Computer Programming

DRAGOS AROTARITEI
dragos.arotaritei@umfiasi.ro

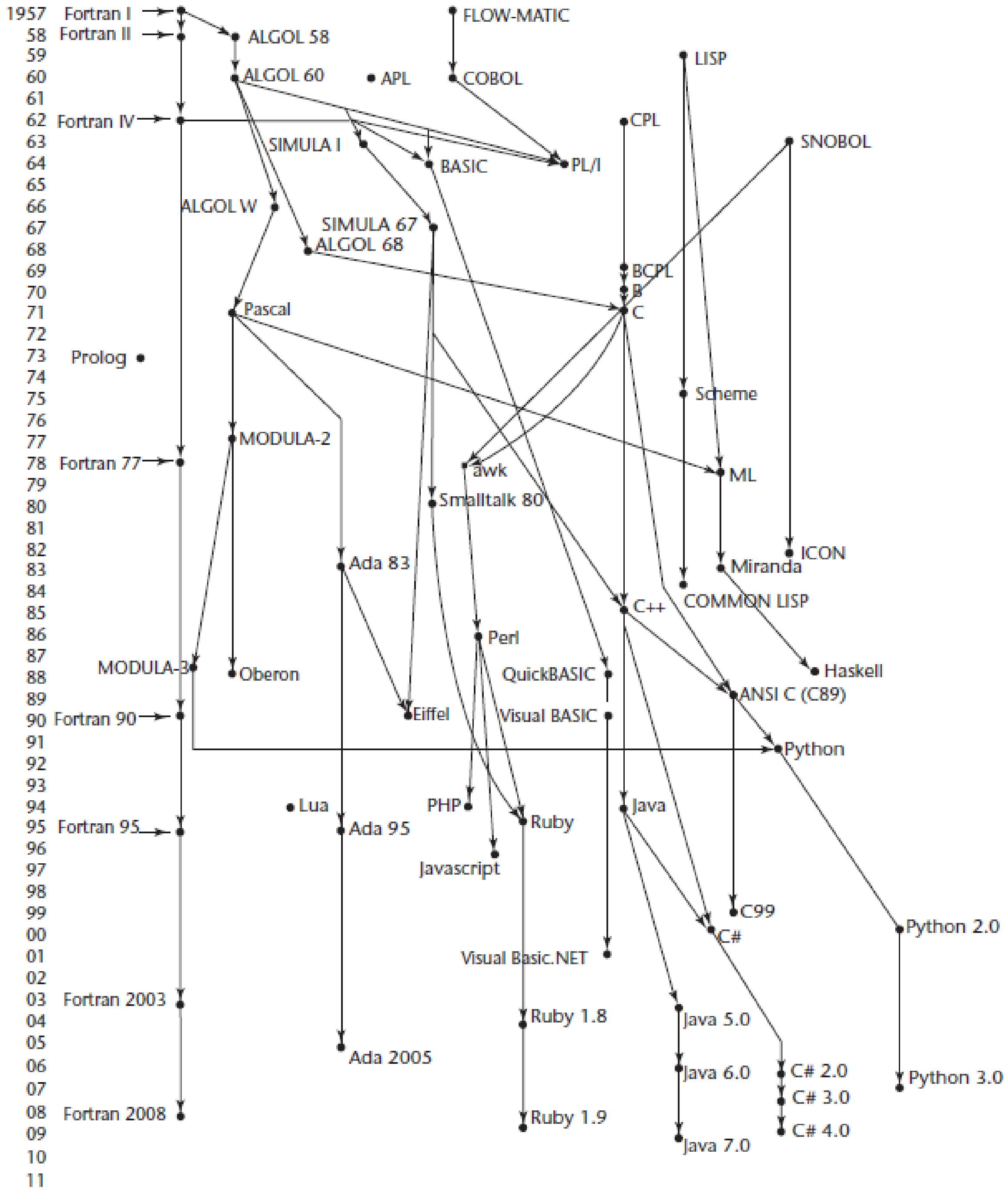
University of Medicine and Pharmacy "Grigore T. Popa" Iasi
Department of Medical Bioengineering
Romania

May 2016

Introduction

- History - before1950’s
- What was the first computer programming language?
- Officially, the first programming language for a computer was Plankalkül - developed by Konrad Zuse for the Z3 (first working computer based on Turing complete machine, constructed in 1941) between 1943 and 1945. However, it was not implemented until 1998.
- First high-level programming language, Short Code, which was proposed by John Mauchly in 1949. It was designed to represent mathematical expressions in a format readable by human beings.
- However, because it had to be translated into machine code before it could be executed, it had relatively slow processing speeds.
- Other early programming languages were developed in the 1950s and 1960s, including Autocode, COBOL, FLOW-MATIC, and LISP. Of these, only COBOL and LISP are still in use today.

- 1972 C language, Dennis Ritchie (*How was made the first C compiler, written in C?*)
- 1983, C++ , Bjarne Stroustrup C with Classes
- How many programming languages exists in the world? More than 500 but probably in reality the number of programming languages goes to over 2750!
- A criteria classification – imperative and declarative
- An imperative program consists of explicit commands for the computer to perform. (e.g. Visual Basic, Java, Visual C++.net, Visual C#)
- A declarative programming, focuses on what the program should accomplish without specifying how the program should achieve the result (relational or functional language), e.g. HTML, MXML, XAML, XSLT, LISP
- Other classifications exists, e.g. procedural, event-driven, object-oriented, declarative, scripts, Page Description Language, and Functional)



Genealogy of computer language

Why Computer Languages?

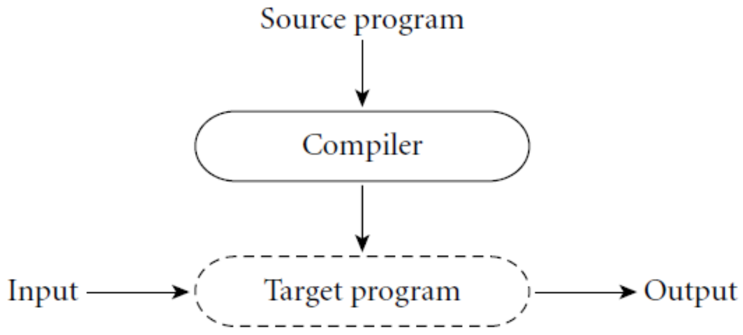
- Understanding computer languages can help us to choose one that is the most appropriate one for a specific task.
- C, C++, C# or C++/CLI[a] for systems programming or desktop applications?
- Fortran, C or Phyton for scientific computations?
- PHP or Ruby for a web-based application?
- Visual Basic, Visual C++ or Java for a graphical user interface?
- C, Basic or Assembly Languages for embedded systems?
- VBScript for EXCEL?
- What language we can choose for grid programming (parallel programming)?
- What could be a good option for mobile programming?

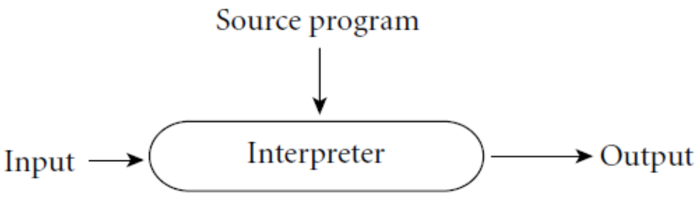
- Most languages are better for some things meanwhile others are most suitable for other types of applications.

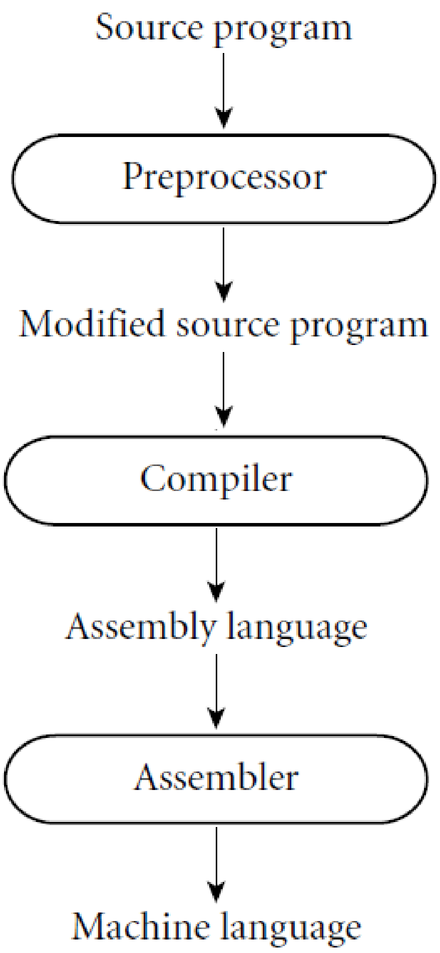
[a] C++/CLI (C++ modified for Common Language Infrastructure) is a language specification created by Microsoft and intended to supersede Managed Extensions for C++.

Compilers and Interpreters

- The compilation (and linking for some languages) and execution of a program in high level language
- The compiler translate a high level source code written in a programming language into a into an equivalent target program (typically in machine language) - .exe, .com, etc. The application can run independently.
- In interpreter execute line by line one application. The interpreter needs a a virtual machine behind them (or an interpreter environment) in order to execute instructions.
- Several scripting languages (e.g. Perl, Tcl, Python, and Ruby) can write new pieces of itself and execute them on the fly.







Source program

Preprocessor

Modified source program

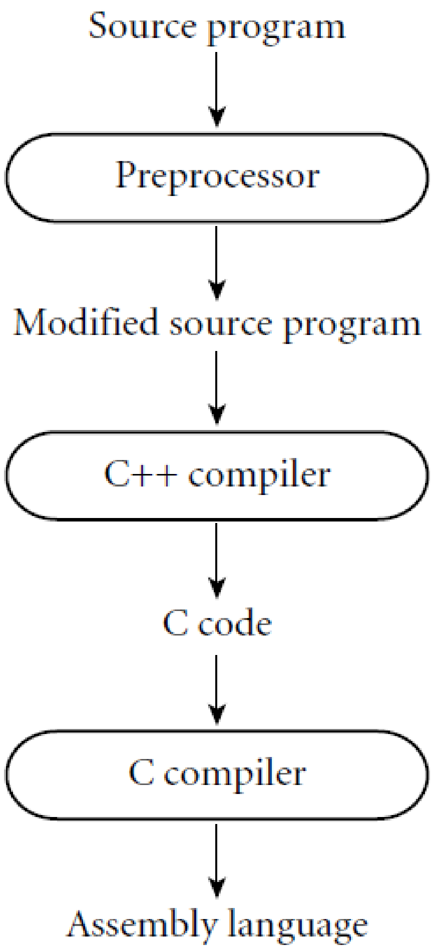
Compiler

Assembly language

Assembler

Machine language

Basic C compiler



Source program

Preprocessor

Modified source program

C++ compiler

C code

C compiler

Assembly language

C++ Compiler

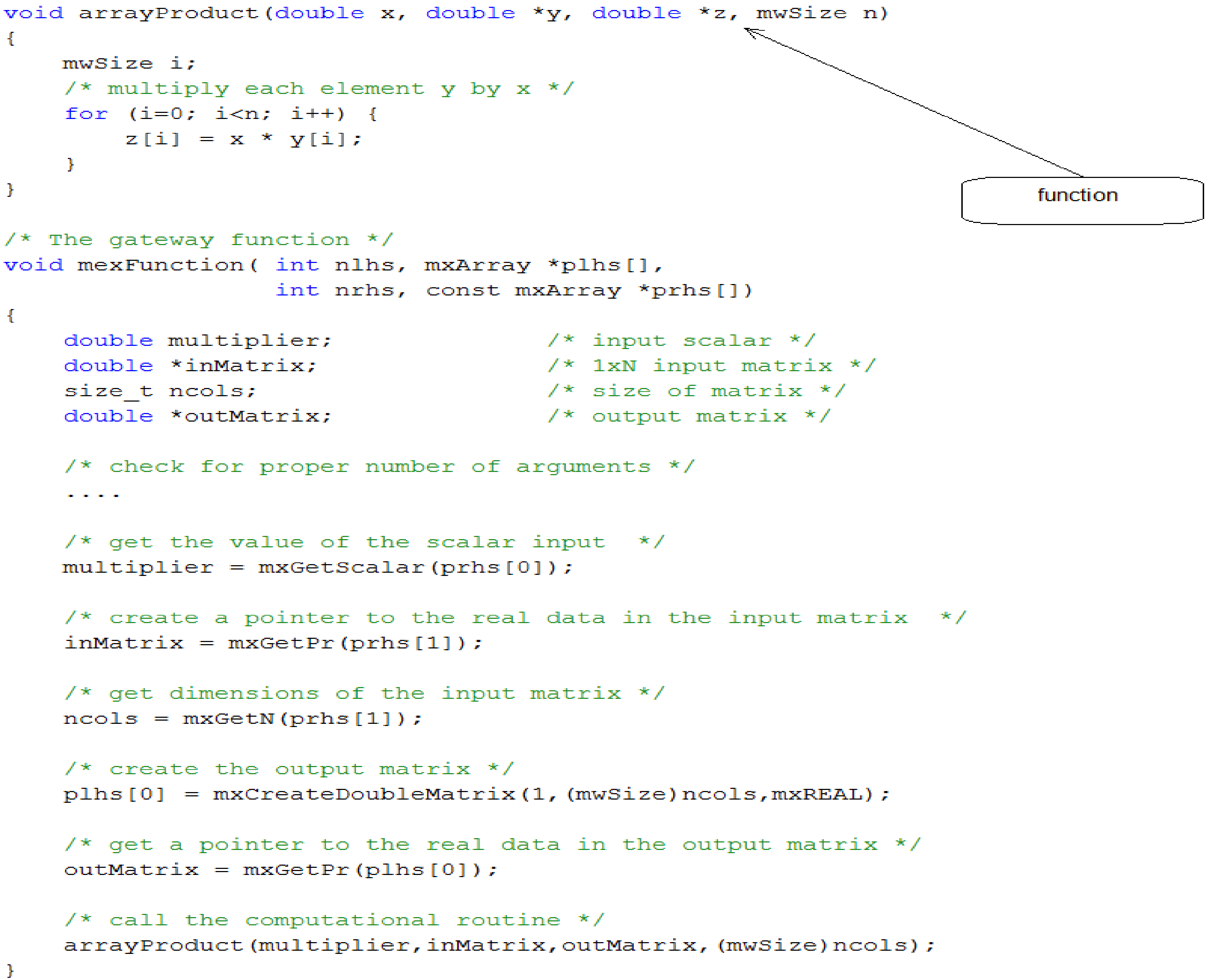
- **Compiler Design** – distinct course in Computer Science and Computer Engineering.
- Bytecode (JVM - Java Virtual Machine, modern Java compilers) and P-Code (Pascal)
- P-code (Portable Code Machine), a virtual machine designed to execute p-code (the assembly language of a hypothetical CPU).
- **Programming Environments** – Compilers and interpreters do not exist as isolated entities. Other tools assist the programmers (Assemblers, debuggers, preprocessors, linkers) and IDE (Interface Developing Environment)
- Open source IDE (Eclipse and NetBeans) and proprietary IDE for compilers (Visual Studio, IAR Embedded Workbench, IAR visualSTATE – event driven state machine)
- However, compiler can be used along with a collection of command-line tools but it is a very hard task.

Programming Language Syntax

- Different from natural languages, the programming language must be precise.
- Both their form (syntax) and meaning (semantics) must be specified without ambiguity
- Specifying Syntax: Regular Expressions and Context-Free Grammars
- The structured program theorem (Böhm-Jacopini theorem, 1966) - class of control flow graphs (historically called charts) can compute any computable function if it combines subprograms in only three specific ways (control structures).
  - *Sequence*: Do this; then do that
  - *Selection* (or *choice*): IF such-&-such is true,
    THEN do this
    ELSE do that
  - *Repetition* (or *looping*): WHILE such-&-such is true
    DO this
- Other structures have been added for facility and clarity of programs (e.g. switch in C)

Application of programming languages called by other Applicantions

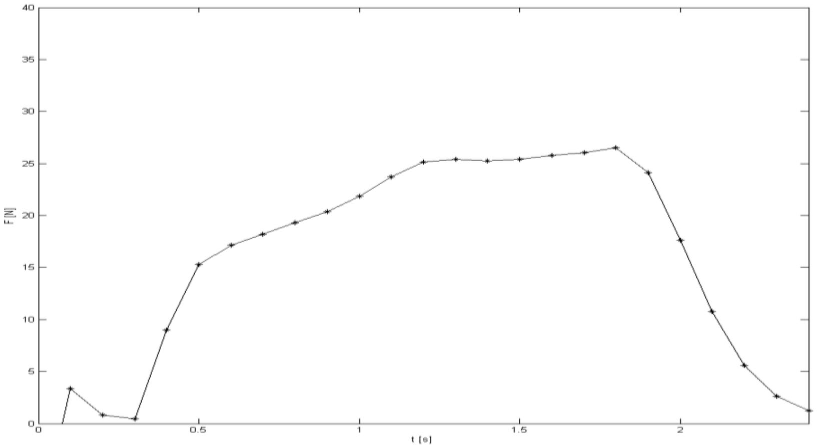
- Calling C++\C functions from Matlab (Fortran functions are also possible)
- Be created with Matlab editor. Compilers suported by Matlab, but Matlab has also a C/C++ compiler.
- The C/C++ Matrix Library API and the C MEX Library API functions.
- The *mex* build script. The result of compile will be a *.dll* file called from *.m* file.
- Functions/subroutine must have a specific template
- Input/outputs are passed via interface
- A snippet code must verify if the number of input variable is correct
- *#include "mex.h“* at start of *arrayProduct.c* file
- *nrhs*, number of inputs
- *nlhs*, number of outputs
- **Code can be magnitude order faster**
- **Encapsulation of proprietary algorithm in called function**

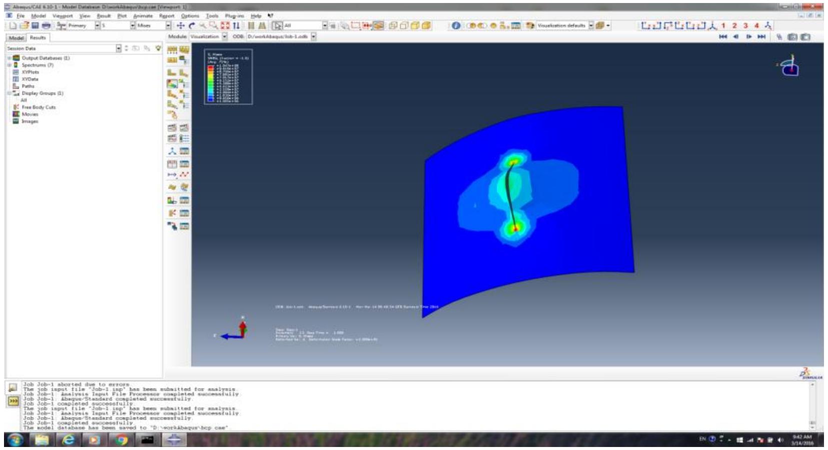


```
void arrayProduct(double x, double *y, double *z, mwSize n)
{
    mwSize i;
    /* multiply each element y by x */
    for (i=0; i<n; i++) {
        z[i] = x * y[i];
    }
}

/* The gateway function */
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
    double multiplier;              /* input scalar */
    double *inMatrix;               /* 1xN input matrix */
    size_t ncols;                   /* size of matrix */
    double *outMatrix;              /* output matrix */

    /* check for proper number of arguments */
    ....

    /* get the value of the scalar input  */
    multiplier = mxGetScalar(prhs[0]);

    /* create a pointer to the real data in the input matrix  */
    inMatrix = mxGetPr(prhs[1]);

    /* get dimensions of the input matrix */
    ncols = mxGetN(prhs[1]);

    /* create the output matrix */
    plhs[0] = mxCreateDoubleMatrix(1,(mwSize)ncols,mxREAL);

    /* get a pointer to the real data in the output matrix */
    outMatrix = mxGetPr(plhs[0]);

    /* call the computational routine */
    arrayProduct(multiplier,inMatrix,outMatrix,(mwSize)ncols);
}
```

function

- CAE (Computer-aided engineering) and Multiphysics tools (Comsol Multiphysics) can use files written in different programming language and interpret them (FEM, finite element method).
- ABAQUS/CAE, use Phyton for scripting and Fortran for subroutines. UEL and UMAT subroutines. UMAT: Define a material’s mechanical behavior, UEL: Define an element. *AMPLITUDE is used for tabular loading[1]





```
Job Job-1 aborted due to errors.
The job input file "Job-1.inp" has been submitted for analysis.
Job Job-1: Analysis Input File Processor completed successfully.
Job Job-1: Abaqus/Standard completed successfully.
Job Job-1 completed successfully.
The job input file "Job-1.inp" has been submitted for analysis.
Job Job-1: Analysis Input File Processor completed successfully.
Job Job-1: Abaqus/Standard completed successfully.
Job Job-1 completed successfully.
The model database has been saved to "D:\workAbaqus\bcp.cae".
```

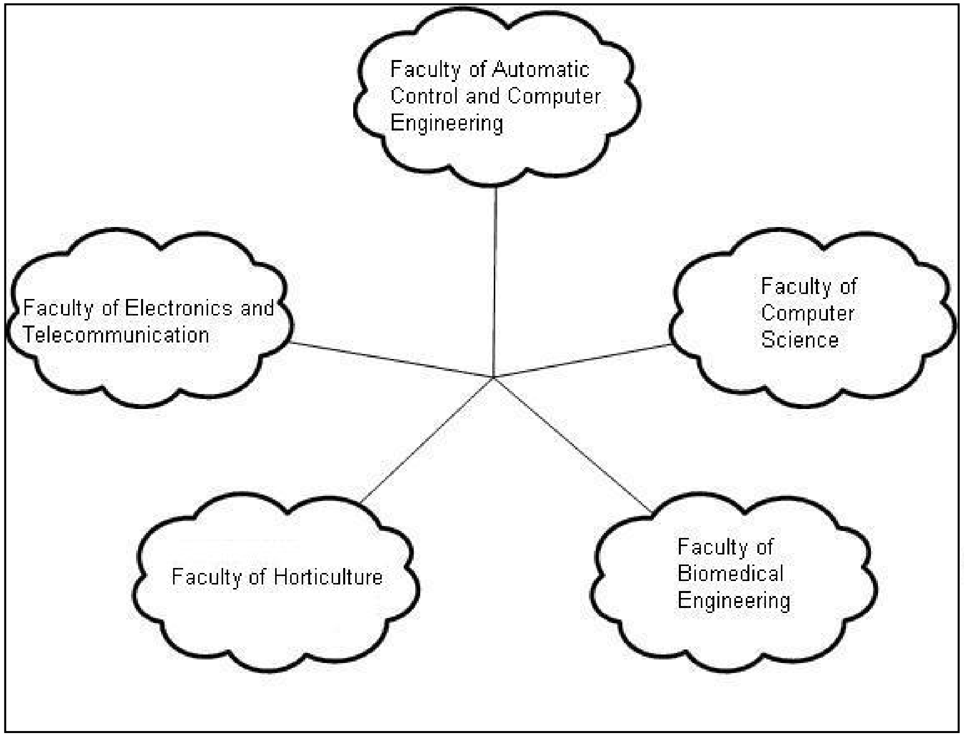
[1] R.Filep, D. Arotaritei, M. Turnea, M. Ilea, M. Rotariu, Crack Development in Prosthetic Skin using Caginalp Phase Field Model, Medical-Surgical Journal, Iasi, Romania, October 2016, paper accepted.

Object Oriented Programming

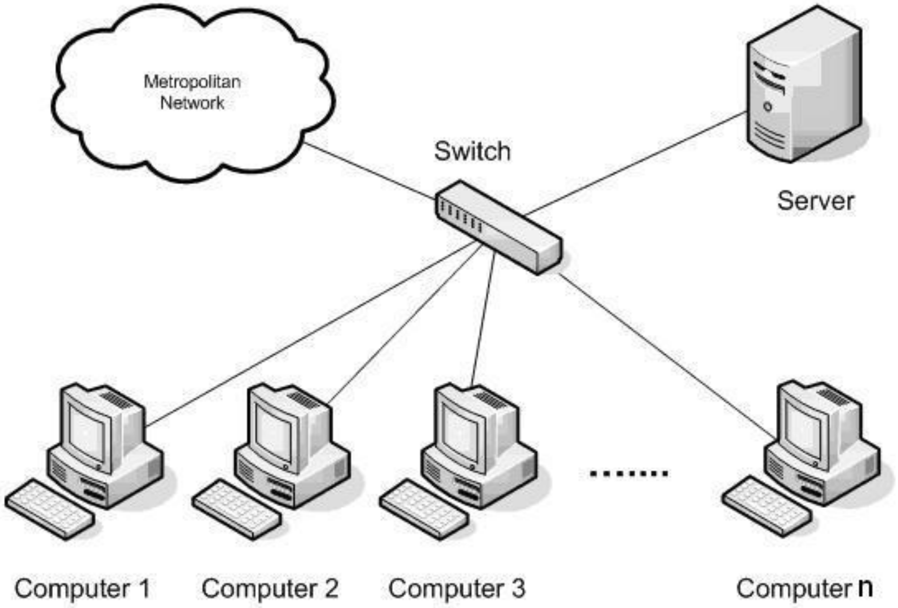
- Object oriented programming (OOP) is programing based on objects.
- E.g. C++, Java, Lisp, Pyton, Smaltalk, C#, Perl, Ruby, and PHP.
- **Classes** - data and procedures (methods) for a given type or class of object
- **Objects** - instances of classes
- Main Features
  - Inheritance - an object or class is based on another object (prototypal inheritance) or class (class-based inheritance), using the same implementation (single, multiple, multilevel); inheritance enables new objects to take on the properties of existing objects
  - Polymorphism - single interface to entities of different types
  - Encapsulation - Encapsulation can be used to hide data members and members function (public, private, protected)
  - Abstraction - Abstraction means working with something we know how to use without knowing how it works internally. It allows us to write code, which works with abstract data structures (like dictionaries, lists, arrays and others).

Grid Computing

- Grid computing is a distributed architecture of large numbers of computers or clusters of computers connected to solve a complex problem.


Faculty of Automatic Control and Computer Engineering
Faculty of Electronics and Telecommunication
Faculty of Computer Science
Faculty of Horticulture
Faculty of Biomedical Engineering

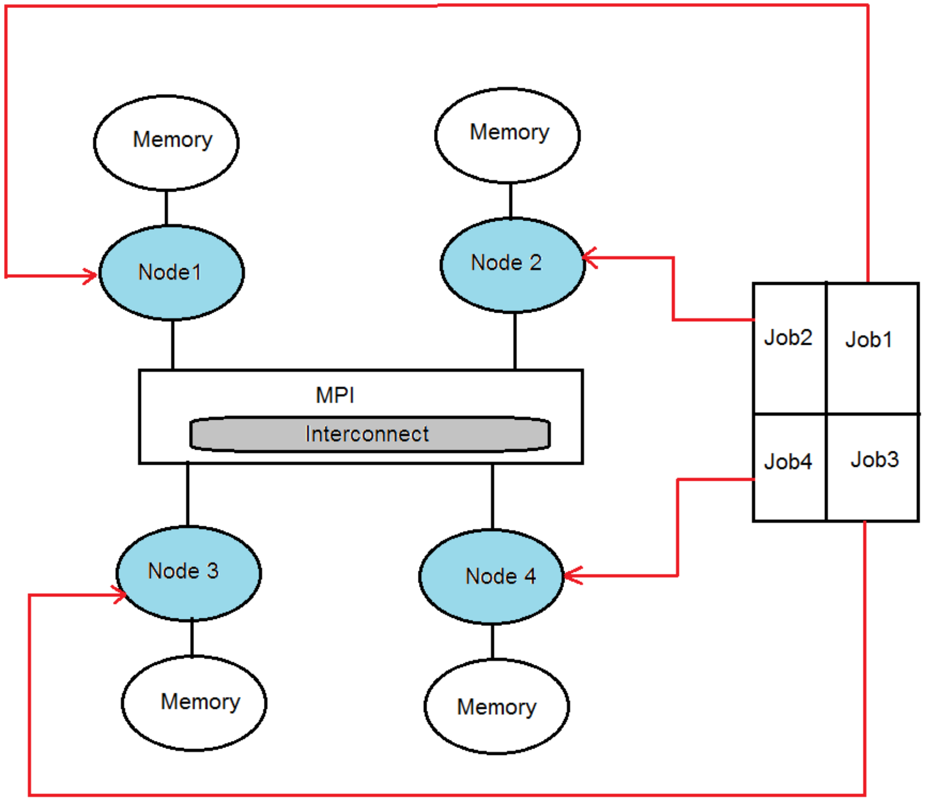
The GRAI network[2]



Node structure[2]

Grid Computing - Programming

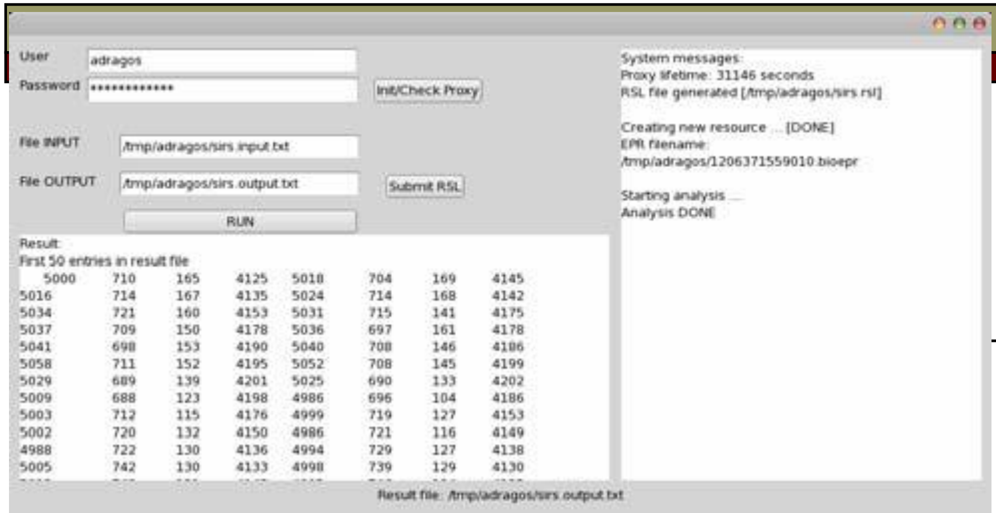
- C and C++ with MPI (Message Passing Interface), STL – Standard Template library



- A good example is presented in a)
- Parallelization of algorithm
- Job allocation
- Collect and assembly the results
- Compile the program
- Submit the jobs to the queue

- Load balancing problem
- Time optimization

a) https://hpcc.usc.edu/support/documentation/examples-of-mpi-programs



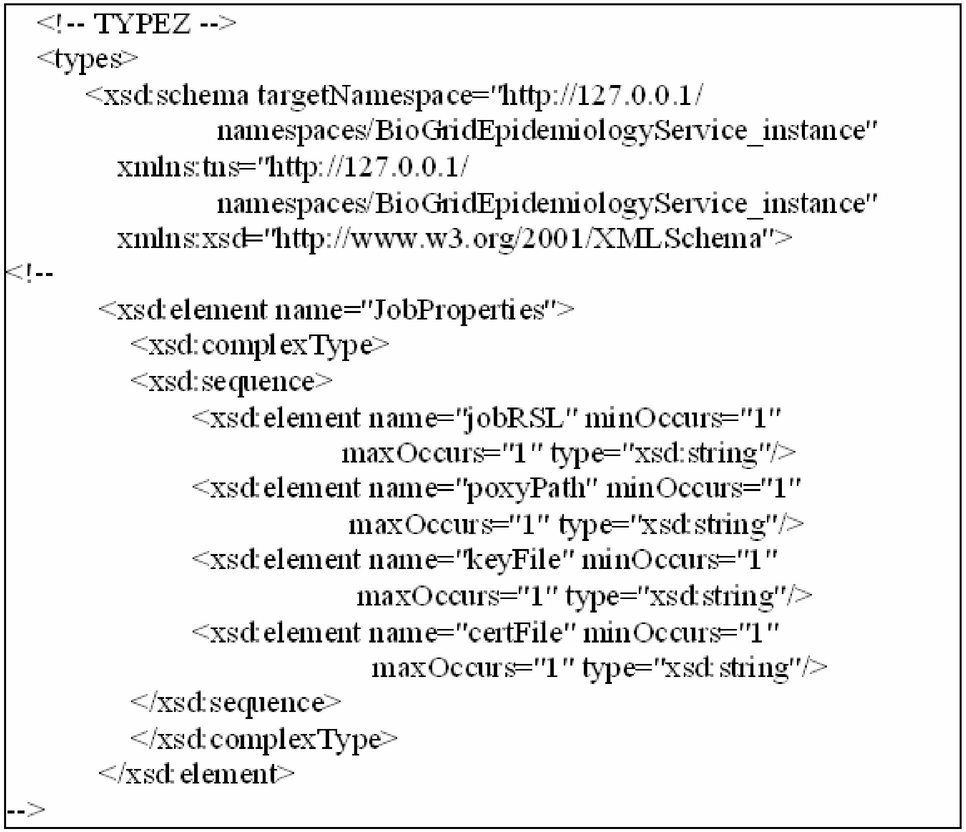
User adragos
Password ************
File INPUT /tmp/adragos/sirs.input.txt
File OUTPUT /tmp/adragos/sirs.output.txt

```
Result:
First 50 entries in result file
    5000      710     165     4125    5018     704     169     4145
5016          714     167     4135    5024     714     168     4142
5034          721     160     4153    5031     715     141     4175
5037          709     150     4178    5036     697     161     4178
5041          698     153     4190    5040     708     146     4186
5058          711     152     4195    5052     708     145     4199
5029          689     139     4201    5025     690     133     4202
5009          688     123     4198    4986     696     104     4186
5003          712     115     4176    4999     719     127     4153
5002          720     132     4150    4986     721     116     4149
4988          722     130     4136    4994     729     127     4138
5005          742     130     4133    4998     739     129     4130
```

```
System messages:
Proxy lifetime: 31146 seconds
RSL file generated [/tmp/adragos/sirs.rsl]

Creating new resource ... [DONE]
EPR filename:
/tmp/adragos/1206371559010.bioepr

Starting analysis ...
Analysis DONE
```

Result file: /tmp/adragos/sirs.output.txt

The GUI for epidimiological service developed in JAVA



```
   <!-- TYPEZ -->
   <types>
        <xsd:schema targetNamespace="http://127.0.0.1/
                      namespaces/BioGridEpidemiologyService_instance"
          xmlns:tns="http://127.0.0.1/
                      namespaces/BioGridEpidemiologyService_instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--
         <xsd:element name="JobProperties">
            <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="jobRSL" minOccurs="1"
                                 maxOccurs="1" type="xsd:string"/>
                  <xsd:element name="poxyPath" minOccurs="1"
                                  maxOccurs="1" type="xsd:string"/>
                  <xsd:element name="keyFile" minOccurs="1"
                                   maxOccurs="1" type="xsd:string"/>
                  <xsd:element name="certFile" minOccurs="1"
                                    maxOccurs="1" type="xsd:string"/>
            </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
-->
```

The scripts for BioGridEpidemiologyService-partial

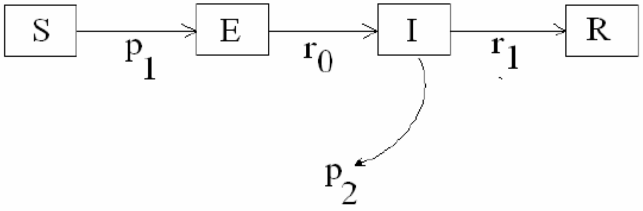


Fig. 1. The four compartmental SEIR model of disease propagation

$$dS/dt = \mu - \beta(t)SI - \mu S \tag{1}$$

$$dE/dt = \beta(t)SI - (\mu + \alpha)E \tag{2}$$

$$dI/dt = \alpha E - (\mu + \gamma)I \tag{3}$$

$$S + E + I + R = N \tag{4}$$

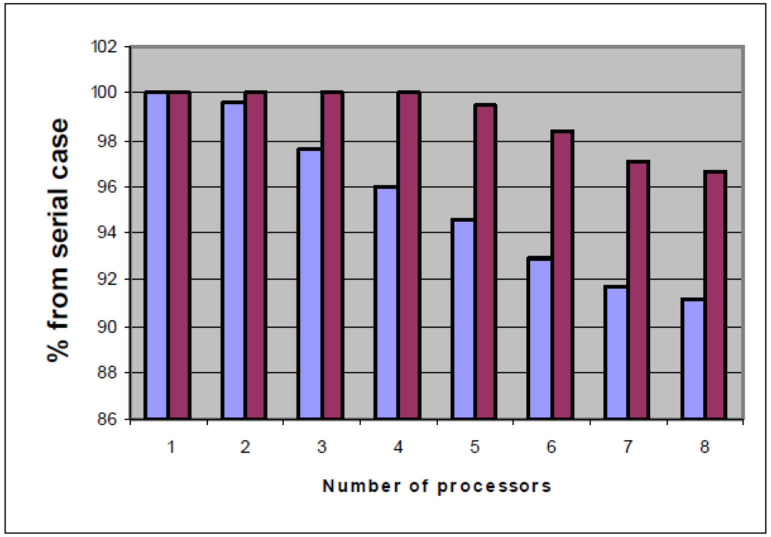
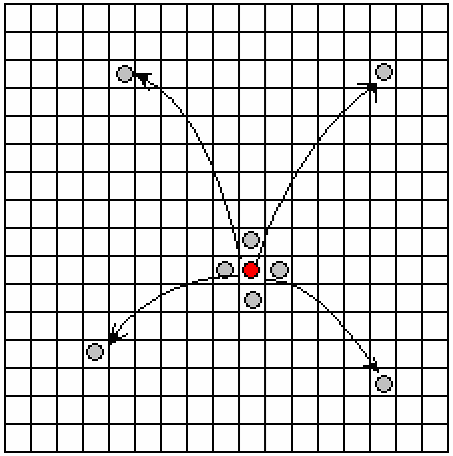
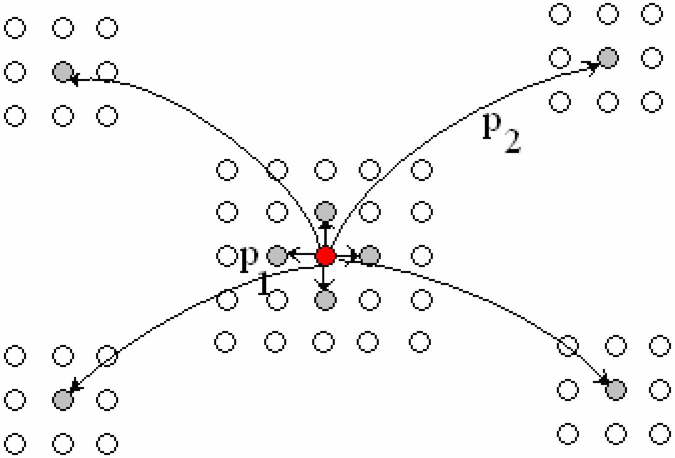


Fig. 8. The result of simulation (the best case and the unfavorable case).



(a)



(b)

Cloud Computing?

- Service models
  - Infrastructure as a service (IaaS)
  - Platform as a service (PaaS)
  - Software as a service (SaaS)
- Free platform: Hadoop, Eucalyptus
- Programming languages: Java, Pyton, Ruby, ECL, etc.
- Applications, e.g. "Hybrid classification engine for cardiac arrhythmia cloud
- service in elderly healthcare management"[3], health care mangement system named CardiaGuard, a cloud service that is an expert system based on hybrid classifier using support vector machine (SVM) and random tree (RT) classification algorithm.
- Preprocess data (filters), HRV (Heart Rate Variability), RR intervals are extracted.

Mobile Programming

- Mobile application development - application software developed for handheld devices, e.g. digital assistants (PDA), enterprise digital assistants (EDA) or mobile phones.
- Operating systems: Android, IOS, Windows 10, Ubuntu, Tizen OS.
- The main programming language for Android is Java.
- Eclipse or Android Studio? (XAMARIN – cross platform, C# language)
- Android Emulator
- Javascript+jQuery (client-side scripting of HTML)
- Mobile database – security of SQL transaction.
- Java, JAVA IDE, Swing

Programming languages for embedded systems

- C (variants) with some extensions. Pointers are recommended to be 2 level maximum (pointer to pointer, or pointer to array of pointers)
- Different from ANSI C, byte and boolean type for some implentations.
- Custom C for Development Board, e.g. (EasyPic v7, Mikro Ccompiler)
- Biomedical applications, MSP430xxx, Code Composer Studio, wireless applications
- Limited options: C, C++, Java, Basic.
- Communication protocol is transparent in most cases for wireless applications

Conclusions

- Assembly languages have actually a small usage. The main applications are that need high speed: drivers (e.g. printers), libraries (A/D conversion).
- C Language is wide spread and despite of numerous challenges it is still very used language
- Some languages that start with great expectation proved to be the a niche one (Prolog, Haskell).
- Writing solid code with comments and test of validity of parameters is good practice
- Using try and catch in release version can help you in future versions of your programs
- Some programming languages and associated methods can have their own philosophy (a thinking mode), e.g. C language and Visual C++ with MFC
- There are other types of languages that are not discussed there: languages for artificial intelligence, functional programming languages, programming in hypercube multiprocessors, etc.

- Microprocessors and Computers have influence on developing or success of some programming languages (or new programming languages, e.g. transputers and Occam).
- Operating system is important in choose of programming language for your application.
- Computer Programming theory (and Compiler Design) can be an accelerator to learn a programming language?
- It is hard to cover all the domains from computer languages. Other topics are not discussed, e.g. Programming Wireless Sensor Networks[4].

References

- Robert W. Sebesta, Concepts of Programming Languages (11th Edition), Pearson, 2015.
- Michael L. Scott, Programming Language Pragmatics, Fourth Edition, Morgan Kaufmann, 2009.
- Maurizio Gabbrielli and Simone Martini, Programming Languages: Principles and Paradigms, Springer, 2010.
- Donald E. Knuth, The Art of Computer Programming, Vol I-IV, Addison-Wesley Professional, 2011
- Blaise Barney, Introduction to Parallel Computing, Lawrence Livermore National Laboratory.
- Michael Quinn, Parallel Programming in C with MPI and OpenMP, McGraw-Hill Science/Engineering/Math, 2003.
- De Joshy Joseph,Craig Fellenstein, Grid Computing, Prentice Hall, 2004.
- Borko Furht, Armando Escalante, Handbook of Cloud Computing, Springer, 2010
- [2]Craus, M. Teodorescu, H.N., Croitoru, C., Brudaru, O., Arotaritei, D., Calin, M., Archip, A. "Academic Grid for Complex Applications – GRAI" , CSCS16, 16th International Conference on Control Systems and Computer Science, mai 22-25, 2007, POLITEHNICA University of Bucharest.
- [3]Huan Chena, Bo-Chao Cheng, Guo-Tan Liao Ting-Chun Kuo, Hybrid classification engine for cardiac arrhythmia cloud service in elderly healthcare management, Journal of Visual Languages & Computing, Volume 25, Issue 6, December 2014, Pages 745–753.
- Luca Mottola, Gian Pietro Picco, Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art (https://core.ac.uk/download/files/362/11435222.pdf)

Thank you!