ΗΥ240 ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Φροντιστήριο στη C

Εαρινό 2015

Τυποι μεταβλητων

Type	Size
int	4 bytes
float	4 bytes
double	8 bytes
char	1 byte
*	4 bytes
struct	?

```
struct s
{
    int am;
    float average;
};
```

struct s *p;	4 bytes
sizeof(p);	
struct s k;	8 bytes
sizeof(k);	

Παραδειγμα με if…else

```
1  void main (void)
2  {
3      int num1, num2;
4      int choice;
5      printf ("Δώσε τον πρώτο ακέραιο: \n");
6      scanf ("%d", &num1);
7      printf ("Δώσε τον δεύτερο ακέραιο: \n");
8      scanf ("%d", &num2);
9      printf ("Ποια πράξη θέλεις να εκτελέσεις; ");
10     printf ("1 = Πρόσθεση");
11     printf ("2 = Αφαίρεση");
12     printf ("3 = Πολλαπλασιασμός");
13     printf ("Επέλεξε: \n");
14     scanf ("%d",&choice);
15     if (choice == 1)
16         printf ("%d+%d=%d\n",num1,num2,num1+num2);
17     else if (choice == 2)
18         printf ("%d-%d=%d\n",num1,num2,num1-num2);
19     else if (choice == 3)
20         printf ("%d*%d=%d\n",num1,num2,num1*num2);
21     else
22         printf ("Λάθος Επιλογή");
23 }
```

Καθολικες-Τοπικες Μεταβλητες

- Οι μεταβλητές ενός προγράμματος διακρίνονται σε δύο κατηγορίες:
  - καθολικές μεταβλητές: δηλώνονται στην αρχή κάθε προγράμματος πριν τις συναρτήσεις και μπορούν να προσπελάσονται από όλες τις συναρτήσεις.
  - τοπικές ή εσωτερικές μεταβλητές κάθε συνάρτησης

```
1  int a = 100; // Μία καθολική μεταβλητή
2
3  void f1 (void)
4  {
5      int a = 300; // Μία τοπική μεταβλητή
6      printf ("Η a στην f1 έχει τιμή: %d", a);
7  }
8
9  void main (void)
10 {
11     int b = 4; // Μία τοπική μεταβλητή
12     printf ("Η a στην main έχει τιμή: %d", a);
13     f1();
14 }
```

Έξοδος παραδείγματος:

```
Η a στην main έχει τιμή: 100
Η a στην f1 έχει τιμή: 300
```

Παραδειγμα swap

```c
#include <stdio.h>

void swap(int i,int j);

main()
{
    int x;
    int y;

    x=10;
    y=12;

    printf("x=%d\ty=%d\n",x,y);
    swap(x,y);
    printf("x=%d\ty=%d\n",x,y);
}


void swap(int i,int j)
{
    int temp;

    temp=i;
    i=j;
    j=temp;
}
```

```c
#include <stdio.h>

void swap(int *i,int*j);

main()
{
    int x;
    int y;

    x=10;
    y=12;

    printf("x=%d\ty=%d\n",x,y);
    swap(&x,&y);
    printf("x=%d\ty=%d\n",x,y);
}


void swap(int *i,int *j)
{
    int temp;

    temp=*i;
    *i=*j;
    *j=temp;
}
```

Παραδειγμα pointers

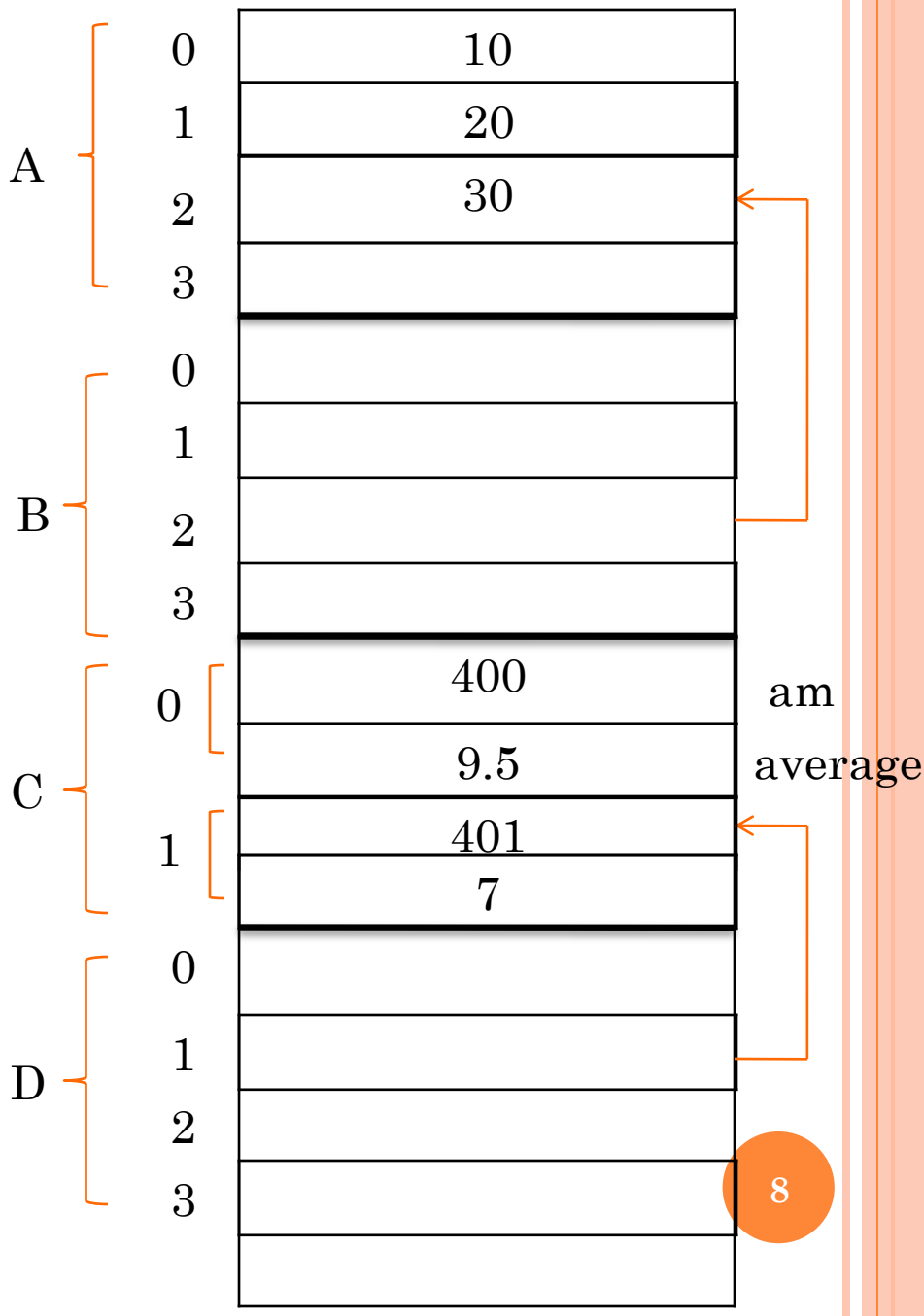
```
1   void main (void)
2   {
3       int x;
4       int *px;
5       x = 10;
6       px = &x;
7
8       printf ("x = %d", x);
9       printf ("px = %u", px);
10      printf ("*px = %d", *px);
11  }
```

Παραδειγμα pointers

```
1  struct s {
2      int am;
3      float average;
4  };
5
6  void main (void)
7  {
8      struct s student;
9      struct s *pstudent;
10     pstudent = &student;
11     pstudent->am = 400;
12     pstudent->average = 9.5;
13     printf ("A.M.:%d -- M.O.: %f", pstudent->am, student->average);
14 }
```

ΠΙΝΑΚΕΣ

```
1  struct s {
2      int am;
3      float average;
4  };
5
6  void main (void)
7  {
8      int A[4];
9      int *B[4];
10     struct s C[2];
11     struct s *D[4];
12
13     A[0] = 10;
14     A[1] = 20;
15     B[2] = &A[2];
16     *B[2] = 30;
17     C[0].am = 400;
18     C[0].average = 9.5;
19     D[1] = &C[1];
20     D[1]->am = 401;
21     D[1]->average = 7;
22 }
```

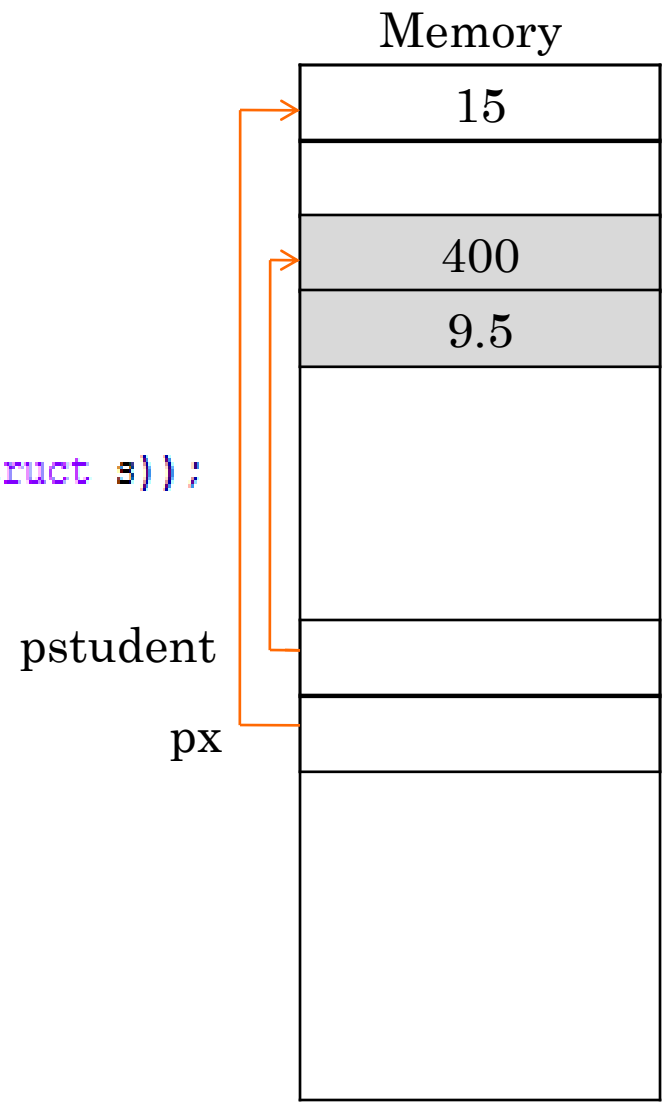


Malloc/stdlib.h

- Επιτρέπει τη ***δυναμική (κατά την εκτέλεση του προγράμματος) δέσμευση μνήμης***
  - παίρνει ως όρισμα το **μέγεθος της μνήμης που θα δεσμευθεί**
  - χρήση της συνάρτησης **sizeof, π.χ. sizeof(int), sizeof(char), …**
  - επιστρέφει έναν δείκτη στην αρχή της δεσμευμένης μνήμης (ή null σε περίπτωση αποτυχίας)
  - ο δείκτης αυτός είναι τύπου **void (casting)**

Παραδειγμα malloc

```
1   void main (void)
2   {
3       int *px;
4       struct s *pstudent;
5       px = (int *) malloc (sizeof(int));
6       pstudent = (struct s *) malloc(sizeof(struct s));
7       *px = 15;
8       pstudent -> am = 400;
9       pstudent -> average = 9.5;
10  }
```

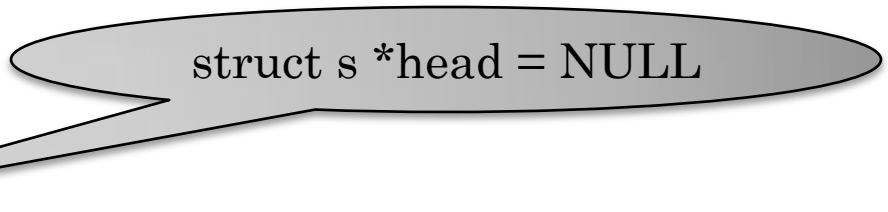


Λιστεσ insert

```c
struct s
{
    int am;
    float average;
    struct s *next;
};

struct s *insert1(struct s *head, int am, float average)
{
    struct s *newnode;

    newnode = malloc(sizeof(struct s));
    newnode->am = am;
    newnode->average = average;
    newnode->next = head;
    head = newnode;
    return head;
}

int main()
{
    struct s *head=NULL;
    int am1=1, am2=2, am3=3;
    float average1=15, average2=12, average3=13;

    head = insert1(head, am1, average1);
    head = insert1(head, am2, average2);
    head = insert1(head, am2, average2);
}
```

Λιστεσ insert

```c
struct s
{
    int am;
    float average;
};

void insert2(struct s **head, int am, float average)
{
    struct s *newnode;

    newnode = malloc(sizeof(struct s));
    newnode->am = am;
    newnode->average = average;
    newnode->next = (*head);
    (*head) = newnode;
    return;
}

int main()
{
    struct s *head;
    struct s*tmp;
    int am1=1, am2=2, am3=3;
    float average1=15, average2=12, average3=13;

    insert2(&head, am1, average1);
    insert2(&head, am2, average2);
    insert2(&head, am2, average2);

    tmp = head;
    while(tmp != NULL){
        printf("%d %lf\n", tmp->am, tmp->average);
        tmp=tmp->next;
    }
}
```

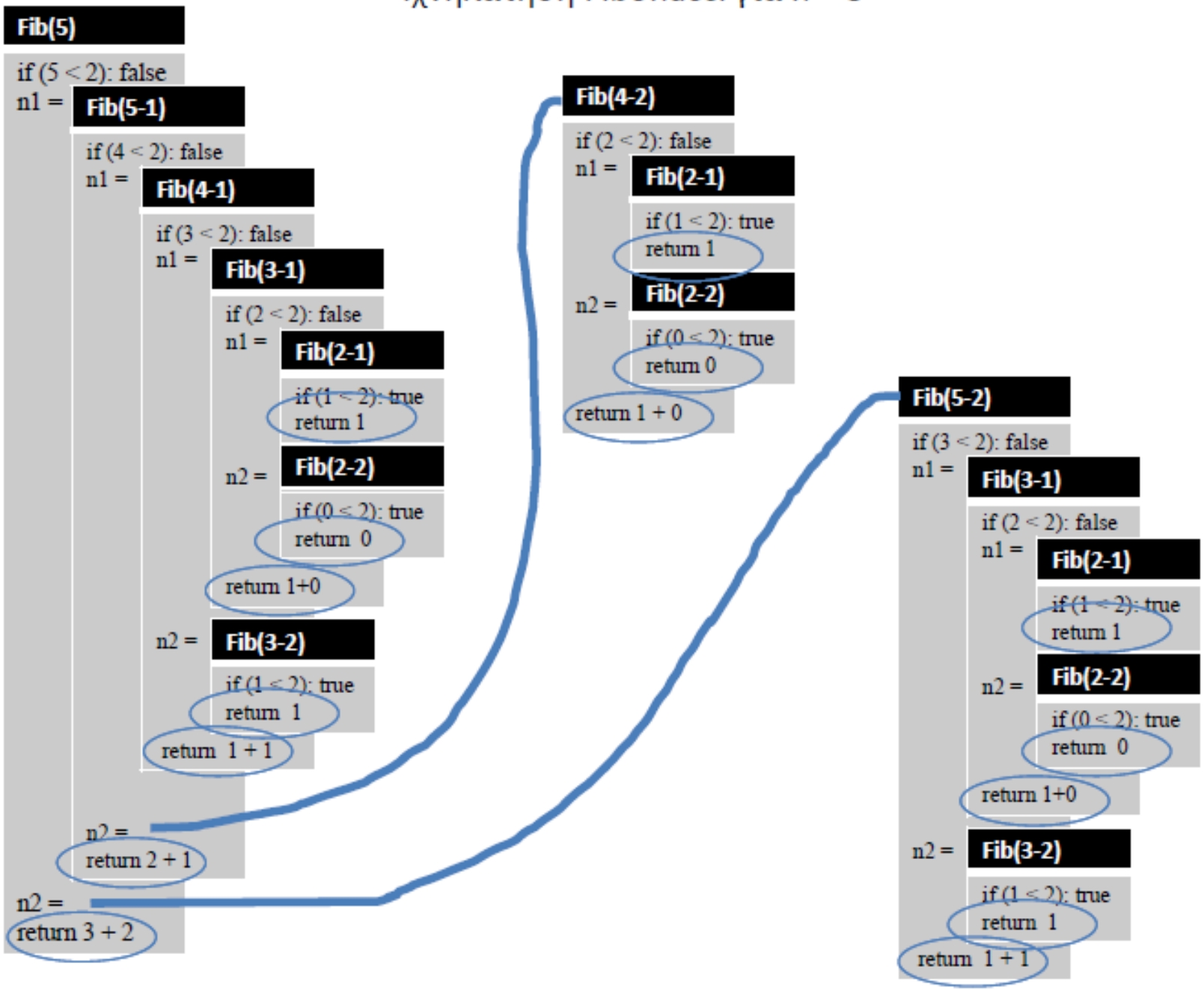


Αναδρομη Fibonacci

$$F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases}$$

```
int Fib(int n) {
   if (n < 2)
      return n;
   else {
      int n1 = Fib(n-1);
      int n2 = Fib(n-2);
      return n1 + n2;
   }
}
```

Ιχνηλάτηση Fibonacci για n = 5



Αθροισμα στοιχειων πινακα

```
1  #include<stdio.h>
2  int recursion( int *A, int size )
3  {
4          static int sum = 0;
5          if (size < 0)
6          {
7                  return sum;
8          }
9          else
10         {
11                 sum = sum + A[size];
12         }
13         recursion(A,size-1);
14 }
15
16 int main()
17 {
18         int A[] = {1,2,3,4,5,6,7,8};
19         printf("sum = %d\n",recursion(A,4));
20         return 0;
21 }
```