

НОВЫЕ ВОЗМОЖНОСТИ системы HomeLisp

**Файфель Б.Л.
СГТУ им Ю.А. Гагарина,
г. Саратов**

Назначение проекта

- Интерпретатор Лиспа, предназначенный для использования в учебном процессе;
- Включающий все возможности создания windows-приложений (графический интерфейс пользователя, оконная графика, файлы, СОМ-объекты);
- Поддерживающий все основные типы данных;
- Не требующий никаких дополнительных библиотек или компонентов.

Реализация основана на подходе, описанном в книге
**С.С. Лаврова и Г.С. Силагадзе “Автоматическая
обработка данных. Язык Лисп и его реализация”. М.
1978 г.**

Ранние версии соответствовали стандарту Лисп 1

Начиная с редакции ядра **1.13.1** предпринимаются систематические попытки приблизить HomeLisp к стандарту Common Lisp (Лисп 2): была введена лексическая область видимости переменных и обеспечены замыкания.

В этом сообщении будут рассмотрены изменения двух последних лет:

- Рациональные и комплексные числа;**
- Многозначные функции;**
- Универсальный итератор `iter`;**
- Структуры;**
- Интерфейс с `winAPI`**

Рациональные числа

До сих пор Лисп остается чуть ли не
единственным языком,
поддерживающим рациональную
арифметику.

Действия с рациональными числами

(+ 4 1/3)

==> 13/3

(+ 1/2 1/3)

==> 5/6

(sin -1/2)

==> -0.479425538604203

(/ 1 2)

==> 1/2

(+ 3.0 1/2)

==> 3.5

Какую пользу можно извлечь из рациональных чисел в учебно-методическом плане?

Известная задача

Вычислить с наперед заданной точностью ϵ синус и косинус по формуле Тэйлора:

$$\cos(x) = 1 - (x^2/2!) + (x^4/4!) - (x^6/6!) + \dots$$

$$\sin(x) = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + \dots$$


```
float Cos(float x, float eps)
{
    float s,a,n;
    s=a=n=1;
    while (1)
    {
        a=-a*x*x/(n*(n+1));
        s+=a;
        if (fabs(a) <= eps) break;
        n+=2;
    }
    return s;
}
```

x	cos(x)	Cos(x)	Расхождение
0.000e+000	1.000000e+000	1.000000e+000	
0.000000e+000			
5.000e-002	9.987503e-001	9.987502e-001	
1.000000e-008			
1.000e-001	9.950042e-001	9.950042e-001	
0.000000e+000			
1.500e-001	9.887711e-001	9.887711e-001	
1.850e+000	9.395249e-001	1.144768e+000	2.052426e-001
1.850000e+000			
2.900e+001	9.889666e-001	9.889666e-001	2.599001e-003
0.000000e+000			
1.950e+001	7.958150e-001	6.106047e-001	1.852103e-001
2.900e+001	-7.480575e-001	1.315880e+004	1.315954e+004
2.950e+001	-3.383192e-001	3.822034e+003	3.822372e+003
3.000e+001	1.542515e-001	-2.368533e+003	2.368688e+003

Катастрофа!

Причина заключается в том, что плавающая арифметика коварна – при суммировании рядов подобного типа происходит катастрофическая потеря точности.

Как решить эту проблему?

Помогут рациональные числа!

```
(defun my-cos (x &optional (eps 1E-8))
  (let ((a 1) (s 1) (k 0))
    (loop
      (when (<= (abs a) eps) (return s))
      (setq a (- (/ (* a x x) (+ k 1) (+ k 2)))
            s (+ s a)
            k (+ k 2))))))
```

(my-cos 50)

**2437061316545411326756033860822195498255
4281385203094554670358004072039248121649
3267961919792183534114282432569016953537
4398450626561195065523779221083103374016
6338199817232878060581913569126766599**

/

**2525541049387318433222564811495881694660
8988211936130235611855567635907889663184
4387898015300688850221053371046957284699
6825946020610949081573617550435820266050
9266505949702813572299506856327202849**

Переведем его в десятичную дробь:

(rat2flo

243706131654541132675603386082219549825542813852030945546703
580040720392481216493267961919792183534114282432569016953537
439845062656119506552377922108310337401663381998172328780605
81913569126766599/252554104938731843322256481149588169466089
882119361302356118555676359078896631844387898015300688850221
053371046957284699682594602061094908157361755043582026605092
66505949702813572299506856327202849)

□ 0.964966028620532

А теперь возьмем значение $\cos(50)$, вычисленное библиотечной функцией

(cos 50)

□ 0.964966028492113

Комплексные числа

В HomeLisp комплексные числа представляются точно в таком же виде, как в Common Lisp:

```
(* 1/3 #C(1 1))
```

```
==> #C(1/3 1/3)
```

```
(* -0.7 #C(1 1))
```

```
==> #C(-0.7 -0.7)
```

```
(* #C(0 1) #C(0 1))
```

```
==> -1
```

```
(sin #C(1 1))
```

```
==> #C(1.29845758141598 0.634963914784736)
```

```
(sqrt -1)
```

```
==> #C(0.0 1.0)
```

Многозначные функции

Для работы с функциями, возвращающими множество значений, предназначены специальные функции

MULTIPLE-VALUE-BIND

и

VALUES


```
(defun truncate (x y) (values (\ x y) (% x y)))
```

```
==> TRUNCATE
```

```
(truncate 1 2)
```

```
==> 0
```

```
1
```

```
(truncate 7 2)
```

```
==> 3
```

```
1
```

```
(multiple-value-bind (a b)
```

```
  (truncate 7 2) (printline a) (printline b))
```

```
3
```

```
1
```

```
==> 1
```

Don't loop, iterate!

<https://common-lisp.net/project/iterate/doc/>

В HomeLisp реализовано достаточно широкое подмножество функций универсального итератора.

```
(defun decart (x y)
  (let ((r nil))
    (iter (for a in x)
          (iter (for b in y)
                (collecting (list a b) into r))) r))
```

```
(decart '(a b c) '(1 2))
==> ((A 1) (A 2) (B 1) (B 2) (C 1) (C 2))
```

Структуры

Главным является макро DEFSTRUCT, которое получает на вход имя структуры и имена полей, а порождает набор функций доступа к полям и копирования структуры. Вызовы функций доступа можно использовать совместно с макро SETF для модификации полей.

```
(defstruct ship x y vx vy w)
```

```
==> (STRUCTURE)
```

```
(setq *s1* (make-ship :x 11 :y 11 :vx 0 :vy 0 :w  
6000))
```

```
==> (SHIP :X 11 :Y 11 :VX 0 :VY 0 :W 6000)
```

```
(setf (ship-x *s1*) 111)
```

```
==> 111
```

```
*s1*
```

```
==> (SHIP :X 111 :Y 11 :VX 0 :VY 0 :W 6000)
```

```
(setq *s2* (copy-ship *s1*))
```

```
==> (SHIP :X 111 :Y 11 :VX 0 :VY 0 :W 6000)
```

```
*s2*
```

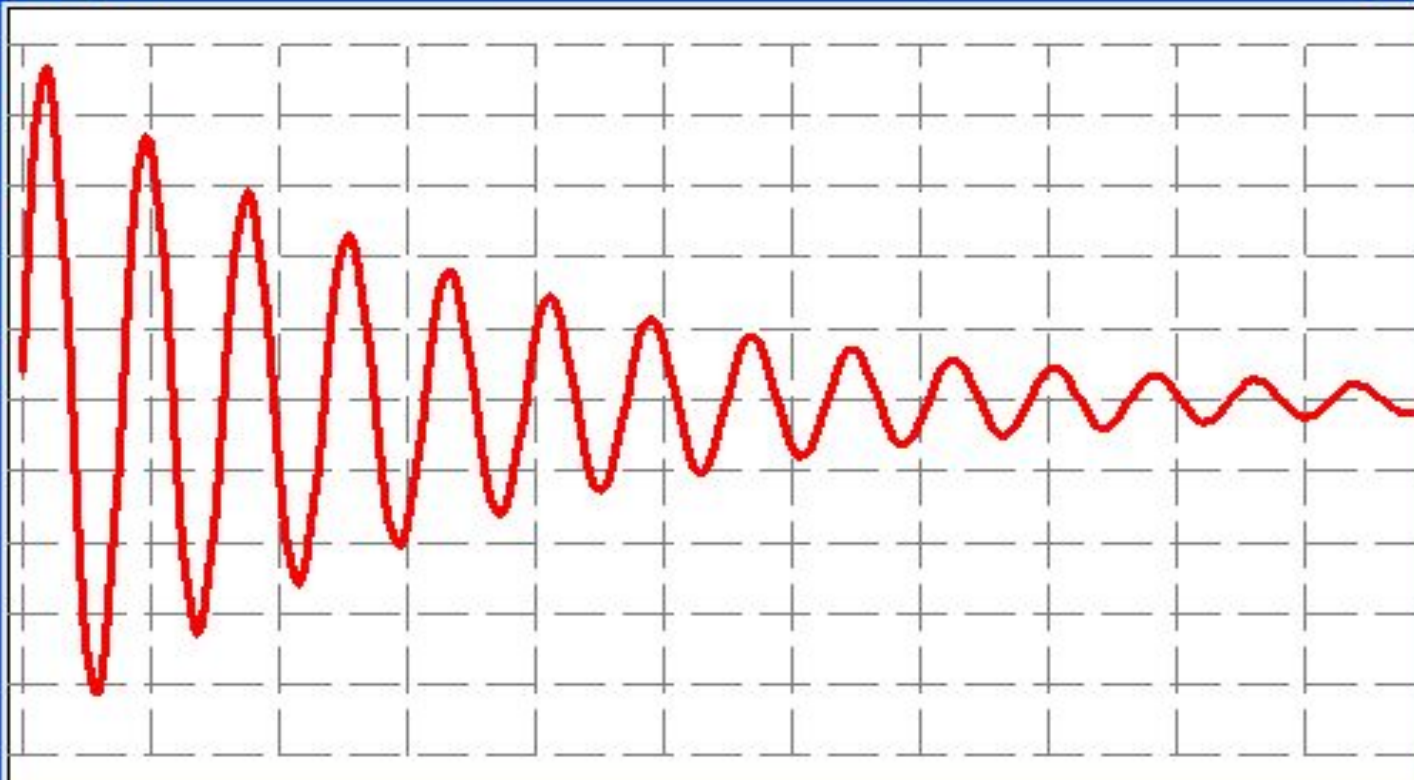
```
==> (SHIP :X 111 :Y 11 :VX 0 :VY 0 :W 6000)
```

Интерфейс с WinAPI

Интерфейс с WinAPI обеспечивается
тремя функциями:

- **LOADLIBRARY;**
- **FREELIBRARY;**
- **CALLAPI.**

Проба



Выполнить

Закреть

Спасибо за внимание!

<http://homelisp.ru>