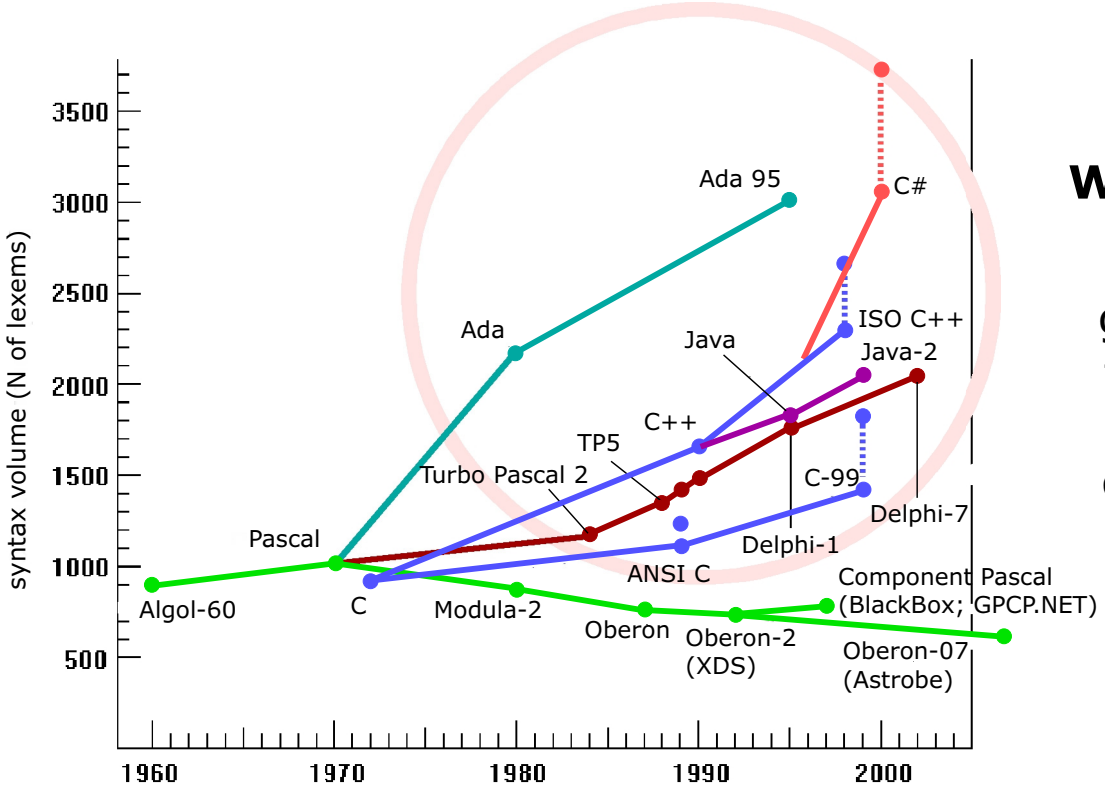
Рациональность и мракобесие в IT

Less is more. Why Oberon beats mainstream in complex applications

Fyodor Tkachov

Institute for Nuclear Research, Russian Academy of Sciences

1. Мифология сферы IT, причины ее возникновения, распространения и живучести.
2. Рациональное ядро технологий программирования (тьюринговские лауреаты Дейкстра, Хоор, Вирт и Гуткнехт) и представленное языками и системами семейства Оберон.
3. Почти 20-летний опыт работы на Оберон-технологиях.



Wirth's Law:

Software gets slower faster than hardware gets faster.

IT-industry's bubble of **excessive complexity**
VS **the rational core (Oberon)**

Original graph in Russian (c) S.Z.Sverdlov "Programming languages and translation methods", Piter Press, 2007

Alchemy came before rational Chemistry.

Astrology came before rational Astronomy.

IT is no exception.

We are currently at a parascientific stage of IT.

Choosing C++
was a gross failure of HEP community
as a *scientific* community.

C++ is best proof of natural origin of human intellect.
The choice of C++ is best proof of irrational forces within scientific community: **combinatorial intellect rules**.

Two causes of the excessive IT-complexity bubble:

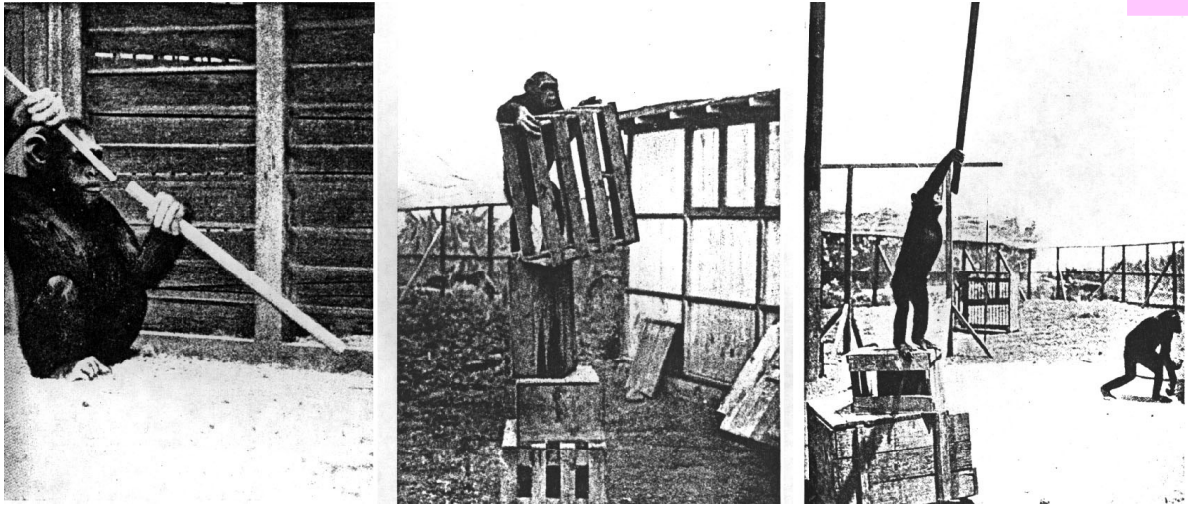
1. **Combinatorial nature** of normal human intellect.

primatologist W.Koeler 1919; zoopsychologists; cognitive sciences

see a banana; scan the scene; identify familiar objects; find banana-getting combination of actions;

if none, get angry and run around, this brings new objects into the scope of attention, with luck ... etc.
Combinatory intellect (99.9% of all human activity**)**

abstraction:
magic wand
cf. the belief that adding features to PL = "progress"



2. **Economic +** social **rent** derived by IT "experts" from the complexity.

Well-known concept of **asymmetry of information**
(e.g. George Akerlof, Nobel Prize for Economics, 2001
for his analysis of "the market for lemons").

Not a plot by IT, but a "collective effect"
in the absence of proper education system:
"normalized" group opinion/myth
(see social psychology)

Эти богатые программисты

Яковенко Дмитрий
«Эксперт» N27 (858) 08 июл 2013

Наблюдающийся рост зарплат в целом
пока не критичен для экономики, к тому же
скоро он должен замедлиться.
Но в некоторых отраслях
зарплаты действительно растут
неоправданно сильно

2. Economic **+ social rent**
   derived by IT "experts" from the complexity.

Рациональное ядро (разобраться с основаниями) 1/3

адресуемая память, регистры, примитивные команды:
load, store, add, jump ... ~машина Тьюринга

Логический уровень (Э.В.Дейкстра, А.Ч.Хоор, ~1970)

Дейкстра: диплом по квантовой теории

императивная программа = **преобразователь предикатов**

начальное состояние → конечное
предусловие → постусловие

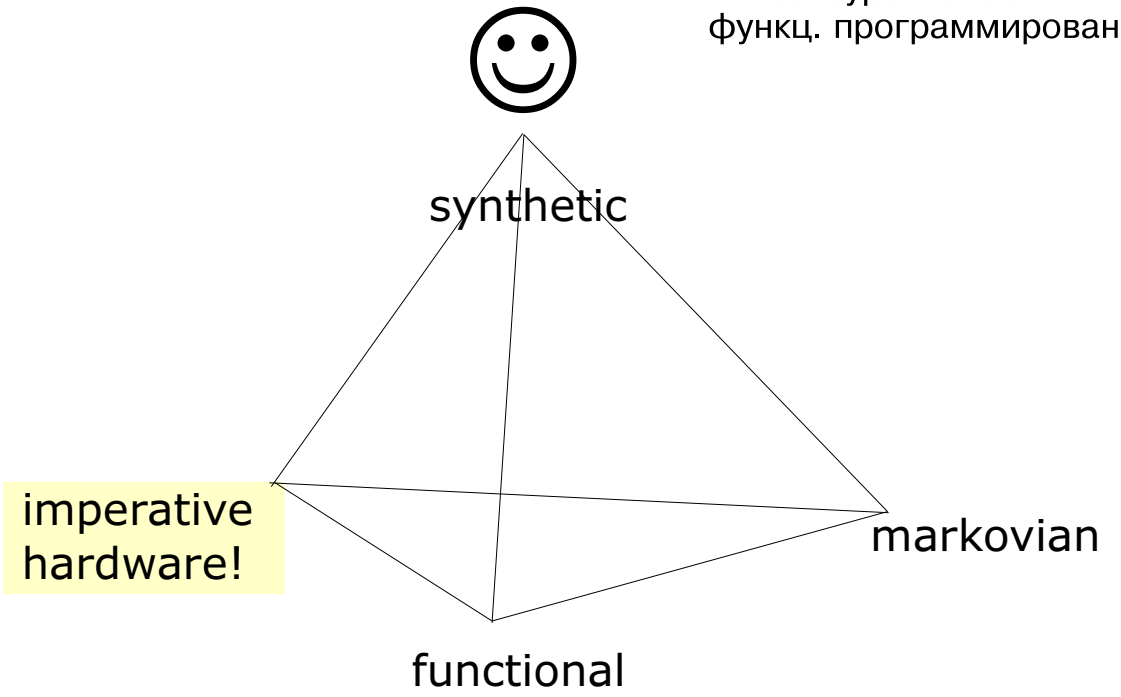
не менее "математично", чем функциональное программирование

систематический **вывод** программ из требуемых постусловий
(ср. интегральное исчисление)
Э.Дейкстра "Дисциплина программирования" (русс.: 1978)

многоветоный IF
многоветочный WHILE
+
процедуры + два вида параметров
типизация переменных

Why imperative

по поводу мифов об особой "математичности" и "высокоуровневости" функц. программирования



farther from hardware # closer to human

Рациональное ядро (разобраться с основаниями) 2/3

...

Модули: независимая разработка + загрузка/выгрузка .dll, .so
divide et impera
инкапсуляция/защита (**по умолчанию** всё спрятано)
контроль типов переменных через границы модулей

полноценных модулей нет практически нигде кроме модулы+оберонов **(???)**

Динамическое управление памятью:
сбор мусора в условиях независимо пишомых модулей
расширение типов
напр.: списки -- один модуль - сервис, другой - наполнение
это и есть **фунд. основа "объектно-ориентир. прог-я"**

```
Element = POINTER TO RECORD
    next: Element;
    data: REAL
END;
```

Рациональное ядро (разобраться с основаниями) 3/3

канал I/O!

Текст-как-интерфейс

все текстовые редакторы превращаются
в (дефективные) системы программирования

все системы программирования превращаются
в (дефективные) текстовые редакторы

не командная строка (слишком мало)
не формы с кнопками (слишком жёстко)

ТЕКСТ: баланс простоты и информац. мощи
см. пример из спектроскопии

Чистенько всё это реализовать >> Оберон

NB хотеть можно много чего, нужно хотеть главного

Oberon ~ Pascal 2K

beware of myths and wrong associations

history
1970 Pascal
1980 Pascal-80=Modula-2
1986 Pascal-86=Oberon

dialects
– (classical | ETH) Oberon (1986)
– Oberon-2 (1992; XDS)
– Component Pascal (1997; BlackBox, .NET)
– Oberon-07 (2007; Astrobe)

Why Oberon "**technologies**"?

rather than simply "programming language" Oberon?

Never just tools,

but tools **+ techniques**

proper balance is key

If you are not aware of your techniques, then it is chiropractic.

To get the most from Oberon the language:
a development environment + a set of skills
(see below)

What *is* Oberon/Component Pascal

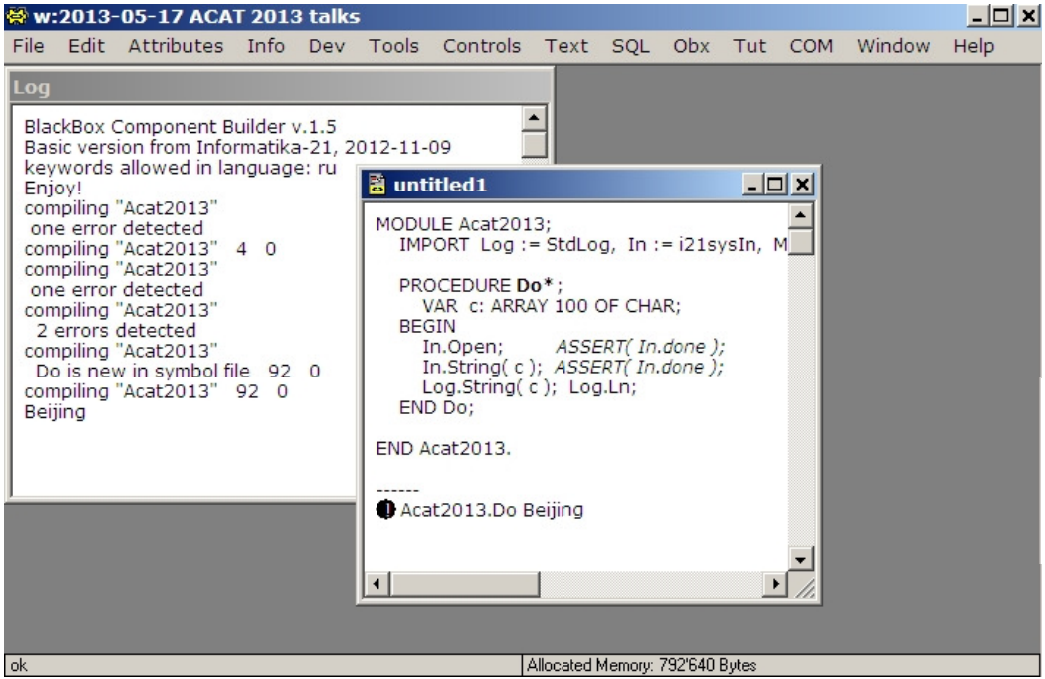
- A "vanishing" imperative programming language
  (anything that can be put away into libraries is excluded from language;
  what remains is designed with utmost care)
- Very small (language report ~20; dialects differ +/- a few pages)
- Pure compiled code (floating point optimization as external tool)
- Highly readable, robust (against typos etc.)
- Statically type safe (including dynamic records > no segviols, **ever**)
- Independently compilable modules (unit of information hiding, dynamic linking and (un)loading > "interactive" feel)
- Object-oriented (extensible records, very efficient)
- Garbage-collected (without affecting purely procedural programs)

Development environment, typically:

- Text-as-interface:
any text document can serve as a command prompt
+ input from any text
= hugely handy: text docs as flexible menus + storage for parameters ...

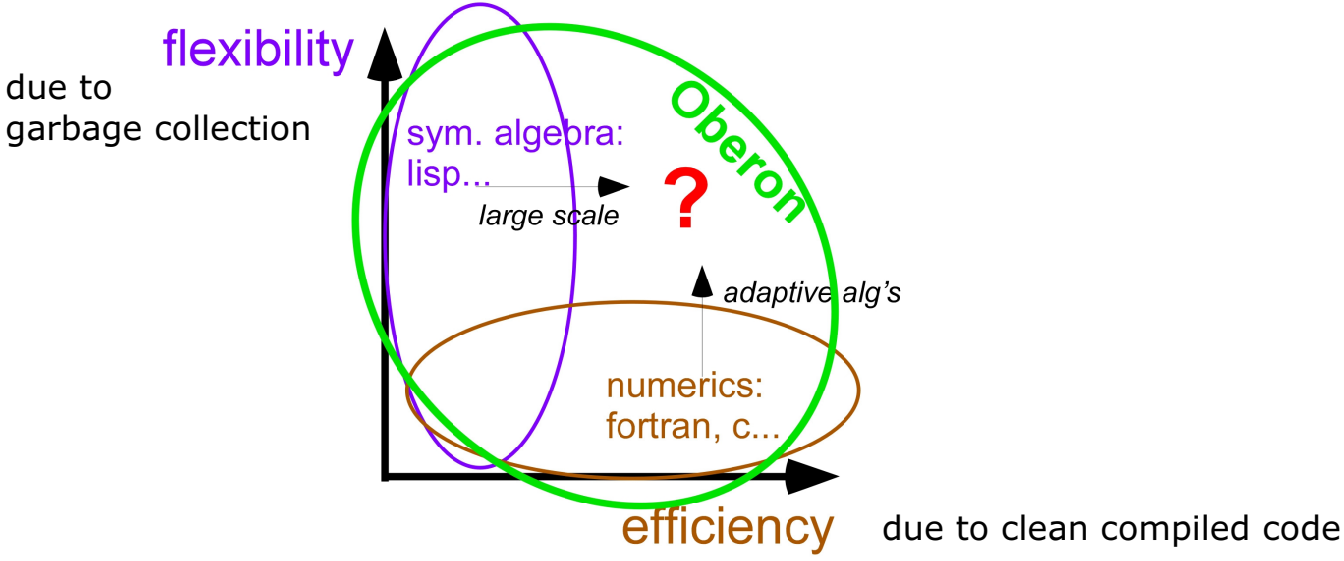
popular general purpose: **BlackBox** Component Builder

A simplistic interface, one becomes fully productive in a week:



92 in the log window means, the compiled code for this module is 92 bytes -- not kilobytes, just bytes.

One Oberon does what is usually achieved via a combination of
- C++ & python
- Mathematica & Fortran
- etc.



non-professional's ego is not attached to IT toys

Oberon experience behind the assertions

BEAR algebra engine since 1997 *single-handedly*

one of the fastest engines, *the* most flexible

array of cutting edge calculations, A.Czarnecki et al.

hep-ph/0511004 Phys. Rev. Lett. (2006)
hep-ph/0506055 Phys. Rev. D
hep-ph/0503039 Phys. Rev. D
hep-ph/0403221 Phys. Rev. Lett. (2004)

Component framework implementing *single-handedly*
quasi-optimal weights (10K l.o.c. with all libraries)

used to reanalyse Troitsk-nu-mass data
arXiv:1108.5034 -- best direct neutrino mass bound

Continuous algorithm development work *single-handedly*
(Optimal Jet Finder etc. hep-ph/0301185; physics/0401012)

International educational project "Informatika-21"
coordinates leading experts from academia, aerospace, publishers ...
authorized revision of int'l bestseller "Algorithms and data structures"
by Turing Award winner N.Wirth www.inr.ac.ru/~info21/

The deal:

C++ (1K pp)

Fortran

Java

Form

Mathematica

.....

Oberon (20 pp)

General algorithmics

Dijkstra's loop & invariant

Architecture patterns

-- Carrier-Rider

-- separation of interface from implementation

-- Oberon message bus

...

After one has learnt to program with Oberon,
learning another language = learning its defects.

Oberon and open source

The world is much more complex than is imagined by software tool (libraries etc.) writers.
No library writer can foresee all the uses and applications.
Access to source for adaptation is essential.

Most open is the code that is most accessible.

Simpler language >> more open the source.

Oberon code is more open than e.g. C++ code.

The ill-recognized **problem of software complexity**
(cf. C++/Root/Mathematica ... crashes)

For the first time in the history of Humanity
the combinatorial/primatic intellect
has become freed from the restrictions of
the resistance of materials.

The Kalashnikov Principle:

Excessive complexity = vulnerability

Asymmetry of information >> customers pay.

Containing the gratuitous growth of complexity must be a permanent concern whenever IT is involved.

Oberon influence in the IT industry:

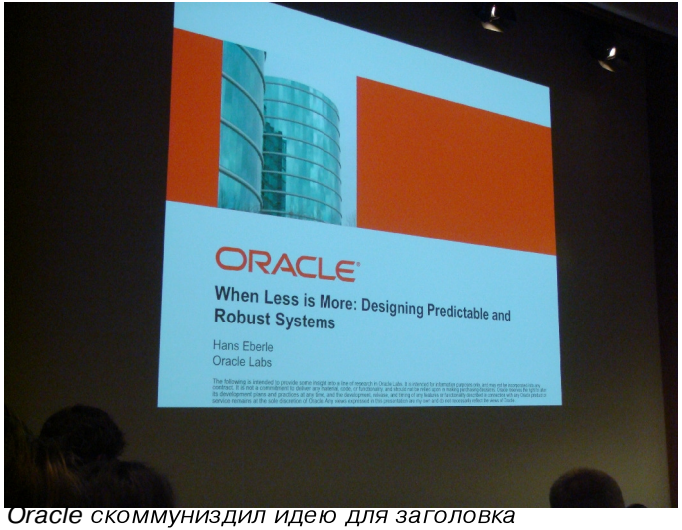
Java emerged after Sun licensed and studied Oberon compiler in 1991; the influence is obvious.

Google's **Go** is a C-syntax clone of Oberon with minor (unnecessary) extensions.

Wirth's student Clemens Szyperski is author of **best selling "Component Software"** and software architect at MS Research working on .NET.

(The books essentially describes the principles that a popular Oberon implementation the BlackBox Component Builder is built on.)

Only physicists are in the dark.



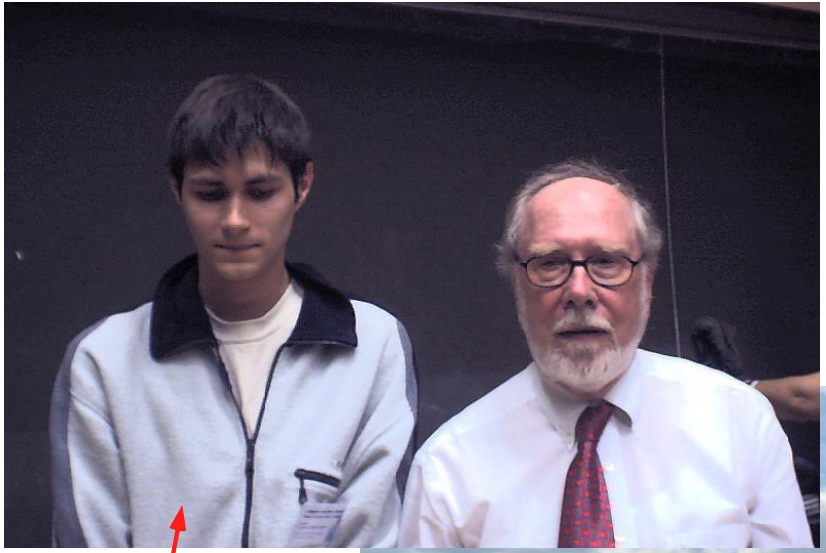
Oracle скоммуниздил идею для заголовка

Clemens Szyperski

Oberon microsystems, Inc. + ФТ
2014-02-20

Symposium Wirth-80, ETH, Zurich, 2014-02-20

Прихотливые пути инноваций



2005: студент физфака МГУ, прослушавший курс на Обероне

2009: он же, внедрил Оберон у Н.Чистякова, конструктора легендарного БПЛА "Пчелка-2" (контртерр.)

References

Prof. Jurg Gutknecht's group at ETH Zurich: http://nativesystems.inf.ethz.ch/

Oberon Day @ CERN 2004: http://www.inr.ac.ru/~blackbox/Oberon.Day/

FT: http://arxiv.org/abs/hep-ph/0202033 (see **testimonies at end**)

BlackBox Component Builder: www.oberon.ch, www.zinnamturm.eu

XDS Oberon (**optimizing**) http://www.excelsior-usa.com/xdsx86.html

Форум российского сообщества: http://forum.oberoncore.ru/

Informatika-21 (educational) http://www.inr.ac.ru/~info21/

N.Wirth "Algorithms and data structures. Version for Oberon"
http://www.inf.ethz.ch/personal/wirth/books/AlgorithmE1/AD2012.pdf

Clemens Szyperski "Component Software: Beyond Object-Oriented Programming"
2nd ed., Addison-Wesley, 2011

Oberon-07 for embedded apps: http://www.astrobe.com/

Gardens Point Component Pascal (.NET etc.): http://gpcp.codeplex.com/

Entire syntax of Component Pascal (BlackBox Oberon)

```
Module    =    MODULE ident ";" [ImportList] DeclSeq
          [BEGIN StatementSeq]
          [CLOSE StatementSeq] END ident ".".
ImportList     =    IMPORT [ident ":="] ident
          {"," [ident ":="] ident} ";".
DeclSeq   =    { CONST {ConstDecl ";" }
          | TYPE {TypeDecl ";"} |
          VAR {VarDecl ";"}}
          {ProcDecl ";" | ForwardDecl ";"}.
ConstDecl      =    IdentDef "=" ConstExpr.
TypeDecl  =    IdentDef "=" Type.
VarDecl   =    IdentList ":" Type.
ProcDecl  =    PROCEDURE [Receiver] IdentDef
          [FormalPars] ["," NEW] [","
          (ABSTRACT | EMPTY | EXTENSIBLE)]
          [";" DeclSeq [BEGIN StatementSeq]
          END ident].
ForwardDecl    =    PROCEDURE "^" [Receiver]
          IdentDef [FormalPars].
FormalPars     =    "(" [FPSection {";" FPSection}] ")"
          [":" Type].
FPSection      =    [VAR | IN | OUT] ident {"," ident} ":" Type.
Receiver  =    "(" [VAR | IN] ident ":" ident ")".
Type      =    Qualident
          | ARRAY [ConstExpr {"," ConstExpr}]
               OF Type
          | [ABSTRACT | EXTENSIBLE | LIMITED]
          RECORD ["("Qualident")"] FieldList
          {";" FieldList} END
          | POINTER TO Type
          | PROCEDURE [FormalPars].
FieldList =    [IdentList ":" Type].
StatementSeq=  Statement {";" Statement}.
Statement      =    [ Designator ":=" Expr
               | Designator ["(" [ExprList] ")"]
               | IF Expr THEN StatementSeq
                    {ELSIF Expr THEN StatementSeq}
                    [ELSE StatementSeq] END
               | CASE Expr OF Case {"|" Case}
                    [ELSE StatementSeq] END
               | WHILE Expr DO StatementSeq END
               | REPEAT StatementSeq UNTIL Expr
               | FOR ident ":=" Expr TO Expr
                 [BY ConstExpr]  DO StatementSeq END
               | LOOP StatementSeq END
               | WITH [ Guard DO StatementSeq ]
                    {"|" [ Guard DO StatementSeq ] }
                    [ELSE StatementSeq] END
               | EXIT | RETURN [Expr] ].
Case      =    [CaseLabels {"," CaseLabels} ":"
          StatementSeq].
CaseLabels     =    ConstExpr [".." ConstExpr].
Guard     =    Qualident ":" Qualident.
ConstExpr      =    Expr.
Expr      =    SimpleExpr [Relation SimpleExpr].
SimpleExpr     =    ["+" | "-"] Term {AddOp Term}.
Term      =    Factor {MulOp Factor}.
Factor    =    Designator | number | character | string \
          | NIL | Set | "(" Expr ")" | " ~ " Factor.
Set  =    "{" [Element {"," Element}] "}".
Element   =    Expr [".." Expr].
Relation  =    "=" | "#" | "<" | "<=" | ">" | ">=" | IN | IS.
AddOp     =    "+" | "-" | OR.
MulOp     =    " * " | "/" | DIV | MOD | "&".
Designator  = Qualident {"." ident | "[" ExprList "]" | " ^ "
          | "(" Qualident ")" | "(" [ExprList] ")"}[ "$" ].
ExprList  =    Expr {"," Expr}.
IdentList =    IdentDef {"," IdentDef}.
Qualident      =    [ident "."] ident.
IdentDef  =    ident [" * " | "-"].
```

A testimony by W.Skulski, exp. physics, USA

"… Someone will ask the following question "why did you choose the BlackBox compiler to develop the Toolbox, while there is a well-known compiler <name>, which everybody knows how to use?" … my answer is as follows: I spent hundreds of hours developing Gr, and BlackBox has not crashed on me even once.
Gr is not entirely trivial, and potential for programming errors is huge. And indeed, I have made many programming mistakes. I dereferenced NIL pointers and I jumped out of array bounds. I unloaded a running code from memory, while the corresponding display was still on screen. I abused the environment in many different ways.
It never crashed.
I never had to worry about leaking memory.
I never saw the words "segmentation violation" or "core dump".
If you can say the same about your compiler <name>, then please tell me its name …
… The reader probably does not fully appreciate the great simplification that this approach is bringing. Full appreciation comes only after the BlackBox environment has been used for some time …"

Миром правит не тайная ложа,
а явная лажа.

Виктор Пелевин