Διάλεξη 15η: Αναδρομή

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Βασίζεται σε διαφάνειες του Κ. Παναγιωτάκη

Κλήσεις συναρτήσεων

- Όταν καλείται μια συνάρτηση, πρέπει
  - Να θυμάται σε ποιο σημείο του προγράμματος θα επιστρέψει
  - Να δεσμεύσει χώρο για την τιμή που θα επιστρέψει
  - Να δεσμεύσει χώρο για τα ορίσματα της
  - Να δεσμεύσει χώρο για τις τοπικές μεταβλητές της
- Η μνήμη για στατικές και καθολικές μεταβλητές δεσμεύεται από την αρχή της εκτέλεσης του προγράμματος

Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```

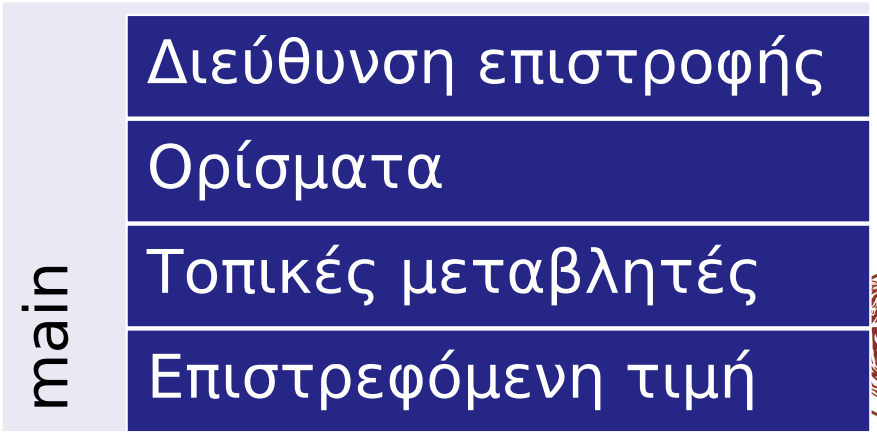
Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10); ←
  int y = g(10);
}
```



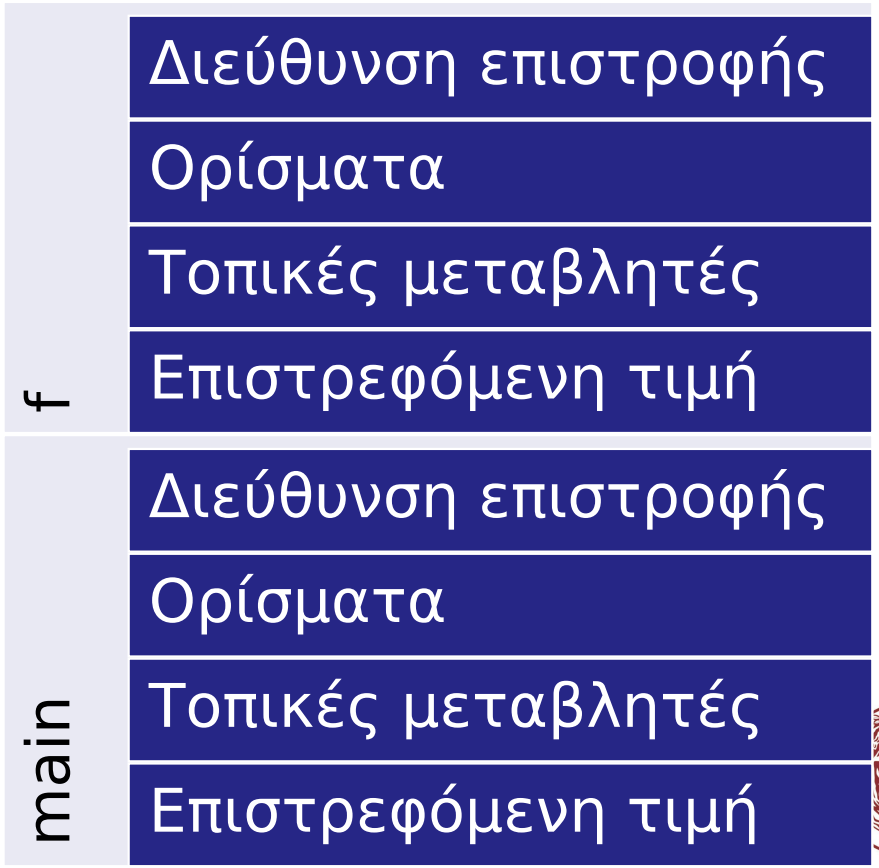
Παράδειγμα

stack-frame.c

```
int f(int a) { ←
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



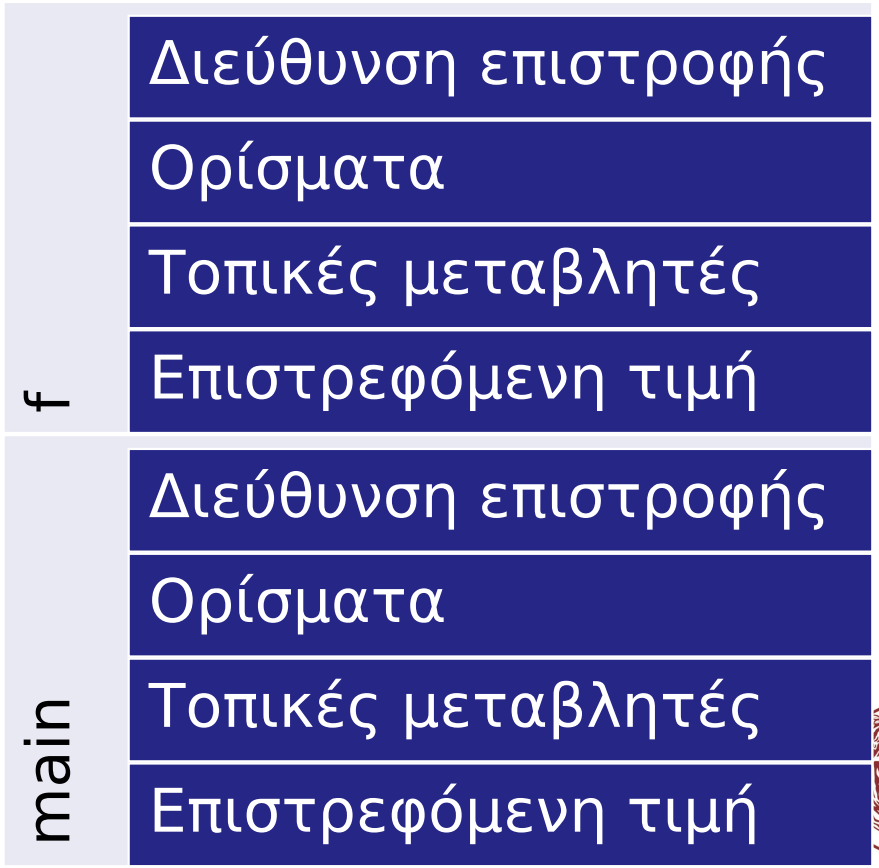
Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2; ←
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



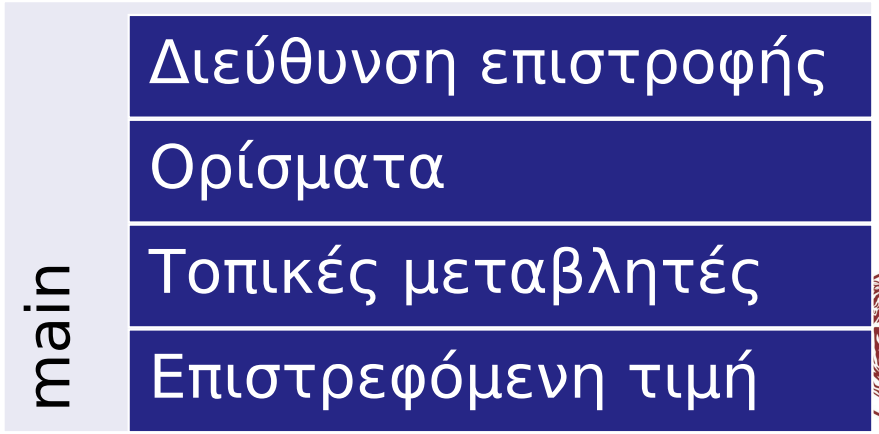
Παράδειγμα

stack-frame.c

```c
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10); ←
}
```



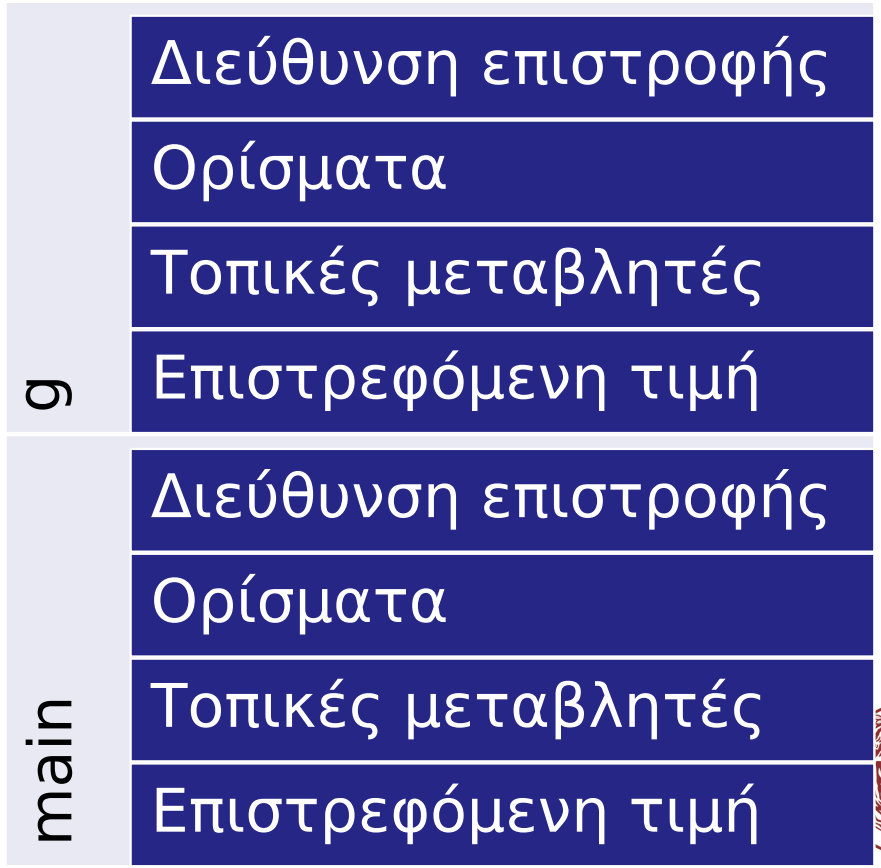
Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



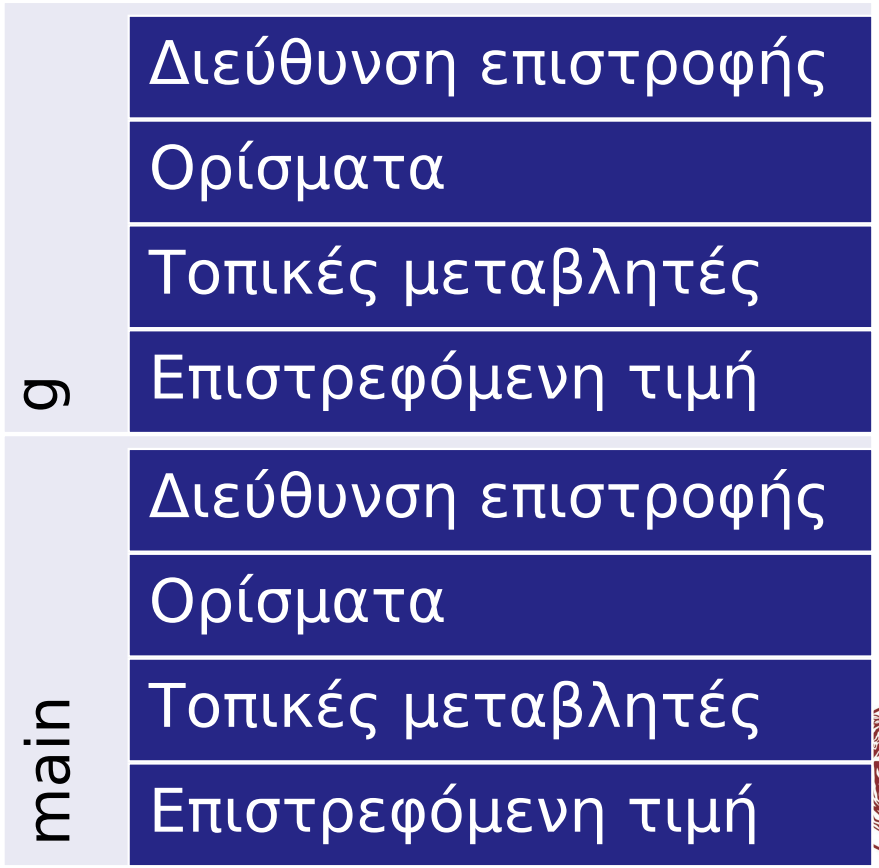
Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3); ←
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



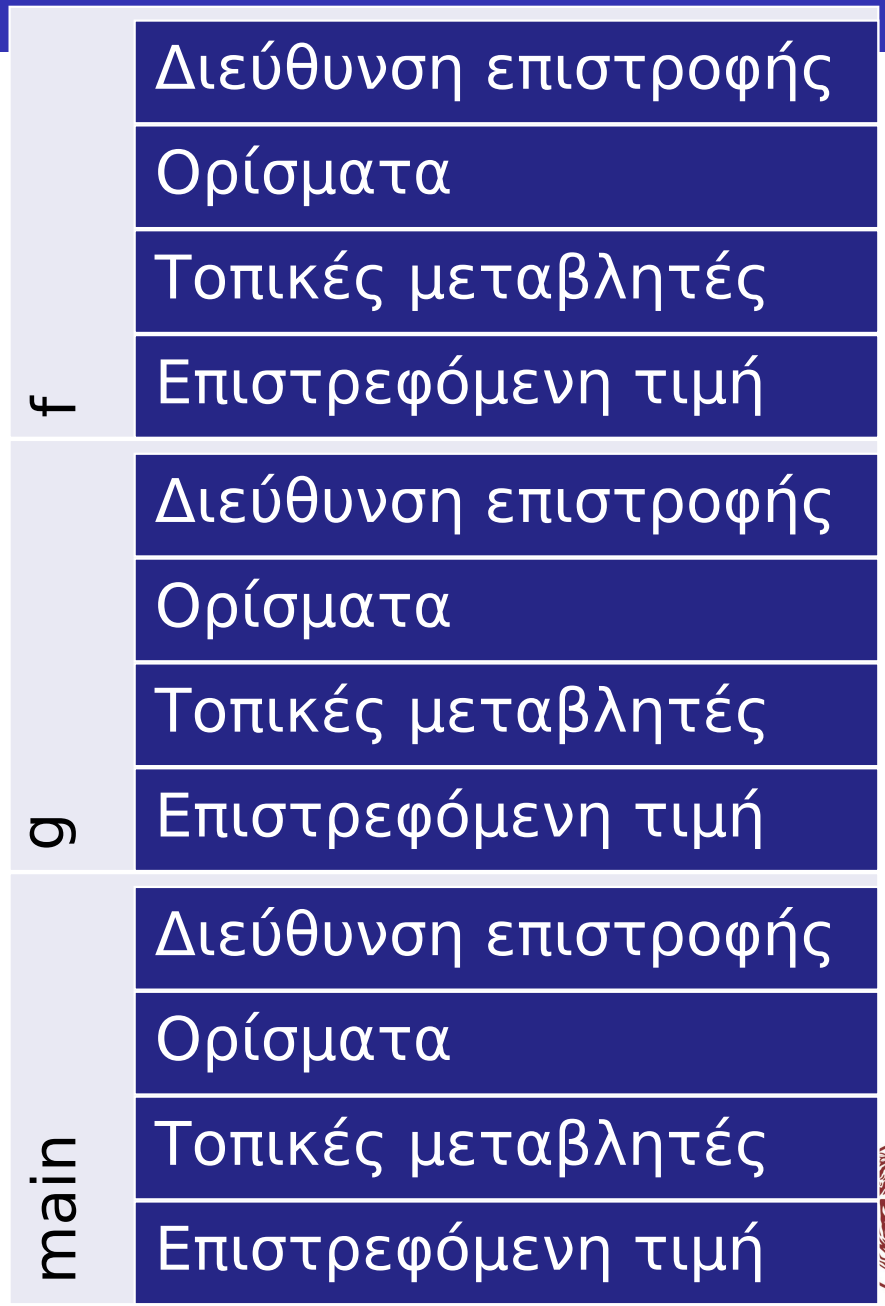
Παράδειγμα

stack-frame.c

```
int f(int a) { ←
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



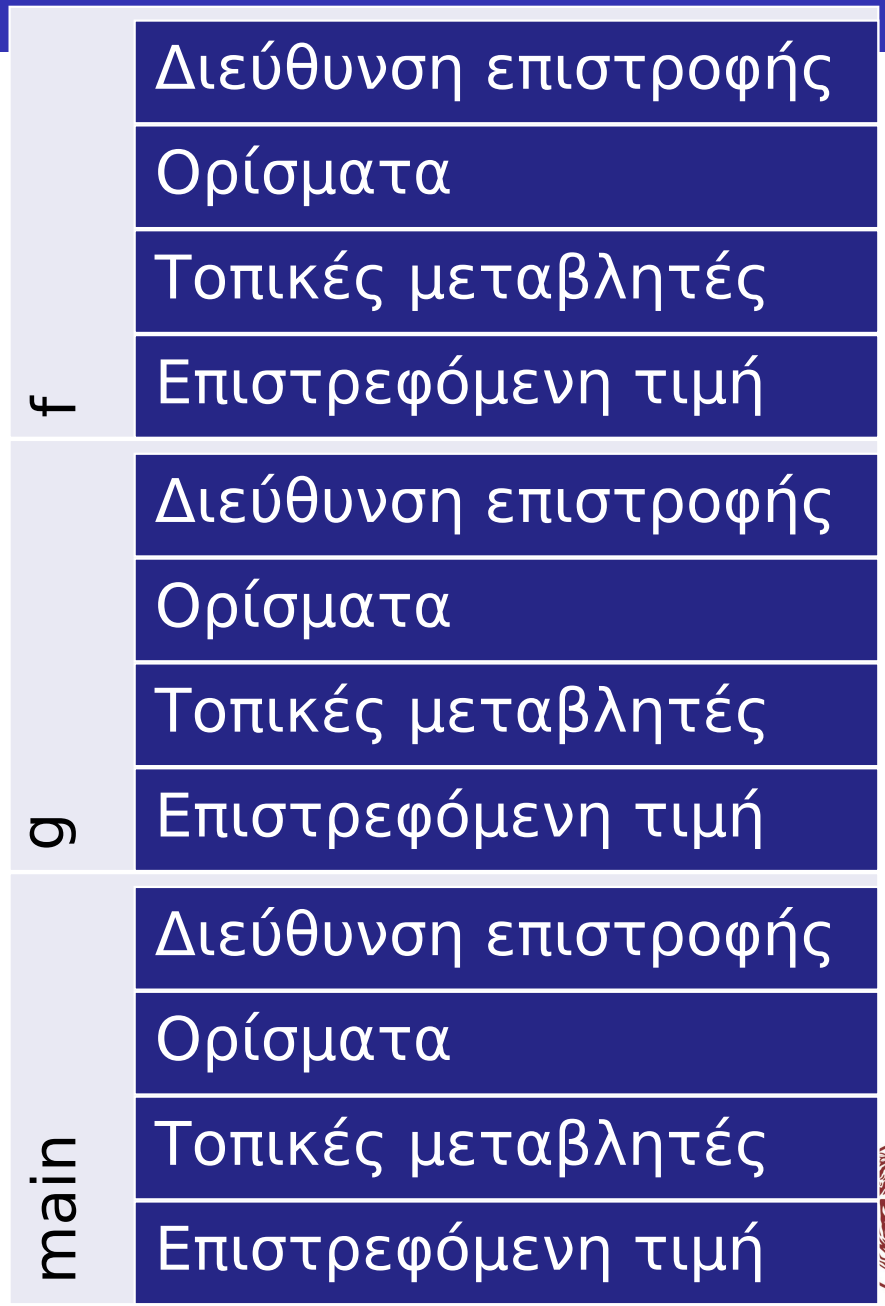
Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2; ←
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



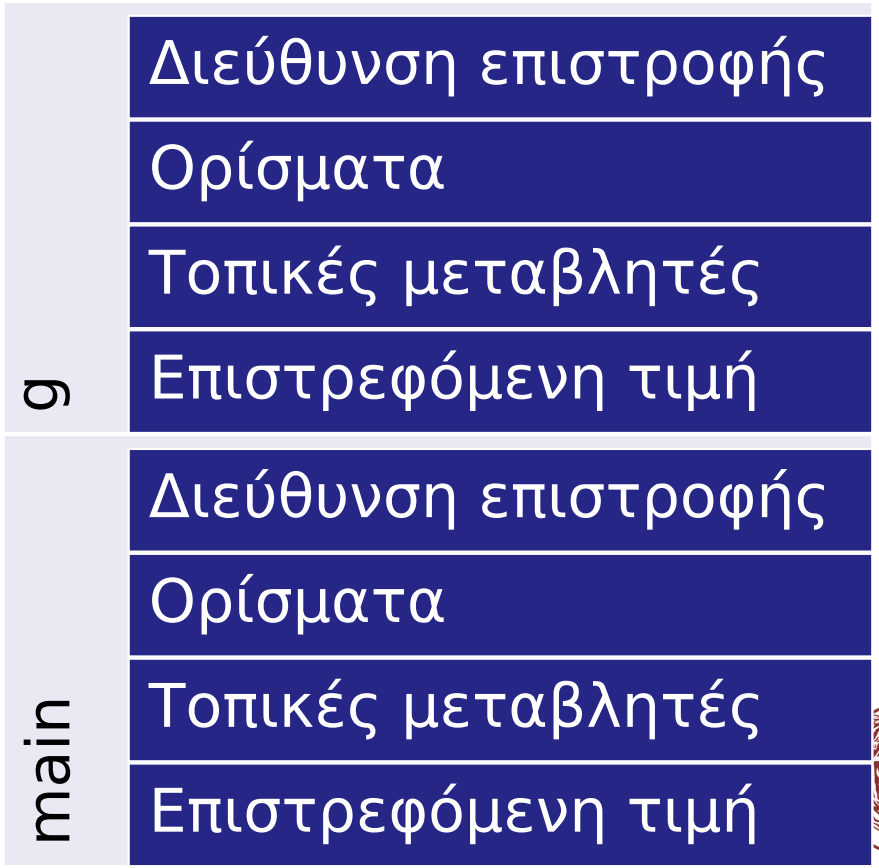
Παράδειγμα

stack-frame.c

```
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3); ←
}

int main(void) {
  int x = f(10);
  int y = g(10);
}
```



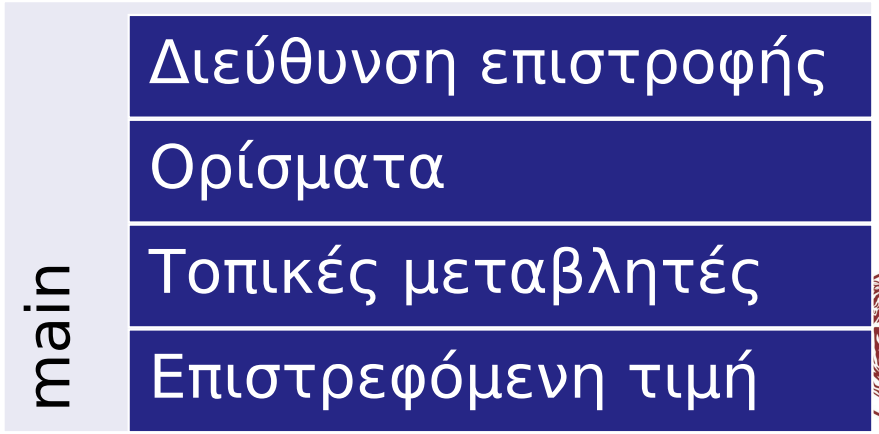
Παράδειγμα

stack-frame.c

```c
int f(int a) {
  return a + 2;
}

int g(int b) {
  return (f(b) + 3);
}

int main(void) {
  int x = f(10);
  int y = g(10); ←
}
```



Το δέντρο κλήσεων

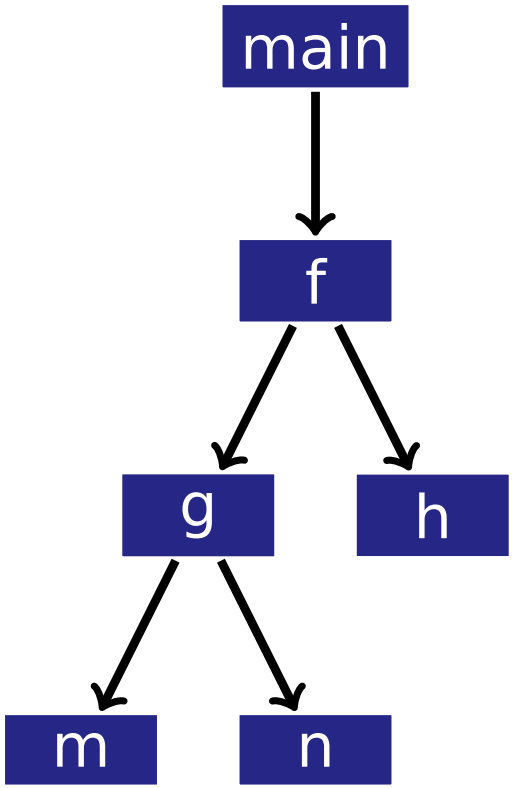
cg.c

```
int m(int);
int n();
int h();

int g() {
  m(n());
}

void f() {
  g() + h();
}

void main() {
  f();
}
```



Αναδρομή

- Γιατί σπάμε το πρόγραμμα σε συναρτήσεις;
- Κάθε συνάρτηση λύνει ένα “μικρότερο” υποπρόβλημα
- Συνδυάζουμε τα μικρότερα προβλήματα με τέτοιο τρόπο που να λύσουμε το συνολικό πρόβλημα
- Παράδειγμα: Φτιάξτε ένα πρόγραμμα που να διαχειρίζεται μια βάση δεδομένων με φοιτητές
  - Συνάρτηση για διαχείριση της εισόδου του χρήστη
  - Συνάρτηση για εισαγωγή/διαγραφή/αλλαγή/αναζήτηση στοιχείων
  - Συνάρτηση για εκτύπωση της βάσης δεδομένων
  - κλπ.

Αναδρομή (2)

- *Αναδρομική Συνάρτηση* είναι μια συνάρτηση που έμμεσα ή άμεσα καλεί τον εαυτό της
  - Κάθε υπο-κλήση μιας συνάρτησης λύνει ένα “μικρότερο” υπο-πρόβλημα *του ίδιου τύπου*
  - Συνδυάζουμε τα μικρότερα προβλήματα με τέτοιο τρόπο που να λύσουμε το συνολικό πρόβλημα
  - Λύνουμε τα “πολύ μικρά προβλήματα” απευθείας
- Που χρησιμεύουν οι αναδρομικές συναρτήσεις
  - Ο κώδικας γίνεται συνήθως πολύ πιο απλός
  - Λύνει πολύ εύκολα προβλήματα που θα φαινόταν πολύ δύσκολα με επανάληψη
  - Το μέγεθος σε γραμμές κώδικα της συνάρτησης συνήθως ελαχιστοποιείται

Παράδειγμα: Fibonacci

- Αριθμοί Fibonacci
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Παράδειγμα: Fibonacci

- Αριθμοί Fibonacci
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
- Ορισμός της ακολουθίας
  - Ο πρώτος αριθμός Fibonacci είναι το 0
  - Ο δεύτερος αριθμός Fibonacci είναι το 1
  - Για n > 1, ο n-οστός αριθμός Fibonacci είναι το άθροισμα των 2 προηγούμενων
- Αναδρομικός ορισμός

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

Παράδειγμα: Fibonacci (2)

fibonacci.c

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

int fib(int n) {
  if(n == 0) {
    return 0;
  }
  if(n == 1) {
    return 1;
  }
  return (fib(n-1) + fib(n-2));
}

int main(int argc, char **argv) {
  int n;
  if(argc != 2) {
    printf("usage: %s <number>\n", argv[0]);
    exit(0);
  }
  n = atoi(argv[1]);
  printf("The %d-th Fibonacci number is %d\n", n, fib(n));
}
```

Αναδρομή

- Αναδρομική Συνάρτηση είναι μια συνάρτηση που έμμεσα ή άμεσα καλεί τον εαυτό της
- Κάθε κλήση λύνει ένα “μικρότερο” υπόπρόβλημα του ίδιου τύπου.
- Συνδυάζουμε τα μικρότερα προβλήματα με τέτοιο τρόπο που να λύσουμε το συνολικό πρόβλημα
- Κάποια στιγμή πρέπει να λύσουμε τα πολύ “μικρά” προβλήματα απευθείας

“To iterate is human, to recurse divine.”
(L. Peter Deutsch)

Άμεση Αναδρομή

- Τρία βασικά συστατικά της αναδρομής
- Για να λύσουμε ένα πρόβλημα μεγέθους n:
  1. Βασική περίπτωση (base-case): Όταν το n είναι αρκετά μικρό δίνουμε μια άμεση (χωρίς άλλη αναδρομή) λύση του "μικρού" προβλήματος
  2. Αναδρομική κλήση: διαιρούμε το πρόβλημα σε "μικρότερα" ίδια προβλήματα και τα λύνουμε με αναδρομική κλήση
  3. Συγχώνευση: συνδυάζουμε τις λύσεις των μικρότερων προβλημάτων, για να λύσουμε το πρόβλημα μεγέθους n
- Το 3ο βήμα δεν είναι απαραίτητο σε όλα τα προβλήματα
- Το μέγεθος του προβλήματος μπορεί να είναι το μέγεθος ενός ορίσματος, το μέγεθος ενός πίνακα, το μέγεθος μιας δομής δεδομένων, κλπ

Αναδρομή και Στοίβα

- Σε αναδρομικές συναρτήσεις η στοίβα μπορεί να περιέχει περισσότερα από ένα stack frames της *ίδιας* συνάρτησης
- “Η αναδρομή είναι θεϊκή” αλλά η κλήση συνάρτησης κοστίζει
- Κάθε αναδρομική συνάρτηση μπορεί να υλοποιηθεί και επαναληπτικά (χωρίς αναδρομή) με χρήση στοίβας

Παράδειγμα: Δύναμη

power1.c

```
int power(int x, int n)
{
  if (n == 0) {
    return 1;
  } else {
    return (x * power(x, n -1));
  }
}
```

Παράδειγμα: Δύναμη (2)

power2.c

```
int power(int x, int n)
{
  int i, result = 1;

  for (i = 1; i <= n; i++) {
    result = result * x;
  }
  return result;
}
```

Παράδειγμα: Παραγοντικό

factorial.c

```
int factorial(int n)
{
  if (n <= 1) {
    return 1;
  }
  return (n * factorial(n -1));
}
```

Παράδειγμα: Παραγοντικό (2)

factorial.c

```
int factorial(int n)
{
  int i, result = 1;

  for (i = 0; i <= n; i++) {
    result = result * i;
  }
  return result;
}
```

Αμοιβαία Αναδρομή

- Μπορεί μια συνάρτηση να καλεί τον εαυτό της έμμεσα καλώντας άλλες συναρτήσεις που την καλούν
- Μια ομάδα συναρτήσεων που καλλούν η μια την άλλη είναι *αμοιβαία αναδρομικές*

oddeven.c

```
#include<stdlib.h>
int odd(int);
int even(int);

int odd(int n) {
  if (n == 0) {
    return 0;
  }
  return even((abs(n) -1);
}
```

```
int even(int n) {
  if (n == 0) {
    return 1;
  }
  return odd(abs(n) -1);
}
```

Αναδρομικοί Αλγόριθμοι

- Ταξινόμηση ενός πίνακα
- Αναζήτηση σε ταξινομημένο πίνακα
- Το μονοπάτι για την έξοδο από το λαβύρινθο
- Μέγιστος κοινός διαιρέτης
- Τα βήματα στον πύργο του Ανόι
- κλπ