

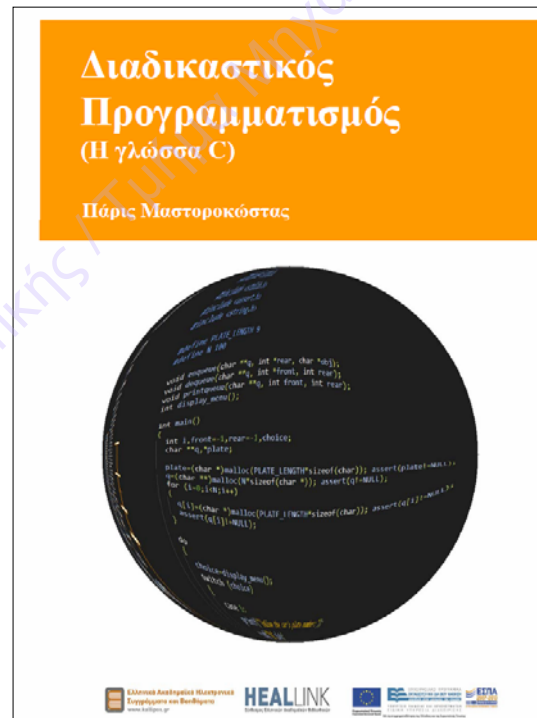
**Διδάσκων: Πάρις Μαστοροκώστας**

**Ώρες επικοινωνίας (μέσω TEAMS): Μετά από συνεννόηση**

**E-mail: [mast@uniwa.gr](mailto:mast@uniwa.gr)**

**URL: <http://users.uniwa.gr/mast>**

**Διδακτικό βοήθημα:**



# ***Εισαγωγή – Βασικά στοιχεία προγράμματος***

## Τι είναι ο υπολογιστής;

- Οι κανόνες αποτελούν την καρδιά της επιστήμης, της κοινωνίας και της ισχύος σε οιαδήποτε μορφή.
- Ο υπολογιστής είναι *μία ταχεία, ακούραστη μηχανή που ακολουθεί κανόνες.*

## Τι είναι ο προγραμματισμός

- **Πρόγραμμα:** Ακολουθία εντολών, με τις οποίες ο υπολογιστής εκτελεί μία συγκεκριμένη εργασία και επιλύει ένα δοθέν πρόβλημα.
- **Προγραμματισμός:** Κατάστρωση και συγγραφή προγραμμάτων.

## Τι θα μάθετε

- 1) Να λύνετε προβλήματα που περιγράφονται από κανόνες: γράφοντας σύνολα κανόνων (συναρτήσεις), οι οποίοι εφαρμόζονται σε σύνολα δεδομένων (δομές), χρησιμοποιώντας ταυτόχρονα και έτοιμα προγράμματα (βιβλιοθήκες)
- 2) Θεμελιώδεις έννοιες, κοινές σε όλες τις γλώσσες προγραμματισμού
- 3) Προχωρημένες έννοιες του δομημένου προγραμματισμού
- 4) Εξοικείωση με ένα αποδοτικό Εργαλείο Προγραμματισμού: Το περιβάλλον προγραμματισμού DEV C++

## Προγραμματισμός: Κοινή λογική και καλλιτεχνία

- 1) **Ανάλυση:** (εύρεση του πυρήνα του προβλήματος)
  - Καθόρισε τις συγκεκριμένες εισόδους και εξόδους
  - Καθόρισε τη σχέση εισόδου - εξόδου
  - Τεμάχισε το πρόβλημα σε υποπροβλήματα
- 2) **Υλοποίηση:** Γράψε το σύνολο κανόνων για κάθε υποπρόβλημα
- 3) **Debug:** Έλεγξε κάθε τμήμα ξεχωριστά. Στη συνέχεια σύνδεσέ τα και έλεγξέ τα, έως ότου το συνολικό πρόγραμμα λειτουργήσει σωστά.

## Ο πυρήνας ενός προβλήματος

**Παράδειγμα:** Να γραφεί πρόγραμμα που υπολογίζει το εμβαδόν ενός σπιτιού, με δεδομένες τις διαστάσεις κάθε δωματίου.

– **Είσοδοι:**

- Ο αριθμός των δωματίων (ακέραιος αριθμός)
- Οι διαστάσεις κάθε δωματίου (πραγματικοί αριθμοί)

– **Έξοδοι:**

- Το εμβαδόν του σπιτιού (πραγματικοί αριθμοί)

## Ανάλυση: Βρείτε τον πυρήνα...

- Καταστείτε σίγουροι ότι καταλάβατε πραγματικά το πρόβλημα!
  - Απλοποιείστε το, διευκρινίστε τα σκοτεινά σημεία, σκεφθείτε. Επαναλάβετε.
  - Τεμαχίστε το σε υποπροβλήματα.
  - Οι πρώτες σκέψεις δεν είναι ΠΟΤΕ οι καλύτερες.
- Παράδειγμα σπιτιού:
  - **Υποπρόβλημα 1:** υπολογισμός του εμβαδού κάθε δωματίου
  - **Υποπρόβλημα 2:** πρόσθεση των εμβαδών των δωματίων

## Ανάλυση: Βρείτε τον πυρήνα...

- Εργαλεία που βοηθούν στην περιγραφή του προβλήματος
  - Φυσική γλώσσα: “κάνε αυτό, στη συνέχεια εκείνο, εκτός εάν, κ.λ.π.”
  - Διάγραμμα ροής
  - Ψευδοκώδικας
- Παράδειγμα: Να μετατραπούν οι βαθμοί Fahrenheit σε βαθμούς Κελσίου

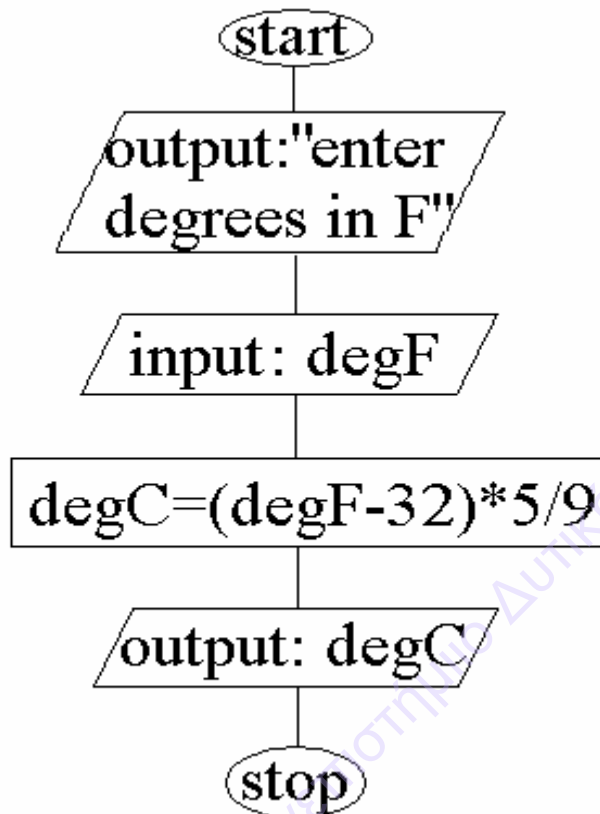





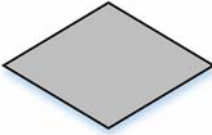

## Εργαλείο ανάλυσης: η φυσική γλώσσα

- Ζήτησε από τον χρήστη τη θερμοκρασία σε βαθμούς  $F$ .
- Διάβασε την τιμή που δίνει ο χρήστης.
- Αποθήκευσε την τιμή σε θέση αποθήκευσης που καλείται **degF**.
- Υπολόγισε τους βαθμούς  $C$  με χρήση μαθηματικής σχέσης.
- Αποθήκευσε το αποτέλεσμα σε θέση αποθήκευσης που καλείται **degC**.
- Τύπωσε το περιεχόμενο της **degC**.

## Εργαλείο ανάλυσης: το διάγραμμα ροής

Προτιμητέο εργαλείο για σύνθετα προβλήματα με περιορισμούς



Σύμβολο	Λειτουργία
	Αρχή/Τέλος
	Είσοδος/έξοδος δεδομένων
	Επεξεργασία
	Έλεγχος
	Μεταφορά ελέγχου ροής

## Εργαλείο ανάλυσης: ο ψευδοκώδικας

```
print "enter degrees in Farenheit"  
read degF  
degC = (degF - 32) * 5 / 9  
print degC
```

## Υλοποίηση

Μετατροπή του συνόλου κανόνων σε συντακτικό της γλώσσας C.

Ο τρόπος μετατροπής εξάγεται διερευνώντας:

Μεταβλητές, τύπους δεδομένων, εκφράσεις, υποθετικές προτάσεις και προτάσεις ελέγχου ροής, συναρτήσεις κ.ά.

## Χαρακτηριστικά της γλώσσας C

- 1) Μπορεί να χρησιμοποιηθεί και ως γλώσσα προγραμματισμού χαμηλού επιπέδου, επιτρέποντας άμεση πρόσβαση στους πόρους του υπολογιστή.
- 2) Είναι σχετικά μικρή και εύκολη στην εκμάθηση.
- 3) Υποστηρίζει δομημένο προγραμματισμό.
- 4) Είναι αποτελεσματική, παράγοντας συμπαγή και γρήγορα στην εκτέλεση προγράμματα.
- 5) Δεκαετίες μετά την επινόησή της αποτελεί μία από τις ευρύτερα χρησιμοποιούμενες γλώσσες σε ερευνητικά και αναπτυξιακά προγράμματα, γεγονός που έχει δημιουργήσει μία πολλή μεγάλη εγκατεστημένη βάση εφαρμογών που αναπτύχθηκαν με αυτές τις γλώσσες και πρέπει να συντηρούνται και να εξελίσσονται.

## Δημοφιλία της γλώσσας C

Oct 2020	Oct 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.95%	+0.77%
2	1	▼	Java	12.56%	-4.32%
3	3		Python	11.28%	+2.19%
4	4		C++	6.94%	+0.71%
5	5		C#	4.16%	+0.30%
6	6		Visual Basic	3.97%	+0.23%
7	7		JavaScript	2.14%	+0.06%
8	9	▲	PHP	2.09%	+0.18%
9	15	▲▲	R	1.99%	+0.73%
10	8	▼	SQL	1.57%	-0.37%

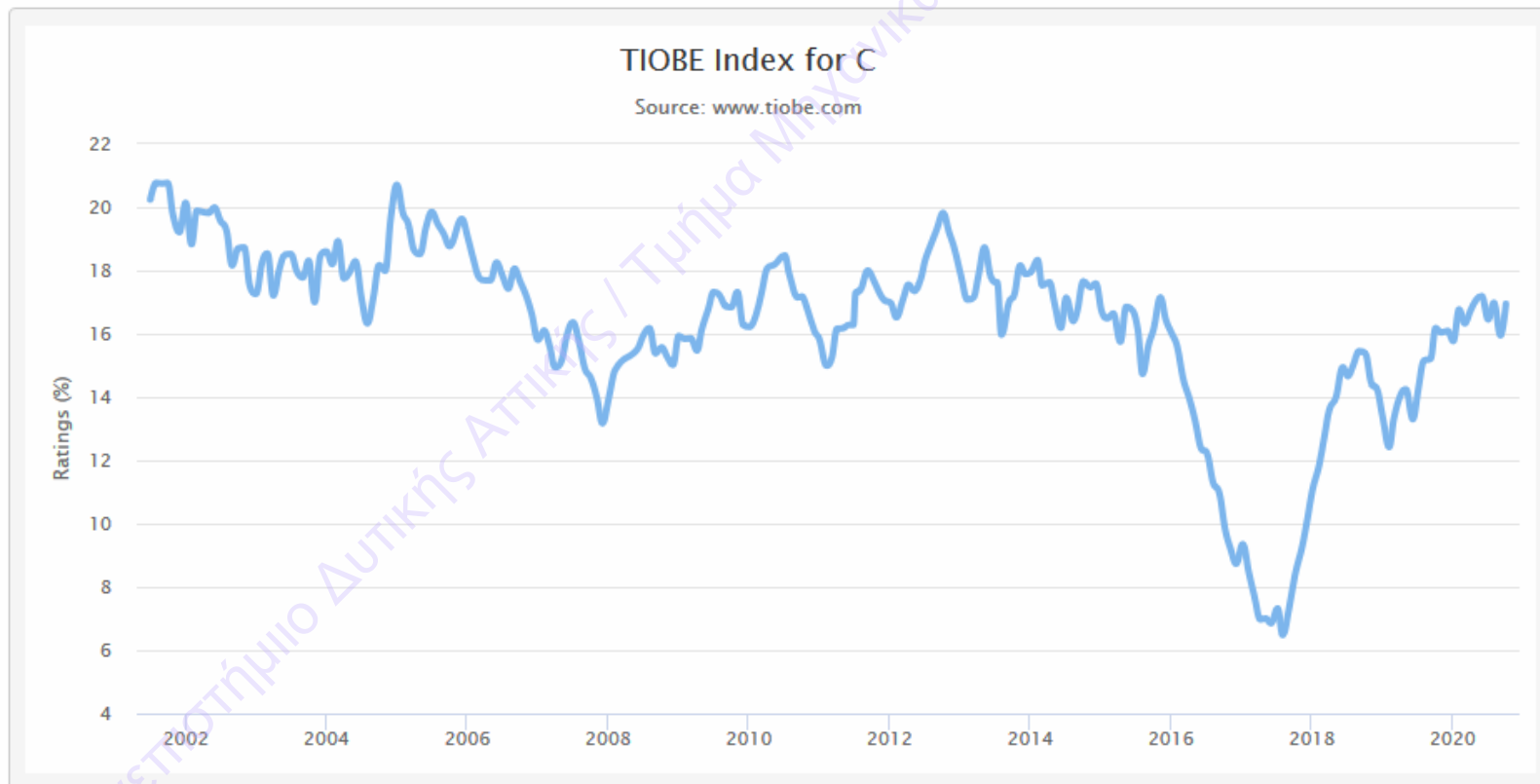
## The C Programming Language

Some information about C:

📈 Highest Position (since 2001): #1 in Oct 2020

📉 Lowest Position (since 2001): #2 in Apr 2020

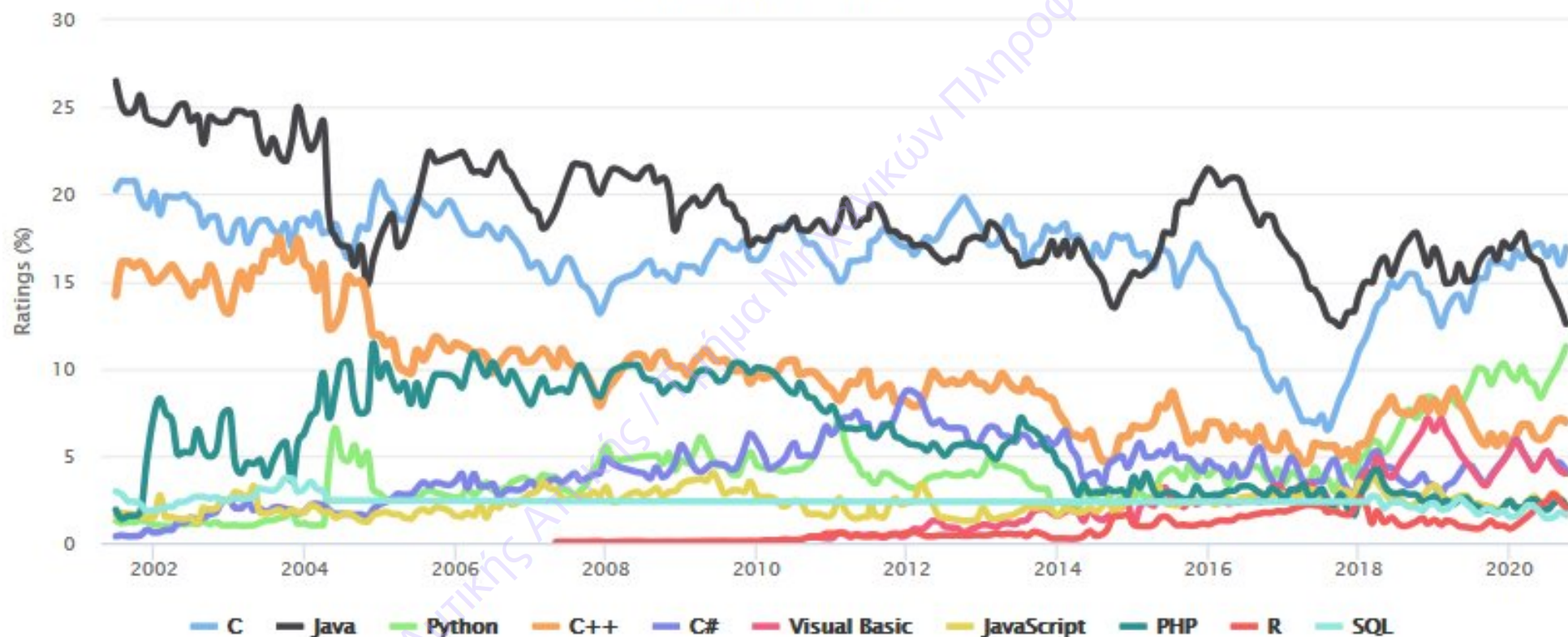
🏆 Language of the Year: 2008, 2017, 2019



# Διαχρονικά ηγεμονική θέση της γλώσσας C

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)





## Ιστορική αναδρομή της γλώσσας C

- Η C επινοήθηκε το 1972 από τον Dennis Ritchie, στα εργαστήρια Bell. Δημιουργήθηκε για να εξυπηρετήσει το λειτουργικό σύστημα Unix, το οποίο έως τότε ήταν γραμμένο σε assembly.
- Ο δημιουργός του Unix, Ken Thompson, φίλος και συνεργάτης του Ritchie, είχε δημιουργήσει την πρόγονο της C, τη γλώσσα B. Και οι δύο γλώσσες έχουν κοινή καταγωγή από τη γλώσσα BCPL, η οποία είχε αναπτυχθεί από τον Martin Richards κατά το πέρασμά του από το MIT το 1967, στηριζόμενη στη γλώσσα CPL (Cambridge Programming Language) του πανεπιστημίου του Cambridge.
- Και οι τρεις γλώσσες κατασκευάστηκαν στο πλαίσιο του προγράμματος MAC και του απογόνου του Multics, τα οποία στόχευαν στην κατανομή των πόρων των υπολογιστών σε πολλούς χρήστες. Τα δύο αυτά προγράμματα, στα οποία συνέπραξαν το MIT, η General Electric και τα εργαστήρια Bell, αποτέλεσαν τη θερμοκοιτίδα πολλών προγραμμάτων λογισμικού, που κυριαρχούν από τη δεκαετία του 1960 έως σήμερα.



## Ιστορική αναδρομή της γλώσσας C

- Η C, ούσα ευέλικτη και αποδοτική, χρησιμοποιήθηκε αρχικά για τον προγραμματισμό συστημάτων στο Unix.
- Το 1974 εμφανίσθηκε από τον Brian Kernighan το πρώτο γραπτό κείμενο για τη γλώσσα, υπό τον τίτλο “Programming in C: A Tutorial”.
- Το 1977 έγινε η πρώτη επίσημη τεκμηρίωση της γλώσσας με το βιβλίο “The C Programming Language” από τους Kernighan και Ritchie. Το βιβλίο αυτό αποτέλεσε το «ευαγγέλιο» των προγραμματιστών της C, αποκαλούμενο «Λευκή Βίβλος» ή «πρότυπο K&R».
- Με την πάροδο του χρόνου η γλώσσα C άρχισε να χρησιμοποιείται και σε άλλα πεδία εφαρμογών, πέραν του προγραμματισμού συστημάτων. Η εμφάνιση των μεταγλωττιστών της γλώσσας στο MS-DOS και ο μεγάλος αριθμός προγραμμάτων βιβλιοθήκης που κατασκευάστηκαν, οδήγησαν τη γλώσσα στο απόγειό της στα τέλη της δεκαετίας του 1980.



## Αναθεωρήσεις της γλώσσας C

- Η γλώσσα γνώρισε πολλές αλλαγές, οδηγούμενη τελικά το 1989 στην επονομαζόμενη ANSI έκδοση, που την προτυποποίησε (το όνομά της το έλαβε από την επιτροπή του American National Standards Institute).
- Μία σημαντική ανανέωση έγινε το 1999 με το πρότυπο C99.
- Μία αναθεώρηση με πολλά καινοτόμα στοιχεία, όπως η προτυποποίηση του πολυνηματικού προγραμματισμού, έλαβε χώρα το 2011 με το πρότυπο C11.
- Η τελευταία αναθεώρηση έγινε το 2017-2018 με το πρότυπο C18.
- Το 2021 αναμένεται η επόμενη αναθεώρηση, με τον προσωρινό τίτλο C2X.

## Μεταγλωττιστής και Συνδέτης (Compiler & Linker)

- Η C είναι **μεταγλωττισμένη** γλώσσα, με στόχο να είναι αναγνώσιμη και κατανοητή.
- **‘Compiler’**: πρόγραμμα μετατροπής αναγνώσιμο → εκτελέσιμο από τον Η/Υ
- **Compile**: C αρχεία κειμένου → **.object** file(s)
- **Linking**: object file(s) → **.executable** file

## Διαδικασία αποσφαλμάτωσης (Debugging)

- ‘bug’ = σφάλμα
- ‘debug’ = εύρεση και διόρθωση σφαλμάτων

Δύο είδη προγραμματιστικών σφαλμάτων:

**Συντακτικά σφάλματα:** Σφάλματα που οφείλονται σε παραβίαση των συντακτικών κανόνων (π.χ. λανθασμένη γραφή μίας εντολής). Ανιχνεύονται από τον μεταγλωττιστή κατά τον χρόνο μεταγλώττισης και αναφέρονται υπό μορφή λίστας (οπότε μπορούν να διορθωθούν προτού τρέξει το πρόγραμμα).

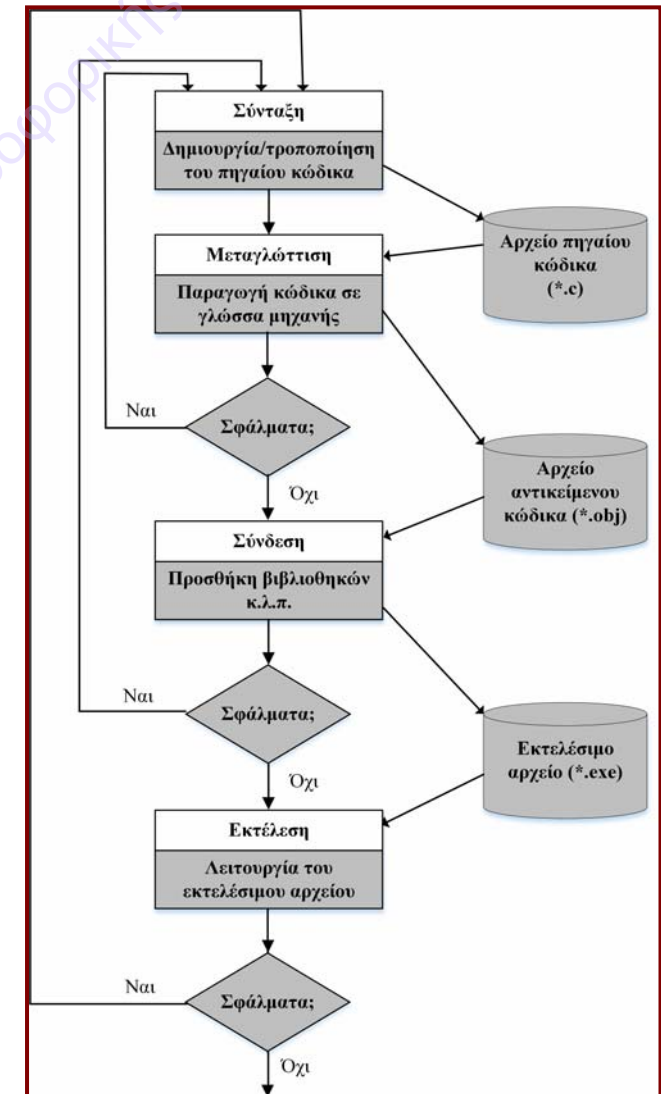
**Σημασιολογικά σφάλματα (semantic errors):** Σφάλματα που οφείλονται σε εσφαλμένη σχεδίαση της λύσης του προβλήματος (π.χ. διαίρεση με το μηδέν). Τα σφάλματα αυτής της κατηγορίας δεν αναγνωρίζονται από τον μεταγλωττιστή, εμφανίζονται αργότερα στο χρόνο εκτέλεσης και είναι δυσκολότερα στη διόρθωση, καθώς απαιτούνται μεταβολές στη σχεδίαση και ακολούθως στον κώδικα).

## Debug: Εύρεση και διόρθωση σφαλμάτων

Κώδικας C → Μεταγλώττιση → Εκτέλεση, επανάληψη

- Γίνεται η μεταγλώττιση;
  - ΟΧΙ → ο μεταγλωττιστής αναφέρει τους λόγους. Τροποποίησε τον κώδικα και δοκίμασε εκ νέου
- ΝΑΙ. Λειτουργεί σωστά;
  - ΟΧΙ → ανάλυσε, τροποποίησε, δοκίμασε εκ νέου
- ΝΑΙ. Αλλά δουλεύει πάντοτε σωστά;

*Μπορείς να αποδείξεις ότι η ανάλυση είναι εύρωστη;*



***IDE DEV C++***

## Ένα απλό πρόγραμμα σε C

```
/* *****  
 * This program prints out the sentence "This is a test." *  
***** */  
  
#include<stdio.h>  
  
int main()  
{  
    printf("This is a test.\n");  
    return 0;  
}
```



## Ένα απλό πρόγραμμα σε C

```
/*
```

This program prints out the sentence “This is a test.”

```
*/
```

```
#include<stdio.h>
```

```
int main ()
```

```
{
```

```
    printf(“This is a test.\n”);
```

```
    return 0;
```

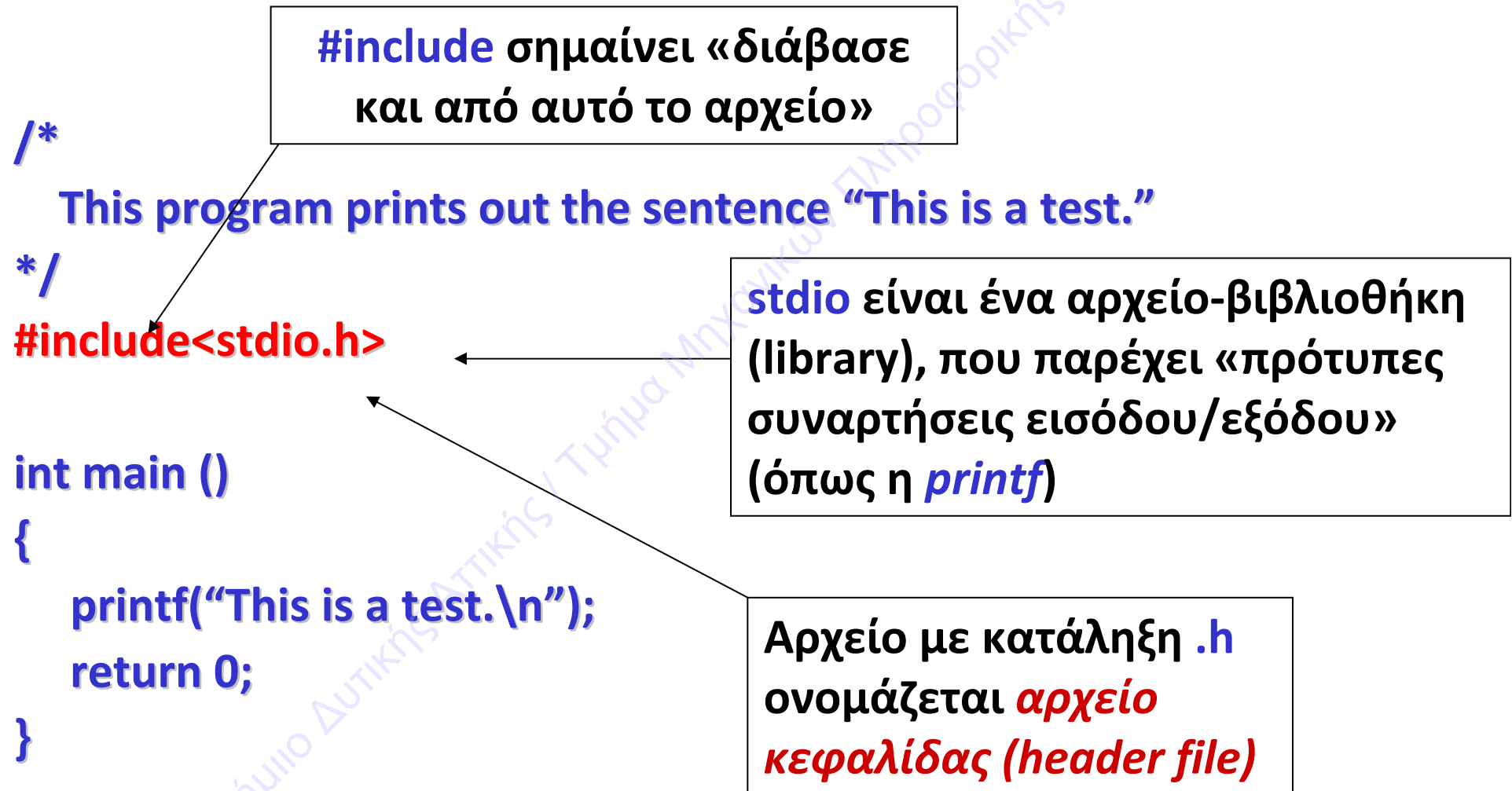
```
}
```

Το **Σχόλιο (comment)** είναι κείμενο ανάμεσα σε **/\*** και **\*/**.

Να χρησιμοποιείτε συχνά σχόλια για να επεξηγείτε το πρόγραμμά σας και τα τμήματά του.

**Ο μεταγλωττιστής δε λαμβάνει υπόψη τα σχόλια.**

## Ένα απλό πρόγραμμα σε C



## Ένα απλό πρόγραμμα σε C

```
/*
```

This program prints out the sentence “This is a test.”

```
*/
```

```
#include<stdio.h>
```

```
int main ()
```

```
{
```

```
    printf(“This is a test.\n”);
```

```
    return 0;
```

```
}
```

Η εκτέλεση του προγράμματος αρχίζει πάντοτε από τη συνάρτηση *main()*.

ΟΛΑ τα προγράμματα σε C έχουν συνάρτηση *main()*, η οποία συνήθως καλεί άλλες συναρτήσεις.

## Ένα απλό πρόγραμμα σε C

```
#include<stdio.h>
```

```
int main ()
```

```
{
```

```
printf("This is a test.\n");
```

```
return 0;
```

```
}
```

Όλες οι συναρτήσεις χρησιμοποιούν **άγκιστρα** για να σημειώσουν την αρχή και το τέλος της συνάρτησης.

Το τμήμα ανάμεσα στα άγκιστρα ονομάζεται **σώμα (body)** της συνάρτησης.

## Ένα απλό πρόγραμμα σε C

```
#include<stdio.h>
```

```
int void main ()
```

```
{
```

```
printf("This is a test.\n");
```

```
return 0;
```

```
}
```

Μία συνάρτηση είναι ένα σύνολο προτάσεων με ένα δεδομένο όνομα, π.χ. *main()*, *printf()*.

Η πρόταση αυτή **καλεί** τη συνάρτηση *printf()* για να τυπώσει το καθορισμένο κείμενο.

Τα **ορίσματα εισόδου (input arguments)** περικλείονται από παρενθέσεις και προσδιορίζουν το προς εκτύπωση κείμενο και τη μορφή με την οποία θα εκτυπωθεί.

## Ένα απλό πρόγραμμα σε C

```
#include<stdio.h>
```

```
int main ()
```

```
{
```

```
    printf("This is a test.\n");
```

```
    return 0;
```

```
}
```

ΟΛΕΣ οι προτάσεις  
τελειώνουν με  
ερωτηματικό (semicolon) ;  
το οποίο ονομάζεται  
**σύμβολο τερματισμού  
πρότασης.**

Λεπτομέρειες της **printf()**:

**\n** σημαίνει «μετακινήσου στην επόμενη γραμμή».  
Ονομάζεται **ακολουθία διαφυγής (escape sequence).**

Η έξοδος στην οθόνη είναι:

```
>This is a test.
```

```
>
```

## Σύνταξη σχολίων

Τα σχόλια πρέπει να χρησιμοποιούνται αφειδώς γιατί καταστούν τον κώδικα ευανάγνωστο και συνεισφέρουν στην επεξήγηση δυσνόητων σημείων.

`/* Το /* σχόλιο */ αυτό είναι λανθασμένο */`

`/* Το σχόλιο αυτό χρησιμοποιεί σωστή σύνταξη */`

`/*`

Ομοίως

και

αυτό.

`*/`

## Σύνταξη σχολίων

`/* Το σχόλιο /* αυτό */ είναι λανθασμένο */`

σχόλιο

κώδικας;

**Να ευθυγραμμίζετε τα σύμβολα των σχολίων και να μην τοποθετείτε ποτέ σχόλια μέσα σε σχόλια (φώλιασμα, *nesting*), γιατί μερικοί μεταγλωττιστές θα μπερδευτούν.**



### Παρατηρήσεις:

- Η γλώσσα C διαχωρίζει τα κεφαλαία γράμματα από τα μικρά (case sensitive). Η εντολή `Printf()` ΔΕΝ είναι ίδια με την `printf()`.  
Όλες οι εντολές στη C γράφονται με μικρά γράμματα!
- Η σωστή στηλοθεσία είναι πολύ σημαντική, καθώς καθιστά τον κώδικα ευανάγνωστο.
- Να γράφετε πάντοτε σχόλια στα προγράμματά σας.
- Η εντολή `printf()` ανήκει στις **μορφοποιούμενες συναρτήσεις εισόδου-εξόδου**. Ονομάζεται μορφοποιούμενη γιατί δίνει τη δυνατότητα στον χρήστη να μορφοποιήσει την έξοδό της, δυνάμενη να εκτυπώσει μεταβλητές διαφόρων τύπων και με διάφορους τρόπους, χρησιμοποιώντας κατάλληλα σύμβολα. Δυαδική της `printf()` είναι η `scanf()`, η οποία λαμβάνει πληροφορία από την είσοδο (πληκτρολόγιο).

### Παρατηρήσεις (συνέχεια):

- Η πρόταση `printf( "This is a test.\n" );` καλεί την `printf()` για να τυπωθεί το καθορισμένο κείμενο. Τα **ορίσματα εισόδου** (input arguments) περικλείονται από παρενθέσεις και προσδιορίζουν το προς εκτύπωση κείμενο και τη μορφή με την οποία θα εκτυπωθεί. Τέλος, το σύμβολο `\n`, που ανήκει στις **ακολουθίες διαφυγής**, σημαίνει «μετακινήσου στην επόμενη γραμμή». Λεπτομερής περιγραφή της λειτουργίας των συναρτήσεων εισόδου – εξόδου δίνεται στη συνέχεια.
- Πέραν της `include`, μία σημαντική οδηγία προς τον προεπεξεργαστή είναι η `define`, η οποία αντιστοιχίζει ένα όνομα με μία σταθερά ή με μία σειρά χαρακτήρων. Οποτεδήποτε εμφανίζεται το όνομα μέσα στον κώδικα, αντικαθίσταται αυτόματα με την τιμή της σταθεράς ή τη συμβολοσειρά.



Για παράδειγμα, εάν χρησιμοποιηθεί η λέξη **TRUE** στη θέση της τιμής **1** και η λέξη **FALSE** στη θέση της τιμής **0**, θα δοθούν δύο *define* ως εξής:

```
#define TRUE 1
```

```
#define FALSE 0
```

Εάν αντικατασταθεί ολόκληρη φράση, μπορεί να εμφανισθεί στην οθόνη με χρήση της *printf()*:

```
#define TITLOS "Dpt of Informatics and Computer Engineering\n"  
printf( TITLOS );
```

Το αποτέλεσμα είναι:

**Dpt of Informatics and Computer Engineering**

- Μία συνηθισμένη χρήση της *define* είναι για τον καθορισμό του μεγέθους στοιχείων, όπως είναι η διάσταση ενός πίνακα, τα οποία μπορεί να αλλάξουν κατά την εκτέλεση του προγράμματος.

## Λεξιλόγιο της γλώσσας C

- Δεσμευμένες λέξεις (*reserved words*)
- Λέξεις κλειδιά (*keywords*)
- Τελεστές (*operators*)
- Αναγνωριστές (*identifiers*)

## Δεσμευμένες λέξεις:

**πρέπει να αποφεύγεται η χρήση τους ως ονόματα**

- **Ονόματα συναρτήσεων** πρότυπης βιβλιοθήκης (runtime function names), όπως `printf()`, `abs()` κ.λ.π.
- **Macro names**. Είναι ονόματα που περιέχονται σε αρχεία κεφαλίδας για ορισμό μακροεντολών, π.χ. `EOF`, `INT_MAX`.
- **Type names**. Είναι ονόματα τύπων σε ορισμένα αρχεία κεφαλίδας, π.χ. `time_t`, `va_list`.
- **Ονόματα εντολών προεπεξεργαστή** (preprocessor). Είναι ονόματα που χρησιμοποιεί ο προεπεξεργαστής της C και έχουν προκαθορισμένη σημασία, π.χ. `include`, `define`.
- **Ονόματα που αρχίζουν με το χαρακτήρα υπογράμμισης `_` και έχουν δεύτερο χαρακτήρα τον ίδιο ή κεφαλαίο γράμμα**, π.χ. `_DATE_`, `_FILE_`.

**Λέξεις κλειδιά:** Λεκτικές μονάδες που μόνες τους ή με άλλες λεκτικές μονάδες χαρακτηρίσουν κάποια γλωσσική κατασκευή. Π.χ. **int**: αναπαριστά τον ακέραιο τύπο δεδομένων **if-else**: χρησιμοποιούνται στον έλεγχο ροής προγράμματος.

- Οι λέξεις κλειδιά, αν και είναι ένας – σχετικός – περιορισμός των γλωσσών, αυξάνουν την αναγνωσιμότητα και αξιοπιστία των προγραμμάτων, ενώ ταυτόχρονα επιταχύνουν τη διαδικασία της μεταγλώττισης.
- Λέξεις κλειδιά όπως **if**, **else**, **for**, **case**, **while**, **do** έχουν γίνει κοινά αποδεκτές, διευκολύνοντας την εκμάθηση των γλωσσών προγραμματισμού.

## Λέξεις κλειδιά στην ANSI C:

<b>auto</b>	<b>else</b>	<b>register</b>	<b>union</b>
<b>break</b>	<b>enum</b>	<b>return</b>	<b>unsigned</b>
<b>case</b>	<b>extern</b>	<b>short</b>	<b>void</b>
<b>char</b>	<b>float</b>	<b>signed</b>	<b>volatile</b>
<b>const</b>	<b>for</b>	<b>sizeof</b>	<b>while</b>
<b>continue</b>	<b>goto</b>	<b>static</b>	
<b>default</b>	<b>if</b>	<b>struct</b>	
<b>do</b>	<b>int</b>	<b>switch</b>	
<b>double</b>	<b>long</b>	<b>typedef</b>	

**Αναγνωριστές:** Λεκτικές μονάδες που κατασκευάζει ο προγραμματιστής. Αυτές οι λεκτικές μονάδες χρησιμοποιούνται συνήθως ως ονόματα που ο προγραμματιστής δίνει σε δικές του κατασκευές, όπως μεταβλητές, σταθερές, συναρτήσεις και δικούς του τύπους δεδομένων. Ένα όνομα προσδιορίζει μοναδιαία (uniquely identifies), από το σύνολο των κατασκευών του προγράμματος, την κατασκευή στην οποία αποδόθηκε, εξ ου και το όνομα αναγνωριστής (identifier).



## Κανόνες δημιουργίας ευανάγνωστου προγράμματος

- Αποφύγετε ονόματα ενός χαρακτήρα, όπως *i*, *j*, *x*, *y* (εκτός από ειδικές περιπτώσεις που θα εξετασθούν αργότερα).
- Χρησιμοποιείτε εκφραστικά ονόματα. Συγκεκριμένα:
  - Ονομάστε τη μεταβλητή που αναπαριστά την ταχύτητα ως *velocity* και τη μέγιστη τιμή της *max\_velocity* ή *maxVelocity*.
  - Ονομάστε τη συνάρτηση που εμφανίζει τα λάθη στην οθόνη *display\_error* ή *displayError*.
- Για καλύτερη αναγνωσιμότητα των μεταβλητών που αποτελούνται από δύο ή περισσότερες λέξεις, αποφασίστε αν θα χρησιμοποιείτε τη μορφή *display\_error* ή τη μορφή *displayError*. Τηρείστε τη σύμβαση σε όλο το πρόγραμμα.
- Χρησιμοποιείτε μικρά γράμματα για ονόματα μεταβλητών.

# **Θεματική ενότητα 2:** **Μεταβλητές – σταθερές – I/O** **κονσόλας**

## Μεταβλητές

- **Ίδια χρήση** με εκείνη της άλγεβρας:  $3x + 5 = y$   
 $x$  και  $y$  είναι οι μεταβλητές
- **Αλλά γενικευμένη**: η μεταβλητή είναι μία θέση μνήμης για ένα δεδομένο. Η τιμή της μπορεί να είναι άγνωστη έως ότου εκτελεσθεί το πρόγραμμα ('run-time').

## Δήλωση μεταβλητών

- Με πρόταση ορισμού (τελειώνει πάντοτε με ;)
  - Η μορφή: `data_type var, var, ... ;`
  - Παράδειγμα: `int counter1, counter2;`
- Πού; Οι μεταβλητές δηλώνονται στην αρχή μίας συνάρτησης, αμέσως μετά το εισαγωγικό άγκιστρο {

- Στην C τα ονόματα των μεταβλητών σχηματίζονται από:
  - τα γράμματα του αλφαβήτου
  - τα ψηφία 0 έως 9
  - τον χαρακτήρα υπογράμμισης  (underscore)
- Το όνομα πρέπει να ξεκινά με γράμμα ή τον χαρακτήρα υπογράμμισης (στη δεύτερη περίπτωση ο επόμενος χαρακτήρας πρέπει να είναι μικρό γράμμα).
- Το όνομα δεν πρέπει να είναι ίδιο με δεσμευμένη λέξη.
- Σημαντικοί είναι μόνο οι πρώτοι 31 χαρακτήρες του ονόματος. Οι υπόλοιποι δε λαμβάνονται υπόψη.