For adults only

http://lem12.uksw.edu.pl/loglan82

repository maintained by: A. Salwicki & A. Zadrożny

Loglan'82

more than a language for object and distributed programming

Loglan'82 is an object and distributed programming language. It has many properties that make it a tool better than others.

- It has a unique, cheap and **safe** system of managing the objects.
- It offers modules of classes, coroutines and processeses. You can create objects of coroutines and objects of processes beside the objects of classes.
- Virtual Loglan processors may connect (through network) thus creating a virtual, multiprocessor Loglan computer.
- Objects of processes can be allocated on distinct nodes of network of connected virtual machines or on the same computer. It means that Loglan'82 has one model of concurrent and distributed computations (less learning!).
- Loglan'82 offfers its own original and fully object protocol of communication between objects of processes. It is *alien call* of methods.
- Each object of process may create and manage its own system of coroutine's objects.

What is offered?

- The report of the language Loglan'82, manuals, guides of installation, **compilers** for Linux and Windows platforms, set of examples with proofs.
- Publications and manuscripts containing the solutions of problems.

Who may profit as an eventual user of Loglan?

We are hoping that know-how contained on these pages will turn out to be valuable for:

- **Ambitious programmers** – for Loglan'82 offers a couple of constructions and inventions not known in other programming languages:
    - If you wish to manage the memory of objects (heap) and to avoid the dangerous phenomemnon of *dangling references* choose the instruction **kill**().
    - If software you are going to build is to serve concurrent or distributed computations or a combination of them choose Loglan'82.
    - Take the time to learn the protocol of *alien call.* This original invention is known only in Loglan'82.
    - If you are planning to program quasi-concurrent computations then choose the **coroutines**.

- **Teachers** -- Loglan'82 is a good choice:
    - if you aim to present all the tools and concepts of object programming, without passing from one to another language during the course,
    - if you wish to demonstrate larger software projects that start with specification of data structures and algorithms, implementation i.e. The programming, and verification of correctness of modules with respect to their specifications.

- **Researchers** – For Loglan'82 appeared in the process of searching answers to several scientific questions and problems. The solutions form the *intellectual contribution* to Loglan'82 project.

- ✔ **P1**: Is it possible to deallocate unused objects in a safe and efficient way? - safe it means free of dangling reference errors and cheap in implementation.
- ✔ **P2**: How to determine the direct superclass? Note, what the compiler and the reader of a program know is the name of a class. However, in Loglan'82, Java, BETA and Simula67 languages there may coexist many classes of a given name. For these languages allow nesting of classes beside the inheritance.
- ✔ **P3**: Where to find a declaration of an identifier *id*, proper for an applicative occurrence of it? In other words – how to compute the relation of static binding of identifiers?
- ✔ **P4**: The programming languages Algol'60 and Pascal apply, with success, the mechanism of Display Vector invented by E.W.D. Dijkstra. Is it possible to apply it as well in Loglan'82 (and nowadays in Java)?
- ✔ **P5**: How to manage the objects of coroutine modules in a way free of inconsistencies known in earlier programming languages?
- ✔ **P6**: Loglan'82 has modules of processes and objects of processes. How to create these objects?
- ✔ **P7**: Are there object tools for communication and synchronisation of objects of process?
- ✔ **P8**: How to manage the distributed computations?
- ✔ **P9**: Is there a mathematical model of parallel computations? Is it equivalent to the interleaving model of concurrent computations?

The solutions of the above listed problems form the foundations of Loglan'82. Some of results are of importance for analysing Java programs. Many others can be useful in various languages.

How much it costs? Can I earn on it?

The compilers, manuals etc are offered on open source licence.

Let us estimate how much you can earn on using Loglan'82 or its inventions.
If the most popular languages C++ and Java adopted the object management system invented by the late professor Antoni Kreczmar (1945 – 1996), the profits probably would reach tens of milions of euro/year.

Look and compare.

Heap system	**Model D** (e.g. Loglan'82)	**Model A** (e.g. C++, Pascal)	**Model B** (e.g. Java 1995, Python)
Pre-	Certain object *o* is referenced by the *n* variables $x_1 = x_2 = ... = x_n$, $1 \le i \le n$.		
Code	kill(x_i)	delete(x_i); x_i = null	x_1 = null; x_2 = null;... x_n = null; Now, the instruction gc() - the object *o* will be deleted.
Post-	All the variables took the value none. Object *o* is deleted.	Object *o* has been deleted. The variable x_i has the value null. Other variables point to the deleted frame.	Object *o* has been deleted – *under condition that **all** the strong* (normal) references to the object have been earlier assigned the null value.
Cost	O(1)	O(1)	O(n + m) m is the global size of the heap of objects.
Risk	No risk(!) Each attempt to read and/or write from the deleted object, will raise an error signal `reference to none`.	If n>1 then **dangling reference error** occurs. High probability of the error of contradicting information and/or destruction error.	Risk of **memory leakage error**, if programmer will forget to nullify some reference to the object *o*. It will remain not deleted.

This concerns solely the problem P1. Learn on other problems and solutions of them.

Join us! mailto:a.salwicki@uksw.edu.pl
You can help in many tasks, from small ones, of editorial character to the autonomous work on difficult problems.