1.

Επαναλήψεις- Βρόχοι

1.1. Δομές Επανάληψης ή Βρόχοι

Πολλές φορές πρέπει να εκτελεστεί μια ομάδα εντολών περισσότερες από μια φορές. Σ' αυτές τις περιπτώσεις χρησιμοποιούνται οι δομές επανάληψης. Υπάρχουν τρεις δομές επανάληψης και μπορούν να κατηγοριοποιηθούν σε δυο περιπτώσεις χρήσης:

- Όταν το πλήθος των επαναλήψεων να είναι γνωστό εξ αρχής:
    - for που εκτελεί τις εντολές, που περικλείει, προκαθορισμένες φορές,
- Όταν το πλήθος των επαναλήψεων ΔΕΝ να είναι γνωστό εξ αρχής:
    - while που εκτελεί τις εντολές που περικλείει όσο η συνθήκη είναι ΑΛΗΘΗΣ και αυτή γράφεται στην αρχή της εντολής,
    - do…while που εκτελεί τις εντολές, που περικλείει, όσο η συνθήκη είναι ΨΕΥΔΗΣ και αυτή γράφεται στο τέλος της εντολής.

<u>Όλες οι δομές επανάληψης έχουν τρία πολύ σημαντικά χαρακτηριστικά, στην πλειοψηφία των περιπτώσεων χρήσης τους:</u>

- Αρχική Τιμή (αρχικοποίηση)
- Τελική Τιμή (Τερματισμός και παράλληλα έλεγχος για τον τερματισμό)
- Βήμα Μεταβολής (Αύξησης/Μείωσης)

1.2. Δομή επανάληψης for

Η δομή επανάληψης for χρησιμοποιείται όταν είναι γνωστό το πλήθος των επαναλήψεων, π.χ. να βρούμε το ΜΟ 35 μαθητών. Έχει σχεδιαστεί για να απλοποιήσει τον έλεγχο των επαναλήψεων, αν και μπορούν όλα τα προβλήματα να υλοποιηθούν και με τις άλλες δύο δομές επανάληψης υπό συνθήκη. Η γενική της σύνταξη είναι:

```
for(αρχική_τιμή_μεταβλητής_ελέγχου;
τελική_τιμή_μεταβλητής_ελέγχου;
βήμα_μεταβολής_μεταβλητής_ελέγχου){
      .
      . <εντολές>
      .
}
```

Η δομή αυτή χειρίζεται από μια μεταβλητή στην οποία καταχωρείται αρχικά η αρχική τιμή. Η τιμή που παίρνει η μεταβλητή συγκρίνεται με τη τελική τιμή και εφόσον δεν έχει φτάσει στην τιμή αυτή (θέλει προσοχή στο χειρισμό της σε περιπτώσεις που αποκλίνει σύμφωνα με τη μεταβολή), τότ7ε εκτελούνται οι εντολές που βρίσκονται στο βρόχο. Στη συνέχεια η μεταβλητή αλλάζει κατά την τιμή του βήματος μεταβολής. Αν η νέα τιμή είναι μικρότερη της τελικής, τότε ο βρόχος εκτελείται ξανά. Η διαδικασία επαναλαμβάνεται μέχρι η μεταβλητή να γίνει ξεπεράσει (θετικά ή αρνητικά, ανάλογα με τον τρόπο μεταβολής) την τελική τιμή ελέγχου, αλλιώς η δομή τερματίζεται και ο έλεγχος του προγράμματος συνεχίζει με την επόμενη- μετά το βρόχο- εντολή.

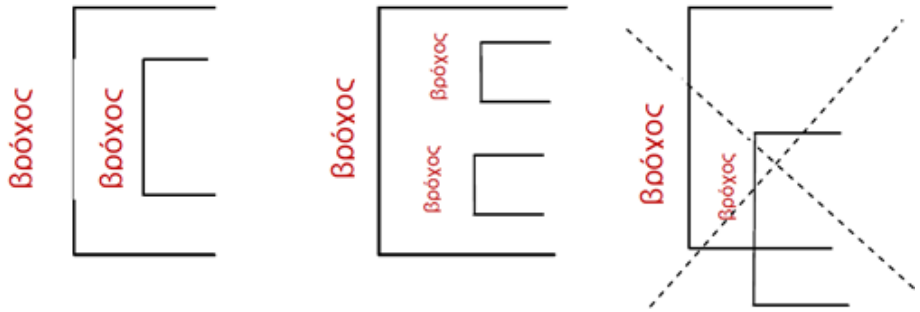
Πρέπει να επισημανθεί ότι:

1. Μέσα στη δομή for **δεν επιτρέπεται να αλλάζετε την τιμή της μεταβλητής ελέγχου της δομής for**, γιατί τότε δεν λειτουργεί σωστά η εντολή επανάληψης.
2. Το βήμα δεν είναι απαραίτητο να είναι ακέραιος αριθμός. Μπορεί να πάρει οποιαδήποτε πραγματική τιμή.

1.3. Εμφωλευμένοι Βρόχοι

Στην περίπτωση αυτή ο ένας βρόχος βρίσκεται μέσα στον άλλο. Για τη σωστή λειτουργία των εμφωλευμένων βρόχων πρέπει να ισχύουν συγκεκριμένοι κανόνες:

- Ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό
- Οι επαναλήψεις του εσωτερικού βρόγχου πρέπει να ολοκληρώνονται για όλες τις τιμές της μεταβλητής του εσωτερικού και έπειτα να μεταβάλλεται το βήμα της εξωτερικής και πάλι να επαναλαμβάνεται ο εσωτερικός βρόγχος για όλες τις τιμές της εσωτερικής και στο τέλος να μεταβάλλεται πάλι το βήμα της εξωτερικής.



Η είσοδος σε βρόχο υποχρεωτικά γίνεται από την αρχή του.

- **Δεν πρέπει να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσοτέρων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου.**

1.4. Δομή Επανάληψης while

Όταν δεν είναι καθορισμένο το πλήθος των επαναλήψεων εξ' αρχής τότε δεν μπορεί να χρησιμοποιηθεί η δομή επανάληψης for. Σε αυτήν την περίπτωση χρησιμοποιείται η δομή while, με την οποίες οι εντολές που περιλαμβάνει επαναλαμβάνονται όσο η συνθήκη που ελέγχει είναι ΑΛΗΘΗΣ και μόνο όταν γίνει ΨΕΥΔΗΣ σταματά να εκτελεί αυτές και εκτελεί την αμέσως επόμενη εντολή μετά της δομής επανάληψης while.

Στον τρόπο λειτουργίας της δομής πρέπει να επισημανθούν τα ακόλουθα χαρακτηριστικά:

- Επειδή η συνθήκη βρίσκεται στην αρχή της εντολής υπάρχει περίπτωση να βγει ΨΕΥΔΗΣ από την αρχή οπότε και να μην εκτελεστούν καμία φορά οι εντολές της επανάληψης.
- Αν η συνθήκη δεν βγει ΨΕΥΔΗΣ ποτέ, τότε οι εντολές της επανάληψης εκτελούνται συνεχώς και ο αλγόριθμος (ή το πρόγραμμα) δεν τελειώνει ποτέ. Τότε λέμε ότι έχουμε "ατέρμων βρόχο".

Κατά τη σύνταξη της:

- Δίνουμε αρχική τιμή στη μεταβλητή της συνθήκης, γιατί η συνθήκη βρίσκεται στην αρχή της while. και πρέπει η μεταβλητή να έχει αρχικοποιηθεί για να μπορεί να εκτελέσει ο υπολογιστής τη συνθήκη και να δώσει τη τιμή ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ.
- Πριν τελειώσει η επανάληψη, δηλαδή πριν το τέλος της επανάληψης πρέπει να αλλάξουμε τη τιμή της μεταβλητής της συνθήκης, όπου αυτό χρειάζεται, για να μην έχουμε ατέρμων βρόχο.

Η γενική σύνταξη της είναι της μορφής:

```
while(συνθήκη){
      .
      . <εντολές>
      .
}
```

Και ορισμένες από πιο συχνές μορφές εμφάνισης της, σε γενική περιγραφή είναι:

```
scanf(μεταβλητή);
while(έλεγχος_τελικής_τιμής_μεταβλητής){
```

```
        .
        . <εντολές>
        .
        scanf(μεταβλητή);
}
```

ή

```
scanf(μεταβλητή);
while(έλεγχος_τελικής_τιμής_μεταβλητής){
        .
        . <εντολές>
        .
        μεταβολή_μεταβλητής_ελέγχου; (εντός της δομής, σε όποιο σημείο
        έχει το επιθυμητό αποτέλεσμα)
}
```

ή

```
αρχικοποίηση μεταβλητής ελέγχου;
while(έλεγχος_τελικής_τιμής_μεταβλητής){
        .
        . <εντολές>
        .
        μεταβολή_μεταβλητής_ελέγχου; (εντός της δομής, σε όποιο σημείο
        έχει το επιθυμητό αποτέλεσμα)
}
```

ή

```
παράσταση υπολογισμού μεταβλητής ελέγχου;
while(έλεγχος_τελικής_τιμής_μεταβλητής){
        .
        . <εντολές>
        .
        μεταβολή_μεταβλητής_ελέγχου; (εντός της δομής, σε όποιο σημείο
        έχει το επιθυμητό αποτέλεσμα)
}
```

Ωστόσο, στη συνθήκη ελέγχου μπορεί αντί για μια μόνο μεταβλητή να υπάρχει παράσταση, κυρίως με τη συνδυαστική χρήση λογικών πράξεων, οπότε είτε η ανάγνωση, είτε η αρχικοποίηση είτε η παράσταση υπολογισμού αναφέρεται σε όλες τις μεταβλητές που εμπλέκονται στο έλεγχο.

1.5. Δομή Επανάληψης do…while

Συμπληρωματικά της δομής while χρησιμοποιείται η δομή do…while. Η σύνταξη της δομής είναι:

```
do{
        .
        . <εντολές>
        .
} while(συνθήκη);
```

Παρατηρούμε ότι σε αυτή τη δομή ο έλεγχος της συνθήκης πραγματοποιείται στο τέλος του βρόχου. Οι εντολές που περιλαμβάνει επαναλαμβάνονται μέχρις ότου η συνθήκη γίνει αληθής. Δηλαδή, όσο η συνθήκη είναι ΨΕΥΔΗΣ έχουμε επανάληψη, ενώ όταν η συνθήκη γίνει ΑΛΗΘΗΣ η επανάληψη σταματά και εκτελείται η αμέσως επόμενη εντολή μετά το βρόγχο.

Μια πιο πρακτική μορφή χρήσης της είναι:

```
αρχικοποίηση μεταβλητής ελέγχου;
do{
```

```
        .
        . <εντολές>
        .
        μεταβολή_μεταβλητής_ελέγχου; (εντός της δομής, σε όποιο σημείο
        έχει το επιθυμητό αποτέλεσμα)
    } while(συνθήκη_έλεγχος_τελικής_τιμής_μεταβλητής_ελέγχου);
```

Και σε αυτήν την περίπτωση, στη συνθήκη ελέγχου μπορεί αντί για μια μόνο μεταβλητή να υπάρχει παράσταση με τη συνδυαστική χρήση λογικών πράξεων.

1.6. Σύγκριση των δομών επανάληψης while και do…while

Στη δομή do…while η συνθήκη που ελέγχεται είναι η αντίθετη από την while.

Επειδή η συνθήκη στην εντολή επανάληψης do…while βρίσκεται στο τέλος τη εντολής οι εντολές, που περιλαμβάνονται σ' αυτήν, θα εκτελεστούν τουλάχιστον μια φορά.

Χρησιμοποιείται:

- Στα προβλήματα που έχουν Μενού επιλογής.
- Στα προβλήματα που έχουμε Έλεγχο δεδομένων από το πληκτρολόγιο.
- Στα προβλήματα που θέλουμε να επαναλάβουμε μια ολόκληρη επεξεργασία (π.χ. Αναζήτηση δεδομένων) απαντώντας στην ερώτηση 'Θέλεις να συνεχίσεις (Ν/Ο)?'.

Στις παραπάνω τρεις περιπτώσεις, που αναφέρθηκαν για επιλογή της εντολής do...while, μπορεί να χρησιμοποιηθεί και η while αλλά οδηγεί σε υλοποιήσεις με περισσότερες εντολές.

1.7. Εργαστηριακές Ασκήσεις

1. Γράψτε ένα πρόγραμμα το οποίο θα εκτυπώνει τους αριθμούς 1 έως 100 διατεταγμένους σε 10 στήλες. Κάθε αριθμός θα πρέπει να χωρίζεται από τον επόμενό του με ένα χαρακτήρα οριζόντιου στηλοθέτη (tab).
2. Να υλοποιηθεί πρόγραμμα σε C που θα εκτυπώνει τους ASCII χαρακτήρες και την αντίστοιχη δεκαδική τους αναπαράσταση. Τα μοναδικά δεδομένα που σας δίνονται είναι ότι η δεκαδική αναπαράσταση του χαρακτήρα μηδέν (0) είναι το 48 και ότι θα εκτυπώνει μέχρι τον χαρακτήρα που η δεκαδική του αναπαράσταση είναι το 126. Η εκτύπωση των χαρακτήρων και της δεκαδικής τους τιμής θα γίνει ως εξής: σε κάθε γραμμή θα εκτυπώνονται δέκα ASCII χαρακτήρες με την αντίστοιχη δεκαδική τους αναπαράσταση μέσα σε παρένθεση. Μεταξύ δύο χαρακτήρων θα παρεμβάλλεται ένας χαρακτήρας ‘\t’, π.χ. 0 (48) 1(..) 2 (..) … 9(..)
3. Δημιουργήστε ένα πρόγραμμα το οποίο δέχεται από το χρήστη το πληθυσμό μιας πόλης και τον ετήσιο ρυθμό αύξησής του και υπολογίστε και εμφανίστε στην οθόνη σε πόσα χρόνια θα έχει διπλασιαστεί.

4. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει έναν ακέραιο (π.χ. N) και να εμφανίζει το αποτέλεσμα της παράστασης: $3^2+6^2+9^2+ ... +(3\times N)^2$. Το πρόγραμμα να υποχρεώνει τον χρήστη να εισάγει έναν θετικό ακέραιο μικρότερο του 10. Να χρησιμοποιήσετε έναν μόνο for βρόχο.

5. Να γράψετε πρόγραμμα που δέχεται ως είσοδο ένα αριθμό και τσεκάρει αν ο αριθμός είναι πρώτος ή όχι. Αν ΔΕΝ είναι πρώτος να εμφανίσει το πλήθος των αριθμών που τον διαιρούν ακριβώς;

6. Να γραφεί πρόγραμμα το οποίο διαβάζει μία ακολουθία από n αριθμούς και εμφανίζει τον αριθμό που εμφανίστηκε στην ακολουθία περισσότερες συνεχόμενες φορές.

7. Υπολογίστε την ακολουθία: 1+ (1+2) +(1+2+3)+(1+2+3+4) ...+(1+2+3+..+n). Ως είσοδο θα δέχεται την τιμή n, π.χ. n =20 : output= 1540. Στην συνέχεια υπολογίστε την ακολουθία :1 -(1+2) +(1+2+3)-(1+2+3+4) … έως n, π.χ. n=20 : output =-110 , n=21 output =121.

8. Ο μέγιστος κοινός διαιρέτης δύο θετικών αριθμών x, y μπορεί να υπολογιστεί σύμφωνα με τον αλγόριθμο του Ευκλείδη (http://el.wikipedia.org/wiki/Αλγόριθμος_του_Ευκλείδη) ως εξής: έστω x>y, αν το y διαιρεί ακριβώς το x τότε ο μέγιστος κοινός διαιρέτης είναι το y, διαφορετικά αντικατέστησε το x με το υπόλοιπο της διαίρεσής του με το y, και επανέλαβε τον έλεγχο. Υλοποιήστε κώδικα που υπολογίζει το μέγιστο κοινό διαιρέτη δύο θετικών αριθμών.

9. Η ακολουθία Fibonacci είναι μια ακολουθία ακεραίων, στην οποία ο κάθε ακέραιος προκύπτει από το άθροισμα των δυο προηγουμένων του, με αρχικές τιμές $F_0=0$, $F_1=1$. Η ακολουθία σχηματίζεται δηλαδή ως εξής:

   0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ..., με $F_n = F_{n-1} + F_{n-2}$

   Να αναπτυχθεί κώδικας σε C ο οποίος θα υπολογίζει και θα εκτυπώνει στην οθόνη τα N πρώτα στοιχεία της ακολουθίας Fibonacci, όπου N ακέραιος τον οποίο θα εισάγει ο χρήστης με χρήση της scanf. Σε περίπτωση που ο N είναι μεγαλύτερος του 20, θα τυπώνονται μέχρι τα 20 πρώτα στοιχεία και όχι παραπάνω. Τα στοιχεία θα πρέπει να εμφανίζονται σε κάθε γραμμή ανά 5.

10. Κατασκευάστε ένα πρόγραμμα που θα προσομοιώνει μία συναλλαγή με ATM. Οι δυνατές συναλλαγές με το χρήστη είναι:
    i. Διαθέσιμο υπόλοιπο
    ii. Κατάθεση μετρητών
    iii. Ανάληψη μετρητών
    iv. Έξοδος

    Ο λογαριασμός θα έχει μέσα αρχικά 100 ευρώ. Αφού θα γίνεται πρώτα έλεγχος του pin έπειτα θα προχωράει στις παραπάνω επιλογές, αλλιώς θα τυπώνει στην οθόνη μήνυμα λάθους και θα ζητάει εκ νέου το pin. Ως pin της κάρτας θεωρήστε τον αριθμό 2014. Το ποσό ανάληψης επιτρέπεται να είναι πολλαπλάσιο των 20 ευρώ. Στην ανάληψη θα τσεκάρει αν υπάρχει το διαθέσιμο υπόλοιπο στο λογαριασμό.

    Υπόδειγμα: ο τύπος δεδομένων των μεταβλητών που έχουν σχέση με χρήματα να δηλωθούν ως unsigned long.

11. Να γράψετε πρόγραμμα σε C, που θα διαβάζει έναν ακέραιο αριθμό Ν από το πληκτρολόγιο και θα εμφανίζει:
    Α. τους Ν πρώτους φυσικούς αριθμούς,
    Β. το άθροισμά τους,
    Γ. την μέση τιμή τους,
    Δ. μια νέα ακολουθία αριθμών, όπου κάθε στοιχείο της θα ισούται με το αντίστοιχο της πρώτης υψωμένο στο τετράγωνο.

2.

Μονοδιάστατοι Πίνακες

2.1. Μονοδιάστατοι-Πολυδιάστατοι Πίνακες

Οι πίνακες εκφράζουν μια μεταβλητή που έχει περισσότερες από μια θέσεις, δηλαδή μια μεταβλητή πολλαπλών θέσεων. Για το λόγο αυτό όταν ορίζουμε ένα πίνακα, χρησιμοποιούμε το όνομα της μεταβλητής και το δείκτη, που είναι ένας αριθμός και εκφράζει τη θέση του κάθε δεδομένου. Ένας πίνακας μπορεί να είναι μονοδιάστατος ή μεγαλύτερης διάστασης.

Ο πίνακας είναι ένα σύνολο δεδομένων ίδιου τύπου. Αποθηκεύει μια διαδοχική συλλογή στοιχείων του ίδιου τύπου σταθερού μεγέθους. Κάθε ένα από τα δεδομένα του πίνακα λέγεται στοιχείο του πίνακα. Όταν αναφερόμαστε σε ένα δεδομένο από τον πίνακα τότε γράφουμε το όνομα του πίνακα ακολουθούμενο από έναν δείκτη.

Όλοι οι πίνακες αποτελούνται από συνεχόμενες θέσεις μνήμης. Η χαμηλότερη διεύθυνση αντιστοιχεί στο πρώτο στοιχείο και η υψηλότερη διεύθυνση στο τελευταίο στοιχείο.

Αντί να δηλώνετε μεμονωμένες μεταβλητές, όπως number0, number1, ... και number99, δηλώνετε μια μεταβλητή πίνακα, π.χ. τη number και χρησιμοποιείτε number[0], number[1], ..., number[99] για να αναπαραστήσετε τις μεμονωμένες μεταβλητές που βρίσκονται στις θέσεις 0, 1, …, 99 του πίνακα.

Πρώτο Στοιχείο 0	1	2	3	…	Τελευταίο Στοιχείο N-1
Array[0]	Array[1]	Array[2]	Array[3]	…	Array[N-1]

Τα πλεονεκτήματα χρήσης πινάκων, είναι η χρήση πίνακα είναι ένας βολικός τρόπος για τη διαχείριση πολλών δεδομένων ίδιου τύπου, ενώ τα μειονεκτήματα είναι το γεγονός ότι απαιτούν

μνήμη, η οποία δεσμεύεται από την αρχή του προγράμματος. Αυτό έχει σαν αποτέλεσμα, σε μεγάλα και σύνθετα προγράμματα την αδυναμία εκτέλεσης τους, ενώ παράλληλα οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος.

Χρησιμοποιούνται στη περίπτωση που έχουμε να διαχειριστούμε πολλά δεδομένα και πρέπει να είναι αποθηκευμένα για μεταγενέστερη επεξεργασία τους.

Για να δηλώσετε έναν πίνακα πρέπει να καθορίσετε τον τύπο των δεδομένων και το πλήθος των στοιχείων που απαιτούνται από έναν πίνακα ως εξής:

<τύπος_δεδομένων> Όνομα_Πίνακα[μέγεθος_πίνακα];

Για παράδειγμα, για να δηλώσετε έναν πίνακα 10 στοιχείων που ονομάζεται MyArray τύπου double, χρησιμοποιείται τη δήλωση:

double MyArray[10];

Μπορείτε να αρχικοποιήσετε έναν πίνακα, αρχικοποιώντας είτε ένα-προς-ένα τα στοιχεία που περιέχει ή χρησιμοποιώντας μια γενικής πρόταση αρχικοποίησης του, όπως για παράδειγμα:

double MyArray[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

Ο αριθμός των τιμών μεταξύ των αγκύλων { } δεν μπορεί να είναι μεγαλύτερος από τον αριθμό των στοιχείων που δηλώνουμε για τον πίνακα μεταξύ αγκύλων δήλωσης μεγέθους του πίνακα ([ ]).

Εάν παραλείψετε το μέγεθος του πίνακα, δημιουργείται ένας πίνακας αρκετά μεγάλος ώστε να διατηρεί τη δήλωση αρχικοποίησης, επομένως, εάν γράψετε:

double MyArray[] = {1000.0, 2.0, 3.4, 7.0, 50.0};

θα δημιουργήσετε ακριβώς τον ίδιο πίνακα όπως παραπάνω.

Μια γενική πρόταση για την εκχώρηση ενός μεμονωμένου στοιχείου στις θέσεις των στοιχείων του πίνακα είναι η ακόλουθη:

MyArray[4] = 50.0;

δήλωση η οποία εκχωρεί στο 5ο στοιχείο του πίνακα την τιμή 50.0.

Όλοι οι πίνακες έχουν το 0 ως δείκτη του πρώτου τους στοιχείου που ονομάζεται επίσης δείκτης βάσης και ο τελευταίος δείκτης ενός πίνακα θα είναι το συνολικό μέγεθος του πίνακα μείον 1.

Για την προσπέλαση ενός μονοδιάστατου πίνακα N θέσεων απαιτείται η χρήση μιας δομής επανάληψης

Ένα παράδειγμα δήλωσης και προσπέλασης πίνακα είναι:

```
#include <stdio.h>

int main () {

   int n[ 10 ];
   int i,j;

   for ( i = 0; i < 10; i++ ) {
      n[ i ] = i + 100;
   }

   for (j = 0; j < 10; j++ ) {
```

```
        printf("Element[%d] = %d\n", j, n[j] );
    }

    return 0;
}
```

Το οποίο όταν εκτελεστεί θα μας παράγει σαν έξοδο το ακόλουθο:

Element[0] = 100

Element[1] = 101

Element[2] = 102

Element[3] = 103

Element[4] = 104

Element[5] = 105

Element[6] = 106

Element[7] = 107

Element[8] = 108

Element[9] = 109

2.2. Εργαστηριακές Ασκήσεις

1. Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει 30 ακεραίους και να τους αποθηκεύει σε ένα πίνακα. Στη συνέχεια το πρόγραμμα να εμφανίζει πόσες φορές επαναλαμβάνονται οι τιμές των στοιχείων του πίνακα και ποια τιμή εμφανίστηκε τις περισσότερες φορές και πόσες είναι αυτές. Ελέγξτε και την περίπτωση που να είναι περισσότεροι από ένας οι τιμές με τον ίδιο μέγιστο αριθμό εμφανίσεων. Θεωρήστε ότι οι τιμές είναι ακέραιοι στο [0, 20] και το πρόγραμμα να κάνει αμυντικό προγραμματισμό ότι οι τιμές που δίνονται είναι στο σωστό εύρος.
2. Γράψτε πρόγραμμα που να υπολογίζει όλους τους πρώτους αριθμούς μεταξύ του 1 και του N χρησιμοποιώντας το κόσκινο του Ερατοσθένη (κοιτάξτε σχετικό άρθρο στη Wikipedia https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes).

   Τα βήματα του αλγορίθμου είναι:

   i. καταγράφουμε όλους τους αριθμούς από το 2 μέχρι το N σε μια λίστα
   ii. για κάθε πρώτο που συναντάμε στη λίστα, διαγράφουμε από τη λίστα όλα τα πολλαπλάσιά του
   iii. αν ο πρώτος αριθμός που συναντάμε στη λίστα είναι μικρότερος της τετραγωνικής ρίζας του N, τότε επαναλαμβάνουμε το βήμα 2, διαφορετικά η λίστα περιλαμβάνει μόνο πρώτους αριθμούς.

3. Να υλοποιήσετε τον αλγόριθμο ταξινόμησης BubbleSort. Ο Bubblesort είναι ένας απλός αλγόριθμος ταξινόμησης (http://en.wikipedia.org/wiki/Bubble_sort) που λειτουργεί σαρώνοντας όλη την ακολουθία στοιχείων που πρέπει να διαταχθεί, συγκρίνοντας κάθε ζεύγος γειτονικών στοιχείων και εναλλάσσοντάς τα αν είναι σε λάθος σειρά. Το πέρασμα από την ακολουθία στοιχείων επαναλαμβάνεται έως ότου δεν χρειάζεται να υπάρχουν εναλλαγές στοιχείων, γεγονός που υποδεικνύει ότι η ακολουθία είναι ταξινομημένη. Τα στοιχεία τα οποία πρέπει να ταξινομηθούν είναι πραγματικοί αριθμοί σε έναν πίνακα μεγέθους 30.

   Για δική σας διευκόλυνση να φτιάξετε μία συνάρτηση η οποία θα τυπώνει τον πίνακα και θα τη χρησιμοποιείτε όταν συμβαίνει μία εναλλαγή μεταξύ στοιχείων του πίνακα. Τα στοιχεία του πίνακα θα τα δίνετε εσείς από το πληκτρολόγιο. Μία δεύτερη συνάρτηση μπορεί να είναι αυτή η οποία θα πραγματοποιεί την εναλλαγή μεταξύ δύο στοιχείων του πίνακα.