}
```

Υπερφόρτωση Κατασκευαστών 1/3

- Όπως και κάθε άλλη συνάρτηση, ο κατασκευαστής μπορεί να υπερφορτωθεί (με το ίδιο όνομα και διαφορετικές παραμέτρους)
- Όταν δηλώνουμε μια κλάση χωρίς να έχουμε ορίσει κατασκευαστή, ο μεταγλωττιστής αυτόματα θεωρεί την ύπαρξη δύο κατασκευαστών:
  - του **προκαθορισμένου** (**default**) **κατασκευαστή:** χωρίς παραμέτρους και με κενό σώμα. Δεν κάνει τίποτα.
  - του **κατασκευαστή αντιγράφου** (**copy**): με μια παράμετρο τύπου αντικειμένου της συγκεκριμένης κλάσης και λειτουργία ανάθεσης σε κάθε μη στατικό μέλος της κλάσης ενός αντιγράφου του αντικειμένου-παραμέτρου

Υπερφόρτωση Κατασκευαστών 2/3

```
class CExample {
  public:
    int a,b,c;
    void multiply (int n, int m) { a=n; b=m; c=a*b; };
  };

CExample::CExample () { };

CExample::CExample (const CExample& rv)
{
  a=rv.a;  b=rv.b;  c=rv.c;
}
```

- Οι δυο αυτοί κατασκευαστές υφίστανται μόνο εάν δεν έχει δηλωθεί στο πρόγραμμα κανένας άλλος κατασκευαστής

Υπερφόρτωση Κατασκευαστών 3/3

```
#include <iostream.h>

class CRectangle {
    int width, height;
  public:
    CRectangle ();
    CRectangle (int,int);
    int area (void) {return (width*height);}
};

CRectangle::CRectangle () {
  width = 5; height = 5;}

CRectangle::CRectangle (int a, int b) {
  width = a; height = b;}

int main () {
  CRectangle rect (3,4);
  CRectangle rectb; //CRectangle rectb(); Λάθος!
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;}
```

Δείκτες σε κλάσεις 1/3

- Δηλώνοντας μια κλάση δημιουργούμε έναν τύπο δεδομένων και συνεπώς μπορούμε να χρησιμοποιούμε το όνομα της κλάσης για τον ορισμό δεικτών του συγκεκριμένου τύπου.
  - `CRectangle * prect;`
- Για την αναφορά σε ένα μέλος αντικειμένου που δεικτοδοτείται μέσω ενός δείκτη χρησιμοποιείται ο τελεστής →

Δείκτες σε κλάσεις 2/3

```
#include <iostream.h>

class CRectangle {
    int width, height;
  public:
    void set_values (int, int);
    int area (void) {return (width * height);}
};

void CRectangle::set_values (int a, int b) {
  width = a;
  height = b;
}
```

Δείκτες σε κλάσεις 3/3

```
int main () {
  CRectangle a, *b, *c;
  CRectangle * d = new CRectangle[2];
  b= new CRectangle;
  c= &a;
  a.set_values (1,2);
  b->set_values (3,4);
  d->set_values (5,6);
  d[1].set_values (7,8);
  cout << "a area: " << a.area() << endl;
  cout << "*b area: " << b->area() << endl;
  cout << "*c area: " << c->area() << endl;
  cout << "d[0] area: " << d[0].area() << endl;
  cout << "d[1] area: " << d[1].area() << endl;
  return 0;
}
```

```
a area: 2
*b area: 12
*c area: 2
d[0] area: 30
d[1] area: 56
```

Σύνοψη τελεστών για λειτ. δεικτών

***x** δείκτης στο x

&x διεύθυνση του x

x.y μέλος y του αντικειμένου x

(*x).y μέλος y του αντικειμένου που δεικτοδοτείται από τον x

x->y (ισοδύναμο με το προηγούμενο)

x[0] το πρώτο αντικείμενο που δεικτοδοτείται από τον x

x[n] το n+1 αντικείμενο που δεικτοδοτείται από τον x

Υπερφόρτωση τελεστών 1/6

- Η C++ υποστηρίζει τη χρήση των εγγενών τελεστών της γλώσσας μεταξύ κλάσεων.
- Αντικείμενα κλάσεων ή σύνθετων τύπων μπορούν να δεχτούν τελεστές, τους οποίους μπορούμε να τους τροποποιήσουμε ως προς το είδος λειτουργίας που θα επιτελούν. Τελεστές που δέχονται υπερφόρτωση είναι:

```
+    -    *    /    =    <    >    +=   -=   *=
 /=   <<   >>  <<=  >>=  ==   !=   <=   >=   ++
--   %    &    ^    !    | ~    &=   ^=   |=   &&
          || %= []   ()   new  delete
```

Υπερφόρτωση τελεστών 2/6

- Για να υπερφορτώσουμε ένα τελεστή πρέπει να υλοποιήσουμε μια συνάρτηση-μέλος κλάσης με όνομα το σύμβολο του τελεστή τον οποίο επιθυμούμε να υπερφορτώσουμε:
  - `type operator sign (parameters);`

Υπερφόρτωση τελεστών 3/6

- Υλοποιήστε μια κλάση, η οποία να υλοποιεί τις πράξεις της πρόσθεσης και του πολλαπλασιασμού μεταξύ μιγαδικών αριθμών

```
class Complex
{
};

Complex::Complex(              )
{
}

Complex Complex::operator + (                  )
{
}

Complex Complex::operator * (                  )
{
}
```

Υπερφόρτωση τελεστών 4/6

- Η γλώσσα υποστηρίζει τον αυτόματο ορισμό του τελεστή ανάθεσης (**=**) για τα αντικείμενα οποιασδήποτε κλάσης.
- Η λειτουργία του τελεστή ανάθεσης μεταξύ αντικειμένων έγκειται στην αντιγραφή ολόκληρου του περιεχομένου (μη στατικών μελών δεδομένων) μεταξύ αντικειμένων της ίδιας κλάσης.
- Η λειτουργία του τελεστή ανάθεσης μπορεί να υπερφορτωθεί.

Υπερφόρτωση τελεστών 5/6

- Μερικοί τελεστές κλάσης μπορούν να υπερφορτωθούν με δύο τρόπους:
  - ως συναρτήσεις-μέλη μιας κλάσης,
  - ως καθολικές συναρτήσεις.
- Ο δεύτερος τρόπος απαιτεί εξασφάλιση προϋποθέσεων πρόσβασης στα μέλη-δεδομένα της κλάσης (**ποιές;**)
- Πίνακας νόμιμων ορισμών συναρτήσεων υπερφόρτωσης τελεστών

Υπερφόρτωση τελεστών 6/6

Expression	Operator (8)	Function member	Global function
@a	+ - * & ! ~ ++ —	A::operator@()	operator@(A)
a@	++ —	A::operator@(int)	operator@(A, int)
a@b	+ -* / % ^ | <> == != <= => << >> && || ,	A::operator@(B)	operator@(A, B)
a@b	= += -= *= /= %= ^= &= |= <<= >>= []	A::operator©(B)	-
a(b, c...)	()	A::operator()(B, C...)	-
a->b	->	A::operator->()	-

Η λέξη-κλειδί `this` 1/2

- Το **`this`** αναπαριστά τη διεύθυνση του αντικειμένου μιας κλάσης στη μνήμη. Είναι ένας δείκτης του οποίου η τιμή είναι η ενίοτε διεύθυνση του αντικειμένου
- Μπορεί να χρησιμοποιηθεί για τον έλεγχο του αν μια παράμετρος που περνάται σε μια συνάρτηση μέλος ενός αντικειμένου είναι το ίδιο το αντικείμενο (#@#????????)

Η λέξη-κλειδί this 2/2

```
#include <iostream.h>

class CDummy {
  public:
    int isitme (CDummy& param);
};

int CDummy::isitme (CDummy& param)
{
  if (&param == this) return 1;
  else return 0;
}

int main () {
  CDummy a;
  CDummy* b = &a;
  if (b->isitme(a)) cout << "yes, &a is b";
  return 0;
}
```

Στατικά μέλη 1/2

- Μια κλάση μπορεί να περιέχει στατικά μέλη, είτε δεδομένα είτε συναρτήσεις
- Τα στατικά μέλη-δεδομένα είναι γνωστά και ως μεταβλητές κλάσης, διότι το περιεχόμενό τους δεν εξαρτάται από τη δημιουργία αντικειμένων της κλάσης
- Ένα στατικό μέλος-δεδομένο έχει μια ενιαία τιμή για όλα τα αντικείμενα της κλάσης
- **Τυπική χρήση:** μετρητής αντικειμένων μιας κλάσης
- Τα στατικά μέλη δεδομένα αρχικοποιούνται εκτός κλάσης
- Επιτρέπεται αναφορά τους είτε μέσω αντικειμένου, είτε μέσω κλάσης
  - `cout << a.n;`
  - `cout << CDummy::n;`

Στατικά μέλη 2/2

```
#include <iostream.h>
class CDummy {
  public:
    static int n; //ορισμός
    CDummy () { n++; };
    ~CDummy () { n--; };
};

int CDummy::n=0; //αρχικοποίηση εκτός κλάσης

int main () {
  CDummy a;
  CDummy b[5];
  CDummy * c = new CDummy;
  cout << a.n << endl;
  delete c;
  cout << CDummy::n << endl;
  return 0;
}
```

7

6

Συναρτήσεις `friend` 1/3

- Σε περιπτώσεις όπου τα μέλη μιας κλάσης είναι **private** ή **protected,** δεν είναι δυνατή η προσπέλασή τους από στοιχεία εκτός της κλάσης αυτής.
- **Εξαίρεση** αποτελούν εξωτερικές συναρτήσεις, οι οποίες έχουν χαρακτηριστεί ως **friend** (με χρήση της ομώνυμης λέξης-κλειδιού).
- Στην κλάση, στα μέλη της οποίας επιθυμούμε να έχει πρόσβαση μια εξωτερική συνάρτηση, πρέπει να ενσωματώσουμε πρωτότυπο της εξωτερικής συνάρτησης με το χαρακτηρισμό **friend.**
- Τυπική χρήση: διεξαγωγή λειτουργιών μεταξύ διαφορετικών κλάσεων.

Συναρτήσεις friend 2/3

```
#include <iostream.h>
class CRectangle {
    int width, height;
  public:
    void set_values (int, int);
    int area (void) {return (width * height);}
    friend CRectangle duplicate (CRectangle);
};

void CRectangle::set_values (int a, int b) {
  width = a;
  height = b;}

CRectangle duplicate (CRectangle rectparam)
{
  CRectangle rectres;
  rectres.width = rectparam.width*2;
  rectres.height = rectparam.height*2;
  return (rectres);}
```

Συναρτήσεις `friend` 3/3


```
int main () {
  CRectangle rect, rectb;
  rect.set_values (2,3);
  rectb = duplicate (rect);
  cout << rectb.area();
}
```

Κλάσεις `friend` 1/2

- Εκτός από συναρτήσεις, μπορούμε να ορίσουμε και κλάσεις **`friend`**, επιτρέποντας με τον τρόπο αυτό σε κάποια κλάση να προσπελαύνει μέλη μιας άλλης κλάσης.

```
#include <iostream.h>

class CSquare;

class CRectangle {
    int width, height;
  public:
    int area (void)
      {return (width * height);}
    void convert (CSquare a);
};
```

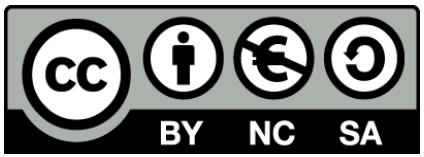
Κλάσεις friend 2/2


```
class CSquare {
  private:
    int side;
  public:
    void set_side (int a)
      {side=a;}
    friend class CRectangle;};

void CRectangle::convert (CSquare a) {
  width = a.side;
  height = a.side;}

int main () {
  CSquare sqr;
  CRectangle rect;
  sqr.set_side(4);
  rect.convert(sqr);
  cout << rect.area();
  return 0;}
```

ΤΕΛΟΣ ΕΝΟΤΗΤΑΣ



Σημειώματα

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Δυτικής Αττικής, Κλειώ Σγουροπούλου 2020. Κλειώ Σγουροπούλου. «Αντικειμενοστραφής Προγραμματισμός-Θ. Ενότητα 1: Περιήγηση στη C++». Έκδοση: 1.0. Αθήνα 2020. Διαθέσιμο από τη δικτυακή διεύθυνση: eclass.uniwa.gr.

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό. Οι όροι χρήσης των έργων τρίτων επεξηγούνται στη διαφάνεια «Επεξήγηση όρων χρήσης έργων τρίτων».

Τα έργα για τα οποία έχει ζητηθεί άδεια αναφέρονται στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Επεξήγηση όρων χρήσης έργων τρίτων

©	Δεν επιτρέπεται η επαναχρησιμοποίηση του έργου, παρά μόνο εάν ζητηθεί εκ νέου άδεια από το δημιουργό.
διαθέσιμο με άδεια CC-BY	Επιτρέπεται η επαναχρησιμοποίηση του έργου και η δημιουργία παραγώγων αυτού με απλή αναφορά του δημιουργού.
διαθέσιμο με άδεια CC-BY-SA	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού, και διάθεση του έργου ή του παράγωγου αυτού με την ίδια άδεια.
διαθέσιμο με άδεια CC-BY-ND	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η δημιουργία παραγώγων του έργου.
διαθέσιμο με άδεια CC-BY-NC	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η εμπορική χρήση του έργου.
διαθέσιμο με άδεια CC-BY-NC-SA	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού και διάθεση του έργου ή του παράγωγου αυτού με την ίδια άδεια. Δεν επιτρέπεται η εμπορική χρήση του έργου.
διαθέσιμο με άδεια CC-BY-NC-ND	Επιτρέπεται η επαναχρησιμοποίηση του έργου με αναφορά του δημιουργού. Δεν επιτρέπεται η εμπορική χρήση του έργου και η δημιουργία παραγώγων του.
διαθέσιμο με άδεια CC0 Public Domain	Επιτρέπεται η επαναχρησιμοποίηση του έργου, η δημιουργία παραγώγων αυτού και η εμπορική του χρήση, χωρίς αναφορά του δημιουργού.
διαθέσιμο ως κοινό κτήμα	Επιτρέπεται η επαναχρησιμοποίηση του έργου, η δημιουργία παραγώγων αυτού και η εμπορική του χρήση, χωρίς αναφορά του δημιουργού.
χωρίς σήμανση	Συνήθως δεν επιτρέπεται η επαναχρησιμοποίηση του έργου.

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.