ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΞΑΣΦΑΛΙΣΗ ΠΟΙΟΤΗΤΑΣ ΚΑΙ ΠΡΟΤΥΠΑ

ΕΡΓΑΣΙΑ 3Η

1 ΕΡΩΤΗΜΑ

Οι τρεις ρουτίνες στις οποίες αναφέρεται η εκφώνηση αφορούν στην `sort_numbers_ascending()`, στην `main()` της Α' υλοποίησης και στην `main()` της Β' υλοποίησης. Επομένως δεν θα χρειαστεί να συμπεριλάβουμε το `#include <stdio.h>` στην αναγραφή των τελεστών/εντέλων, αν και είναι προφανές ότι τελεστής θα ήταν το `#include < >` και έντελο το `stdio.h`.

- Αυτός είναι ο κώδικας της `sort_numbers_ascending()` χωρισμένος σε τελεστές και έντελα:

```
1  /* Fuction for getting sorting number in ascending order*/
2  void sort_numbers_ascending(int number[], int count)
3  {
4      int temp, i, j, k;
5
6      for (j = 0; j < count; ++j)
7      {
8          for (k = j + 1; k < count; ++k)
9          {
10             if (number[j] > number[k])
11             {
12                 temp = number[j];
13                 number[j] = number[k];
14                 number[k] = temp;
15             }
16         }
17     }
18     printf("Numbers in ascending order:\n");
19     for (i = 0; i < count; ++i)
20     printf("%d\n", number[i]);
21 }
```

Τελεστές	Εμφανίσεις	Έντελα	Εμφανίσεις
void	1	count	4
sort_numbers_ascending(,)	1	temp	3
int	3	i	5
number[ ]	8	j	8
{}	4	k	7
,	4	0	2
;	6	1	1
for (; ;)	3	Numbers in ascending order\n	1
=	6	%d\n	1
+	2		
<	3		
++	3		
if ( )	1		
>	1		
printf( )	2		
"	4		
$n_1 = 16$	$N_1 = 52$	$n_2 = 9$	$N_2 = 32$

Η καταγραφή των τελεστών και των εντέλων γίνεται ανά γραμμή κώδικα, και δεν λαμβάνουμε υπόψιν μας τα σχόλια, δηλαδή τα σημεία του κώδικα που ξεκινούν με `/*` και καταλήγουν με `*/`.

Για την δήλωση της συνάρτησης, καταρχάς το void είναι προφανώς τελεστής. Σε παρόμοια λογική με την αντιμετώπιση των πινάκων, θα ορίσουμε ως τελεστή το όνομα της συνάρτησης μαζί με τις παρενθέσεις και το

κόμμα (`sort_numbers_ascending( , )`).[1] Στον τελεστή θα εντάξουμε και το κόμμα καθώς ο αριθμός των εσωτερικών μεταβλητών θα είναι πάντα δύο, άρα πάντα θα υπάρχει ένα κόμμα να τις χωρίζει. Όσον αφορά τις μεταβλητές της συνάρτησης, έντελο είναι το `count` και τελεστές τα `int` και το `number[]` ως πίνακας. Τα άγκιστρα `{}` που περιβάλλουν το πεδίο ορισμού της συνάρτησης ορίζονται και αυτά ζευγαράκι ξεχωριστά ως τελεστές[2]. Το περιεχόμενο των `[]` του `number` είναι έντελο.

Τελεστές ορίζονται επίσης τα `for ( ; ; )` (με τα ερωτηματικά τα οποία πάντα υπάρχουν και διαχωρίζουν τις συνθήκες[3]), το `if ()` και το `while ()` όπως επίσης και το `printf()`. Όσον αφορά το `printf()`, υπάρχουν δύο μορφές με τις οποίες χρησιμοποιείται στον κώδικα: είτε περιλαμβάνοντας μόνο ένα string, είτε περιλαμβάνοντας το string και μια μεταβλητή. Συνεπώς, οι επιπλέον τελεστές που χρησιμοποιούνται στο `printf()` όπως τα εισαγωγικά `""` ή το κόμμα `,` ορίζονται ξεχωριστά. Ως έντελο θα οριστεί το περιεχόμενο που περικλείεται από τα εισαγωγικά `""` και οι πιθανές μεταβλητές που μπορεί να περιλαμβάνονται.

Παρόμοια είναι και η λογική της προσθήκης των `;` στον ορισμό του τελεστή `for` – είναι τα μόνα σταθερά στοιχεία της `for`.

Σημειώνουμε ότι έχουν διαχωριστεί οι τελεστές `+` και `++` διότι εκτελούν διαφορετικές λειτουργίες. Ως έντελα πέρα από τις μεταβλητές έχει οριστεί και το οποιοδήποτε constant υπάρχει στον κώδικα (`20`, `0`).

- Αυτός είναι ο κώδικας της `main()` χωρισμένος σε τελεστές και έντελα:

```
1  void main()
2  {
3        int i, count, number[20], t=0;
4
5        printf("How many numbers you are going to enter:");
6        scanf("%d", &count);
7        printf("\nEnter the numbers one by one:");
8
9     while (t>20)
10    {
11     printf("\nThis is a test");
12     scanf("%d", &count);
13    }
14    for (i = 0; i < count; ++i)
15       scanf("%d", &number[i]);
16 /* Calling the Function*/
17       sort_numbers_ascending(number, count);
18 }
```

Τελεστές	Εμφανίσεις	Έντελα	Εμφανίσεις
void	1	i	5
main()	1	count	5
{ }	2	20	2
int	1	t	2
,	6	0	2
;	8	How many numbers you are going to enter:	1

[1] Ορίζουμε τα void και το όνομα της συνάρτησης ξεχωριστά καθώς το void είναι ένα keyword που θα μπορούσε να χρησιμοποιηθεί και σε άλλο περιβάλλον· δεν ορίζει μόνο τις συναρτήσεις. Η συνάρτηση είναι μια ενέργεια, οπότε ταιριάζει να οριστεί σαν τελεστής ξεχωριστά καθώς τελεστής είναι οτιδήποτε ορίζει μια δράση πάνω στα έντελα. Μαζί με την συνάρτηση θα μπορούσαμε επίσης να ορίσουμε και τα άγκιστρα `{ }` που ακολουθούν, αλλά δεν το επιλέγουμε στην δεύτερη ρουτίνα `main()` εγκαλείται (γραμμή 17), όπου προφανώς δεν γίνεται να καλεστεί μαζί με τα άγκιστρα.

[2] Δεν υπάρχει λογική στην ξεχωριστή αναγραφή του αγκίστρου `{` και του αγκίστρου `}` ως τελεστές, καθώς σε έναν ορθά γραμμένο κώδικα πάντα και τα δύο άγκιστρα θα έχουν ίδιο αριθμό εμφανίσεων.

[3] Ως αποτέλεσμα, ο αριθμός των ξεχωριστών `;` που υπάρχουν θα είναι μικρότερος από τον συνολικό, αφού κάποια από αυτά θα βρίσκονται μέσα στη `for`.

number[ ]	2	%d\n	1
=	2	\nEnter the numbers one by one:	1
printf( )	3	\nThis is a test	1
"	12	%d	3
scanf( )	3	number	1
&	1		
while ( )	1		
>	1		
&	3		
for (; ;)	1		
<	1		
++	1		
sort_numbers_ascending(,)	1		
$n_1 = 19$	$N_1 = 51$	$n_2 = 11$	$N_2 = 24$

Χρησιμοποιούμε παρόμοια λογική για την συμπλήρωση του πίνακα όπως και στην πρώτη ρουτίνα. Σημειώνουμε ότι η `scanf()` ορίζεται και αυτή μόνη της ως τελεστής και διαχωρίζουμε τα υπόλοιπα στοιχεία που την αποτελούν (όπως το κόμμα , ή το ampersand `&`).

- Αυτός είναι ο κώδικας της `main()` της Β' υλοποίησης χωρισμένος σε τελεστές και έντελα:

```
1  void main()
2  {
3     int i, num[20], t=0;
4     int n, count, j, a, x, b;
5
6     printf("How many numbers you are going to enter:");
7     scanf("%d", &count);
8     printf("\nEnter the numbers one by one:");
..[4] /* * * * Test this code * * * */
18 while (t>20)
19 {
20 /*test*/
21    printf("\nThis is a test");
22    scanf("%d", &count);
23    printf("\nThis is my test");
24    scanf("%d", &count);
25 }
26
27 for(t=20; t<20; t--)
28 {
29 scanf("%d", &count);
30 }
31 /*My loop begins*/
32 for (i = 0; i < count; ++i)
33 scanf("%d", &num[i]);
34
35 for (i = 0; i < n; ++i){
36 for (j = i + 1; j < n; ++j){
37    if (num[i] > num[j]){
38       a = num[i];
39       num[i] = num[j];
40    num[j] = a;
41    }
42  }
43 }
44 /* Here are the data */
45 printf("Numbers in ascending order:\n");
46   for (i = 0; i < count; ++i)
47      printf("%d\n", num[i]);
48 }
```

[4] Λόγω εξοικονόμησης χώρου έχουν συμπτυχτεί οι γραμμές των σχολίων. Ο αριθμός των γραμμών όμως διατηρείται σωστός με βάση τον αρχικό κώδικα της εκφώνησης, κάτι που θα χρησιμεύσει στο ερώτημα 2δ.

Τελεστές	Εμφανίσεις	Έντελα	Εμφανίσεις
void	1	i	16
main( )	1	20	3
{ }	6	t	5
int	2	0	4
num[ ]	9	n	3
,	13	count	7
;	16	j	7
=	9	a	3
printf( )	6	x	1
"	22	b	1
scanf( )	5	How many numbers you are going to enter:	1
&	5	%d	5
while ( )	1	\nEnter the numbers one by one:	1
>	1	\nThis is a test	1
for (; ;)	5	\nThis is my test	1
<	5	1	1
--	1	Numbers in ascending order\n	1
++	4	%d\n	1
+	1		
if ( )	1		
$n_1 = 20$	$N_1 = 114$	$n_2 = 18$	$N_2 = 62$

2α ΕΡΩΤΗΜΑ

Για την πρώτη ρουτίνα:

Ισχύουν:[5]

$$N_{est(1)} = n_1 \log_2 n_1 + n_2 \log_2 n_2 = 16 \log_2 16 + 9 \log_2 9 = 92.53$$

$$N_{(1)} = N_1 + N_2 = 52 + 32 = 84$$

Επομένως:

$$\frac{N_{est}}{N}_{(1)} = \frac{92.53}{84} = 1.1015$$

Για την δεύτερη ρουτίνα:

$$N_{est(2)} = n_1 \log_2 n_1 + n_2 \log_2 n_2 = 19 \log_2 19 + 11 \log_2 11 = 118.76$$

$$N_{(2)} = N_1 + N_2 = 51 + 24 = 75$$

$$\frac{N_{est}}{N}_{(2)} = \frac{118.76}{75} = 1.5835$$

Για την τρίτη ρουτίνα:

$$N_{est(3)} = n_1 \log_2 n_1 + n_2 \log_2 n_2 = 20 \log_2 20 + 18 \log_2 18 = 161.50$$

[5] Οι δείκτες με τον αριθμό σε παρένθεση υποδηλώνουν την ρουτίνα στην οποία αναφερόμαστε. Πχ στο $N_{est(2)}$ αναφερόμαστε στον εκτιμητή μήκους της **δεύτερης** ρουτίνας.

$$N_{(3)} = N_1 + N_2 = 114 + 62 = 176$$

$$\frac{N_{est}}{N}_{(3)} = \frac{161.50}{176} = 0.9176$$

Συνολικά:

	sort_numbers_ascending()	main() (A)	main() (B)
$\frac{N_{est}}{N}$	1.1015	1.5835	0.9176

2β ΕΡΩΤΗΜΑ

Υπολογίζουμε την εκτίμηση του επιπέδου προγράμματος.

Για την πρώτη ρουτίνα:

$$L_{est(1)} = \frac{2n_2}{n_1 N_2} = \frac{2 \cdot 9}{16 \cdot 32} = \frac{9}{256} = 0.035$$

Για την δεύτερη ρουτίνα:

$$L_{est(2)} = \frac{2n_2}{n_1 N_2} = \frac{2 \cdot 11}{19 \cdot 24} = \frac{11}{228} = 0.048$$

Για την τρίτη ρουτίνα:

$$L_{est(3)} = \frac{2n_2}{n_1 N_2} = \frac{2 \cdot 18}{20 \cdot 62} = \frac{9}{310} = 0.029$$

Συνολικά:

	sort_numbers_ascending()	main() (A)	main() (B)
L_{est}	0.035	0.048	0.029

2γ ΕΡΩΤΗΜΑ

Οι όγκοι των ρουτίνων είναι οι εξής:

$$V_{(1)} = N_{(1)} \log_2 n_{(1)} = 84 \log_2 25 = 390.10$$

$$V_{(2)} = N_{(2)} \log_2 n_{(2)} = 75 \log_2 30 = 368.025$$

$$V_{(3)} = N_{(3)} \log_2 n_{(3)} = 176 \log_2 38 = 923.65$$

Για το επίπεδο γλώσσας ισχύει:

$$\lambda_{(1)} = {L_{est(1)}}^2 V_{(1)} = 0.035^2 \cdot 390.10 = 0.478$$

$$\lambda_{(2)} = {L_{\mathrm{est}(2)}}^2 V_{(2)} = 0.048^2 \cdot 368.025 = 0.848$$

$$\lambda_{(3)} = {L_{est(3)}}^2 V_{(3)} = 0.029^2 \cdot 923.65 = 0.777$$

	sort_numbers_ascending()	main() (A)	main() (B)
λ	0.478	0.848	0.777

2δ ΕΡΩΤΗΜΑ

Λαμβάνουμε υπόψη μόνο τα σχόλια τα οποία αφορούν τις ρουτίνες. Επομένως θα αγνοήσουμε τα αρχικά σχόλια της Α' υλοποίησης που περιγράφουν την λειτουργία του προγράμματος, αφού δεν περιγράφουν κάποια ρουτίνα συγκεκριμένα. Αυτά είναι τα σχόλια που αφορούν τις ρουτίνες:

Για την πρώτη ρουτίνα:

```
1  /* Fuction for getting sorting number in ascending order*/
```

$$Lines\ of\ Comments_{(1)} = 1$$

Για την δεύτερη ρουτίνα:

```
1  /* Calling the Function*/
```

$$Lines\ of\ Comments_{(2)} = 1$$

Παρόμοια στην δεύτερη υλοποίηση δεν λαμβάνουμε υπόψη το αρχικό σχόλιο `My code` αφού δεν προσδιορίζει συγκεκριμένα την `main()`.

Για την τρίτη ρουτίνα:

```
1  /*
2  *
3  *
4  *
5  Test this code
6  *
7  *
8  *
9  */
10 /*test*/
11 /*My loop begins*/
12 /*Here are the data*/
```

$$Lines\ of\ Comments_{(3)} = 12$$

Για τον υπολογισμό των φυσικών γραμμών κώδικα υπολογίζουμε τον συνολικό αριθμό των γραμμών μαζί με τα σχόλια και τις κενές γραμμές, όπως εμφανίζονται στην εκφώνηση.

Η ακριβής καταμέτρηση των γραμμών του κομματιού του κώδικα που μας ενδιαφέρει σε κάθε ρουτίνα βρίσκεται ήδη στο ερώτημα 1. Συγκεντρωτικά:

$$PLoC_{(1)} = 21 \qquad PLoC_{(2)} = 18 \qquad PLoC_{(3)} = 48$$

Επομένως:

$$\frac{Lines\ Of\ Comments}{PLoC}_{(1)} = \frac{1}{21} = 0.048$$

$$\frac{Lines\ Of\ Comments}{PLoC}_{(2)} = \frac{1}{18} = 0.056$$

$$\frac{Lines\ Of\ Comments}{PLoC}_{(3)} = \frac{12}{48} = 0.25$$

3.Σ1 ΕΡΩΤΗΜΑ

Υπολογίζουμε τους μέσους όρους:

$$\mathrm{n}_{1(1,2)} = \frac{\mathrm{n}_{1(1)} + \mathrm{n}_{1(2)}}{2} = \frac{16 + 19}{2} = 17.5$$

$$n_{2_{(1,2)}} = \frac{n_{2_{(1)}} + n_{2_{(2)}}}{2} = \frac{9 + 11}{2} = 10$$

$$N_{1_{(1,2)}} = \frac{N_{1_{(1)}} + N_{1_{(2)}}}{2} = \frac{52 + 51}{2} = 51.5$$

$$N_{2_{(1,2)}} = \frac{N_{2_{(1)}} + N_{2_{(2)}}}{2} = \frac{32 + 24}{2} = 28$$

$n_{1_{(1,2)}}$	17.5	$n_{2_{(1,2)}}$	10
$N_{1_{(1,2)}}$	51.5	$N_{2_{(1,2)}}$	28

3.Σ2 ΕΡΩΤΗΜΑ

Θεωρούμε ως βάρη του σταθμισμένου μέσου όρου τα $N_{(1)}$ και $N_{(2)}$. Επομένως:

$$\overline{n_{1}}_{(1,2)} = \frac{n_{1_{(1)}}N_{(1)} + n_{1_{(2)}}N_{(2)}}{N_{(1)} + N_{(2)}} = \frac{16 \cdot 84 + 19 \cdot 75}{84 + 75} = 17.42$$

$$\overline{n_{2}}_{(1,2)} = \frac{n_{2_{(1)}}N_{(1)} + n_{2_{(2)}}N_{(2)}}{N_{(1)} + N_{(2)}} = \frac{9 \cdot 84 + 11 \cdot 75}{84 + 75} = 9.94$$

$$\overline{N_{1}}_{(1,2)} = \frac{N_{1_{(1)}}N_{(1)} + N_{1_{(2)}}N_{(2)}}{N_{(1)} + N_{(2)}} = \frac{52 \cdot 84 + 51 \cdot 75}{84 + 75} = 51.53$$

$$\overline{N_{2}}_{(1,2)} = \frac{N_{2_{(1)}}N_{(1)} + N_{2_{(2)}}N_{(2)}}{N_{(1)} + N_{(2)}} = \frac{32 \cdot 84 + 24 \cdot 75}{84 + 75} = 28.23$$

Συνολικά:

	Μέσος όρος		Σταθμ. μέσος όρος	
Τελεστές	$n_{1_{(1,2)}}$	17.5	$\overline{n_{1}}_{(1,2)}$	17.42
	$N_{1_{(1,2)}}$	51.5	$\overline{N_{1}}_{(1,2)}$	51.53
Έντελα	$n_{2_{(1,2)}}$	10	$\overline{n_{2}}_{(1,2)}$	9.94
	$N_{2_{(1,2)}}$	28	$\overline{N_{2}}_{(1,2)}$	28.23

Δεδομένου ότι συγκρίνουμε ένα μικρό πλήθος από ρουτίνες (μόνο 2), οι οποίες μάλιστα έχουν παρεμφερή μήκη ($N_{(1)} - N_{(1)} = 84 - 75 = 9$) δεν υπάρχει κάποια μεγάλη διαφορά ανάμεσα στους δύο μέσους όρους.

Παρόλα αυτά σε περιπτώσεις όπου θα υπήρχε ένας αξιόλογος αριθμός από ρουτίνες οι οποίες μάλιστα έχουν ποικιλόμορφα μήκη, ο σταθμισμένος μέσος όρος θα ήταν ένας ακριβέστερος δείκτης των ολικών

μετρικών. Η ποικιλομορφία των μηκών έχει σημασία, καθώς αυτά είναι τα βάρη τα οποία πολλαπλασιάζουμε στον αριθμητή του σταθμισμένου μέσου όρου και είναι η ειδοποιός διαφορά σε σχέση με τον απλό μέσο όρο.

Μια ρουτίνα με μικρό μήκος θα πρέπει να ληφθεί υπόψη λιγότερο σε σύγκριση με μια ρουτίνα με μεγάλο μήκος, καθώς το μικρό της μήκος την κάνει πιο "ασήμαντη" συγκριτικά με την άλλη.Όμως στον υπολογισμό του απλού μέσου όρου, κάθε ρουτίνα λαμβάνεται υπόψη εξίσου (έχουν ίσα βάρη), άρα είναι πιθανό να οδηγούμαστε σε πιο ανακριβή αποτελέσματα.

Συνεπώς καταλληλότερο είναι το Σ2 σενάριο.

4 ΕΡΩΤΗΜΑ

Αυτός είναι ο συγκεντρωτικός πίνακας των μετρικών των δύο υλοποιήσεων. Οι υπόλοιπες μετρικές της Α' υλοποίησης είναι υπολογισμένες βάσει του Σ2 σεναρίου, δηλαδή με σταθμισμένο μέσο όρο.

	Α' υλοποίηση		**Β' υλοποίηση**	
Τελεστές	$\overline{n_{1}}_{(1,2)}$	17.42	$n_{1(3)}$	20
	$\overline{N_{1}}_{(1,2)}$	51.53	$N_{1(3)}$	114
Έντελα	$\overline{n_{2}}_{(1,2)}$	9.94	$n_{2(3)}$	18
	$\overline{N_{2}}_{(1,2)}$	28.23	$N_{2(3)}$	176
Λεξιλόγιο	$\bar{n}_{(1,2)}$	27.35	$n_{(3)}$	38
Μήκος	$\bar{N}_{(1,2)}$	79.75	$N_{(3)}$	176
$\frac{N_{est}}{N}$	$\frac{N_{est}}{N}_{(1,2)}$	1.329	$\frac{N_{est}}{N}_{(3)}$	0.918
Επίπεδο προγράμματος	$\overline{L_{est}}_{(1,2)}$	0.041	$L_{est(3)}$	0.029
Επίπεδο γλώσσας	$\bar{\lambda}_{(1,2)}$	0.653	$\lambda_{(3)}$	0.777

Συγκρίνοντας τις μετρικές διαπιστώνουμε ότι το σύνολο των τελεστών και των εντέλων, το **λεξιλόγιο** της Α' υλοποίησης αλλά και το συνολικό **μήκος** του προγράμματος είναι μικρότερα από της Β' υλοποίησης. Επομένως το "πλουσιότερο" λεξιλόγιο της Β' υλοποίησης και το μεγαλύτερο μήκος δείχνουν μια πιθανή μεγαλύτερη πολυπλοκότητα της υλοποίησης, καθώς περιλαμβάνονται παραπάνω τελεστές και έντελα.

Συμπέρασμα για την πολυπλοκότητα μπορούμε να εκλάβουμε και από το **επίπεδο λ** της γλώσσας. Στην Β' υλοποίηση είναι μεγαλύτερο από την Α, κάτι που οδηγεί στο συμπέρασμα ότι η Β' υλοποίηση είναι πιο «πνιγμένη» και με μεγαλύτερη δυσγραφία από την Α (όσο μικρότερο είναι το λ, τόσο πιο απλοϊκά και "στρωτά" είναι γραμμένος ο κώδικας).

Επίσης το **επίπεδο προγράμματος** είναι μικρότερο στην υλοποίηση Β, άρα η δυσκολία της Β' υλοποίησης είναι μεγαλύτερη (η δυσκολία D είναι αντιστρόφως ανάλογη του L).

Ο λόγος $\frac{N_{est}}{N}$ της Β' υλοποίησης πλησιάζει πιο κοντά στο 1 σε σχέση με την Α, κάτι που είναι το ιδανικό σενάριο για τον λόγο, δηλαδή την ταύτιση του εκτιμητή μεγέθους N_{est} και του μήκους N.Άρα ο εκτιμητής είναι πιο ακριβής στην Β' υλοποίηση σε σχέση με την Α. Παρόλα αυτά τα μεγέθη N των υλοποιήσεων είναι αρκετά μικρά σε σχέση με το N = 500 των στατιστικών μετρήσεων και συμπερασμάτων, επομένως ίσως δεν είναι τόσο βάσιμη η σύγκριση του $\frac{N_{est}}{N}$ με τόσο μικρές υλοποιήσεις.

Το συμπέρασμα μετά την σύγκριση των μετρικών των δύο υλοποιήσεων είναι πως η υλοποίηση Β συγκριτικά με την Α έχει **μεγαλύτερο μήκος και μεγαλύτερη πολυπλοκότητα** και στον τρόπο γραφής αλλά και στην αυξημένη χρήση διαφορετικών τελεστών και εντέλων .