Προγραμματισμός Ι
Εισαγωγή

Πανεπιστήμιο Πελοποννήσου
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Νικόλαος Δ. Τσελίκας

Σύντομη Αναδρομή

- **Αναδρομή**
  - Η γλώσσα προγραμματισμού C είναι μία γλώσσα υψηλού επιπέδου
  - Δημιουργήθηκε από τον Dennis Richie στις αρχές της δεκαετίας του 1970 στα εργαστήρια Bell Labs της εταιρείας AT&T
  - Η γλώσσα ονομάστηκε C, γιατί πολλά από τα χαρακτηριστικά της προήλθαν από μια παλαιότερη γλώσσα, η οποία είχε αναπτυχθεί από τον Ken Thompson και ονομαζόταν "B"
- **Γιατί δημιουργήθηκε η C ?**
  - Ο κύριος σκοπός της δημιουργίας της γλώσσας C ήταν η χρήση της για την ανάπτυξη του λειτουργικού συστήματος UNIX

Το πρότυπο ANSI (I)

- **Το πρόβλημα:**
  - Η Γρήγορη και Άμεση Αποδοχή της C είχε σαν αποτέλεσμα
    - διάφοροι οργανισμοί και εταιρείες να δημιουργούν τις δικές τους εκδόσεις της γλώσσας, οι οποίες όμως εμφάνιζαν διαφορές μεταξύ τους
    - η ύπαρξη πολλών εκδόσεων είχε σαν συνέπεια να εμφανιστούν προβλήματα ασυμβατότητας, δηλαδή προγράμματα τα οποία να μεταγλωττίζονται επιτυχώς με μία έκδοση να μην μεταγλωττίζονται με κάποια άλλη
- **Η λύση (1983):**
  - ANSI (American National Standard Institute)
    - ορισμός επιστημονικής επιτροπής για την καθιέρωση ενός προτύπου της C, το οποίο να καθορίζει πλήρως τους κανόνες, τα χαρακτηριστικά και τη λειτουργικότητα της γλώσσας

Το πρότυπο ANSI (II)

- **ANSI C**
  - Το πρότυπο ολοκληρώθηκε το 1989 και ονομάζεται ANSI C
  - Από τότε όλοι οι C μεταγλωττιστές υποστηρίζουν το συγκεκριμένο πρότυπο
- **C99**
  - Στο τέλος της δεκαετίας του '90 το πρότυπο ANSI C επανεξετάστηκε
  - Οι αλλαγές που έγιναν οδήγησαν στη δημιουργία ενός νεότερου προτύπου, γνωστού ως C99
  - Οι περισσότεροι μεταγλωττιστές υποστηρίζουν το μεγαλύτερο τμήμα του, αν όχι όλο
- **C11, C17, C2x**
  - Πρόκειται για εμπλουτισμό του προτύπου C99, το 2011, το 2017 και μετά το 2020...
- **Αποτέλεσμα:**
  - Το παλαιότερο πρότυπο (ANSI C) παραμένει το επικρατέστερο μέχρι και σήμερα

Πλεονεκτήματα της γλώσσας C

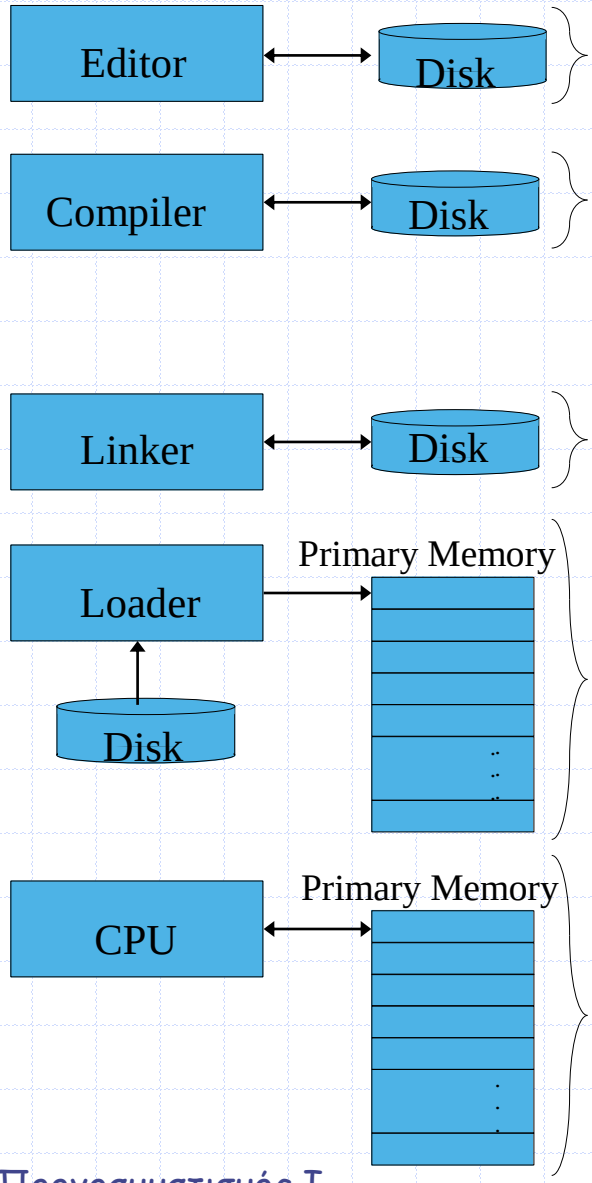
- Πολύ γρήγορη
- Απλό συντακτικό και λίγες δεσμευμένες λέξεις
- Φορητότητα κώδικα (εκτελείται σε διαφορετικά λειτουργικά συστήματα)
- Υποστηρίζει τον δομημένο προγραμματισμό και την ύπαρξη και κλήση συναρτήσεων
- Υποστηρίζει μέχρι και την ενσωμάτωση Assembly κώδικα
- Παρέχει έτοιμες συναρτήσεις
- Δημοφιλής γλώσσα (παρά την ηλικία της)
- Αποτελεί το πρώτο βήμα για την εκμάθηση κάποιας αντικειμενοστρεφούς γλώσσας προγραμματισμού (π.χ. Java, C++, C# κτλ)

Μειονεκτήματα της γλώσσας C

- **Έλλειψη περιορισμών**
  - Επομένως, ο προγραμματιστής πρέπει να είναι πολύ προσεκτικός κατά τη συγγραφή του προγράμματος, γιατί μπορεί εύκολα να εισάγει λάθη, τα οποία ο μεταγλωττιστής δεν θα αντιληφθεί
- **«Δύσκολη» (συντακτικά) γλώσσα**
  - Η C, αν και είναι μία μικρή γλώσσα (υπό την έννοια ότι περιέχει λίγες λέξεις με ειδική σημασία), δεν είναι μία «εύκολη» γλώσσα προγραμματισμού για να μάθει κάποιος
  - Ένα C πρόγραμμα μπορεί να γραφεί με αρκετά πολύπλοκο τρόπο ακόμα και αν αποτελείται από λίγες γραμμές κώδικα
- **Δεν είναι αντικειμενοστρεφής γλώσσα**
  - Άρα δεν υποστηρίζει λειτουργίες, όπως η ενθυλάκωση, ο πολυμορφισμός, η κληρονομικότητα κ.ά.

Κύκλος δημιουργίας ενός προγράμματος C

- **Συγγραφή κώδικα (Edit)**
- **Μεταγλώττιση (Compile)**
- **Σύνδεση (Link)**
- **Φόρτωση (Load)**
- **Εκτέλεση (Execute)**



Το πρόγραμμα γράφεται σε έναν συντάκτη κειμένου και αποθηκεύεται στον δίσκο με την κατάληξη **.c**

Ο μεταγλωττιστής μεταγλωττίζει το πρόγραμμα σε κώδικα γλώσσας μηχανής, δημιουργεί και αποθηκεύει το αποτέλεσμα στο δίσκο ως αρχείο με κατάληξη **.o** (ή **.obj**)

Ο συνδέτης (linker) συνδέει τον κώδικα που παράγεται από τον μεταγλωττιστή **.o** (Unix/Linux) ή **.obj** (Windows) με τις βιβλιοθήκες, με αποτέλεσμα τη δημιουργία του εκτελέσιμου αρχείου (**a.out** ή **.exe** αντίστοιχα)

Το πρόγραμμα φορτώνεται στη μνήμη

Η CPU παίρνει τις εντολές του προγράμματος και τις εκτελεί «σειριακά», πιθανώς αποθηκεύοντας νέες τιμές δεδομένων καθώς εκτελείται το πρόγραμμα

Το πρώτο πρόγραμμα

```
#include <stdio.h>
int main(void)
{
        printf("Ramones: Hey Ho, Let's Go\n");
        return 0;
}
```

Η οδηγία #include

- #include <stdio.h>

  - Με την οδηγία #include <όνομα_αρχείου> ο μεταγλωττιστής υποχρεώνεται να ενσωματώσει και να συμπεριλάβει (include) τα περιεχόμενα του αρχείου στον κώδικα του προγράμματος

  - Το αρχείο stdio.h (standard input output) περιέχει τις βασικές (standard) δηλώσεις των συναρτήσεων με τις οποίες γίνεται εμφάνιση δεδομένων στην οθόνη (output) και εισαγωγή δεδομένων από το πληκτρολόγιο (input)

Παρατηρήσεις

- Το αρχείο `stdio.h` συμπεριλαμβάνεται σχεδόν πάντα σε ένα C πρόγραμμα, αφού περιέχει τις συναρτήσεις για εμφάνιση και διάβασμα δεδομένων
- Συνήθως, τα αρχεία που συμπεριλαμβάνονται έχουν κατάληξη `.h` και ονομάζονται αρχεία επικεφαλίδας (header files)
- Η οδηγία `#include` ξεκινάει πάντα με το σύμβολο # και ΔΕΝ τελειώνει με ελληνικό ερωτηματικό (;)
- Ο προγραμματιστής μπορεί να δημιουργήσει ένα δικό του αρχείο και να το συμπεριλάβει στο πρόγραμμά του με την οδηγία `#include`
- Κατά τη μεταγλώττιση του προγράμματος, ο μεταγλωττιστής ψάχνει να βρει σε ποιο σημείο του δίσκου είναι αποθηκευμένο το αρχείο για να το ενσωματώσει στο πρόγραμμα
  - Όταν το όνομα του αρχείου είναι γραμμένο μέσα σε < >, τότε ο μεταγλωττιστής ψάχνει σε προεπιλεγμένους φακέλους
  - Όταν το όνομα του αρχείου είναι γραμμένο μέσα σε διπλά εισαγωγικά (" "), τότε ο μεταγλωττιστής ψάχνει πρώτα στον ίδιο φάκελο που βρίσκεται το πρόγραμμα και μετά σε προεπιλεγμένους φακέλους

Η συνάρτηση `main()`

- `int main (void)`
  - Ένα πρόγραμμα γραμμένο στη γλώσσα C μπορεί – όπως θα δούμε – να περιέχει αρκετές συναρτήσεις
    - πρέπει όμως οπωσδήποτε να περιέχει τη συνάρτηση `main()`, η οποία καλείται και «κύρια συνάρτηση του προγράμματος»
  - Η εκτέλεση του προγράμματος ξεκινάει από την πρώτη εντολή που υπάρχει μέσα στη συνάρτηση `main()`
  - Η εκτέλεση του προγράμματος τελειώνει με την τελευταία εντολή που υπάρχει μέσα στη συνάρτηση `main()`, εκτός αν κληθεί νωρίτερα μία εντολή εξόδου, όπως η `return`
  - Οι εντολές της συνάρτησης `main()` ή αλλιώς το «σώμα της συνάρτησης» πρέπει να περιέχονται μέσα σε άγκιστρα `{...}`

Η εντολή printf()

- `printf("Ramones: Hey Ho, Let's Go\n");`
  - Οι εντολές γράφονται συνήθως μία σε κάθε γραμμή και σχεδόν πάντα τελειώνουν με το ελληνικό ερωτηματικό (;)
  - Η εντολή που χρησιμοποιείται στο παραπάνω πρόγραμμα είναι η `printf()`
  - Η `printf()` δηλώνεται μέσα στο αρχείο `stdio.h` και γι' αυτό το λόγο συμπεριλαμβάνουμε αυτό το αρχείο με την οδηγία `#include`
  - Η εντολή `printf()` χρησιμοποιείται για την εμφάνιση δεδομένων στην οθόνη
  - Ο χαρακτήρας `'\n'` μέσα στην εντολή `printf()` δημιουργεί μία νέα γραμμή μετά την εμφάνιση του μηνύματος στην οθόνη
    - Ισοδυναμεί δηλ. με το πάτημα του πλήκτρου Enter

Η εντολή return

- `return 0;`
  - Στην απλούστερη μορφή της, η συνάρτηση `main()` γράφεται όπως στο παράδειγμά μας, δηλαδή σαν `int main(void)`
  - Η ειδική λέξη `int` υποδηλώνει ότι η συνάρτηση `main()` πρέπει να επιστρέψει έναν ακέραιο αριθμό (integer) στο λειτουργικό σύστημα
  - Ο ακέραιος αυτός αριθμός επιστρέφεται με την εντολή `return` (που αποτελεί δεσμευμένη λέξη της C)
  - Κατά σύμβαση, όταν ο αριθμός που επιστρέφεται είναι το μηδέν (0), τότε σημαίνει ότι το πρόγραμμά μας τερματίστηκε κανονικά

Παρατηρήσεις (Ι)

- Ένα πρόγραμμα γραμμένο σε C πρέπει να περιέχει ΜΟΝΟ ΜΙΑ συνάρτηση `main()`

- Μπορεί να συναντήσετε και άλλες δηλώσεις της `main()`
  - π.χ.

```
void main(void)
{
...
}
```

  - ή π.χ.

```
int main()
{
...
}
```

- Προτείνεται ο τρόπος δήλωσης της `main()` όπως έγινε στο παράδειγμα, διότι ακολουθεί το πρότυπο ANSI

  - Δηλ.

```
int main(void)
{
...
}
```

Παρατηρήσεις (II)

- Η δεσμευμένη λέξη `void` μέσα στις παρενθέσεις δηλώνει ρητά ότι η συνάρτηση `main()` δεν δέχεται κάποια ορίσματα

- Σε επόμενη διάλεξη, θα συναντήσουμε μία εναλλακτική δήλωση της `main()`, όπου δέχεται παραμέτρους

Εισαγωγή Σχολίων σε πρόγραμμα (Ι)

- **Τι είναι τα «Σχόλια»???**
  - Είναι κείμενο το οποίο εισάγεται από τον προγραμματιστή στον κώδικα προκειμένου να καταστήσει τον ίδιο τον κώδικα περισσότερο ευανάγνωστο και σαφή
  - Τα σχόλια εισάγονται είτε για επεξήγηση προς τρίτους (που πιθανόν διαβάσουν τον κώδικά μας) είτε και για μας τους ίδιους (ιδίως όταν πρόκειται να διαβάσουμε τον κώδικά μας μετά από αρκετά μεγάλο χρονικό διάστημα)
  - Περιέχεται μεταξύ `/*` και `*/` (και αγνοείται από τον υπολογιστή)
  - Επίσης σχόλιο είναι και το κείμενο στην ίδια γραμμή μετά από `//`

Εισαγωγή Σχολίων σε πρόγραμμα (ΙΙ)

- π.χ.

```
#include <stdio.h>
/* This program shows the message "Ramones: Hey Ho, Let's Go" on
the screen. */
int main(void)
{
      printf("Ramones: Hey Ho, Let's Go");
      return 0;
}
```

- π.χ.

```
#include <stdio.h>
int main(void)     // This program shows a message on the screen
{
      printf("Ramones: Hey Ho, Let's Go");
      return 0;
}
```

Εισαγωγή Σχολίων σε πρόγραμμα (ΙΙΙ)

- π.χ. το σχόλιο

```
/* This is a comment
which takes two lines. */
```

- μπορεί να γραφεί ισοδύναμα ως:

```
// This is a comment
// which takes two lines.
```

Παρατηρήσεις

- Η χρήση των πλάγιων καθέτων // σαν ένδειξη εισαγωγής σχολίων **δεν είναι σύμφωνη** με το ANSI C πρότυπο
- Άρα, αν και υπάρχουν μεταγλωττιστές που την υποστηρίζουν, δεν είναι υποχρεωτικό να υποστηρίζεται από όλους τους C μεταγλωττιστές
- Αντίθετα, η χρήση των /* */ για εισαγωγή σχολίων υποστηρίζεται από όλους τους C μεταγλωττιστές, αφού υποστηρίζεται από το ANSI C πρότυπο
- Προτείνεται η εισαγωγή περιγραφικών σχολίων σε πολύπλοκα σημεία του κώδικα, ώστε ο κώδικας να γίνεται πιο κατανοητός τόσο στον ίδιο τον προγραμματιστή όσο και σε συνεργάτες του που ενδέχεται να πρέπει να κατανοήσουν τον κώδικά του

```
8   // Dear programmer:
9   // When I wrote this code, only god and
10  // I knew how it worked.
11  // Now, only god knows it!
12  //
13  // Therefore, if you are trying to optimize
14  // this routine and it fails (most surely),
15  // please increase this counter as a
16  // warning for the next person:
17  //
18  // total_hours_wasted_here = 254
19  //
20
```

Μεταγλώττιση Προγράμματος

- Όταν έχουμε γράψει τον κώδικα του προγράμματός μας π.χ. αρχείο `my_example.c`, θα πρέπει αρχικά να τον μεταγλωττίσουμε (να κάνουμε compilation) στη γλώσσα που αντιλαμβάνεται ο υπολογιστής (γλώσσα μηχανής)

- Τη διαδικασία αυτή την αναλαμβάνει ο compiler (μεταγλωττιστής)

- Αν ο κώδικάς μας μεταγλωττιστεί επιτυχώς, τότε δημιουργείται το αρχείο `my_example.o` (Unix/Linux) ή `my_example.obj` (Windows), και είμαστε έτοιμοι να μεταβούμε στο επόμενο στάδιο, δηλαδή στη σύνδεση (linking) του κώδικά μας με τις βιβλιοθήκες της γλώσσας C

Σύνδεση Προγράμματος (1/2)

- Η επόμενη διαδικασία πραγματοποιείται από το πρόγραμμα σύνδεσης *(linker)*, το οποίο συνδέει τον κώδικα αντικείμενου που δημιουργήθηκε στη φάση μεταγλώττισης με πρόσθετο κώδικα που χρειάζεται για τη δημιουργία του τελικού εκτελέσιμου

- Ο πρόσθετος κώδικας περιλαμβάνει τον κώδικα των συναρτήσεων βιβλιοθήκης που χρησιμοποιεί το πρόγραμμα (π.χ. `printf()`)

- Αυτός ο κώδικας περιέχεται σε αρχεία βιβλιοθήκης *(library files)* που παρέχονται μαζί με τον μεταγλωττιστή
  - Σημειώστε ότι τα αρχεία βιβλιοθήκης είναι διαφορετικά από τα αρχεία επικεφαλίδας
  - ΠΡΟΣΟΧΗ! Ο κώδικας των συναρτήσεων βρίσκεται στα αρχεία βιβλιοθηκών και όχι σε αρχεία επικεφαλίδας

Σύνδεση Προγράμματος (2/2)

- Για παράδειγμα, αν χρησιμοποιείτε τον *gcc* μεταγλωττιστή και το πρόγραμμά σας χρησιμοποιεί μία μαθηματική συνάρτηση (π.χ. `sqrt()`) μπορεί να χρειαστεί να φορτώσετε την αντίστοιχη βιβλιοθήκη γράφοντας:

```
gcc test.c -lm
```

- Σε ένα ενοποιημένο περιβάλλον, συνήθως υπάρχει επιλογή με το όνομα *Build*, με την οποία η μεταγλώττιση και σύνδεση του προγράμματος γίνεται σε ένα βήμα

- Αν η σύνδεση είναι επιτυχής, τότε δημιουργείται το εκτελέσιμο (executable) αρχείο του προγράμματός μας `a.out` (σε Unix/ Linux με τον μεταγλωττιστή gcc) ή `my_example.exe` (σε Windows) και είμαστε σε θέση να το εκτελέσουμε

Εκτέλεση Προγράμματος

- Η εκτέλεση ενός προγράμματος γίνεται με αρκετούς τρόπους
- Για παράδειγμα, αν χρησιμοποιήσουμε τον μεταγλωττιστή gcc γράφουμε σε μία γραμμή εντολών:

```
$ a.out
```

- Αν εμφανιστεί ένα μήνυμα λάθους παρόμοιο με `No such file or directory` ή `command not found` γράψτε:

```
./a.out
```

- Σε ένα ενοποιημένο περιβάλλον, π.χ. Dev-C++, θα υπάρχει επιλογή παρόμοια με **Execute->Compile & Run**

Παρατηρήσεις (Ι)

- Η C είναι μία γλώσσα προγραμματισμού που κάνει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων (case sensitive)
  - Δηλαδή, αν γράψουμε `Printf()` αντί για `printf()` ο μεταγλωττιστής θα εμφανίσει μήνυμα λάθους, γιατί ο χαρακτήρας `'P'` είναι διαφορετικός από τον χαρακτήρα `'p'`

- Ένα μήνυμα λάθους που εμφανίζει ένας μεταγλωττιστής μπορεί να μην συμβαίνει στη γραμμή που υποδεικνύει, αλλά στις αμέσως προηγούμενες γραμμές

- Αν ο μεταγλωττιστής εμφανίζει πολλά μηνύματα λάθους, τότε προσπαθήστε να βρείτε και να διορθώσετε **μόνο το πρώτο λάθος** και μεταγλωττίστε πάλι το πρόγραμμά σας. Είναι πολύ πιθανό, η νέα μεταγλώττιση να εμφανίσει πολύ λιγότερα ή και καθόλου λάθη

- Ο μεταγλωττιστής μπορεί να μην εμφανίσει μηνύματα λάθους, αλλά να εμφανίσει μηνύματα προειδοποίησης (warnings). Ένα μήνυμα προειδοποίησης μπορεί να ενημερώνει τον προγραμματιστή για ενδεχόμενη δυσλειτουργία του προγράμματός του, οπότε καλό είναι να μην τα αγνοείτε

Παρατηρήσεις (ΙΙ)

- Ο μεταγλωττιστής ανιχνεύει λανθασμένη εφαρμογή των κανόνων της γλώσσας που τον εμποδίζει να μεταγλωττίσει το πρόγραμμα σωστά (π.χ. ορθογραφικά λάθη, συντακτικά λάθη, αδήλωτες μεταβλητές, ...) και όχι λάθη λογικής που ενδέχεται να περιέχονται στο πρόγραμμά σας

- Δηλαδή, ο μεταγλωττιστής δεν είναι «μέσα στο κεφάλι σας» για να ξέρει ποιος είναι ο σκοπός του προγράμματός σας

- Άρα, αν το πρόγραμμά σας μεταγλωττίζεται σωστά, δεν σημαίνει απαραίτητα ότι θα παράγει και σωστά αποτελέσματα

Παρατηρήσεις (III)

- Π.χ. αν θέλετε να δημιουργήσετε ένα πρόγραμμα που να εμφανίζει τη λέξη **OLE**, μόνο αν η τιμή μίας ακέραιας μεταβλητής **a** είναι μεγαλύτερη από 5, και εσείς γράψετε:

```
if(a < 5)
        printf("OLE");
```

τότε ο μεταγλωττιστής δεν θα εμφανίσει κάποιο μήνυμα λάθους, γιατί αυτό το λάθος δεν ανήκει στα λάθη που αναγνωρίζει

- Το λάθος αυτό ονομάζεται **λογικό λάθος**, και ο μεταγλωττιστής δεν το αντιλαμβάνεται

- Τα λογικά λάθη ονομάζονται και **bugs** και είναι λάθη τα οποία εισάγει άθελά του ο προγραμματιστής και μόνο αυτός μπορεί να τα εντοπίσει και να τα διορθώσει

- **ΜΕΓΑΛΗ ΠΡΟΣΟΧΗ** λοιπόν, γιατί τα bugs είσαστε αναγκασμένοι να τα εντοπίζετε και να τα διορθώνετε μόνοι σας, κάνοντας τη λεγόμενη αποσφαλμάτωση (debugging), με ή χωρίς τη βοήθεια κάποιου ειδικού περιβάλλοντος ανίχνευσης λαθών (debugger), ανάλογα με την πολυπλοκότητα του προγράμματός σας