Διάλεξη 2η: Αλγόριθμοι και Προγράμματα

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Βασίζεται σε διαφάνειες του Κ. Παναγιωτάκη

Αλγόριθμοι

- Αλγόριθμος είναι μια ακολουθία από βήματα ή ενέργειες που είναι:
  - Καλώς (σαφώς) ορισμένα
  - Αποτελεσματικά (μπορούν να εκτελεστούν)
  - Πεπερασμένα (τερματισμός)
  - Συνήθως εφαρμόζονται πάνω σε δεδομένα
- Παραδείγματα αλγορίθμων
  - Διαίρεση ακεραίων
  - Υπολογισμός τετραγωνικής ρίζας
  - Πολλαπλασιασμός ή αντιστροφή πινάκων
  - Αναζήτηση και ταξινόμηση
  - Εύρεση συντομότερης διαδρομής
  - Ανάλυση ιατρικών εικόνων

Χαρακτηριστικά

- Απαιτήσεις, Πρόβλημα, Προδιαγραφές
- Σχεδίαση και Ορθότητα
- Πολυπλοκότητα σε σχέση με το μέγεθος των δεδομένων (προαιρετικά για αυτό το μάθημα)
  - Υπολογιστικό κόστος, χρόνος
  - Κόστος σε μνήμη, χώρος
- Βελτιστοποίηση (προαιρετικά)
- **Υλοποίηση**

Παράδειγμα: Πρόσθεση N αριθμών

- Απαιτήσεις, Πρόβλημα, Προδιαγραφές
  - Υπολογίστε το άθροισμα

$$\sum_{i=1}^{N} i = 1 + 2 + 3 + \ldots + N$$

- Σχεδίαση και Ορθότητα
  - Ξεκινώ από το $i = 1$
  - Αρχικά, το άθροισμα "μέχρι τώρα" είναι $\Sigma = 0$
  - Για $i = 1, 2, \ldots, N$, προσθέτω i στο μέχρι τώρα Σ
- Πολυπλοκότητα
  - Υπολογιστικό κόστος: $2N$, ή $O(N)$
  - Κόστος σε μνήμη: 2, ή $O(1)$
- Βελτιστοποίηση:

$$\Sigma = \frac{N(N+1)}{2}$$

  - Υπολογιστικό κόστος: 3, ή $O(1)$
  - Κόστος σε μνήμη: 1, ή $O(1)$

Παράδειγμα: Ταξινόμηση

- Απαιτήσεις, Πρόβλημα, Προδιαγραφές
  - Ταξινομήστε N φυσικούς αριθμούς, μικρότερους από 1000
  - $a[1], a[2], \ldots a[N] < 1000$
- Σχεδίαση και Ορθότητα
  - Βρίσκω το μικρότερο και τον βάζω πρώτο
  - Επαναλαμβάνω για τον επόμενο μικρότερο, κ.ο.κ.

Πιό τυπικά

Για $\kappa = 1$ έως N
  Για $\lambda = \kappa + 1$ έως N
    Αν $a[\lambda] < a[\kappa]$
    τότε άλλαξε θέση των $a[\kappa], a[\lambda]$

- Πολυπλοκότητα
  - Υπολογιστικό κόστος:

$$T(N) = T(N-1) + N \Rightarrow T(N) = \frac{N(N+1)}{2}$$

    ή $O(N^2)$
  - Κόστος σε μνήμη: 2, ή $O(1)$

Παράδειγμα: Ταξινόμηση (2)

- Βελτιστοποίηση

Βελτιστοποίηση

Για $\kappa = 1$ έως 1000
  θέτω $h[\kappa] = 0$
Για $\kappa = 1$ έως N
  αυξάνω το $h\,[a\,[\kappa]]$ κατά 1
Θέτω $\mu = 1$
Για $\kappa = 1$ έως 1000
  Για $\lambda = 1$ έως $h[\kappa]$
    Θέτω $a[\mu] = \kappa$
    Θέτω $\mu = \mu + 1$

- Πολυπλοκότητα
  - Υπολογιστικό κόστος: $1000 + N + 1 + 2N$ ή $O(N)$
  - Κόστος σε μνήμη: 1003, ή $O(1)$

Προγραμματισμός

- Πρόγραμμα: ένας αλγόριθμος σε συγκεκριμένη τυπική γλώσσα
- Γλώσσα μηχανής: η γλώσσα που ερμηνεύει το hardware, μόνο λογικά 0 και 1 (ή οι αντίστοιχες τάσεις στο κύκλωμα)
- Επίπεδα γλωσσών προγραμματισμού
  - Χαμηλού επιπέδου, κοντά στη γλώσσα μηχανής
  - Μεσαίου επιπέδου
  - Υψηλού επιπέδου, κοντά στα μαθηματικά (ή την καθομιλουμένη)
- Η επιλογή γλώσσας προγραμματισμού εξαρτάται από την εφαρμογή
- Συνήθως:
  - Χαμηλό επίπεδο $\Rightarrow$ γρήγορο πρόγραμμα
  - Υψηλό επίπεδο $\Rightarrow$ εύκολος προγραμματισμός, λιγότερα λάθη
  - Πολύ βασικό: η ικανότητα και εμπειρία του προγραμματιστή

Γλώσσες προγραμματισμού

- Ανάλογα με τον τρόπο περιγραφής του αλγορίθμου
  - Προστακτικές (imperative): C, Pascal, Fortran
    - Παραδοσιακός τρόπος προγραμματισμού
    - Το πρόγραμμα είναι ακολουθία εντολών και ορισμός νέων εντολών (ρουτίνες)
  - Συναρτησιακές (functional): ML, Haskel, Lisp
    - Μαθηματική περιγραφή υπολογισμών
    - Το πρόγραμμα είναι ορισμοί μαθηματικών συναρτήσεων
  - Δηλωτικές (declarative): Prolog, XML
    - Δηλώσεις και ορισμοί
    - Το πρόγραμμα είναι η λεπτομερής περιγραφή του αποτελέσματος
- Ανάλογα με τον τρόπο οργάνωσης του κώδικα
  - Διαδικαστικές (procedural): C, Pascal
  - Αντικειμενοστραφείς (Object-oriented): C++, Java
- Ανάλογα με το χώρο των προβλημάτων που λύνουν
  - Παράλληλες, κατανεμημένες (parallel, distributed): CML, Erlang
  - Διαδικτυακές: PHP, Javascript
  - ...

Προγράμματα σε C

- Εντολές και δεδομένα
- Εντολές
  - Μια σειρά από βασικές πράξεις, εντολές στον επεξεργαστή
  - Αποθηκεύονται στη μνήμη του υπολογιστή
- Δεδομένα
  - Γράφονται και διαβάζονται από εντολές
  - Αποθηκεύονται στη μνήμη του υπολογιστή
- Οι εντολές πρέπει να έχουν μια λογική αλληλουχία
- Να ακολουθούν συγκεκριμένα βήματα
- Αλγόριθμοι ...

Παράδειγμα

- Το πρώτο C πρόγραμμα

hello.c

```
#include <stdio.h>
int main()
{
  printf(”Hello, world!\n”);
  return 0;
}
```

Παράδειγμα

- Το πρώτο C πρόγραμμα

Βιβλιοθήκη Input/Output:
χρειάζεται για την `printf()`

hello.c

```
#include <stdio.h>
int main()
{
  printf(”Hello, world!\n”);
  return 0;
}
```

Παράδειγμα

- Το πρώτο C πρόγραμμα

hello.c

```
#include <stdio.h>
int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

Τύπος συνάρτησης:
η `main` επιστρέφει ακέραιο

Παράδειγμα

- Το πρώτο C πρόγραμμα

Σε όλα τα προγράμματα C, η εκτέλεση ξεκινάει από τη συνάρτηση `main`

hello.c

```
#include <stdio.h>
int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

Παράδειγμα

- Το πρώτο C πρόγραμμα

hello.c

```
#include <stdio.h>
int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

Συνάρτηση εκτύπωσης: γράφει `Hello, world!` στην οθόνη και πηγαίνει στην επόμενη γραμμή

Παράδειγμα

- Το πρώτο C πρόγραμμα

hello.c

```
#include <stdio.h>
int main()
{
  printf(”Hello, world!\n”);
  return 0;
}
```

Κάθε εντολή τελειώνει με semicolon “;”

Παράδειγμα

- Το πρώτο C πρόγραμμα

hello.c

```
#include <stdio.h>
int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

Η συνάρτηση επιστρέφει 0

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

Όρισμα συνάρτησης: οι τιμές του x και του y καθορίζονται όταν καλούμε τη συνάρτηση

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Δήλωση τοπικής μεταβλητής: θα χρειαστούμε τη z για να αποθηκεύσουμε το αποτέλεσμα

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Η εντολή `if` ελέγχει μια συνθήκη

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Συνθήκη: είναι το x μικρότερο από το y;

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Αν ισχύει η συνθήκη, εκτελούνται οι εντολές του πρώτου block

Δομή προγράμματος

- Στη C το πρόγραμμα είναι μια λίστα από συναρτήσεις

max.c

```
int max(int x, int y)
{
  int z;

  if (x < y)
  {
    z = y;
  } else {
    z = x;
  }
  return z;
}
```

Αν δεν ισχύει η συνθήκη, εκτελούνται οι εντολές του `else` block

Μεταβλητές

- Θέση στη μνήμη που χωράει δεδομένα κάποιου συγκεκριμένου τύπου
- Ένα “κουτί” μνήμης: βάζοντας κάτι, χάνεται το προηγούμενο
- Κατασκευή μεταβλητής: Δήλωση τύπου, όνομα.

```
int z;
int x = 0;
float pi = 3.14;
```

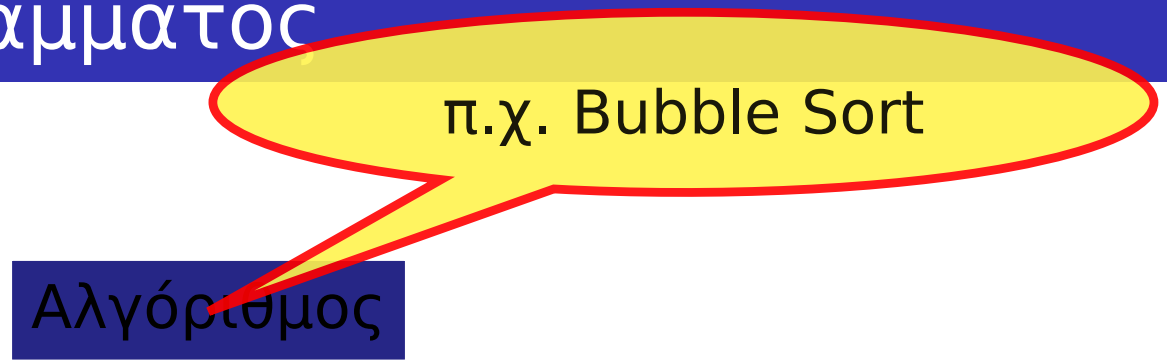
Σχόλια

- Βοηθούν στην ανάγνωση και κατανόηση του κώδικα
- Τα αγνοεί ο μεταγλωττιστής
- Ο κώδικας γράφεται για να διαβαστεί από άλλους προγραμματιστές
  - και για να εκτελεστεί από υπολογιστές

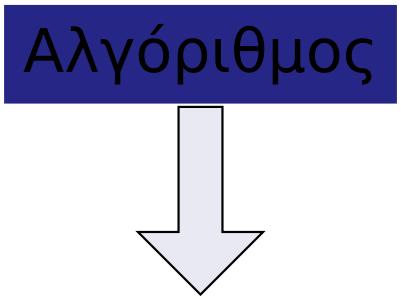
Δημιουργία προγράμματος

Αλγόριθμος

Δημιουργία προγράμματος

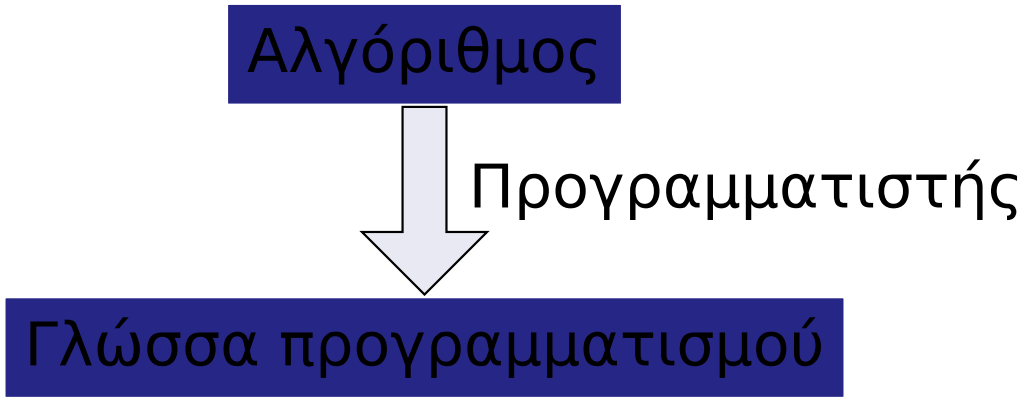


Δημιουργία προγράμματος

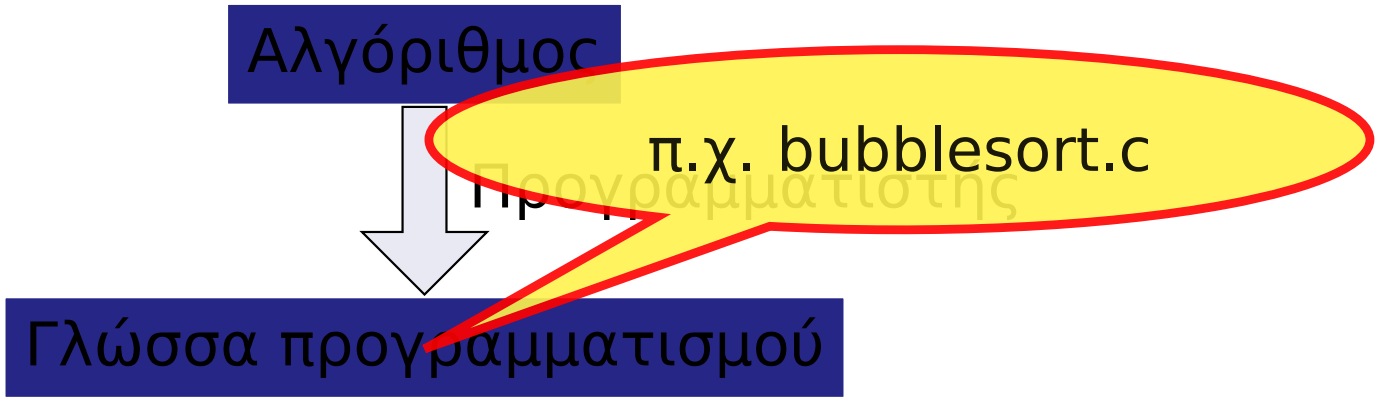


Δημιουργία προγράμματος

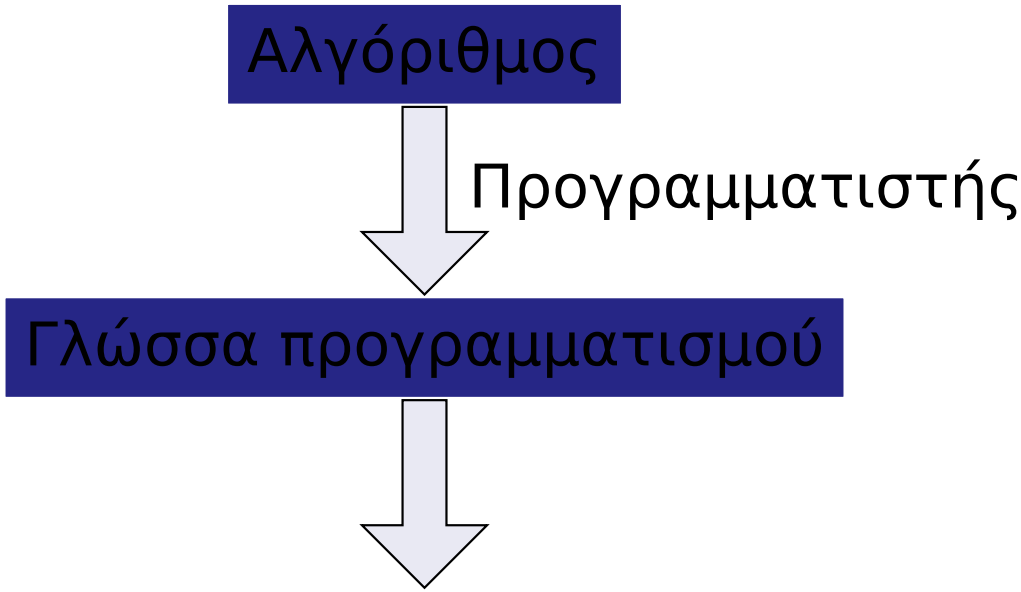
Δημιουργία προγράμματος



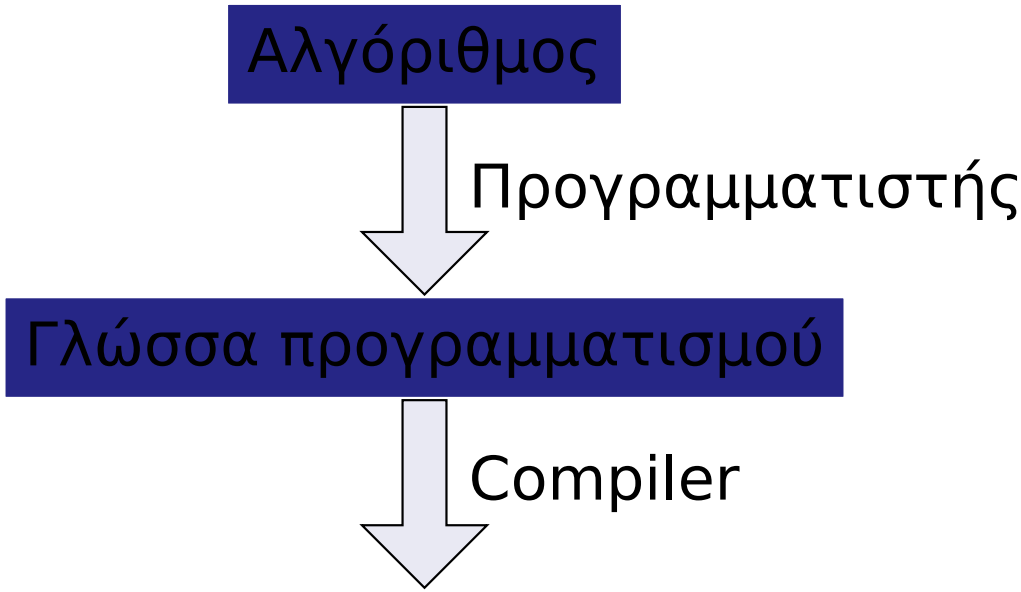
Δημιουργία προγράμματος



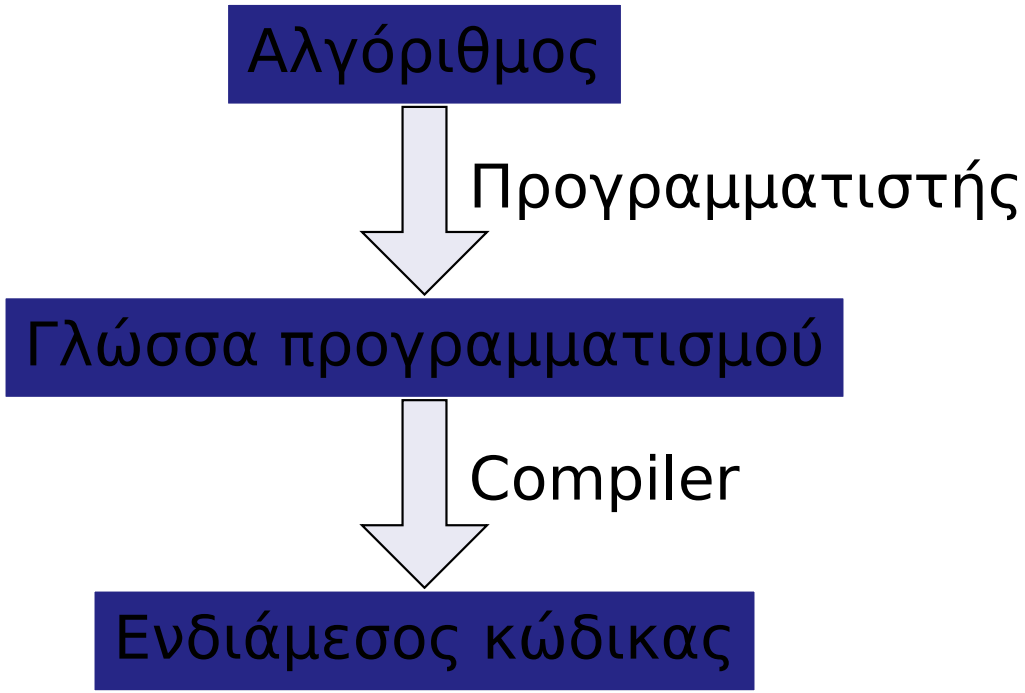
Δημιουργία προγράμματος



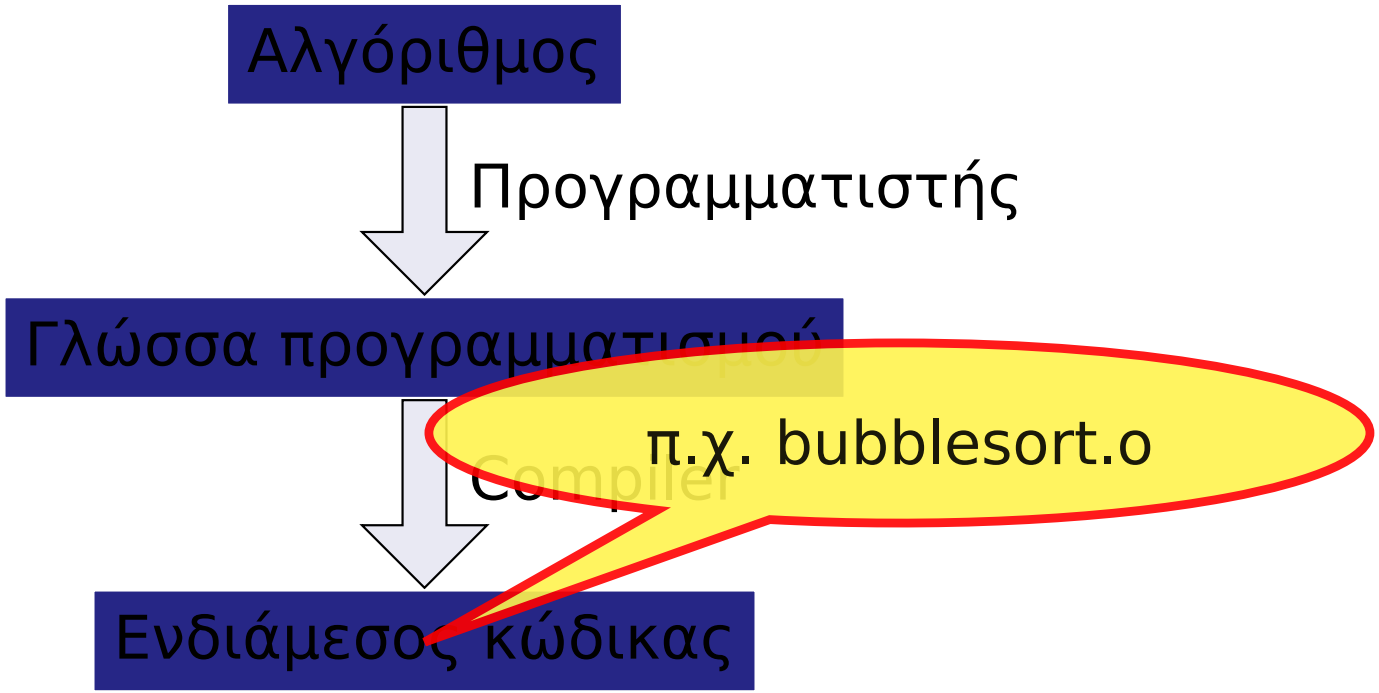
Δημιουργία προγράμματος



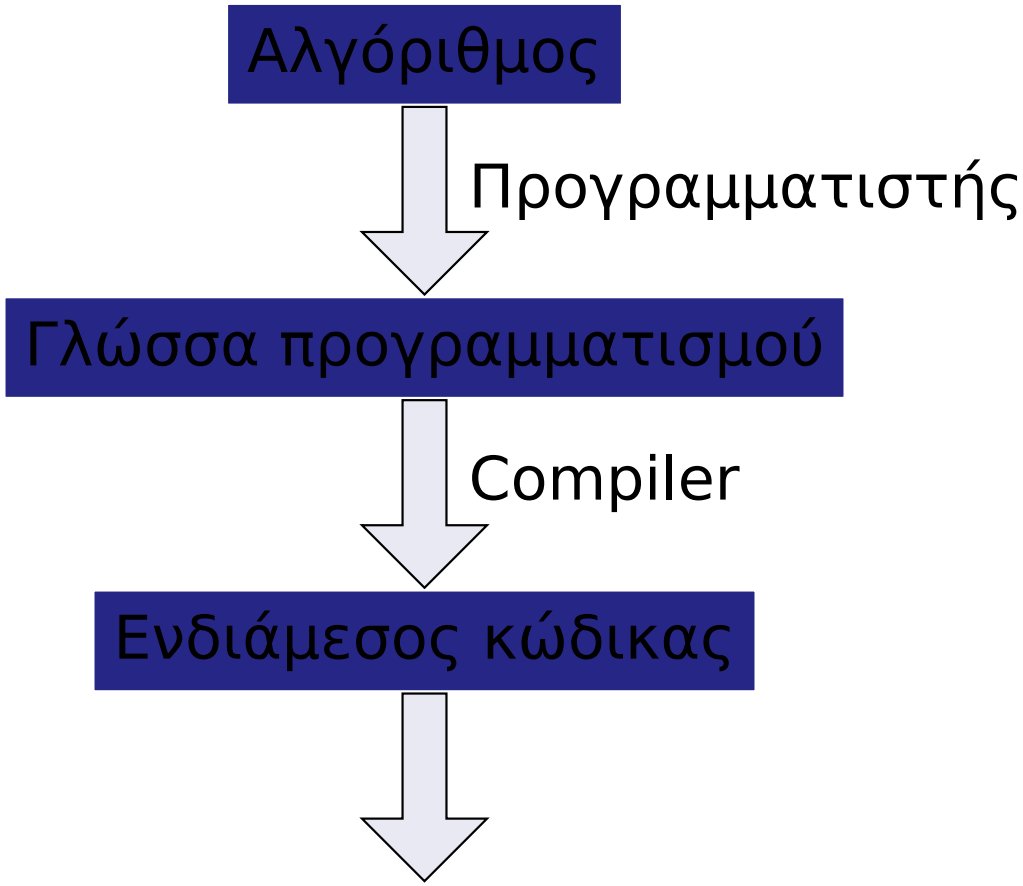
Δημιουργία προγράμματος



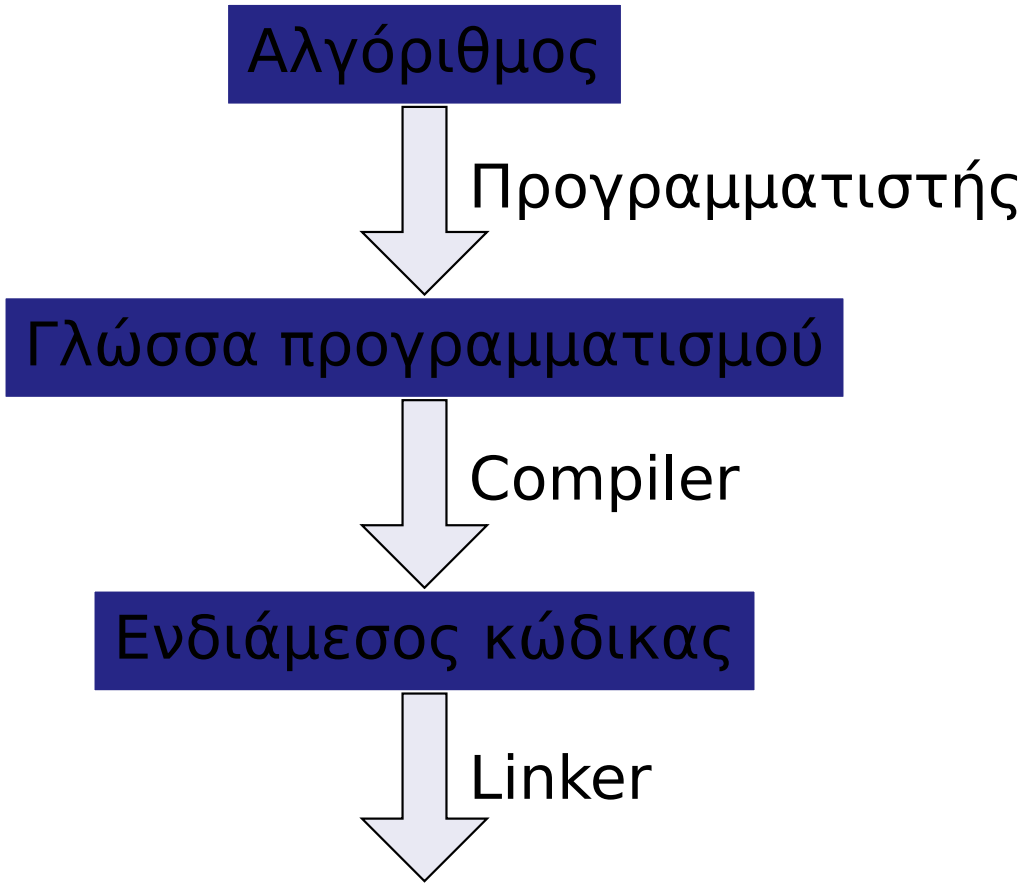
Δημιουργία προγράμματος



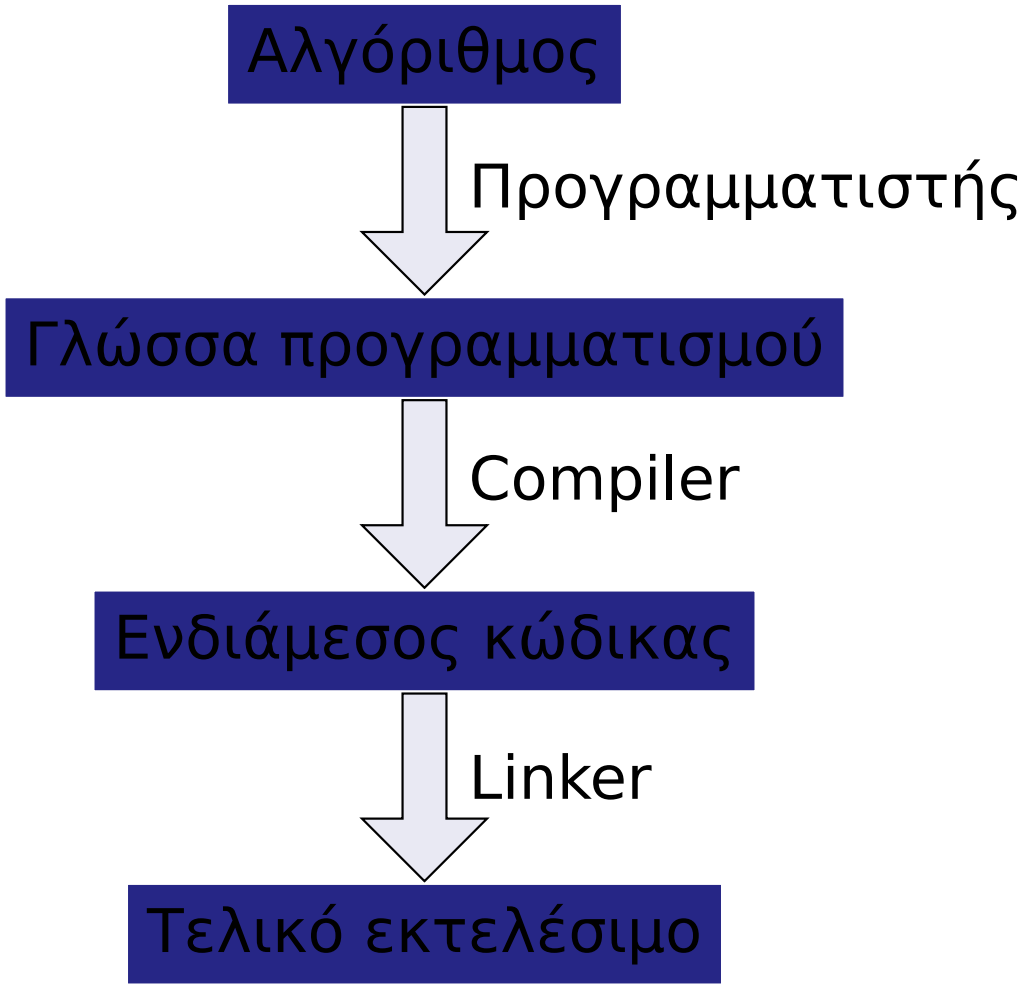
Δημιουργία προγράμματος



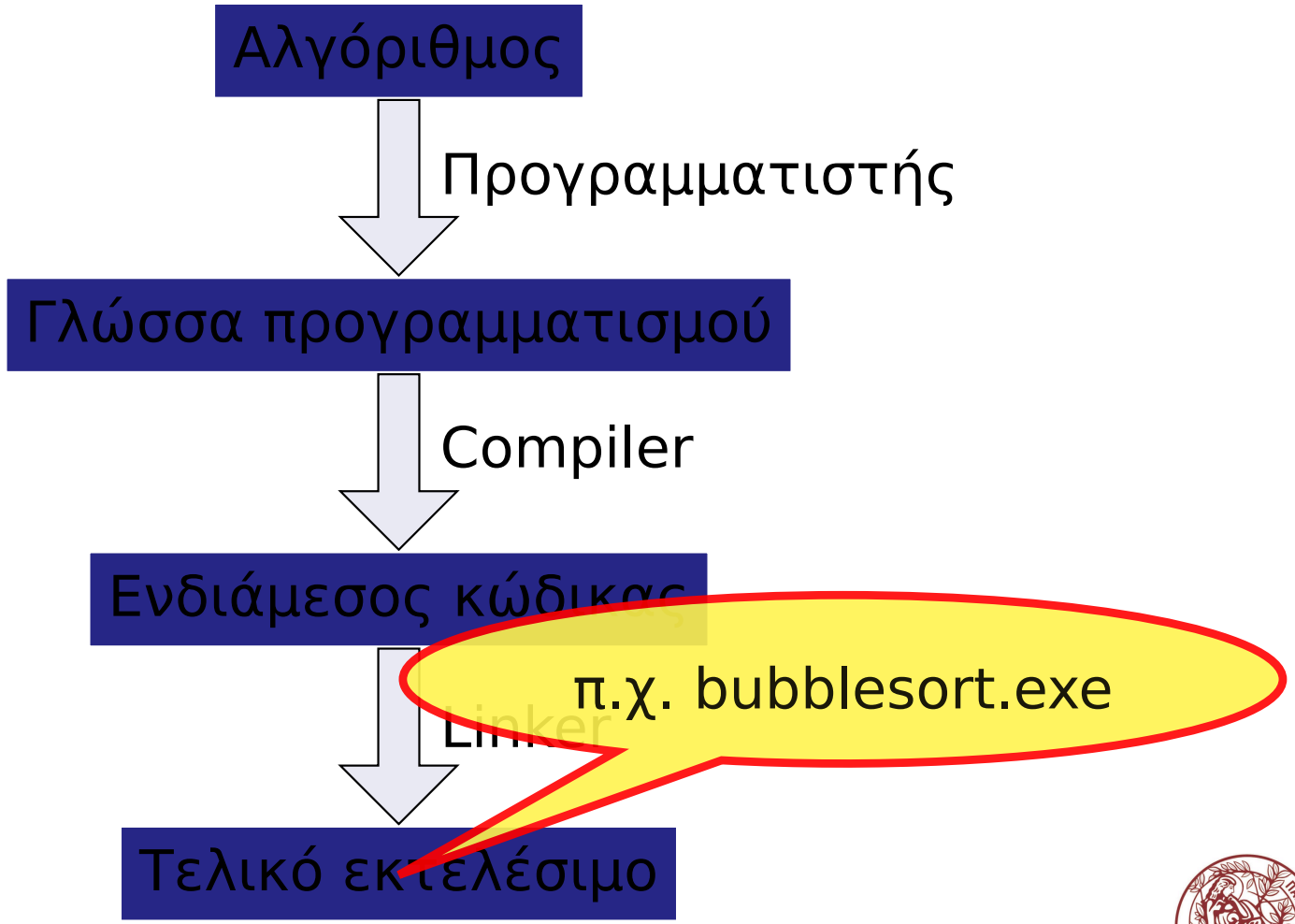
Δημιουργία προγράμματος



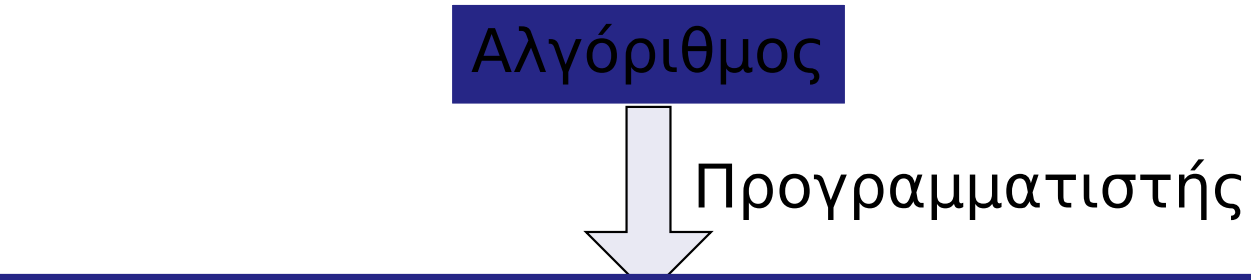
Δημιουργία προγράμματος



Δημιουργία προγράμματος



Δημιουργία προγράμματος

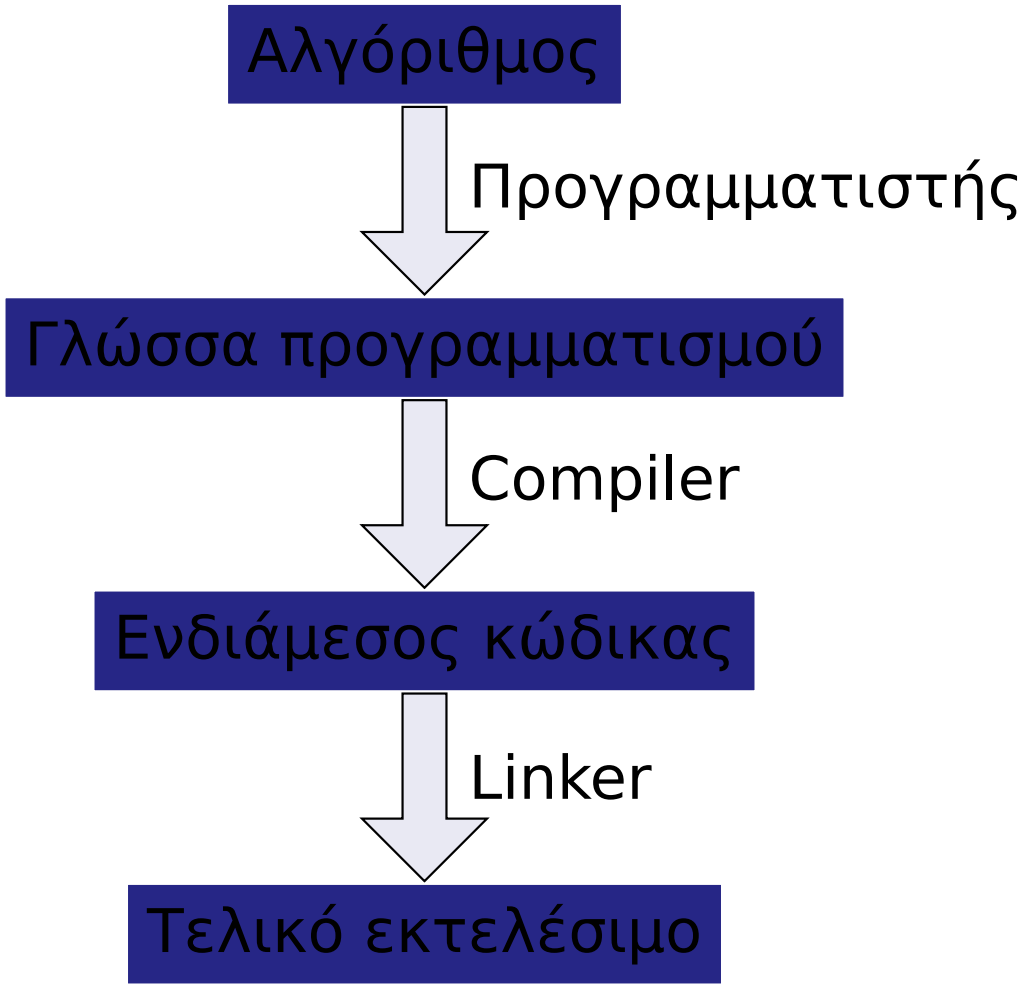


Στο Linux

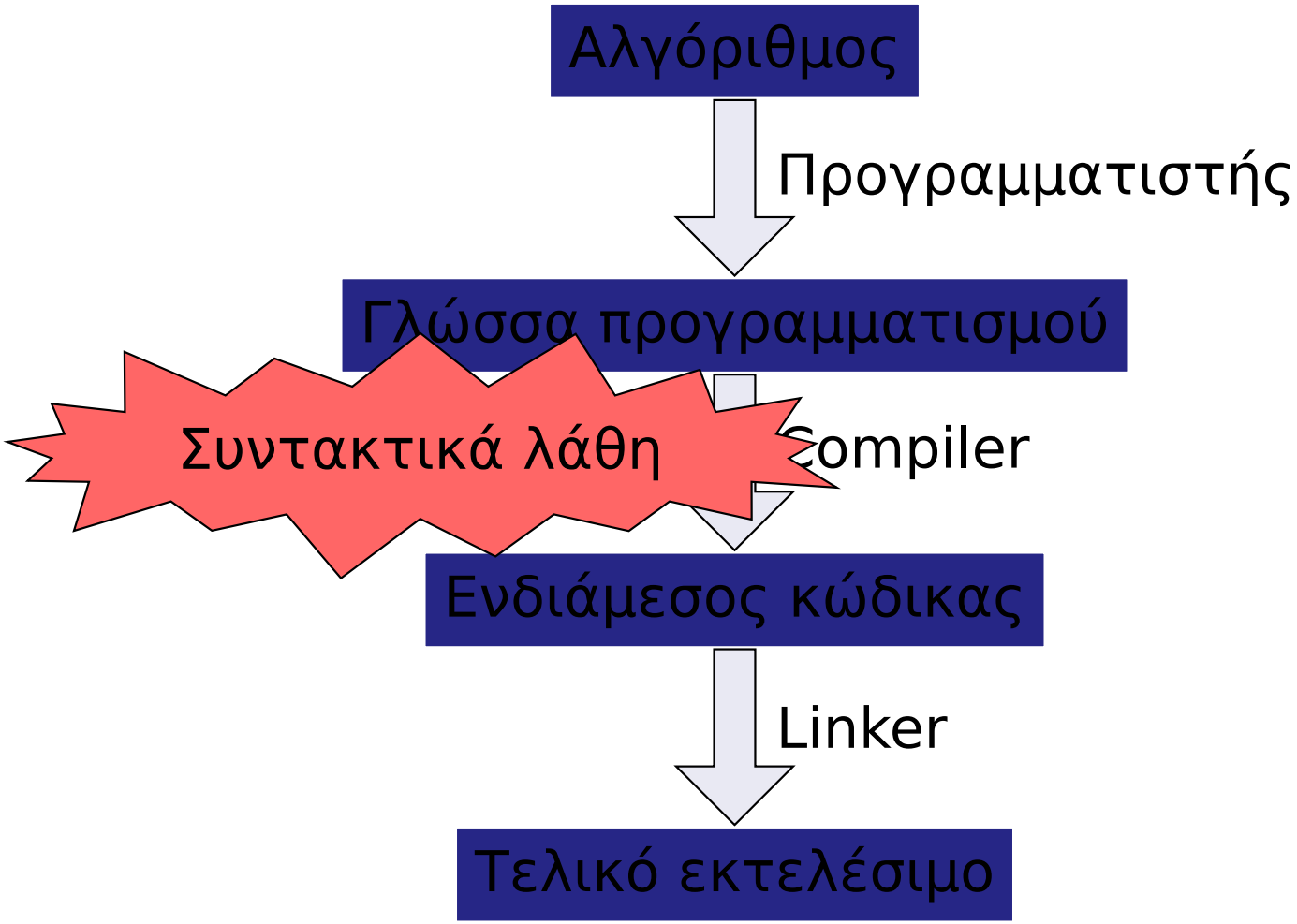
- Compile & Link:
  ```
  $ gcc -Wall -g bubblesort.c -o bubblesort
  ```
- Execute:
  ```
  $ ./bubblesort
  ```

Τελικό εκτελέσιμο

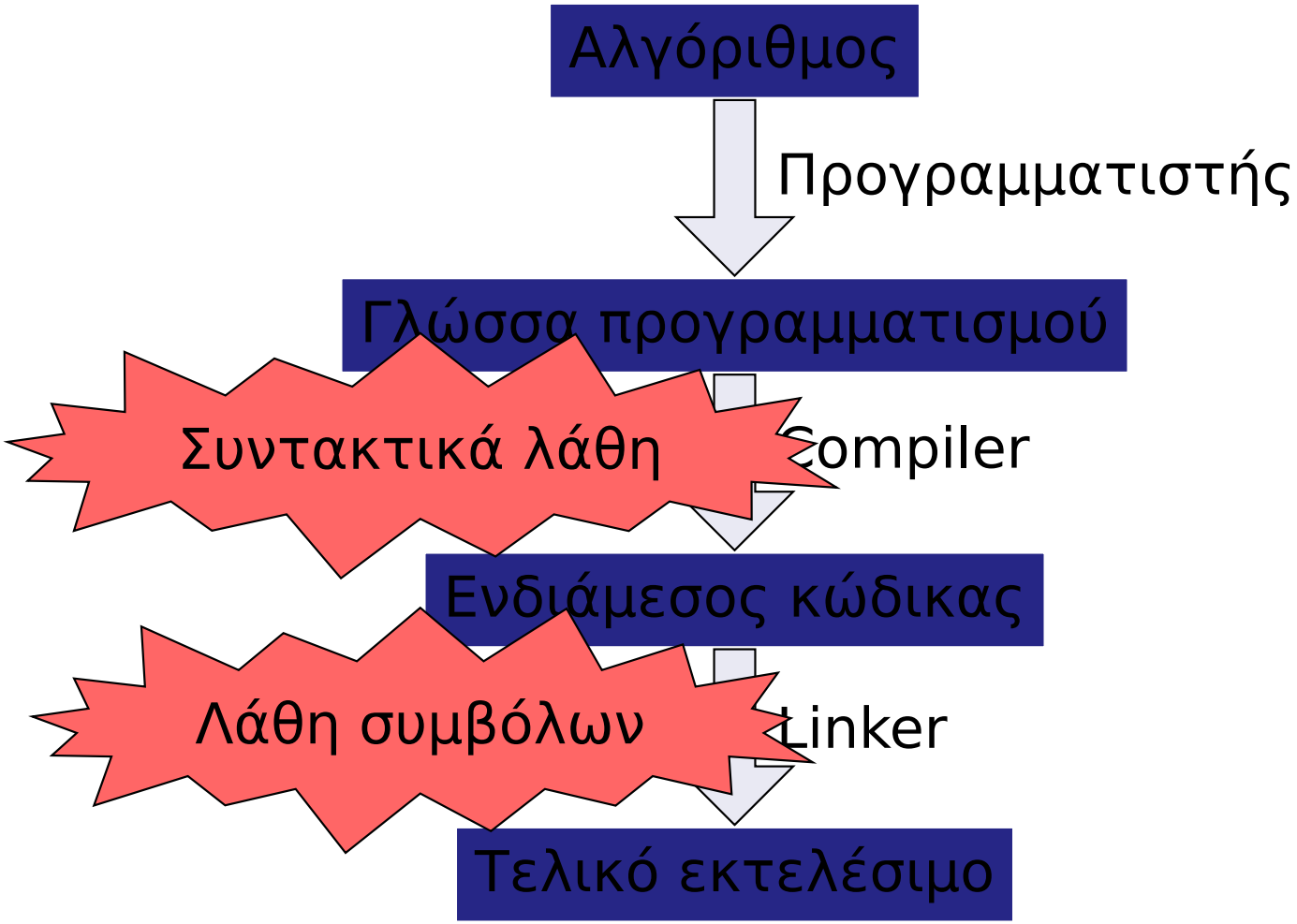
Δημιουργία προγράμματος



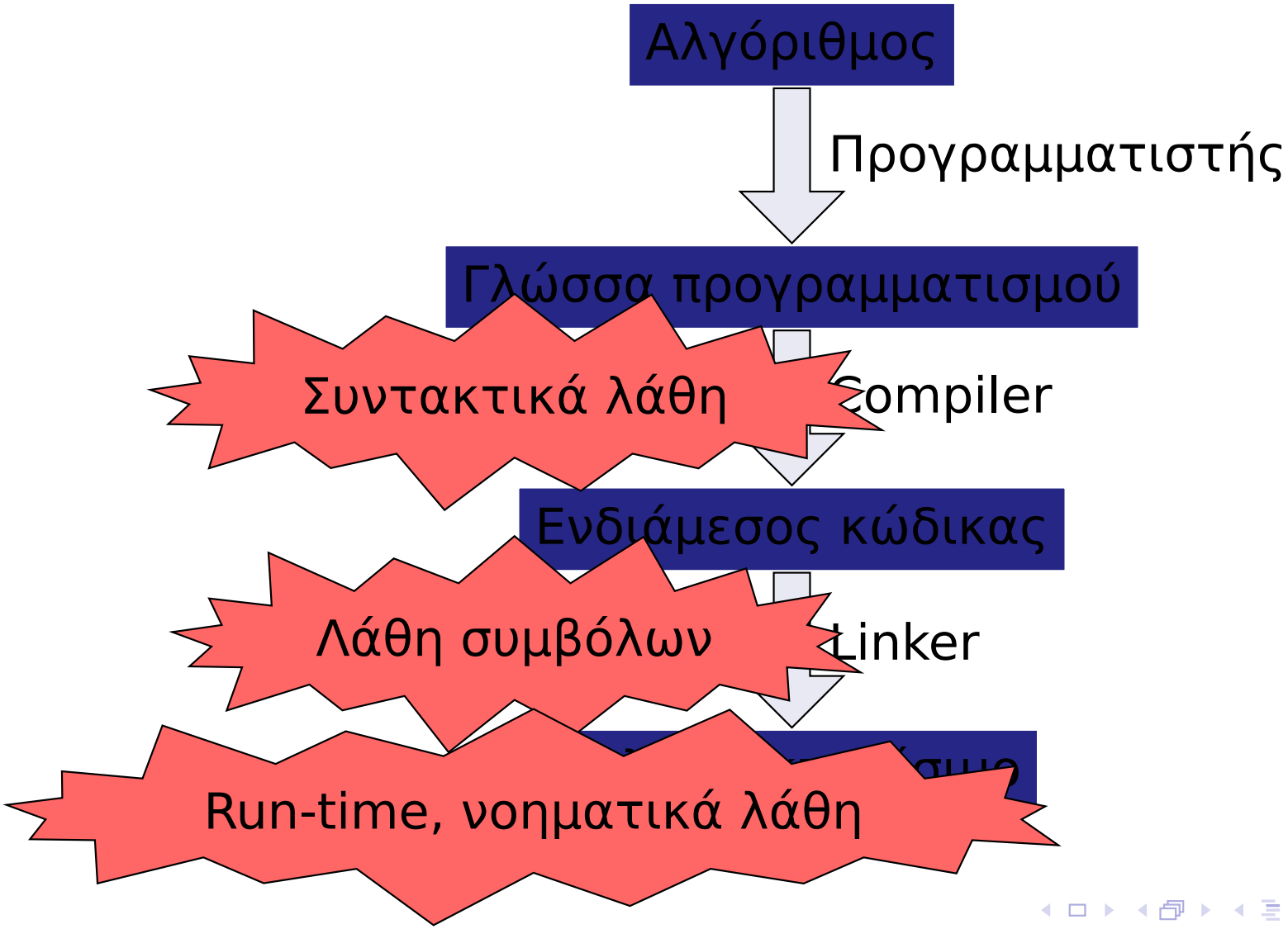
Δημιουργία προγράμματος



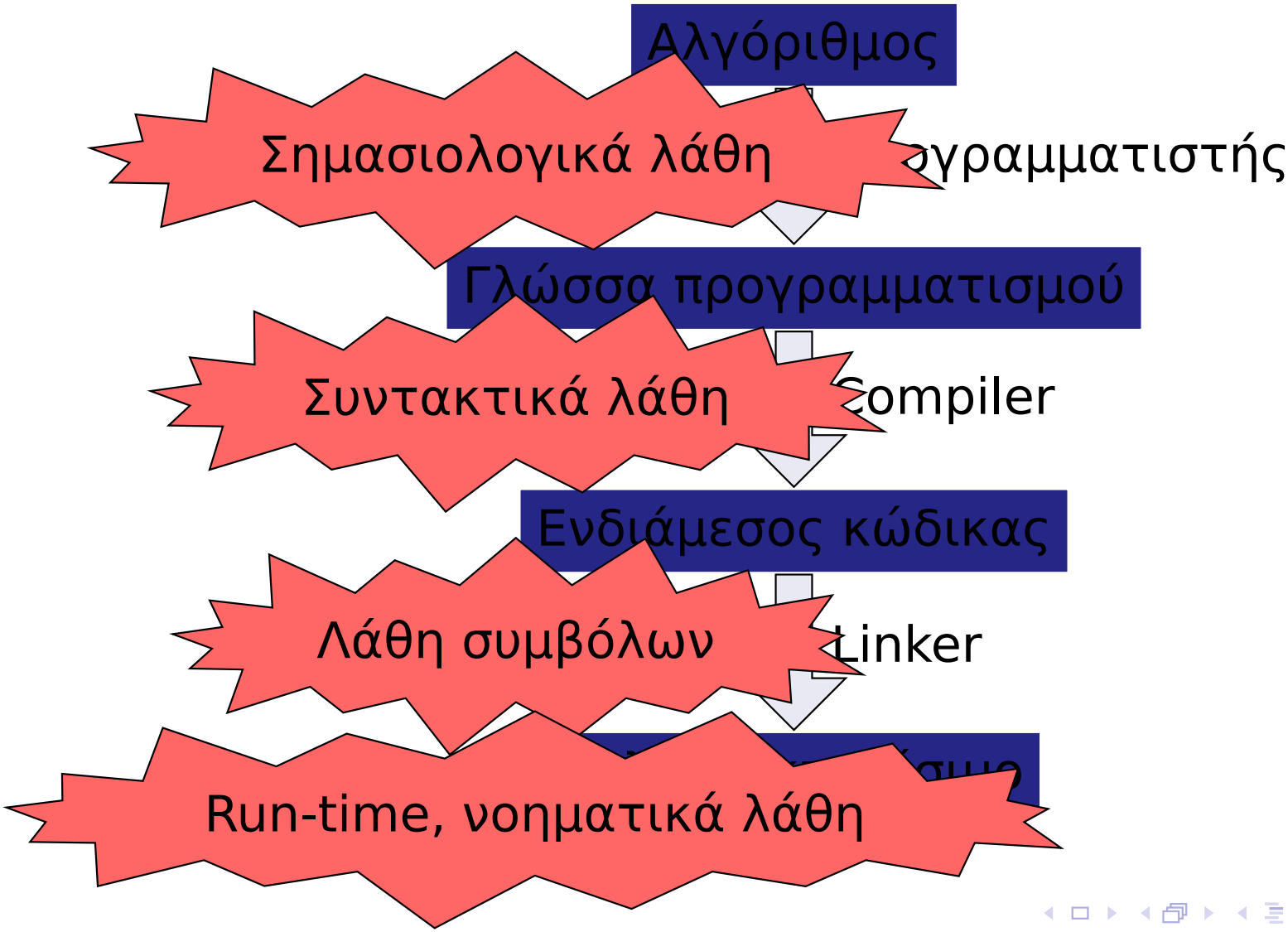
Δημιουργία προγράμματος



Δημιουργία προγράμματος



Δημιουργία προγράμματος



Διερμηνευόμενες Γλώσσες

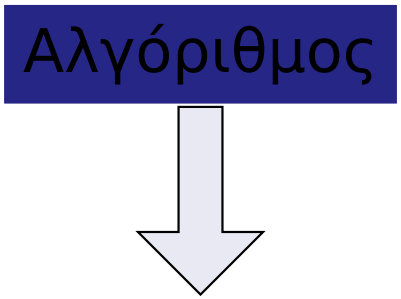
Αλγόριθμος

Διερμηνευόμενες Γλώσσες

π.χ. Bubble Sort

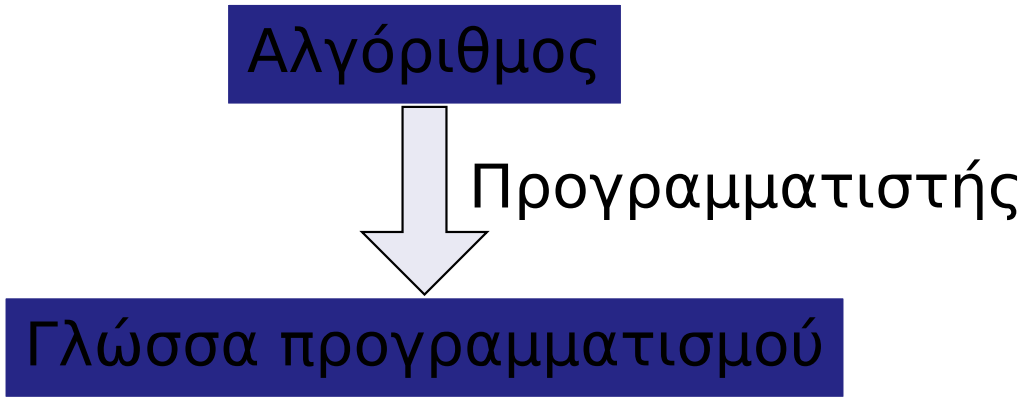
Αλγόριθμος

Διερμηνευόμενες Γλώσσες

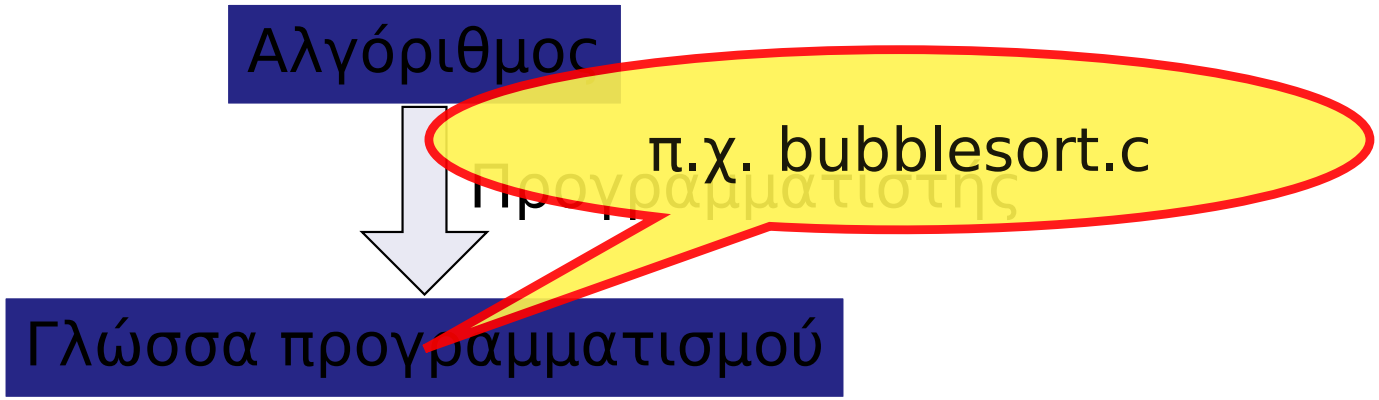


Διερμηνευόμενες Γλώσσες

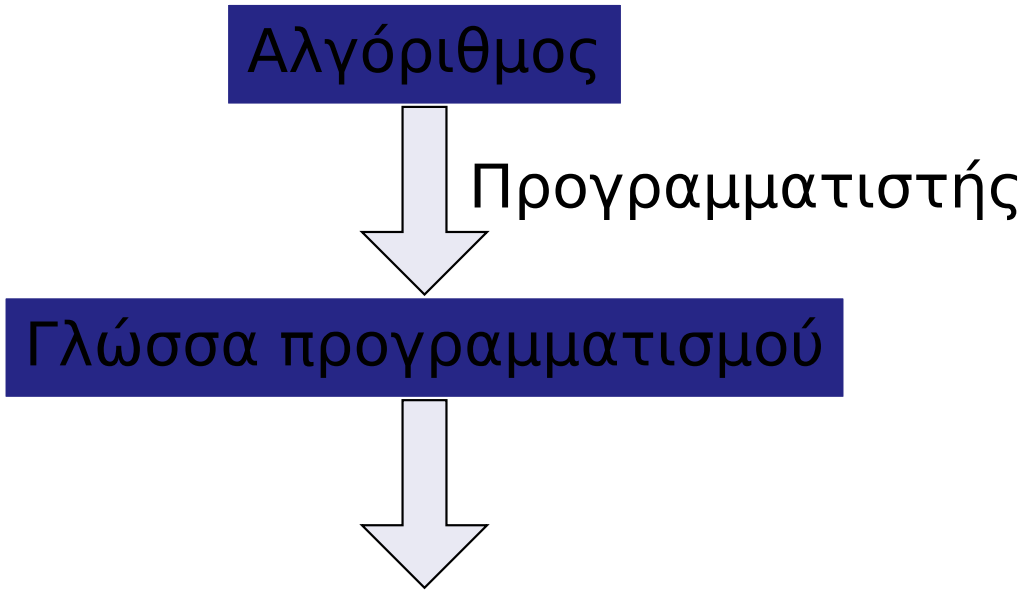
Διερμηνευόμενες Γλώσσες



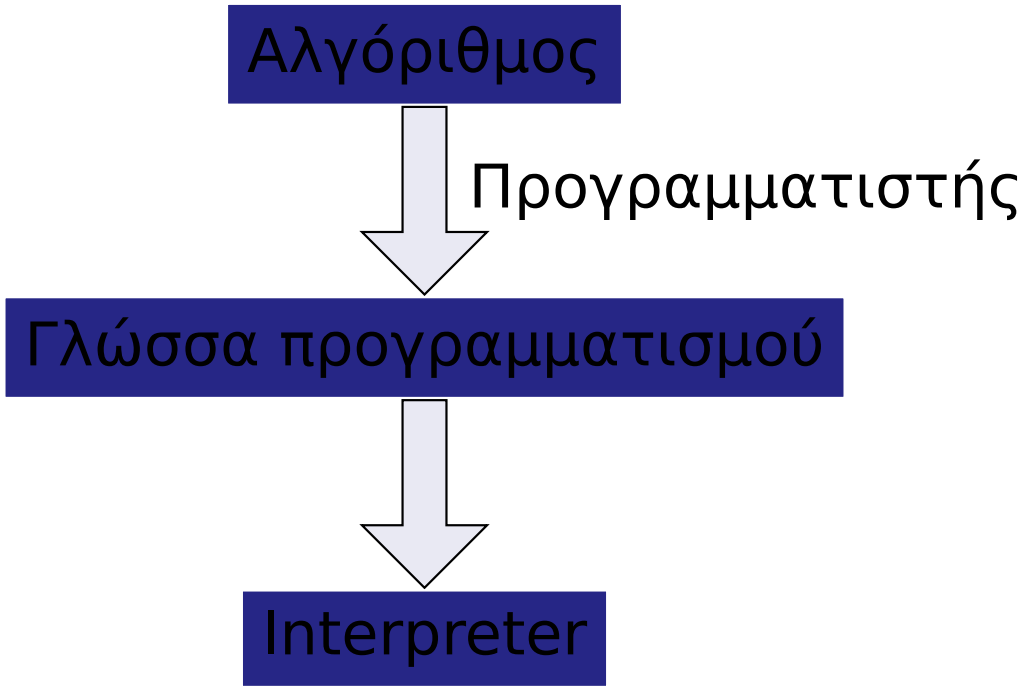
Διερμηνευόμενες Γλώσσες



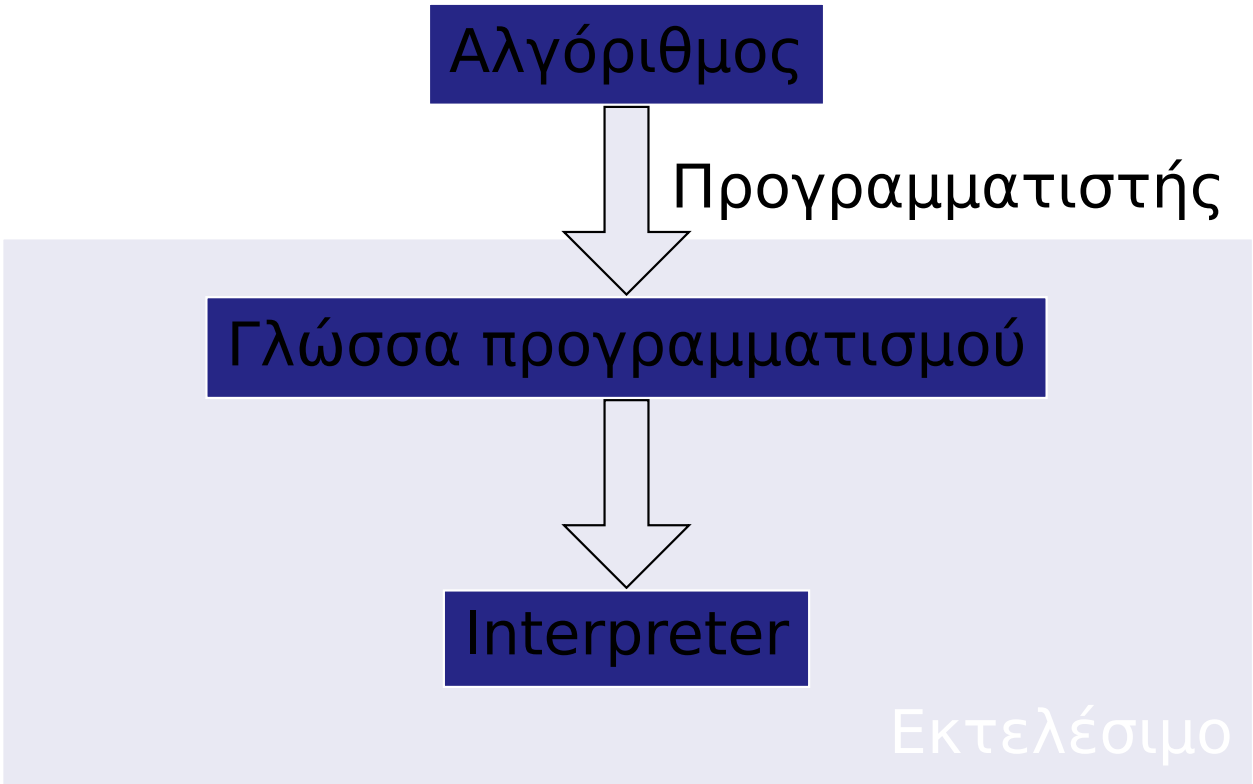
Διερμηνευόμενες Γλώσσες



Διερμηνευόμενες Γλώσσες



Διερμηνευόμενες Γλώσσες



Σωστό Πρόγραμμα

- Δεν έχει συντακτικά λάθη

```
int z
int x ==== 0;
floa pi = 3.14;
```

- Δεν έχει νοηματικά λάθη

```
int x = 0 + ”a”;
x = x + y;
```

- Δεν έχει σημασιολογικά λάθη

```
int x = 0;
int z = 3/x;
if (x = 3) {
  /* ... */
}
```

Debugging

- Λάθη χρόνου μεταγλώττισης: συντακτικά, νοηματικά
- Λάθη χρόνου εκτέλεσης: Bugs

Διαδικασία εκτέλεσης

- Φόρτωμα του κώδικα στη μνήμη
- Δέσμευση χώρου στη μνήμη για τις καθολικές μεταβλητές
- Εκτέλεση της συνάρτησης `main()`

- Οι τοπικές μεταβλητές;
  - On-the-fly: Κάθε συνάρτηση δεσμεύει όση μνήμη χρειάζεται όταν αρχίζει
  - Τελειώνοντας, απελευθερώνει ξανά τη μνήμη των τοπικών μεταβλητών