

Κεφάλαιο

10

Εμβέλεια μεταβλητών



Εμβέλεια μεταβλητών

Σε ένα πρόγραμμα της C μπορεί να δηλωθούν μεταβλητές σε διαφορετικά σημεία του προγράμματος. Η θέση και το είδος της δήλωσης μιας μεταβλητής καθορίζει την εμβέλειά της.

Εμβέλεια μιας μεταβλητής είναι ο χώρος του προγράμματος μέσα στον οποίο είναι γνωστή η μεταβλητή.

Ανάλογα με τη θέση και τον τρόπο δήλωσής τους, οι μεταβλητές χωρίζονται σε **τοπικές** (local), **καθολικές** (global), και **στατικές** (static).

Τοπικές μεταβλητές (local variables)

Όπως έχουμε δει μέχρι τώρα, οι μεταβλητές μιας συνάρτησης δηλώνονται αμέσως μετά από το αριστερό της άγκιστρο (`{`). Στην περίπτωση που η συνάρτηση έχει τυπικές παραμέτρους, αυτές δηλώνονται πριν από το αριστερό της άγκιστρο. Οι τυπικές παράμετροι είναι και αυτές μεταβλητές της συνάρτησης.

Οι μεταβλητές μιας συνάρτησης, καθώς και οι τυπικές της παράμετροι (αν υπάρχουν), καλούνται **τοπικές μεταβλητές** (local variables) και έχουν ισχύ μόνο μέσα στη συνάρτηση που δηλώθηκαν, παραμένοντας άγνωστες στο υπόλοιπο πρόγραμμα.

Με το παράδειγμα που ακολουθεί γίνεται κατανοητή η χρήση των τυπικών παραμέτρων και των τοπικών μεταβλητών.

```
main()
{
    int a,b,sum;
    printf("Δώσε δύο αριθμούς:");
    scanf("%d %d",&a,&b);
    sum=add(a,b);
    printf("Αθροισμα=%d\n",sum);
}
```

Τοπικές μεταβλητές
συνάρτησης main().

Στην επόμενη συνάρτηση `add()`, η μεταβλητή `ss` είναι τοπική μεταβλητή της συνάρτησης και έχει ισχύ μόνο μέσα στη συνάρτηση `add()`.

Οι τυπικές παράμετροι `x` και `y` αποτελούν και αυτές τοπικές μεταβλητές της συνάρτησης και έχουν ισχύ μόνο μέσα στη συνάρτηση.

```
add(x, y)
```

```
int x, y;
```

```
{
```

```
    int ss;
```

```
    ss=x+y;
```

```
    return(ss);
```

```
}
```

Δήλωση τυπικών παραμέτρων της συνάρτησης `add()`. Οι τυπικές παράμετροι παίζουν και το ρόλο τοπικών μεταβλητών.

Δήλωση τοπικής μεταβλητής `ss`.

Η συνάρτηση `add()` επιστρέφει ως τιμή το περιεχόμενο της μεταβλητής `ss`.

Οι τυπικές παράμετροι παίζουν και αυτές το ρόλο των τοπικών μεταβλητών, με μόνη διαφορά ότι μέσα σε αυτές αντιγράφονται οι τιμές των ορισμάτων με τα οποία καλείται η συνάρτηση.

Αν δεν γίνει κατανοητή η χρήση και η "εμβέλεια" των τυπικών παραμέτρων και των τοπικών μεταβλητών, είναι πολύ εύκολο να οδηγηθούμε σε λανθασμένα προγράμματα.

Στη συνέχεια βλέπετε το προηγούμενο πρόγραμμα, γεμάτο ενδεχόμενα λάθη.

```
main()
```

```
{
```

```
    int sum;
```

```
    float a,b;
```

```
    printf("Δωσε δύο αριθμούς:");
```

```
    scanf("%f %f",&a,&b);
```

```
    sum=add(a,b);
```

```
    printf("ss=%d\n",ss);
```

```
    printf("Αθροισμα=%d\n",sum);
```

```
}
```

Τα ορίσματα με τα οποία πρέπει να κληθεί η `add()` πρέπει να είναι τύπου `int` και όχι `float`.

Η συνάρτηση `main()` αγνοεί την ύπαρξη της `ss` η οποία είναι τοπική μεταβλητή της `add()`.


```
add(x,y)
int x,y;
{
    int ss;
    ss=a+b;
    return(ss);
}
```

Η συνάρτηση add() αγνοεί την ύπαρξη των a και b, οι οποίες είναι τοπικές μεταβλητές της main().

Στη συνέχεια βλέπετε ένα πιο ολοκληρωμένο πρόγραμμα με τρεις συναρτήσεις (εκτός από τη `main()`). Το πρόγραμμα ζητάει δύο ακέραιους αριθμούς και υπολογίζει το άθροισμα, το γινόμενο, και το μέσο όρο τους.

```
float mo();
main()
{
    int a,b;
    printf("Υπολογισμός Αθροίσματος, Γινομένου και Μέσου όρου\n");
    printf("-----\n");
    printf("Δώσε δύο αριθμούς:");
    scanf("%d %d", &a, &b);
    printf("Αθροισμα=%d\n", add(a,b));
    printf("Γινόμενο=%d\n", gin(a,b));
    printf("Μέσος όρος=%f\n", mo(a,b));
}

//υπολογισμός αθροίσματος
add(x,y)
int x,y;
{
    int ss;
    ss=x+y;
    return(ss);
}
```

```
//υπολογισμός γινομένου
```

```
gin(x,y)
```

```
int x,y;
```

```
{
```

```
    int gg;
```

```
    gg=x*y;
```

```
    return(gg);
```

```
}
```

```
//υπολογισμός μέσου όρου
```

```
float mo(x,y)
```

```
int x,y;
```

```
{
```

```
    float mmm;
```

```
    mmm=(x+y)/2.0;
```

```
    return(mmm);
```

```
}
```

Οι συναρτήσεις `add()` και `gin()` επιστρέφουν τιμές τύπου `int`. Όταν παραλειφθεί ο τύπος μιας συνάρτησης, η C θεωρεί ότι η συνάρτηση επιστρέφει τιμή `int`.

Η συνάρτηση `mo()` επιστρέφει τιμή `float`. Παρατηρούμε ότι η συνάρτηση `mo()` δηλώνεται δύο φορές, μία πριν από τη `main()` και μία όταν ορίζεται. Αυτό είναι απαραίτητο επειδή σε αντίθετη περίπτωση, όταν θα τη συναντούσε ο μεταγλωττιστής στη `main()`, θα υπέθετε ότι επιστρέφει ακέραια τιμή (τύπου `int`).

Με τη δήλωση του τύπου της `mo()` πριν από τη `main()`, προειδοποιούμε το μεταγλωττιστή ότι αν συναντήσει (πριν από τον ορισμό της) τη συνάρτηση `mo()`, να έχει υπόψη του ότι επιστρέφει τιμή τύπου `float` και όχι `int`.

Καθολικές μεταβλητές (global variables)

Στη C μπορούμε να δηλώσουμε μια μεταβλητή που να μην ανήκει σε καμία συνάρτηση. Μια τέτοια μεταβλητή δηλώνεται έξω από οποιαδήποτε συνάρτηση και καλείται **καθολική μεταβλητή** (global variable).

Οι καθολικές μεταβλητές έχουν εμβέλεια σε όλο το πρόγραμμα και κάθε συνάρτηση του προγράμματος μπορεί να τις προσπελάσει και να τις χρησιμοποιήσει.

Στο επόμενο πρόγραμμα, οι μεταβλητές **a** και **b** δηλώνονται ως καθολικές μεταβλητές στις οποίες μπορεί να έχουν πρόσβαση όλες οι συναρτήσεις.

```
int a,b;
main()
{
    a=8;
    b=10;
    printf("Αθροισμα=%d\n",add(a,b));
}
add(x,y)
int x,y;
{
    int ss;
    ss=x+y;
    printf("a=%d b=%d\n",a,b);
    return(ss);
}
```

Δήλωση των καθολικών μεταβλητών a και b.

Χρήση των a και b μέσα στη main().

Μεταβίβαση τιμών στις x και y.

Χρήση των a και b μέσα στην add().

Η χρήση γενικών μεταβλητών παρέχει έναν εναλλακτικό τρόπο μεταβίβασης πληροφοριών μέσα σε μια συνάρτηση.

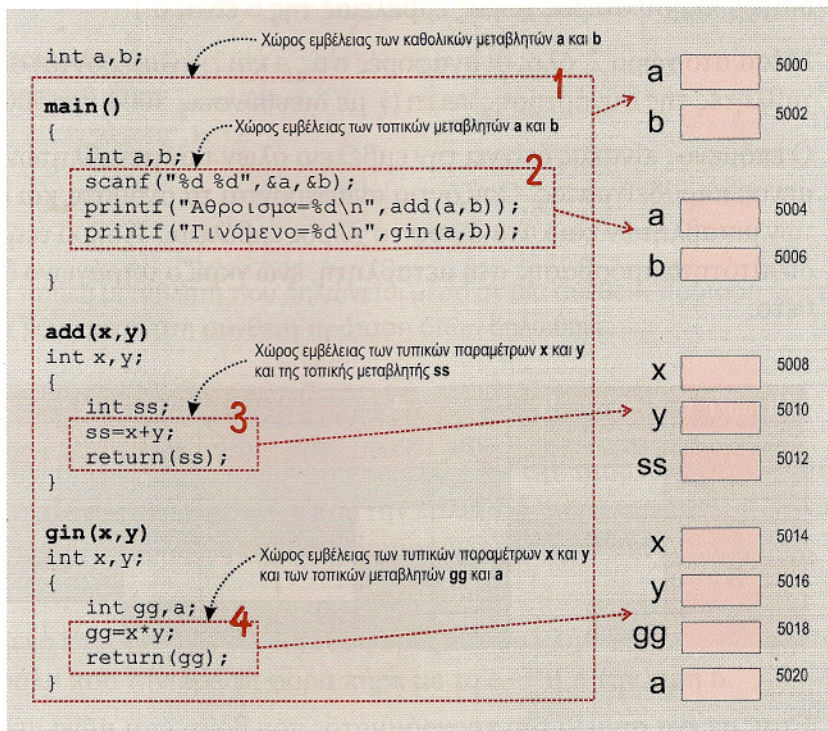
```
add()
{
    int ss;
    ss=a+b;
    return(ss);
}
```

Παρατηρούμε ότι με αυτόν τον τρόπο δεν υπάρχει ανάγκη μεταβίβασης των τιμών των a και b στη συνάρτηση add() μέσω παραμέτρων, δεδομένου ότι η συνάρτηση έχει άμεση πρόσβαση στις μεταβλητές a και b.

Η μέθοδος μεταβίβασης τιμών σε μια συνάρτηση μέσω καθολικών μεταβλητών πρέπει γενικά να αποφεύγεται και να χρησιμοποιείται μόνο όταν πραγματικά υπάρχει ανάγκη.

- ☞ Σε μια καθολική μεταβλητή έχουν πρόσβαση όλες οι συναρτήσεις που βρίσκονται μετά από το σημείο του προγράμματος όπου δηλώθηκε. Όταν λοιπόν θέλουμε να έχουν πρόσβαση σε μια καθολική μεταβλητή όλες ανεξαιρέτως οι συναρτήσεις πρέπει να δηλώσουμε τη μεταβλητή στην αρχή του προγράμματος (πριν από τη `main()`).
- ☞ Στην περίπτωση που θα δηλωθεί μια τοπική μεταβλητή με όνομα ίδιο με μια καθολική μεταβλητή, τότε στο χώρο εμβέλειας της τοπικής μεταβλητής αναφερόμαστε στην τοπική μεταβλητή ενώ οπουδήποτε αλλού στη καθολική μεταβλητή.

Το επόμενο παράδειγμα δείχνει την εμβέλεια τοπικών και καθολικών μεταβλητών:



Παρατηρούμε ότι κάθε φορά που δηλώνεται μια τοπική μεταβλητή (ή μια τυπική παράμετρος), δεσμεύεται διαφορετική θέση μνήμης παρόλο που μπορεί να υπάρχει άλλη τοπική ή καθολική μεταβλητή με το ίδιο όνομα.

Οι μεταβλητές **a** και **b** της συνάρτησης `main()` του παραδείγματος χρησιμοποιούν εντελώς διαφορετικές θέσεις μνήμης από τις καθολικές μεταβλητές **a** και **b**.

Όταν γίνεται αναφορά σε μια μεταβλητή, χρησιμοποιείται η μεταβλητή που έχει τον πλησιέστερο χώρο εμβέλειας.

Για παράδειγμα, όταν μέσα στο χώρο εμβέλειας 4 γίνει αναφορά στην μεταβλητή **a**, ουσιαστικά γίνεται αναφορά στη θέση με διεύθυνση 5020 και όχι στη θέση της καθολικής μεταβλητής **a** (με διεύθυνση 5000). Αναφορά στη μεταβλητή **b** (μέσα στο χώρο 4) σημαίνει αναφορά στη θέση με διεύθυνση 5002 διότι ο πλησιέστερος χώρος εμβέλειας της **b** είναι ο 1.

Μέσα στο χώρο 2, όλες οι αναφορές στις **a** και **b** αναφέρονται στις τοπικές μεταβλητές της συνάρτησης `main()` με διευθύνσεις 5004 και 5006 αντίστοιχα.

Ο επόμενος πίνακας δείχνει την εμβέλεια όλων των μεταβλητών του προηγούμενου παραδείγματος. Οριζόντια αναφέρονται τα ονόματα και οι διευθύνσεις των μεταβλητών και κατακόρυφα ο χώρος εμβέλειας. Άσπρο τετράγωνο δείχνει δυνατότητα πρόσβασης στη μεταβλητή, ενώ γκριζό τετράγωνο δείχνει το αντίθετο.

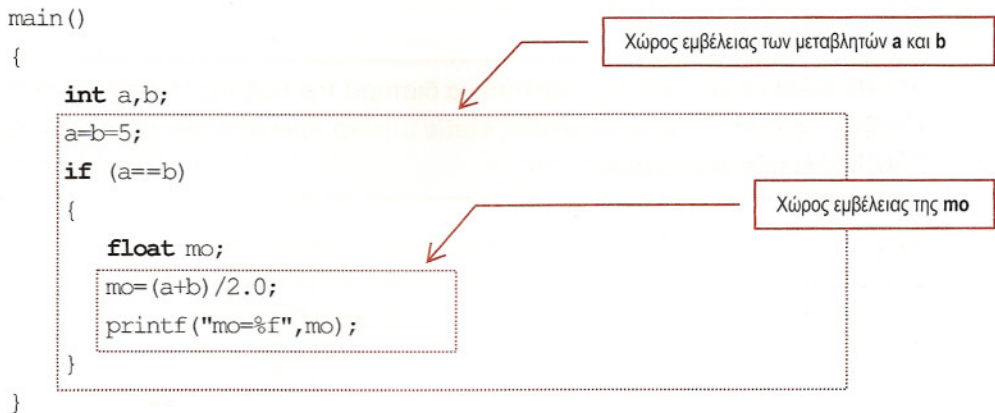
Μεταβλητή	a	b	a	b	x	y	ss	x	y	gg	a
Διεύθυνση	5000	5002	5004	5006	5008	5010	5012	5014	5016	5018	5020
1											
2											
3											
4											

Έτσι, σε ένα σημείο του προγράμματος που βρίσκεται **μόνο** μέσα στο χώρο 1 υπάρχει δυνατότητα πρόσβασης **μόνο** στις μεταβλητές **a** (5000) και **b** (5002).

Ένα σημείο του προγράμματος που βρίσκεται στο χώρο 4 έχει πρόσβαση στις μεταβλητές **b** (5002), **x** (5014), **y** (5016), **gg** (5018) και **a** (5020) και σε **καμία** άλλη.

Δήλωση τοπικών μεταβλητών σε σύνθετη πρόταση

Όπως ακριβώς δηλώνεται μια τοπική μεταβλητή μέσα σε μια συνάρτηση, μπορεί να δηλωθεί και αμέσως μετά την αριστερή αγκύλη μιας σύνθετης πρότασης (compound statement).



👉 Μια τοπική μεταβλητή που δηλώνεται μέσα σε μια σύνθετη πρόταση έχει εμβέλεια μόνο μέσα στη σύνθετη πρόταση όπου δηλώθηκε.

Στατικές τοπικές μεταβλητές (static local variables)

Οι τοπικές μεταβλητές μιας συνάρτησης χάνουν την τιμή τους όταν η συνάρτηση επιστρέψει στο πρόγραμμα που την κάλεσε. Όταν η συνάρτηση ξανακληθεί, οι τοπικές μεταβλητές της ξαναδημιουργούνται **χάνοντας** τις παλιές τους τιμές.

Στο επόμενο παράδειγμα, και τις δύο φορές που θα κληθεί η `go()` θα εμφανίσει το 5, παρόλο που την πρώτη φορά πήρε με το `a++` η `a` την τιμή 6.

```
main()
{
    go();
    go();
}

go()
{
    int a=5;
    printf("a=%d\n", a++);
}
```

5
5

Όταν θέλουμε μια τοπική μεταβλητή να διατηρεί την τιμή της, πρέπει να τη δηλώνουμε ως **static**. Ο προσδιορισμός **static** μπαίνει πριν από τον τύπο της μεταβλητής, στην πρόταση δήλωσής της.

```
main()
{
    go();
    go();
}



go()
{
    static int a=5;
    printf("a=%d\n", a++);
}
```

5
6

📖 Όταν μια τοπική μεταβλητή δηλώνεται ως **static**, τότε η πρόταση της δήλωσής της εκτελείται **μόνο μία** φορά. Αν στην πρόταση δήλωσης δίδεται και αρχική τιμή, τότε η αρχική τιμή δίνεται στη μεταβλητή **μόνο** την πρώτη φορά.

Έτσι, στο προηγούμενο παράδειγμα, η πρόταση **static int a=5;** εκτελείται μόνο την πρώτη φορά που καλείται η συνάρτηση **go()** και η **a** παίρνει την τιμή 5 μόνο την πρώτη φορά.

Με την παράσταση `a++` η `a` παίρνει την τιμή 6. Τη δεύτερη φορά που εκτελείται η συνάρτηση, η δηλωτική πρόταση δεν εκτελείται ξανά και η `a` διατηρεί την τιμή που είχε πριν (την τιμή 6).

-  Μια στατική μεταβλητή διατηρεί την τιμή της όταν επιστρέφει η συνάρτηση στην οποία δηλώθηκε, αλλά ο χώρος εμβέλειάς της παραμένει ο ίδιος όπως μιας οποιασδήποτε τοπικής μεταβλητής. Στο παραπάνω παράδειγμα ο χώρος εμβέλειας της `a` είναι η συνάρτηση `go()`.
-  Οι στατικές μεταβλητές δεσμεύουν μία θέση μνήμης στη φάση της μεταγλώττισης (όπως και οι καθολικές) και τη διατηρούν σε όλη τη διάρκεια του προγράμματος. Διαφέρουν με τις καθολικές μεταβλητές μόνο ως προς την εμβέλεια τους, η οποία περιορίζεται στη συνάρτηση ή στη σύνθετη πρόταση που δηλώθηκαν.

Παραδείγματα

- Π.1** Η επόμενη συνάρτηση επιστρέφει κάθε φορά που καλείται τον επόμενο αριθμό από αυτόν που επέστρεψε την προηγούμενη φορά. Ο πρώτος αριθμός που επιστρέφει είναι το 0, ο δεύτερος το 1 κ.ο.κ. Μετά από το 100 αρχίζει πάλι από το 0.

```
int nextnum()
{
    static int a=0;
    if(a>100) a=0;
    return(a++);
}
```


Ανασκόπηση Κεφαλαίου 10

- Οι μεταβλητές που δηλώνονται μέσα σε μια συνάρτηση καλούνται τοπικές μεταβλητές και έχουν εμβέλεια μόνο μέσα στη συνάρτηση στην οποία ορίζονται.
- Τοπικές μεταβλητές μπορούν να δηλωθούν και αμέσως μετά από την αριστερή αγκύλη οποιασδήποτε σύνθετης πρότασης. Μια τέτοια τοπική μεταβλητή έχει εμβέλεια μόνο μέσα στη σύνθετη πρόταση όπου ορίστηκε.
- Οι τυπικές παράμετροι μιας συνάρτησης είναι και αυτές τοπικές μεταβλητές, με εμβέλεια μόνο μέσα στην ίδια τη συνάρτηση.
- Οι καθολικές μεταβλητές δηλώνονται έξω από κάθε συνάρτηση (συνήθως στην αρχή του κώδικα) και έχουν εμβέλεια σε ολόκληρο το πρόγραμμα.
- Μια στατική μεταβλητή έχει περιορισμένη εμβέλεια όπως και μια τοπική μεταβλητή, αλλά διατηρεί την τιμή της και όταν επιστρέφει η συνάρτηση στην οποία δηλώθηκε.
- Όταν στη δήλωση μιας στατικής μεταβλητής της δοθεί ταυτόχρονα και αρχική τιμή, π.χ. `static int a=5`, η αρχική τιμή θα ανατεθεί στη στατική μεταβλητή μόνο την πρώτη φορά που θα εκτελεστεί η πρόταση δήλωσης.
- Μια τοπική μεταβλητή δεσμεύει μια θέση μνήμης η οποία απελευθερώνεται όταν λήξει η εμβέλειά της μεταβλητής και τα περιεχόμενά της χάνονται.
- Τόσο οι καθολικές, όσο και οι στατικές μεταβλητές, δεσμεύουν μια θέση μνήμης στη φάση της μεταγλώττισης και τη διατηρούν σε όλη τη διάρκεια του προγράμματος.

Ασκήσεις Κεφαλαίου 10

10.1 Ποιο θα είναι το αποτέλεσμα του επόμενου προγράμματος; Σχεδιάστε το χώρο εμβέλειας κάθε μεταβλητής. ★

```
void set();  
float a;  
float mo();
```

```
int x,y;

main()
{
    int x,y;
    x=y=4;
    set();
    a=mo(x,y);
    pp();
}

void set()
{
    int aa;
    x=10;
    y=20;
}

pp()
{
    int x,y;
    printf("a=%f\n",a);
}

float mo(k,l)
int k,l;
{
    float mesos;
    mesos=(k+l)/2.0;
    return mesos;
}
```

10.2 Τι κάνει το επόμενο πρόγραμμα; Να σχεδιαστούν (με ορθογώνια) οι εμβέλεις των μεταβλητών. ★ ★

```
void disp();
int step;
main()
{
    step=2;
    disp();
}

void disp()
{
    int a,b;
    printf("Μια συνάρτηση\n");
    a=get();
    b=get();
    if (a==0 && b==0)
        exit();
    else
    {
        int i;
        for (i=a;i<=b;i=i+step)
            printf("%d\n",i);
    }
}

get()
{
    int x;
    scanf("%d",&x);
    return x;
}
```


10.3 Βρείτε τα λάθη του επόμενου προγράμματος: ★

```
int step;

main()
{
    step=2;
    disp(4);
}

void disp(st)
{
    int a,b,c;
    printf("this is a function\n");
    a=get();
    c=x;
    b=get();
    if(a==0 && b==0)
        exit();
    else
    {
        int i;
        for(i=a;i<=b;i=i+step)
            printf("%d\n",i);
    }
    printf("i=%d\n",i);
    printf("step=%d\n",step);
}

get()
{
    float x;
    scanf("%f",&x);
    return x;
}
```

10.4 Τι αποτέλεσμα θα έχει το επόμενο πρόγραμμα; Να σχεδιαστούν οι εμβέλεις των μεταβλητών. ★★

```
void out();
int d,e;
main()
{
    int m=2,n=3,j;
    j=func1(m,n);
    func2(j);
    out(m,n,4);
}

func1(x,y)
int x,y;
{
    d=12;
    return x+y;
}

func2(d)
int d;
{
    d=20;
    e=d;
}

void out(a,b,c)
int a,b,c;
{
    printf("a=%d b=%d\n",a,b);
    printf("c=%d d=%d e=%d\n",c,d,e);
}
```

10.5 Τι αποτέλεσμα θα έχει το επόμενο πρόγραμμα; ★★

```
int x=10;
void out1();
void out2();
void out3();

main()
{
    int i;
    for(i=1;i<=5;i++) out1();
    for(i=1;i<=5;i++) out2();
    for(i=1;i<=5;i++) out3();
}

void out1()
{
    static int x=4;
    printf("%d\n",x++);
}

void out2()
{
    static int x;
    x=4;
    printf("%d\n",x++);
}

void out3()
{
    printf("%d\n",x++);
}
```


10.6 Να γραφεί συνάρτηση η οποία, κάθε φορά που θα καλείται, να επιστρέφει ως τιμή έναν λατινικό πεζό χαρακτήρα, σύμφωνα με την εξής λογική:
★ ★ ★

- Την πρώτη φορά που θα κληθεί, θα επιστρέψει το 'a', τη δεύτερη το 'b' και, γενικά, κάθε φορά που θα καλείται θα επιστρέφει τον επόμενο χαρακτήρα από αυτόν που επέστρεψε την προηγούμενη φορά. Όταν επιστρέψει το 'z' ο επόμενος θα είναι πάλι το 'a' κ.ο.κ.
- Να μη χρησιμοποιηθούν καθολικές μεταβλητές και η συνάρτηση να μην έχει παράμετρο.

10.7 Ποια από τα επόμενα αληθεύουν: ★

- ☐ Η εμβέλεια μιας στατικής μεταβλητής είναι όση και μιας καθολικής μεταβλητής.
- ☐ Η παράμετρος μιας συνάρτησης αποτελεί και τοπική μεταβλητή της συνάρτησης.
- ☐ Οι στατικές μεταβλητές διατηρούν την τιμή τους ανάμεσα στις κλήσεις μιας συνάρτησης.
- ☐ Μια μεταβλητή μπορεί να δηλωθεί αμέσως μετά την αριστερή αγκύλη μιας εντολής `for`.
- ☐ Στις τοπικές μεταβλητές, μόλις λήξει η εμβέλειά τους, χάνουν ταυτόχρονα και τα περιεχόμενά τους.