Язык J

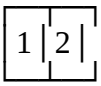
Язык J является диалектом языка APL. Преимуществом данного языка является лаконичность и выразительность кода недостатки являются следствиями преимуществ - код, часто малопонятен.

J является языком парадигмы массивного программирования - основным типом данных является массивы, к примеру
J> 1 2 3 4 **NB**. *исполняемая строка* **NB.** - *комментарий*
1 2 3 4
NB. ^ *результат*
J> 1 + 1
2
J> 1 2 3 4 + 1 2 3 4
2 4 6 8
Практически все операции в J обладают одинаковым приоритетом и приоритет отдается "правейшей" :
J> 2 2 2 2 * 1 2 3 4 + 1 2 3 4
4 8 12 16

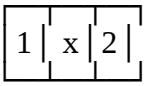
Явно понятие типа в J отсутствует, но фактически переменные могут быть следующих "типов":

число - 1
буква - 'x'
массив - 1 2 3
строка - 'hi'
короб - 1; 2

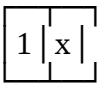
Стоит сказать несколько слов о последнем. Основным типом в J являются массивы но они могут переносить данные только одного типа, те
J> 1 , 'x' **NB.** *оператор , конкатенирует два массива те 1 , 2 , 3 , 4 тоже самое что 1 2 3 4*
domain error
Короба позволяют создавать массивы из разных типов
J> 1; 2



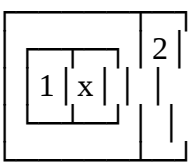
J> 1; 'x';2



J> 1; 'x'



J> (1; 'x');2



Типы операторов

J различает насколько типов операторов, для более интуитивного разделения вводится новая терминология. Для её описания будет применена запись эмулирующая математическую

Существительное – это непосредственно массив
Монада = Существительное → Существительное т.е. оператор берущий в качестве аргумента существительное и возвращающий существительное
Дидада = Существительное → Существительное → Существительное
Глагол = (Монада,Дидада) т.е. Монады и Дидады являются глаголами
Наречие = Глагол → Глагол
Союз = Глагол → Глагол → Глагол

Практически все арифметические символы в J имеют как дидадную так и монадную форму
J> - 1 NB. Дидадная форма символа “–“ - отрицание
_1
J> 1 – 1 NB. Вычитание
0
J> i:10 NB. Отрицательные числа записываются как _<число>
_10 _9 _8 _7 _6 _5 _4 _3 _2 _1 0 1 2 3 4 5 6 7 8 9 10
J> (i: 10) * (i:10)
100 81 64 49 36 25 16 9 4 1 0 1 4 9 16 25 36 49 64 81 100
J> * i:10
_1 _1 _1 _1 _1 _1 _1 _1 _1 _1 0 1 1 1 1 1 1 1 1 1 1
~ - пример наречия (f~) x = x f x наречия обладают приоритетом обратному обычному – те прервем выполняется левевший

Тактическое программирование

Часто в J аргументы некому оператору можно предоставить неявно пусть
f, g, h – глаголы
x,y - существительные

Выражение вира f g не имеет смысла так - как глагл в качестве аргумента использует существительные а не другие глаголы. Такого рода выражения J интерпретируются следующим образом:

x (f g) y ↔ x f g y ↔ x f (g y)
(f g) y ↔ y f g y ↔ y f (g y)

x (f g h) y ↔ (x f y) g (x h y)
(f g h) y ↔ (f y) g (h y)

J> x =: + * NB. Создаем функцию
J> x i: 10
_11 _10 _9 _8 _7 _6 _5 _4 _3 _2 0 2 3 4 5 6 7 8 9 10 11
J> (i:10) + * i:10
_11 _10 _9 _8 _7 _6 _5 _4 _3 _2 0 2 3 4 5 6 7 8 9 10 11

Несмотря на недостатки J является очень удобным языком для создания CAS (Computer Algebra System), так-так при использовании CAS большое количество времени тратится на составление единичных выражений, а не на организацию кода. Часто код написанный в CAS используется однажды и поэтому ключевым фактором, котором руководствуется программист, является скорость написания программ, а не легкость её поддержки и скорость исполнения. Примером CAS является языковая среда MatLab.

Несмотря на вышесказанное ПО написанное на J за определенный временной интервал может превзойти по скорости схожее ПО написанное на C за тоже время. Причиной является большое количество оптимизаций встроенных в интерпретатор J.

Redis

Redis является базой данных вида NoSQL. Особенностью Redis является скорость. Данные хранящееся в базе данных Redis хранятся в оперативной памяти и не записываются на диск, что ускоряет доступ к ним. Чаще всего Redis используют в качестве БД в веб разработке при реализации функций типа автозаполнения и пр.

Redis является крайне популярной БД, и поэтому доступ к ней может осуществляться на многих языках. Совместимость БД Redis со многими языковыми средами позволяет использовать её в качестве средства обмены данными между двумя несовместимые языковыми средами.

Redis является БД вида ключ-значение, в Redis возможно хранение следующих видов данных строковые переменные, числа, списки содержащие строковые переменные и числа.

Проект

Мною был написан интерфейс для Redis на J, таковой ранее отсутствовал. Теперь, не составляет труда при неких расчетах перевести большой массив данных из языковой среды IPython в языковую среду J.

CFFI (C Foreign Function Interface)

Для реализации проекта мною был прежде всего разработан интерфейс к коду C, облегчающий работу со структурами и типами данных. Ранее такой интерфейс отсутствовал.

CFFI API

В J присутствуют следующие библиотечных функции облегчающие взаимодействии с C.

fn **cd** args - функция подгружает библиотеку, где
fn и args имеют следующий формат:
fn '<имя библиотеки> <функция> <сигнатура>'
args <'аргумент1, аргумент2...'
memr addr, offset, num [,size] - функция считывает побайтово данные начиная с
addr + offset*size закаичивая addr + offset + num*size

Основные реализованные функции:

build (hdrs;macros;dt) - при вызове данной функции, генерируется библиотека на C, из которой добывается следующая информация:
- соответствие Макрос -> значение макроса
- соответствия структура.элемент -> размер элемента и его offset

стоит заметить, что такая информация доступна только во время сборки программы на C и варьируется в зависимости от компьютера и даже компилятора, при помощи которого собирается программа. Информация добывается при помощи создания функций возвращающих её и их вызова. Функция **build** альтерирует глобальную переменную, из которой функциями **sizeof** и **offsetof** можно извлечь соответствующую информацию. Также **build** вбрасывает в глобальное именное пространство одноименные с макросами переменные имеющие соответствующее макросу значение.

аргументы к **build** следующие:

hdrs - короб заголовочных файлов

macros - макросы,

dt - короб вида (структура0; элемент0, элемент1);(структура1; элемент0, элемент1);

sizeof 'тип' - возвращает размер типа

offsetof 'структура/тайпдеф структуры';'элемет' - возвращает соответствующий offsetof

Redis API

При помощи вышеописанного CFFI, производится взаимодействие с библиотекой hiredis - официальной библиотекой для работы с Redis. hiredis оперирует следующими структурами данных:

redisContext - структура хранящая сокет связывающий программу с БД. Кроме сокета структура содержит информацию о ошибках возникших при подключении.

redisReply - результат вызова некой функции, взаимодействующей с БД.

Реализованы следующие функции:

Функции для работы с **redisContext**

redisConn addr;port - возвращает указатель на структуру типа redisContext, addr - это ip адрес на котором запущен redis сервер, port соответствующий порт

redisFree - освобождает redisContext

Низкоуровневые функции для работы с **redisReply**

redisCommand ctx;cmd - выполняет, соответствующую cmd команду, ctx - указатель на redisContext, возвращает, указатель на структуру redisReply

reply_str_i - извлекает из указателя на структуру типа redisReply строковую переменную

reply_num_i - извлекает из указателя на структуру типа redisReply числовую переменную

reply_array_i - извлекает из указателя на структуру типа redisReply массив

reply_auto_i - извлекает из указателя на структуру типа redisReply переменную угадывая тип.

freeReplyObject - освобождает структуру redisReply

Высокоуровневые функции для работы с **redisReply**

redisCmd ctx;cmd - выполняет команду и возвращает результат автоматически извлекая его из **redisReply**.

Пример использования

```
'err ctx' =: redisConn '127.0.0.1';6379
if. err = 0 do.
   NB. Пример использования низкоуроснего API
   rep =: redisCommand ctx;'PING'
   r =: reply_str_i rep NB. Результат r = 'PONG'
   freeReplyObject rep
   rep =: redisCommand ctx;'INCR counter'
   r   =: reply_int_i rep              NB. Результат r = 1
   freeReplyObject rep
   rep =: redisCommand ctx;'LRANGE mylist 0 -1'
   r   =: 'auto'reply_array_i rep     NB. Результат r = 1,2,3,4,5
   freeReplyObject rep
   NB. Пример использования высокоуровнего API
   r =: redisCmd ctx; 'PING'    NB. r = 'PONG'
   echo 'HELLO'
   r =: redisCmd ctx;'INCR counter' NB. r = 2
end.
redisFree ctx
```