2ο Εργαστήριο Τεχνικών Προγραμματισμού

Στόχοι: Δείκτες, Αριθμητική Δεικτών, swap

Σας δίνονται μικρά προγράμματα και σας ζητείται να τα ολοκληρώσετε σύμφωνα με τα σχόλια που εμπεριέχουν.

Ακολουθήστε βήμα-βήμα τις οδηγίες (Χρονική Πρόβλεψη)

1. Συμπληρώστε το ακόλουθο πρόγραμμα C (10 λεπτά)

```
#include <stdio.h>

int main()
{
	int i = 4;
	double d = 3.14;
	char c = 'a';

	/* Ορίστε ένα δείκτη σε κάθε μία από τις τρεις μεταβλητές */

	/* Τυπώστε τις διευθύνσεις των δεικτών.  Χρησιμοποιείστε το
	   προσδιοριστικό "%p" για την εκτύπωση διευθύνσεων μνήμης
	   της μορφής '0xbe10a22f'. */

	/* Τυπώστε το μέγεθος της κατειλημμένης μνήμης για κάθε μία
	   από τις 6 μεταβλητές (μεταβλητές και δείκτες).
	   Χρησιμοποιείστε τον τελεστή sizeof. */

	/* Τυπώστε τις τιμές του περιεχομένου μνήμης */
	return 0;
}
```

2. Συμπληρώστε το ακόλουθο πρόγραμμα C (10 λεπτά)

```
#include <stdio.h>

int main()
{
	int i = 4;
	double d = 3.14;
	char c = 'a';

	/* Ορίστε ένα δείκτη σε κάθε μία από τις τρεις μεταβλητές. */
	/* Αλλάξτε τις τιμές περιεχομένων των δεικτών ως εξής:
		*iptr <- 6
		*dptr <- 5.0
		*cptr <- 'd'  */
	/* Τυπώστε τις τιμές των μεταβλητών i, d, c */
	return 0;
}
```

3. Βρείτε (τυπώστε) τη διεύθυνση της μεταβλητής x στη συνάρτηση foo και τη διεύθυνση της μεταβλητής y στη συνάρτηση bar(). Τι παρατηρείτε; (10 λεπτά)

```
#include <stdio.h>
void foo(int xval)
{
	int x;
```

```
	x = xval;
	/* Τυπώστε τη διεύθυνση και την τιμή της x εδώ. */
}

void bar(int yval)
{
	int y;
	/* Τυπώστε τη διεύθυνση και την τιμή της y εδώ. */
}

int main()
{
	foo(5);
	bar(3);
}
```

4. Το παρακάτω πρόγραμμα χρησιμοποιεί αριθμητική δεικτών (pointer arithmetic) για να εξάγει το μέγεθος μιας μεταβλητής τύπου char. Χρησιμοποιώντας αριθμητική δεικτών μπορούμε να εξάγουμε την τιμή της 'cp' και την τιμή της 'cp+1'. Αφού η cp είναι δείκτης, η πρόσθεση αυτή περιλαμβάνει αριθμητική δεικτών: Προσθέτοντας μία μονάδα σε ένα δείκτη κάνει τον δείκτη να δείχνει στο επόμενο στοιχείο του ίδιου τύπου. Για ένα δείκτη σε char, η πρόσθεση με μονάδα ουσιαστικά σημαίνει πρόσθεση μονάδας στη διεύθυνση, αλλά αυτό συμβαίνει επειδή ο τύπος char είναι μεγέθους 1 byte.

   i. Μεταγλωττίστε και εκτελέστε το πρόγραμμα. Παρατηρείστε την έξοδο του.
   ii. Γράψτε κώδικα που να κάνει την αντίστοιχη αριθμητική δεικτών με μεταβλητή τύπου int. Τι μέγεθος έχει ένας int;
   iii. Κάντε τις αντίστοιχες ενέργειες με μεταβλητή τύπου double.
   iv. Τι θα συμβεί αν αντί για 1 προσθέτουμε 2 στα ερωτήματα i, ii, iii;

(30 λεπτά)

```
#include <stdio.h>
int main( )
{
	char c = 'Z';
	char *cp = &c;
	printf("cp is %p\n", cp);
	printf("The character at cp is %c\n", *cp);

	cp = cp+1;
	printf("cp is %p\n", cp);
	/* Μην τυπώσετε το *cp τώρα γιατί δείχνει σε μνήμη
	   μη αρχικοποιημένη. */
}
```

5. Η `swap_nums` δουλεύει σωστά, ενώ η `swap_pointers` όχι. Διορθώστε τη `swap_pointers`.

(20 λεπτά)

```
#include <stdio.h>

void swap_nums(int *x, int *y)
{
	int tmp;
	tmp = *x;
	*x = *y;
	*y = tmp;
}

void swap_pointers(char *x, char *y)
```

```
{
	char *tmp;
	tmp = x;
	x = y;
	y = tmp;
}

int main()
{
	int a = 3;
	int b = 4;
	char *s1 = "I should print first";
	char *s2 = "I should print second";

	swap_nums(&a, &b);
	printf("a is %d\n", a);
	printf("b is %d\n", b);

	swap_pointers(s1, s2);
	printf("s1 is %s\n", s1);
	printf("s2 is %s\n", s2);

	return 0;
}
```

6. Υλοποιήστε και καλέστε τη συνάρτηση swap_struct(). (20 λεπτά)

```
#include <stdio.h>

typedef struct {
	int i;
	char *s;
} D;

void print_D(D d)
{
	printf("Struct contains int: %d, and string: %s\n", d.i, d.s);
}

void swap_struct(D* d1, D* d2);
/* Υλοποιήστε την swap_struct() */

int main()
{
	D d1, d2;
	d1.i = 5;
	d1.s = "Hello.";
	d2.i = 2;
	d2.s = "World.";

	print_D(d1);
	print_D(d2);

	/* Καλέστε την swap_struct() */

	print_D(d1);
	print_D(d2);
	return 0;
}
```