

#1 Tutorial - Existing languages

Working in pairs

During your time at University and in work placement you have seen several programming languages.

The tutorial this week is to compare 2 of these languages. I would suggest that you compare VB or Ada 95 with either C++ or Java.

The first task is to identify criteria for comparing the languages. To help you here is a list of some of the point that you may wish to compare.

- Program Structure
- Data types
- Libraries
- Parameter passing, subroutine structure
- Statement types
- Error handling
- Readability issues

You should look at Similarities and differences, were possible identifying the common factors/ principles etc.

The result of your work should be a table of criteria, with several columns containing your findings. In some cases were the languages implement similar concepts score the implementation for a specific language with a mark out of 5. You will need to be able to justify your score. Using these marks, which language is the best for a particular application area.

#2 Tutorial - Existing languages

Working in pairs

With your thoughts from the tutorial in week 1, produce a score for the suitability of the languages that you considered for the following application areas.

1. Writing a program to access and display results from a database. There will be some minor processing of the data extracted before it is presented to the user. This processing will mainly involve string manipulation.
2. A compiler for a simple block structured language. This will involve processing of tokens and the use of linked data structures to hold the intermediate state of the compilation.
3. An application to control the rudder movement of a new commercial airline. The data for the application is provided in real time from various sensors.
4. A web based application to implement a chat room. The application will be required to use low level communications protocols such as tcp/ip.
5. A payroll program.
6. An application area of your own choosing
7. An application area of your own choosing

Hint:

Select the criteria that you think are important for each application area. You may wish to add to the criteria that you thought of in tutorial #1.

Score each of these criteria (out of 5) on their suitability as implemented in your chosen language.

Total up the score for each language to give a figure of merit for that language in that application area.

#3 Tutorial - Language influence

Working in pairs

Using your knowledge of programming languages (e.g. Ada, Java, C++, VB,...) identify:

- Machine architecture features that have been propagated into the languages.
- Features that are architectural neutral
- Features that can be implemented efficiently using a particular architecture
- Architectural features that are not represented.

You may wish to consider the following architecture's

- A multi register architecture
In this there are many CPU registers that are used to hold intermediate results.
- A stack based architecture
Intermediate results are pushed onto the stack, there are no registers
- A stack based architecture with CPU registers
- The new I64 architecture that contains 3 instructions per word.
- A RISC architecture (Few instructions but execute very fast)

Before you start this comparison you may wish to populate the following table

Criteria	Ada	C++	Java	VB
Available on more than 1 environment/ OS / Architecture				
Data representation defined exactly				
Library semantics defined precisely				

#4 Tutorial - Errors

Working in pairs

What is the difference between a syntax and a semantic error.

Are some languages better at detecting errors than others, think of an example.

Do the languages you know have an international standard.

Using your knowledge of programming languages (e.g. Ada, Java, C++, VB,...) identify:

- 3 Syntax errors that you consistently make
- 3 Run-time errors that occur frequently in your program
- 3 Errors that the compiler detects for you at compile time that if left un-detected would lead to errors in the running of the program.
- 3 Errors that the compiler does not detect, but cause problems later on.

#5 Tutorial - Data structures

Working in pairs

Complete the following table

You may wish to give a figure of merit for the ease or otherwise of using this feature in the languages shown below.

You may also add other language know to you to this table.

Feature	Ada	C++	Java	VB	other1
Ability to hold data in an Array					
Ability to hold data in a List					
Ability to hold data in a Container (Containers part of language specification)					
Ability to hold and manipulate persistent data.					
Ability of the language to interface to other systems					
Ability to create a new data structure (e.g. a description of a person)					
Range of data structures supported in language					
Ease of creating new data structures					

Are some languages better at holding and manipulating data:

- Give an example of a language which is good at holding and allowing the programmer to manipulate the held data. Justify your answer.
- Give an example of a language which is bad at holding and allowing the programmer to manipulate the held data. Justify your answer.

What are the advantages and disadvantages of holding data in linked storage in languages that are known to you.

Has the class construct effectively replaced the need for a programmer to create data structure in many OO programming languages.

#6 Tutorial - Complexity

Working in pairs

How have languages designers helped a programmer control complexity in a program. For each feature that you identify give a score for effectiveness at controlling complexity in the languages shown. Do not worry if you are unfamiliar with any of the languages you can always replace them with others.

Feature	Usefulness of feature in controlling complexity (1..10)				
	Ada	C++	Java	VB	other1
Subroutine					

List in order of successfulness in controlling complexity the different features found in programming languages.

#7 Tutorial - Complexity

Working in pairs

- Identify 3 constructs/ concepts/ idioms in programming languages known to you that cause complexity in a program.

For example, in the 1950's and 1960's you might have identified the uncontrolled use of the goto statement as the top contender for introducing complexity into a program.

- What programming guidelines would you introduce to control the complexity introduced by the items that you have selected. For example, in the case of the goto statement you might have suggest that goto's are not allowed to:
 - Exit a function/ procedure in transferring control to their destination
 - Cross iteration loops that have been created in the program

Simply saying do not use the construct may leave the programmer unable to write a solution. In the 1950's and 1960's languages did not have the rich control structure that is available in today's languages.

- Identify the top 3 constructs/ concepts/ idioms in programming languages known to you that reduce complexity in a program.

Justify you choice of constructs