ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.1

Πρόβλημα

- Να γίνει πρόγραμμα το οποίο να επιλύει το Διαγώνιο Σύστημα:

$$A \cdot x = b$$

ή το σύστημα :

$$\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & 0 & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

ή σε μορφή εξισώσεων το σύστημα :

$$\begin{array}{cccccccccc} a_{11}x_1 & & & & & & & & = & b_1 \\ & a_{22}x_2 & & & & & & & = & b_2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & \dots & a_{ii}x_i & & & = & b_i \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & \dots & \dots & & a_{nn}x_n & = & b_n \end{array}$$

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b` (**μόνο** για τα **μη μηδενικά** στοιχεία) .
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Για** τις τιμές του $x_i, i = 1,2,\dots,n$:

   Υπολογίζουμε το $x_i = \frac{b_i}{a_{i,i}}$.
5. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

4.1 Δήλωση Πινάκων Δύο Διαστάσεων

- Η δήλωση γενικά ενός δυσδιάστατου πίνακα `10×10` θέσεων για πραγματικούς αριθμούς γίνεται με την εντολή :

```
double pin[10][10];
```

με στοιχεία `pin[0][0], pin[0][1],…, pin[0][9], pin[1][0], p[1][1],…, p[1][9],…, pin[9][0], pin[9][1],…, pin[9][9].`

- Με την εντολή :

```
double a[n+1][n+1];
```

δηλώνουμε τον δυσδιάστατο πίνακα `A` με `n+1` γραμμές και `n+1` στήλες με στοιχεία `a[0][0], a[0][1],…, a[0][n],a[1][0], a[1][1],…, a[1][n],…, a[n][0], a[n][1],…, a[n][n],` απ' τα οποία μπορούμε να μη χρησιμοποιήσουμε τα στοιχεία της $1^{ης}$ γραμμής και της $1^{ης}$ στήλης .

Παρατηρήσεις

- Η δήλωση του πλήθους των στοιχείων του πίνακα μπορεί να γίνει και με τη δήλωση σταθεράς. Έτσι, θα είχαμε το ίδιο αποτέλεσμα με τις εντολές :

```
#define nmax 10
double pin[nmax][nmax];
```

Αυτό μας δίνει τη δυνατότητα να χρησιμοποιήσουμε πίνακα με μεταβλητή διάσταση, την οποία θα δίνουμε απ' το πληκτρολόγιο.

- Όταν αναφερόμαστε στα στοιχεία ενός διδιάστατου πίνακα πρέπει να χρησιμοποιούμε δύο μεταβλητές για δείκτες. π.χ. `a[i][j]` είναι το στοιχείο του πίνακα που βρίσκεται στην `i` γραμμή και στη `j` στήλη.

4.2 Διάβασμα Στοιχείων ενός Διδιάστατου Πίνακα

- Με την ομάδα εντολών :

```
for ( i = 1; i<=n; i++ )
 for ( j = 1; j<=n; j++ )
   if ( i == j )
     {
     printf("Dose timh gia to a[%d,%d] : ",i,j);
     scanf ("%lf",&a[i][j]);
     }
   else
     a[i][j] = 0;
```

διαβάζουμε `n = 3` πραγματικούς αριθμούς απ' το πληκτρολόγιο χωρίς `format` **(έναν αριθμό σε κάθε γραμμή)** και τους αποθηκεύουμε στον πίνακα `a`.

Παρατηρήσεις

- Αν θέλαμε να διαβάσουμε τα στοιχεία ενός Διδιάστατου πίνακα κατά γραμμές (n στοιχεία στην κάθε γραμμή) θα χρησιμοποιούσαμε την ομάδα εντολών :

```
for ( i = 1; i<=n; i++ )
{
for ( j = 1; j<=n; j++ )
     scanf ("%lf",&a[i][j]);
```

και στην εκτέλεση του προγράμματος θα δίναμε `n` στοιχεία σε κάθε γραμμή χωρισμένα με κενά.

4.3 Εμφάνιση n Στοιχείων Διδιάστατου Πίνακα

- Με την ομάδα εντολών :

```
for ( i = 1; i<=n; i++ )
 {
 for ( j = 1; j<=n; j++ )
   printf("%lf",a[i][j]);
 printf("\n");
 }
```

εμφανίζουμε στην οθόνη `n × n` πραγματικούς αριθμούς (τα στοιχεία του πίνακα `a`) χωρίς `format`.

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.2

Πρόβλημα

- Να τροποποιηθεί η Άσκηση `4.1`, ώστε η επίλυση του Συστήματος να γίνεται με την κλήση της συνάρτησης `diag()`.

Αλγόριθμος `main()`

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b` (**μόνο** για τα **μη μηδενικά** στοιχεία) .
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη σνάρτηση `diag()` για τον Υπολογισμό** των $x_i, i = 1..n$.
5. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

Αλγόριθμος `function diag()`

1. **Για** τις τιμές του $x_i, i = 1,2,...,n$:

   Υπολογίζουμε το $x_i = \frac{b_i}{a_{i,i}}$.

4.4 Πέρασμα Διδιάστατων Πινάκων σαν Παραμέτρων σε functions

➢ Για το πέρασμα Διδιάστατων Πινάκων σαν παραμέτρων σε functions ισχύουν τα παρακάτω :

♦ Οι Πίνακες - Παράμετροι θα πρέπει να δηλωθούν στο κυρίως Πρόγραμμα.

♦ Ο Πίνακας περνάει σαν παράμετρος στη δήλωση της `function` με το όνομά του και αγκύλες, όπου όμως στο δεύτερο ζεύγος αγκυλών περνάει ο αριθμός των στηλών. Στο προηγούμενο παράδειγμα, η δήλωση της `function` πρέπει να γίνει με την εντολή :

```
void diag(int n, double a[][nmax+1], double b[], double x[]);
```

♦ Στη χρήση ή κλήση της `function` χρησιμοποιείται μόνο το όνομα του πίνακα. Στο προηγούμενο παράδειγμα, η κλήση της function πρέπει να γίνει με την εντολή :

```
diag (n, a, b, x); //Επίλυση Διαγωνίου Συστήματος με κλήση function
```

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.3

Πρόβλημα

- Να γίνει πρόγραμμα το οποίο με τη χρήση συνάρτησης να επιλύει το Αντι-Διαγώνιο Σύστημα :

$$A \cdot x = b$$

ή το σύστημα :

$$\begin{bmatrix} 0 & 0 & \dots & 0 & a_{1n} \\ 0 & 0 & \dots & a_{2,n-1} & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n-1,2} & \dots & 0 & 0 \\ a_{n1} & 0 & \dots & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_{n-1} \\ b_n \end{bmatrix} \end{bmatrix}$$

ή σε μορφή εξισώσεων το σύστημα :

$$\begin{array}{cccccccccccc} & & & & & & & a_{1n}x_n & = & b_1 \\ & & & & & & a_{2,n-1}x_{n-1} & & = & b_2 \\ \dots & & \dots & & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & & \dots & & \dots & \dots & \dots & \dots & = & \\ & a_{n-1,2}x_2 & & & & & & & = & b_{n-1} \\ a_{n1}x_1 & & & & & & & & = & b_n \end{array}$$

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b` (**μόνο** για τα **μη μηδενικά** στοιχεία) .
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη σνάρτηση `antidiag()` για τον Υπολογισμό** των $x_i, i = 1..n$.
5. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

Αλγόριθμος `function antidiag()`

1. **Για** τις τιμές του $x_i, i = 1,2,...,n$:

   Υπολογίζουμε το $x_i = \dfrac{b_{n-i+1}}{a_{n-i+1,i}}$.

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.4

Πρόβλημα

- Να γίνει πρόγραμμα, το οποίο με τη χρήση συνάρτησης να επιλύει το Γραμμικό **Κάτω Τριγωνικό** Σύστημα :

$$A \cdot x = b$$

ή το σύστημα : $\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & 0 \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$

ή σε μορφή εξισώσεων το σύστημα :

$$\begin{bmatrix} a_{11}x_1 & & & & & & & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & & & & & = & b_2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & = & b_n \end{bmatrix}$$

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b` (**μόνο** για τα **μη μηδενικά** στοιχεία) .
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη σνάρτηση `kato_trig()` για τον Υπολογισμό** των $x_i, i = 1..n$.
5. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

Αλγόριθμος `function kato_trig()`

1) **Λύνουμε** το σύστημα ως προς το $x_1 = \frac{b_1}{a_{1,1}}$.

2) **Για** τις τιμές του $x_i, \quad i = 2,3,\dots,n$

   a) **Θέτουμε** sum = 0

   b) **Υπολογίζουμε** το άθροισμα $sum = \sum_{k=1}^{i-1} a_{i,k} x_k$.

   c) **Υπολογίζουμε** το $x_i = \frac{b_i - sum}{a_{i,i}}$

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.5

Πρόβλημα

- Να γίνει πρόγραμμα, το οποίο με τη χρήση συνάρτησης να επιλύει το Γραμμικό **Άνω Τριγωνικό** Σύστημα :

$$A \cdot x = b$$

ή το σύστημα :
$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

ή σε μορφή εξισώσεων το σύστημα :
$$a_{11}x_1 + a_{12}x_2 + \dots\dots\dots\dots + a_{1n}x_n = b_1$$
$$a_{22}x_2 + a_{23}x_3 \dots\dots\dots\dots + a_{2n}x_n = b_2$$
$$\dots\dots$$
$$a_{nn}x_n = b_n$$

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b` (**μόνο** για τα **μη μηδενικά** στοιχεία) .
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη συνάρτηση `anw_trig()` για τον Υπολογισμό** των $x_i, i = 1..n$.
5. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

Αλγόριθμος `function anw_trig()`

1) **Λύνουμε** το σύστημα ως προς το $x_n = \frac{b_n}{a_{n,n}}$.

2) **Για** τις τιμές του x_i, $i = n-1, n-2, \dots, 1$

   a) **Θέτουμε** sum = 0.

   b) **Υπολογίζουμε** το άθροισμα $sum = \sum_{k=i+1}^{n} a_{i,k}x_k$.

   c) **Υπολογίζουμε** το $x_i = \frac{b_i - sum}{a_{i,i}}$

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.6

Πρόβλημα

- Να γίνει πρόγραμμα, το οποίο με την κλήση της συνάρτησης **`gauss()`** μετατρέπει ένα πλήρες Γραμμικό Σύστημα $A \cdot x = b$ σε **Άνω Τριγωνικό** και το επιλύει με την κλήση της συνάρτησης **`anw_trig()`**. Το Γραμμικό Σύστημα θα έχει τη μορφή :

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

ή σε μορφή εξισώσεων

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \dots\dots\dots\dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots\dots\dots\dots + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots\dots\dots\dots + a_{nn}x_n = b_n \end{bmatrix}$$

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b`.
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη συνάρτηση `gauss(n, a, b)`** για τη **μετατροπή** του συστήματος σε **άνω τριγωνικό.**
5. **Καλούμε τη συνάρτηση `anw_trig()` για τον Υπολογισμό** των $x_i, i = 1..n$.
6. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.7

Πρόβλημα

- Να γίνει πρόγραμμα, το οποίο με την κλήση της συνάρτησης **`gauss_odhghsh()`** μετατρέπει ένα πλήρες Γραμμικό Σύστημα $A \cdot x = b$ σε **Άνω Τριγωνικό** και το επιλύει με την κλήση της συνάρτησης **`anw_trig()`**.

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b`.
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη συνάρτηση `gauss_odhghsh(n, a, b)`** για τη **μετατροπή** του συστήματος σε **άνω τριγωνικό.**
5. **Καλούμε τη συνάρτηση `anw_trig()` για τον Υπολογισμό** των $x_i, i = 1..n$.
6. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

Αλγόριθμος `function gauss_odhghsh()`

Αντί να κάνουμε έλεγχο για μηδενικό διαγώνιο στοιχείο,

1) **Για** κάθε στήλη J
   I. **Βρίσκουμε** τη γραμμή `max_thesi` που περιέχει το μέγιστο κατ' απόλυτη τιμή στοιχείο του Πίνακα `A` στη στήλη
   II. **Ανταλάσσουμε**, αν χρειαστεί, τα στοιχεία των γραμμών `j` και `max_thesi`

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.8

Πρόβλημα

- Να γίνει πρόγραμμα, το οποίο με την κλήση της συνάρτησης **`gauss_jordan()`** μετατρέπει ένα πλήρες Γραμμικό Σύστημα $A \cdot x = b$ σε **Διαγώνιο** και το επιλύει με την κλήση της συνάρτησης **`diag()`**.

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b`.
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη συνάρτηση `gauss_jordan(n, a, b)`** για τη **μετατροπή** του συστήματος σε **Διαγώνιο.**
5. **Καλούμε τη συνάρτηση `diag()` για τον Υπολογισμό** των $x_i, i = 1..n$.
6. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.

Αλγόριθμος `function gauss_jordan()`

Στη μέθοδο **Gauss** κάνουμε **απαλοιφή** αγνώστων για κάθε στήλη στις **γραμμές** `i=j+1..n`, ενώ στη μέθοδο **Gauss-Jordan** κάνουμε **απαλοιφή** για κάθε στήλη στις γραμμές `i=1..j-1` και `i=j+1..n`.

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 4.9

Πρόβλημα

- Να γίνει πρόγραμμα, το οποίο με την κλήση της συνάρτησης **`gauss_seidel()`** επιλύει ένα πλήρες Γραμμικό Σύστημα $A \cdot x = b$ ελέγχοντας αν ο Πίνακας των συντελεστών των αγνώστων έχει Διαγώνια Υπεροχή (Σε κάθε Γραμμή ή Στήλη του Πίνακα A το Διαγώνιο στοιχείο να είναι κατ΄απόλυτη τιμή μεγαλύτερο από το άθροισμα των απολύτων τιμών των υπολοίπων στοιχείων της γραμμής ή της στήλης). Ο έλεγχος τερματισμού θα γίνεται με τη χρήση της συνάρτησης `norm()`, η οποία θα υπολογίζει το άθροισμα των απολύτων τιμών της διαφοράς του κάθε στοιχείου του πίνακα `x` και του `oldx`, $\sum_{i=1}^{n} |x_i - oldx_i|$.

Αλγόριθμος main()

1. **Διαβάζουμε** την τιμή του n (η διάσταση του Πίνακα `A`).
2. Δίνουμε **τιμές** στα στοιχεία των πινάκων `A` και `b`.
3. **Εμφανίζουμε** στην οθόνη τα στοιχεία των πινάκων `A` και `b`.
4. **Καλούμε τη `boolean` συνάρτηση `yperoxh(n, a)`**, η οποία θα επιστρέφει την τιμή `true` ή `false`, ανάλογα με το αν υπάρχει διαγώνια υπεροχή ή όχι.
5. Στην περίπτωση που υπάρχει διαγώνια υπεροχή, κ**αλούμε τη συνάρτηση `gauss_seidel()` για τον Υπολογισμό** των $x_i, i = 1..n$.
6. **Εμφανίζουμε** στην οθόνη τα στοιχεία του `x`.
7. Στην περίπτωση που δεν υπάρχει διαγώνια υπεροχή, **εμφανίζουμε αντίστοιχο μήνυμα.**

Αλγόριθμος `function gauss_seidel()`

1. **Θέτουμε** αρχικές τιμές στα στοιχεία των πινάκων `x` και `oldx`
2. **Για όσο** το σφάλμα $\sum_{i=1}^{n} |x_i - oldx_i|$ είναι μεγαλύτερο του 10^{-15}
   a) **Αποθηκεύουμε** τις προηγούμενες τιμές του πινακα `x` στον πίνακα `oldx`
   b) **Για** τις τιμές του $x_i, i = 1,2,\ldots,n$:
      i) **Θέτουμε** `sum = 0`.
      ii) **Υπολογίζουμε** το άθροισμα $sum = \sum_{j=1}^{n} a_{ij} \cdot x_j$
      iii) **Αφαιρούμε** απ' το sum το $a_{ii} \cdot x_i$
      iv) **Υπολογίζουμε** το νέο $x_i = \frac{b_i - sum}{a_{ii}}$

Αλγόριθμος function bool `yperoxh(n, a)`

1. **Θέτουμε** `yp_g = true`
2. **Θέτουμε** i = 1
3. **Για** Όσο ($i \le n$) και (`yp_g = true`):

   a) **Θέτουμε** sum = 0

   b) **Υπολογίζουμε** το άθροισμα $sum = \sum_{j=1}^{n} |a_{i,j}|$.

   c) **Αφαιρούμε** απ' το `sum` το $|a_{i,i}|$.

   d) Αν $(|a_{i,i}| < sum)$, **θέτουμε** στο `yp_g` την τιμή `false`

   e) Αυξάνουμε την τιμή του μετρητή i κατά 1

4. **Αν** `yp_g = true`**, επιστρέφουμε την τιμή του** `yp_g`
5. **Αν** `yp_g = false`**, Ελέγχουμε** αν για τον πίνακα A υπάρχει Διαγώνια **Υπεροχή** Κατά Στήλες**.**
6. **Θέτουμε** `yp_s = true`
7. **Θέτουμε** j = 1
8. **Για** Όσο ($j \le n$) και (`yp_s = true`):

   a) **Θέτουμε** sum = 0

   b) **Υπολογίζουμε** το άθροισμα $sum = \sum_{i=1}^{n} |a_{i,j}|$.

   c) **Αφαιρούμε** απ' το `sum` το $|a_{j,j}|$.

   d) Αν $(|a_{j,j}| < sum)$, **θέτουμε** στο `yp_s` την τιμή `false`

   e) Αυξάνουμε την τιμή του μετρητή j κατά 1

9. **Επιστρέφουμε την τιμή του** `yp_s`

4.5 Συναρτήσεις τύπου boolean

- Η συνάρτηση, όπως και οι μεταβλητές δηλώνονται με τη λέξη **`bool`** :

```
bool yperoxh(int n, double a[][nmax+1]);
bool yp_g, yp_s;
```

- Στον έλεγχο μπορούμε να χρησιμοποιήσουμε τη μεταβλητή χωρίς την τιμή της. Π.χ. `if (yp_g)` αντί του `if (yp_g == true )`.