Προγραμματισμός Υπολογιστών με C++

ΕΙΣΑΓΩΓΗ

Γεώργιος Παπαϊωάννου (2013-20)
gepap@aueb.gr

Περιεχόμενο Παρουσίασης

- Περιγραφή:
  - Εισαγωγή στη C++
  - Ιστορικά στοιχεία για τη C/C++
  - Ένα βασικό πρόγραμμα
  - Μεταγλώττιση ενός προγράμματος

- Τελευταία ενημέρωση: Σεπτέμβριος 2020

Ιστορικά Στοιχεία: η γλώσσα C

- Δεκαετία '70: Ο D. Ritchie δημιουργεί στα Bell Labs τη **C** επηρεασμένος από τις B και BCPL.
- Η C χρησιμοποιείται στην ανάπτυξη του **UNIX**
- Υποκαθιστά σε πολλές περιπτώσεις τη χρήση συμβολογλώσσας (assembly)
- 1978: ***The C Programming Language*** των Kernighan & Ritchie
- Δεκαετία '80: Η C σταδιακά κυριαρχεί
- Πρώτα πρότυπα **ANSI C** (1989) και **ISO C** (1999)

Ιστορικά Στοιχεία: C και C++

- Δεκαετία '80: Ο B. Stroustrup δημιουργεί στα Bell Labs την **C++** ως επέκταση της C.
- Η C++ υποστηρίζει **αντικειμενοστρεφή**, **διαδικασιακό** και **γενικευμένο** προγραμματισμό.
- Πρώτο πρότυπο **ANSI/ISO C++** (1998).
- Κοινά υλοποιημένο πρότυπο: C++14 (2014)
- Πρόσφατο **νέο πρότυπο ISO C++17 (2017)**.
  - Πολλοί μεταγλωττιστές υποστηρίζουν ήδη τις περισσότερες προσθήκες της C++17
- Η C είναι **περίπου υποσύνολο** της C++, αλλά υπάρχουν εξαιρέσεις και πρακτικές της C που δεν συνιστώνται στην C++.

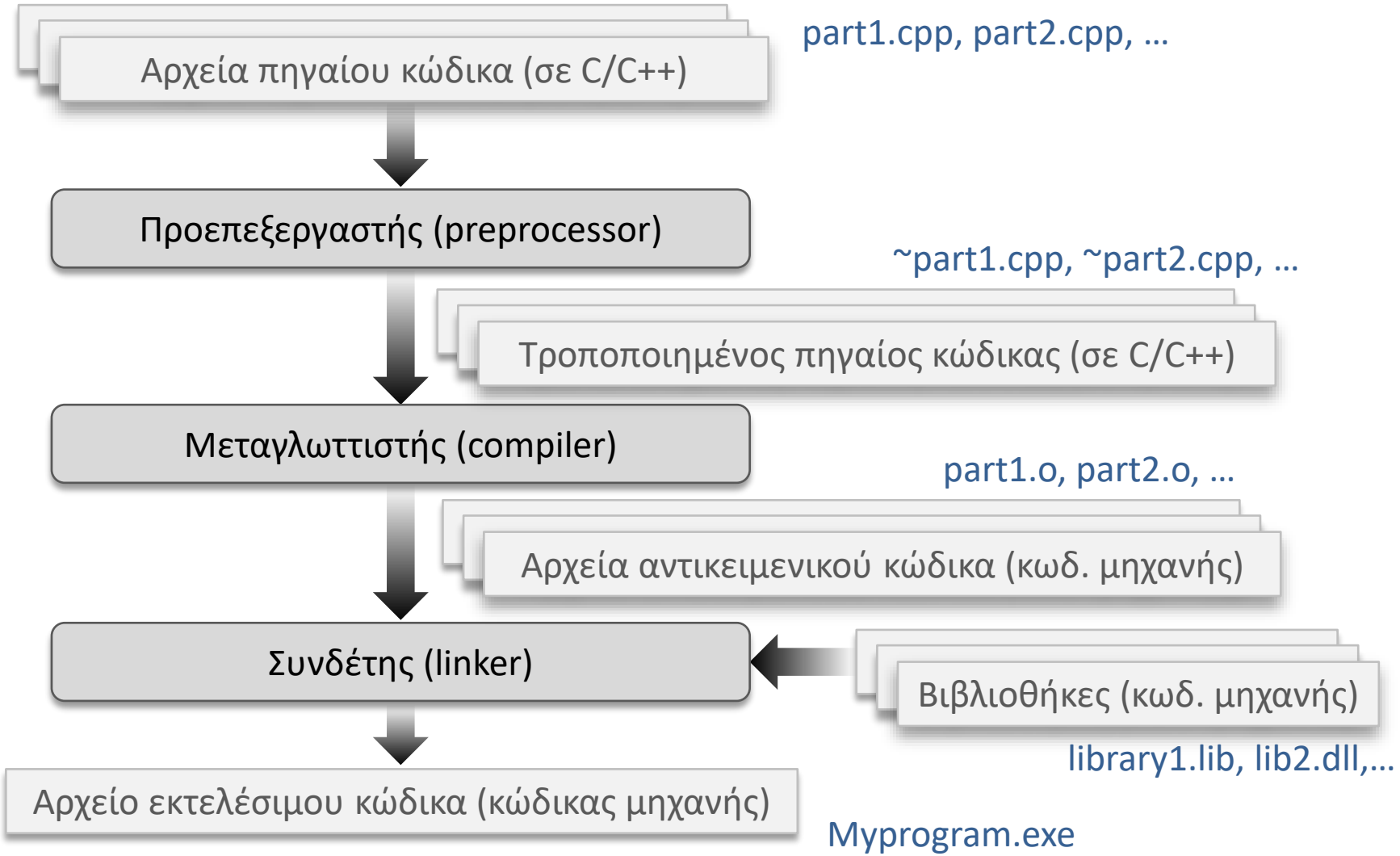
C++ και Java

- Δεκαετία '90: Η Sun (τώρα μέρος της Oracle) δημιουργεί την **Java**, δανειζόμενη πολλά στοιχεία από τη C++
- Παρόλο που η Java μοιάζει με τη C++, πρόκειται για **διαφορετικές γλώσσες**. Π.χ:
  - Η C++ επιτρέπει **συναρτήσεις που δεν είναι μέθοδοι κλάσεων**
  - Η C++ επιτρέπει τη **χρήση δεικτών** και δεν παρέχει αυτόματη **αποκομιδή απορριμμάτων**
  - Ο κάθε μεταγλωττιστής της C++ παράγει εκτελέσιμο κώδικα για **συγκεκριμένο ρεπερτόριο εντολών επεξεργαστή**
- C++: **περισσότερες δυνατότητες, πιο δύσκολη**

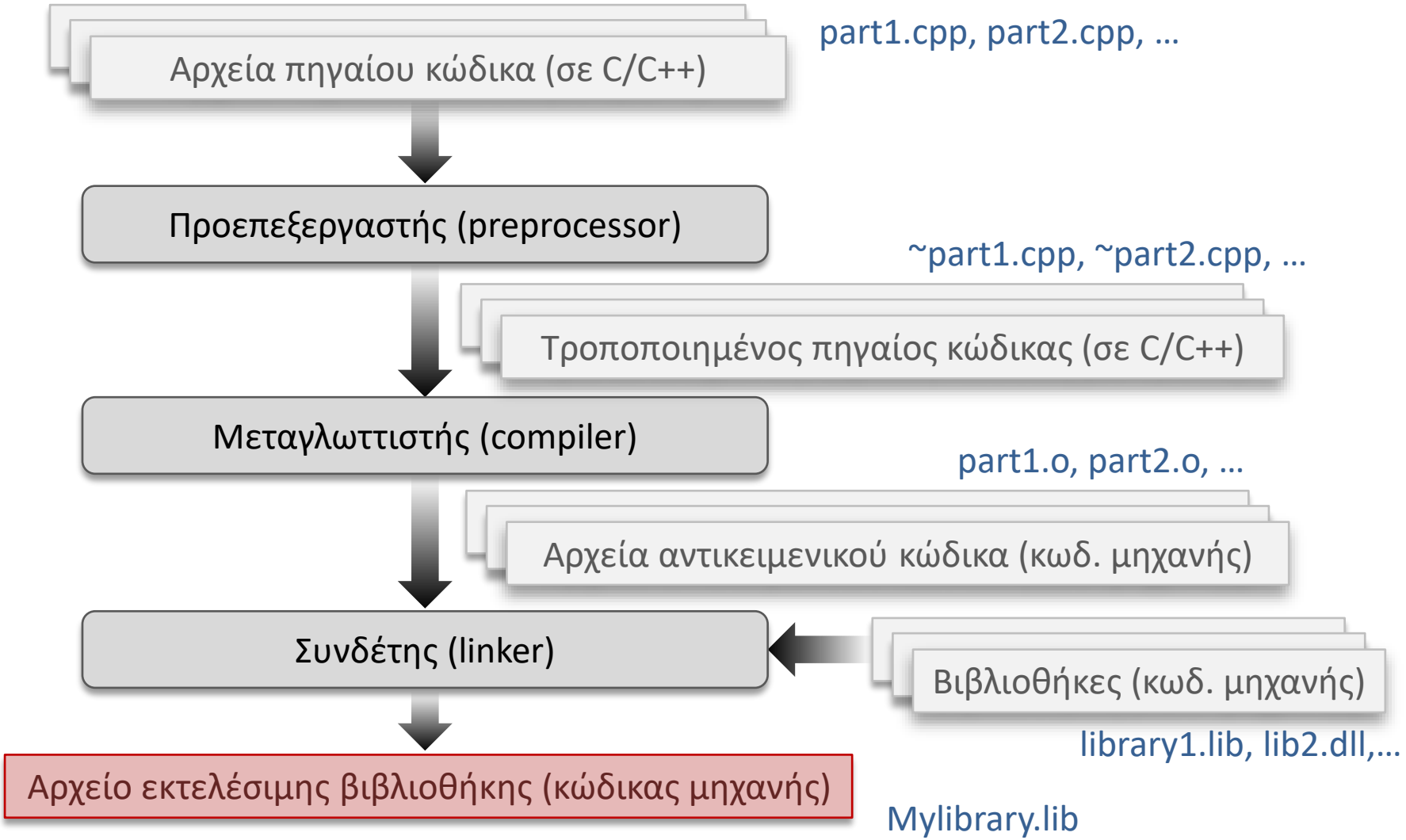
Ο Εκτελέσιμος Κώδικας

- Ο μεταγλωττιστής της C/C++ παράγει τελικά εκτελέσιμο κώδικα σε γλώσσα μηχανής
  - Εκτελέσιμα αρχεία (π.χ. .exe στα Windows)
  - Βιβλιοθήκες εκτελέσιμου κώδικα
    - Στατικές (.lib στα Windows, .o στο Unix/Linux)
    - Δυναμικές (.dll στα Windows, .so στο Unix/Linux)
- Τα εκτελέσιμα μπορούν να φορτωθούν ως αυτόνομα προγράμματα και να εκτελεστούν απευθείας σε μια συγκεκριμένη αρχιτεκτονική (π.χ. Win32, x64 κλπ)
- Τα εκτελέσιμα αρχεία δεν είναι μεταφέρσιμα σε άλλα λειτουργικά συστήματα (π.χ. από Windows σε Linux)

Η Μεταγλώττιση Προγραμμάτων



Η Μεταγλώττιση Βιβλιοθηκών



Τι διαφορές Εντόπισα σε Σχέση με τη Java;

C++	Java
Ύπαρξη σταδίου προεπεξεργασίας του κώδικα	Δεν υπάρχει αντίστοιχο στάδιο
Ο μεταγλωττιστής παράγει κώδικα μηχανής	Ο μεταγλωττιστής παράγει bytecode
Τα αρχεία αντικειμενικού κώδικα αντιστοιχούν σε αρχεία C/C++	Τα αρχεία που παράγει ο μεταγλωττιστής αντιστοιχούν σε κλάσεις που έχουν δηλωθεί
Η σύνδεση του κώδικα γίνεται κατά τη δημιουργία του προγράμματος (*)	Η σύνδεση του κώδικα και η μετατροπή σε κώδικα μηχανής γίνεται από τη JVM κατά το φόρτωμα των κλάσεων

(*) Οι βιβλιοθήκες που φορτώνονται δυναμικά έχουν άλλο μηχανισμό που πάλι πραγματοποιεί ένα στάδιο σύνδεσης

Ένα Πρώτο Πρόγραμμα σε C++

```
#include <iostream>       // Εντολή προς τον προεπεξεργαστή.
using namespace std;

int main( ) {
   cout << "Το πρώτο μου πρόγραμμα\n";
   cout << "Γεια σου " << "κόσμε!" << endl;
   /*
   cout << "Αυτό δεν τυπώνεται." << endl;
   cout << "Ούτε και αυτό." << endl;
   */
   return 0;  // Επιστρέφεται στο λειτουργικό. Όλα καλά.
}
```

Ένα Πρώτο Πρόγραμμα σε C++: Τι Βλέπω;

Έναρξη εντολής προεπεξεργαστή

Ενσωμάτωση δηλώσεων από υπάρχοντα κώδικα

```
#include <iostream>       // Εντολή προς τον προεπεξεργαστή.
using namespace std;

int main( ) {
    cout << "Το πρώτο μου πρόγραμμα (!)";
    cout << "Γεια σου " << "κόσμε!" << endl;
    /*
    cout << "Αυτό δεν τυπώνεται." << endl;
    cout << "Ούτε και αυτό." << endl;
    */
    return 0;
}
```

Δήλωση χρήσης του χώρου ονομάτων («πακέτου») std. Μας επιτρέπει να αναφερόμαστε στο αντικείμενο std::cout ως cout

Κύρια συνάρτηση του προγράμματος (code entry point)

Τελεστής που έχει επαπροσδιοριστεί (όχι, δεν είναι το binary shift!)

Εμβόλιμη Πληροφορία: Που Βάζουμε τα Άγκιστρα;

```
int main( ) {
   cout << "Το πρώτο μου πρόγραμμα\n";
   return 0;
}
```

+ Εξοικονόμηση γραμμών
+ Στοίχιση } με το όνομα

```
int main( )
{
   cout << "Το πρώτο μου πρόγραμμα\n";
   return 0;
}
```

+ Ξεκάθαρη αντιστοίχιση { με }
+ Συμβατό με preprocessor conditionals
- Σπάταλο σε γραμμές

- Πολλές παραλλαγές
  - Βλ. http://www.riedquat.de/prog/style
- Υπάρχει κάποια καθιερωμένη;
  - ΟΧΙ !!

Τι διαφορές Εντόπισα σε Σχέση με τη Java; (1)

C++	Java
Στον κώδικά μου ενσωματώνω (#include) δηλώσεις κώδικα από εξωτερικά αρχεία, δικά μου, άλλων ή του συστήματος	Στον κώδικά μου εισάγω (γνωστοποιώ - #import) δηλώσεις κλάσεων από άλλα πακέτα, δικά μου ή άλλων
Διαθέτω στάδιο προεπεξεργασίας στο οποίο: • Εκτελώ «μακροεντολές» τροποίησης και εισαγωγής κώδικα (βλ. Preprocessor directives) • Επιτρέπω την «υπό συνθήκη» μεταγλώττιση κώδικα • Επιτρέπω την παραμετροποίηση του κώδικά μου (βλ. templates) • Επιτρέπω τη χρήση διαφορετικού / εξειδικευμένου κώδικα για κάθε αρχιτεκτονική	Δεν υπάρχει η έννοια της προεπεξεργασίας του κώδικα

Τι διαφορές Εντόπισα σε Σχέση με τη Java; (2)

C++	Java
Ο κώδικάς μου μπορεί να βρίσκεται σε μια ελεύθερη συνάρτηση ή διαδικασία που δεν ανήκει σε κάποια κλάση (π.χ. εδώ η main())	Όλος ο κώδικας υποχρεωτικά εντάσσεται σε μεθόδους κάποιας κλάσης Δεν υπάρχουν «συναρτήσεις»
Μπορώ να υπερφορτώσω και να επαναπροσδιορίσω όλους τους γνωστούς τελεστές σε μια κλάση, ακόμα και τους: =, <, new, [ ]	Δεν υπάρχει η έννοια του custom τελεστή

Ο Κώδικας μετά την Προεπεξεργασία

```
Ενσωματωμένες δηλώσεις από το iostream…

…
…
…

using namespace std;


int main( )
{
   cout << "Το πρώτο μου πρόγραμμα\n";
   cout << "Γεια σου " << "κόσμε!" << endl;
   return 0;
}
```

Το #include <iostream> εισάγει εδώ τα περιεχόμενα του αρχείου κεφαλίδας iostream. Το αρχείο αυτό υπάρχει σε έναν ειδικό κατάλογο της γλώσσας. Περιέχει δηλώσεις που χρειάζονται π.χ. για να χρησιμοποιήσουμε το cout.

Τα σχόλια απομακρύνονται

Ένα πιο Σύνθετο Πρόγραμμα

```
#include <iostream>
using namespace std;


int compute(int arg) {
    return 2 * arg;
}

int main( ) {
    int i = 5;
    cout << compute(i) << endl
         << i << endl;
    return 0;
}
```

Δήλωση «ελεύθερης» συνάρτησης

Χρήση συνάρτησης

Δηλώσεις και Ορισμοί Συναρτήσεων

```
#include <iostream>
using namespace std;

int compute(int arg);

int main( )
{
   int i;
   cin >> i;
   cout << compute(i) << endl;
 return 0;
}

int compute(int arg)
{
   return 2 * arg;
}
```

Δήλωση (declaration) συνάρτησης

Χρήση της συνάρτησης πριν τον προσδιορισμό (του περιεχομένου) της

Προσδιορισμός (definition) συνάρτησης:

Το σώμα μιας συνάρτησης μπορεί να δοθεί σε άλλο σημείο του κώδικα ή και σε άλλο αρχείο

Τι διαφορές Εντόπισα σε Σχέση με τη Java;

C++	Java
Μια συνάρτηση (ή έναν τύπο δεδομένων π.χ. μια κλάση) μπορώ να τα δηλώσω σε ένα σημείο του κώδικα και να τα προσδιορίσω αλλού Έτσι αποσυσχετίζω τη δήλωση («υπογραφή») μιας συνάρτησης από την υλοποίησή της	Η δήλωση μιας μεθόδου και η υλοποίησή της είναι ενοποιημένες. Μόνο οι διεπαφές και οι απροσδιόριστες μέθοδοι δε συνοδεύονται από υλοποίηση αλλά δε μπορούν να προσδιοριστούν σε κάποιο άλλο σημείο για την ίδια δήλωση κλάσης
Μπορώ να έχω χωριστά αρχεία δηλώσεων («header» files - .h) και χωριστά αρχεία υλοποίησης (.cpp files)	Η υλοποίηση και η δήλωση μιας κλάσης και των μεθόδων της βρίσκονται στο ίδιο αρχείο υποχρεωτικά (.java)
Η ονομασία ενός αρχείου δεν έχει καμία σχέση με το περιεχόμενό του	Το όνομα του αρχείου πρέπει να ταυτίζεται με τη μοναδική public κλάση μέσα σε αυτό

Διαδικαστικός Προγραμματισμός

- Στη C/C++ μπορώ να οργανώσω τον κώδικά μου ως μια ιεραρχική κλήση αυτόνομων συναρτήσεων ή «διαδικασιών»
- Η εκτέλεση ξεκινά από μία κεντρική συνάρτηση (τη main), η οποία καλεί αναδρομικά με τη σειρά όλες τις περιεχόμενες συναρτήσεις
- Μια «αμιγής» διαδικασία (procedure) μπορεί να είναι μια συνάρτηση που δεν επιστρέφει τίποτα (void)

Διαδικαστικός Προγραμματισμός: Παράδειγμα

```
#include <iostream>
using namespace std;

void compute(float);

int main( )
{
   float f;
   cin >> f;
   compute(f);
   return 0;
}

void compute(float arg)
{
   cout << arg / 2 << endl;
}
```

Στις δηλώσεις συναρτήσεων και μεθόδων, τα ονόματα είναι προαιρετικά (γιατί;) αλλά για να είναι ο κώδικας ευανάγνωστος να τα βάζετε πάντα!

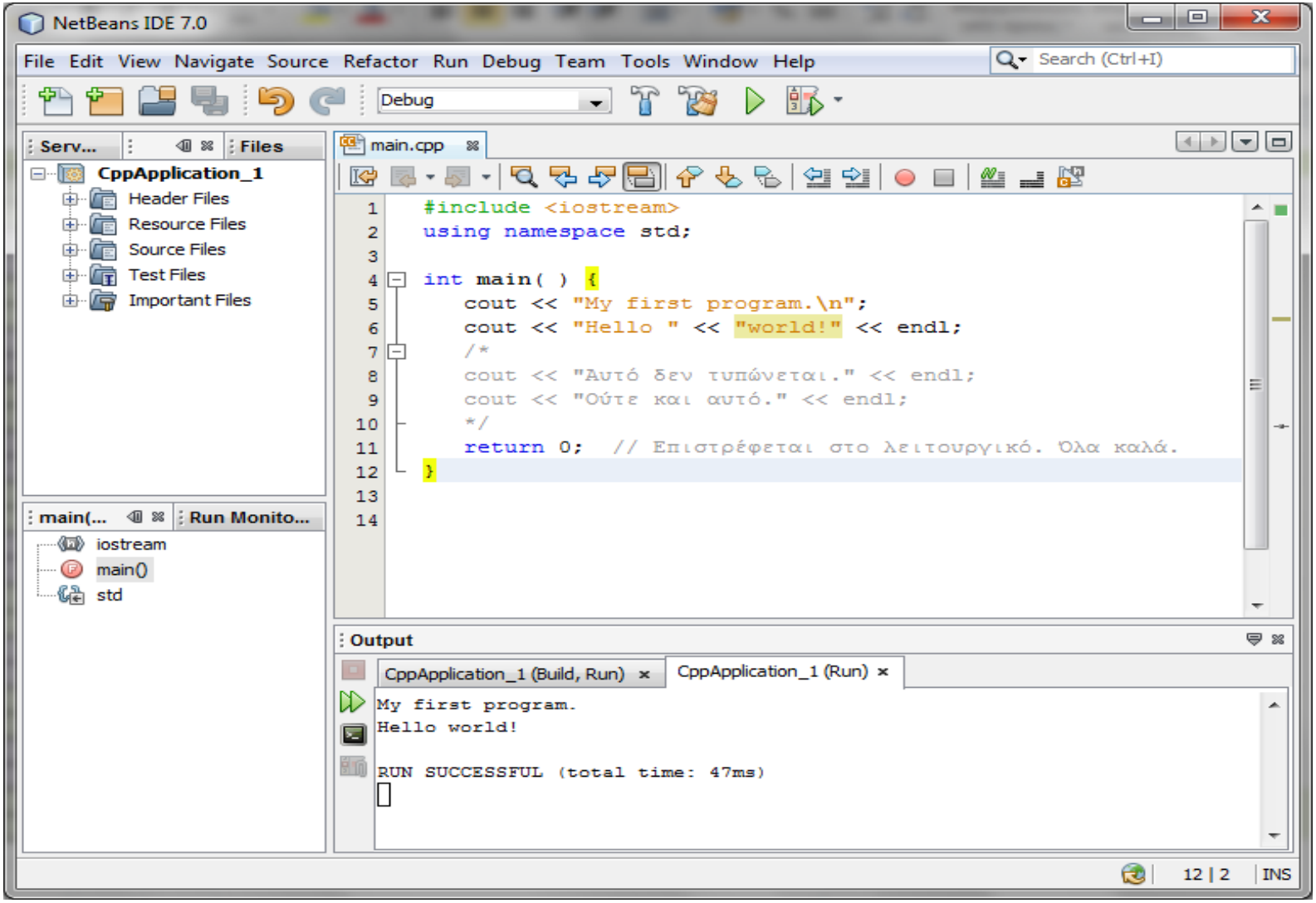
Διαθέσιμοι Μεταγλωττιστές και Εκδόσεις C++

- Για το μάθημα, μπορείτε να χρησιμοποιήσετε είτε τον GCC, είτε τον Microsoft C/C++ Compiler μέσω του Integrated Development Environment (IDE) Visual Studio
- GCC:
  - Διατίθεται ελεύθερα
  - Μπορεί να χρησιμοποιηθεί από γραμμή εντολών σε **Unix**
  - Σε **Windows** μπορεί να χρησιμοποιηθεί μέσω της γραμμής εντολών της συλλογής **Cygwin**, ή μέσω του περιβάλλοντος **MinGW**.
  - Μπορεί να χρησιμοποιηθεί και μέσω των **Eclipse** και **NetBeans**.
- Visual C++:
  - **Συστήνεται η χρήση του δωρεάν Microsoft Visual Studio Community Edition**.

Μεταγλώττιση από γραμμή εντολών

- Όλα τα βήματα μαζί:
  - g++ -o myprog.exe part1.cpp part2.cpp part3.cpp
- Δημιουργία αντικειμενικού κώδικα χωρίς σύνδεση:
  - g++ -c part1.cpp part2.cpp part3.cpp
  - g++ -c part2.cpp (αν έγιναν αλλαγές μόνο σε αυτό το αρχείο .cpp)
- Σύνδεση αρχείων αντικειμενικού κώδικα:
  - g++ -o myprog.exe part1.o part2.o part3.o

Χρήση GCC μέσα από το NetBeans



Χρήση Visual C++