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 jest językiem programowania obiektowego i rozproszonego, ma wiele cech, które czynią z niego narzędzie programowania lepsze od innych:

- Ma unikalny, tani i bezpieczny system zarządzania obiektami.
- Oprócz modułów klas (**class**) oferuje moduły współprogramów (**coroutine**) i procesów (**process**). Możesz więc tworzyc nie tylko obiekty klas, ale także *obiekty współprogramów* i *obiekty procesów*.
- Loglanowskie maszyny wirtualne mogą się łączyć (przez sieć) w wirtualny, wieloprocesorowy komputer loglanowski by wspólnie wykonywać program(y).
- Obiekty procesów mogą być alokowane na różnych węzłach sieci połączonych maszyn wirtualnych, bądź na tym samym komputerze. Oznacza to, że Loglan ma *jeden model* obliczeń współbieżnych i rozproszonych (mniej uczenia się).
- Loglan'82 oferuje własny, oryginalny i w pełni obiektowy protokół komunikacji pomiędzy obiektami wątków tzw. obce wołanie metod (*ang.* alien call).
- Każdy wątek może stworzyć swój system współprogramów i zarządzać nim.

Kto może skorzystać na Loglanie?

Mamy nadzieję, że informacje zawarte na tych stronach okażą się interesujące i przydatne dla:

- **Ambitnych programistów** -- język Loglan'82 oferuje garść konstrukcji i rozwiązań nieznanych w innych językach programowania :
    - Jeśli chcesz świadomie zarządzać pamięcią obiektów i uniknąć przy tym niebezpiecznego zjawiska *wiszacych referencji* to spróbuj instrukcji **kill**( ).
    - Jeśli oprogramowanie, które tworzysz ma służyć obliczeniom współbieżnym, lub rozproszonym lub jakiejś ich kombinacji to wybierz Loglan'82 z jego jednym modelem obejmującym wszystkie te rozdaje obliczeń, zredukuje to znacznie koszty opanowania programowania współbieżnego i/lub rozproszonego.
    - Na pewno warto zapoznać sie z *protokołem obcego wołania* metod wątku A podczas wykonywania instrukcji wątku B, jest to oryginalny wynalazek znany jak dotąd tylko w Loglanie.
    - Jeśli masz zamiar zaprogramować obliczenia quasi-współbieżne to zapoznaj się z współprogramami (**coroutines**).

- **Nauczycieli** -- Loglan'82 jest dobrym wyborem:
    - jeśli chcesz przedstawić wszystkie narzędzia i metody programowania obiektowego unikając przy tym przechodzenia od jednego do kolejnego języka programowania dla zilustrowania kolejnych narzędzi programowania obiektowego.
    - jeśli chcesz to możesz programować w Loglanie w ramach większych projektów programistycznych obejmujących specyfikowanie struktur danych i algorytmów, implementację specyfikacji czyli programowanie i weryfikację poprawności modułów oprogramowania względem specyfikacji.

- **Researchers ─** For Loglan'82 appeared in the process of searching solutions of several problems. . These answers form the intellectual base of 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 supeclass? Note, what the compiler and the reader of a program know is the name of a class. However, in Loglan'82, Java, BETA and Simla67 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 difference between concurrent and parallel computations?

Wyniki uzyskane podczas prac nad Loglanem okazały się przydatne w analizie języka Java. Mogą znaleźć zastosowanie w innych językach.

Co możesz pobrać i użytkować?

- Raport języka, podręczniki, instrukcje użytkownika, kompilatory na platformy Linux i Windows, zestaw przykładowych programów.
- Raporty zawierające rozwiązania wielu problemów wymienionych powyżej, czyli wcześniejsze wersje publikacji.

Ile to kosztuje? Ile na tym można zarobić?

Wszystko udostępniamy na licencji otwartego oprogramowania i otwartej nauki. Nie płacisz nic.

Spróbujmy oszacować zyski jakie można uzyskać:
Gdyby najczęściej stosowane języki programowania Java i C++ zaadaptowały system zarządzania obiektami wymyślony przez prof. Antoniego Kreczmara(1945 – 1996), to zyski możnaby liczyć w dziesiątkach milionów euro/rok.

Zechciej porównać.

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.

To jest tylko odpowiedź na pytanie P1. Należy do tego dodać zyski, jakie można osiągnąć stosując inne wynalazki oferowane przez twórców Loglanu.

Dołącz do nas!

mailto:a.salwicki@uksw.edu.pl

Możesz pomóc w wielu pracach od drobnych, porządkowych do samodzielnego rozwiązywania problemów.