ΜΑΘΗΜΑ 11

ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ DO...WHILE

Γιώργος Διάκος - Full Stack Developer

3. Η δομή do..while

1. Συντακτικό της do..while

- Το συντακτικό της εντολής do…while (επανέλαβε…όσο) είναι:

```
do
{
    (Εντολές)
}
while (Συνθήκη);
```

- Με την παρατήρηση ότι εδώ δεν μπορούμε να παραλείψουμε τα άγκιστρα!.

- Με το παραπάνω συντακτικό γίνονται οι εξής ενέργειες:
  - 1. Εκτελούνται οι εντολές
  - 2. Ελέγχεται η συνθήκη: (την οποία έχουμε συντάξει με λογικούς και σχεσιακούς τελεστές)
    - 2.1 Αν η απάντηση είναι ΑΛΗΘΗΣ (ισχύει η συνθήκη), τότε ξαναρχίζουμε από την αρχή (πρώτη εντολή μετά από το do)
    - 2.2 Αν η απάντηση είναι ΨΕΥΔΗΣ (δεν ισχύει η συνθήκη), τότε βγαίνουμε από την επανάληψη και τρέχουμε την αμέσως επόμενη εντολή μετά τη while.

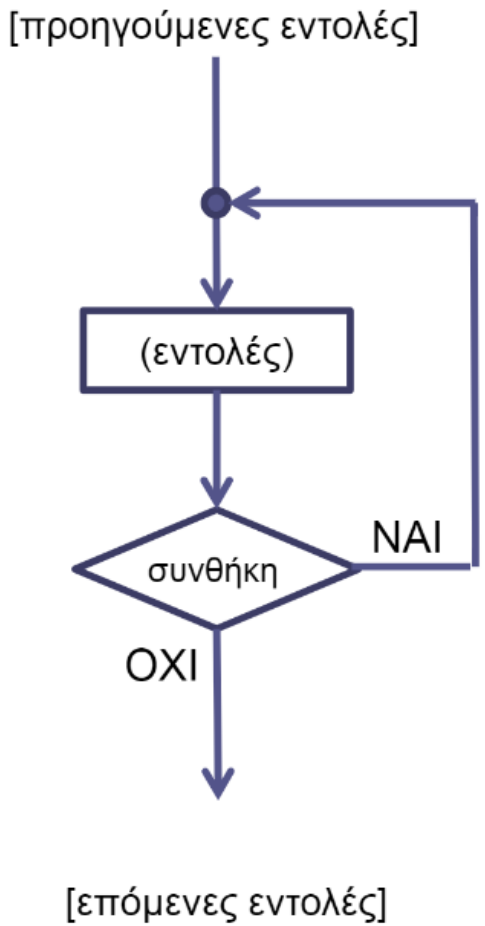
3. Η δομή do..while

2. Διάγραμμα Ροής Προγράμματος

- Και το διάγραμμα ροής προγράμματος:

```
[προηγούμενες εντολές]

do
{
   (Εντολές)
}
while (Συνθήκη);

[επόμενες εντολές]
```

- Παρατηρήστε ότι η συνθήκη ελέγχεται αφού έχουν εκτελεστεί οι εντολές



3. Η δομή do…while

3. Παραδείγματα Εκτέλεσης

- Μπορούμε να χρησιμοποιήσουμε την do..while με αντίστοιχο τρόπο με την for.
- Παράδειγμα 1:
  - Τι κάνει το ακόλουθο τμήμα κώδικα (η `i` είναι ακέραια μεταβλητή);

```
i=0;
do{
   i=i+1;
   printf("\n%d",i);
} while(i<3);
```

  - Αρχικά `i=0`
    - Γίνεται επανάληψη:
      - Τίθεται `i=0+1=1` και τυπώνεται «1»
      - Γίνεται έλεγχος συνθήκης. `1<3` είναι ΑΛΗΘΗΣ, άρα θα επαναλαβουμε
    - Γίνεται επανάληψη:
      - Τίθεται `i=1+1=2` και τυπώνεται «2»
      - Γίνεται έλεγχος συνθήκης. `2<3` είναι ΑΛΗΘΗΣ, άρα θα επαναλαβουμε
    - Γίνεται επανάληψη:
      - Τίθεται `i=2+1=3` και τυπώνεται «3»
      - Γίνεται έλεγχος συνθήκης. `3<3` είναι ΨΕΥΔΗΣ, άρα δεν θα επαναλαβουμε και πηγαίνουμε στην επόμενη εντολή του κώδικα.

3. Η δομή do…while

3. Παραδείγματα Εκτέλεσης

- Και εδώ θα κάνουμε ένα από τα πιο συχνά λάθη! Θα πέσουμε σε ατέρμονα βρόχο!
- Παράδειγμα 2: Τι κάνει το ακόλουθο τμήμα κώδικα (η `i` είναι ακέραια μεταβλητή);

```
i=5;
do{
   i=i+1;
   printf("\n%d",i);
} while(i>3);
```

  - Αρχικά `i=5`
    - Γίνεται επανάληψη:
      - Τίθεται `i=5+1=6` και εκτυπώνεται «`6`»
      - Γίνεται έλεγχος συνθήκης. `6>3` είναι ΑΛΗΘΗΣ, άρα θα επαναλαβουμε
    - Γίνεται επανάληψη:
      - Τίθεται `i=6+1=7` και εκτυπώνεται «`7`»
      - Γίνεται έλεγχος συνθήκης. `7>3` είναι ΑΛΗΘΗΣ, άρα θα επαναλαβουμε....
  - ..... Θα τερματίσει ποτέ το πρόγραμμα?
  - Η απάντηση είναι ΟΧΙ! Σε κάθε επανάληψη το i θα αυξάνεται, άρα ποτέ δεν θα γίνει <= 3!

- **Ατέρμων βρόχος** (infinite loop) είναι μία (επανάληψη που δεν ολοκληρώνεται ποτέ)
- Αποτελεί από τα συχνότερα προγραμματιστικά λάθη!

3. Η δομή do…while

3. Παραδείγματα Εκτέλεσης

Παρατήρηση:

- Η σύνταξη της δομής είναι αρκετά απλή, αλλά θα πρέπει εμείς, ως προγραμματιστές να συντάξουμε σωστά τις υπόλοιπες εντολές. Συγκεκριμένα:
  - Πρέπει να αρχικοποιήσουμε σωστά την μεταβλητή που θα έχουμε στην εντολή συνθήκης. Έτσι πριν την do θα πρέπει να αρχικοποιήσουμε την μεταβλητή που θα χρησιμοποιήσουμε (**εντολή αρχικοποίησης**)
  - Πρέπει η μεταβλητή που έχουμε στην εντολή αρχικοποίησης να επηρεάζεται στο σώμα της επανάληψης (**εντολή αύξησης μεταβλητής**)

- Σχηματικά:

```
i=0;                    <- Εντολή Αρχικοποίησης (την γράφουμε εμείς)
do{
    (εντολή ή εντολές)
    i=i+1;               <- Εντολή αύξησης μεταβλητής (την γράφουμε εμείς)
} while (i<=3);          <- Συνθήκη
```

3. Η δομή do…while

4. Αμυντικός Προγραμματισμός

- **Αν ξέρουμε πόσες φορές θέλουμε να τρέξει η επανάληψη, είναι προτιμότερο να χρησιμοποιήσουμε for**, γιατί στο συντακτικό της γράφουμε τις εντολές αρχικοποιήσης και αύξησης, και έτσι ελαχιστοποιείται η πιθανότητα λάθους.

- Ωστόσο **η δομή do..while** είναι πολύ χρήσιμη **όταν θέλουμε να διαβάσουμε μεταβλητές που να έχουν συγκεκριμένες τιμές.**
  - Π.χ. Αν θέλουμε να διαβάσουμε έναν ακέραιο μεταξύ 1 και 100 και πρέπει να αποφύγουμε ο χρήστης να εισάγει μία λανθασμένη τιμή τότε εφαρμόζουμε **αμυντικό προγραματισμό** και κάνουμε έλεγχο αν η τιμή που εισήγαγε ο χρήστης είναι σωστή.
  - Έτσι τον αναγκάζουμε να επαναπληκτρολογήσει μέχρι να βάλει την σωστή τιμή.
  - Ο αμυντικός προγραμματισμός είναι καλή προγραμματιστική τακτική σε προγράμματα που ζητάμε είσοδο από τον χρήστη.

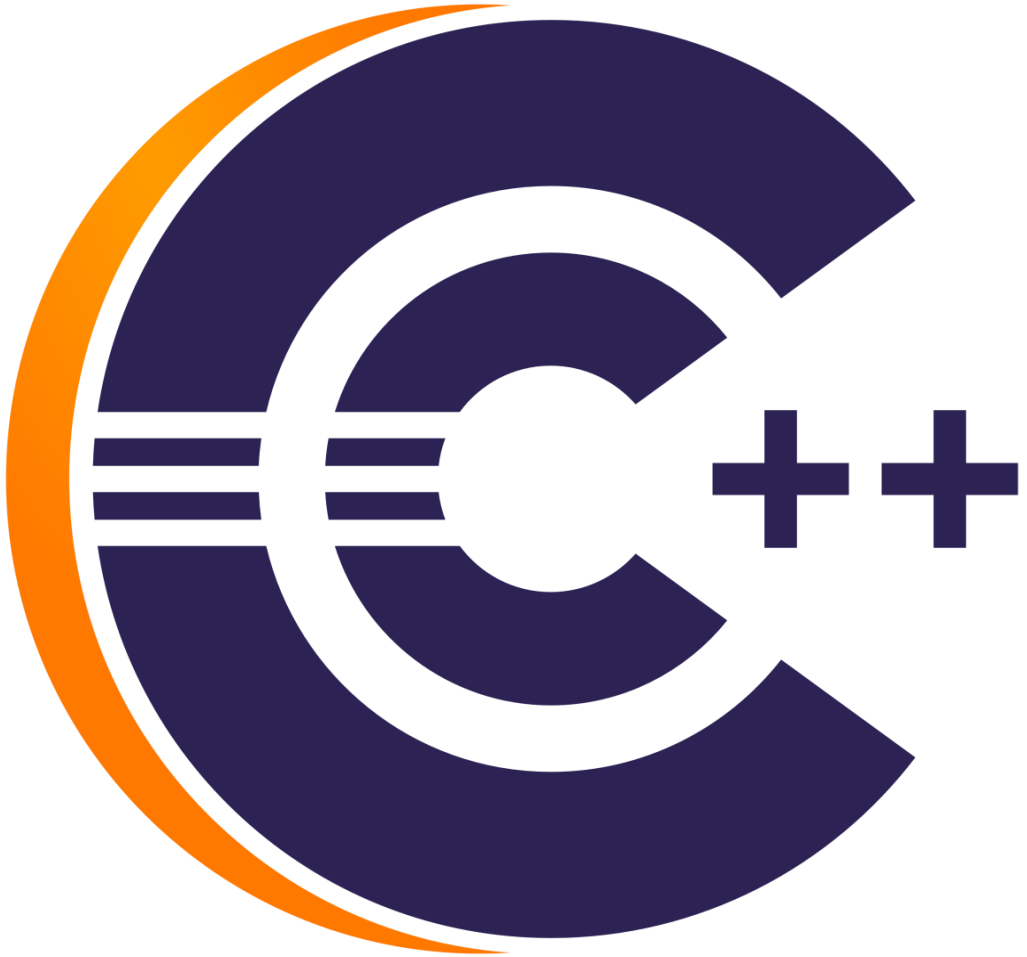
- Στην επόμενη διαφάνεια φαίνεται πως θα κάνουμε αμυντικό προγραμματισμό για να διαβάσουμε έναν αριθμό από 1 εώς 100.

3. Η δομή do…while

4. Αμυντικός Προγραμματισμός

```
/* defensive.c: Amintikos Programmatismos gia to diavasma enos akeraioy
*/

#include <stdio.h>

main()
{
   int i;

   do {
      printf("Dwste enan akeraio apo 1 ews 100: ");
      scanf("%d",&i);
   } while (i<1 || i>100);

   printf("Eisagate arithmo mesa sta oria 1 ews 100: %d",i);

}
```

ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ while



4. Η δομή while

1. Συντακτικό της while

- Το συντακτικό της δομής while (επανέλαβε) είναι ίδια με την do..while με την μόνη διαφορά ότι ο έλεγχος γίνεται στην αρχή της επανάληψης και όχι στο τέλος της επανάληψης :

```
while(Συνθήκη)
{
    (Εντολή ή Εντολές)
}
```

- Και εδώ αν έχουμε μόνο μία εντολή μπορούμε να παραλείψουμε τα άγκιστρα!

- Με το παραπάνω συντακτικό γίνονται οι εξής ενέργειες:
  - 1. Ελέγχεται η συνθήκη: (την οποία έχουμε συντάξει με λογικούς και σχεσιακούς τελεστές)
    - 1.1 Αν η απάντηση είναι ΑΛΗΘΗΣ (ισχύει η συνθήκη), τότε ξαναρχίζουμε από την αρχή (πρώτη εντολή μετά από το do)
    - 1.2 Αν η απάντηση είναι ΨΕΥΔΗΣ (δεν ισχύει η συνθήκη), τότε βγαίνουμε από την επανάληψη και τρέχουμε την αμέσως επόμενη εντολή μετά τη while.

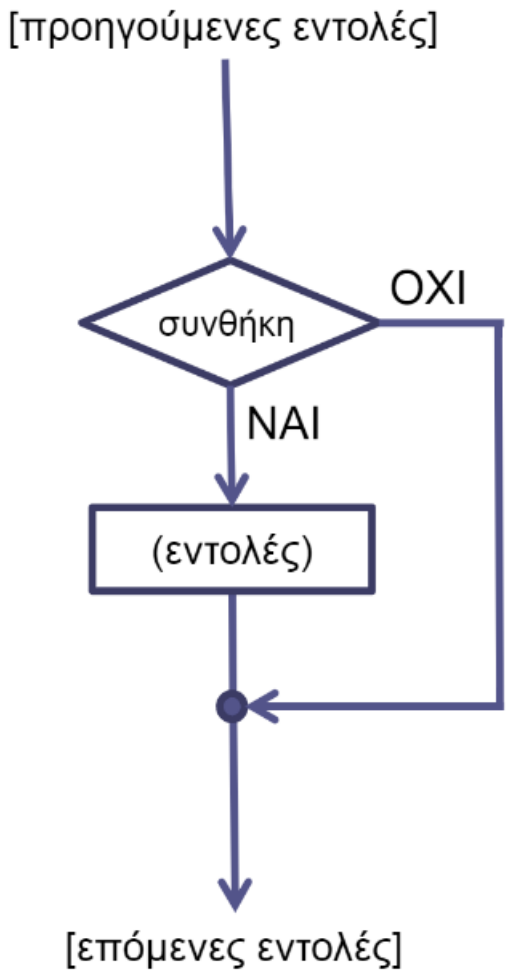
4. Η δομή while

2. Διάγραμμα Ροής Προγράμματος

- Και το διάγραμμα ροής προγράμματος:

```
[προηγούμενες εντολές]

while(Συνθήκη)
{
    (Εντολές)
}


[επόμενες εντολές]
```

- Παρατηρήστε ότι η συνθήκη ελέγχεται πριν εκτελεστούν οι εντολές



4. Η δομή while

3. Παραδείγματα Εκτέλεσης

- Ένα πρώτο παράδειγμα, για την αντιστοιχία με τις άλλες δομές επανάληψης
- Παράδειγμα 1:
  - Τι κάνει το ακόλουθο τμήμα κώδικα (οι `k, l` είναι ακέραιες μεταβλητές);

```
k=5;
while(k<8)
{
   l=2*k+1;
   k=k+1;
   printf("%d",l);
}
```

  - Απάντηση:
    - Αρχικοποιείται το `k` με 5
    - Γίνεται ο έλεγχος συνθήκης (`5<8`). Είναι ΑΛΗΘΕΣ, άρα γίνονται τα βηματα `l=2*5+1=11` και `k=5+1=6`. Τυπώνεται «`11`».
    - Γίνεται ο έλεγχος συνθήκης (`6<8`). Είναι ΑΛΗΘΕΣ, άρα γίνονται τα βηματα `l=2*6+1=13` και `k=6+1=7`. Τυπώνεται «`13`».
    - Γίνεται ο έλεγχος συνθήκης (`7<8`). Είναι ΑΛΗΘΕΣ, άρα γίνονται τα βηματα `l=2*7+1=15` και `k=7+1=8`. Τυπώνεται «`15`».
    - Γίνεται ο έλεγχος συνθήκης (`8<8`). Είναι ΨΕΥΔΕΣ, άρα τερματίζει η επανάληψη.

5. Συμπεράσματα

1. Προτεινόμενη Χρήση των Δομών Επανάληψης

- Βλέπουμε ότι και οι 3 εντολές επανάληψης με παρόμοιο τρόπο κάνουν τις ίδιες ενέργειες.
  - Η πιο συνηθισμένη δομή είναι η **for** και την χρησιμοποιούμε όταν ξέρουμε ποιες τιμές θα πάρει η μεταβλητή.
  - Αν δεν ξέρουμε ακριβώς ποιες τιμές θα πάρει ή μεταβλητή ή πόσες φορές πρέπει να γίνει η επανάληψη, τότε χρησιμοποιούμε την δομή **while**.
  - Η δομή **do…while** χρησιμοποιείται πιο σπάνια, κυρίως για αμυντικό προγραμματισμό σε διάβασμα μεταβλητών.

5. Συμπεράσματα

2. Προσομοίωση της for από την while και την do..while

- Ενδιαφέρον επίσης έχει ότι η εντολή for μπορεί να προσομοιωθεί από τις άλλες δύο ως ακολούθως:

```
for (i=1; i<=10; i++)
{
    (Εντολή ή εντολές)
}
```

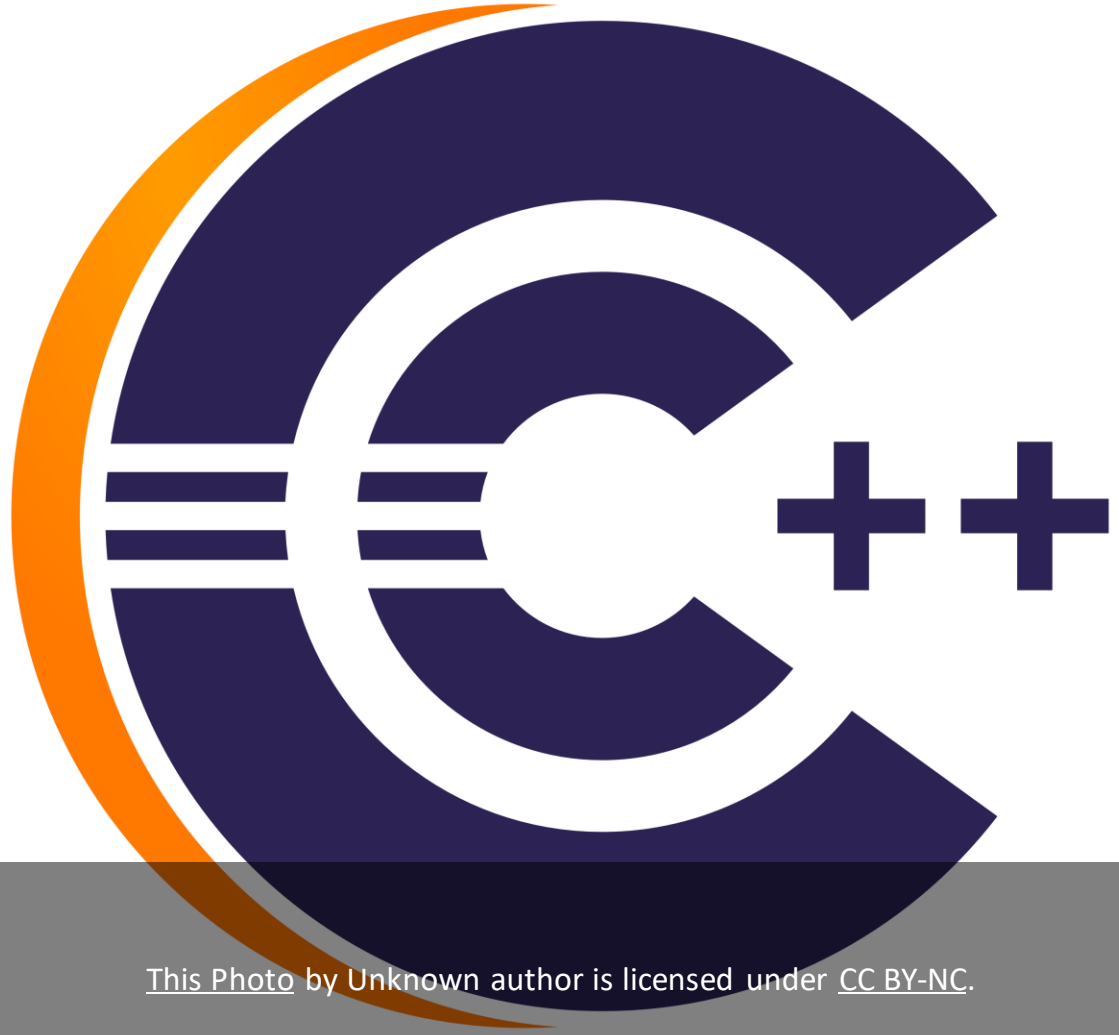
- Με εντολή while

```
i=1;
while(i<=10)
{
    (Εντολή ή εντολές)
    i=i+1;
}
```

- Με εντολή do…while

```
i=1;
do{
    (Εντολή ή εντολές)
    i=i+1
}while(I<=10);
```

ΕΦΑΡΜΟΓΕΣ



ΕΦΑΡΜΟΓΗ 1.

- (α) Τι κάνει το πρόγραμμα efarmogi1.c;
- (β) Πως μπορούμε να τροποποιήσουμε το πρόγραμμα μας , έτσι ώστε να προστίθενται 10 αριθμοί αντί για 3.
- (γ) Γράψε το πρόγραμμα ask1_ginomeno.c που θα υπολογίζει το γινόμενο των 3 αριθμών που διαβάζει από την είδοσδο.

ΕΦΑΡΜΟΓΗ 2.

Μεταγλώττισε , εκτέλεσε , και μελέτησε το πρόγραμμα efarmogi2.c

ΣΗΜΕΙΩΣΗ : Επειδή έχουμε επανάληψη μέσα στην επανάληψη , η παραπάνω δομή χαρακτηρίζεται πραγραμματιστικά **<<εμωλιασμένοι βρόχοι>> (nested loops)**

ΕΦΑΡΜΟΓΗ 3.

- (α) Χωρίς να εκτελέσεις το πρόγραμμα efarmogi3.c υπολόγισε πόσα Χ θα εκτυπώσει το πρόγραμμα και με ποια μορφή.
- (β) Εκτέλεσε το πρόγραμμα και τροποποίησε το κατά βούληση το παραλληλόγραμμο που εκτυπώνεται.
- (γ) Μετάτρεψε το πρόγραμμα σε ενα ισοδύναμο που χρησιμοποιεί την εντολή while , αντι για την εντολή for.

ΕΦΑΡΜΟΓΗ 4.

- (α) Χωρίς να εκτελέσεις το πρόγραμμα efarmogi4.c υπολόγισε πόσα Χ θα εκτυπώσει το πρόγραμμα και με ποια μορφή.
- (β) Εκτέλεσε το πρόγραμμα και τροποποίησε το κατά βούληση το "Τρίγωνο " που εκτυπώνεται.
- (γ) Μετάτρεψε το πρόγραμμα σε ενα ισοδύναμο που χρησιμοποιεί την εντολή do...while , αντι για την εντολή for.

ΕΦΑΡΜΟΓΗ 5.

Κατασκεύασε πρόγραμμα που :

- Προτρέπει τον χρήστη να εισάγει 10 ακεραίους αριθμούς και τους αποθηκεύει σε εναν μονοδιάστατο πίνακα 10 θέσεων.
- Έπειτα υπολογίζει το άθροισμα τους και το τυπώνει στην οθόνη.

ΕΦΑΡΜΟΓΗ 6.

Κατασκεύασε πρόγραμμα που :

- Προτρέπει τον χρήστη να εισάγει 5 ακεραίους αριθμούς και τους αποθηκεύει σε εναν μονοδιάστατο πίνακα 5 θέσεων. Επίσης να εφαρμόζεται αμυντικός προγραμματισμός , έτσι ώστε κάθε ακέραιος που εισάγει ο χρήστης να έχει τιμή απο 1 εως 8 .
- Έπειτα υπολογίζει το γινόμενο τους και το τυπώνει στην οθόνη.

ΕΦΑΡΜΟΓΗ 7.

Κατασκεύασε πρόγραμμα που :

- Προτρέπει τον χρήστη να εισάγει εναν αριθμό Ν. Ο αριθμός Ν να είναι μεταξύ 1 και 20 με εφαρμογή αμυντικού προγραμματισμού.
- Έπειτα να διαβάζει από την είσοδο και να εισάγει Ν αριθμούς σε εναν μονοδιάστατο πίνακα.
- Τέλος , να βρίσκει και να τυπώνει το ελάχιστο από τους Ν αριθμούς.

ΥΠΟΔΕΙΞΗ : Όταν δεν ξέρουμε εκ των προτέρων το μέγεθος του πίνακα που θα χρησιμοποιήσουμε , δεσμεύουμε προκαταβολικά τον μέγιστο χώρο που μπορεί το πρόγραμμα να χρησιμοποιήσει.

- **Για παράδειγμα σε αυτή την άσκηση , θα πρέπει να κατασκευάσεις εναν πίνακα 20 θέσεων , παρόλο που το πρόγραμμα θα χρησιμοποιήσει τόσες θέσεις , όσο το Ν που θα εισάγει ο χρήστης.**
- **Σε επόμενο μάθημα θα μάθουμε πως δεσμεύεται ο χώρος στην μνήμη δυναμικά. Δηλαδή κατά τον χρόνο εκτέλεσης να δεσμεύεται ο χώρος μνήμης που απαιτείται.**

ΕΦΑΡΜΟΓΗ 8.

Κατασκεύασε πρόγραμμα που :

- Προτρέπει τον χρήστη να εισάγει εναν αριθμό Ν. Ο αριθμός Ν να είναι μεταξύ 1 και 20 με εφαρμογή αμυντικού προγραμματισμού.
- Έπειτα να διαβάζει από την είσοδο και να εισάγει Ν αριθμούς σε εναν μονοδιάστατο πίνακα.
- Τέλος , να βρίσκει και να τυπώνει το Μ.Ο των Ν αριθμών.

ΤΕΛΟΣ ΜΑΘΗΜΑΤΟΣ

ΔΟΜΕΣ ΕΠΑΝΑΛΗΨΗΣ (for , while , do...while)

Γιώργος Διάκος - Full Stack Developer