Спецификация прототипа

v 1.0

Основные свойства

1) C-подобный синтаксис
2) Программа в одной файле
3) Отсутствие ООП
4) Стандартная библиотека - read, write
5) Система типов - примитивные (int, float)
6) Только автоматический класс памяти

Лексемы

Ключевые слова

Типы данных - void, int, float
Структурные операторы - if, else, while, return

Литералы

Целочисленные: 0, 42, 256
Вещественные: 0.0, 3.14, 256.1024
Строковые: “abc”, “Hello, world!”, “qq\n\tww”

Операторы

Арифметика: +, - (бинарный), *, /, %
Присваивание: =
Логические: ==, !=, <, >, <=, >=, &&, ||, !
Скобки: {}, ()
Служебные: ; ,

Идентификаторы

a..zA..Z_[a..zA..Z_0..9]*
Примеры: i, j, abc, x42, _myVariable, foo, bar_baz_37

Синтаксис

Функция - **type1 name1**([**type2 name2**]*) { [**op**]* }, где:

1) type1, type2 - типы данных
2) name1, name2 - идентификаторы
3) op - оператор

Оператор:

1) if (**cond**) { [**op1**]* } [else { [**op2**]* }], где:
    a) cond - выражение
    b) op1, op2 - операторы
2) while (**cond**) { [**op**]* }, где:
    a) cond - выражение
    b) op - оператор
3) return [**value**];, где:
    a) value - выражение
4) **type name**;, где
    a) type - тип данных
    b) name - идентификатор
5) **value1 = value2**;, где
    a) value1, value2 - выражения
6) **value**;, где
    a) value - выражение

Выражение - набор операндов и операторов, поддающийся вычислению по RPN.

Семантика

1) Точка входа в программу - функция void main().
2) Строгое соответствие типов, операнды должны быть одного типа
3) Передача параметров только по значению
4) Встроенные функции:
    a) write(x) - выводит значение x в зависимости от его типа
    b) read(x) - считывает в x значение, в зависимости от его типа

Байткод

Текстовый формат (для простоты отладки).

Список функций, каждая функция описывает количество необходимых регистров и список команд.

Байткод:

```
N // количество функций
F1 // описание функции
…
FN // описание функции
```

Описание функции:

name // имя функции
X // количество целочисленных регистров
Y // количество вещественных регистров
Z // количество команд
C1 // команда
…
CZ // команда

Команда:

OP A1...An R
OP - идентификатор команды
A1...An - номера регистров-операндов
R - номер регистра-результата

Список команд

{ IADD, ISUB, IMUL, IDIV, IMOD } R0 R1 R2 - целочисленные +, -, *, /, %
{ FADD, FSUB, FMUL, FDIV } R0 R1 R2 - вещественные +, -, *, /
{ LAND, LOR, LNOT } R0 R1 R2 - логические &&, ||, !
MOV R0 R1 - присваивание R0 в R1
{ ILOAD, FLOAD } const R0 - загрузка константы в регистр
{ CMPEQ, CMPNE, CMPBG, CMPLS, CMPBE, CMPGE } R0 R1 R2 - сравнения
GOTO X - безусловный переход на команду №X
IF R0 X - переход на команду №X, если в R0 - не 0
RET - выход из функции
{ WRITE_INT, WRITE_FLOAT, READ_INT, READ_FLOAT } R0 - ввод-вывод
WRITE_STR str - вывод строки str
CALL name R0…RN RR - вызов функции name, R0 … RN - аргументы, RR - результат

Соглашение о вызовах

Аргументы для метода на входе лежат в начале регистров соответствующих типов. Результат складывается в первый регистр соответствующего типа.

Пример

```
int fibonacci(int n) {
	if ((n == 1) || (n == 2)) return 1;
	return fibonacci(n - 1) + fibonacci(n - 2);
}

void main() {
```

```
        int number;
        write(“Hello!\n”);
        while (1) {
                write(“Enter number: ”);
                read(number);
                if (number > 0) {
                        write(“Fibonacci number is: ”);
                        write(fibonacci(number));
                        write(“\n”);
                        return;
                } else {
                        write(“Number should be positive!”);
                }
        }
}

2
fibonacci
5
0
15
/* 00 */ ILOAD 1 1             //
/* 01 */ CMPEQ 0 1 2           // (n == 1)
/* 02 */ ILOAD 2 3             //
/* 03 */ CMPEQ 0 3 4           // (n == 2)
/* 04 */ LOR 2 4 2             // (n == 1) || (n == 2)
/* 05 */ IF 2 7                // if …
/* 06 */ GOTO 9                //
/* 07 */ ILOAD 1 0             // 1
/* 08 */ RET                   // return
/* 09 */ ISUB 0 1 1            // n-1
/* 10 */ CALL fibonacci 1 1    // fibonacci(n-1)
/* 11 */ ISUB 0 3 3            // n-2
/* 12 */ CALL fibonacci 3 3    // fibonacci(n-2)
/* 13 */ IADD 1 3 0            // +
/* 14 */ RET                   // return
```