Διάλεξη 18η: Δομημένοι Τύποι

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Δομημένοι τύποι (Structure Types)

- Καινούργιοι τύποι δεδομένων που αποτελούνται από την ομαδοποίηση υπαρχόντων τύπων δεδομένων
- Ομαδοποίηση πληροφορίας που δεν μπορεί να αποθηκευτεί σε μία μόνο μεταβλητή από τους δοσμένους τύπους της C
- Συλλογή μη διατεταγμένων και ομοιογενών τιμών
- Πλεονέκτημα: Καλύτερη οργάνωση και διαχείριση πληροφορίας, πιο ευέλικτος σχεδιασμός

Παράδειγμα

Όνομα	Τίτλος	Α.Τ.	Μισθός	Χρόνια
Γιώργος Α.	Διευθυντής	Α123123	1500	4
Χάρης Λ.	Υπάλληλος	Δ424344	1000	3
Μαρία Σ.	Υπάλληλος	Ζ824344	1000	2
Νίκος Κ.	Υπάλληλος	Α848239	750	1

Ορισμός structs

- Ορισμός νέου τύπου struct

```
struct employee {
  char *name;
  char title[20];
  char id[8];
  double salary;
  int years;
};
```

- Τα name, title, id, salary, years ονομάζονται τα μέλη ή πεδία της δομής
- Το employee ονομάζεται η ετικέτα της δομής

Δήλωση μεταβλητών

- Δηλώνονται όπως οι άλλες μεταβλητές
- Ο τύπος της μεταβλητής είναι π.χ., `struct employee`

Παράδειγμα

```
struct employee e1, e2;
struct employee *pe;
struct employee manager = {"AB", "manager", "A11111", 2000, 2};
struct employee e3 = {0};
```

Πρόσβαση στα Περιεχόμενα της Δομής

- Με τον τελεστή '.' (τελεία) αποκτούμε πρόσβαση στα μέλη της δομής

```
struct point {
  int x;
  int y;
} p;

p.x = 10;
p.y = 20;
```

Φωλιασμένες δομές

- Δομές ως μέλη άλλων δομών επιτρέπονται

Παράδειγμα

```
struct point {
  int x;
  int y;
};

struct rectangle {
  struct point topleftcorner;
  struct point bottomrightcorner;
} rect1;
```

- Πρόσβαση με διαδοχικές τελείες (.)
  `rect1.topleftcorner.x = 10;`
- Παίρνουμε το πεδίο `x` του πεδίου `topleftcorner` της μεταβλητής `rect1`

Πίνακες από Δομές

- Όπως και για κάθε άλλο τύπο

Παράδειγμα

```
struct point {
  int x;
  int y;
};
struct point points[100];
points[2].x = 10;
```

Χρήση Δομών

- Σαν κανονικές μεταβλητές για την αποθήκευση πληροφορίας
- Σαν ορίσματα σε Συναρτήσεις:

```
void printEmployee(struct employee e);
```

- Σαν επιστρεφόμενες τιμές από συναρτήσεις:

```
struct employee newEmployee(char *name, char *pos,
                            char *id, double sal, int y);
```

- Σαν επιστρεφόμενες τιμές από συναρτήσεις ως δείκτες:

```
struct employee *newEmployee(char *name, char *pos,
                             char *id, double sal, int y);
```

Δείκτες σε Δομές

- Δήλωση

```
struct employee {
  char *name;
  char title[20];
  char id[8];
  double salary;
  int years;
};
```

- Δήλωση της μεταβλητής d ως δομή και της e ως δείκτη σε δομή

```
struct employee d, *e;
e = &d;
```

- Δείκτης σε πίνακα από 100 δομές employee

```
e = (struct employee *) malloc(100 * sizeof(struct employee));
e[0].salary = 1000;
```

Δείκτες σε Δομές (2)

- Έστω ο δείκτης `struct employee *e;`
- Πρόσβαση στα περιεχόμενα του δείκτη
  - `*e`: τα περιεχόμενα του δείκτη είναι η δομή
  - `(*e).title` το πεδίο `title` της δομής που δείχνει ο δείκτης `e`
  - `e->title`: συνώνυμο του παραπάνω
  - `e[0].title`: συνώνυμο του παραπάνω
- Προσοχή στις προτεραιότητες των τελεστών:
  - `(*e).title` σημαίνει `e->title`
  - Το `*e.title` είναι ισοδύναμο με `*(e.title)` που δεν είναι αποδεκτό για την συγκεκριμένη δήλωση του `e`

Πράξεις σε Δομές

- Έστω οι δομές:

```
struct point
{
  int x;
  int y;
} p1, p2;
```

- Αντιγραφή:
  - `p1 = p2` – τα περιεχόμενα του `p1` παίρνουν τις τιμές που έχουν στο `p2`
- Απόδοση τιμής:
  - `p1 = {42, 5}` – ταυτόχρονη ανάθεση στα πεδία της δομής
- Διεύθυνση:
  - `&p1` – η διεύθυνση της δομής, δείκτης σε δομή
- Προσπέλαση ενός πεδίου:
  - `&p1.x` – ανάγνωση του πεδίου `x` της δομής
- *Δεν επιτρέπεται η σύγκριση δομών*

typedef

- Ορισμός νέου ονόματος για ένα τύπο

Γενική μορφή

typedef type name;

- Δημιουργεί το `name` ως συνώνυμο του τύπου `type`

Παράδειγμα

typedef unsigned long int seconds_t;

  - Το όνομα `seconds_t` είναι συνώνυμο του `unsigned long int`
  - Αντί να δηλώσουμε μεταβλητές `unsigned long int x;` ...
  - μπορούμε να δηλώσουμε `seconds_t x;`

Παράδειγμα typedef

- Δήλωση τύπου δομής, και μετά δήλωση ψευδωνύμου

typedef1.c

```
struct employee {
  char* name;
  char title[20];
}
typedef struct employee employee_t;
```

- Ταυτόχρονη δήλωση τύπου δομής και ψευδωνύμου
- Δημιουργεί το `name` ως συνώνυμο του τύπου `type`

typedef2.c

```
typedef struct employee {
  char* name;
  char title[20];
} employee_t;
```

  - Ισοδύναμο με το παραπάνω
  - Αν έλειπε το `typedef` θα ήταν δήλωση μεταβλητής!