ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

ΑΡΧΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κεφάλαιο 15

Επιμέλεια:
Βασίλης Παλιουράς , Αναπληρωτής Καθηγητής
Ευάγγελος Δερματάς , Αναπληρωτής Καθηγητής
Σταύρος Νούσιας , Βοηθός Ερευνητή
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών

Άδειες Χρήσης

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

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου των διδασκόντων καθηγητών.
- Το έργο «Ανοικτά Ακαδήμαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών» έχει χρηματοδοτηθεί μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκη Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικους πόρους.

Ανάπτυξη

Το παρόν εκπαιδευτικό υλικό αναπτύχθηκε στο τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών

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

Σκοπός

Στόχος της παρακάτω ενότητας είναι

Όχι καλό

```
#include <stdio.h>

int main(int argc, char *argv[]) {
   int a, b, c, d, e;

   do {
      scanf("%d", &a);
      scanf("%d", &b);
      scanf("%d", &c);
      scanf("%d", &d);
      scanf("%d", &e);
   } while ((a<5 || a>7) ||
       (b<5 ||  b>7) ||
       (c<5 || c>7) ||
        (d<5 || d>7) ||
        (e<5 || e>7) )  ;

   return 0;
}
```

Καλύτερο

```c
#include <stdio.h>
int getnum(void);

int main(int argc, char *argv[]) {
    int a, b, c, d, e;

    a = getnum();
    b = getnum();
    c = getnum();
    d = getnum();
    e = getnum();

    return 0;
}
int getnum(void) {
    int a;
    while (scanf("%d", &a), a<5 || a >7) ;
    return a;
}
```

Γενικότερος κώδικας

```
#include <stdio.h>
#define N 5
int getnum(void);

int main(int argc, char *argv[]) {
        int a[N], i;

        for (i=0;i < N; i++)
                a[i] = getnum();

        return 0;
}

int getnum(void) {
        int a;
        while (scanf("%d", &a), a<5 || a >7) ;
        return a;
}
```

Διαβάζω μια λέξη – v0

Ζητείται ένα πρόγραμμα που να διαβάζει λέξεις από αρχείο και στη συνέχεια αφού τις αποθηκεύσει να τις τυπώσει.

Ακολουθούμε διαδικασία bottom-up

```
#include <stdio.h>

int main () {
        FILE *myfile ;
        char aword[20];


        myfile = fopen ("test.txt", "rt");

        fscanf(myfile, "%s", aword);
        printf("%s\n", aword);

        fclose(myfile);

        return 0;
}
```

Διαβάζω πολλές λέξεις v1

```c
#include <stdio.h>

int main () {
        FILE *myfile ;
        char aword[20];


        myfile = fopen ("test.txt", "rt");

        while (fscanf(myfile, "%s", aword)!=EOF)
                printf("%s\n", aword);

        fclose(myfile);

        return 0;
}
```

Διαβάζω πολλές λέξεις και τις αποθηκεύω - v2

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

int main () {
        FILE *myfile ;
        char aword[20];
        char manywords[100][20];
        int i=0;

        myfile = fopen ("test.txt", "rt");

        while (fscanf(myfile, "%s", aword)!=EOF) {
                strcpy(manywords[i++], aword);
                printf("%s\n", aword);
                }

        fclose(myfile);

        return 0;
}
```

Διαβάζω λέξεις και μετά τις τυπώνω v3

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

int main () {
          FILE *myfile ;
          char aword[20];
          char manywords[100][20];
          int j, i=0;

          myfile = fopen ("test.txt", "rt");

          while(fscanf(myfile, "%s", aword)!=EOF) {
                    strcpy(manywords[i++], aword);
                    }

          for (j=0; j<i;j++)
                     printf("%s\n", manywords[j]);

          fclose(myfile);

          return 0;
}
```

Refactoring v4

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

int readdtext(FILE *f, char text[100][20]);
void displaytext (char text[100][20], int words);

int main () {
  FILE *myfile ;
  char manywords[100][20];
  int wordcount;

  myfile = fopen ("test.txt", "rt");
  wordcount = readtext(myfile, manywords);

  displaytext(manywords, wordcount);
  fclose(myfile);

  return 0;
}
```

Διαδικασία bottom up

Εδώ φτιάχνουμε
Συνάρτηση που διαβάζει
Και μετράει τις λέξεις

```c
int readtext(FILE *f, char manywords[100][20]) {
  char aword[20];
  int i = 0;
    while(fscanf(f, "%s", aword)!=EOF) {
          strcpy(manywords[i++], aword);
        }
    return i;
}
void displaytext(char manywords[100][20], int words) {
 int j;
 for (j=0; j<words; j++)
    printf("%s\n", manywords[j]);
 }
```

Απλοποίηση διεπαφών με δομές

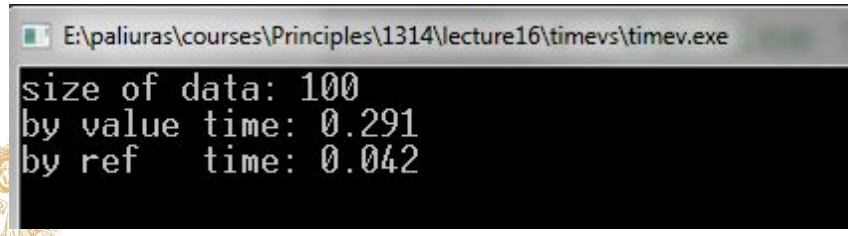
```c
#include <stdio.h>
#include <string.h>
typedef struct text {
  char manywords[100][20];
  int wordcount;
} Text;
void displaytext(Text mytext);
Text readtext (FILE *f);

int main () {
  FILE *myfile ;
  Text mytext;

  myfile = fopen ("test.txt", "rt");

  mytext = readtext (myfile);

  displaytext(mytext);

  fclose(myfile);

  return 0;
}
```

```c
Text readtext (FILE *f)  {
 char aword[20];
 int i = 0;
  Text text;
   while (fscanf(f, "%s", aword)!=EOF)
      strcpy( text.manywords[i++] , aword);
   text.wordcount = i;
 return text;
}

void displaytext(Text text) {
 int j;
 for (j=0; j <text.wordcount; j++)
  printf("%s\n", text.manywords[j]);
}
```

Τι είναι ταχύτερο;

```c
#include <stdio.h>
#include <time.h>
#define TIMES 10000000

typedef struct test {
            char data[100];
} Test;

void byvalue(Test a) {
          Test b;
          /* do something */
}

void byref(Test *a ) {
          Test b;
          /* do something */
}
```

```c
int main(int argc, char *argv[]) {
  Test a ;
  int i;
  clock_t start, stop;

  start = clock();
  printf("size of data: %d\n", sizeof (Test));
  for (i=0;i<TIMES; i++)
      byvalue(a);
  stop = clock();
  printf("by value time: %g\n",
      (double) (stop - start)/CLOCKS_PER_SEC);

  start = clock();
  for (i=0;i<TIMES; i++)
            byref(&a);
  stop = clock();
  printf("by ref   time: %g\n",
            (double) (stop - start)/CLOCKS_PER_SEC);

  return 0;
}
```



Κλήση με αξία και
Κλήση με αναφορά

Δυναμική διαχείριση μνήμης στη C

- Δέσμευση μνήμης:
  - **void** *malloc(size_t size);
  - Επιστρέφει δείκτη σε εξασφαλισμένη περιοχή μεγέθους size bytes ή NULL αν δεν υπάρχει τέτοια.
- Απελευθέρωση μνήμης:
  - **void** free(**void** *pointer);

Πώς δουλεύει ο μηχανισμός;

- Χρησιμοποιεί
  - static δεδομένα
  - Λεπτομερή διαχείριση ανά block
    - Διεύθυνση αρχής
    - Μέγεθος
- Μοιράζεται πληροφορία μεταξύ διαφορετικών συναρτήσεων
  - malloc ( ) , free ( )
  - Πώς γίνεται αυτό;

Παράδειγμα

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

main ( ) {

char matrix[N];

scanf("%s", matrix);

printf("Hello %s!\n", matrix);


}
```

Παράδειγμα

```c
#include <stdio.h>
#include <stdlib.h>
#define N 10

main ( ) {

char matrix[N];
char *dynamicdata;

scanf("%s", matrix);

printf("Hello %s!\n", matrix);

dynamicdata = (char *) malloc( N * sizeof (char));

scanf("%s", dynamicdata);

printf("Hello dynamic %s!", dynamicdata);


}
```

Παράδειγμα

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

main ( ) {

char matrix[N];
char *dynamicdata;
int i;

scanf("%s", matrix);

printf("Hello %s!\n", matrix);

dynamicdata = (char *) malloc( N * sizeof (char));
scanf("%s", dynamicdata);

printf("Hello dynamic %s!\n", dynamicdata);

for (i=0;dynamicdata[i]!=0;i++)
      printf("%c\n", dynamicdata[i]);


}
```

Παράδειγμα

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

main ( ) {
char matrix[N];
char *dynamicdata;
int i, nchars;

scanf("%s", matrix);
printf("Hello %s!\n", matrix);

while (1) {
     printf("How many chars?");
     scanf("%d", &nchars);
     dynamicdata = (char *) malloc( nchars * sizeof (char));
     scanf("%s", dynamicdata);
     printf("Hello dynamic %s!\n", dynamicdata);
     for (i=0;dynamicdata[i]!=0;i++)
             printf("%c\n", dynamicdata[i]);
     free(dynamicdata);
     }
}
```

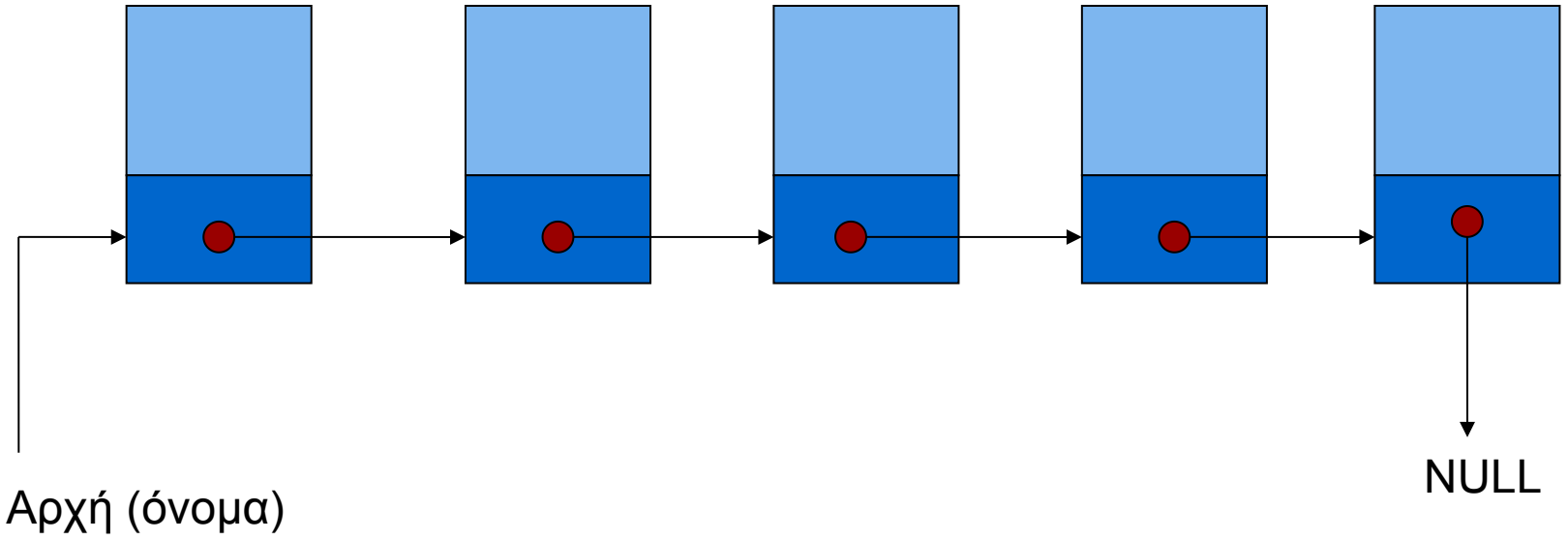
Επόμενα

- Δυναμικές δομές δεδομένων
- Διασυνδεδεμένες λίστες
- Διαχείριση μνήμης (memory management)

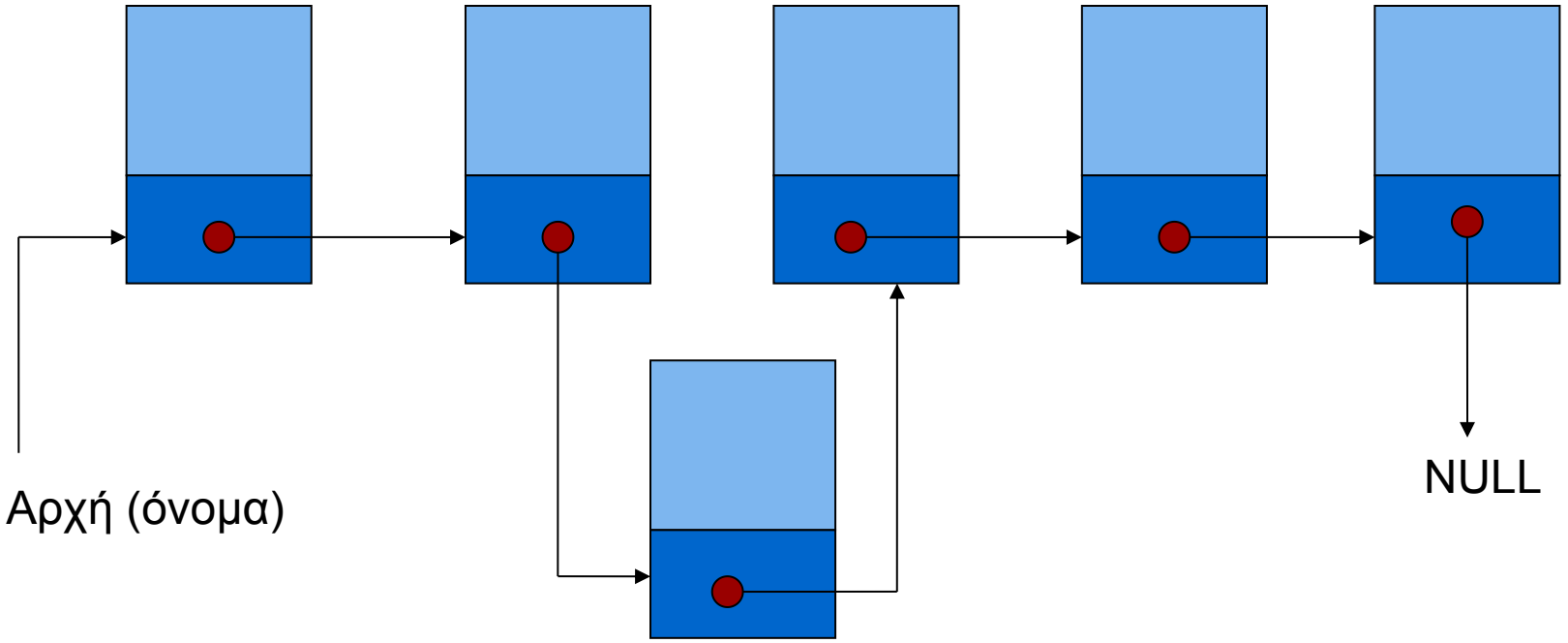
Στατικές και δυναμικές δομές δεδομένων

- **Στατικές**: θέση και μέγεθος καθορίζονται στη μεταγλώττιση.
  - `int array[10];`
- **Δυναμικές**: θέση και μέγεθος καθορίζονται κατά την εκτέλεση.
  - Απλούστερος τύπος: η λίστα
- Λίστα: Έτοιμη σε C++/Java, σε C γράφουμε κώδικα

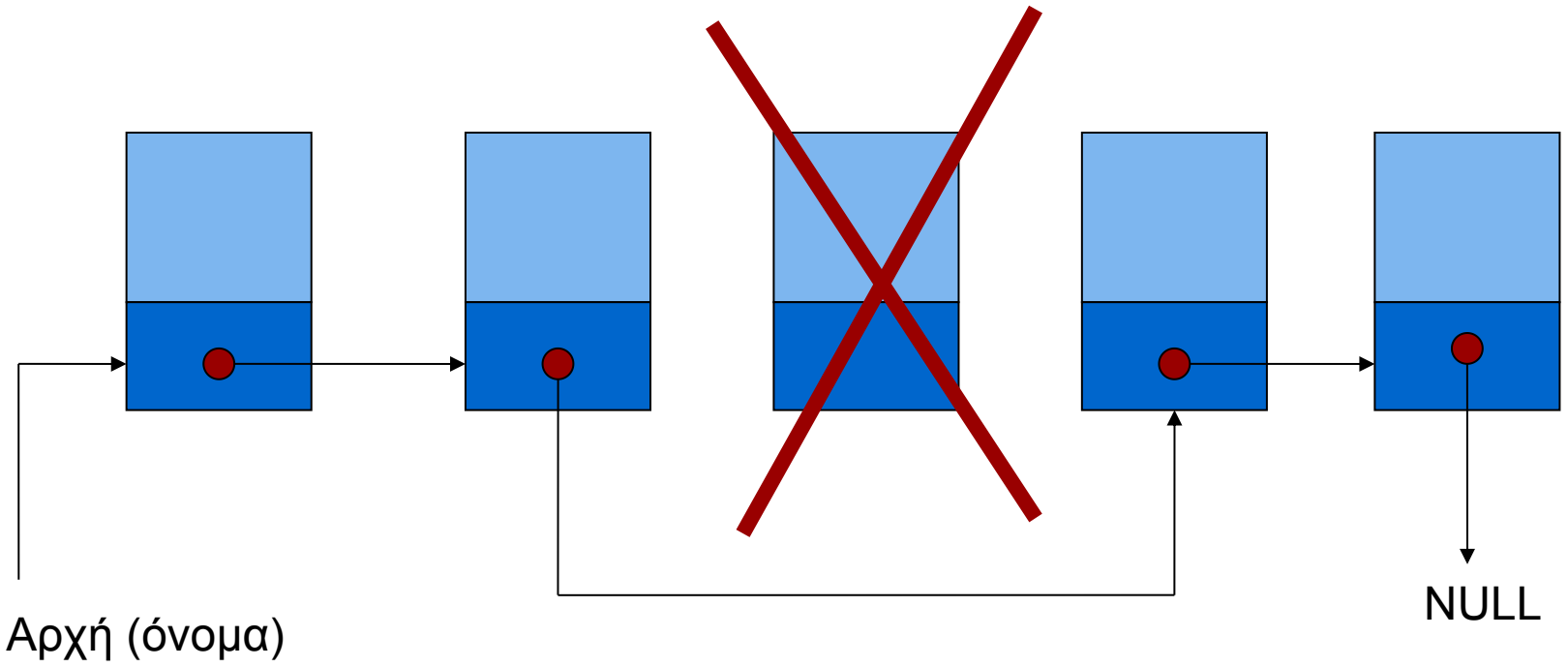
Λίστες και στοιχεία τους



Εισαγωγή στοιχείου



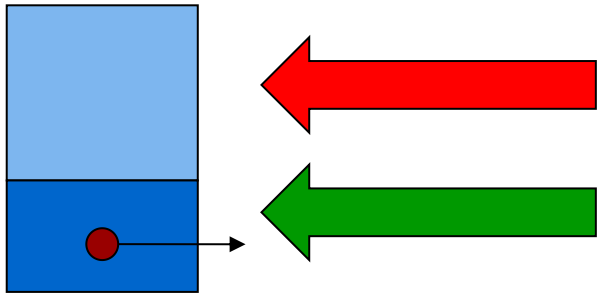
Διαγραφή στοιχείου



Τι γίνεται με τη μνήμη την οποία το στοιχείο καταλάμβανε;

Στοιχείο λίστας

Δεδομένα προβλήματος



Θέση επόμενου στοιχείου

```
#define N 10
struct elem {
    char value[N];
    int count ;
    struct elem *next;
} ;

typedef struct elem Element;
```

Χρήσιμοι συμβολισμοί

```
struct test {
        int a ;
        struct test *next;
        } atest, *atest_ptr, *btest_ptr;
```

Διεύθυνση στοιχείου μετά το `atest -> atest.next`

```
atest.a = 4;
/* δείκτης σε δομή */
atest_ptr = atest.next;
btest_ptr = &atest;
(*btest_ptr).a ++;
/* το ίδιο με το προηγούμενο */
(btest_ptr->a)++ ;
```

Σημείωμα αναφοράς

- Copyright Πανεπιστήμιο Πατρών, Παλιουράς Βασίλειος , Δερματάς Ευάγγελος «Αρχές Προγραμματισμού ». Έκδοση: 1.0. Πάτρα 2015
- Διαθέσιμο από τη δικτυακιακή διέυθυνση https://eclass.upatras.gr/modules/