Διάλεξη 20η: Αναδρομικές δομές

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Αναδρομικές Δομές

- Αυτοαναφορικές ή αναδρομικές δομές
- Δομές που περιέχουν ως μέλη δείκτες σε δομές ίδιου τύπου

```
struct node
{
  int x;
  struct node *next;
};
```

- Μπορούμε να αναφερθούμε στον τύπο της δομής μέσα στην ίδια δομή!
- Μόνο με χρήση δείκτη

Αναδρομικές Δομές (2)

- Με περισσότερους από ένα δείκτες μπορούμε να έχουμε πολύ πολύπλοκες δομές δεδομένων

Παράδειγμα

```
struct person
{
  char name[20];
  int age;
  struct node *father;
  struct node *mother;
  struct node *sibling;
};
```

- Δεν έχουν όλες οι “συνδεσμολογίες” νόημα
- Δουλειά του προγραμματιστή να κατασκευάζει και να συνδέει δομές με “χρήσιμο” τρόπο

Δείκτες σε δομές

- Έστω η δήλωση

```
struct employee *ptr;
```

- Πως μπορεί να χρησιμοποιείται ένας τέτοιος δείκτης;

Δείκτες σε δομές (2)

- Περίπτωση 1: δείκτης σε μια μόνο δομή
  - Στατική δέσμευση

```
struct employee *ptr;
struct employee e;
ptr = &e;
```

  - Δυναμική δέσμευση

```
struct employee *ptr;
ptr = (struct employee *) malloc(sizeof(struct employee));
```

  - Πρόσβαση μέσω δείκτη

```
ptr->name = "Thomas Anderson";
```

Δείκτες σε δομές (3)

- Περίπτωση 2: δείκτης σε πίνακα από δομές
  - Στατική δέσμευση

```
struct employee *ptr;
struct employee a[100];
ptr = a;
ptr = &a[0];
```

  - Δυναμική δέσμευση

```
struct employee *ptr;
ptr = (struct employee *) malloc(100 * sizeof(struct employee));
```

  - Πρόσβαση μέσω δείκτη

```
ptr[42].age = 49;
```

Δείκτες σε δομές (4)

- Περίπτωση 3: δυναμική δομή δεδομένων
  - Η δέσμευση εξαρτάται από τη δομή δεδομένων
  - Η πρόσβαση εξαρτάται από τη δομή δεδομένων
- Παράδειγμα:
  - Σε απλή λίστα

```
ptr->next = (struct employee *) malloc(100 * sizeof(struct employee));
ptr = ptr->next;
```

  - Σε διπλή λίστα

```
strcpy(ptr->prev->id, "neo");
```

Παράδειγμα

maketable.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

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

typedef struct employee employee_t;

employee_t *everyone[4];

const char *Admin = "Administrator";
const char *Prog = "Programmer";
const char *Mgr = "Manager";
```

Παράδειγμα (2)

maketable.c

```
employee_t *create_emp(const char *name, const char *title,
                       const char *id, double salary, int years)
{
  employee_t *new;

  new = (employee_t *) malloc(sizeof(employee_t));
  new->name = (char*) malloc(strlen(name) + 1);
  strcpy(new->name, name);
  strcpy(new->title, title);
  strcpy(new->id, id);
  new->salary = salary;
  new->years = years;
}

void free_emp(employee_t *emp)
{
  free(emp->name);
  free(emp);
}
```

Παράδειγμα (2)

maketable.c

```
int main()
{
  int i;

  everyone[0] = create_emp(”George M.”, Mgr, ”A123123”, 1500.0, 4);
  everyone[1] = create_emp(”Harry C.”, Prog, ”B123123”, 1000.0, 4);
  everyone[2] = create_emp(”Mary S.”, Prog, ”Z123123”, 1000.0, 3);
  everyone[3] = create_emp(”Nick K.”, Admin, ”H123123”, 800.0, 1);

  for(i = 0; i < 4; i++) {
    printf(”%15s %15s %10s %10.2f %4d\n”,
           everyone[i]->name, everyone[i]->title, everyone[i]->id,
           everyone[i]->salary, everyone[i]->years);
    free_emp(everyone[i]);
    everyone[i] = NULL;
  }
  return 0;
}
```

Παράδειγμα (3)

Έξοδος

```
$ gcc -ansi -pedantic -Werror maketable.c
$ ./a.out
      George M.          Manager    A123123     1500.00     4
       Harry C.       Programmer    B123123     1000.00     4
        Mary S.       Programmer    Z123123     1000.00     3
        Nick K.   Administrator    H123123      800.00     1
$
```