Η ΓΛΩΣΣΑ C

Μάθημα 12:

Δυναμική Δέσμευση Μνήμης

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Α. Στατική και Δυναμική Δέσμευση Μνήμης

1. Στατική Δέσμευση Μνήμης
2. Δυναμική Δέσμευση Μνήμης
    1. Η συνάρτηση malloc
    2. Η συνάρτηση free
    3. Δέσμευση Μεταβλητής
    4. Δέσμευση Μονοδιάστατου Πίνακα
    5. Δέσμευση Διδιάστατου Πίνακα

Β. Ασκήσεις

Α. Στατική και Δυναμική Δέσμευση Μνήμης

1. Στατική Δέσμευση Μνήμης

- Οι τρόποι που έχουμε δει για να δηλώνουμε έναν πίνακα (άρα και μία συμβολοσειρά) είναι να δηλώσουμε εκ των προτέρων το μέγεθος του.
  - Το γεγονός αυτό κάνει το πρόγραμμα μας να μην είναι πολύ ευέλικτο.
  - Είναι συχνό, να καταλαβαίνουμε κατά τον χρόνο εκτέλεσης πόσο θα θέλουμε να είναι το μέγεθος του πίνακα μας.
- Ο τρόπος δήλωσης ενός πίνακα με τον τρόπο αυτό, δεσμεύει στατικά τον χώρο μνήμης.
- Συνεπώς με τον όρο στατική δέσμευση μνήμης αναφερόμαστε στην δήλωση μιας μεταβλητής με τον συνηθισμένο τρόπο, π.χ.:

```
int x;
```

  - ή και στον συνηθισμένο τρόπο για την δήλωση ενός πίνακα, π.χ.:

```
int pinakas[10];
```

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

1. Η συνάρτηση malloc

- Η συνάρτηση malloc:

```
void *malloc(size_t size)
```

  - Δεσμεύει δυναμικά τόσα bytes όσα και το όρισμα που της διοχετεύουμε.
  - Έχει οριστεί στην βιβλιοθήκη συναρτήσεων: `stdlib.h`
  - Επιστρέφει είτε έναν δείκτη στην αρχή του χώρου μνήμης που δεσμεύθηκε δυναμικά. Είτε την τιμή NULL, αν δεν βρέθηκε κατάλληλος χώρος μνήμης.
    - Θα πρέπει πάντα να ελέγχουμε αν η malloc κατάφερε να δεσμεύσει τον χώρο και δεν επέστρεψε NULL

- Μέσω της συνάρτησης malloc, μπορούμε κατά τον χρόνο εκτέλεσης (όταν δηλαδή τρέχει το πρόγραμμα) να βρίσκουμε τον χώρο μνήμης που θα χρειαστούν οι μεταβλητές.

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

2. Η συνάρτηση free

- Η συνάρτηση free:

```
void free(void *ptr)
```

  - Απελευθερώνει τα bytes που είχαμε δεσμεύσει δυναμικά με την συνάρτηση malloc, στην αρχή των οποίων δείχνει ο δείκτης ptr.
  - Έχει οριστεί στην βιβλιοθήκη συναρτήσεων: `stdlib.h`
  - Θα πρέπει να γνωρίζουμε ότι ΠΑΝΤΑ θα πρέπει να απελευθερώνουμε όση μνήμη δεσμεύσαμε δυναμικά με την malloc.

- Θεωρείται σημαντικό προγραμματιστικό λάθος να μην αποδεσμεύουμε τον χώρο που έχουμε δεσμεύσει δυναμικά με την malloc κάνοντας κατάλληλη χρήση της εντολής free.

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

3. Δέσμευση Μεταβλητής

- Για να δεσμεύσουμε μνήμη για μια μεταβλητή, δηλώνουμε έναν δείκτη στην μεταβλητή.
- Έπειτα κάνουμε χρήση της malloc και δεσμεύουμε τόσο χώρο όσο πιάνει ο αντίστοιχος τύπος δεδομένων στην μνήμη.
  - Για να το κάνουμε αυτό χρησιμοποιούμε τον τελεστή sizeof
- Χρησιμοποιούμε κανονικά την μεταβλητή μας μέσω του δείκτη.
- Δεν ξεχνάμε ποτέ να αποδεσμεύσουμε τον χώρο που δεσμεύσαμε με την free.
- Ας δούμε ένα παράδειγμα για την χρήση με μία μεταβλητή:

```
int *p; //Εδώ δηλώνουμε τον δείκτη στην μεταβλητή.

p=malloc(sizeof(int)); //Δεσμεύουμε δυναμικά την μνήμη για τον ακέραιο

*p=4; //Εδώ μπορούμε να κάνουμε χρήση του ακεραίου

free(p); //αποδεσμεύουμε τον χώρο μνήμης
```

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

3. Δέσμευση Μεταβλητής

- Τελείως αντίστοιχα θα δουλεύαμε και για έναν float:

```
float *p; //Εδώ δηλώνουμε τον δείκτη στην μεταβλητή.

p=malloc(sizeof(float)); //Δεσμεύουμε δυναμικά την μνήμη για τον float

*p=10.4; //Εδώ μπορούμε να κάνουμε χρήση του float

free(p); //αποδεσμεύουμε τον χώρο μνήμης
```

- ή για έναν long

```
long *p; //Εδώ δηλώνουμε τον δείκτη στην μεταβλητή.

p=malloc(sizeof(long)); //Δεσμεύουμε δυναμικά την μνήμη για τον long

*p=1132; //Εδώ μπορούμε να κάνουμε χρήση του long

free(p); //αποδεσμεύουμε τον χώρο μνήμης
```

- Και εντελώς όμοια μπορούμε να κάνουμε το ίδιο και για οποιοδήποτε άλλο τύπο δεδομένων

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

3. Δέσμευση Μεταβλητής

- Πρέπει πάντα να ελέγχουμε αν η εκτέλεση της malloc πέτυχε (δηλαδή ότι δεσμευτηκε ο απαιτούμενος χώρος στη μνήμη)
- Αυτό γίνεται με τον έλεγχο της επιστρεφόμενης τιμής:
  - Αν είναι NULL τότε είχαμε αποτυχία στη δέσμευση της μνήμης
  - Αν δεν είναι NULL τότε όλα πήγαν καλά.

- Θα ακολουθήσουμε και την ακόλουθη μορφή όταν θα δεσμεύουμε τη μνήμη με την malloc

```
p=malloc(sizeof(int)
if (!p)
{
   printf(“Apotyxia Desmeysis Mnimis”);
   exit(0);
}
```

  - Όπου ελέγχουμε επιτόπου αν δεσμεύτηκε χώρος στη μνήμη και σε περίπτωση αποτυχίας προκαλούμε βίαιο τερματισμό στο πρόγραμμά μας.

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

3. Δέσμευση Μεταβλητής

- Συνολικά λοιπόν ένα πρόγραμμα που αναδεικνύει την δυναμική δέσμευση μνήμης για μία μεταβλητή είναι το ακόλουθο

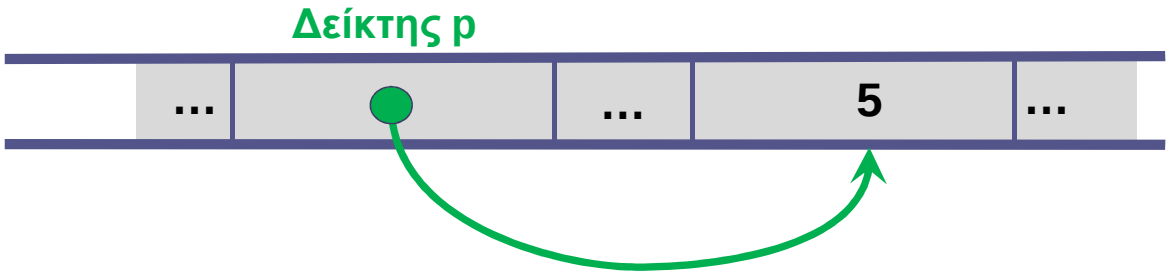
```
/*malloc_var.c Deixnei pws desmeuoume xwro gia mia metavliti */
#include <stdio.h>
#include <stdlib.h>

main()
{
   int *p;

   p=malloc(sizeof(int));
   if (!p)
   {
      printf("Adynamia desmeusis mnimis");
      exit(0);
   }

   printf("Dwse enan akeraio arithmo: ");
   scanf("%d",p);
   printf("Pliktrologisate %d",*p);

   free(p);
}
```

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

3. Δέσμευση Μεταβλητής

- Ο τρόπος που διαχειριστήκαμε την μεταβλητή (δήλωση δείκτη, δέσμευση χώρου, ανάθεση τιμής) αντιστοιχεί στην εξής εικόνα της μνήμης:



- Ο τρόπος χρήσης είναι ελαφρά ανισόρροπος. Η χρήση της δυναμικής δέσμευσης μνήμης γίνεται συνήθως για τους πίνακες και όχι για μεταβλητές.

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

4. Δέσμευση Μονοδιάστατου Πίνακα

- Η πραγματική χρησιμότητα της δυναμικής δέσμευσης μνήμης είναι η δυναμική δέσμευση μνήμης για έναν πίνακα.
- Κάνουμε χρήση της malloc και δεσμεύουμε τόσο χώρο όσο πιάνει ο αντίστοιχος τύπος δεδομένων (επί) το πλήθος των θέσεων του πίνακα
  - Για να το κάνουμε αυτό χρησιμοποιούμε τον τελεστή sizeof
- Χρησιμοποιούμε κανονικά τον πίνακά μας όπως έχουμε μάθει.
- Δεν ξεχνάμε ποτέ να αποδεσμεύσουμε τον χώρο που δεσμεύσαμε με την free.
- Βλέπουμε ένα παράδειγμα με πίνακα ακεραίων:

```
int *p; //Εδώ δηλώνουμε τον δείκτη σε ακέραια μεταβλητή.

p=malloc(sizeof(int)*10); //Δεσμεύουμε δυναμικά χώρο για πίνακα 10 ακεραίων

p[3]=5; //Εδώ κάνουμε χρήση του πίνακα όπως έχουμε μάθει

free(p); //αποδεσμεύουμε τον χώρο μνήμης
```

  - Και αντίστοιχα δουλεύουμε για οποιοδήποτε άλλο τύπο δεδομένων

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

4. Δέσμευση Μονοδιάστατου Πίνακα

- Το ακόλουθο πρόγραμμα επιδεικνύει έναν συνήθη κώδικα για την δυναμική δέσμευση ενός μονοδιάστατου πίνακα:

```
/*malloc_1d-array.c Deixnei pws desmeuoume xwro gia enan monodiastato
pinaka */
#include <stdio.h>
#include <stdlib.h>

main()
{
  int *p; //Dilwsi deikti
  int i,N;

  /* Diavasma Diastasis Pinaka */
  printf("Dwse ti diastasi tou pinaka: ");
  scanf("%d",&N);
```

συνεχίζεται...

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

4. Δέσμευση Μονοδιάστατου Πίνακα

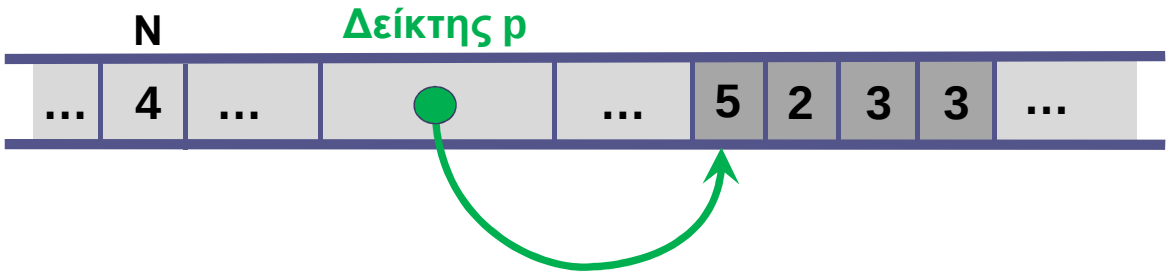
```
   /* Dynamiki Desmeysi mnimis */
   p=malloc(sizeof(int)*N);
   if (!p)
   {
      printf("Adynamia desmeusis mnimis");
      exit(0);
   }

   /* Kapoios Ypologismos ston pinaka */
   for (i=0; i<N; i++)
   {
      p[i]=i*i*i;
      printf("\np[%d]=%d",i,p[i]);
   }

   /* Apodesmeysi Mnimis */
   free(p);
}
```

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

4. Δέσμευση Μονοδιάστατου Πίνακα

- Ο τρόπος που διαχειριστήκαμε την δέσμευση του πίνακα αντιστοιχεί στην εξής εικόνα της μνήμης (π.χ. αν ο χρήστης εισήγαγε το N=4 και τις τιμές 5,2,3,3) φαίνεται στο σχήμα:



- Και αντιστοιχεί στην κωδικοποίηση του πίνακα

$$p = [5 \quad 2 \quad 3 \quad 3]$$

- Ο τρόπος χρήσης είναι ελαφρά ανισόρροπος. Η χρήση της δυναμικής δέσμευσης μνήμης γίνεται συνήθως για τους πίνακες και όχι για μεταβλητές.

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

4. Δέσμευση Μονοδιάστατου Πίνακα

- Καταλαβαίνουμε ότι με τον τρόπο αυτό μπορούμε να δεσμεύσουμε όσο χώρο θέλουμε κατά τον χρόνο εκτέλεσης.

- Π.χ. Μπορεί ο χρήστης να εισάγει ότι θέλει έναν πίνακα 10 ακεραίων, οπότε εμείς προγραμματιστικά μπορούμε να δεσμεύσουμε ακριβώς όσα δεδομένα χρειαζόμαστε. Αυτό ακριβώς είναι η **δυναμική δέσμευση μνήμης** σε αντίθεση με την στατική δέσμευση μνήμης που καθορίζουμε κατά το χρόνο μεταγλώττισης ποιο θα είναι το μέγεθος του πίνακα.

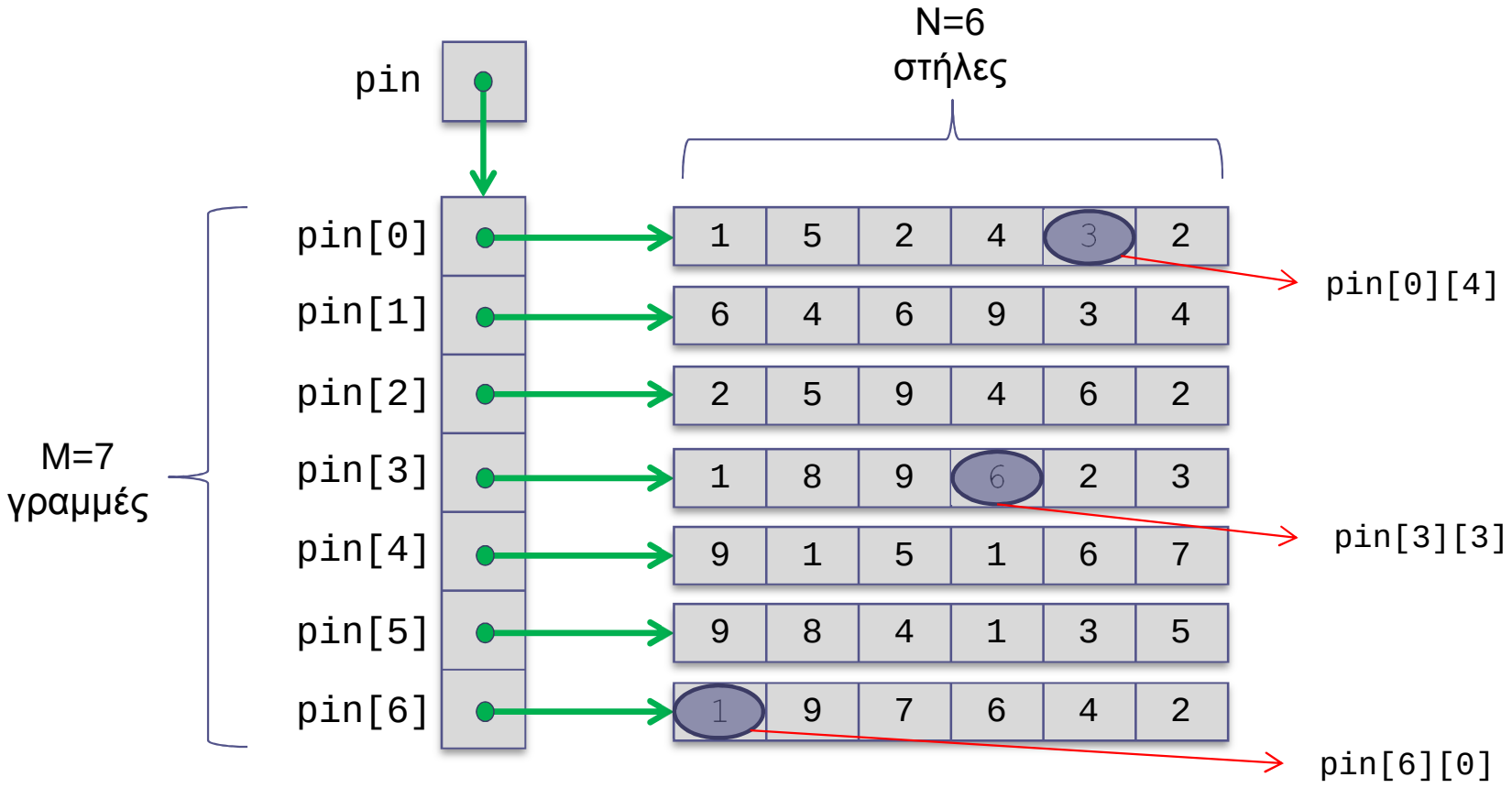
  - Αυτό είναι ιδιαίτερα χρήσιμο και με τις συμβολοσειρές που το μέγεθος τους μπορεί να είναι μεταβλητό και δεν θέλουμε να κάνουμε καταχρήσεις με την μνήμη μας.

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

5. Δέσμευση Διδιάστατου Πίνακα

- Αντίστοιχα μπορούμε να εργαστούμε για έναν διδιάστατο πίνακα.
- Ένας διδιάστατος πίνακας μπορεί να ειδωθεί ως ένας πίνακας που περιέχει μονοδιάστατους πίνακες:



Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

5. Δέσμευση Διδιάστατου Πίνακα

- Με άλλα λόγια θέλουμε έναν πίνακα (pin) του οποίου κάθε στοιχείο θα είναι δείκτης!
  - Έτσι αντίστοιχα με το γεγονός ότι για να δεσμεύσουμε δυναμικά έναν πίνακα ακεραίων, δηλώνουμε έναν δείκτη σε ακέραιο και κάνουμε δέσμευση Ν ακεραίων.
  - Έτσι για να δεσμεύσουμε δυναμικά έναν πίνακα δεικτών ακεραίων, θα δηλώσουμε έναν δείκτη σε δείκτη ακεραίων και κάνουμε δέσμευση Μ δεικτών ακεραίων!
    - Ο δείκτης σε δείκτη ακεραίου θα αναφέρεται ως διπλός δείκτης και θα δηλώνεται με την εντολή:

```
int **p;
```

    - Η δέσμευση θα γίνεται με τις ακόλουθες εντολές:

```
p=malloc(sizeof(int*)*M);
for (i=0; i<M; i++)
   p[i]=malloc(sizeof(int)*N);
```

    - Και η αποδέσμευση θα γίνει με τις εντολές:

```
for (i=0; i<M; i++)
   free (p[i]);
free(p);
```

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

5. Δέσμευση Διδιάστατου Πίνακα

- Το ακόλουθο πρόγραμμα επιδεικνύει έναν συνήθη κώδικα για την δυναμική δέσμευση ενός διδιάστατου πίνακα:

```
*malloc_2d-array.c Deixnei pws desmeuoume xwro gia enan didiastato pinaka
*/
#include <stdio.h>
#include <stdlib.h>

main()
{
   int **p; //Dilwsi diplou deikti-pinaka
   int i,j,M,N;

   /* Diavasma Diastasewn Pinaka */
   printf("Dwse to plithos twn grammwn tou pinaka: ");
   scanf("%d",&M);
   printf("Dwse to plithos twn stilwn tou pinaka: ");
   scanf("%d",&N);
```

συνεχίζεται...

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

5. Δέσμευση Διδιάστατου Πίνακα

```
/* Dynamiki Desmeysi mnimis */
p=malloc(sizeof(int*)*M);
if (!p)
{
   printf("Adynamia desmeusis mnimis");
   exit(0);
}
for (i=0; i<M; i++)
{
   p[i]=malloc(sizeof(int)*N);
   if (!p[i])
   {
       printf("Adynamia desmeusis mnimis");
       exit(0);
   }
}
```

συνεχίζεται...

Α. Στατική και Δυναμική Δέσμευση Μνήμης

2. Δυναμική Δέσμευση Μνήμης

5. Δέσμευση Διδιάστατου Πίνακα

```
   /* Kapoios Ypologismos ston pinaka */
   for (i=0; i<M; i++)
   {
      for (j=0; j<N; j++)
      {
         p[i][j]=1+(i+j)%5;
         printf("%2d ",p[i][j]);
      }
      printf("\n");
   }

   /* Apodesmeysi Mnimis */
   for (i=0; i<M; i++)
      free (p[i]);
   free(p);
}
```

Γ. Ασκήσεις

1. Χώρος Αποθήκευσης Διδιάστατου Πίνακα

- Κατασκευάστε ένα πρόγραμμα C το οποίο να κάνει μια μελέτη του χώρου αποθήκευσης ενός διδιάστατου πίνακα:
  - Να ζητάει από τον χρήστη να εισάγει το πλήθος των γραμμών (M) και το πλήθος των στηλών (N).
  - Να δεσμεύει δυναμικά τον χώρο αποθήκευσης για έναν πίνακα double MxN
  - Να υπολογίζει το μέγεθος σε bytes που απαιτήθηκαν για την αποθήκευση του πίνακα και να τυπώνει το αποτέλεσμα
  - Να αποδεσμεύει τον χώρο μνήμης που δεσμεύθηκε δυναμικά.

Γ. Ασκήσεις

2. Κάτω Τριγωνικός Πίνακας

- Στα μαθηματικά, ένας κάτω τριγωνικός πίνακας, είναι ένας πίνακας διάστασης NxN που τα στοιχεία κάτω από την κύρια διαγώνιο είναι ίσα με το 0. Για παράδειγμα ο ακόλουθος πίνακας είναι ένας 4x4 κάτω τριγωνικός πίνακας:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 4 & 5 & 4 & 0 \\ 3 & 1 & 3 & 2 \end{bmatrix}$$

- Να κατασκευαστεί ένα πρόγραμμα το οποίο να διαχειρίζεται κάτω τριγωνικούς πίνακες:
  - Να δεσμεύει στατικά και να δηλώνει έναν κάτω τριγωνικό πίνακα Α διάστασης NxN (το Ν να εισάγεται από το χρήστη με τιμή 5..20). Να αρχικοποιεί τα στοιχεία του πίνακα με τυχαίους αριθμούς στο εύρος [1..9]
  - Να δεσμεύει δυναμικά έναν κάτω τριγωνικό πίνακα Β διάστασης NxN. Προσοχή! Να δεσμευτεί χώρος μόνο για τα στοιχεία του πίνακα και όχι για τα στοιχεία που γνωρίζουμε ότι δεν είναι πάντα μηδενικά!

Γ. Ασκήσεις

3. Κάτω Τριγωνικός Πίνακας (συνέχεια)

- Να επεκταθεί το πρόγραμμα της προηγούμενης εφαρμογής έτσι ώστε:
  1. Να αντιγράφονται τα στοιχεία του πίνακα Α στον πίνακα Β.
  2. Να γίνεται εκτύπωση των στοιχείων των δύο πινάκων.

Γ. Ασκήσεις
4. Ένας Πίνακας από Λέξεις

- Να γραφεί πρόγραμμα το οποίο θα διαβάζει ακριβώς 10 λέξεις και θα τις αποθηκεύει σε μία δυναμική δομή δεδομένων:
  1. Να δηλώνει έναν πίνακα Ν=10 συμβολοσειρών (χωρίς να δεσμεύεται χώρος για κάθε συμβολοσειρά)
  2. Να διαβάζονται οι διαδοχικά οι λέξεις με τον εξής τρόπο:
     1. Να αποθηκεύεται σε έναν προσωρινό χώρο μνήμης
     2. Να ελέγχεται ότι ο χρήστης έχει εισάγει λέξη. Στην εφαρμογή αυτή ως λέξη εννοούμε μία συμβολοσειρά που αποτελείται από μόνο μικρούς λατινικούς χαρακτήρες (χρησιμοποιείστε την συνάρτηση που κατασκευάσαμε στο «Μάθημα 10: Χαρακτήρες και Συμβολοσειρές – Εφαρμογή 5»). Σε αντίθετη περίπτωση να τερματίζει το πρόγραμμα.
     3. Να δεσμεύεται χώρος στον πίνακα σύμφωνος με το μήκος της λέξης (χρησιμοποιείστε τη συνάρτηση που κατασκευάσαμε στο «Μάθημα 10: Χαρακτήρες και Συμβολοσειρές – Εφαρμογή 1»).
     4. Να αντιγράφεται η συμβολοσειρά στην αντίστοιχη θέση του πίνακα (χρησιμοποιείστε τη συνάρτηση που κατασκευάσαμε στο «Μάθημα 10: Χαρακτήρες και Συμβολοσειρές – Εφαρμογή 2»).
  3. Να γίνεται εκτύπωση των 10 λέξεων που διαβάστηκαν
  4. Να αποδεσμεύεται ο χώρος στη μνήμη.