Η ΓΛΩΣΣΑ C

Μάθημα 7:

Πίνακες

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

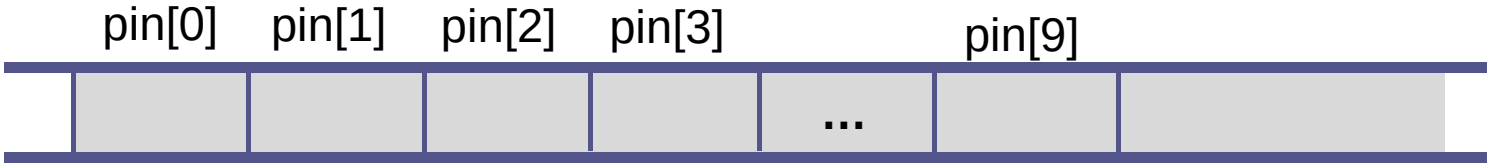
Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

1. Δήλωση Πίνακα

- Ένας πίνακας είναι μια σειρά από μεταβλητές ίδιου τύπου αποθηκευμένες στην μνήμη.

- Για παράδειγμα ένας πίνακας 10 ακεραίων με όνομα ***pin,*** είναι 10 ακέραιες μεταβλητές αποθηκευμένες στην σειρά (η μία μετά την άλλη) στην μνήμη.



- Ένας πίνακας θα δηλώνεται (στο τμήμα δήλωσης μεταβλητών) με εντολή της μορφής:

```
Τύπος_Δεδομένων ΟΝΟΜΑ_ΠΙΝΑΚΑ[ΠΛΗΘΟΣ];
```

- Όπου:
  - Τύπος_Δεδομένων: Ο τύπος δεδομένων των μεταβλητών του πίνακα.
  - ΟΝΟΜΑ_ΠΙΝΑΚΑ: Το όνομα που επιλέγουμε εμείς για τον πίνακα.
  - ΠΛΗΘΟΣ (προσέξτε ότι είναι μέσα σε αγκύλες): Πόσες μεταβλητές θα περιέχει ο πίνακας

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

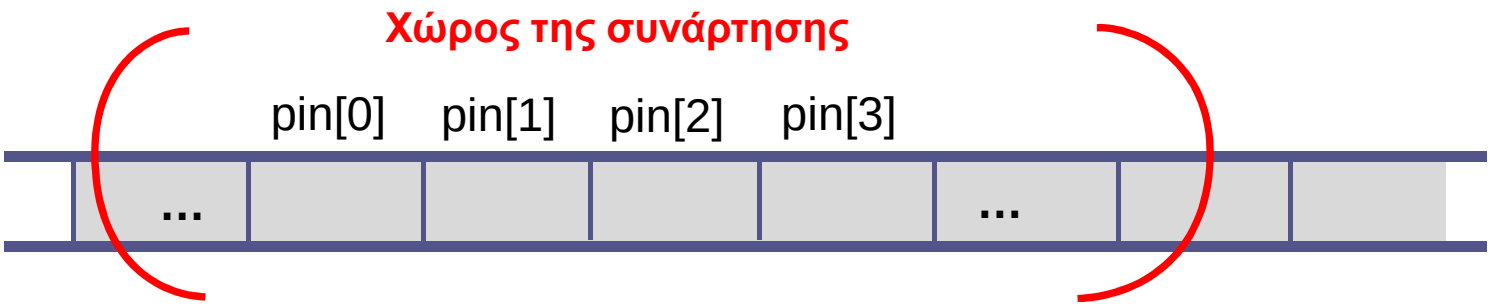
1. Δήλωση Πίνακα

- Για παράδειγμα με την δήλωση:

```
int pin[4];
```

  - Δηλώνουμε έναν πίνακα με 4 ακέραιες μεταβλητές.
  - Με την εντολή δήλωσης, δεσμεύεται στην μνήμη χώρος για τον πίνακα και είναι σημαντικό ότι οι θέσεις αυτές είναι σε μια σειρά.

- Τα ονόματα των 4 μεταβλητών που κατασκευάσαμε είναι:
  - pin[0],pin[1],pin[2],pin[3]
    - Παρατηρούμε ότι η αρίθμηση ξεκινά από το 0 εως και ένα λιγότερο από τον αριθμό που δηλώσαμε.
    - Άρα μετά την εντολή δήλωσης, έχουν δεσμευτεί 4 διαδοχικές θέσεις στην μνήμη για να αποθηκεύσουν τιμές σε αυτές τις μεταβλητές, στον χώρο της συνάρτησης που έχει δηλωθεί ο πίνακας:



Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

2. Παράδειγμα Χρήσης Πίνακα

- Πλέον μετά την δήλωση έχουμε στα χέρια μας τις μεταβλητές και μπορούμε να τις χρησιμοποιήσουμε.
- Δείτε για παράδειγμα το εξής απλό πρόγραμμα:
- Οι μεταβλητές διαχειρίζονται όπως οι τυπικές ακέραιες μεταβλητές

```
/* array.c: Aplo programma me pinaka */

#include <stdio.h>

int main()
{
   int pin[3];
   int sum;

   pin[0]=1;
   pin[1]=3;
   pin[2]=4;
   sum=pin[0]+pin[1]+pin[2];
   printf("\n%d+%d+%d=%d",pin[0],pin[1],pin[2],sum);
}
```

Προσοχή, μην γράψουμε αριθμό για όριο του πίνακα που είναι εκτός του επιτρεπτού.

Π.χ. Αν έχουμε δήλωση `int pin[4];` και γράψουμε για όνομα μεταβλητής `pin[5]` θα έχουμε σφάλμα όταν θα εκτελέσουμε το πρόγραμμα!

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

3. Αρχικοποίηση Πίνακα κατά την Δήλωση

- Όπως όταν έχουμε μια μεταβλητή, μπορούμε να την αρχικοποιήσουμε με μια τιμή όταν την δηλώνουμε με μία εντολή της μορφής:

```
int x=5;
```

- Το ίδιο ισχύει και για τους πίνακες. Έχουμε το δικαίωμα να αρχικοποιήσουμε τον πίνακα με μια εντολή ως εξής:

```
int pinakas[5] = {0,4,9,2,1};
```

- Που αντιστοιχεί στην δήλωση και καταχώρηση των μεταβλητών:

```
int pinakas[5];

pinakas[0]=0;
pinakas[1]=4;
pinakas[2]=9;
pinakas[3]=2;
pinakas[4]=1;
```

- Θέλει προγραμματιστική προσοχή! Διότι θα πρέπει να έχουμε ακριβώς τόσες τιμές όσες και οι θέσεις του πίνακα.
- Αν βάλουμε λιγότερες ή περισσότερες τιμές μέσα στα άγκιστρα, τότε πιθανόν το πρόγραμμα μας να έχει μη αναμενόμενη συμπεριφορά!

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

3. Αρχικοποιήση Πίνακα κατά την Δήλωση

Λεπτομέρειες του μεταγλωττιστή:
Α) Αν δεν δηλώσουμε διάσταση κατά την αρχικοποίηση τότε ο μεταγλωττιστής αρχικοποιεί τη διάσταση στο πλήθος των στοιχείων του πίνακα. Έτσι οι δύο ακόλουθες δηλώσεις είναι ισοδύναμες:

```
int pin[4]={2,4,6,8};
```

↕

```
int pin[]={2,4,6,8};
```

Λεπτομέρειες του μεταγλωττιστή:
Β) Ωστόσο αν δηλώσουμε μεγαλύτερη διάσταση στον πίνακα, μπορούμε να αρχικοποίησουμε με λιγότερα στοιχεία τον πίνακα. Θα δεσμευτεί χώρος για τα επόμενα στοιχεία, αλλά δεν μπορούμε να είμαστε σίγουροι για την τιμή που θα έχουν
Για παράδειγμα η ακόλουθη δήλωση είναι έγκυρη:

```
int pin[10]={2,4,6,8};
```

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

4. Στατική Δέσμευση Μνήμης Πίνακα

- Στην πράξη χρησιμοποιούνται 4 τρόποι για την δήλωση της διάστασης του πίνακα. Η επιλογή του τρόπου εξαρτάται από το πρόγραμμα που γράφουμε.

Α' τρόπος: Δηλώνουμε την διάσταση «καρφωτά» μέσω ενός αριθμού.

- Π.χ. στο τμήμα δήλωσης των δεδομένων γράφουμε:

```
{
   int pinakas[100]; //Εδώ δηλώνουμε τη διάσταση του πίνακα
   int i;

   for (i=0; i<100; i++)
   {
       /*Κάνε κάποια πράγματα στον πίνακα*/
   }
}
```

- Ο τρόπος αυτός είναι αντιαισθητικός και δείχνει μικρή προγραμματιστική εμπειρία. Π.χ. αν χρειαστεί να τροποποιήσουμε το πρόγραμμα μας για να τρέχει για 1000 ακεραίους, τότε πρέπει να βρούμε όλες τις εμφανίσεις του 100 και να το αλλάξουμε σε 1000.
- Αυτόν τον τρόπο θα τον εφαρμόζουμε μόνο όταν θέλουμε να «τσεκάρουμε» έναν πολύ μικρό κώδικα.

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

4. Στατική Δέσμευση Μνήμης Πίνακα

Β' τρόπος: Δηλώνουμε την διάσταση μέσω μίας συμβολικής σταθεράς.

- Παράδειγμα:

```
#define N 100 // Εδώ δηλώνουμε το μέγεθος του πίνακα

main()
{
   int pinakas[N]; //Δήλωση του πίνακα
   int i;

   for (i=0; i<N; i++)
   {
       /*Κανε κάποια πράγματα στον πίνακα*/
   }
}
```

- Ο τρόπος αυτός είναι ισοδύναμος με τον προηγούμενο και έχει το πλεονέκτημα ότι η διάσταση του πίνακα δηλώνεται μόνο μία φορά μέσω της συμβολικής σταθεράς.
- Σε περίπτωση που θέλουμε να αλλάξουμε τη διάσταση του πίνακα αρκεί να αλλάξουμε την τιμή στη συμβολική σταθερά.

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

4. Στατική Δέσμευση Μνήμης Πίνακα

Γ' τρόπος: Δηλώνουμε τη μέγιστη διάσταση του πίνακα μέσω συμβολικής σταθεράς, και έπειτα έχουμε μία μεταβλητή για το πλήθος των θέσεων που χρησιμοποιούμε

- Είναι πολύ συχνό στα προγράμματα μας, να μην ξέρουμε πόσες ακριβώς θέσεις θα χρειαστούμε στον πίνακα.
- Στις περιπτώσεις αυτές, κάνουμε μια εκτίμηση του πόσες θέσεις (ένα άνω όριο) θα χρειαστούμε και δηλώνουμε τον πίνακα με αυτό το όριο.
- Στην συνέχεια έχουμε μια μεταβλητή που χρησιμοποιούμε ως άνω όριο του πίνακα, ανάλογα με το πόσες θέσεις χρειαζόμαστε πραγματικά.

```
#define SIZE 1000 // Εδώ δηλώνουμε το μέγιστο μέγεθος του πίνακα
main()
{
   int pinakas[SIZE];
   int N=100;
   int i;

   for (i=0; i<N; i++)
   {
       /*Κανε κάποια πράγματα στον πίνακα*/
   }
}
```

Α. Πίνακες

1. Μονοδιάστατοι Πίνακες

4. Στατική Δέσμευση Μνήμης Πίνακα

- Λίγα παραπάνω πράγματα για τους μονοδιάστατους πίνακες...
  - Αντιλαμβανόμαστε ότι υπάρχει ένα πρόβλημα με τους πίνακες και το μέγεθος που θέλουμε να έχουν στο πρόγραμμα μας.
    - Τα πράγματα είναι καλά, όταν ξέρουμε εκ των προτέρων πόσα στοιχεία θα έχει. Τότε τρόποι δήλωσης σαν αυτούς που είδαμε είναι μια χαρά! Αυτοί ο τρόποι δήλωσης του πίνακα αναφέρονται σαν **στατική δέσμευση μνήμης**.
    - Αντίθετα αν οι τρόποι που αναφέραμε δεν μπορούν να εξυπηρετήσουν το πρόγραμμα μας (φανταστείτε για παράδειγμα, ένα πρόγραμμα που το μέγεθος του πίνακα εξαρτάται από την είσοδο του χρήστη σε πολύ μεγάλο βαθμό. Δηλαδή την μία φορά πρέπει να τρέχει για 10 ακέραιους και την άλλη για 1.000.000 ακεραίους). Εκεί είναι κατάχρηση να δηλώνουμε τόσο μεγάλο πίνακα για πιθανόν τόσα λίγα δεδομένα. Γι' αυτόν τον λόγο θα δούμε και έναν δεύτερο τρόπο να δεσμεύουμε χώρο στην μνήμη για έναν πίνακα, στο μάθημα 9 και καλείται **δυναμική δέσμευση μνήμης**.

Δ' τρόπος: Ζητάμε από τον χρήστη να μας εισάγει κατά τον χρόνο εκτέλεσης την διάσταση του πίνακα και δεσμεύουμε δυναμικά τον χώρο στην μνήμη (δυναμική δέσμευση μνήμης)

Α. Πίνακες

2. Διδιάστατοι Πίνακες

1. Δήλωση Πίνακα

- Ένας διδιάστατος πίνακας (πίνακας 2 διαστάσεων) είναι και πάλι ένας αποθηκευτικός χώρος στην μνήμη, όπου έχουμε δύο διαστάσεις: Μία για τις γραμμές και μία για τις στήλες.
- Έτσι θα έχουμε το δικαίωμα να ορίσουμε π.χ. έναν πίνακα ακεραίων 6x4, δηλαδή έναν πίνακα ακεραίων με 6 γραμμές και 4 στήλες
- Θα έχουμε τα εξής στοιχεία στον πίνακα (προσέξτε και πάλι ότι η αρίθμηση των γραμμών και των στηλών ξεκινάει από το 0). Όνομα του πίνακα στο παράδειγμα είναι Α:

	Στήλη 0	Στήλη 1	Στήλη 2	Στήλη 3
Γραμμή 0	`A[0][0]`	`A[1][1]`	`A[0][2]`	`A[0][3]`
Γραμμή 1	`A[1][0]`	`A[1][1]`	`A[1][2]`	`A[1][3]`
Γραμμή 2	`A[2][0]`	`A[2][1]`	`A[2][2]`	`A[2][3]`
Γραμμή 3	`A[3][0]`	`A[3][1]`	`A[3][2]`	`A[3][3]`
Γραμμή 4	`A[4][0]`	`A[4][1]`	`A[4][2]`	`A[4][3]`
Γραμμή 5	`A[5][0]`	`A[5][1]`	`A[5][2]`	`A[5][3]`

Α. Πίνακες

2. Διδιάστατοι Πίνακες

1. Δήλωση Πίνακα

- Ένας διδιάστατος πίνακας θα δηλώνεται (στο τμήμα δήλωσης μεταβλητών) με εντολή της μορφής:

```
Τύπος_Δεδομένων ΟΝΟΜΑ_ΠΙΝΑΚΑ[ΠΛΗΘΟΣ-ΓΡΑΜΜΩΝ][ΠΛΗΘΟΣ-ΣΤΗΛΩΝ];
```

- Όπου:
  - Τύπος_Δεδομένων: Ο τύπος δεδομένων των μεταβλητών του πίνακα.
  - ΟΝΟΜΑ_ΠΙΝΑΚΑ: Το όνομα που επιλέγουμε εμείς για τον πίνακα.
  - ΠΛΗΘΟΣ-ΓΡΑΜΜΩΝ, ΠΛΗΘΟΣ-ΣΤΗΛΩΝ (προσέξτε ότι είναι μέσα σε αγκύλες πρώτα οι γραμμές και έπειτα οι στήλες): Πόσες γραμμές και στήλες θα περιέχει ο πίνακας.
    - Το πλήθος των μεταβλητών που θα οριστούν θα είναι ίσο με το γινόμενο του πλήθος των γραμμών με το πλήθος των στηλών.
- Έτσι ο πίνακας που είδαμε προηγουμένως θα δηλώνεται με μία εντολή δήλωσης της μορφής:

```
int A[6][4];
```

- Ενώ για παράδειγμα ένας 8x10 πίνακας μεταβλητών double θα δηλώνεται με την εντολή:

```
double A[8][10];
```

Α. Πίνακες

2. Διδιάστατοι Πίνακες

2. Αρχικοποίηση κατά τη Δήλωση

- Αντίστοιχα με τους μονοδιάστατους πίνακες και οι διδιάστατοι πίνακες μπορούν να αρχικοποιηθούν με μια εντολή δήλωσης της μορφής:

```
int pinakas[2][5] = {{0,4,9,2,1},{4,3,2,5,2}};
```

  - Όπου φαίνεται ότι γράφουμε τα περιεχόμενα κάθε διαδοχικής γραμμής μέσα σε άγκιστρα.
  - ...ενώ οι διαδοχικές γραμμές του πίνακα χωρίζονται με κόμματα και βρίσκονται σε ένα μεγάλο ζεύγος άγκιστρων.
- Υπάρχει και μια πιο εκκεντρική μορφή δήλωσης:

```
int pinakas[2][5] = {0,4,9,2,1,4,3,2,5,2};
```

  - Όπου η αντιστοίχιση είναι νοητή, δηλαδή τα 5 πρώτα στοιχεία είναι για την 1η γραμμή κ.λπ.
  - Ο λόγος ύπαρξης αυτής της εντολής δήλωσης είναι η απεικόνιση διδιάστατων πινάκων στην μνήμη που θα δούμε αμέσως τώρα!

Α. Πίνακες

2. Διδιάστατοι Πίνακες

3. Απεικόνιση στη Μνήμη

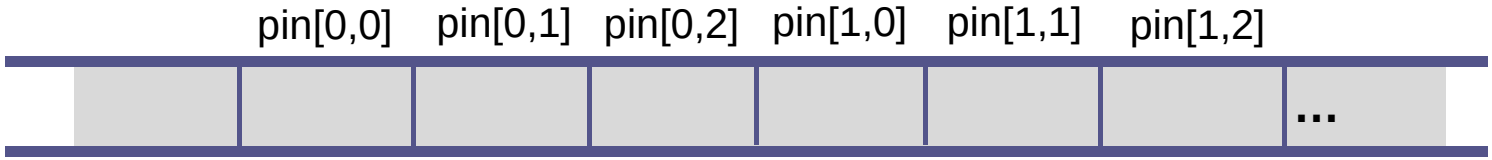
- Βλέπουμε πως δεσμεύεται η μνήμη για έναν πίνακα που δηλώνεται με την εντολή:

```
int pin[2][3];
```

  - Όπως είδαμε εδώ δηλώνουμε 3x2=6 μεταβλητές.
- Τα ονόματα των 6 μεταβλητών που δηλώσαμε είναι:

pin[0,0]	pin[0,1]	pin[0,2]
pin[1,0]	pin[1,1]	pin[1,2]

- Είναι σημαντικό ότι οι μεταβλητές δηλώνονται στον χώρο μνήμης της συνάρτησης σε διαδοχικές θέσεις γραμμή προς γραμμή:



Α. Πίνακες

2. Διδιάστατοι Πίνακες

4. Χρήση Διδιάστατων Πινάκων

- Ο πιο συνηθισμένος τρόπος για να διαχειριστούμε έναν διδιάστατο πίνακα, είναι μέσω μιας διπλής (εμφωλιασμένης) for.
- Συνήθως λοιπόν κάνουμε μια επανάληψη στις γραμμές και μία εμφωλιασμένη επανάληψη στις στήλες, προκειμένου να κάνουμε μια διαπέραση των στοιχείων του πίνακα:
- Δείτε για παράδειγμα τον ακόλουθο κώδικα που δηλώνει έναν 10x5 πίνακα και αρχικοποιεί τα στοιχεία του σε 0.

```
#define M 10
#define N 5

main()
{
   int pin[M][N];
   int i,j; //Ενας μετρητής για τις γραμμές και ένας για τις στήλες

   for (i=0; i<M; i++)
       for (j=0; j<N; j++)
           pin[i][j]=0;
}
```

Οι παρατηρήσεις που κάναμε περί στατικής δέσμευσης μνήμης ισχύουν και για τους διδιάστατους πίνακες!

Α. Πίνακες
3. Πολυδιάστατοι Πίνακες

- Αντίστοιχα ένας τριδιάστατος πίνακας (πίνακας 3 διαστάσεων) είναι και πάλι ένας αποθηκευτικός χώρος στην μνήμη, όπου έχουμε τρεις διαστάσεις
- Έτσι ένας 5x2x8 πίνακας θα δηλώνεται με μία εντολή της μορφής:

```
int A[5][2][8];
```

  - Ο οποίος θα περιέχει 5x2x8=80 ακέραιες μεταβλητές.
  - Οι ίδιες παρατηρήσεις που κάναμε για τους μονοδιάστατους και τους διδιάστατους πίνακες ισχύουν βεβαίως και εδώ.
- Αντίστοιχα μπορούμε να ορίσουμε πίνακες πολλών διαστάσεων (δεν υπάρχει πρακτικά περιορισμός στις διαστάσεις που μπορούμε να χρησιμοποιήσουμε). Π.χ. Ένας 5-διάστατος πίνακας 2x4x9x2x4 δηλώνεται με μία εντολή της μορφής:

```
int A[2][4][9][2][4];
```

  - Τέτοιοι πίνακες είναι κυρίως για εξειδικευμένες εφαρμογές και δεν θα μας απασχολήσουν στους εκπαιδευτικούς στόχους των μαθημάτων.

Α. Πίνακες
4. Πίνακες και Συναρτήσεις

- Ένας πίνακας μπορεί να διοχετευτεί ως όρισμα σε μία συνάρτηση με τον εξής τρόπο:

```
#define N 100

void function(int pin[]);    // Πρωτότυπο συνάρτησης

main()
{
    int array[N];

    function(array);         // Διοχέτευση πίνακα ως όρισμα στη συνάρτηση
}

void function(int pin[])     // Σώμα Συνάρτησης
{
    ...ενέργειες στον pin[0...99]
}
```

- Ωστόσο δεν θα επεκταθούμε περαιτέρω! Σε επόμενο μάθημα θα δούμε έναν εναλλακτικό τρόπο διοχέτευσης πινάκων σε συναρτήσεις, με χρήση δεικτών τον οποίο και θα επιλέξουμε να εφαρμόζουμε στην πράξη.

Β. Τυχαίοι Αριθμοί
1. Η συνάρτηση rand()

- Εδώ θα κάνουμε μια σύντομη αναφορά σε δύο συναρτήσεις βιβλιοθήκης της C που μας επιτρέπουν να κάνουμε παραγωγή τυχαίων αριθμών.
- Η συνάρτηση rand() επιστρέφει τυχαίους αριθμούς. Το πρωτότυπό της είναι:

```
unsigned int rand()
```

  - Η οποία επιστρέφει έναν τυχαίο αριθμό στο διάστημα 0...MAX_INT όπου MAX_INT ο μέγιστος μη προσημασμένος ακέραιος που μπορεί να απεικονίσει το σύστημά μας (συνήθως το 2^{32}-1)

- Είναι ορισμένη στην βιβλιοθήκη συναρτήσεων `stdlib.h`

- Ας δούμε δύο τρόπους να την χρησιμοποιήσουμε:

```
x=rand()%100
```

  - Στο x θα αποθηκευτεί ένας αριθμός από το 0 έως το 99

```
x=10+rand()%91
```

  - Στο x θα αποθηκευτεί ένας αριθμός από το 10 έως το 100

Β. Τυχαίοι Αριθμοί
2. Η συνάρτηση srand()

- Ο τρόπος λειτουργίας της rand() στηρίζεται σε κάποια περίπλοκα μαθηματικά και κάθε φορά παράγει την ίδια ακολουθία τυχαίων αριθμών.
  - Για τον λόγο αυτό, λέμε ότι παράγει ψευδοτυχαίους αριθμούς.
- Προκειμένου κάθε φορά που τρέχουμε το πρόγραμμα να παράγει άλλη ακολουθία τυχαίων αριθμών, πρέπει αρχικά στο πρόγραμμά μας να τρέξουμε την συνάρτηση:

```
void srand(int seed)
```

  - Που έχει δηλωθεί στη βιβλιοθήκη:

```
stdlib.h
```

- Ο πιο συνηθισμένος τρόπος χρήσης είναι στην αρχή του προγράμματός μας να γράψουμε την εντολή:

```
srand(time(NULL));
```

  - Έτσι ώστε να αρχικοποιηθεί η ακολουθία τυχαίων αριθμών με μία παράμετρο που εξαρτάται από την τρέχουσα ώρα. Θα απαιτηθεί να ενσωματώσουμε και το αρχείο κεφαλίδας `time.h` στο οποίο έχει οριστεί η συνάρτηση time() που χρησιμοποιεί το πρόγραμμά μας.

Β. Τυχαίοι Αριθμοί
3. Παράδειγμα παραγωγής τυχαίων αριθμών

- Το παρακάτω πρόγραμμα τυπώνει στην οθόνη 10 τυχαίους αριθμούς στο διάστημα 0...99

```
/* random.c: Typwnei 10 tyxaioys akeraioys sto diastima 0..99 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

main()
{
   int x,i;

   srand(time(NULL));

   for (i=0; i<10; i++)
   {
       x=rand()%100;
       printf("\n%d",x);
   }
}
```

Γ. Ασκήσεις

1. Αρχικοποίηση και Εκτύπωση Μονοδιάστατου Πίνακα

Γράψτε ένα πρόγραμμα το οποίο:

1. Να δηλώνει έναν πίνακα 100 ακεραίων (SIZE)
2. Ναι διαβάζει από τον χρήστη έναν αριθμό Ν από το 20 έως το 100 (που θα δηλώνει το πλήθος των στοιχείων του πίνακα που θέλει να χρησιμοποιήσει). Να πραγματοποιηθεί έλεγχος ότι ο χρήστης πληκτρολόγησε τιμή από το 20 έως το 100.
3. Να αρχικοποιηθεί ο πίνακας με τυχαίους αριθμούς
4. Να προβάλλει τα στοιχεία του πίνακα στην οθόνη με έναν κομψό τρόπο.

Γ. Ασκήσεις
2. Εκτύπωση Διδιάστατων Πινάκων

Γράψτε ένα πρόγραμμα το οποίο:

1. Να δηλώνει έναν διδιάστατο πίνακα 5x8
2. Να αρχικοποιεί τις τιμές του πίνακα, αναθέτοντας σε κάθε θέση έναν τυχαίο αριθμό από 0 έως 200
3. Να προβάλλει τα στοιχεία του πίνακα στην οθόνη με έναν κομψό τρόπο (σε κάθε γραμμή της οθόνης να είναι και μία γραμμή του πίνακα)

Γ. Ασκήσεις

3. Στατική Δέσμευση Μνήμης σε Διδιάστατο Πίνακα

Γράψτε ένα πρόγραμμα σε γλώσσα C που:

1. Να δηλώνει έναν διδιαστατο πίνακα 100x200 ακεραίων (SIZE1xSIZE2)
2. Ναι διαβάζει από τον χρήστη έναν αριθμό M από το 10 έως το 100 (που θα δηλώνει το πλήθος των γραμμών του πίνακα που θέλει να χρησιμοποιήσει). Να πραγματοποιηθεί έλεγχος ότι ο χρήστης πληκτρολόγησε τιμή από το 10 έως το 100.
3. Ναι διαβάζει από τον χρήστη έναν αριθμό N από το 10 έως το 100 (που θα δηλώνει το πλήθος των στηλών του πίνακα που θέλει να χρησιμοποιήσει). Να πραγματοποιηθεί έλεγχος ότι ο χρήστης πληκτρολόγησε τιμή από το 10 έως το 100.
4. Να αρχικοποιηθεί ο πίνακας με τυχαίους αριθμούς
5. Να προβάλλει τα στοιχεία του πίνακα στην οθόνη με έναν κομψό τρόπο.

Γ. Ασκήσεις

4. Κατασκευή Παιχνιδιού!

Γράψτε ένα πρόγραμμα σε γλώσσα C που:

1. Θα αρχικοποιεί μία μεταβλητή με όνομα hidden με έναν τυχαίο ακέραιο αριθμό από το 1 έως το 100.
2. Θα ζητεί από τον χρήστη να πληκτρολογήσει (μαντέψει) τον αριθμό
    1. Αν ο χρήστης εισάγει τον αριθμό, θα εκτυπώσει κατάλληλο μήνυμα επιβράβευσης και θα τερματίσει.
    2. Αν ο χρήστης εισάγει μικρότερο αριθμό, θα εκτυπώσει κατάλληλο μήνυμα στον χρήστη να πληκτρολογήσει μεγαλύτερο αριθμό και θα επαναλάβει από το βήμα 2
    3. Αν ο χρήστης εισάγει μεγαλύτερο αριθμό, θα εκτυπώσει κατάλληλο μήνυμα στον χρήστη να πληκτρολογήσει μικρότερο αριθμό και θα επαναλάβει από το βήμα 2

Γ. Ασκήσεις

Όπως είδαμε το συντακτικό και η χρήση των πινάκων δεν είναι ιδιαίτερα δύσκολο.

Σημαντικότερο είναι να αναπτύξουμε αλγοριθμική σκέψη, δηλαδή να μάθουμε σκεπτικά που έχουν αναπτυχθεί για την αποδοτική επίλυση προβλημάτων με τον υπολογιστή.

Συνίσταται λοιπόν στο σημείο αυτό να μελετηθούν τα ακόλουθα τέσσερα μαθήματα αλγορίθμων με τη βοήθεια της γλώσσας C:

- Αλγόριθμοι σε C – Μάθημα 1: Διαπέραση Πίνακα
- Αλγόριθμοι σε C – Μάθημα 2: Αναζήτηση σε Πίνακα
- Αλγόριθμοι σε C – Μάθημα 3: Ταξινόμηση Πίνακα
- Αλγόριθμοι σε C – Μάθημα 4: Αλγεβρικές Πράξεις Πινάκων