# Διαδικαστικός Προγραμματισμός

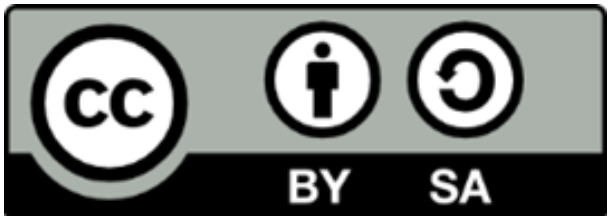
## **Ενότητα 10:** Εγγραφές/Δομές

**Καθηγήτρια Μαρία Σατρατζέμη**

**Τμήμα Εφαρμοσμένης Πληροφορικής**

# Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη 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.

# Σκοποί ενότητας

- Να κατανοήσετε την έννοια της εγγραφής/δομής και τη σημασία της για τον προγραμματισμό.
- Να αντιληφθείτε τη διαφορά μεταξύ του ορισμού ενός τύπου εγγραφής και της δήλωσης μιας μεταβλητής εγγραφής.
- Να μπορείτε να χειρίζεστε εγγραφές χρησιμοποιώντας τον τελεστή τελεία (dot).
- Να μάθετε πώς να δηλώνετε και να χρησιμοποιείτε πίνακες εγγραφών και εγγραφές με πίνακες ως πεδία.

# Η έννοια της εγγραφής δεδομένων (1)

- Οι πίνακες, όπως έχουμε ήδη αναφέρει επιτρέπουν την αποθήκευση διατεταγμένων συλλογών ομοιογενών τιμών.
- Στο μάθημα αυτό θα παρουσιαστεί η έννοια της δομής (structure) ή εγγραφής (record), η οποία επιτρέπει την αποθήκευση μη διατεταγμένων και ανομοιογενών στοιχείων ως μιας συλλογής.
- Δομή ή εγγραφή, λοιπόν, ονομάζεται μια συλλογή μη διατεταγμένων και ετερογενών τιμών που θεωρούνται μια οντότητα.

# Η έννοια της εγγραφής δεδομένων (2)

- **Παράδειγμα**: ας υποθέσουμε ότι σε μια εταιρεία πρέπει να γνωρίζουμε το όνομα κάθε υπαλλήλου, τη θέση εργασίας του, τον αριθμό κοινωνικής ασφάλισης, τον ακαθάριστο μισθό του και το ποσοστό των κρατήσεων, για να μπορούμε -για παράδειγμα- να εκτυπώνουμε τις επιταγές μισθοδοσίας.
- Όλες οι παραπάνω πληροφορίες απαρτίζουν την εγγραφή δεδομένων κάθε υπαλλήλου.
- Με τι μοιάζουν όμως οι εγγραφές των υπαλλήλων;

| Όνομα | Θέση | Αριθμ. κοιν. Ασφ. | Ακαθ. μισθός | Ποσοστό κρατήσεων |
|---|---|---|---|---|
| **John Smith** | **Accountant** | **271-82-8183** | **1300** | **0.20** |
| **Pappa Maria** | **Manager** | **314-15-9265** | **2500** | **0.25** |

# Η έννοια της εγγραφής δεδομένων (3)

| Όνομα | Θέση | Αριθμ. κοιν. Ασφ. | Ακαθ. μισθός | Ποσοστό κρατήσεων |
|---|---|---|---|---|
| John Smith | Accountant | 271-82-8183 | 1300 | 0.20 |
| Pappa Maria | Manager | 314-15-9265 | 2500 | 0.25 |

- Κάθε εγγραφή διαιρείται σε μεμονωμένα συστατικά στοιχεία, καθένα από τα οποία παρέχει μια συγκεκριμένη πληροφορία για ένα υπάλληλο.
- Τα μεμονωμένα συστατικά στοιχεία μιας εγγραφής ονομάζονται πεδία (fields) ή, όπως συνηθίζεται στη C, μέλη (members).
- Κάθε πεδίο σχετίζεται με έναν τύπο δεδομένων, ο οποίος μπορεί να είναι διαφορετικός για τα διάφορα πεδία της εγγραφής.
- Βέβαια, παρά το γεγονός ότι μια εγγραφή απαρτίζεται από μεμονωμένα πεδία, θα πρέπει να έχει νόημα και ως συνεκτικό σύνολο, όπως στο παράδειγμα όπου μια εγγραφή αναπαριστά έναν υπάλληλο.

# Δημιουργία δεδομένων εγγραφών (1)

Για να δημιουργήσετε δεδομένα εγγραφών στη C, πρέπει να ακολουθήσετε μια διαδικασία δύο βημάτων:

1. **Να ορίσετε ένα νέο τύπο δομής**:
    - Πριν δηλώσετε οποιεσδήποτε εγγραφές, θα πρέπει να έχετε δηλώσει ένα νέο τύπο δομής.
    - Στον ορισμό του τύπου καθορίζονται τα πεδία από τα οποία θα αποτελείται η δομή, ποια θα είναι τα ονόματά τους, και ποιος είναι ο τύπος πληροφοριών που θα περιέχει κάθε πεδίο.
    - Αυτός ο τύπος δομής θα αποτελεί το μοντέλο για όλες τις μεταβλητές αυτού του νέου τύπου, αλλά από μόνος του δεν δεσμεύει χώρο στη μνήμη.
2. **Να δηλώσετε μεταβλητές του νέου τύπου.**
    - Αφού ορίσετε το νέο τύπο, το επόμενο βήμα σας είναι η δήλωση μεταβλητών αυτού του τύπου ώστε να μπορείτε να αποθηκεύσετε σε αυτές πραγματικές τιμές δεδομένων.

# Δημιουργία δεδομένων εγγραφών (2)

**Ορισμός νέου τύπου εγγραφής/δομής**

```
typedef struct {
        δηλώσεις_πεδίων
}όνομα_νέου_τύπου;
```

όπου:

- δηλώσεις_πεδίων — είναι καθιερωμένες δηλώσεις μεταβλητών οι οποίες χρησιμοποιούνται εδώ για τον ορισμό των πεδίων της δομής
- όνομα_νέου_τύπου — είναι το όνομα του τύπου που ορίζετε

```
typedef struct{
   char name[30];
   char jobTitle[20];
   char secNumber[10];
   double salary;
   double percentage;
} employeeT;
```

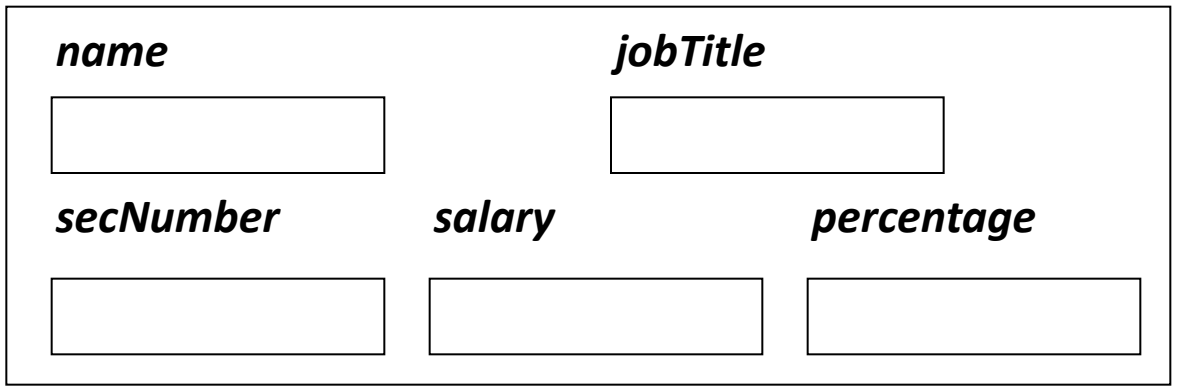
# Δημιουργία δεδομένων εγγραφών (3)

**Δήλωση μεταβλητών εγγραφών/δομών**

- Με δεδομένο τον τύπο **employeeT** μπορούμε να δηλώσουμε μια μεταβλητή, έστω **emp**, αυτού του τύπου ως εξής:

**employeeT emp;**

```
typedef struct{
   char name[30];
   char jobTitle[20];
   char secNumber[10];
   double salary;
   double percentage;
}employeeT;
```



# Δημιουργία δεδομένων εγγραφών (4)

**Δήλωση μεταβλητών εγγραφών/δομών(άλλοι τρόποι)**

- Μπορούμε να δηλώσουν μια δομή με 3 τρόπους

a)Δήλωση τύπου και στη συνέχεια δήλωση της μεταβλητής

```
struct employeeT{
   char name[30];
   char jobTitle[20];
   char secNumber[10];
   double salary;
   double percentage;
};

employeeT  emp;
```

***Προτείνεται η χρήση του α) τρόπου***

b) Δήλωση μεταβλητής μαζί με τη δήλωση του τύπου

```
struct employeeT{
   char name[30];
   char jobTitle[20];
   char secNumber[10];
   double salary;
   double percentage;
} emp;
```

c) Δήλωση μόνο μεταβλητής όχι τύπο

```
struct {
      char name[30];
   char jobTitle[20];
   char secNumber[10];
   double salary;
   double percentage;
} emp;
```

# Επιλογή εγγραφών

- Μετά τη δήλωση της μεταβλητής **emp** :

  **employeeT emp;**

θα μπορείτε να αναφέρεστε στη μεταβλητή συνολικά χρησιμοποιώντας απλά το όνομά της.

- Κάποια στιγμή, ωστόσο, θα χρειαστεί να αναφερθείτε σε συγκεκριμένα πεδία της εγγραφής.
- Η αναφορά σε συγκεκριμένα πεδία μιας εγγραφής γίνεται γράφοντας το όνομα της εγγραφής, μια τελεία και το όνομα του πεδίου:

  **emp.title;**

- Όταν χρησιμοποιείται σε αυτά τα πλαίσια, η τελεία ονομάζεται **τελεστής τελεία** (dot).
- Η επιλογή ενός πεδίου με τη χρήση του τελεστή τελεία ονομάζεται **επιλογή εγγραφής** (record selection).

# Ανάθεση αρχικών τιμών σε εγγραφές (1)

- Όπως και με οποιονδήποτε άλλο τύπο μεταβλητής, μπορείτε να αναθέσετε αρχικές τιμές στα περιεχόμενα μιας μεταβλητής εγγραφής αναθέτοντας αρχικές τιμές στα συστατικά της στοιχεία.
- Για το παράδειγμα μας, θα έχουμε:

```
strcpy(emp.name, “John Smith”);
strcpy(emp.jobTitle ,“Accountant”);
strcpy(emp.secNumber, “271-82-8183”);
emp.salary = 1300;
emp.percentage = 0.25;
```

- Φυσικά, τα πεδία μιας εγγραφής μπορούν να πάρουν τιμές από το πληκτρολόγιο, χρησιμοποιώντας τις γνωστές συναρτήσεις GetInteger(), GetReal() κτλ.
- Στην περίπτωση δήλωσης ενός πεδίου ως πίνακα χαρακτήρων, μπορούμε να διαβάσουμε σε αυτό τιμή από το πληκτρολόγιο με τη διαδικασία gets().

# Προσοχή!

```
typedef struct{
   char name[30];
   char jobTitle[20];
   char secNumber[10];
   double salary;
   double percentage;
} employeeT;
```

Στη σελίδα 647 του βιβλίου του *Roberts «Η Τέχνη και Επιστήμη της C»*, η εγγραφή έχει δηλωθεί με το 2ο τρόπο όσον αφορά τα αλφαριθμητικά πεδία.

```
typedef struct{
   string name;
   string jobTitle;
   string secNumber;
   double salary;
   double percentage;
} employeeT;
```

*Τη δήλωση αλφαριθμητικού με τον **τύπο string εμείς δε τη χρησιμοποιούμε στα μαθήματα μας**. Ο τύπος string ορίζεται από τον Roberts στη βιβλιοθήκη "genlib.h".*

*Η δήλωση αλφαριθμητικού γίνεται είτε με ένα μονοδιάστατο πίνακα χαρακτήρων ή με ένα δείκτη προς τη διεύθυνση του 1ου χαρακτήρα.*

# Ανάθεση αρχικών τιμών σε εγγραφές (2)

- Αν μια μεταβλητή εγγραφής έχει δηλωθεί ως στατική καθολική μεταβλητή, μπορείτε να αναθέσετε αρχικές τιμές με την ίδια σύνταξη που χρησιμοποιήσατε για τους πίνακες.
- Για το παράδειγμα μας, θα έχουμε:

```
static employeeT manager ={
"John Smith","Accountant","271-82-8183", 1300,0.25
};
```

- Επίσης μπορείτε να αναθέσετε το περιεχόμενο μιας εγγραφής σε μια άλλη εγγραφή αντί να αναθέτετε το περιεχόμενο κάθε πεδίου.

```
    employeeT Employee1, Employee2;
  Employee1 = Employee2; //είναι ισοδύναμο με
/*  strcpy(Employee1.name, Employee2.name);
  strcpy(Employee1.jobTitle, Employee2.jobTitle);
   strcpy(Employee1.secNumber, Employee2.secNumber);
   Employee1.salary = Employee2.salary;
   Employee1.percentage = Employee2.percentage;
*/
```

# Απλές εγγραφές (1)

- Τύπος εγγραφής για την αναπαράσταση ενός σημείου με συντεταγμένες ***x*** και ***y***:

```
typedef struct{
        double x, y;
} pointT;
```

- Στη συνέχεια, μπορείτε να γράψετε συναρτήσεις και διαδικασίες, οι οποίες χειρίζονται τιμές τύπου ***pointT*** :

```
pointT CreatePoint( double x, double y)
{
        pointT p;

        p.x = x;
        p.y = y;

        return (p);
}
```

# Απλές εγγραφές (2)

- Στη συνέχεια, μπορείτε να χρησιμοποιήσετε αυτή τη συνάρτηση για να αναθέσετε αρχική τιμή σε μια μεταβλητή τύπου ***pointT*** :

```
pointT origin;
origin = CreatePoint(0, 0);
```

- Οι εγγραφές είναι δυνατό επίσης να μεταβιβάζονται ως ορίσματα σε συναρτήσεις:

```
pointT AddPoint( pointT p1, pointT p2)
{
        pointT p;

        p.x = p1.x + p2.x;
        p.y = p1.y + p2.y;

        return (p);
}
```

# Απλές εγγραφές (3)

- Αν εκτελέσετε την κλήση ***AddPoint(p1, p2)***, όπου ***p1*** και ***p2*** είναι τιμές τύπου ***pointT***, η συνάρτηση θα επιστρέψει την τιμή τύπου ***pointT*** της οποίας οι συντεταγμένες αποτελούν το άθροισμα των αντίστοιχων συντεταγμένων των τιμών ***p1*** και ***p2***.
- Όταν καλείτε μια συνάρτηση η οποία δέχεται ως όρισμα κάποια εγγραφή, η τιμή της εγγραφής ***αντιγράφεται*** στην αντίστοιχη τυπική παράμετρο της συνάρτησης.
- Αν στο εσωτερικό μιας συνάρτησης τροποποιηθεί η τιμή της τυπικής παραμέτρου ή οι τιμές οποιονδήποτε από τα εσωτερικά της πεδία, το όρισμα με το οποίο κλήθηκε η συνάρτηση θα διατηρήσει την αρχική του τιμή.
- Αυτή η συμπεριφορά είναι συνεπής με τον τρόπο με τον οποίο χειρίζεται η C όλες τις παραμέτρους εκτός από τους πίνακες.

# Συνδυασμός εγγραφών και πινάκων (1)

- Συχνά, είναι χρήσιμος ο ορισμός πινάκων από εγγραφές ή ακόμα και εγγραφών που περιέχουν πίνακες.
- Μπορούμε για παράδειγμα να δηλώσουμε –με το γνωστό τρόπο- ένα πίνακα όπου κάθε στοιχείο του θα είναι τύπου **employeeT**:

```
#define MaxEmployees 100   /*σταθερά που δηλώνει το μέγιστο
                                πλήθος στοιχείων του πίνακα*/

employeeT staff[MaxEmployees];      /*πίνακας 100 εγγραφών τύπου
                                        employeeT */

int nEmployees;                 /*μεταβλητή που αποθηκεύει το τρέχον
                                πλήθος εγγραφών του πίνακα*/
```

# Συνδυασμός εγγραφών και πινάκων (2)

Συνάρτηση για την εμφάνιση στην οθόνη των ονομάτων όλων των υπαλλήλων, μαζί με τους τίτλους εργασίας τους:

```
void ListEmployees( employeeT staff[], int nEmployees)
{
   int i;

   for(i = 0; i < nEmployees; i++){
            printf("%s (%s)\n", staff[i].name, staff[i].jobTitle);
   }
}
```

# Συνδυασμός εγγραφών και πινάκων (3)

Συνάρτηση για τον υπολογισμό και την επιστροφή του μέσου μισθού όλων των υπαλλήλων:

```
double AverageSalary( employeeT staff[], int nEmployees)
{
   double total, average;
   int i;

   total = 0;
   for(i = 0; i < nEmployees; i++){
            total = total + staff[i].salary;
   }
   average = total/nEmployees;

   return (average);

}
```

# Συνδυασμός εγγραφών και πινάκων (4)

Συνάρτηση που επιστρέφει εγγραφή/δομή :

- *Βρίσκει τον υπάλληλο με το μικρότερο μισθό και επιστρέφει την εγγραφή του*

```
employeeT empMinSalary(employeeT staff[MaxEmployees],
                                                    int nEmployees)
{
   double minSalary;
   int i;
   employeeT emp;

   minSalary=staff[0].salary;
   emp=staff[0];

   for(i = 1; i < nEmployees; i++)
                if (minSalary>staff[i].salary) {
        minSalary=staff[i].salary;
        emp=staff[i];
   }

  return emp;
}
```

# Συνδυασμός εγγραφών και πινάκων (5)

Συνάρτηση που μεταβιβάζει εγγραφή/δομή με αναφορά:

- *Βρίσκει τον υπάλληλο με το μικρότερο μισθό και επιστρέφει την εγγραφή του με αναφορά.*

***1ος τρόπος***

```
void empMinPerce(employeeT
                 staff[MaxEmployees],
                 int nEmployees,
                 employeeT *AnEmployee)
{
   int i;
   double minPer; int index;

   minPer = staff[0].percentage;
   index = 0;
   for(i = 1; i < nEmployees; i++)
     if                                    (minPer >
staff[i].percentage)
                 {
        minPer = staff[i].percentage;
        index = i;
                 }
  *AnEmployee = staff[index];
}
```

***2ος τρόπος***

```
void empMinPerce(employeeT
                 staff[MaxEmployees],
                 int nEmployees,
                 employeeT *AnEmployee)
{
   int i;

   *AnEmployee = staff[0];
   for(i = 1; i < nEmployees; i++)
                 if ((*AnEmployee).percentage >
                 staff[i].percentage)
                 {
        (*AnEmployee).percentage =
staff[i].percentage;
        *AnEmployee = staff[i];
                 }
}
```

***1ος τρόπος:*** *Χρήση 2 βοηθητικών μεταβλητών για να αποθηκεύσω το μικρότερο ποσοστό (**minPer** ) και σε ποιον υπάλληλο αντιστοιχεί (**index**) και δε χρησιμοποιώ την αναφερόμενη από το δείκτη μεταβλητή τύπου δομής ***AnEmployee** , όπως γίνεται στο 2ο τρόπο.*

# Συνδυασμός εγγραφών και πινάκων (6)

```
#define Elements 56                    /*σταθερά που δηλώνει το μέγιστο πλήθος στοιχείων του πίνακα*/
typedef struct{
      double timi;
      char xroma[20];
      char iliko[20];
      char etairia[30];
   } chairT;

void ReadAchair(chairT *k);                              //μεταβιβάζω με αναφορά μια μεταβλητή τύπου δομής chairT
void ReadChairs(chairT k[],int size);     //μεταβιβάζω με αναφορά ένα στοιχείο του πίνακα που είναι μια μεταβλητή
                                          τύπου δομής chairT
void PrintChairs(chairT k[],int size);    //μεταβιβάζω ολόκληρο τον πίνακα που ο τύπος κάθε στοιχείου του πίνακα είναι
                                          τύπου δομής chairT
main()
{
   chairT kareklitsa;
   chairT karekles[Elements];

   printf("Stoixeia karekla 1\n");
   //μεταβιβάζω με αναφορά μια μεταβλητή τύπου δομής chairT
   ReadAchair(&kareklitsa);
   printf("%.2f, %s, %s, %s\n", kareklitsa.timi, kareklitsa.xroma, kareklitsa.iliko, kareklitsa.etairia);

   printf("Stoixeia karekla 10\n");
   //μεταβιβάζω με αναφορά ένα στοιχείο του πίνακα που είναι μια μεταβλητή τύπου δομής chairT
   ReadAchair(&karekles[10]);
   printf("%.2f, %s, %s, %s\n", karekles[10].timi, karekles[10].xroma, karekles[10].iliko, karekles[10].etairia);

   printf("Stoixeia karekles treis\n");
   //μεταβιβάζω ολόκληρο τον πίνακα που ο τύπος κάθε στοιχείου του πίνακα είναι τύπου δομής chairT
   ReadChairs(karekles, 3);
   PrintChairs(karekles, 3);
   system("PAUSE");

}
```

# Τέλος Ενότητας



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης