

17, 2023

## what is that

### BCPL

- most programming in BCPL family of languages (B, Bliss, C)
- easy to implement efficiently
- close to the target machine
- pointers are identified with arrays
- and address arithmetic is ubiquitous
- dangerous

### c++

- easy to write unportable code even when disciplined
- normal to mix safe and unsafe code
- C++, has enriched C by adding objects; but it has also given up C's best virtue: simplicity, without relieving C's worst drawback: its low-level programming model.
- on simplicity: macros?

from m3 introduction

- Buffer overruns are not ruled out by design: gets, sscanf etc.
- Interface inconsistencies: gets vs fgets, fgets vs fscanf (note the position of the file stream parameter)
- Bad interfaces like that of getchar() whose return code can be a character or an error code
- Particularly bad buffering system which
- ignores the block structure of underlying file systems, and

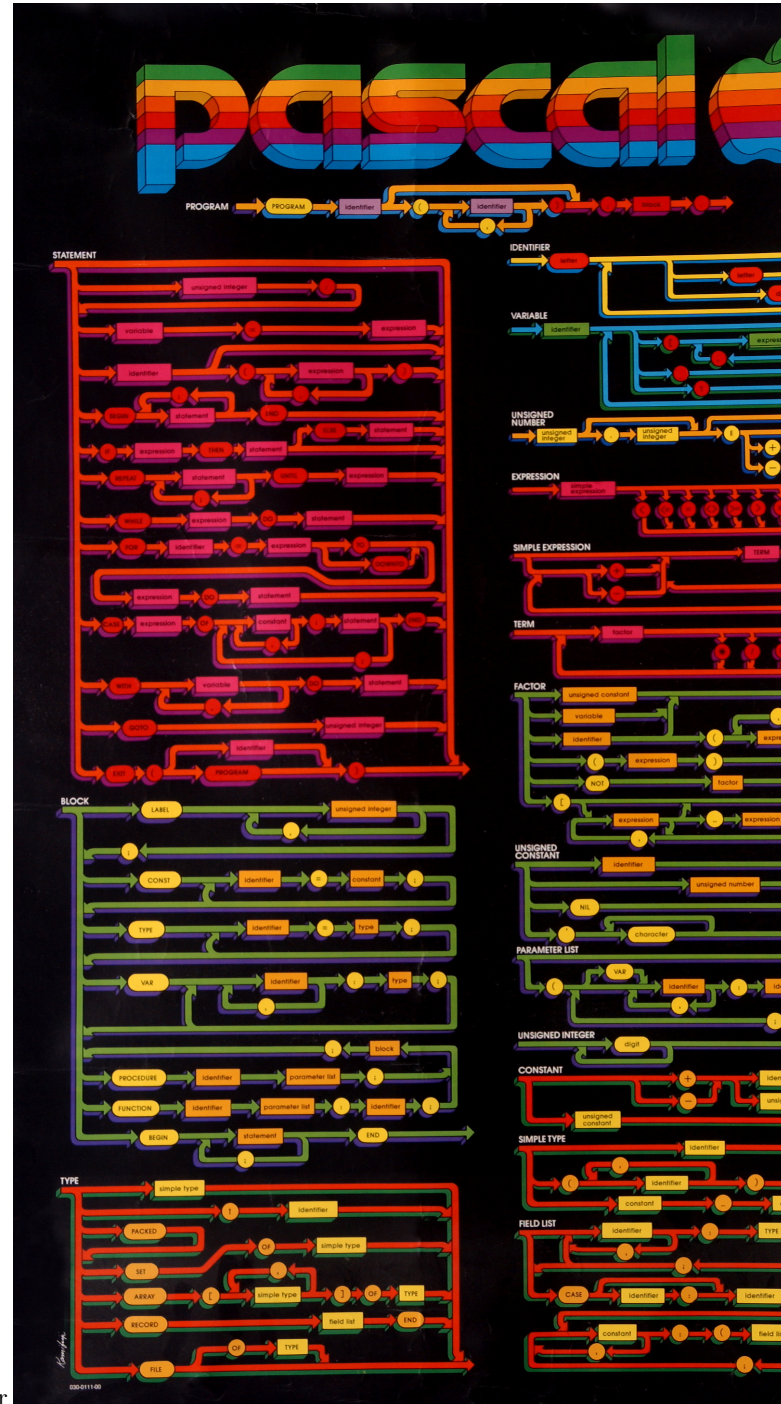
- does not support bidirectional buffering
- No provisions exist such that independent libraries can cooperate with each other in
- signal handling,
- setting up alarms, and
- tracking childs.
- Please note that I do not want to bash Ritchie, Kernighan etc. The libc is history and should be taken as such... It is time to abandon C and the libc and it does not help to place other systems on top of this historic relic.

source

## **implementations**

### **ucsd**

apple pascal poster



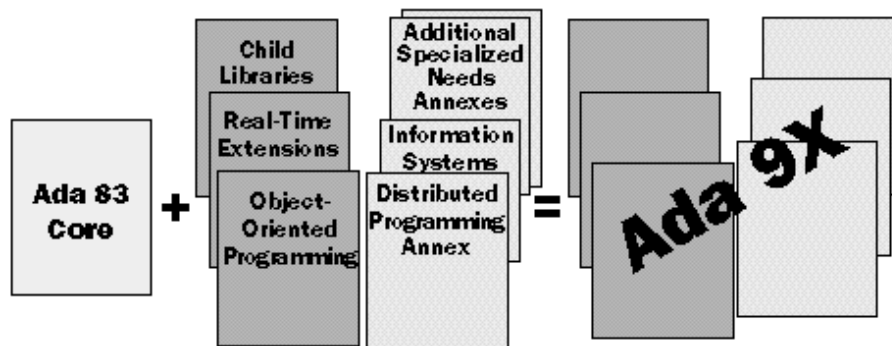
poster history jeff raskin the poster # poster

# iso pascal

iso

standards faq

## Ada 83 -> Ada95



## Ada OOP

any further components. For example

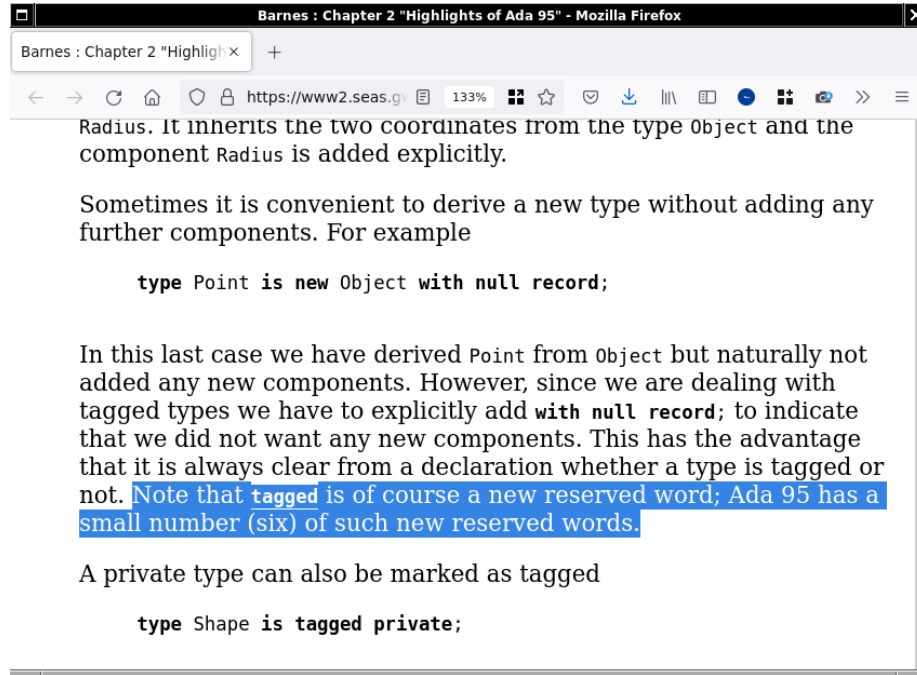
```
type Point is new Object with null record;
```

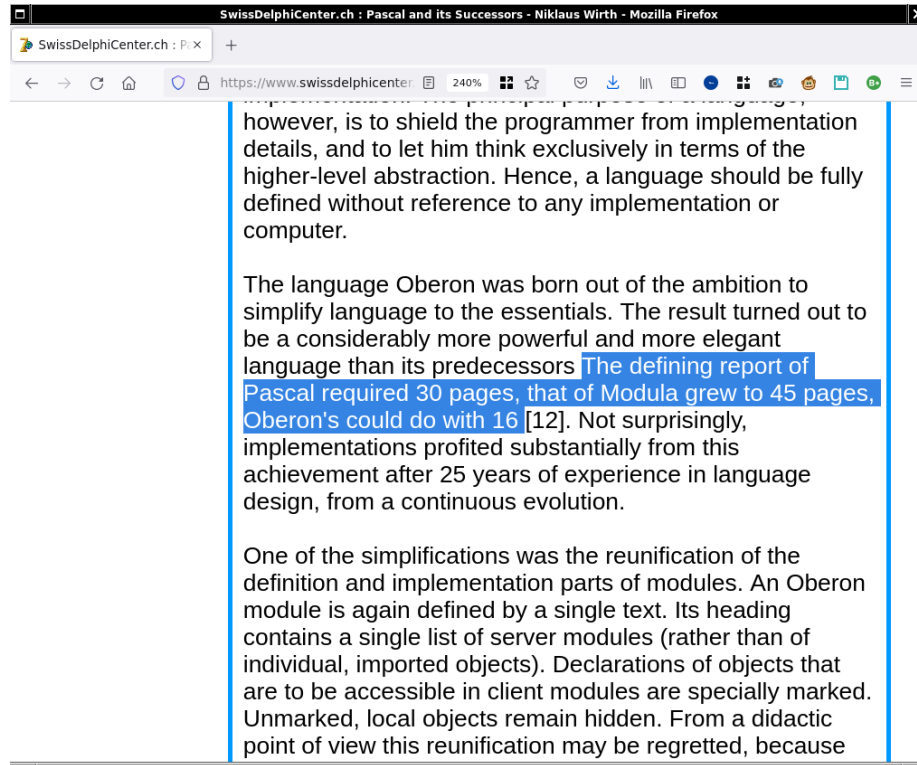
In this last case we have derived Point from Object but naturally not added any new components. However, since we are dealing with tagged types we have to explicitly add with null record; to indicate that we did not want any new components. This has the advantage that it is always clear from a declaration whether a type is tagged or not. Note that tagged is of course a new reserved word; Ada 95 has a small number (six) of such new reserved words.

A private type can also be marked as tagged

```
type Shape is tagged private;
```

## Ada OOP





- report -

## apple version

```
program ObjectPascalExample;

type
  THelloWorld = object
    procedure Put;
  end;

var
  HelloWorld: THelloWorld;

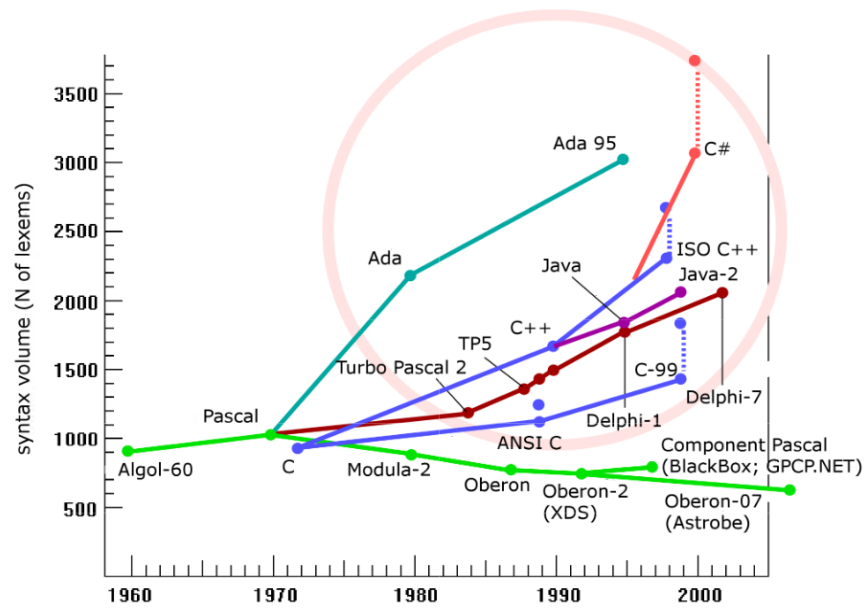
procedure THelloWorld.Put;
begin
  ShowMessage('Hello, World!');
end;
```

```

begin
  New(HelloWorld);
  HelloWorld.Put;
  Dispose(HelloWorld);
end.

```

## complexity



sources:

less is more, why oberon beats mainstream in complex applications





## headers/classes/namespaces are not modules

- one header can describe the interface of several object files
- one object file can be represented by several headers
- no way to ensure type checks across module boundaries
- we need extra tools like make to solve design problems we introduced

...



```
#include "stdio.h"

int add (int a, int b);

int main()
{
    int i, j, k;
    i = 40; j = 2;
    k = add (i, j);
    printf ("res = %d\n", k);
    return 0;
}
```

[illegible]

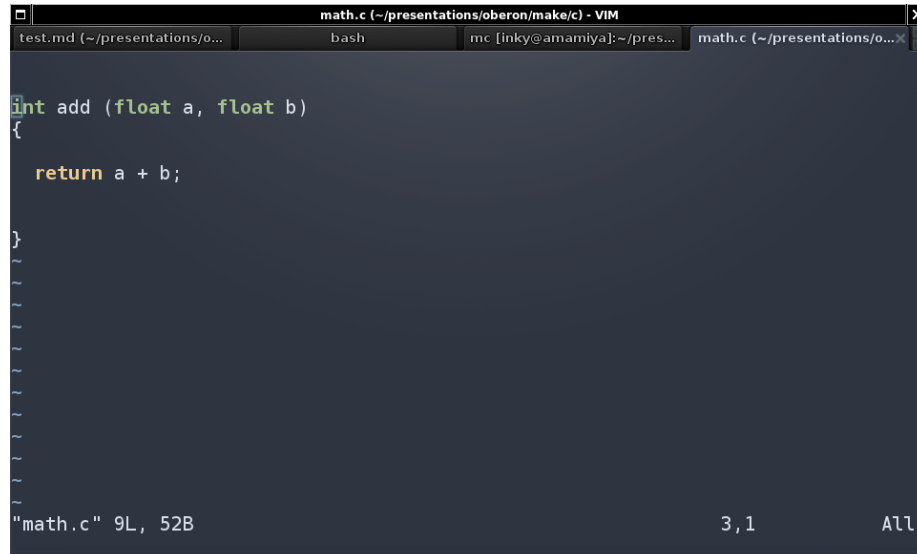
```
#include "stdio.h"
//int add (int a, int b);
#include "refrigerator.h"

int main()
{
    int i, j, k;
    i = 40; j = 2;
    k = add (i, j);
    printf ("res = %d\n", k);
    return 0;
}

~
~
~
~
```

"test.c" 17L, 182B written

...

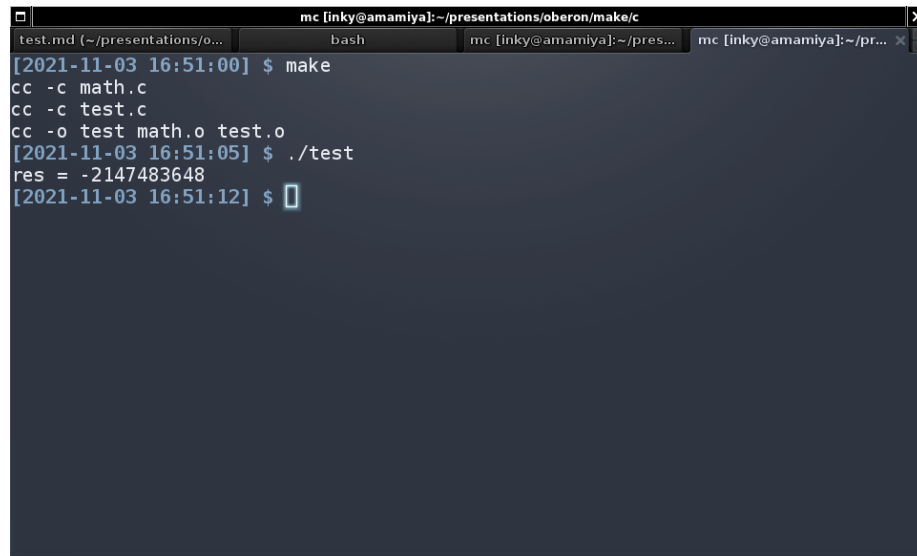


The screenshot shows a Vim editor window with the title bar "math.c (~/.presentations/oberon/make/c) - VIM". The editor is displaying a C function named "add" that takes two float arguments, "a" and "b", and returns their sum. The code is as follows:

```
int add (float a, float b)
{
    return a + b;
}
```

The status bar at the bottom of the window indicates the file name and line/byte counts: "math.c" 9L, 52B. The cursor is positioned at line 3, column 1.

...



The screenshot shows a terminal window with the title bar "mc [inky@amamiya]: ~/.presentations/oberon/make/c". The terminal displays the following commands and output:

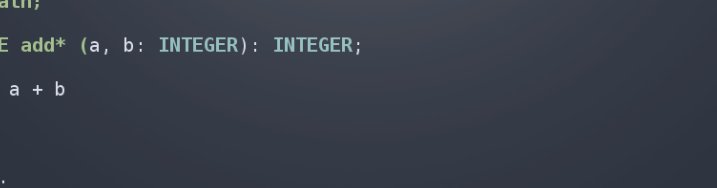
```
[2021-11-03 16:51:00] $ make
cc -c math.c
cc -c test.c
cc -o test math.o test.o
[2021-11-03 16:51:05] $ ./test
res = -2147483648
[2021-11-03 16:51:12] $
```

```
test.md (~/presentations/o/...  bash  mc [inky@amamiya]:~/pres...  test.Mod (~/presentation... x
MODULE test;
IMPORT math, Out;

VAR
    i, j, k: INTEGER;

BEGIN
    i := 40;
    j := 2;
    k := math.add (i, j);
END test.

"test.Mod" 12L, 119B 1,1 All
```



The screenshot shows a Vim editor window with the title bar "math.Mod (~/.presentations/oberon/make/o) - VIM". The editor displays the following code:

```
MODULE math;  
  
PROCEDURE add* (a, b: INTEGER): INTEGER;  
BEGIN  
    RETURN a + b  
END add;  
  
END math.
```

The status bar at the bottom of the window shows "math.Mod" 9L, 97B, indicating the current file, line, and byte position.

...

```
mc [inky@amamiya]: ~/presentations/oberon/make/c
test.md (~/presentations/o... bash mc [inky@amamiya]: ~/pres... mc [inky@amamiya]: ~/pr...
[2021-11-03 17:20:46] $ voc -s math.Mod
math.Mod Compiling math. 365 chars.
[2021-11-03 17:20:58] $
[2021-11-03 17:20:59] $
[2021-11-03 17:20:59] $ cat math.sym
00mathmath||a|badd[2021-11-03 17:21:04] $
[2021-11-03 17:21:05] $
[2021-11-03 17:21:05] $
[2021-11-03 17:21:05] $ showdef math.sym
DEFINITION math;

    PROCEDURE add(a: INT16; b: INT16): INT16;

END math.
[2021-11-03 17:21:07] $
```

...

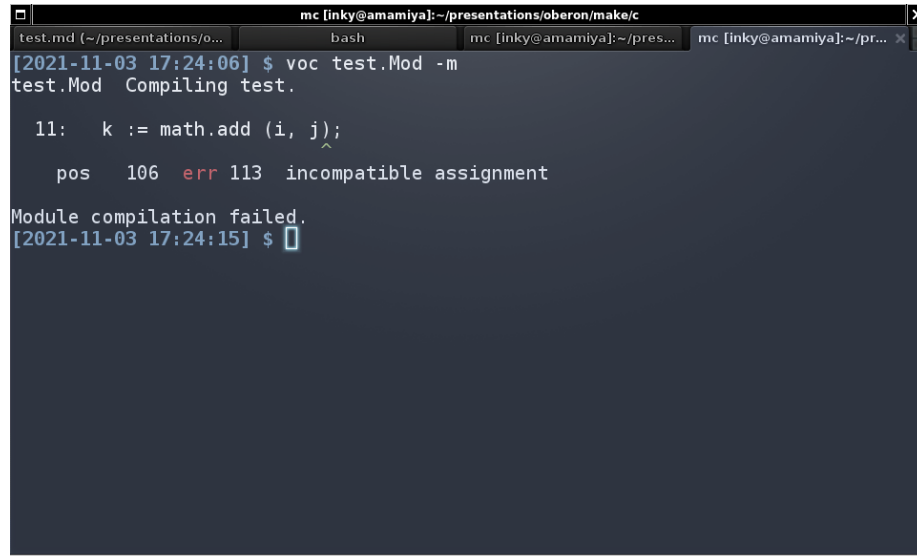
```
mc [inky@amamiya]: ~/presentations/oberon/olr
test.md (~/presentations/o... bash mc [inky@amamiya]: ~/pres... math.Mod (~/presentatio...
MODULE math;

PROCEDURE add* (a, b: REAL): REAL;
BEGIN
    RETURN a + b
END add;

END math.

3,1 All
```

...



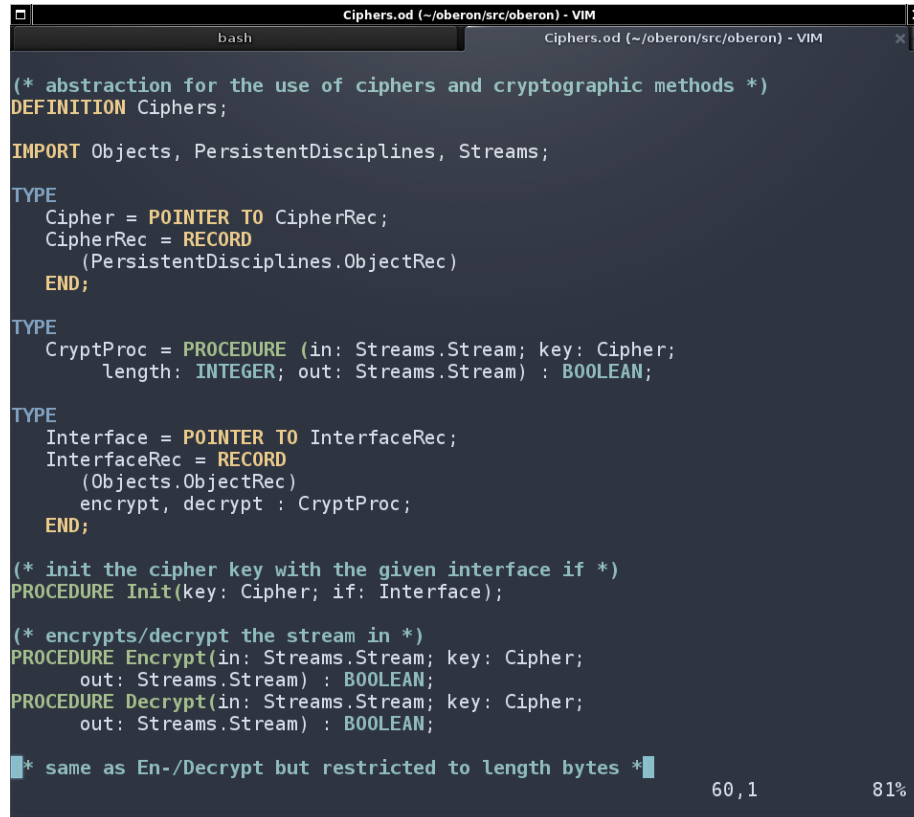
The screenshot shows a terminal window with a dark background. The title bar of the window reads "mc [inky@amamiya]: ~/presentations/oberon/make/c". The terminal content shows the execution of the command "voc test.Mod -m", which triggers the compilation of "test.Mod". A syntax error is reported at line 11, column 106, stating "err 113 incompatible assignment". The error points to the variable "j" in the expression "math.add (i, j);". The message "Module compilation failed." is displayed, followed by a timestamp "[2021-11-03 17:24:15]" and a prompt "\$ " with a cursor.

```
mc [inky@amamiya]: ~/presentations/oberon/make/c
test.mod (~/presentations/o... bash mc [inky@amamiya]: ~/pres... mc [inky@amamiya]: ~/pr...
[2021-11-03 17:24:06] $ voc test.Mod -m
test.Mod Compiling test.

11: k := math.add (i, j);
      ^
pos 106 err 113 incompatible assignment

Module compilation failed.
[2021-11-03 17:24:15] $ 
```

## definition



```
(* abstraction for the use of ciphers and cryptographic methods *)
DEFINITION Ciphers;

IMPORT Objects, PersistentDisciplines, Streams;

TYPE
  Cipher = POINTER TO CipherRec;
  CipherRec = RECORD
    (PersistentDisciplines.ObjectRec)
  END;

TYPE
  CryptProc = PROCEDURE (in: Streams.Stream; key: Cipher;
    length: INTEGER; out: Streams.Stream) : BOOLEAN;

TYPE
  Interface = POINTER TO InterfaceRec;
  InterfaceRec = RECORD
    (Objects.ObjectRec)
    encrypt, decrypt : CryptProc;
  END;

(* init the cipher key with the given interface if *)
PROCEDURE Init(key: Cipher; if: Interface);

(* encrypts/decrypt the stream in *)
PROCEDURE Encrypt(in: Streams.Stream; key: Cipher;
  out: Streams.Stream) : BOOLEAN;
PROCEDURE Decrypt(in: Streams.Stream; key: Cipher;
  out: Streams.Stream) : BOOLEAN;

* same as En-/Decrypt but restricted to length bytes *
```

## encapsulation

c

## initialization order

Another often cited criticism of C++, the missing initialization order, also solved by Oberon's modules. In contrast of cpp's include mechanism, the import relationship forbids cycles. Thus, the imported modules can always be initialized before their clients.

## separation of safe and dangerous code

SYSTEM

,

SYSTEM

## modules vs java packages

Modules vs. Java packages

Java has the notion of a package. Classes belonging to the same package may access non-public members of each other. The same can be said of Modula-2 or Oberon Modules. Within a module boundary, no restriction exists, while only explicitly exported types, variables and procedures can be accessed from outside a module.

But there is a fundamental difference between the Modules and Java packages. The Module is also the basic compilation unit. Thus when you collect things into a module, you know, that they are protected from outside interferences. Java choose to elect the class as basic compilation unit. Thus everybody may add classes to a package, and you loose the protection against outsiders.

## books

### brinch hansen on pascal compilers

### software tools in pascal

### pascal implementation

pascal implementation book

## criticism

### my criticism

- TMultiReadExclusiveWriteSynchronizer
- we need a subset.



## **language lineage**

sverdlov

## **power saving**

## **comparison**

## **features**

closures properties & events

## **implementation of fpc**

no glibc

## **inline variables**

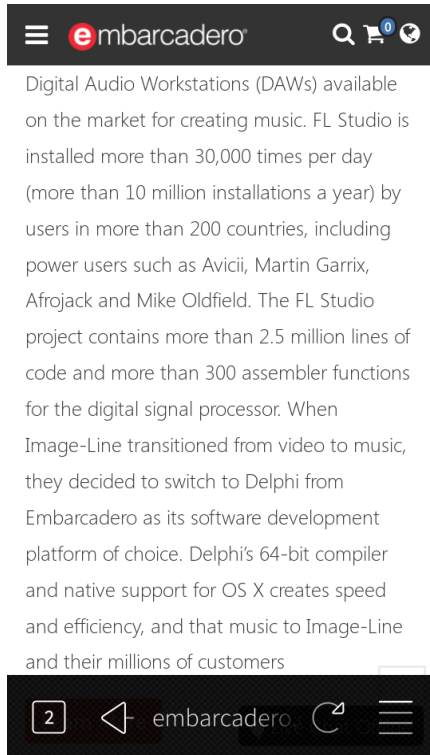
marco cantu blog lazarus forum discussion

## **c++ builder**

c++ became so complicated that embarcadero gave up developing own c++ compiler they apply patches to clang's compiler cppcast link

cpp delegates in builder my blog post on this # featured project photoshop

## featured project



freetype # flstudio  
amiga blog post on amiga & lazarus

# crossplatform:

## crossplatform app

android app

## featured project

MIUET

## featured project

gopher viewer

## software written in pascal

total commander tuxcmd # lazarus is community choice on sourceforge in 2019  
lazarus - community choice

— fpc & lazarus

## swag

link

## other sw written in pascal

[https://wiki.freepascal.org/Lazarus\\_Application\\_Gallery](https://wiki.freepascal.org/Lazarus_Application_Gallery) [https://wiki.freepascal.org/FPC\\_Applications/Projects\\_Gallery](https://wiki.freepascal.org/FPC_Applications/Projects_Gallery) [https://wiki.freepascal.org/Projects\\_using\\_Lazarus](https://wiki.freepascal.org/Projects_using_Lazarus) [https://delphi.fandom.com/wiki/Good\\_Quality\\_Applications\\_Built\\_With\\_Delphi](https://delphi.fandom.com/wiki/Good_Quality_Applications_Built_With_Delphi) [http://www.edm2.com/index.php/Category:Software\\_written\\_in\\_Pascal](http://www.edm2.com/index.php/Category:Software_written_in_Pascal)

## links

[https://wiki.lazarus.freepascal.org/Lazarus\\_videos](https://wiki.lazarus.freepascal.org/Lazarus_videos)  
<http://www.copperwood.com/pub/FreePascalFromSquareOne.pdf>  
<http://code.sd/startprog/>  
<http://code.sd/startprog/StartProgUsingPascal.pdf>  
[https://castle-engine.io/modern\\_pascal](https://castle-engine.io/modern_pascal)  
[https://castle-engine.io/modern\\_pascal\\_introduction.pdf](https://castle-engine.io/modern_pascal_introduction.pdf)  
<https://www.blaisepascalmagazine.eu/product/learn-program-using-lazarus-electron>  
<https://www.win.tue.nl/~wstomv/edu/delphi/>  
[https://www.learndelphi.org/wp-content/uploads/2020/03/DelphiProgrammingForBeginners\\_ENG-Cre](https://www.learndelphi.org/wp-content/uploads/2020/03/DelphiProgrammingForBeginners_ENG-Cre)  
[https://wiki.freepascal.org/Lazarus\\_Tutorial](https://wiki.freepascal.org/Lazarus_Tutorial)  
[https://wiki.freepascal.org/Lazarus\\_Resources](https://wiki.freepascal.org/Lazarus_Resources)  
[https://wiki.freepascal.org/Overview\\_of\\_Free\\_Pascal\\_and\\_Lazarus](https://wiki.freepascal.org/Overview_of_Free_Pascal_and_Lazarus)  
[https://wiki.freepascal.org/Pascal\\_and\\_Lazarus\\_Books\\_and\\_Magazines](https://wiki.freepascal.org/Pascal_and_Lazarus_Books_and_Magazines)

...