CreativE

Tomasz Wiszkowski

COLLABORATORS

	TITLE : CreativE	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Tomasz Wiszkowski	April 15, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

CreativE

1.1 CreativE

```
                  CreativE 2.0  [giftware]
~~~~~~~~~~~~~~~
```

```
Introduction
About CreativE

Compatibility
About the things that may not work

Commands
About some new commands

Preprocessor
About new preprocessor commands

Keywords
About some new keywords

Variables
About some new variables

Constants
About some new constants

Operators
About some new operators (new!)

LONG strings
About "LONG" strings

Normal strings
About 'Normal' strings
```

BOLD means: new features here

1.2 Introduction

Few months ago author of great programming language Wouter van Oortmerssen decided to release a free version of E with sources and (afaik) gave up developing it. Now there are only a few people who decided to keep it alive, but it's a very hard work. One of those people is me. Now I present a bit enhanced version of this great language. I will develope it as long as possible.

Current version of CreativE is the first public release, and, I hope, it will work fine with all the sources for previous versions of E (see Compatibily).

CreativE is GIFTWARE, so if you use it and like it ;), you should send me a gift (or sth ;).

I TAKE NO RESPONSIBILITY blah blah blah. You know what I want to say ;)

Please note this version is rewritten. I had a hd crash few weeks ago and I lost all the sources; anyway - this should be _STABLE_ now :). Almost everything has changed comparing to the pre-2.0 versions. Some commands are skipped, some things has changed completely or partially; I suggest reading this guide again...

1.3 Compatibility

This node should never appear, but because of some changes I have made to E it did. The changes give You more power but force You to fix Your code a bit. So, here's the list:

String formatting

The problem is in the '%' sign. EC v3.3a placed a normal percent char there. My version lets you use C alike format strings (%ld, %s etcetera). That's why you MUST place double percent sign (%%) in command if you want it in your output (only for commands that use formatted output, e.g. WriteF, PrintF, Vfprintf etcetera).

1.4 New commands

This is the whole list of commands I have added to current release.

Alloc()

Chk()

CoerceMethod()

CoerceMethodA()

CtrlD()

CtrlE()

CtrlF()

DoMethod()

DoMethodA()

DoSuperMethod()

DoSuperMethodA()

```
                Eof()

                Fclose()

                Fopen()

                Free()

                Get()

                GetA4()

                Gets()

                PutF()

                ReadB()

                Set()

                Sets()

                Size()

                WriteB()
```

1.5 Alloc()

```
                   Alloc()

SYNOPSIS
  mem:=Alloc(size)

FUNCTION
  Function allocates POOL memory if it is present. Pool
  allocations are quite fast and prevent memory fragmentation.
  If no pool is created, function will call New() instead.

INPUTS
  size - size of memory to alloc

RESULT
  mem - pointer to allocated memory or 0 if allocation failed

SEE ALSO

                Free()
```

1.6 Chk()

```
  Chk()

SYNOPSIS
  bool:=Chk(a)

FUNCTION
  Function checks parameter and returns FALSE if it is equal to
  0 or TRUE if it's not.

INPUTS
  a - variable, expression or anything else to be checked

RESULT
  bool - boolean value

SEE ALSO
```

1.7 CoerceMethod()/CoerceMethodA()

```
                  CoerceMethod()/CoerceMethodA()

SYNOPSIS
  res:=CoerceMethod(class, object, message, ...)
  res:=CoerceMethodA(class, object, message)

FUNCTION
  Function invokes the supplied message on the specific object as
  though it were the specified class

INPUTS
  class   - pointer to boopsi class
  object  - pointer to boopsi object
  message - method-specific message to be send

RESULT
  res - class and message specific result

NOTE
  This function is v36+ only!

SEE ALSO

               DoMethod()
              ,
               DoSuperMethod()
```

1.8 CtrlD()/CtrlE()/CtrlF()

```
  CtrlD()/CtrlE()/CtrlF()
```

```
SYNOPSIS
  bool:=CtrlD()
  bool:=CtrlE()
  bool:=CtrlF()

FUNCTION
  Function checks for breaks and returns TRUE if the signal was
  received

INPUTS
  none

RESULT
  bool - holds TRUE if break was received

SEE ALSO
```

1.9 DoMethod()/DoMethodA()

```
                  DoMethod()/DoMethodA()

SYNOPSIS
  res:=DoMethod(object, message, ...)
  res:=DoMethodA(object, message)

FUNCTION
  Function invokes the supplied message on the specified object

INPUTS
  object  - pointer to boopsi object
  message - method-specific message to be send

RESULT
  res - object and message specific result

NOTE
  This function is v36+ only!

SEE ALSO

               CoerceMethod()
              ,
               DoSuperMethod()
```

1.10 DoMethod()/DoMethodA()

```
                  DoSuperMethod()/DoSuperMethodA()

SYNOPSIS
  res:=DoSuperMethod(class, object, message, ...)
  res:=DoSuperMethodA(class, object, message)
```

```
FUNCTION
  Function invokes the supplied message on the specified object
  though as it were the superclass of the specified class

INPUTS
  class   - pointer to boopsi class
  object  - pointer to boopsi object
  message - method-specific message to be send

RESULT
  res - class and message specific result

NOTE
  This function is v36+ only!

SEE ALSO

               CoerceMethod()
              ,
               DoMethod()
```

1.11 Eof()

```
                  Eof()

SYNOPSIS
  bool:=Eof(fh)

FUNCTION
  Function checks if the EOF has been reached

INPUTS
  fh - pointer to DOS filehandle structure

RESULT
  bool - holds TRUE if the file reached EOF, otherwise it's false

SEE ALSO

               Size()
```

1.12 Fclose()

```
                  Fclose()

SYNOPSIS
  Fclose(fh)

FUNCTIONS
  Function closes file opened previously with Fopen()
```

```
INPUTS
  fh - filehandle obtained from Fopen()

RESULT
  none

SEE ALSO

               Fopen()
```

1.13 Fopen()

```
                  Fopen()

SYNOPSIS
  fh:=Fopen(name, mode)

FUNCTION
  function opens DOS file using standard Open() command and stores
  the filehandle in global list of filehandles. All the opened files
  will be closed automatically at the end of program

INPUTS
  name - name of file to be opened
  mode - open file mode

RESULT
  fh - filehandle that can be used with any DOS command

SEE ALSO

               Fclose()
              ,
               ReadB()
              ,
               WriteB()
```

1.14 Free()

```
                  Free()

SYNOPSIS
  Free(mem)

FUNCTION
  Function disposes memory allocated previously with Alloc() command

INPUTS
  mem - pointer to memory obtained from Alloc()
```

```
RESULT
  none

SEE ALSO

               Alloc()
```

1.15 Get()/Gets()

```
                  Get()/Gets()

SYNOPSIS
  res:=Get(object, attr, store)
  res:=Gets(object, attr)

FUNCTION
  Ask specified object for a value assigned to specified attribute

INPUTS
  object - pointer to boopsi object
  attr   - attribute tag id
  store  - pointer to storage for the answer

RESULT
  res - value assigned to specified attribute (Gets);
        FALSE if the inquiries of attribute are not provided by the
        object's class (Get)

NOTE
  This function is v36+ only!

SEE ALSO

               Set()
```

1.16 GetA4()

```
  GetA4()

SYNOPSIS
  GetA4()

FUNCTION
  Restore A4 register

INPUTS
  none

RESULT
  none
```

```
NOTE
  This function don't have to be called before use. You can use it
  only in places You need it. It won't work with library mode.

BUGS
  None known.

SEE ALSO
```

1.17 PutF()

```
  PutF()

SYNOPSIS
  PutF(fh, formatstr, args...)

FUNCTION
  Function writes formatted string to selected filehandle

INPUTS
  fh        - filehandle
  formatstr - C or E alike formatstring
  args      - list of arguments

RESULT
  none

SEE ALSO
```

1.18 ReadB()

```
                  ReadB()

SYNOPSIS
  blks:=ReadB(fh, blksize, numblocks, mem)

FUNCTION
  This function reads numblocks blocks of data, each block is
  blksize long into continuous memory starting at mem

INPUTS
  fh        - DOS filehandle
  blksize   - size of one block
  numblocks - number of blocks to be read
  mem       - memory location to store blocks

RESULT
  blks - number of read blocks

SEE ALSO

               WriteB()
```

1.19 Set()/Sets()

```
                    Set()/Sets()

  SYNOPSIS
    Set(object, attr, value, ...)
    Sets(object, attr, value)

  FUNCTION
    Assign a value assigned to specified attribute of the object

  INPUTS
    object - pointer to boopsi object
    attr   - attribute tag id
    value  - value to be assigned to the attribute

  RESULT
    none

  NOTE
    This function is v36+ only!

  SEE ALSO

                  Get()
```

1.20 Size()

```
                    Size()

  SYNOPSIS
    len:=Size(fh)

  FUNCTION
    Obtain current file size

  INPUTS
    fh - DOS filehandle

  RESULT
    len - file size

  SEE ALSO

                  Eof()
```

1.21 WriteB()

```
                    WriteB()

  SYNOPSIS
```

```
  blks:=WriteB(fh, blksize, numblocks, mem)

FUNCTION
  This function writes numblocks blocks of data, each block is
  blksize long from continuous memory starting at mem

INPUTS
  fh        - DOS filehandle
  blksize   - size of one block
  numblocks - number of blocks to be read
  mem       - memory location storing blocks

RESULT
  blks - number of written blocks

SEE ALSO

               ReadB()
```

1.22 New Keywords

```
                  This is the whole list of keywords I have added to current  ↩
                     release
  of CreativE


               LINKABLE

               NOSTARTUP

               POOL

               UTILLIB

               INLINE

               UNION

               INCLIB
```

1.23 LINKABLE

```
  OPT LINKABLE

USAGE
  OPT LINKABLE

ABOUT
  This keyword (option) allows You creating linkable object code
  (.o) instead of normal executable or library.
```

```
NOTE
  This is still a ß version and the output code may not be properly
  created. I need to find some more docs about it.
```

1.24 NOSTARTUP

```
  OPT NOSTARTUP

USAGE
  OPT NOSTARTUP

ABOUT
  This switch lets You write Your own startup code. No libraries are
  opened and nothing is initialized (except execbase) in Your output
  code, You have to do everythnig Yourself. It gives You the power
  to request user about too old os version or cpu.

NOTE
  arg string is placed in A0, not in the arg variable. You don't
  have to initialize it. You must open libraries You will use later
  in your code, stdio if You want to use i/o functions and get
  wbmessage. Nothing (except allocated memory and files opened with
  Fopen()) will be closed at the end of program. You MUST close
  everything yourself!
```

1.25 k0002

```
                  OPT POOL

USAGE
  OPT POOL (memtype, puddlesize, threshsize)

ABOUT
  This switch lets You create pool that can be used later in Your
  programm e.g. via
               Alloc
               or anything else. Pool
  pointer is stored in
               __pool
               variable.
  Parameters are optional, so You can write simply OPT POOL to use
  it.

NOTE
  This is v39+ only!
```

1.26 UTILLIB

```
    OPT UTILLIB

  USAGE
    OPT UTILLIB

  ABOUT
    This switch enables  utility.library  to  be  used  in  your
    programm.  Some  utility  functions are used by  patched
    commands All the offsets appear automatically when You simply
    switch this option on.

  NOTE
    This is v37+ only!
```

1.27 INLINE

```
                    OPT INLINE

  USAGE
    OPT INLINE

  ABOUT
    This option marks some -short- E internal commands to be placed immediately
    in the code. This makes Your programs faster, but also a bit longer.
    For inline command list can be found
                here
```

1.28 k0005

```
    UNIONs in objects

  USAGE
    UNION [ [a],[b], ... ]

  ABOUT
    Since 2.04 it is possible to UNION some members in object
    definition; the main rules are:
    - All the members that needs to be unified must be placed in "[]"
    - Each "[]" repressents one group of members to union
    - Union must start with "[" and end with "]".
    - All members must be separated with commas (",")
    - members that follow each union start after the biggest unioned group
    - union declaration may be spreaded into several lines

  EXAMPLE

    OBJECT a
        UNION
        [
            [
                a, b, c
```

```
        ],[
            d:INT, e:INT, f:INT
        ],[
            g:CHAR, h:CHAR, i:CHAR
        ]
    ]
    j
ENDOBJECT

will produce:

(----) OBJECT a
(   0)   a:LONG
(   4)   b:LONG
(   8)   c:LONG
(   0)   d:INT
(   2)   e:INT
(   4)   f:INT
(   0)   g:CHAR
(   1)   h:CHAR
(   2)   i:CHAR
(  12)   j:LONG
(----) ENDOBJECT     /* SIZEOF=16 */
```

1.29 Lib support

```
Lib files support?

USAGE
  INCLIB 'libname', 'libname'...

ABOUT
  After quite hard work I managed to add sth like "lib" files support.. This
  is still a beta version and many things may change; I'm sure it will support
  inlines and default args, but as for now it's hard to say; that's why I'm not
  including any example lib files, yet.

  The main requirement (hehe ;) is to have a "ELIB:" assignment (suggested place ↩
     :
  "E:LIB"). For each lib file two must exists:

      - *.lib - the main LIB file
      - *.m   - description module

  module format is very simple - each PROC represents one entry in lib file, e. ↩
     g.

  PROC Whatever(x,y,z,a,b,c)

  Please note that _ALL_ .LIB functions MUST START WITH Capital letter preceeded ↩
      by
  a small one. Another important thing is the sequence; files in descriptor must
  be sorted as those in lib file. Example use - run an assembler, i.e. AsmOne.
  Write sth. like e.g.
```

```
        MOVE.L  4(A7),A0
        MOVE.L  8(A7),(A0)
        RTS

    Compile and write as link; note that ".lib" suffix is necessary . Now run an
    editor (ced, ged or whatever) and write sth. like:

    PROC PutLong(what,where)

    and save with same name but different suffix (this time - ".m"). Now write a
    program, e.g.

    PROC main()
        DEF a

        PutLong(5, {a})
    ENDPROC

    This will change the internal "PutLong" function with the new one You wrote.

    Advantages of lib files here is that not whole lib is included; only used  ↩
       parts
    are linked.

NOTES
    -This is still a beta feature which may (but shouldn't) disappear in one or  ↩
       more
     new versions (but will appear again ;).
    -RELOC hunks AREN'T supported! If Your link contains a reloc hunk, compiler  ↩
       will
     return an error!

BUGS
    None found (yet?)
```

1.30 New variables

```
                    This is the list of new variables I have added to current  ↩
                       release
    of CreativE

utilitybase
    points to utility.library if it was opened (see
                 UTILLIB
                )

__pool
    points to internal pool, if it was created (see
                 POOL
                )
```

1.31 New constants

```
This is the whole list of constants I have added to current
release of CreativE

TAG_DONE            = 0
TAG_END             = 0
TAG_IGNORE          = 1
TAG_MORE            = 2
TAG_SKIP            = 3
TAG_USER            = $80000000

OFFSET_BEGINNING    = -1
OFFSET_CURRENT      = 0
OFFSET_END          = 1

READWRITE           = 1004
```

1.32 New operators

```
//          equivalent to "->"
&           equivalent to "AND"
||          equivalent to "OR"
=>          equivalent to ">="
=<          equivalent to "<="

>> and <<

works just like "Shr" and "Shl", but NO  function  is  called  for
this.  Rotation  is  made  in  the  place you use it, so it's much
faster. This works exactly like in C/C++


NEW!:
<var><oper>=<expr>, e.g. a+=3

This is a quite new thing in E, but it  works  pretty  good.  Such
expression is equivalent to <var>:=<var><oper><expr> (a:=a+3).
```

1.33 LONG strings

```
\x

  This  lets You insert any long value into your "LONG" string  '\x'
  MUST be followed by two HEX digits describing the ascii number You
  want  to put instead of \x. Function will return  "ERROR:  Unknown
  HEX  number  following  \x"  if  You write something wrong. Please
  note you ALWAYS have to put TWO digits, even if the  whole  number
  fits in one. You can't use signs here!
```

1.34 Normal strings

\x

This lets You insert any value into string. "\x" MUST be followed by two digits describing HEX number which is the number of ASCII char You want to put there. You will have "ERROR: Unknown HEX number following \x" if You do something wrong.

\!

This will insert a BELL ($07) char to Your string. When You put this char to console, screen will flash

\v

This inserts a vertical tabulator ($0B)

1.35 Formatted I/O functions

\u

This puts unsigned decimal number. This is equal to %lu (RawDoFmt)

1.36 Assembler part

because of the number of added commands, I have put only the most-important informations about improvements here. Sorry, folx.

- No more "weird" operand sizes (like RTS.x or MOVE.S)
- Multiplication and division can operate on longs (020+)
- Quite big instruction set - support for CPUs (68k family), FPUs and MMUs.
- Over 400 assembler commands

Please note I don't know all the assembler instructions so I could miss some or some adressing modes might be not enabled. Please, let me know if You find some ;). Also, I am looking for some good documentations for asm instructions/adressing modes/whatever. If You have some or You know where to get these documentations from, please, let me know. Thanks.

Not supported commands:

-Pack
-Unpk
-Cas
-Cas2
-Chk2
-Cmp2
-CallM
-RtM

1.37 The Patcher

```
  The patcher allows generating code for different cpus/fpus/osversions
  and other options; Currently only those functions are patched

Kickstart

  WriteF()                37+
  PutF()                  37+
  PrintF()                37+
```

1.38 I wish to thank to...

```
  It's a list of people I wish to thank

Wouter van Oortmerssent
  ... for writing the best programming language ever!

Gateway
  ... for keeping alive the best computer ever!

Dietmar Eilert
  ... for writing the BEST developers editor EVER!

Tomasz Bielinski, Tomasz Korolczuk, Grzegorz Chmiel, Rainer Müller,
 Esteve Boix, Alex van Niel

  for nice emails, betatesting and everything

Stephane Tavenard, Thomas Wenzel

  for nice & friendly emails and for everything


  Special thanks to Waldemar Skiba for patience :)... and for selling me
  040 turbo card. Thanks dude, I don't know what would I do without You 8).


  and to all people I know for mailing me =)

  Amiga, forever
                              d#P d#P
                             d#P d#P
                            d#P d#P
                           d#P d#P
                    ¶#b ¶#b d#P d#P
                     ¶#b ¶#b P d#P
                      ¶#b ¶#b d#P
                       ¶#b ¶#b#P
                        ¶#P ¶#P
                          "   "
```

1.39 Hi, it's me! :)

I don't know what to write here :). It might be because of lazyness or that I don't know who I am for real.

Ok, shortly: feel free to email me, snail me or even call me. Here are my addresses (snail and email ofcourse :)

snail:
Tomasz Wiszkowski
Katowicka 23/4
44-335 Jastrzebie Zdroj
POLAND

email:
error@alpha.net.pl

phone:
+48-36-471-23-21

Any new ideas? Write to me, too!!!

That's all! Enjoy using CreativE!

ps. I NEED ß-TESTERS ;)

1.40 PreProcessor

List of preprocessor commands I have added to this release of CreativE:

#date

1.41 #date

This preprocessor keyword is very useful when You need to place compilation date of Your programm. Imagine You have more than one such date. What will You do then? You simply use this keyword! Usage is very simple

#date store fmtstring

Ok. This macro keyword allows You to insert current date in any format You wish. The format is defined in format string. Supported keys:

- %d - day nr
- %m - month nr
- %y - year nr (4 digits)

```
    %D      -   day (name)
    %M      -   month (name)
    %Y      -   year nr (2 digits)
    %aD     -   day (abbreviated name)
    %aM     -   month (abbreviated name)

So, if You want a e.g. version string, You can write sth like:

#date Version '$VER: SpaceSheep v0.0 (%d.%aM.%Y) by LittleGreenMan'

and put it somewhere in your source... the only thing You must take care of
is the version ;).
```

1.42 What's that?

```
Blabla is a polish group that associates OS-friendly  programmers.
All  the  programms written by our members will work on most Amiga
machines and are compatible with all  the  better  configurations.
Our  programms are usually released as PD, FD or shareware, so You
can spread them as long as You want.

For more informations, suggestions, knowledge exchange or anything
else write to any of BlaBla member
```

1.43 New error and warning messages

```
A short description of error mesages...

    unknown HEX value after \x

PROBLEM:
    This error will appear every time when You use a "\x" sentence in Your  ↩
       string
    with incorrect HEX number (e.g. "\xZG" etc). Please note "\x" always eats  ↩
       TWO
    characters, not one.

HELP:
    Check Your strings for an incorrect "hex" numbers


    value expected

PROBLEM:
    Error appears every time You use e.g. a variable when a value is expected, ↩
        e.g.
    when You use a variable describing extra place in Your library definition

HELP:
    Simply place an immediate value or constant in such place.
```

```
    ")" expected

PROBLEM:
    You've probably forgotten to close a bracket :)... most commands does not ↩
       support
    it, yet, but they surely will. Example: GetA4(

HELP:
    Put a closing bracket


    even number expected

PROBLEM:
    An even number is required. This pops up especially when You try to create
    a library space with odd number of bytes

HELP:
    Round Your value to two


    you need a newer OS for this

PROBLEM:
    You have probably used a compiler option which is not supported by Your
    operating system. This message will usually pop when You're tring to use
    some compiler ops (e.g. #date) on a pre-2.0 operating system

HELP:
    Well... You must try to remove the object that causes this error or map ↩
       Your
    ROM/buy a new kickstart


    unable to open resource

PROBLEM:
    CreativE is unable to open a required resource (e.g. battclock.resource to ↩
        use
    with #date preprocessor macro)

HELP:
    Reboot Your amiga and try again


    this instruction needs a newer OS version (see OPT)

PROBLEM:
    The command(s) You've used are not supported by the system you're ↩
       compiling
    a program to.

HELP:
    change OSVERSION setting (e.g. OPT OSVERSION=37)


    illegal size
```

```
PROBLEM:
    The assembler size is not supported by mnemonic (e.g. ADDA.B)

HELP:
    fix up the size or remove it.


    fpu register expected

PROBLEM:
    The assembler mnemonic requires at last one FPU register (usually all
    Fxxxx do) or at last one FPU control register

HELP:
    Check out the command syntax


    ":" expected

PROBLEM:
    a colon is required

HELP:
    put a colon in a proper place


    mmu register expected

PROBLEM:
    The assembler mnemonic requires at last one MMU register (usually all
    Pxxxx do) or at last one MMU control register

HELP:
    Check out the command syntax


    contol register expected

PROBLEM:
    (usually appears with MOVEC mnemonic) - the assembler command requires
    at last one control register

HELP:
    Check out the command syntax/put a control register in a proper place


    cpu register expected

PROBLEM:
    The mnemonic You've used requires a CPU register

HELP:
    Check out the command syntax/put a control register in a proper place


    this instruction works only in pool mode
```

```
PROBLEM:
    You've used probably an instruction which needs a pool to be present (e.g.
    Alloc())

HELP:
    add a POOL keyword to OPT settings


    not allowed in library mode

PROBLEM:
    The instruction You've used does not work in library mode (e.g. GetA4())

HELP:
    Remove or replace the instruction.



                    WARNINGS

    040/060 emulated instruction(s) used

PROBLEM:
    You've used some emulated instructions in Your assembly code; it means
    that Your programm will work slower on 040/060 CPUs
```

1.44 Inline commands

```
                Since 2.01 I've enhanced E with Inline commands. It means that
such commands are NO LONGER called (with BSRs); they're placed
directly in the place of call, so are working as fast as they
were a simple "+" or "*" in the expression. To use the inline commands
please look at description of
            INLINE
             keyword.
Patched are:

- Abs
- And
- Car
- Cdr
- Char
- Div (020+)
- Eor
- Eval
- Even
- Exit
- Fabs
- Facos
- Fatan
- Fasin
- Fceil
```

- Fcos
- Fcosh
- Fexp
- Ffieee
- Ffloor
- Flog
- Flog10
- Fpow
- Fsin
- Fsincos
- Fsinh
- Fsqrt
- Ftan
- Ftanh
- Ftieee
- Int
- Long
- Mod (020+)
- MouseX
- MouseY
- MsgCode
- MsgIaddr
- MsgQual
- Mul (020+)
- Not
- Odd
- Or
- PutChar
- PutInt
- PutLong
- RndQ
- Shl
- Shr

1.45 Other things

Yes... Well, this chapter contains only stuff I don't know where to put... hehe q:)c=

Object members

Another IF format

Modules vs inlines

Expressions swap

1.46 Member

Object members

Since 2.01 it is possible to make an assignment to any object's member inside the immediate list. Please note it is a beta feature but it _should_ work anyway. So, it is now possible to do things like:

```
a:=[b[c].d:=e]:x
```

1.47 Another IF expression format

Another IF format

I'm sure that many ppl will blame me for using C/C++ features in E compiler but I think that if there's a way to do sth easier, it should be implemented.. Ok, and here it is. The format is:

```
<exp> ? <texp> : <fexp>
```

Ofcourse this feature can be a bit deeper (read: you can use one in another, like brackets). <exp> stands for any expression to be checked, <texp> stands for expression to be calculated in case of true and <fexp> is the one to be calculated in case of false. THIS IS A BETA FEATURE, but should work fine.

1.48 Modules vs inline code

Modules vs inline code

Yes... this was the main problem the programs compiled with INLINE option crashes because of. Now (hehe... a bit too late) the compiler is a bit smarter - checks whether the inline code is used by the module to encounter whether copy it or not.

1.49 Expressions swap

Expressions swap

I think it's a useable feature ;). Imagine You can calculate at once two expressions and store them in specified variables later. It means that i.e. You don't have to define additional vars for temporary use; now:

```
<exp1>:=:<exp2>
```

will solve this problem. The results will be stored in the last-used variables (i.e. in a+b/c*d, "d" is the last one). Multiple use of it is possible but the variables are set after each two (i.e. when You write sth like:

```
a:=:b:=:c
```

first a is swapped with b and then b is swapped with c)

I know it’s a bit weird but it’s very difficult to explain it. Maybe a little example:

```
PROC main()
    DEF a=1, b=10, c=2

    WriteF(’a=\d, b=\d, c=\d\n’, a, b, c)
    a+15:=:b/2:=:c*2
    WriteF(’a=\d, b=\d, c=\d\n’, a, b, c)
ENDPROC
```

will produce:

```
a=1, b=10, c=2
a=5, b=4, c=16
```

so, first a+15 is calculated (=16) and put on stack; then b/2 is calculated and stored in "a". Then, b is set with value from stack and another swap is done.

IMPORTANT NOTE!

This "thing" is only for variables, NOT for objects and members. SO! If you do sth. like:

```
a.b.c:=:d.e.f
```

then ONLY POINTERS are set with expression values! In this case it is equal to sth. like:

```
t:=a
a:=d.e.f
d:=t.b.c
```