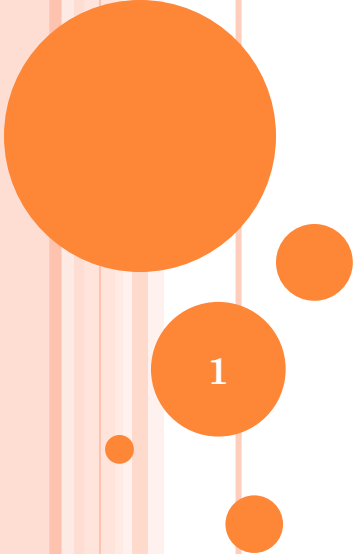




ΗΥ240 ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Φροντιστήριο στη C

1

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

```
1  #include <stdio.h>
2
3  void swap(int i,int j);
4
5  main()
6  {
7      int x;
8      int y;
9
10     x=10;
11     y=12;
12
13     printf("x=%d\ty=%d\n",x,y);
14     swap(x,y);
15     printf("x=%d\ty=%d\n",x,y);
16 }
17
18
19 void swap(int i,int j)
20 {
21     int temp;
22
23     temp=i;
24     i=j;
25     j=temp;
26 }
```

```
1  #include <stdio.h>
2
3  void swap(int *i,int *j);
4
5  main()
6  {
7      int x;
8      int y;
9
10     x=10;
11     y=12;
12
13     printf("x=%d\ty=%d\n",x,y);
14     swap(&x,&y);
15     printf("x=%d\ty=%d\n",x,y);
16 }
17
18
19 void swap(int *i,int *j)
20 {
21     int temp;
22
23     temp=*i;
24     *i=*j;
25     *j=temp;
26 }
```

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 }
```

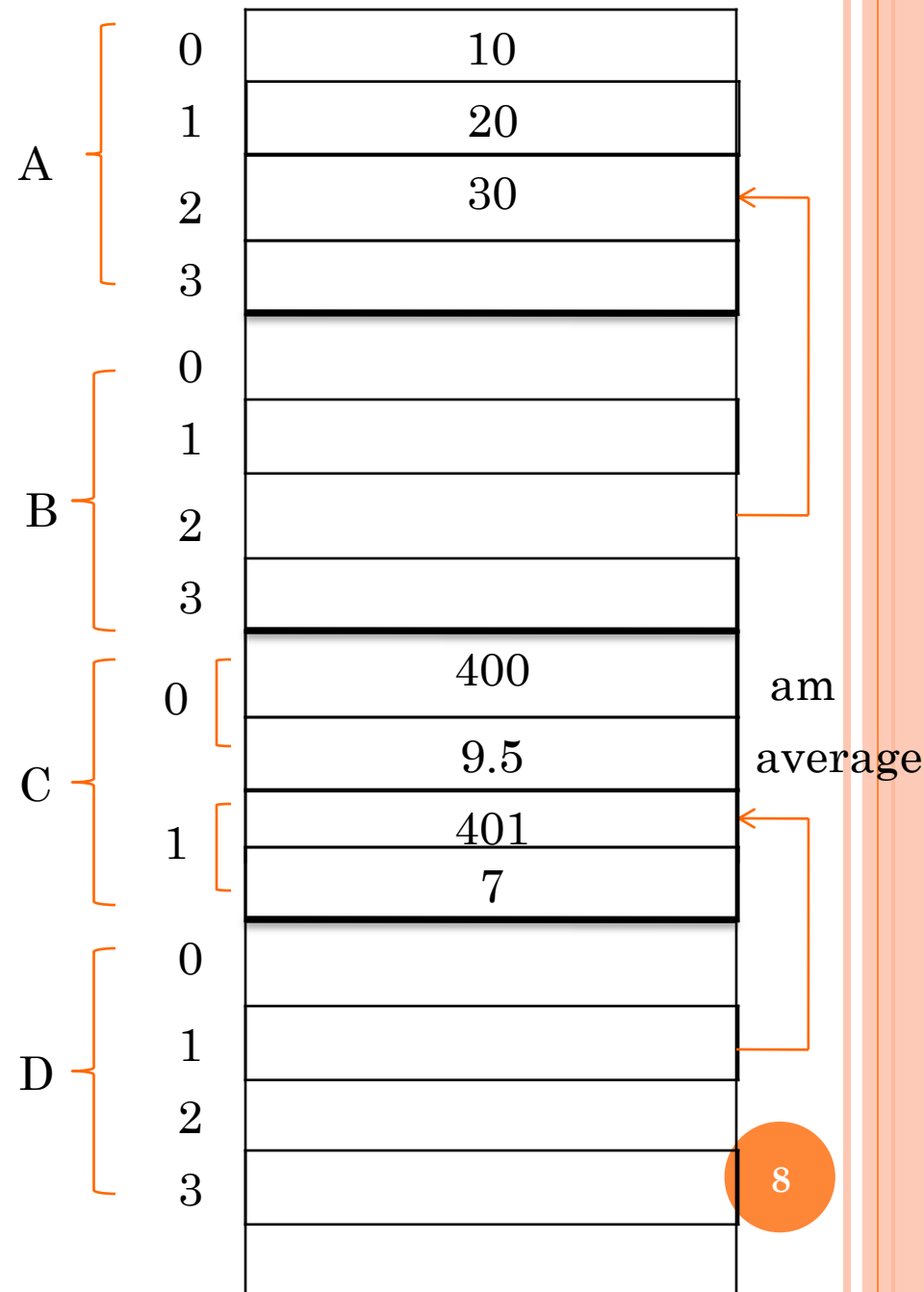
A diagram illustrating pointer assignment. A small box containing an equals sign (=) is positioned to the left of line 2, where the opening curly brace of the main function is located. A vertical line extends downwards from this box, ending at line 6, where the assignment 'px = &x;' is made. This visualizes the memory address of variable 'x' being stored in the pointer variable 'px'.

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

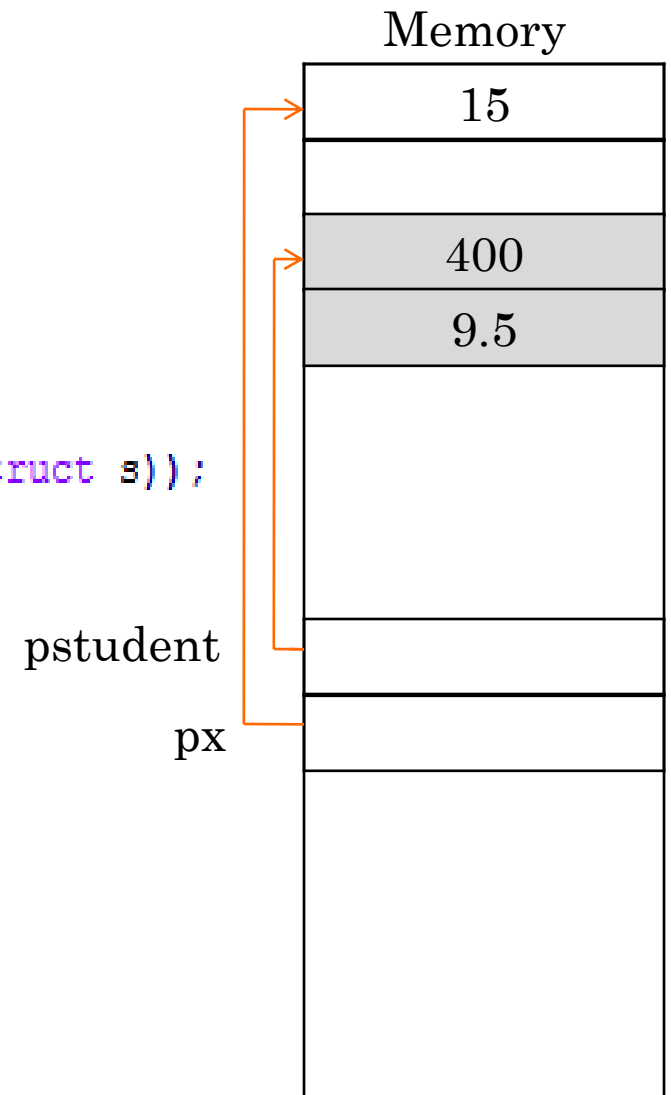
```
1 struct s {  
2     int am;  
3     float average;  
4 };  
5  
6 void main (void)  
7 {  
8     int A[4];  
9     int *B[4];  
10    struct s C[2];  
11    struct s *D[4];  
12  
13    A[0] = 10;  
14    A[1] = 20;  
15    B[2] = &A[2];  
16    *B[2] = 30;  
17    C[0].am = 400;  
18    C[0].average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



- 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

```
1 struct s
2 {
3     int am;
4     float average;
5     struct s *next;
6 };
7
8 struct s *insert1(struct s *head, int am, float average)
9 {
10     struct s *newnode;
11
12     newnode = malloc(sizeof(struct s));
13     newnode->am = am;
14     newnode->average = average;
15     newnode->next = head;
16     head = newnode;
17     return head;
18 }
19
20 int main()
21 {
22     struct s *head=NULL;
23     int am1=1, am2=2, am3=3;
24     float average1=15, average2=12, average3=13;
25
26     head = insert1(head, am1, average1);
27     head = insert1(head, am2, average2);
28     head = insert1(head, am2, average2);
29 }
```

ΛΙΣΤΕΣ INSERT

```
1 struct s
2 {
3     int am;
4     float average;
5 };
6
7 void insert2(struct s **head, int am, float average)
8 {
9     struct s *newnode;
10
11     newnode = malloc(sizeof(struct s));
12     newnode->am = am;
13     newnode->average = average;
14     newnode->next = (*head);
15     (*head) = newnode;
16     return;
17 }
18
19 int main()
20 {
21     struct s *head;
22     struct s *tmp;
23     int am1=1, am2=2, am3=3;
24     float average1=15, average2=12, average3=13;
25
26     insert2(&head, am1, average1);
27     insert2(&head, am2, average2);
28     insert2(&head, am2, average2);
29
30     tmp = head;
31     while(tmp != NULL){
32         printf("%d %lf\n", tmp->am, tmp->average);
33         tmp=tmp->next;
34     }
35 }
```

struct s *head = NULL

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 $n = 5$

Fib(5)

if (5 < 2): false

n1 = **Fib(5-1)**

if (4 < 2): false

n1 = **Fib(4-1)**

if (3 < 2): false

n1 = **Fib(3-1)**

if (2 < 2): false

n1 = **Fib(2-1)**

if (1 < 2): true
return 1

n2 = **Fib(2-2)**

if (0 < 2): true
return 0

return 1+0

n2 = **Fib(3-2)**

if (1 < 2): true
return 1

return 1+1

n2 =
return 2+1

n2 =
return 3+2

Fib(4-2)

if (2 < 2): false

n1 = **Fib(2-1)**

if (1 < 2): true
return 1

n2 = **Fib(2-2)**

if (0 < 2): true
return 0

return 1+0

Fib(5-2)

if (3 < 2): false

n1 = **Fib(3-1)**

if (2 < 2): false

n1 = **Fib(2-1)**

if (1 < 2): true
return 1

n2 = **Fib(2-2)**

if (0 < 2): true
return 0

return 1+0

n2 = **Fib(3-2)**

if (1 < 2): true
return 1

return 1+1

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 }
```

