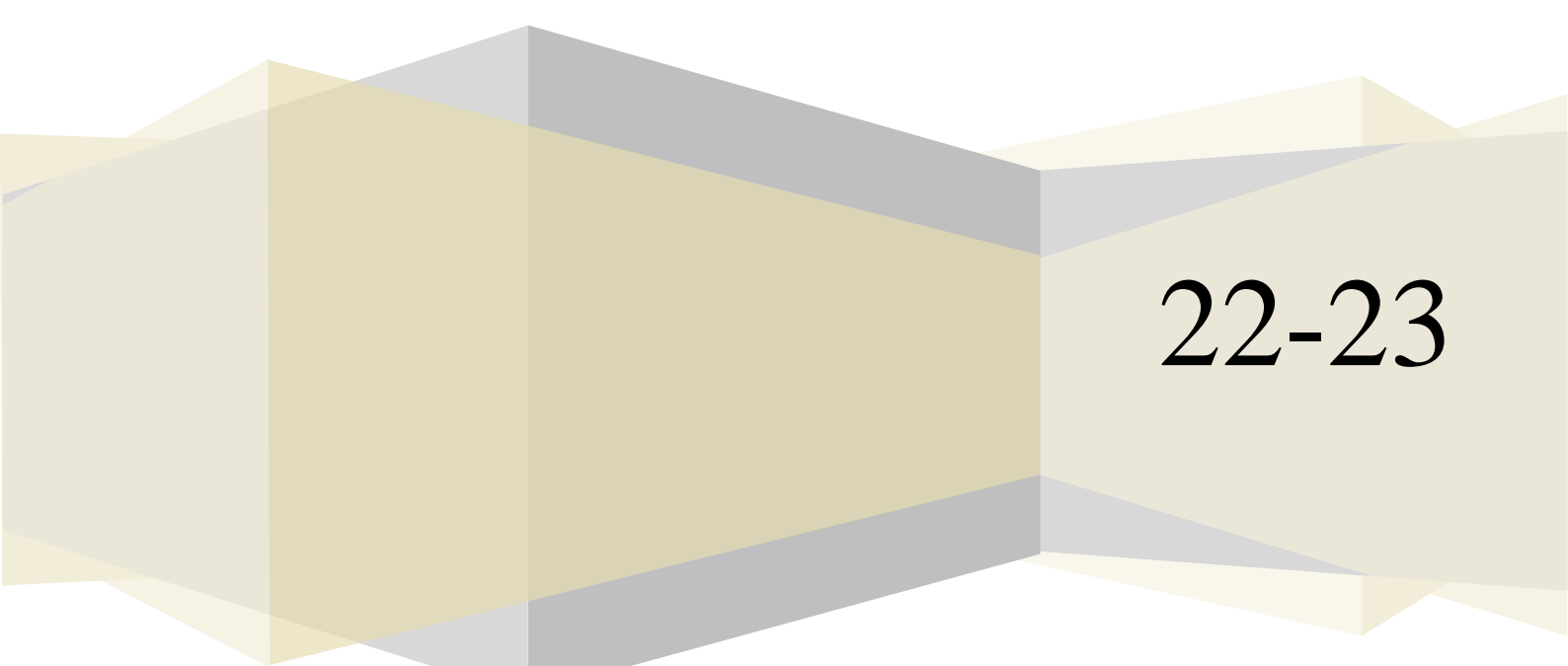


Εξασφάλιση Ποιότητας και Πρότυπα

Άσκηση 4



22-23

Ζητούμενο 1:

Αρχικοποιούμε τις μεταβλητές a , $b = 0$ και $c = 10$.

Το πρόγραμμα αρχικά ζητάει από τον χρήστη να εισάγει έναν ακέραιο αριθμό (a).

Αν ο χρήστης εισάγει αρνητικό ακέραιο ή τον αριθμό 0 τότε το σύστημα του ζητάει να εισάγει έναν άλλον αριθμό, επομένως κατανοούμε ότι μία από τις προδιαγραφές του προγράμματος είναι η είσοδος του χρήστη να είναι ένας θετικός ακέραιος.

Αφότου ο χρήστης εισάγει τον θετικό ακέραιο συμβαίνουν οι εξής έλεγχοι:

- Όσο το a είναι διάφορο του b αυξάνουμε το b κατά 1 και αν το c είναι μεγαλύτερο του a αυξάνουμε το a κατά 1, αλλιώς αυξάνουμε το c κατά 1
- Στη συνέχεια αν το a είναι ίσο με το b και το c διάφορο του b τότε αυξάνουμε το c κατά 1, αλλιώς θέτουμε το c ίσο με το b
- Αν το c είναι μεγαλύτερο από 50 τότε θέτουμε το c ίσο με το άθροισμα του c με το a

Τέλος, στην οθόνη εμφανίζεται το c .

Ζητούμενο 2:

Ο editor που χρησιμοποιήθηκε για τη δοκιμή του κώδικα είναι ο DEV C++. Κατά την εκτέλεση του προγράμματος παρουσιάστηκε ένα ζήτημα, καθώς όταν εισαγόταν ο ακέραιος που ζητείται από τον χρήστη και πληκτρολογούνταν το enter το πρόγραμμα τερματιζόταν χωρίς να εμφανίσει πρώτα τα αποτελέσματα. Το θέμα αυτό επιλύθηκε εισάγοντας δύο εντολές `getchar()` μία μετά τη `scanf` και μία πριν τη `return`, ώστε να αναμένεται το enter σαν πλήκτρο για τη διαδικασία καταχώρησης του a και το δεύτερο enter για τον τερματισμό του προγράμματος. Οπότε ο νέος κώδικας φαίνεται παρακάτω:

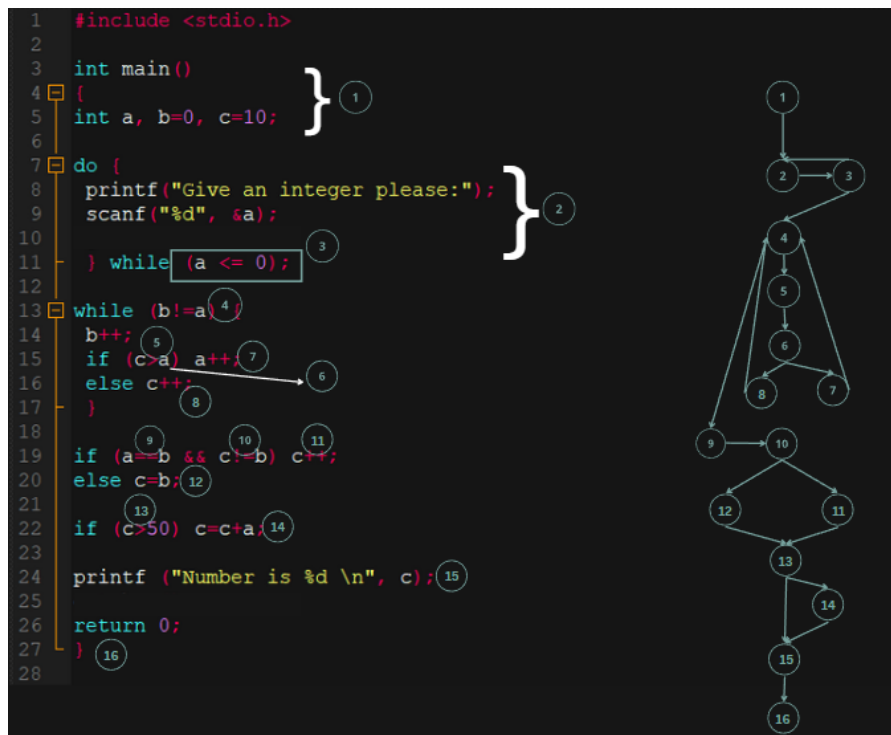
```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a, b=0, c=10;
6
7      do {
8          printf("Give an integer please:");
9          scanf("%d", &a);
10         getchar();
11     } while (a <= 0);
12
13     while (b!=a) {
14         b++;
15         if (c>a) a++;
16         else c++;
17     }
18
19     if (a==b && c!=b) c++;
20     else c=b;
21
22     if (c>50) c=c+a;
23
24     printf ("Number is %d \n", c);
25     getchar();
26     return 0;
27 }
```

Τρέχουμε τον κώδικα με τις παρακάτω τιμές και λαμβάνουμε τα εξής αποτελέσματα:

| Τιμές Εισόδου (a) | Αποτελέσματα (c) |
|-------------------|------------------------|
| 5 | 16 |
| 11 | 22 |
| 95 | 210 |
| 0 | Give an integer please |
| -1 | Give an integer please |
| 1 | 12 |
| 58 | 136 |
| 20 | 31 |
| 79 | 178 |
| 56 | 132 |

Ζητούμενο 3:

Για το γράφο δε λαμβάνεται υπόψη η αλλαγή με τα getchar, δουλεύουμε πάνω στον αρχικό κώδικα που δίνεται στην εκφώνηση. Παρακάτω φαίνεται ο γράφος του κώδικα και η διαδικασία που ακολουθήθηκε για την εύρεση των κόμβων:



Κυκλωματική Πολυπλοκότητα:

Α' τρόπος:

$$V(g) = e - n + 2p$$

$$V(g) = 20 - 16 + 2 \cdot 1 = 6$$

Β' τρόπος:

$$V(g) = 1 + \text{εντολές}$$

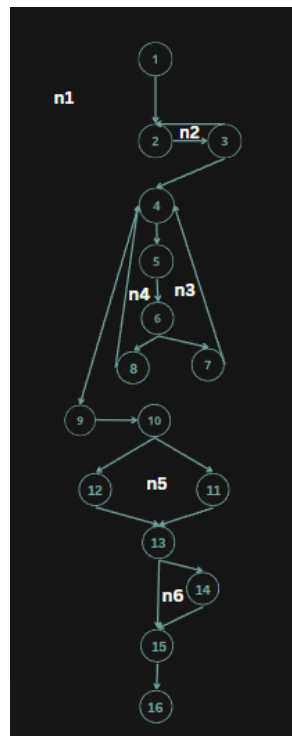
```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a, b=0, c=10;
6
7      do {
8          printf("Give an integer please:");
9          scanf("%d", &a);
10     } while (a <= 0);
11
12     while (b!=a) {
13         b++;
14         if (c>a) a++;
15         else c++;
16     }
17
18     if (a==b && c!=b) c++;
19     else c=b;
20
21     if (c>50) c=c+a;
22
23     printf ("Number is %d \n", c);
24
25     return 0;
26 }
27
```

Diagram illustrating the counting of instructions (εντολές) in the code snippet:

- 1: `do {` (1)
- 2: `printf("Give an integer please:");`
- 3: `scanf("%d", &a);`
- 4: `} while (a <= 0);`
- 5: `while (b!=a) {` (2)
- 6: `b++;`
- 7: `if (c>a) a++;`
- 8: `else c++;`
- 9: `}`
- 10: `if (a==b && c!=b) c++;` (3)
- 11: `else c=b;`
- 12: `if (c>50) c=c+a;` (4)
- 13: `printf ("Number is %d \n", c);` (5)

$$V(g) = 1 + 5 = 6$$

Γ' τρόπος:



$$V(g) = 6$$

Ζητούμενο 4:

M1: 1, 2, 3, 4, 5, 6, (8, 4)₁₁, 9, 10, 11, 13, 15, 16

M2: 1, 2, 3, 4, 5, 6, (8, 4)₅₁, 9, 10, 11, 13, 14, 15, 16

M3: 1, 2, 3, 4, 5, 6, (7, 4)₁₀, 9, 10, 11, 13, 15, 16

Μπορούν να υπάρξουν τα ίδια μονοπάτια και με επανάληψη των βημάτων 3 και 2 (do while) στην περίπτωση όπου ο χρήστης εισάγει λάθος τύπου αριθμό στην αρχή.

Από τα μονοπάτια παραπάνω παρατηρούμε ότι δεν μπαίνει ποτέ το πρόγραμμα στον κόμβο 12 κι αυτό γιατί για να μπει εκεί θα πρέπει να μην ισχύει η συνθήκη της if παραπάνω, δηλαδή είτε να έχουμε $a \neq b$ είτε $b = c$, όπου εξαιτίας της δομής της while παραπάνω είναι αδύνατο, εφόσον για να βγει αρχικά από την επανάληψη θα πρέπει το a να γίνει ίσο με το b , ενώ κατά τη διάρκεια των επαναλήψεων όπου το b θα φτάσει το a , με τις υπάρχουσες συνθήκες πάντα το c θα φτάσει να ξεπερνάει το a , άρα και το b , συνεπώς πάντα θα ισχύουν οι δύο συνθήκες της if που συνδέονται με λογικό ΚΑΙ, επομένως η if θα είναι πάντα αληθής, άρα δε θα μπει ποτέ το πρόγραμμα στην else.

Συνεπώς το πρόγραμμα μπορεί να ελεγχθεί από τρία μονοπάτια δηλαδή λιγότερά από την κυκλωματική πολυπλοκότητα.

Ζητούμενο 5:

| Μονοπάτι | Τιμή Ελέγχου | Αποτέλεσμα |
|------------|--------------|---------------------------|
| M1, M2, M3 | a | Give an integer please |
| M1, M2, M3 | -1 | Give an integer please |
| M1, M2, M3 | 0 | Give an integer please |
| M1 | 1 | Number is 12 |
| M2 | 11 | Number is 22 |
| M3 | 50 | Number is 120 |
| M1, M2, M3 | 0.1 | Number is 12 |
| M1, M2, M3 | 1.1 | Το πρόγραμμα τερματίζεται |

Στις δύο τελευταίες περιπτώσεις παρατηρούμε ότι ενώ η είσοδος είναι εσφαλμένη το πρόγραμμα είτε βγάζει αποτέλεσμα είτε τερματίζει, ενώ και στις δύο περιπτώσεις θα έπρεπε να ξαναζητάει από το χρήστη μία σωστή είσοδο.