# Η ΓΛΩΣΣΑ C

## Μάθημα 3:

## Μεταβλητές και Σταθερές

Δημήτρης Ψούνης



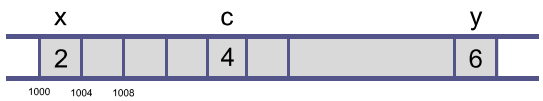

# Περιεχόμενα Μαθήματος





## Στόχος του Μαθήματος

- Κάθε πρόγραμμα αποθηκεύει δεδομένα στην μνήμη προκειμένου να τα χρησιμοποιήσει για να κάνει τους υπολογισμούς του.
- Η C χρησιμοποιεί δύο τρόπους για να αποθηκεύσει τα δεδομένα της:
  - Τις μεταβλητές, που είναι θέσεις αποθήκευσης δεδομένων στις οποίες μπορούμε να παρέμβουμε και να αλλάξουμε την τιμή τους, όσες φορές θέλουμε κατά τη διάρκεια εκτέλεσης του προγράμματος.
  - Τις σταθερές, που είναι θέσεις αποθήκευσης δεδομένων στις οποίες δεν μπορούμε να παρέμβουμε και σε όλη την διάρκεια του προγράμματος έχουν την ίδια τιμή. Την τιμή αυτή την δηλώνουμε στην αρχή του προγράμματος




- Πρέπει να διαχωρίσουμε εξαρχής στο μυαλό μας, ότι κάθε **μεταβλητή** έχει το **όνομα** της (π.χ η μεταβλητή x), που έχει μία **θέση μνήμης** (π.χ. η x είναι στη θέση μνήμης 1000), στην οποία αποθηκεύεται η **τιμή** της μεταβλητής (π.χ x=3) και έχει και έναν **τύπο δεδομένων** (π.χ η x είναι ακέραια)



# Α. Μεταβλητές

## 1. Ονόματα Μεταβλητών

**Κάθε μεταβλητή έχει ένα όνομα.** Μπορούμε να χρησιμοποιήσουμε (σχεδόν) οποιοδήποτε όνομα θέλουμε, σεβόμενοι τους ακόλουθους **κανόνες**:

- Το όνομα μπορεί να χρησιμοποιεί τους ακόλουθους χαρακτήρες (και μόνον αυτούς)
  - Γράμματα (a-z, A-Z)
  - Ψηφία (0-9)
  - Το underscore ( _ )
- Το όνομα υποχρεωτικά πρέπει να ξεκινάει με γράμμα ή με underscore (όχι ψηφίο).
- Τα κεφαλαία-μικρά έχουν σημασία (case-sensitive). Έτσι η μεταβλητή sum θα είναι διαφορετική από την μεταβλητή sUm και από την μεταβλητή Sum.
- Δεν μπορούν να χρησιμοποιηθούν οι λέξεις-κλειδια της γλώσσας (ονόματα τύπων δεδομένων, include κ.λπ.)

# Α. Μεταβλητές
## 1. Ονόματα Μεταβλητών

**Συμβουλές:**

- Είναι καλή τακτική, να χρησιμοποιούμε ονόματα που σχετίζονται με την λειτουργία των μεταβλητών. Π.χ. Αν θέλουμε να αποθηκεύσουμε σε μια μεταβλητή το άθροισμα κάποιων τιμών, είναι προτιμότερο να ονομάσουμε την μεταβλητή *sum*, παρά να την ονομάσουμε με ένα ξερό π.χ. *z*.
- Σπάνια χρησιμοποιούμε κεφαλαία γράμματα για μεταβλητές. Έχουν επικρατήσει τα μικρά γράμματα
- Έχουν επικρατήσει δύο στυλ γραφής των μεταβλητών όταν θέλουμε 2 λέξεις στις μεταβλητές για να τις περιγράψουμε:
  - Να τις χωρίζουμε με _, π.χ. interest_rate
  - Να τις κάνουμε 2 λέξεις, με την δεύτερη να αρχίζει με κεφαλαίο π.χ. interestRate



# Α. Μεταβλητές
## 2. Τύποι Δεδομένων

- Ανάλογα με τα δεδομένα που θέλουμε να αποθηκεύσουμε πρέπει να χρησιμοποιήσουμε και διαφορετικό τύπο μεταβλητής. Με τον όρο **τυποι δεδομένων** ονομάζουμε τους διαφορετικούς τύπους μεταβλητών που μπορούμε να χρησιμοποιήσουμε.

- Είναι δική μας ευθύνη να επιλέξουμε τον τύπο δεδομένων των μεταβλητών.
  - Αν θέλουμε να αποθηκεύσουμε σε μια μεταβλητή τον βαθμό μας σε ένα μάθημα, πρέπει να χρησιμοποιήσουμε μια ακέραια μεταβλητή διότι ο βαθμός μας είναι ακέραιος.
  - Αν θέλουμε να αποθηκεύσουμε σε μια μεταβλητή το επιτόκιο ενός δανείου, πρέπει να χρησιμοποιήσουμε μια πραγματική μεταβλητή μικρής ακρίβειας, χωρίς να μας ενδιαφέρει η ακρίβεια πολλών δεκαδικών ψηφίων γιατί μας αρκούν – συνήθως - 2 δεκαδικά ψηφία
  - Αν θέλουμε να αποθηκεύσουμε σε μια μεταβλητή τα πρώτα 15 ψηφία του αρρητου αριθμού π, θα χρειαστούμε μια πραγματική μεταβλητή διπλής ακρίβειας
  - κ.ο.κ.



# Α. Μεταβλητές
## 2. Τύποι Δεδομένων
### 1. Τύποι Αριθμητικών Δεδομένων

- Η C έχει πολλούς τύπους δεδομένων με τους οποίους μπορούμε να αποθηκεύσουμε αριθμούς.
- Όσο μεγαλύτερη πληροφορία θέλουμε να αποθηκεύσουμε σε έναν αριθμό, τόσο περισσότερα bytes απαιτούνται στην μνήμη από τον αριθμό.

- Οι **αριθμητικές μεταβλητές** χωρίζονται σε δύο μεγάλες κατηγορίες:
  - Τις **ακέραιες μεταβλητές** που αποθηκεύουμε ακέραιες τιμές (char, int, long, short)
  - Τις **κλασματικές μεταβλητές** που αποθηκεύουμε δεκαδικές τιμές (float, double)

- Ειδικά για τις ακέραιες μεταβλητές όλες οι παραπάνω χωρίζονται σε **προσημασμένες** και **μη προσημασμένες**



# Α. Μεταβλητές
## 2. Τύποι Δεδομένων
### 1. Τύποι Αριθμητικών Δεδομένων (1.Προσημασμένοι Ακέραιοι)

- Οι **προσημασμένες ακέραιες μεταβλητές**, επιτρέπουν την αποθήκευση και αρνητικών αριθμών.

| Όνομα Τύπου Δεδομένων | Συμβολισμός | bytes | Εύρος τιμών |
|---|---|---|---|
| Χαρακτήρας | `char` | 1 | -128 εώς 127 |
| Μικρός Ακέραιος | `short` | 2 | -32768 εώς 32767 |
| Ακέραιος | `int` | 4 | -2147483648 εώς 2147438647 |
| Μεγάλος Ακέραιος | `long` | 4 | -2147483648 εώς 2147438647 |

- Σημείωση: Τα bytes που αντιστοιχούν όπως φαίνεται στον πίνακα εξαρτώνται από το σύστημα μας, αλλά συνήθως είναι όπως φαίνεται εδώ:

# Α. Μεταβλητές
## 2. Τύποι Δεδομένων
### 1. Τύποι Αριθμητικών Δεδομένων (2.Μη Προσημασμένοι Ακέραιοι)

- Οι **μη προσημασμένες ακέραιες μεταβλητές**, επιτρέπουν την αποθήκευση μόνο θετικών αριθμών.
  - Έτσι ξεκινούν από το 9 και επιτρέπουν την αποθήκευση διπλάσιας τιμής σε σχέση με τις προσημασμένες.

| Όνομα Τύπου Δεδομένων | Συμβολισμός | bytes | Εύρος τιμών |
|---|---|---|---|
| Μη προσημασμένος Χαρακτήρας | `unsigned char` | 1 | 0 εώς 255 |
| Μη προσημασμένος Μικρός Ακέραιος | `unsigned short` | 2 | 0 εώς 65535 |
| Μη προσημασμένος Ακέραιος | `unsigned int` | 4 | 0 εώς 4294967295 |
| Μη προσημασμένος Μεγάλος Ακέραιος | `unsigned long` | 4 | 0 εώς 4294967295 |



# Α. Μεταβλητές
## 2. Τύποι Δεδομένων
### 1. Τύποι Αριθμητικών Δεδομένων (3.Πραγματικοί Αριθμοί)

- Οι **πραγματικές μεταβλητές** είναι δύο κατηγοριών **float** και **double** με μόνη διαφορά τα bytes (άρα και την ακρίβεια) κάθε τύπου δεδομένων.

| Όνομα Τύπου Δεδομένων | Συμβολισμός | bytes | Εύρος τιμών |
|---|---|---|---|
| Κινητής Υποδιαστολής Μονής Ακρίβειας | `float` | 4 | $1.2x10^{-38}$ εώς $3.4x10^{38}$ |
| Κινητής Υποδιαστολής Διπλής Ακρίβειας | `double` | 8 | $2.2x10^{-308}$ εώς $1.8x10^{308}$ |

- Πρακτικά:
  - Οι float αποθηκεύουν ικανοποιητικά μέχρι 7 ψηφία
  - Οι double μέχρι 19 ψηφία.



# Α. Μεταβλητές
## 2. Τύποι Δεδομένων
### 1. Τύποι Αριθμητικών Δεδομένων

- Η πληθώρα τύπων δεδομένων μας δίνει περιθώρια για ευέλικτο προγραμματισμό ανάλογα με τις ανάγκες του προγράμματος.
- Δεν σημαίνει ότι πρέπει να επιλέξουμε οπωσδήποτε τον καλύτερο τύπο δεδομένων για την κάθε περίπτωση.

Πρακτικά στα περισσότερα προγράμματα για να αποθηκεύσουμε αριθμούς θα αρκεστούμε στους τύπους δεδομένων:
- int: για τις ακέραιες μεταβλητές
- long: αν πρόκειται να αποθηκευτούν μεγάλες τιμές ακεραίων
- float: για δεκαδικούς αριθμούς μικρής ακρίβειας
- double: για δεκαδικούς αριθμούς μεγάλης ακρίβειας



# Α. Μεταβλητές
## 2. Τύποι Δεδομένων
### 1. Τύποι Αριθμητικών Δεδομένων

- Το ακόλουθο πρόγραμμα προβάλλει στην οθόνη τα μεγέθη σε bytes των τύπων δεδομένων που μελετήσαμε.
- Ο τελεστής sizeof που χρησιμοποιείται επιστρέφει τα bytes του τύπου που δέχεται ως όρισμα.(θα τον μελετήσουμε σε επόμενο μάθημα πιο αναλυτικά)
- Εκτελέστε το πρόγραμμα και κάντε αντιπαραβολή με τους πίνακες των διαφανειών.

```
/* sizes.c: Provallei stin othoni ta megethi se bytes twn tipwn dedomenvn */
#include <stdio.h>

main()
{
  printf("\nTa bytes enos char einai: %d", sizeof(char));
  printf("\nTa bytes enos short einai: %d", sizeof(short));
  printf("\nTa bytes enos int einai: %d", sizeof(int));
  printf("\nTa bytes enos long einai: %d", sizeof(long));

  printf("\nTa bytes enos unsigned short einai: %d", sizeof(unsigned short));
  printf("\nTa bytes enos unsigned int einai: %d", sizeof(unsigned int));
  printf("\nTa bytes enos unsigned long einai: %d", sizeof(unsigned long));

  printf("\nTa bytes enos float einai: %d", sizeof(float));
  printf("\nTa bytes enos double einai: %d", sizeof(double));
}
```

# A. Μεταβλητές
## 2. Τύποι Δεδομένων
### 2. Άλλοι τύποι Δεδομένων

- Εκτός από τους αριθμητικούς τύπους δεδομένων (που αποθηκευουν αριθμούς), υπάρχουν και τύποι δεδομένων που:
  - Αποθηκευουν χαρακτήρες και συμβολοσειρές (ακολουθίες χαρακτήρων). Θα τις δούμε αναλυτικά σε επόμενο μάθημα.
  - Οριζόμενες από τον χρήστη. Δηλαδή ο χρήστης μπορεί να ορίσει δικούς του τύπους δεδομένων. Θα τις δούμε σε επόμενο μάθημα.



# A. Μεταβλητές
## 3. Δήλωση Μεταβλητών
### 1. Εντολή Δήλωσης Μεταβλητής

- Προτού χρησιμοποιήσουμε μια μεταβλητή, θα πρέπει να την δηλώσουμε.
- Η δήλωση μιας μεταβλητής γίνεται με μια εντολή της μορφής:

```
τυπος_δεδομένων όνομα_μεταβλητής;
```

  - όπου τύπος_δεδομένων κάποιος από αυτούς που είδαμε στην προηγούμενη ενότητα και όνομα_μεταβλητής είναι το όνομα που επιλέγουμε εμείς σεβόμενοι τους κανόνες που έχουμε αναφέρει

- Είναι δυνατό να ορίσουμε και παραπάνω από μία μεταβλητές του ίδιου τύπου σε μία γραμμή, χωρίζοντας τα ονόματα των μεταβλητών με κόμματα:

```
τυπος_δεδομένων  όν_μετ1, ον_μετ2, ον_μετ3;
```

  - Ο παραπάνω κώδικας ορίζει 3 μεταβλητές με τα αντίστοιχα ονόματα και οι μεταβλητές είναι του τύπου δεδομένων.



# A. Μεταβλητές
## 3. Δήλωση Μεταβλητών
### 2. Παραδείγματα

- Το ακόλουθο τμήμα κώδικα:

```
int x,y,z;
float percent, amount;
```

  - Δηλώνει 3 ακέραιες μεταβλητές με ονόματα x,y,z
  - Και 2 αριθμούς κινητής υποδιαστολής μονής ακρίβειας με ονόματα percent, amount
- Επίσης το ακόλουθο τμήμα κώδικα:

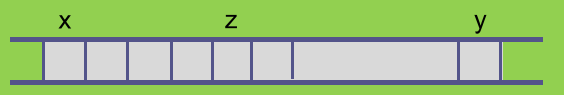
```
int x;
int y;
int z;
float percent, amount;
```

  - Κάνει ακριβώς την ίδια ενέργεια με το παραπάνω τμήμα κώδικα.

Η εντολή δήλωσης μεταβλητής ισοδυναμεί με τη δέσμευση της μνήμης από το μεταγλωττιστή, τη συσχέτιση του χώρου αποθήκευσης με το όνομα της μεταβλητής.

Έτσι η εντολή:
- int x,y,z

αντιστοιχεί στην εικόνα μνήμης:






# A. Μεταβλητές
## 3. Δήλωση Μεταβλητών
### 3. Δήλωση με Αρχικοποίηση

- Είναι σημαντικό να καταλάβουμε ότι μια έντολη δήλωσης:
  - Απλά δεσμεύει τον χώρο για να αποθηκευτεί η μεταβλητή
  - Δεν δίνει τιμή στην μεταβλητή.
- Έτσι αρχικά η μεταβλητή, έχει όπως λέμε «σκουπίδια», δηλαδή έχει ότι περιείχαν τα bytes της προτού δεσμευθεί ο χώρος της.
- Για το λόγο αυτό συχνά είναι χρήσιμο με το που δηλώνουμε μια μεταβλητή να αρχικοποιούμε με κατάλληλη τιμή:
- Π.χ. Η δήλωση

```
int x=5;
```

  - Δηλώνει μια ακέραια μεταβλητή και της αναθέτει την τιμή 5
- Ενώ η δήλωση

```
float y=5.5, z=4.44;
```

  - Δηλώνει δύο πραγματικές μεταβλητές με τιμές 5.5 και 4.44 αντίστοιχα.

# Α. Μεταβλητές
## 3. Δήλωση Μεταβλητών
### 3. Δήλωση με Αρχικοποίηση

- Η δήλωση

```
int x=5;
```

  - Δηλώνει μια ακέραια μεταβλητή x και εκχωρεί σε αυτήν την τιμή 5.

- Αυτό μπορεί εναλλακτικά να γίνει και με τις ακόλουθες γραμμές κώδικα:

```
int x;

x=5;
```

  - Προσοχή! Το = είναι ο τελεστής καταχώρησης (ή τελεστής εκχώρησης) που δίνει στην μεταβλητή που είναι αριστερά του, την τιμή που βρίσκεται δεξιά του.
  - (Δεν έχει λοιπόν καμία σχέση με την γνωστή μαθηματική μας ισότητα)
- Έτσι οι δύο παραπάνω τρόποι είναι ισοδύναμοι μεταξύ τους.



# Α. Μεταβλητές
## 3. Δήλωση Μεταβλητών
### 4. Που δηλώνουμε μεταβλητές

- Στο μάθημα: Εμβέλεια Μεταβλητών, θα δούμε ότι ανάλογα με το που δηλώνουμε τις μεταβλητές επηρεάζεται ο τρόπος που χρησιμοποιούνται από το πρόγραμμα μας.

- Εδώ θα δούμε αρχικά ότι υπάρχουν δύο τύποι μεταβλητών:
  - Οι **καθολικές μεταβλητές** στις οποίες έχουν πρόσβαση και η main και όλες οι συναρτήσεις.
    - Οι καθολικές μεταβλητές, δηλώνονται ακριβώς πριν την main
  - Οι **τοπικές μεταβλητές**, τις οποίες δηλώνει κάποια συνάρτηση και στις οποίες έχουν πρόσβαση μόνο η συνάρτηση.
    - Οι τοπικές μεταβλητές δηλώνονται στο σώμα της συνάρτησης χρήστη( ή της main) αμέσως μετά το άγκιστρο που ανοίγει το σώμα της συνάρτησης χρήστη (ή της main αντίστοιχα)
- Το σχήμα της επόμενης διαφάνειας συνοψίζει τους κανόνες αυτούς



# Α. Μεταβλητές
## 3. Δήλωση Μεταβλητών
### 4. Που δηλώνουμε μεταβλητές

```
#include <stdio.h>

int function(int x); // Το πρωτότυπο μιας συνάρτησης

int g; /* Η g είναι καθολική. Σε αυτήν έχει πρόσβαση και η
                main και η συνάρτηση f */

main()
{
   int m; /* Η m είναι τοπική. Σε αυτήν έχει πρόσβαση μόνο
                η main  */

   ...(Εντολές της main)...
}

int function(int x) // Το σώμα της συνάρτησης
{
   int k; /* Η k είναι τοπική  μεταβλητή. Σε αυτήν έχει
                πρόσβαση μόνο η συνάρτηση f() */

   ...(Εντολές της function)...
}
```



# Α. Μεταβλητές
## 4. Συνώνυμα τύπων δεδομένων
### 1. Η λέξη-κλειδί typedef

- Η C μας δίνει το δικαίωμα να ορίσουμε ένα συνώνυμο ενός τύπου δεδομένων.
- Η εργασία αυτή γίνεται με την typedef που συντάσσεται ως εξής:

```
typedef παλιο_ονομα_ΤΔ νεο_ονομα_ΤΔ;
```

  - Με την εντολή αυτή ορίζουμε ότι ο τύπος δεδομένων με όνομα νεο_ονομα_ΤΔ θα έχει ακριβώς την ίδια συμπεριφορά με τον υπάρχοντα τύπο δεδομένων με όνομα παλιο_όνομα_ΤΔ

- Για παράδειγμα με την εντολή

```
typedef int akeraios;
```

  - Μπορούμε να γράφουμε στο πρόγραμμά μας, αντί για int τον τύπο δεδομένων akeraios, που θα έχει την ίδια συμπεριφορά με το int
- Οι εντολές typedef πρέπει να βρίσκονται ακριβώς μετά τις οδηγίες #include στην αρχή του προγράμματός μας. Μεταγλωττίστε και εκτελέστε το παράδειγμα της επόμενης διαφάνειας που αναδεικνύει την χρήση της typedef.

# A. Μεταβλητές
## 4. Συνώνυμα τύπων δεδομένων
### 1. Η λέξη-κλειδί typedef

```
/* typedef.c: Paradeigma xrisis tis entolis typedef */

#include <stdio.h>

typedef int akeraios;

main()
{
   akeraios x,y,z;

   printf("Dwste enan akeraio: ");
   scanf("%d",&x);
   printf("Dwste akomi enan akeraio: ");
   scanf("%d",&y);
   z=x+y;
   printf("To athroisma toys einai: %d", z);

}
```



# A. Μεταβλητές
## 4. Συνώνυμα τύπων δεδομένων
### 1. Η λέξη-κλειδί typedef

- …τέτοιου τύπου χρήσεις ωστόσο δεν είναι χρήσιμες.
- Τους βασικούς τύπους δεδομένων θα πρέπει να μάθουμε να τους χρησιμοποιούμε ως έχουν.
- Ωστόσο για δευτερεύοντες τύπους δεδομένων (όπως τύπους δεδομένων που καθορίζονται από τον χρήστη), ο ορισμός συνωνύμων μέσω της typedef θα μας φανεί ιδιαίτερα χρήσιμος!



# B. Σταθερές
## 1. Αριθμητικές σταθερές

- Ήδη στα προηγούμενα παραδείγματα, είδαμε πολλές φορές να γράφουμε έναν αριθμό στο πρόγραμμα (προκειμένου π.χ. να τον αποθηκεύσουμε σε μία μεταβλητή)
- Η απεικόνιση αριθμών στην C, είναι η πρώτη κατηγορία σταθερών, οι λεγόμενες αριθμητικές σταθερές, όπου αναφερόμαστε σε έναν αριθμό απλά καταγράφοντας τον.

- Υπάρχουν 3 ειδών **αριθμητικές σταθερές** (δηλαδή αριθμοί) που μπορούμε να γράψουμε στο πρόγραμμα μας:
  - Οι **ακέραιοι αριθμοι**. Όταν γράφουμε έναν ακέραιο αριθμό στο πρόγραμμα μας π.χ. 2, η C καταλαβαίνει ότι πρέπει να αποθηκευτεί προσωρινά σε έναν χώρο αποθήκευσης ακεραίου (int)
  - Οι **πραγματικοί αριθμοί**. Η C αντιλαμβάνεται έναν πραγματικό αριθμό με 2 τρόπους:
    - Βάζοντας τελεία ( . ) για να απεικονίσουμε την υποδιαστολή (π.χ. 3.45)
    - Δίνοντας την επιστημονική μορφή του αριθμού (π.χ. 1.23e6 που απεικονίζει τον αριθμό $1.23 \times 10^6 = 1230000$)



# B. Σταθερές
## 2. Συμβολικές Σταθερές

- Μια συμβολική σταθερά είναι μια σταθερά που έχει ένα όνομα, σε αντίθεση με τις αριθμητικές σταθερές που είναι ένας ξερός αριθμός.
  - Για παράδειγμα αν γράφουμε ένα πρόγραμμα που θα χρησιμοποιήσει τον αριθμό π (π.χ. αν υπολογίζουμε το εμβαδόν ενός κύκλου), θα θέλαμε να απεικονίσουμε τον αριθμό σε μία μεταβλητή
  - Ωστόσο η τιμή αυτής της μεταβλητής δεν πρόκειται να αλλάξει ποτέ στην διάρκεια του προγράμματος.
  - Για το λόγο αυτό θα αποθηκεύσουμε την μεταβλητή αυτή ως σταθερά.

- Υπάρχουν δύο ειδών **συμβολικές σταθερές**, που ορίζονται μέσω:
  - Της οδηγία **#define**
  - Της λέξης κλειδί **const**
  - ….και οι δύο τρόποι χρησιμοποιούνται εξίσου συχνά. Η διαφορά τους θα γίνει πλήρως κατανοητή σε επόμενα μαθήματα, αλλα θα δούμε τώρα ότι:
    - Με την **#define** ορίζουμε σταθερά που την βλέπει υποχρεωτικά **όλο το πρόγραμμα** (όλες οι συναρτήσεις)
    - Με την **const** μπορούμε να ορίσουμε σταθερά που την επεξεργάζεται μόνο **μια** συγκεκριμένη **συνάρτηση.**

# Β. Σταθερές
## 2. Συμβολικές Σταθερές
### 1. Η οδηγία #define

- Μπορούμε να ορίσουμε μια στάθερα μέσω της οδηγία #define ακολουθώντας την σύνταξη:

```
#define ΟΝΟΜΑ_ΣΤΑΘΕΡΑΣ ΤΙΜΗ_ΣΤΑΘΕΡΑΣ
```

  - Για παράδειγμα το π θα το ορίσουμε με την δήλωση:

```
#define PI 3.1415
```

  - Προσοχή! Μετά την δήλωση δεν βάζουμε ερωτηματικό!
  - Οι δηλώσεις #define γράφονται αμέσως μετά τις δηλώσεις #include.
- Η ακριβής λειτουργία της define είναι ότι ο μεταγλωττιστής αναζητά κάθε εμφάνιση της PI στο πρόγραμμα και την αντικαθιστά με την αριθμητική τιμή.
- Κάνει δηλαδή την ίδια ενέργεια με το να πηγαίναμε με το χέρι και να αντικαταστήσουμε τις εμφανίσεις της PI με την συγκεκριμένη τιμή



# Β. Σταθερές
## 2. Συμβολικές Σταθερές
### 2. Η λέξη-κλειδί const

- Μία οποιαδήποτε μεταβλητή μπορούμε να την ορίσουμε ως σταθερά, αν γνωρίζουμε η τιμή της δεν πρόκειται να αλλάξει κατά την διάρκεια εκτέλεσης του προγράμματος.
- Ο καθορισμός ότι η μεταβλητή μετατρέπεται σε σταθερά γίνεται προσθέτοντας την λέξη κλειδί const μπροστά από την δήλωση της μεταβλητής.
- Προσοχή! Μία σταθερά θα πρέπει να αρχικοποιείται οπωσδήποτε κατά την δήλωσή της!
- Έτσι μια ακεραια σταθερά με την τιμή 100 δηλώνεται ως εξής:

```
const int x=100;
```

- Μπορούμε να έχουμε σταθερά οποιουδήποτε τύπου δεδομένων!
- Δήλωση σταθεράς με την λέξη const μπορούμε να έχουμε στα σημεία όπου επιτρέπεται δήλωση μεταβλητών (αρχή συναρτήσεων και πριν την main)



# Β. Σταθερές
## 2. Συμβολικές Σταθερές
### 2. Η λέξη-κλειδί const

- Σε μεγάλα προγράμματα είναι συχνό να κρατάμε άμυνες για να αποφύγουμε λάθη που θα είναι δύσκολο να εντοπιστούν
- Ο ορισμός μεταβλητών ως σταθερών με χρήση της λεξης κλειδί const είναι μια συνηθισμένη πολιτική
  - Πράγματι έστω ότι δηλώνουμε μια σταθερά στην αρχή ενός προγράμματος.
  - Τότε αν επιχειρήσουμε να αλλάξουμε την τιμή της σε επόμενη γραμμή κώδικα, ο μεταγλωττιστής θα διαμαρτυρηθεί!
- Έτσι έχουμε μια άμυνα, αφού ούτε εμείς οι ίδιοι δεν θα μπορέσουμε να αλλάξουμε την τιμή της, όταν γράφουμε τον κώδικα.

- Μελετήστε συστηματικά το πρόγραμμα της επόμενης διαφάνειας. Συγκεντρώνει όλες τις πληροφορίες για την δομή ενός προγράμματος C, ενσωματώνοντας όλα τα χαρακτηριστικά που έχουμε αναφέρει.



```
/* statheres.c: Programma poy deixnei tin xrisi statherwn */

#include <stdio.h> //1.grafoume ta arxeia kefalidas

#define N 100 //2.Grafoume odigies define statherwn

//3. Edw mporoume na orisoume katholikes metavlites

//4. Edw mporoume na orisoyme prwtotipa sinartisewn

main()
{
  //5.1 Dilwsi statherwn kai metavlitwn tis main
  int i,sum;
  const int number=10;

  //5.2 entoles tis main
  sum=0;
  for (i=number; i<=N; i++)
    sum=sum+i;

  printf("To athroisma twn arithmwn [%d..%d] einai %d",number,N,sum);

}
//6. Edw tha exoyme ta swmata twn sinartisewn
```

# Γ. Ασκήσεις

## Εφαρμογή 1

Γράψτε ένα πρόγραμμα C το οποίο:

1. Δηλώνει μία σταθερά με τιμή 100 με χρήση της const
2. Δηλώνει μια ακέραια μεταβλητή
3. Διαβάζει την τιμή της ακέραιας μεταβλητής από το πληκτρολόγιο
4. Εκτυπώνει στην οθόνη τις τιμές της σταθεράς και της μεταβλητής