СОЗДАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ BETTERCRY

Кучуков Д. М., науч. рук. ст. преп. Первова Н.В., доцент, к.т.н Павлов Л.А.

ФГБОУ ВО «ЧГУ им. И.Н. Ульянова»

Чувашский государственный университет имени И. Н. Ульянова

Цель работы: Создание языка программирования, замену известного изотерического ЯП brainf\*ck, пригодную при решении реальных задач программирования.

Чтобы не потерять основную идею brainf\*ck, а конкретно минималистичность синтаксиса и простоту интерпретации, первым решением было наложить жёсткие рамки на синтаксис языка:

* Максимум один байт на команду
* Команда может иметь максимум один аргумент
* На хранение аргумента не должно тратится более одного байта

Язык работает напрямую с памятью, необходимое количество которой выделяется при запуске программы пользователем. Также язык предусматривает один регистр, который можно читать и модифицировать по ходу работы программы.

Значения в памяти и регистре всеми командами рассматриваются как целые беззнаковые байты, любые переполнения игнорируются.

Язык состоит из команд, каждая из которых может принимать (а может не принимать) аргумент одного из данных типов:

* CHAR – Символьное значение, команда получает ASCII код указанного в качестве аргумента символа
* CHEX – Числовое значение в диапазоне от 0 до F16
* HEX – CHEX, но позволяется также использование оператора чтения из памяти ‘.’

Всего в языке на данный момент 25 различных команд, куда входят **команды перемещения по коду**:  
*‘L’ - метка,  
‘G[CHEX]’ – переход на указанное число меток вперёд,  
‘B[CHEX]’ – переход на указанное число меток назад;  
Стоит отметить, что G и B игнорируются если на момент из исполнения в регистре находится значение 0, что позволяет производить условные переходы.  
Также решение реализовать goto и backto как относительные переходы позволило реализовать неограниченные переходы без нарушения постулат, поставленных нами выше.*

*Пример: L […] $T B1 [Бесконечный цикл]***Команды перемещения по памяти:**  
*‘A’ – переход на указанную в буфере ячейку памяти,  
‘P’ – чтение в буфер адреса текущей ячейки (записываются только младшие 8 бит),  
‘>’ и ‘<’ – переходы к следующей и предыдущей ячейкам памяти соответственно;*

*Пример: ‘A W > ‘B W > ‘C W [вывод ‘ABC’ в память]***Команды, выполняющие арифметические операции над регистром:** *‘+’, ‘-‘, ‘\*’, ‘/’, ‘%’, вид которых хорошо описывает их функционал,  
команда, генерирующая случайные числа ‘N [HEX]’;*

*Пример: I -C -C -C -C W \*. [Перевод символа введённого пользователем в число (‘0’ → 0) и возведение его в квадрат]***Операции ввода/вывода:** *‘I’ – ввод символа с клавиатуры,  
‘’ [CHAR]’ – вывод символа в буфер,  
‘#’ – вывод символа из регистра в буфер  
‘F’ – flush – вывод содержимого буфера на экран*

*Пример: ‘H ‘e ‘l ‘l ‘o ‘ ‘w ‘o ‘r ‘l ‘d F [вывод ‘Hello world’ на экран]***Команды для работы с памятью, чтению/записи:** *‘R’ – чтение значения текущей ячейки памяти в регистр  
‘W’ – запись в текущую ячейку памяти значения из регистра*

*Пример: ^5 -C -C -C -C W L R =0 ! G1 R -1 > W $T B1 L [цикл for, записывающий в память значения от 5 до 0 по порядку]***Логические и побитовые операции:** *‘= [HEX]’ – сравнивает значение с регистром, если равны, то в регистр записывается FF16, иначе 0  
‘? [HEX]’ – сравнивает значение с регистром, если регистр больше – пишет FF16 в регистр, иначе 0  
‘& [HEX]’ – побитовое И с регистром  
‘| [HEX]’ – побитовое ИЛИ с регистром  
‘!’ – побитовое отрицание регистра, т.к. логические значения хранятся как 0 и FF16 то служит и как логическое отрицание*

*Пример: $02 W $02 =. G1 ‘W ‘h ‘a ‘t ‘! ‘? $T G2 L ‘O ‘k ‘. L F [если математика работает правильно и 2 = 2, то выведет ‘Ok.’ иначе ‘What!?’]***И операция записи в регистр:** *‘^ [CHAR]’ – запись кода символа в регистр*

*Пример: ^5 -C -C -C -C W ^3 -C -C -C -C +. +C +C +C +C # F [перевод символов ‘5’ и ‘3’ в числа, их сложение, перевод результата в символ и вывод на экран]*

Также имеется синтаксический сахар – конструкции, упрощающие процесс написания кода, но не являющиеся фактической частью языка. Такие конструкции будут преобразованы в BetterCry код на этапе компиляции:

* Импорт (*формат: [&dir/file.bc]*) – компилирует содержимое файла ‘*dir/file.bc*’ и вставляет его в текущий файл.
* Макросы (*формат: [:name: ..here comes a body..]*) – описывает макрос, который можно будет использовать где угодно далее по коду (*формат: :name*). Макрос – это кусок кода, который будет вставлен в первозданном виде везде, где он был вызван.
* Комментарии (*формат: [your useful comment]*)
* Запись значения в регистр (*формат: $F0*). Данная конструкция была введена с целью дать программисту возможность безболезненно вводить в регистр значения целого байта и при этом не противоречить постулатам, данным в самом начале статьи. Во время компиляции преобразуется в команду ‘^’
* Запись логического значения в регистр (*формат: $T*) –*‘$T’* можно считать синонимом *‘$FF’*, когда *‘$F ’* синонимом *‘$00’.* Важно использовать пробел после *‘$F’*
* Строки (*формат: “yeah, strings!”*) – позволяют отправлять в буфер вывода символы с удобством, преобразуется в последовательность *‘’ [CHAR]’*. Позволяет записать популярную программу «Привет, мир!» как *”Hello, world!” F*, делая BetterCry языком с одним из самых коротких «Привет, мир», который по компактности ровняется с KiXtart!

Пример программы-игры «Rock Paper Scissors»:  
*[&io.bc] [Importing lib for useful macros]  
  
[Defining a macro which will print our sign]  
[:sign:  
 R =0 ! G1  
 “Rock” G3 L  
 R =1 ! G1  
 “Paper” G2 L  
 R =2 ! G1  
 “Scissors” L  
]  
  
[Printing a prompt]  
“RPS Game” :ln  
“Choose your sign:” :ln  
“0 – Rock, 1 – Paper, 2 – Scissors” :ln  
F  
  
[Getting user input]  
I :todig W  
  
?2 GA [Checking value that we got from user]  
 [10 labels total (4 here and 3 in each macro we are calling)]  
 [Printing gotten sign]  
 “Your choice is: ” :sign :ln F  
  
 [Generating and printing opponents sign]  
 > N2 W  
 “AIs choice is: ” :sign :ln F  
  
 [Calculating the winner]  
 R +3 < -. %3 >> W  
  
 R =0 ! G1  
 “Draw!” G3 L  
 R =1 ! G1  
 “AI win!” G2 L  
 R =2 ! G1  
 “You win!” L  
 :ln F  
  
 [Finishing the program]  
 $T G2  
L  
 [Wrong input message]  
 “Incorrect sign value, must be one of 0, 1 or 2” :ln F  
L*