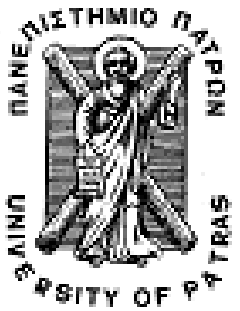
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ



ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΙΑΔΙΚΑΣΤΙΚΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2010-2011)

ΥΠΕΥΘΥΝΟΙ ΔΙΔΑΣΚΟΝΤΕΣ ΕΡΓΑΣΤΗΡΙΟΥ: Α. ΦΩΚΑ, Κ. ΣΤΑΜΟΣ

1° ΣΕΤ ΑΣΚΗΣΕΩΝ

(Ενδεικτικές Λύσεις)

Οι ασκήσεις αυτού του φυλλαδίου καλύπτουν τα παρακάτω θέματα και δίνονται ενδεικτικά οι αντίστοιχες ενότητες στο βιβλίο The GNU C Programming Tutorial που μπορείτε συμβουλευτείτε (http://crasseux.com/books/ctutorial/) .

- Τύποι δεδομένων, δήλωση μεταβλητών, αρχικοποίηση μεταβλητών, Μετατροπή τύπων (κεφάλαιο Variables and Declarations)
- Τελεστές, προτεραιότητα τελεστών, λογικοί τελεστές, Εκφράσεις (κεφάλαιο Expressions and Operators)
- printf () και scanf() (http://www.cs.utah.edu/~phister/K_n_R/appb.html)

Άσκηση 1η

Σημειώστε ποιες από τις παρακάτω εκφράσεις είναι σωστές και ποιες λάθος;

a. Η έκφραση x=(y=200) αναθέτει και στο x και στο y τον αριθμό 200.
b. Η έκφραση (2>3) && (x=1) τοποθετεί το 1 στην μεταβλητή x.
c. Η έκφραση (3>1) || (x=1) τοποθετεί το 1 στην μεταβλητή x.
d. Έστω ότι x= (2>3+4). Τότε η printf("%d ",x) θα εκτυπώσει 0.
e. Η ανάθεση int x='a'; είναι συντακτικά λάθος.
f. Η έκφραση 'A'+1 έχει σαν αποτέλεσμα 66.
g. Η ανάθεση x=x++; είναι συντακτικά ορθή.
h. Η ανάθεση x=(2++); είναι συντακτικά ορθή.

a. Σ
b. Λ
c. Λ
d. Σ
e. Λ
f. Σ
g. Σ
h. Λ

Άσκηση 2η

Να γραφεί πρόγραμμα που να εμφανίζει στην οθόνη το παρακάτω κείμενο (μαζί με το κουτάκι από αστερίσκους):

```
************************************************
* A regular expression to match emails is:     *
* "\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b" *
*                                              *
* Source: http:// regular-expressions.info     *
************************************************
```

Υπόδειξη: Ελέγξτε στη παρακάτω ιστοσελίδα πώς εμφανίζονται οι ειδικοί χαρακτήρες. http://crasseux.com/books/ctutorial/Special-characters.html

```
#include <stdio.h>
main()
{
 printf("*********************************************\n");
 printf("* A regular expression to match emails is:     *\n");
 printf("* \"\\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,4}\\b\" *\n");
 printf("*                                               *\n");
 printf("* Source: http://regular-expressions.info      *\n");
 printf("*********************************************\n");
}
```

Άσκηση 3η

Έχουμε το παρακάτω τμήμα κώδικα. Βρείτε τι θα εκτυπωθεί πριν τρέξετε τον κώδικα.

```
int z;
int x=2;
int y=4;
x=--y;
printf("%d %d\n", x,y);
z=-(x++)-(--y);
printf("%d %d %d\n", x,y,z);
y=(z--)+4;
printf("%d %d\n", y,z);
x= 12 % 5 / 1.5;
y= 101 + (int)(1.5);
z= 'd' - 3;
printf("%d %d %c\n", x,y,z);
```

```
3 3
4 2 -5
-1 -6
1 102 a
```

Άσκηση 4η

Λογικές Εκφράσεις: Στην λογική πρόταση A && B, αν το A είναι ψευδές τότε όλη η πρόταση είναι ψευδής ανεξάρτητα από την B. Αντίστοιχα, στην πρόταση A || B, αν το A είναι αληθές τότε η πρόταση είναι αληθής ανεξάρτητα από το B. Γράψτε το παρακάτω πρόγραμμα:

```
main(void)
{
      int i, j;
      int res;
      scanf("%d", &i);
      scanf("%d", &j);
      res = (++i == 3) || (--j == 4);
      printf("%d\n", i);
      printf("%d\n", j);
      printf("%d\n", res);
}
```

- Αν δώσουμε στα i και j αρχικά τις τιμές 4 και 4 παρατηρήστε την έξοδο του προγράμματος. Κάντε το ίδιο για τιμές 3 και 10. Εξηγήστε αν η συμπεριφορά του προγράμματος σας φαίνεται λογική ή παρατηρείτε κάτι παράξενο και που οφείλεται αυτό.

Για **i=4** και **j=4** η έξοδος είναι **5 3 0**. Η **res** έχει τιμή **0** (δηλ. **FALSE**) γιατί πρώτα αυξάνεται η τιμή της **i** και **j** και μετά γίνεται ο έλεγχος ισότητας **== 4**.

Για **i=3** και **j=10** η έξοδος είναι **4 9 0**.

- Αλλάξτε την εντολή res = (++i == 3) || (--j == 4); σε res = (i++ == 3) || (j-- == 4); Τι τιμές πρέπει να δώσετε για να έχει η μεταβλητή res τιμή 1; Βάλτε τις τιμές που δοκιμάσατε προηγουμένως (i=4, j=4 και i=3, j=10) και δείτε αν και πως αλλάζει η συμπεριφορά του προγράμματός σας.

Τώρα πρώτα γίνεται ο έλεγχος ισότητας και μετά αυξάνεται η τιμή των **i** και **j**. Συνεπώς, για να έχει η μεταβλητή **res** τιμή **1** πρέπει να δώσουμε για το **i** τιμή **3** ή για το **j** τιμή **4**.

Για τις προηγούμενες τιμές έχουμε έξοδο **5 3 1** και **4 10 1**. Η **res** έχει τιμή **1** γιατί πρώτα γίνεται ο έλεγχος ισότητας **i++ == 3** και μετά αυξάνεται η τιμή της **i**. Σε αυτή την περίπτωση η τιμή της **j** παραμένει **10** και δεν έχει μειωθεί κατά **1** όπως θα περιμέναμε. Αυτό γίνεται διότι στην εντολή **res = (i++ == 3) || (j-- == 4);** ο έλεγχος της πρώτης συνθήκης μας δίνει **TRUE** και συνεπώς η τιμή όλης της παράστασης θα είναι **TRUE** ανεξαρτήτως της τιμής της δεύτερης συνθήκης δεδομένου ότι συνδέονται με **OR**. Γι' αυτό το λόγο η **C** δεν προχωράει στον υπολογισμό της τιμής της δεύτερης συνθήκης, και έτσι δεν εκτελείται το **j--**.

- Αλλάξτε την εντολή res = (++i == 3) || (--j == 4); σε res = (i++ == 13) && (j-- == 4); Δώστε τιμές 13 και 4. Στη συνέχεια δώστε τις τιμές 1 και 4. Εξηγήστε τη συμπεριφορά του προγράμματος.

Για **i=13** και **j=4** η έξοδος είναι **14 3 1**. Η **res** έχει τιμή **1** (δηλ. **TRUE**) γιατί πρώτα γίνονται οι έλεγχοι ισότητας και μετά αυξάνονται οι τιμές.

Για **i=1** και **j=4** η έξοδος είναι **2 4 0**. Η τιμή της **j** δεν έχει μειωθεί διότι στον υπολογισμό της τιμής της **res** η πρώτη συνθήκη ήταν **FALSE** και δεδομένου ότι οι δύο συνθήκες συνδέονται με **AND** δεν γίνεται ο υπολογισμός της τιμής της δεύτερης συνθήκης, αφού η **res** θα είναι **FALSE** ανεξαρτήτως της τιμής της, και έτσι δεν εκτελείται το **j--**.

- Αλλάξτε την εντολή res = (++i == 3) || (--j == 4); σε res = (--i == 3) && (--j == 4); Τι τιμές πρέπει να δώσετε για να έχει η μεταβλητή res τιμή 1; Βάλτε τις τιμές που δοκιμάσατε προηγουμένως (i=13, j=4 και i=1, j=4) και δείτε αν και πως αλλάζει η συμπεριφορά του προγράμματός σας.

Τώρα πρώτα μειώνεται η τιμή των **i** και **j** και μετά γίνεται ο έλεγχος ισότητας. Συνεπώς, για να έχει η μεταβλητή **res** τιμή **1** πρέπει να δώσουμε για το **i** τιμή **4** και για το **j** τιμή **5**.

Για τις προηγούμενες τιμές έχουμε έξοδο **12 4 0** και **0 4 0**. Εδώ βλέπουμε ότι η **j** στην πρώτη περίπτωση δεν έχει γίνει **3** διότι αντίθετα με την προηγούμενη περίπτωση δεν χρειάστηκε να υπολογιστεί η τιμή και των δύο συνθηκών και έτσι δεν εκτελέστηκε η **--j**.

Άσκηση 5η

Να γραφεί πρόγραμμα σε C το οποίο να διαβάζει ένα ακέραιο από 1 μέχρι 1000 και να τον αναλύει σε χαρτονομίσματα ή κέρματα των 50, 20, 5, 2, 1 ευρώ. Πχ.

Dose ena poso: 487
To poso 487 analyetai ws exis:
9 x 50 Euros
1 x 20 Euros
1 x 10 Euros
1 x 5 Euros
1 x 2 Euros
0 x 1 Euro

```
#include <stdio.h>
main()
{
 int poso, euro50, euro20, euro10, euro5, euro2, euro1, ypoloipo;

 printf("Dose ena poso: ");
 scanf("%d", &poso);

 euro50 = poso / 50;
 ypoloipo = poso % 50;
 euro20 = ypoloipo / 20;
 ypoloipo = ypoloipo % 20;
 euro10 = ypoloipo / 10;
 ypoloipo = ypoloipo % 10;
 euro5 = ypoloipo / 5;
 ypoloipo = ypoloipo % 5;
 euro2 = ypoloipo / 2;
 euro1 = ypoloipo % 2;

 printf("To poso %d analyetai ws exis:\n", poso);
 printf("%d x 50 Euros\n", euro50);
 printf("%d x 20 Euros\n", euro20);
 printf("%d x 10 Euros\n", euro10);
 printf("%d x 5 Euros\n", euro5);
 printf("%d x 2 Euros\n", euro2);
 printf("%d x 1 Euro\n", euro1);
}
```

Άσκηση 6η

Γράψτε ένα πρόγραμμα το οποίο να διαβάζει από το πληκτρολόγιο ένα αριθμό δευτερολέπτων (αποθηκεύστε τον σε μια μεταβλητή τύπου int). Στην συνέχεια θα υπολογίζει σε πόσες ώρες, λεπτά και δευτερόλεπτα αντιστοιχούν και θα εκτυπώνει στην οθόνη τα αποτελέσματα. Π.χ.

Dose deyterolepta: 5000
Ta 5000 deyterolepta antistoixoyn se:
1 hours, 23 minutes, 20 seconds

Τρέξτε το πρόγραμμα σας για τις τιμές 10000, 40000, 2000000000, 4000000000 και καταγράψτε τα αποτελέσματα. Τι παρατηρείτε;

```
#include <stdio.h>

int main()
{
 int total_seconds,hours,minutes,seconds,ypoloipo;

 printf("Dose deyterolepta: ");
 scanf("%d", &total_seconds);

 hours = total_seconds / 3600;
 ypoloipo = total_seconds % 3600;
 minutes = ypoloipo / 60;
 seconds = ypoloipo % 60;

 printf("Ta %d deyterolepta antistoixoyn se:\n", total_seconds);
 printf("%d hours, %d minutes, %d seconds\n", hours, minutes,
seconds);
}

Dose deyterolepta: 10000
Ta 10000 deyterolepta antistoixoyn se:
2 hours, 46 minutes, 40 seconds

Dose deyterolepta: 40000
Ta 40000 deyterolepta antistoixoyn se:
11 hours, 6 minutes, 40 seconds

Dose deyterolepta: 2000000000
Ta 2000000000 deyterolepta antistoixoyn se:
555555 hours, 33 minutes, 20 seconds

Dose deyterolepta: 4000000000
Ta -294967296 deyterolepta antistoixoyn se:
-81935 hours, -21 minutes, -36 seconds
```