Εξασφάλιση Ποιότητας και Πρότυπα
3η Εργασία

Ζητούμενο 1

Παρακάτω φαίνονται οι αναλυτικοί πίνακες για τις τρεις συνολικά ρουτίνες που διαθέτουν οι υλοποιήσεις Α και Β, στους οποίους παρουσιάζονται αναλυτικά οι διάφοροι τελεστές και τα έντελα, καθώς και το πλήθος των εμφανίσεών τους.

Παραδοχές:

1. Όσον αφορά στον πίνακα number[], τον αντιμετωπίζω ως μία “συνάρτηση” (αρά τελεστή), η οποία δέχεται ως είσοδο παραδείγματος χάριν μια μεταβλητή, k. Γι΄ αυτό ακόμα το λόγο δεν μετράω ως ξεχωριστό τελεστή τις αγκύλες “[]”.
2. Η sort_numbers_ascending αποτελεί μια ρουτίνα, η οποία εξ’ ορισμού (μιας και αναφερόμαστε στη γλώσσα C) απαιτεί τα άγκιστρα “{}” καθώς και τις παρενθέσεις “()”. Γι΄αυτό το λόγο δεν τα μετράω ως ξεχωριστούς τελεστές.
3. Οι ελάχιστες υλοποιήσεις που αφορούν στη δομή ελέγχου if και στη δομή επανάληψης for και while δεν απαιτούν τα άγκιστρα “{}”, συνεπώς μετράω τα τελευταία ως ξεχωριστό τελεστή. Παρομοίως και για τις υπόλοιπες ρουτίνες της άσκησης.
4. Δεν προσμετρώ την εισαγωγή του header “#include <stdio.h>”.
5. Η εντολή printf (στη C) εμφανίζεται υποχρεωτικά με ένα ζεύγος παρενθέσεων “()”, συνεπώς δεν το μετράω ως ξεχωριστό τελεστή. Ωστόσο, μιας και η ελάχιστη υλοποίηση της παραπάνω εντολής είναι χωρίς κόμμα “,”, μετράω το τελευταίο ως ξεχωριστό τελεστή.
6. Αντίστοιχα, η εντολή for (στη C) εμφανίζεται υποχρεωτικά με ένα ζεύγος παρενθέσεων και δύο ερωτηματικά, τα παραπάνω επομένως δεν προσμετρώνται ξεχωριστά ως τελεστές.
7. Η εντολή scanf (στη C) εμφανίζεται υποχρεωτικά (ως η ελάχιστη υλοποίησή της) με ένα ζεύγος παρενθέσεων καθώς και ένα κόμμα “,” συνεπώς τα παραπάνω δεν προσμετρώνται ξεχωριστά ως τελεστές. Σημαντικό να αναφερθεί είναι πως το σύμβολο “&” δεν συνοδεύει πάντοτε την εντολή scanf, αφού αυτό εξαρτάται από τον τύπο του input που θα δώσει ο χρήστης, επομένως θεωρείται ξεχωριστός τελεστής.
8. Η εντολή while (στη C) εμφανίζεται υποχρεωτικά με ένα ζεύγος παρενθέσεων “()”, συνεπώς το τελευταίο δεν προσμετράται ως ξεχωριστός τελεστής.
9. Η μεταβλητή number που δίνεται ως παράμετρος στη συνάρτηση sort_numbers_ascending στη main της Α υλοποίησης μετράται ως εμφάνιση του τελεστή number[].

Α' υλοποίηση (sort_numbers_ascending)

Τελεστές	Αριθμός Εμφανίσεων
void	1
sort_numbers_ascending() {}	1
int	3
number[]	8
for (; ;)	3
=	6
<	3
++	3
{}	3
if ()	1
;	6
+	1
printf()	2
,	5
>	1
n_1 = 15	**N_1 = 47**

Έντελα	Αριθμός Εμφανίσεων
count	4
temp	3
i	5
j	8
k	7
0	2
1	1
"Numbers in ascending order:\n"	1
"%d\n"	1
n_2 = 9	**N_2 = 32**

Α' υλοποίηση (main)

Τελεστές	Αριθμός Εμφανίσεων
void	1
main() {}	1
int	1
,	4
number[]	3

Έντελα	Αριθμός Εμφανίσεων
i	5
count	5
20	2
t	2
0	2

=	2
;	8
printf()	3
scanf(,)	3
while ()	1
{}	1
>	1
for (; ;)	1
<	1
++	1
sort_numbers_ascending()	1
&	3
n_1 = 17	**N_1 = 36**

"How many numbers you are going to enter:"	1
"%d"	3
"\nEnter the numbers one by one:"	1
"\nThis is a test"	1
n_2 = 9	**N_2 = 22**

Β' υλοποίηση (main)

Τελεστές	**Αριθμός Εμφανίσεων**
void	1
main() {}	1
int	2
num[]	9
,	8
=	9
;	16
printf()	6
scanf(,)	5
while ()	1

Έντελα	**Αριθμός Εμφανίσεων**
i	16
20	4
t	4
0	4
n	3
count	7
j	7
a	3
x	1
b	1

>	2
{}	5
for (; ;)	5
<	5
--	1
++	4
+	1
&	5
$n_1 = 18$	**$N_1 = 86$**

"How many numbers you are going to enter:"	1
"%d"	5
"\nEnter the numbers one by one:"	1
"\nThis is a test"	1
"\nThis is my test"	1
"Numbers in ascending order:\n"	1
"%d\n"	1
1	1
$n_2 = 18$	**$N_2 = 62$**

Ζητούμενο 2

Σε αυτό το μέρος για κάθε μία από τις παραπάνω ρουτίνες υπολογίζονται ορισμένες μετρικές, σύμφωνα με την εκφώνηση. Προηγουμένως ωστόσο θα αναφερθούν τα μεγέθη που απαιτούνται για τον υπολογισμό των ζητουμένων[1].

- Λόγος του εκτιμητή μήκους προς το μήκος προγράμματος (N_{est}/N): $\mathbf{N_{est} = n_1 * \log_2(n_1) + n_2 * \log_2(n_2)}$ και $\mathbf{N = N_1 + N_2}$
- Επίπεδο Προγράμματος (L): $\mathbf{L_{est} = 2 * n_2 / n_1 * N_2}$
- Επίπεδο Γλώσσας (λ):
    - Λεξιλόγιο: $n = n_1 + n_2$
    - Όγκος: $V = N * \log_2(n)$
    - Άρα, $\mathbf{\lambda = L^2 * V}$
- Λόγος αριθμού γραμμών σχολίων προς τον αριθμό φυσικών γραμμών κώδικα: **C = γραμμές σχολίων / φυσικός αριθμός**[2] κώδικα όπου φυσικός αριθμός κώδικα = # των γραμμών λαμβάνοντας υπόψιν σχόλια και κενές γραμμές

[1] Πραγματοποιείται στρογγυλοποίηση των δεκαδικών αριθμών έως και το τρίτο ψηφίο μετά την υποδιαστολή
[2] Οι γραμμές κώδικα κάθε ρουτίνας μετρώνται ξεκινώντας από την επικεφαλίδας της μέχρι το σύμβολο "}" που υποδεικνύει την ολοκλήρωσή της

Α' υλοποίηση (sort_numbers_ascending)

Υπενθυμίζεται ότι $\mathbf{n_1 = 15}$, $\mathbf{N_1 = 47}$, $\mathbf{n_2 = 9}$ και $\mathbf{N_2 = 32}$

1. N_{est}/N
    a. $N = 47 + 32 = 79$
    b. $N_{est} = 15 * \log_2(15) + 9 * \log_2(9) = 15 * 3.906 + 9 * 3.169 = 58.59 + 28.521 = 87.111$
    c. Άρα, $N_{est}/N = 87.111 / 79 = 1.102$
2. $L = L_{est} = 2 * 9 / 15 * 32 = 18 / 480 = 0.037$
3. λ
    a. $n = 15 + 9 = 24$
    b. $V = 79 * \log_2(24) = 79 * 4.584 = 362.136$
    c. Άρα, $\lambda = L^2 * V = 0.037^2 * 362.136 = 0.495$
4. $C = 1 / 21 = 0.047$[3]

Α' υλοποίηση (main)

Υπενθυμίζεται ότι $\mathbf{n_1 = 17}$, $\mathbf{N_1 = 36}$, $\mathbf{n_2 = 9}$ και $\mathbf{N_2 = 22}$

1. N_{est}/N
    a. $N = 36 + 22 = 58$
    b. $N_{est} = 17 * \log_2(17) + 9 * \log_2(9) = 17 * 4.087 + 9 * 3.17 = 69.479 + 28.53 = 98.009$
    c. Άρα, $N_{est}/N = 98.009 / 58 = 1.689$
2. $L = 2 * 9 / 17 * 22 = 18 / 374 = 0.048$
3. λ
    a. $n = 17 + 9 = 26$
    b. $V = 58 * \log_2(26) = 58 * 4.7 = 272.6$
    c. Άρα, $\lambda = 0.048^2 * 272.6 = 0.628$
4. $C = 5 / 24 = 1 / 3 = 0.208$[4]

Β' υλοποίηση (main)

Υπενθυμίζεται ότι $\mathbf{n_1 = 18}$, $\mathbf{N_1 = 86}$, $\mathbf{n_2 = 18}$ και $\mathbf{N_2 = 62}$

1. N_{est}/N
    a. $N = 86 + 62 = 148$
    b. $N_{est} = 18 * \log_2(18) + 18 * \log_2(18) = 18 * 4.17 + 18 * 4.17 = 2 * 75.06 = 150.12$
    c. Άρα, $N_{est}/N = 150.12 / 148 = 1.014$

[3] Όσον αφορά στην ρουτίνα sort_numbers_ascending λαμβάνεται υπόψιν το σχόλιο που βρίσκεται ακριβώς πάνω από το όνομά της αλλά όχι αυτό που βρίσκεται στην αρχή του προγράμματος, μιας και αυτό δεν αφορά την συγκεκριμένη ρουτίνα ξεκάθαρα

[4] Θεωρώ πως το σχόλιο που βρίσκεται στην αρχή του προγράμματος αφορά στην συνάρτηση main ενώ όλες υπόλοιπες γραμμές του προγράμματος μετρώνται ως γραμμές κώδικα (εκτός από αυτές που εντοπίζεται η ρουτίνα sort_numbers_ascending) συμπεριλαμβανομένου και αυτών που βρίσκονται πάνω από την main άρα και η γραμμή "#include <stdio.h>" ως κεντρική ρουτίνα του προγράμματος

2. L = 2 * 18 / 18 * 62 = 36 / 1116 = 0.032
3. λ
    a. n = 18 + 18 = 36
    b. V = 148 * $\log_2(36)$ = 148 * 5.17 = 765.16
    c. Άρα, λ = 0.032^2 * 765.16 = 0.783
4. C = 19 / 56 = 0.339[5]

Ζητούμενο 3

Για το Σ1:

- N_{est} / N → (1.102 + 1.689) / 2 = 2.791 / 2 = 1.395
- L → (0.037 + 0.048) / 2 = 0.085 / 2 = 0.049
- λ → (0.495 + 0.986) / 2 = 0.042
- C → (0.047 + 0.208) / 2 = 0.255 / 2 = 0.127

Για το Σ2:

- N_{est} / N → (1.102 * 79 + 1.689 * 58) / (79 + 58) = (87.058 + 97.962) / 137 = 185.02 / 137 = 1.35
- L → (0.037 * 79 + 0.048 * 58) / (79 + 58) = (2.923 + 2.784) / 137 = 5.707 / 137 = 0.041
- λ → (0.495 * 79 + 0.628 * 58) / (79 + 58) = (39.105 + 36.424) / 137 = 75.529 / 137 = 0.551
- C → (0.047 * 79 + 0.208 * 58) (79 + 58) = (3.713 + 12.064) / 137 = 15.777 / 137 = 0.115

Στην πρώτη περίπτωση, στην οποία υπολογίζεται ο μέσος όρος, όλοι οι όροι επηρεάζουν το τελικό αποτέλεσμα με τον ίδιο τρόπο (ομοιόμορφα), αφού ουσιαστικά το βάρος του καθενός είναι 1. Ωστόσο, στη δεύτερη περίπτωση, στην οποία υπολογίζεται ο σταθμισμένος μέσος όρος, γίνεται χρήση ενός κριτηρίου στάθμισης, συνεπώς κάθε όρος επηρεάζει το τελικό αποτέλεσμα, ανάλογα με το βάρος του (ανομοιόμορφη επιρροή των όρων στο τελικό αποτέλεσμα).
Πιο συγκεκριμένα, έστω ότι έχει υλοποιηθεί πρόγραμμα, το οποίο διαθέτει κάποιες μικρές ρουτίνες (και άρα αντίστοιχα μικρή πιθανότητα εμφάνισης σφάλματος) καθώς και μία μεγάλη και πολύπλοκη ρουτίνα (και άρα αντίστοιχα μεγάλη πιθανότητα εμφάνισης σφάλματος). Στην περίπτωση επιλογής χρήσης του απλού μέσου όρου για τον υπολογισμό μίας συνολικής μετρικής, καθίσταται αρκετά πιθανό το αποτέλεσμα που θα προκύψει, να οδηγήσει σε λανθασμένα συμπεράσματα, μιας και η πολύπλοκη ρουτίνα θα επηρεάσει το αποτέλεσμα ισοβαρώς με τις υπόλοιπες. Αντίθετα, η επιλογή χρήσης του σταθμικού μέσου όρου για τον υπολογισμό μίας συνολικής μετρικής, λόγω της ύπαρξης του κριτηρίου στάθμισης, θα οδηγήσει σε ένα πιο ορθό αποτέλεσμα, αφού η “επίδραση” της πολύπλοκης ρουτίνας στο αποτέλεσμα θα είναι πιο αντιπροσωπευτική, εξαιτίας του βάρους της.

[5] Έχει ληφθεί υπόψιν η γραμμή του #include <stdio.h> ενώ από τη στιγμή που το πρόγραμμα αποτελείται αποκλειστικά από τη main μετρώνται όλες οι γραμμές από την αρχή του προγράμματος

Λαμβάνοντας υπόψιν τα παραπάνω ο σταθμισμένος μέσος όρος αποτελεί πιο ακριβή και αντικειμενική μέθοδος - ανάλογα βέβαια και με τη φύση των μετρικών που απαιτείται να υπολογιστούν - σε σύγκριση με τον υπολογισμό του μέσου όρου.

Ζητούμενο 4

Συνοπτικά, για τις παραπάνω δύο υλοποιήσεις προέκυψαν οι παρακάτω μετρικές.

Μετρικές	**Υλοποίηση Β**	**Υλοποίηση Α**
$\mathbf{N_{est}}$ **/ N**	1.014	1.35
$\mathbf{L_{est}}$	0.032	0.041
λ	0.783	0.551
C	0.339	0.115

- Όσον αφορά στο ποσοστό σχολίων, φαίνεται πως η υλοποίηση Β διαθέτει περισσότερα σχόλια, πράγμα που σημαίνει πως είναι πιθανόν (εξαρτάται βέβαια και από την ποιότητα των σχολίων) πιο εύκολα κατανοήσιμη ως προς έναν τρίτο που μελετά τον κώδικα για πρώτη φορά
- Το μήκος του προγράμματος καθώς και η πολυπλοκότητα του κώδικα είναι μεγαλύτερα στην υλοποίηση Α, το οποίο βέβαια έρχεται σε αντίθεση με την παραπάνω πρόταση, μιας και κάποιος θα περίμενε πως μεγαλύτερο μήκος προγράμματος και πολυπλοκότητα κώδικα συνεπάγεται αυτόματα και μεγαλύτερο ποσοστό σχολίων
- Παρατηρείται πως η υλοποίηση Α χαρακτηρίζεται από χαμηλότερο επίπεδο γλώσσας σε σύγκριση με την υλοποίηση Β με αποτέλεσμα να συνεπάγεται δυσκολία στη κατανόησή της. Το παραπάνω εντείνεται ακόμη περισσότερο, αν ληφθούν υπόψιν και οι παραπάνω δύο προτάσεις