Fortran και Αντικειμενοστραφής προγραμματισμός

www.corelab.ntua.gr/courses/fortran_naval/naval

Διδάσκοντες: Άρης Παγουρτζής (pagour@cs.ntua.gr)
(Επίκουρος Καθηγητής ΣΗΜΜΥ)
Δώρα Σούλιου (dsouliou@mail.ntua.gr)
(ΕΔΙΠ ΣΗΜΜΥ)

1η ενότητα

- ✓ Διαδικαστικά
- ✓ Εισαγωγή στην πληροφορική
- ✓ Εισαγωγή στη γλώσσα Fortran
- ✓ Παραδείγματα προγραμμάτων Fortran

Αρχικές Διαφάνειες σε Pascal: Ε. Ζάχος, Ν. Παπασπύρου
Προσαρμογή σε Fortran - συμπληρώσεις: Α. Παγουρτζής, Δ. Σούλιου

Διαδικαστικά

- Τετάρτη 12:45 – 14:30
  Αίθουσα 5 ισόγειο κτηρίου Ηλεκτρολόγων
- Παρασκευή 10:45 – 12:30
  Αίθουσα 5 ισόγειο κτηρίου Ηλεκτρολόγων
- Ώρες γραφείου
  Τετάρτη 12:45 – 14:30
  Γραφείο 1.1.30 παλαιό κτίριο ΣΗΜΜΥ
  Τηλέφωνο: 210 7721644
- Βαθμολογία (πρωτοετείς φοιτητές)
  Τελική Εξέταση (9 μονάδες)
  Σειρές ασκήσεων (1 μονάδα)
  με προφορική εξέταση
  Πρόοδος (1 μονάδα)

Εισαγωγή (i)

Σκοπός του μαθήματος

- Εισαγωγή στην πληροφορική (computer science)
- Εισαγωγή στον προγραμματισμό ηλεκτρονικών υπολογιστών (Η/Υ)
- Μεθοδολογία αλγοριθμικής επίλυσης προβλημάτων

Εισαγωγή (ii)

Αλγόριθμος

Πεπερασμένη ακολουθία ενεργειών
που περιγράφει τον τρόπο επίλυσης ενός προβλήματος

Εφαρμόζεται σε δεδομένα (data)

Πρόγραμμα

Ακριβής περιγραφή ενός αλγορίθμου σε μια τυπική γλώσσα που ονομάζεται γλώσσα προγραμματισμού

Εισαγωγή (iii)

Φυσική γλώσσα

- Χωρίς τόσο αυστηρούς συντακτικούς περιορισμούς
- Μεγάλη πυκνότητα και σημασιολογική ικανότητα

Τυπική γλώσσα

- Αυστηρότατη σύνταξη και σημασιολογία

Γλώσσα προγραμματισμού

- Τυπική γλώσσα στην οποία μπορούν να περιγραφούν υπολογισμοί
- Εκτελέσιμη από έναν ηλεκτρονικό υπολογιστή

Εισαγωγή (iv)

Πληροφορική

Ηλεκτρονικοί υπολογιστές (engineering) — Μαθηματικά

Σχεδίαση και κατασκευή

Θεωρία και αναλυτική μέθοδος

- Κεντρική έννοια: υπολογισμός (computation)

Εισαγωγή (v)

Πληροφορική: μαθηματικοποίηση της μεθοδολογίας των μηχανικών

- Απαιτήσεις – Πρόβλημα
- Προδιαγραφές
- Σχεδίαση
- Υλοποίηση
- Εμπειρικός έλεγχος – Θεωρητική επαλήθευση
- Βελτιστοποίηση
- Πολυπλοκότητα (κόστος πόρων-αγαθών)
- Τεκμηρίωση
- Συντήρηση

Έννοιες που υπήρχαν για τους μηχανικούς, στην πληροφορική τυποποιήθηκαν, πήραν μαθηματική μορφή, άρα μπορεί κανείς να επιχειρηματολογήσει με αυτές τις έννοιες χρησιμοποιώντας αποδείξεις.

Εισαγωγή (vi)

Δευτεροβάθμια εκπαίδευση

Σκοπός: να μάθεις να σκέφτεσαι

- Η Ευκλείδεια Γεωμετρία (με τη βασική διδακτική της αξία) απουσιάζει από το πρόγραμμα σπουδών εδώ και χρόνια.
- Αποτέλεσμα: όπως είδαμε και στις πανελλήνιες εξετάσεις δίνεται έμφαση στην αποστήθιση ανουσίων θεωρημάτων και γνώσεων διαφορικού και απειροστικού λογισμού. Η ικανότητα μαθηματικής επίλυσης απλών αλλά πρωτότυπων προβλημάτων δεν παίζει ρόλο.
- Απουσία γνώσεων συνδυαστικής (μέτρηση περιπτώσεων, τρίγωνο Pascal).
- Εφαρμογή των αποστηθισμένων κανόνων;
- Άλγεβρα: αν ρωτήσω έναν τελειόφοιτο Λυκείου πόσο κάνει 107×93 θα δυσκολευτεί πολύ να απαντήσει, ενώ φυσικά γνωρίζει ότι $(\alpha+\beta)(\alpha-\beta) = \alpha^2-\beta^2$

Εισαγωγή (vii)

- Οι μαθητές αγνοούν την έννοια του "αποδοτικού αλγόριθμου"
  - π.χ. μαθαίνουν ένα μη-αποδοτικό αλγόριθμο για την εύρεση του Μ.Κ.Δ. ενώ ο αλγόριθμος του Ευκλείδη απουσιάζει από την ύλη
- Πρόταση
  - Εισαγωγή της Αλγοριθμικής Πληροφορικής στη δευτεροβάθμια εκπαίδευση για όλους τους μαθητές
  - Μεθοδολογία επίλυσης προβλημάτων με σχεδίαση και υλοποίηση αλγορίθμων

Εισαγωγή (viii)

Τριτοβάθμια εκπαίδευση

- Η τεχνολογία αλλάζει αέναα και γρήγορα – τα θεμέλια μένουν
- Αυτά τα θεμέλια πρέπει να είναι η ραχοκοκαλιά στην τριτοβάθμια εκπαίδευση: έμφαση στην αλγοριθμική σκέψη σε αντιδιαστολή με τις τεχνολογικές δεξιότητες (computer literacy)
- Computer science, computing science, informatics
- Dijkstra: η Επιστήμη των Υπολογιστών έχει τόση σχέση με τους υπολογιστές όση και η Αστρονομία με τα τηλεσκόπια
- Primality: σημαντικό επίτευγμα σε μία χώρα χωρίς υποδομές

Εισαγωγή (ix)

Να μην ξεχνάμε ότι

- Το να κάνεις λάθη είναι ανθρώπινο.
- Για να τα κάνεις τελείως "μπάχαλο" χρειάζεσαι υπολογιστή!

Εισαγωγή (x)

Κατασκευή υπολογιστικών μηχανών

Αρχαιότητα: υπολογιστικές μηχανές, μηχανισμός των Αντικυθήρων, κ.λπ.

17ος αιώνας, Pascal και Leibniz,
μηχανικές υπολογιστικές αριθμομηχανές
$\Rightarrow$ στοιχειώδεις αριθμητικές πράξεις

1830–1840, Babbage, "αναλυτική μηχανή"
$\Rightarrow$ λογάριθμοι, τριγωνομετρικές συναρτήσεις

1880–1890, Hollerith, μηχανή με διάτρητες κάρτες για την αυτοματοποίηση των εκλογών

Εισαγωγή (xi)

Κατασκευή υπολογιστών

1920–1930, Bush, ηλεκτρική (αναλογική) υπολογιστική μηχανή ⇒ διαφορικές εξισώσεις

~1940, Zuse, ηλεκτρονική (ψηφιακή) υπολογιστική μηχανή
⇒ πρόγραμμα και δεδομένα, χωριστά

1945–1950, μοντέλο von Neumann
⇒ πρόγραμμα και δεδομένα, από κοινού

1950–σήμερα, ραγδαία ανάπτυξη της τεχνολογίας των ηλεκτρονικών υπολογιστών

Εισαγωγή (xii)

Κατασκευή υπολογιστών

1952–	main frames	IBM 650, 7000, 360
1965–	mini computers	DEC PDP-8
1977–	personal computers	Apple II
1981		IBM PC
1983, 1984		Apple: Lisa, Macintosh
1985–	internet	
1990–	world wide web	
2000–	PDA, smartphones, κ.λπ.	

Εισαγωγή (xiii)

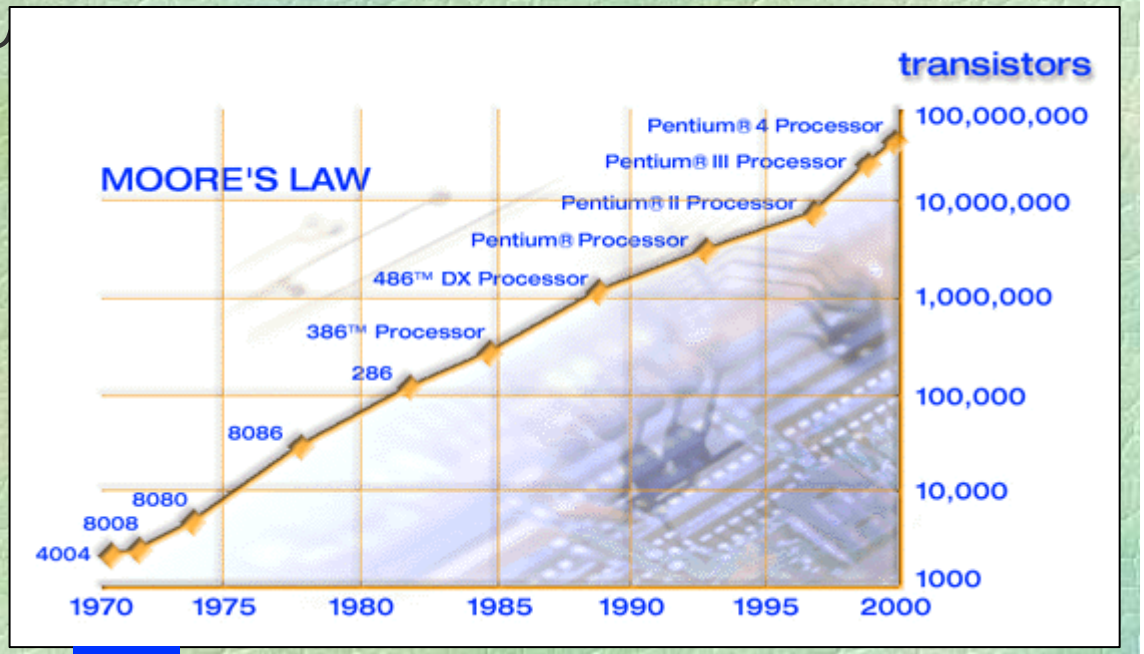
Μηχανικοί υπολογιστών

Tom Watson, IBM, 1945

Ο κόσμος χρειάζεται περίπου 5 υπολογιστές

Gordon Moore, Intel, 1965

Η πυκνότητα του hardware στα ολοκληρωμένα κυκλώματα διπλασιάζεται κάθε 18 μήνες



© intel. http://www.intel.com/research/silicon/mooreslaw.htm

Εισαγωγή (xiv)

Θεμέλια της πληροφορικής

Μαθηματική λογική

Αριστοτέλης: συλλογισμοί

$$\frac{A \qquad A \rightarrow B}{B} \quad \textit{(modus ponens)}$$

- Ευκλείδης: αξιωματική θεωρία
- Hilbert: αξίωμα, θεώρημα, τυπική απόδειξη

Εισαγωγή (xv)

Πρόγραμμα του Leibniz:

θεμελίωση των μαθηματικών

γλώσσα *για όλα τα μαθηματικά*

θεωρία

συνεπής (consistent) και πλήρης (complete)

$$A \wedge \neg A \qquad \textit{αντίφαση}$$

- Γλώσσα (Boole, De Morgan, Frege, Russel)
  - προτασιακός λογισμός $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
  - κατηγορηματικός λογισμός $\forall, \exists$

Εισαγωγή (xvi)

Θεωρία

Συνολοθεωρία, Cantor, Frege $\in$

Παράδοξο του Russel

$$A = \{ x \mid x \notin x \}$$

$$A \in A \rightarrow A \notin A$$
$$A \notin A \rightarrow A \in A$$

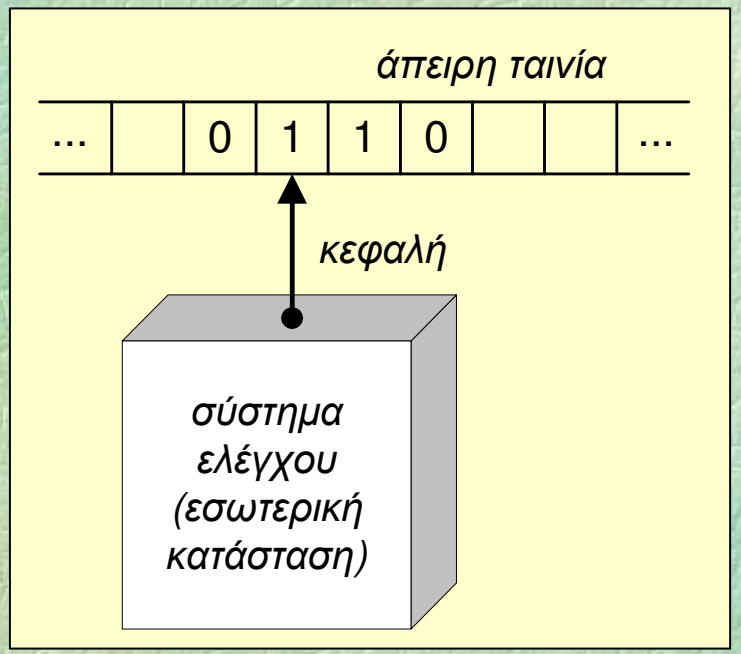
- Άλλες θεωρίες συνόλων (ZF, κ.λπ.)
- Άλλες θεωρίες για τη θεμελίωση των μαθηματικών (θεωρία συναρτήσεων, κατηγοριών, κ.λπ.)
- 1920–1930, προσπάθειες για απόδειξη συνέπειας

Εισαγωγή (xvii)

Συνέπεια και πληρότητα

1931, Gödel, θεώρημα μη πληρότητας
⇒ δεν είναι δυνατόν να κατασκευαστεί συνεπής και πλήρης θεωρία της αριθμητικής

1936, Turing,
⇒ μη αποκρίσιμες (undecidable) προτάσεις
⇒ μηχανή Turing, υπολογισιμότητα



Εισαγωγή (xviii)

Μη πληρότητα (incompleteness)

David Hilbert, 1862-1943

Kurt Gödel, 1906-1978 (ασιτία)

Δοξιάδης

Incompleteness: a play and a theorem

Ο θείος Πέτρος και η εικασία του Goldbach

Παπαδημητρίου

Το χαμόγελο του Turing

Hoffstader

Gödel, Escher, and Bach

Εισαγωγή (xix)

Κλάδοι της πληροφορικής

- Αλγόριθμοι και δομές δεδομένων
- Γλώσσες προγραμματισμού
- Αρχιτεκτονική υπολογιστών και δικτύων
- Αριθμητικοί και συμβολικοί υπολογισμοί
- Λειτουργικά συστήματα
- Μεθοδολογία – τεχνολογία λογισμικού
- Βάσεις δεδομένων και διαχείριση πληροφοριών
- Τεχνητή νοημοσύνη και ρομποτική
- Επικοινωνία ανθρώπου – υπολογιστή

Εισαγωγή (xx)

Υπολογιστής

- επεξεργαστής
- μνήμη
- συσκευές εισόδου/εξόδου

Ιδιότητες

- αυτόματο χωρίς εξυπνάδα
- μεγάλη ταχύτητα
- ακρίβεια στις πράξεις

Γλώσσες προγραμματισμού (i)

Γλώσσα μηχανής

0110110 **11011011**

διεύθυνση εντολή

Συμβολική γλώσσα (assembly)

label: **add** **ax, bx**

διεύθυνση πράξη δεδομένα

Γλώσσες χαμηλού και υψηλού επιπέδου

Υλοποίηση γλωσσών προγραμματισμού

μεταγλωττιστής (compiler)

διερμηνέας (interpreter)

Γλώσσες προγραμματισμού (ii)

Κυριότερες γλώσσες, ιστορικά

1950 1960	FORTRAN, LISP, COBOL, Algol, BASIC, PL/I
1970	Pascal, C, Smalltalk, Prolog, ML, Logo
1980	C++, Modula-2, Ada, Perl, MATLAB, Erlang, Eiffel
1990	Java, Python, Ruby, Haskell, PHP, Visual Basic, Javascript
2000	C#, F#, ……

FORTRAN (FORmulae TRANslator) (i)

- Από τις πρώτες γλώσσες προγραμματισμού υψηλού επιπέδου
- Δημιουργήθηκε τη δεκαετία του '50 (1956) για τον IBM704 και στη συνέχεια προσαρμόστηκε σε διάφορα υπολογιστικά συστήματα
- 1958: Fortran II με πολλά υποπρογράμματα
- Ακολούθησε η Fortran IV η οποία δόθηκε στην επιστημονική κοινότητα
- Ανάγκη τυποποίησης (πολλές υλοποιήσεις σε διάφορες μηχανές, συμβατότητα μεταξύ των μηχανών)
- Fortran 66: η πρώτη τυποποιημένη Fortran από το αμερικάνικο ινστιτούτο ANSI
- Fortran 77: η πρώτη τυποποιημένη Fortran από τον ISO

FORTRAN (FORmulae TRANslator) (ii)

- Fortran 77: συμπλήρωνε τις ελλείψεις της Fortran 66 και υιοθετούσε χαρακτηριστικά που είχαν επιτυχώς υλοποιηθεί σε άλλες γλώσσες (Algol, Modula, Pascal)
- Fortran 90: δόθηκε από τον ISO το 1992 με ριζικές αλλαγές
- Fortran 95: δόθηκε από τον ISO το 1997 με δυνατότητα παράλληλης επεξεργασίας
- Fortran 2003/2008: με πολλά νέα χαρακτηριστικά , κυρίως αντικειμενοστραφούς προγραμματισμού
- Αναμένονται νεώτερες εκδόσεις (Fortran 2015)
- 2 βασικές διάλεκτοι της γλώσσας
  - Παλαιά (πριν την Fortran90) εκατομμύρια προγράμματα σε αρχεία με επέκταση .for ή .f
  - Νέα (Fortran 90,95,2003, 2008) προγράμματα σε αρχεία με επέκταση .f90, .f95, .f03, .f08

FORTRAN (FORmulae TRANslator) (iii)

- Πλεονεκτήματα άλλων γλωσσών
  - επικοινωνία με τη μηχανή (γλώσσες χαμηλού επιπέδου)
  - δημιουργία γραφικών
  - χειρισμό αρχείων
  - ευκολία χρήσης
  - επικοινωνία με το διαδίκτυο
- Πλεονεκτήματα Fortran
  - ταχύτερη σε πολύπλοκους επιστημονικούς και τεχνικούς υπολογισμούς,
  - προβλήματα με πολυδιάστατους πίνακες,
  - πολλές επαναλήψεις,
  - προβλήματα βελτιστοποίησης

Παραδείγματα (i)

```
PROGRAM Hello1
IMPLICIT NONE
    WRITE(*,*) 'hello world'
END
```

```
PROGRAM Hello2
IMPLICIT NONE
    WRITE(*,*) 'hello ', 'world'
END
```

```
PROGRAM Hello3
IMPLICIT NONE
    WRITE(*,*) 'hello '
    WRITE(*,*) 'world'
END
```

Παραδείγματα (σε γλώσσα C) (i)

```
#include <stdio.h>
void main ()
{
   printf("hello world\n");
}
```

```
#include <stdio.h>
void main ()
{
   printf("hello " "world\n");
}
```

```
#include <stdio.h>
void main ()
{
   printf("hello ");
   printf("world\n");
}
```

Παραδείγματα (ii)

```
PROGRAM Hello4
IMPLICIT NONE
    WRITE(*, "(A)", advance="no")'hello'
    WRITE(*,*) 'world'
END
```

Παραδείγματα (σε γλώσσα C) (ii)

```
#include <stdio.h>
void main ()
{
   printf("hello world");
   printf("\n");
}
```

Παραδείγματα (iii)

```
PROGRAM Hello5
    IMPLICIT NONE
    CALL hello
    CALL hello
    CALL hello
    CALL hello
END
SUBROUTINE hello
    IMPLICIT NONE
    WRITE(*,*) 'hello world'
END
```

Παραδείγματα (σε γλώσσα C) (iii)

```
#include <stdio.h>

void hello()
{
   printf("hello world\n");
}

void main()
{
   hello(); hello(); hello(); hello();
}
```

Παραδείγματα (iv)

```
PROGRAM Hello6
   IMPLICIT NONE
   INTEGER:: i
   DO i=1,20
       CALL hello
   END DO
END
SUBROUTINE hello
   IMPLICIT NONE
   WRITE(*,*) 'hello world'
END
```

Παραδείγματα (σε γλώσσα C) (iv)

```
#include <stdio.h>

void hello()
{
   printf("hello world\n");
}

void main()
{
   int i;
   for (i=0; i<20; i++)
      hello();
}
```

Παραδείγματα (v)

```
PROGRAM Hello7
   IMPLICIT NONE
   INTEGER:: i=1
   INTEGER, PARAMETER:: n=21
   DO WHILE (i<n)
      CALL hello(i)
      i=i+1
   END DO
END
SUBROUTINE hello(j)
   IMPLICIT NONE
   INTEGER:: j
   WRITE(*,*) j, ' hello world'
END
```

Παραδείγματα (σε γλώσσα C) (v)

```
#include <stdio.h>
const int n=20;
int i;

void num_hello()
{
   printf("%d hello world\n", i);
}

void main()
{
   for (i=0; i<n; i++)
      num_hello();
}
```

Παραδείγματα (vi)

```
PROGRAM Hello8
   IMPLICIT NONE
   INTEGER:: i,n
   WRITE(*,*) 'Give number of greetings ',&
              'then press <enter>:'
   READ(*,*) n
      DO i=1,n
   CALL hello(i)
      END DO
 END
 SUBROUTINE hello(j)
   IMPLICIT NONE
   INTEGER:: j
   WRITE(*,*)j,'hello world'
END
```

Παραδείγματα (σε γλώσσα C) (vi)

```
#include <stdio.h>
void hello()
{
   printf("hello world\n")
}
void main()
{
   int i,n;
   printf("Give number of greetings"
                  "then press <enter>:");
   scanf("%d\n", n);
   for (i=0; i<n; i++) hello();
}
```

Παραδείγματα (vii)

```
PROGRAM Hello9
   IMPLICIT NONE
   INTEGER:: i,n
   WRITE(*,*) 'Give number of greetings ',&
              'then press <enter>:'
   READ(*,*) n
      DO i=1,n,2
   CALL hello(i)
      END DO
 END
 SUBROUTINE hello(j)
   IMPLICIT NONE
   INTEGER:: j
   WRITE(*,*)j,'hello world'
END
```

Παραδείγματα (σε γλώσσα C) (vii)

```
#include <stdio.h>
void hello()
{
   printf("hello world\n")
}
void main()
{
   int i,n;
   printf("Give number of greetings"
                  "then press <enter>:");
   scanf("%d\n", &n);
   for (i=0; i<n; i++) hello();
}
```

Παραδείγματα (viii)

```
PROGRAM Hello10
   IMPLICIT NONE
   INTEGER:: i,n
   WRITE(*,*) 'Give number of greetings ', &
         'then press <enter>:'
   READ(*,*) n
   IF (n<0) THEN
      WRITE(*,*) 'Ο n είναι αρνητικός'
      ELSE
      DO i = 1,n,1
         CALL hello
         END DO
   END IF
END
SUBROUTINE hello
   IMPLICIT NONE
   WRITE(*,*)'hello world'
END
```

Παραδείγματα (σε γλώσσα C) (viii)

```
#include <stdio.h>
void hello()
{
   printf("hello world\n");
}
void main()
{
   int i,n;
   printf("Give number of greetings"
                "then press <enter>");
   scanf("%d\n",&n);
   if (n<0)
      printf("#is negative\n");
   else
      for (i=0;i<n;i++) hello();
}
```

Πώς θα τρέξετε τα προγράμματα

- Εγκαταστήστε μια έκδοση Linux (π.χ. Ubuntu)
- Ανοίξτε ένα τερματικό (τρέχοντας π.χ. **`xterm`**)
- Εγκαταστήστε την GNU Fortran με την εντολή

  `> sudo apt-get install gfortran`
- Δημιουργήστε ένα αρχείο για κάθε πρόγραμμα, χρησιμοποιώντας έναν κειμενογράφο (π.χ. **`gedit`**)
- Το αρχείο σας να έχει κατάληξη **`.f95`** (π.χ. **`prog1.f95`**)
- Δώστε την εντολή

  `> gfortan prog1.f95 -o prog1`
- Το εκτελέσιμο αρχείο που θα δημιουργηθεί λέγεται **`prog1`** και εκτελείται δίνοντας **`> ./prog1`**