ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

Ενότητα 5: Τύποι Δεδομένων

Καθ. Γιάννης Γαροφαλάκης
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Γενικά (1)

Όλοι μας έχουμε αναπτύξει μια διαισθητική άποψη για το τι είναι ένας Τύπος Δεδομένων. Δηλαδή:

- Συλλογή τιμών από ένα πεδίο («**δηλωτική**» προσέγγιση - denotational)
- Εσωτερική δομή συνόλου δεδομένων, που περιγράφεται μέχρι το επίπεδο ενός μικρού συνόλου βασικών τύπων («**κατασκευαστική**» προσέγγιση - constructive)
- Συλλογή καλά ορισμένων λειτουργιών που μπορούν να εφαρμοστούν στα στοιχεία ενός Τύπου Δεδομένων («**αφαιρετική**» προσέγγιση – abstraction based)

Γενικά (2)

Τύπος Δεδομένων (ΤΔ):

Ένα σύνολο *στοιχείων* και ένα σύνολο *ενεργειών* πάνω στα στοιχεία αυτά, οι οποίες ενέργειες *δημιουργούν*, *υποστηρίζουν*, *καταστρέφουν*, *τροποποιούν* και *συλλέγουν* εμφανίσεις των στοιχείων.

- Κάθε ΓΠ παρέχει ένα καταρχήν σύνολο ΤΔ (**βασικοί ΤΔ** – primitive). Π.χ.
  - *Επιτακτικές Γλώσσες*: Integer, Real, Character, Boolean,…
- Ο προγραμματιστής συνήθως έχει τη δυνατότητα ορισμού νέων ΤΔ.

Γενικά (3)

Οι ΤΔ εξυπηρετούν 2 βασικούς **στόχους**:

1. Δημιουργούν ένα συγκεκριμένο *περιβάλλον* (context) στο οποίο εκτελούνται διάφορες λειτουργίες, απαλλάσσοντας τον προγραμματιστή από τη σχετική δουλειά. Π.χ.

   Η έκφραση A + B θα επιβάλλει ακέραιη πρόσθεση αν τα A, B είναι τύπου integer, ενώ αν τα A, B είναι τύπου float, θα επιβάλλει πρόσθεση αριθμητικής floating-point.

2. Αποτρέπουν τους προγραμματιστές από *λάθος λειτουργίες* που ίσως προσπαθήσουν να κάνουν. Π.χ. να προσθέσουν ένα χαρακτήρα με ένα record.

Γενικά (4)

- **Σύστημα Τύπων** (Type System):

  Η δυνατότητα ορισμού νέων ΤΔ και δήλωσης μεταβλητών, οι τιμές των οποίων περιορίζονται στα στοιχεία ενός ΤΔ, με πραγματοποίηση *ελέγχου τύπου.*
  Ένα Type System αποτελείται από:

  1. Ένα μηχανισμό **ορισμού** ΤΔ και συσχέτισής τους με συγκεκριμένες κατασκευές της γλώσσας, δηλαδή με αυτές που έχουν τιμή, όπως μεταβλητές, παράμετροι, εκφράσεις, …
  2. Ένα σύνολο **κανόνων** για *Ισοδυναμία ΤΔ* (type equivalence), *Συμβατότητα ΤΔ* (type compatibility) και *Εξαγωγή ΤΔ* (type inference).

Γενικά (5)

- **Έλεγχος Τύπου** (Type Checking):

  Διαδικασία επιβεβαίωσης ότι το πρόγραμμα υπακούει στους κανόνες Συμβατότητας ΤΔ της ΓΠ. Δύο έννοιες σχετίζονται με τον Έλεγχο Τύπου:

  – *Γλώσσες Ισχυρών Τύπων* (Strongly Typed)

    Απαγορεύουν την εφαρμογή ενεργειών σε στοιχεία που δεν πρέπει να υποστηρίζουν τις ενέργειες αυτές.

  – *Γλώσσες Στατικών Τύπων* (Statically Typed)

    Strongly Typed γλώσσες στις οποίες ο Έλεγχος Τύπου γίνεται κατά τη μετάφραση (συνήθως, μόνο το μεγαλύτερο μέρος του ελέγχου)

Γενικά (6)

Παραδείγματα:

- Η Common Lisp είναι strongly typed, αλλά όχι statically typed.
- Οι C, Ada είναι statically typed.
- Η Pascal είναι σχεδόν statically typed.
- Η Java είναι strongly typed, με περίεργο μίγμα πραγμάτων που άλλα ελέγχονται στατικά, και άλλα δυναμικά.

Γενικά (7)

- **Πολυμορφισμός** (polymorphism):

  Επιτρέπει σε ένα τμήμα κώδικα να δουλεύει με αντικείμενα πολλαπλών τύπων.

  – Αναγκαίος ο έλεγχος ΤΔ στο χρόνο εκτέλεσης (κόστος).

  – Lisp, Smalltalk

  – Σε αρκετές object-oriented ΓΠ (C++, Java, …) υπάρχει ο Πολυμορφισμός Υποτύπων (subtype polymorphism): Επιτρέπει σε μια μεταβλητή *X* τύπου *T* να συσχετίζεται με ένα αντικείμενο οποιουδήποτε υπο-τύπου του *T*.

  – Καθώς οι υπο-τύποι υποστηρίζουν όλες τις λειτουργίες του *T*, ο μεταφραστής είναι σίγουρος ότι κάθε πράξη αποδεκτή για αντικείμενο τύπου *T*, θα είναι αποδεκτή για κάθε αντικείμενο που συσχετίζεται με τη *X*.

  – Μπορεί να υλοποιηθεί στο χρόνο μετάφρασης.

Κατηγορίες ΤΔ

- **Βαθμωτοί (scalar) ή Απλοί ΤΔ**

  Το Πεδίο Τιμών (domain) τους αποτελείται από σταθερές τιμές που έχουν μόνο ένα χαρακτηριστικό

  - *Βασικοί* ΤΔ (primitive types): int, real/float, bool, char
  - Τύποι *Απαρίθμησης* (enumeration types)
  - Τύποι *Υποπεριοχής* (subrange types)

- **Διακριτοί ή Τακτικοί (ordinal) ΤΔ**

  Βαθμωτοί ΤΔ των οποίων τα μέλη του Πεδίου Τιμών τους αντιστοιχίζονται με ακέραιους (από τους παραπάνω βαθμωτούς, όχι οι real/float).

- **Σύνθετοι ή Δομημένοι (structured) ΤΔ**

  Το domain αποτελείται από μέλη που έχουν συντεθεί από ένα σύνολο άλλων ΤΔ (arrays, records, pointers, …)

Βασικοί ΤΔ (Primitive)

- Δεν ορίζονται με βάση άλλους ΤΔ
- Ορισμένοι βασικοί ΤΔ απλώς αντανακλούν το H/W (π.χ. integer)
- Άλλοι βασικοί ΤΔ θέλουν μόνο λίγο επιπλέον S/W υποστήριξη
- C, C++ : int, float, double, char
- C++, Java : Οι παραπάνω, και επιπλέον bool (-ean)

Βασικοί ΤΔ (2)

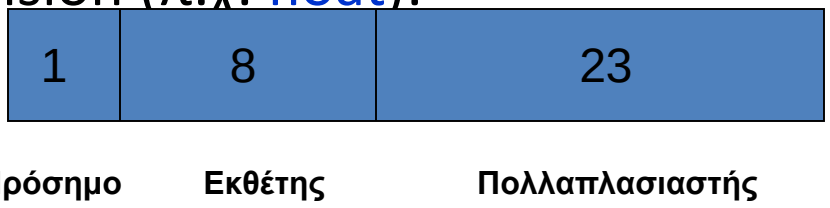
1. Ακέραιοι Αριθμοί (Integer)

- Το υποσύνολο των ακέραιων από MININT έως MAXINT
- Πολλοί Η/Υ και Γλώσσες υποστηρίζουν διάφορα μεγέθη. Π.χ.
  - int (4 bytes)
  - short int (2 bytes)
  - long int (4 bytes)
  - unsigned int (2 bytes)

Βασικοί ΤΔ (3)

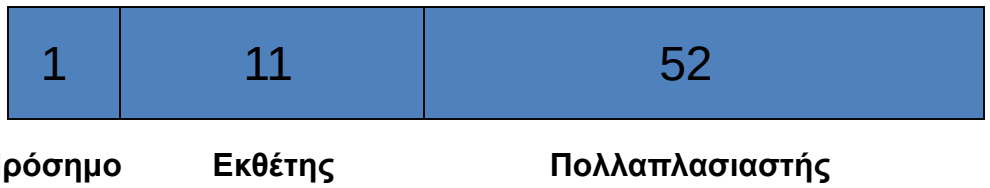
2. Αριθμοί Κινητής Υποδιαστολής (Floating – Point)

- Είναι προσεγγιστικές αναπαραστάσεις πραγματικών αριθμών.
- IEEE floating – point standard 754:
  - Single precision (π.χ. float):



4 bytes (32 bits)

  - Double



8 bytes (64 bits)

Βασικοί ΤΔ (4)

3. Boolean

- Δύο Τιμές: FALSE, TRUE (1 byte)
- Όλες οι γλώσσες έχουν, εκτός της C.

4. Χαρακτήρες

- char στη C.
- Κωδικοποίηση ASCII: 1 byte
- Κωδικοποίηση UNICODE: τουλάχιστον 2 bytes

Ακολουθίες Χαρακτήρων (1) (Character Strings)

Σχεδιαστικά ερωτήματα:

- Βασικός ΤΔ ή ειδικός τύπος char array;
- Στατικό ή Δυναμικό μήκος;

- Pascal, Ada, C, C++ : Όχι βασικός ΤΔ. Char Array. Π.χ. char line[100]
- C : String operations από Standard Library string.h :
  - strcpy (μετακίνση string)
  - strcat (πρόσθεση strings)
  - strcmp (σύγκριση strings)
  - strlen (αριθμός χαρακτήρων)

Ακολουθίες Χαρακτήρων (2)

- C : Char pointers (δείχνουν σε char array):

  char *str = “mary”;

  Τα strings τελειώνουν με null, οπότε δεν χρειάζεται η γνώση του τρέχοντος μήκους του string.

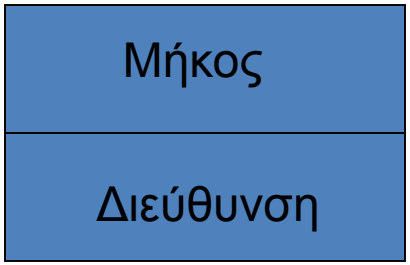
- Java, C++ :

  String class για string objects

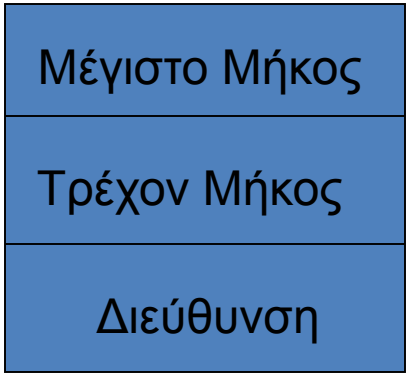
- SNOBOL, Perl : Πλήρως δυναμικά strings

Ακολουθίες Χαρακτήρων (3)

- FORTRAN, COBOL, Pascal : Static Strings



- C, C++ : Limited Dynamic Strings



Τύποι Απαρίθμησης (enumeration types)

- Στις περισσότερες (και παλιότερες) γλώσσες, το domain των τύπων απαρίθμησης περιγράφεται από μια *διατεταγμένη* λίστα τιμών (σταθερών).
- Στις γλώσσες αυτές, οι μόνες πράξεις που επιτρέπονται, είναι ο *έλεγχος ισότητας* και *διάταξης,* και η *ανάθεση*.
- Π.χ. στην Pascal:

```
type days = (mon, tue, wed, thu, fri, sat, sun);

var x: days;
```

  - Υπάρχουν οι predefined functions: ORD, PRED, SUCC
  - Απαγορεύεται η χρήση ίδιας σταθεράς σε άλλο ορισμό

Τύποι Απαρίθμησης (2)

- Στις C, C++, οι τιμές είναι ουσιαστικά ακέραιες σταθερές, οπότε μπορούν να χρησιμοποιηθούν όπως αυτές (ορθογωνιότητα).
- Παράδειγμα:

```
enum boolean {NO, YES, FALSE = 0, TRUE}
boolean A;
A = YES - 1;
```

- Επίσης:

```
enum days {mon=1, tue=2, wed=3, …}
```

ή

```
enum days {mon=1, tue, wed, …}
```

ή

```
enum days {mon=1, tue, wed, … , MON=1, TUE, WED, …}
days i = MON;
```

Αν δεν αντιστοιχίσουμε την 1η τιμή, το default είναι 0.

Τύποι Υποπεριοχής (subrange types)

- Είναι ΤΔ οι τιμές του οποίου είναι ένα *συνεχές υποσύνολο* των τιμών κάποιου διακριτού (ordinal) ΤΔ (ακέραιοι, χαρακτήρες, απαριθμήσεις, άλλες υποπεριοχές).
- Εμφανίστηκαν για πρώτη φορά στην Pascal, και στη συνέχεια σε μεταγενέστερες ΓΠ της οικογένειας Algol.
- Pascal:

```
type test_score = 0 . . 100;

     workday = mon . . fri;
```

ΤΔ Array (1)

- Ομάδα από ομογενή στοιχεία δεδομένων που προσδιορίζονται από τη θέση τους στην ομάδα, σε σχέση με το πρώτο. Ορισμός:

  C: int A[20] Pascal: A: ARRAY [0..19] of INTEGER

- **Σχεδιαστικά θέματα**

1. Default κατώτερη τιμή του δείκτη:
   - C, C++ : 0
   - FORTRAN : 1
   - Pascal: Ορίζεται από το χρήστη

ΤΔ Array (2)

2. Διαστάσεις δεικτών:
   - C, C++ : 1 διάσταση, αλλά κάθε element του array μπορεί να είναι array (multidimensional): int B[5][4]
   - FORTRAN : 3 διαστάσεις στην I, 7 στη FORTRAN IV
3. Αρχικοποίηση τιμών array κατά τη δήλωση:
   - FORTRAN 77 : INTEGER L(3)

     DATA L /0, 5, 9/
   - C, C++ : int L [ ] = {4, 7, 8, 53} Ορίζεται και το μήκος array (4)

     char name [ ] = “freddie” Array μήκους 8 (null στο τέλος)

     char *names [ ] = {“Bob”, “Jake”, “Mary”}

     Array of pointers σε χαρακτήρες: Το names[0] είναι pointer στο γράμμα ‘B’ στο char array “Bob/null”
   - Pascal : Όχι

ΤΔ Record (1)

- Ομάδες στοιχείων που αποτελούν συνθέσεις από συγκεκριμένο αριθμό (πιθανόν) ανομοιογενών στοιχείων δεδομένων, που αναγνωρίζονται από το όνομά τους.

- Διαφορές από arrays:
  - Τα συστατικά των records μπορεί να είναι ετερογενή.
  - Τα στοιχεία των records έχουν συμβολικά ονόματα (id), ενώ των arrays καθορίζονται από το δείκτη.

ΤΔ Record (2)

- C, C++ :

```
struct EmployeeType
        {  int ID;
           int Age;
           float Salary;
           char Dept;
        }   Employee[500] ;
```

Ορίζεται record ΤΔ με όνομα EmployeeType, και παράλληλα δηλώνεται ένα array 500 θέσεων με όνομα Employee, με στοιχεία τύπου EmployeeType.

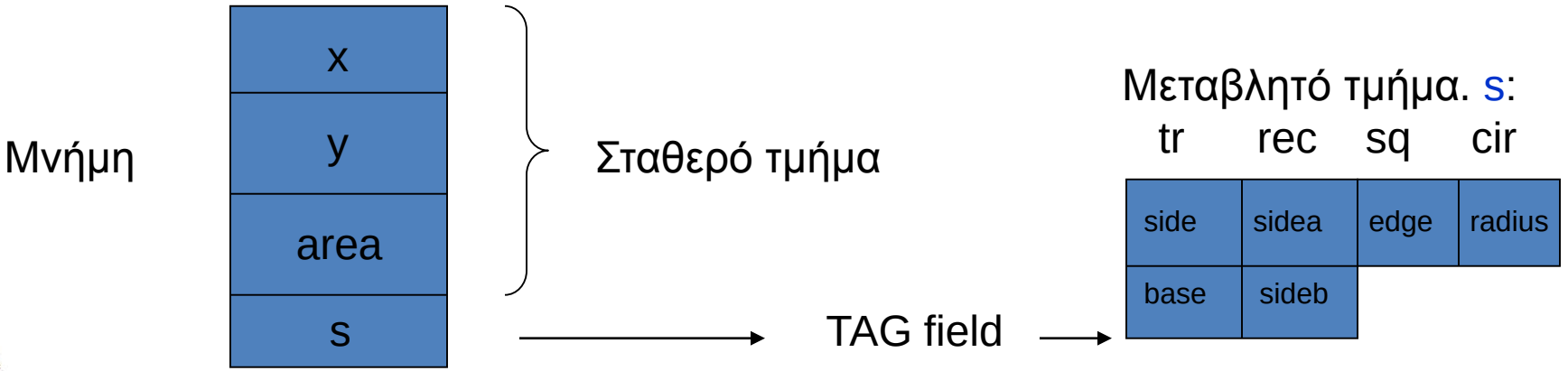
Πρόσβαση στα στοιχεία του: Employee[3].Salary

Άλλος ορισμός (απλής μεταβλητής): struct EmployeeType A;

ΤΔ Record (3)

- Pascal :

```
type shape = (triangle, rectangle, square, circle);
        coordinates = RECORD
          x, y: real;
          area: real;
          case s: shape of
                triangle: (side: real; base: real);
                rectangle: (sidea, sideb: real);
                square: (edge: real);
                circle: (radius: real)
end;
```



ΤΔ δείκτη (pointer)

- **Δείκτης**: Αναφορά σε θέση μνήμης
- **Μεταβλητή ΤΔ δείκτη**: Όνομα (id) με τιμή που είναι αναφορά σε διεύθυνση μνήμης. Δηλαδή, οι τιμές της περιέχονται στο σύνολο [διευθύνσεις μνήμης, nil (τίποτα)].
- **Λόγοι ύπαρξης** ΤΔ δείκτη:
  - Πολλά στοιχεία μπορούν να συνδέονται μεταξύ τους χωρίς να παρέχονται συγκεκριμένα ονόματα για όλα.
  - Επιτρέπουν την ταυτόχρονη τοποθέτηση των στοιχείων σε πολλές δομές (π.χ. λίστες, arrays).
  - Με λίγες δηλώσεις, μπορούμε να έχουμε μεγάλη ποικιλία στοιχείων που συνδέονται με πολλούς τρόπους και προσπελαύνονται με ομοιόμορφο τρόπο.

ΤΔ δείκτη (2)

- Συνέπειες (και πιθανά προβλήματα) ύπαρξης δεικτών:
  - Μπορεί αρκετές Μεταβλητές Δείκτη (ΜΔ) να αναφέρονται στο ίδιο αντικείμενο, στο ίδιο σημείο, ή σε διαφορετικά σημεία του ίδιου αντικειμένου.
  - Κάποια αντικείμενα μπορεί να μη δεικτοδοτούνται πλέον από καμία ΜΔ.
  - Πρέπει η ΓΠ να παρέχει σημειογραφία για διάκριση *θέσης*, *τιμής-r* και *τιμής-l*.

ΤΔ δείκτη (3)

- C, C++ : Οι δείκτες και τα arrays είναι στενά συνδεδεμένες έννοιες:

```
int n;
int *a;        (δείκτης σε integer)
int b[10];     (array 10 integers)
```

Τα παρακάτω είναι νόμιμα:

```
a = b;
```

(το a θα δείχνει στο b[0])

```
n = a[3];
```

(το n παίρνει την τιμή του b[3])

```
n = *(a+3)
```

(ίδιο με το προηγούμενο)

ΤΔ δείκτη (4)

***** για αποαναφοροποίηση και ορισμό μεταβλητών δείκτη

& διεύθυνση μνήμης

```
int *ptr;
int count, init;
...
ptr = &init;
count = *ptr;
```

ptr = 0 (δηλαδή ptr ίσο με nil)

Γενικά οι δείκτες δείχνουν σε αντικείμενα της heap memory. Στη C μπορεί να δείχνει ο δείκτης και σε αντικείμενα της stack μνήμης. Στην Pascal όχι.

ΤΔ δείκτη (5)

- Δημιουργία αντικειμένου από τη heap memory:
  - C : `int *x;`
    `x = malloc(sizeof(int))`
  - C++ : `int *x;`
    `x = new int;`
  - Pascal : `var ^x: integer;`
    `new (x)`
- Καταστροφή αντικειμένου από τη heap memory:
  - C : `free(x)`
  - C++ : `delete x`
  - Pascal : `dispose(x)`

Ισοδυναμία ΤΔ (1)

- Ο Έλεγχος Τύπου (στατικός ή δυναμικός) εμπεριέχει σύγκριση μεταξύ του ΤΔ του πραγματικού ορίσματος που δίνεται σε μια operation, με τον ΤΔ που αναμένεται στην operation.
- Αν είναι ίδιοι, γίνεται αποδεκτό το όρισμα και προχωράει η operation.
- Αν δεν είναι ίδιοι, τότε είτε error, ή γίνεται μετατροπή του ΤΔ του πραγματικού ορίσματος, ώστε να είναι συμβατό με αυτό που αναμένεται.

Ισοδυναμία ΤΔ (2)

- Το πρόβλημα της αναγνώρισης της ισοδυναμίας ΤΔ, οφείλεται στη δυνατότητα να ορίζει νέους ΤΔ ο χρήστης. Παράδειγμα (γλώσσα τύπου Pascal):

```
type V1: array[1..10] of real;
       V2: array{1..10] of real;
var X, Z: V1;
         Y: V2;
procedure Sub(A: V1);
end;
BEGIN
          X:= Y;
          Sub(Y);
END.
```

Ερωτήματα:

- X, Y, Z, A έχουν ίδιο ΤΔ;
- Επιτρέπεται το X:= Y;
- Επιτρέπεται το Sub(Y);

Ισοδυναμία ΤΔ (3)

Δύο λύσεις στο πρόβλημα:

- **Α. Ισοδυναμία Ονομάτων** (name equivalence)

  Δύο ΤΔ είναι ισοδύναμοι μόνο αν έχουν το ίδιο όνομα. Δηλαδή στο παράδειγμα:

  V1 ≠ V2 και X:= Z σωστό, X:= Y, Sub(Y) λάθος

  Πιο δημοφιλής λύση: Java, C++, Ada

- **Β. Ισοδυναμία δομών** (structural equivalence)

  Δύο ΤΔ είναι ισοδύναμοι αν ορίζουν μεταβλητές με ίδια συστατικά και δομή. Δηλαδή στο παράδειγμα:

  V1 = V2 και X:= Z , X:= Y, Sub(Y) σωστά

  C, Algol-68, FORTRAN, COBOL

Pascal : Συνδυασμός των 2 (Ισοδυναμία Δήλωσης)

Συμβατότητα ΤΔ

- **Μετατροπή ΤΔ**

  Διαδικασία στην οποία μια τιμή ενός ΤΔ, μετατρέπεται σε τιμή άλλου ΤΔ.

  Κατηγορίες Μετατροπών:

  – *Διεύρυνση* (widening). Π.χ. από int σε float

  – *Περιορισμός* (narrowing). Π.χ. από float σε int (truncation)

  Δύο Τρόποι:

  – *Implicit* (στη C όλα επιτρεπτά, στην Pascal μόνο διεύρυνση INT σε REAL)

  – *Explicit* (στην Pascal : i:= trunc(r), i:= round(r))

Εξαγωγή ΤΔ (type inference)

```
type Atype = 0..20;
     Btype = 10..20;
var  a : Atype;
     b : Btype;
```

Ποιος είναι ο ΤΔ του a + b ;

Συνήθης απάντηση: Ο αρχικός βασικός ΤΔ του subrange ΤΔ, δηλαδή integer

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών**» 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

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης