5. Встроенные алгоритмы

5.1. Общие сведения

Встроенные алгоритмы языка Кумир могут быть использованы в любой программе.

5.2. Текстовое представление чисел

5.2.1. цел_в_лит

Синтаксис:

```
алг лит цел_в_лит(цел число)
```

Возвращает строковое представление параметра *число*.

Скопировать пример

```
алг
нач
цел а
  лит б
  а := 5
  б := цел_в_лит(а)
  вывод б
кон
```

Пример 5.1. цел_в_лит

5.2.2. вещ_в_лит

Синтаксис:

```
алг лит вещ_в_лит(вещ число)
```

Возвращает строковое представление параметра *число*.

Скопировать пример

```
алг
нач
  вещ а
  лит б
  а := 5.9999
  б := вещ_в_лит(а)
  вывод б
кон
```

Пример 5.2. вещ_в_лит

5.2.3. лит_в_вещ

Синтаксис:

```
алг вещ лит_в_вещ(лит строка, рез лог успех)
```

Переводит строку *строка* в вещественное представление. Если *строка* содержит только вещественное число, то в *успех* записываетсяалгоритм возвращает вещественное значение, иначе в *успех* записываетсяи алгоритм возвращает значение `0.0`.

Скопировать пример

```
алг
нач
  лит а
  вещ б
  лог усп
  а := "5.9999"
  б := лит_в_вещ(а, усп)
  вывод б, " ", усп
кон
```

Пример 5.3. лит_в_вещ

5.2.4. лит_в_цел

Синтаксис:

```
алг цел лит_в_цел(лит строка, рез лог успех)
```

Переводит строку *строка* в целочисленное представление. Если *строка* содержит только целое число, то в *успех* записываетсяалгоритм возвращает вещественное значение, иначе в *успех* записываетсяи алгоритм возвращает значение 0.

Скопировать пример

```
алг
нач
  лит а
  цел б
  лог усп
  а := "5"
  б := лит_в_цел(а, усп)
  вывод б, " ", усп
кон
```

Пример 5.4. лит_в_цел

5.2.5. Цел

Синтаксис:

```
алг цел Цел(лит строка, цел по умолчанию)
```

Переводит строку *строка* в целочисленное представление. Если строка не является целым числом, возвращается значение *по умолчанию*.

Скопировать пример

```
алг
нач
  лит неправ = "1а"
  лит прав = "2"
  вывод Цел(неправ, 0), нс
  вывод Цел(прав, 0)
```

```
кон
```

Пример 5.5. Цел

5.2.6. Вещ

Синтаксис:

```
алг вещ Вещ(лит строка, вещ по умолчанию)
```

Переводит строку *строка* в вещественное представление. Если строка не является вещественным числом, возвращается значение *по умолчанию*.

Скопировать пример

```
алг
нач
  лит неправ = "1,4"
  лит прав = "2.5"
  вывод Вещ(неправ, 0.0), нс
  вывод Вещ(прав, 0.0)
кон
```

Пример 5.6. Вещ

5.2.7. Лог

Синтаксис:

```
алг лог Лог(лит строка, лог по умолчанию)
```

Переводит строку *строка* в логическое представление. Если строка не является логическим значением, возвращается значение *по умолчанию*.

Данная функция распознает такие строки:

- "да",
- "1",
- "истина"

как истину, а строки:

- "нет",
- "0",
- "ложь"

как ложь.

Скопировать пример

```
алг
нач
  лит неправ = "плюс"
  лит прав_1 = "да"
  лит прав_2 = "1"
  вывод Лог(неправ, нет), нс
  вывод Лог(прав_1, нет), нс
  вывод Лог(прав_2, нет)
кон
```

Пример 5.7. Лог

5.3. Математика

5.3.1. sqrt

Синтаксис:

```
алг вещ sqrt(вещ x)
```

Возвращает – квадратный корень из *x*.

Значение *x* должно быть ≥ 0.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := sqrt (x)
  вывод "корень квадратный из числа x равен ", x
кон
```

Пример 5.8. sqrt

5.3.2. abs

Синтаксис:

```
алг вещ abs(вещ x)
```

Возвращает абсолютное (неотрицательное) значение вещественного числа *x*.

Скопировать пример

```
вещ а, б
алг
нач
  ввод а,б
  а := а+б
  а := abs(а)
  вывод "Модуль суммы чисел равен ", а
кон
```

Пример 5.9. abs

5.3.3. iabs

Синтаксис:

```
алг цел iabs(цел x)
```

Возвращает абсолютное (неотрицательное) значение целого числа *x*.

Скопировать пример

```
цел а, б
алг
нач
  ввод а,б
  а := iabs(а)
  б := iabs(б)
  вывод а+б
кон
```

Пример 5.10. iabs

5.3.4. sign

Синтаксис:

```
алг цел sign(вещ x)
```

Возвращает абсолютное знак числа *x*:

- -1, если $x < 0$;
- 0, если $x = 0$;
- 1, если $x > 0$.

Скопировать пример

```
цел a, б
алг
нач
  ввод a
  б := sign(a)
  если б=-1
    то вывод a, "<=0"
    иначе
      если б=0
        то вывод a, "=0"
        иначе вывод a, ">=0"
      все
  все
кон
```

Пример 5.11. sign

5.3.5. sin

Синтаксис:

```
алг вещ sin(вещ x)
```

Возвращает синус *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := sin (x)
  вывод "синус угла x равен ", x
```

```
кон
```

Пример 5.12. sin

5.3.6. cos

Синтаксис:

```
алг вещ cos(вещ x)
```

Возвращает косинус *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := cos (x)
  вывод "косинус угла x равен ", x
кон
```

Пример 5.13. cos

Скопировать пример

```
вещ x, y
алг
нач
  вывод "угол x="
  ввод x
  y := 2*sin(x)*cos(x)
  вывод "sin2x = ", y
кон
```

Пример 5.14. cos

5.3.7. tg

Синтаксис:

```
алг вещ tg(вещ x)
```

Возвращает тангенс *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := tg (x)
  вывод "тангенс угла x равен ", x
кон
```

Пример 5.15. tg

5.3.8. ctg

Синтаксис:

```
алг вещ ctg(вещ x)
```

Возвращает котангенс *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := ctg (x)
  вывод "котангенс угла x равен ", x
кон
```

Пример 5.16. ctg

5.3.9. arcsin

Синтаксис:

```
алг вещ arcsin(вещ x)
```

Возвращает арксинус *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := arcsin (x)
  вывод "арксинус числа x равен ", x
кон
```

Пример 5.17. arcsin

5.3.10. arccos

Синтаксис:

```
алг вещ arccos(вещ x)
```

Возвращает арккосинус *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := arccos (x)
  вывод "арккосинус числа x равен ", x
кон
```

Пример 5.18. arccos

5.3.11. arctg

Синтаксис:

```
алг вещ arctg(вещ x)
```

Возвращает арктангенс *x*.

Скопировать пример

```
вещ x
```

```
алг
нач
  ввод x
  x := arctg (x)
  вывод "арктангенс числа x равен ", x
кон
```

Пример 5.19. arctg

5.3.12. arcctg

Синтаксис:

```
алг вещ arcctg(вещ x)
```

Возвращает арккотангенс *x*.

Скопировать пример

```
вещ x
алг
нач
  ввод x
  x := arcctg (x)
  вывод "арккотангенс числа x равен ", x
кон
```

Пример 5.20. arcctg

5.3.13. ln

Синтаксис:

```
алг вещ ln(вещ x)
```

Возвращает натуральный логарифм *x*.

Скопировать пример

```
вещ a,б,c
алг
нач
```

```
  ввод а, б
  с := а+б
  с := ln(с)
  вывод "Натуральный логарифм от суммы чисел ",а," и ",б," равен ",с
кон
```

Пример 5.21. ln

5.3.14. lg

Синтаксис:

```
алг вещ lg(вещ x)
```

Возвращает десятичный логарифм *x*.

Скопировать пример

```
вещ а,б,с
алг
нач
  ввод а, б
  с := а+б
  с := lg(с)
  вывод "Десятичный логарифм от суммы чисел ",а," и ",б," равен ",с
кон
```

Пример 5.22. lg

5.3.15. exp

Синтаксис:

```
алг вещ exp(вещ x)
```

Возвращает e^x.

Скопировать пример

```
вещ x
цел а
алг
```

```
нач
  ввод a
  x := exp(a)
  вывод "число e в степени ", a, " равно ", x
кон
```

Пример 5.23. exp

5.3.16. min

Синтаксис:

```
алг вещ min(вещ x, вещ y)
```

Возвращает наименьшее из вещественных значений *x* или *y*.

5.3.17. max

Синтаксис:

```
алг вещ max(вещ x, вещ y)
```

Возвращает наибольшее из вещественных значений *x* или *y*.

Скопировать пример

```
вещ a,б,c1, c2
алг
нач
  ввод a, б
  c1 := max(a,б)
  c2 := min(a,б)
  вывод c1, нс
  вывод c2, нс
кон
```

Пример 5.24. max

5.3.18. imin

Синтаксис:

```
алг цел imin(цел x, цел y)
```

Возвращает наименьшее из целых значений *x* или *y*.

5.3.19. imax

Синтаксис:

```
алг цел imax(цел x, цел y)
```

Возвращает наибольшее из целых значений *x* или *y*.

Скопировать пример

```
цел а,б,с1, с2
алг
нач
  ввод а, б
  с1 := imax(а,б)
  с2 := imin(а,б)
  вывод с1, нс
  вывод с2, нс
кон
```

Пример 5.25. imax

5.3.20. div

Синтаксис:

```
алг цел div(цел x, цел y)
```

Возвращает частное от деления целых значений *x* на *y*.

5.3.21. mod

Синтаксис:

```
алг цел mod(цел x, цел y)
```

Возвращает остаток от деления целых значений *x* на *y*.

Скопировать пример

```
цел а,б,с1, с2
алг
нач
  ввод а, б
  с1 := imax(а,б)
  с2 := imin(а,б)
  вывод с1, нс
  вывод с2, нс
кон
```

Пример 5.26. mod

5.3.22. int

Синтаксис:

```
алг цел int(вещ x)
```

Возвращает целую часть числа *x* – максимальное целое число, не превосходящее вещественного значения *x*.

Скопировать пример

```
вещ а, б
алг
нач
  ввод а
  б := int(а)
  вывод "Целая часть ", а, " равна ", б
кон
```

Пример 5.27. int

5.3.23. rnd

Синтаксис:

```
алг вещ rnd(вещ x)
```

Случайное число из интервала от 0 до *x*: при последовательных вызовах этой функции получается последовательность случайных чисел, равномерно распределенных на [0, *x*].

Скопировать пример

```
алг Построение последовательности случайных вещественных чисел
нач
  вещ таб a [1:10]
  цел л
  вещ б
  ввод б
  нц для л от 1 до 10
    a[л] := rnd(б)
  кц
  нц для л от 1 до 10
    вывод a[л], " "
  кц
кон
```

Пример 5.28. rnd

5.3.24. rand

Синтаксис:

```
алг вещ rand(вещ x, вещ y)
```

Случайное число из интервала от *x* до *y*: при последовательных вызовах этой функции получается последовательность случайных чисел, равномерно распределенных на [*x*, *y*].

5.3.25. irnd

Синтаксис:

```
алг цел irnd(цел x)
```

Случайное целое число из интервала от 0 до *x*: при последовательных вызовах этой функции получается последовательность случайных чисел, равномерно распределенных на [0, *x*].

5.3.26. irand

Синтаксис:

```
алг цел irnd(цел x, цел y)
```

Случайное целое число из интервала от *x* до *y*: при последовательных вызовах этой функции получается последовательность случайных чисел, равномерно распределенных на [*x*, *y*].

5.3.27. МАКСЦЕЛ

Синтаксис:

```
алг цел МАКСЦЕЛ
```

Самое большое целое число, которое можно использовать в языке Кумир: . Величина этого числа определяется способом представления чисел в современных компьютерах и примерно одинаковая в большинстве современных языков программирования.

5.3.28. МАКСВЕЩ

Синтаксис:

```
алг вещ МАКСВЕЩ
```

Самое большое вещественное число, которое можно использовать в языке Кумир: значение,

несколько меньшее чем .

5.4. Обработка текста

5.4.1. длин

Синтаксис:

```
алг цел длин(лит строка)
```

Возвращает количество символов в строке *строка*.

Скопировать пример

```
алг
нач
  лит а
  цел ц
  вывод "введите строку"
  ввод а
```

```
  ц := длин(а)
  вывод ц
кон
```

Пример 5.29. длин

5.4.2. код

Синтаксис:

```
алг цел код(сим с)
```

Возвращает порядковый номер символа в таблице СР-1251.

Скопировать пример

```
алг
нач
  сим а
  цел ц
  вывод "введите символ "
  ввод а
  ц := код(а)
  вывод ц
кон
```

Пример 5.30. код

5.4.3. юникод

Синтаксис:

```
алг цел юникод(сим с)
```

Возвращает порядковый номер символа в таблице Unicode.

5.4.4. символ

Синтаксис:

```
алг сим символ(цел n)
```

Возвращает символ, имеющий в таблице CP-1251 порядковый номер *n*.

Скопировать пример

```
алг
нач
  сим а
  цел ц
  вывод "введите число "
  ввод ц
  а := символ(ц)
  вывод а
кон
```

Пример 5.31. символ

5.4.5. юнисимвол

Синтаксис:

```
алг сим юнисимвол(цел n)
```

Возвращает символ, имеющий в таблице Unicode порядковый номер *n*.

5.5. Алгоритмы обработки строк

5.5.1. верхний регистр

Синтаксис:

```
алг лит верний регистр(лит строка)
```

Возвращает строку, все символы которой приведены к верхнему регистру символов.

5.5.2. нижний регистр

Синтаксис:

```
алг лит нижний регистр(лит строка)
```

Возвращает строку, все символы которой приведены к нижнему регистру символов.

5.5.3. позиция

Синтаксис:

```
алг цел позиция(лит строка, лит фрагмент)
```

или сокращенный вариант:

Синтаксис:

```
алг цел поз(лит строка, лит фрагмент)
```

Возвращает позицию первого символа подстроки *фрагмент* в строке *строка*. Если фрагмент не найден, то возвращает значение `0`.

5.5.4. позиция после

Синтаксис:

```
алг цел позиция после(цел начало, лит строка, лит фрагмент)
```

или сокращенный вариант:

Синтаксис:

```
алг цел поз после(цел начало, лит строка, лит фрагмент)
```

Возвращает позицию первого символа подстроки *фрагмент* в строке *строка*, при этом поиск подстроки начинается с позиции *начало*. Если фрагмент не найден, то возвращает значение `0`.

5.5.5. вставить

Синтаксис:

```
алг вставить(лит фрагмент, аргрез лит строка, арг цел начало)
```

В строку *строка* вставляет строку *фрагмент*, начиная с символа с номером *начало*.

Если *начало* = длин (*строка*)+1, то фрагмент добавляется в конец строки.

Если *начало* < 1 или *начало* > длин (*строка*)+1, то возникает ошибка выполнения.

5.5.6. заменить

Синтаксис:

```
алг заменить(аргрез лит строка, арг лит старый фрагмент, арг лит
новый фрагмент, арг лог каждый)
```

Меняет в строке *строка* подстроку *старый фрагмент* на *новый фрагмент*.

Если *каждый* = **да**, то меняет все вхождения подстроки.

Если *каждый* = **нет**, то меняет только первое вхождение подстроки.

5.5.7. удалить

Синтаксис:

```
алг удалить(аргрез лит строка, арг цел начало, арг цел количество)
```

Удаляет *количество* символов из строки *строка* начиная с позиции *начало*.

Если *начало* + *количество* > длин (*строка*)+1, то удаляется текст до конца строки.

Если *начало* < 1 или *начало* > длин (*строка*)+1, то возникает ошибка выполнения.

5.6. Алгоритмы работы с файлами

5.6.1. открыть на чтение

Синтаксис:

```
алг файл открыть на чтение(лит имя файла)
```

Открывает текстовый файл с именем *имя файла* для чтения.

Имя файла может быть как абсолютным, так и относительным, – в этом случае поиск файла осуществляется относительно рабочего каталога.

Если файл не существует, или отсутствуют права на чтение файла, то возникает ошибка выполнения.

Если файл ранее уже был открыт, то возникает ошибка выполнения.

5.6.2. открыть на запись

Синтаксис:

```
алг файл открыть на запись(лит имя файла)
```

Открывает текстовый файл с именем *имя файла* для записи.

Имя файла может быть как абсолютным, так и относительным, – в этом случае поиск файла осуществляется относительно рабочего каталога.

Если файл не существует, то он будет создан, иначе – содержимое существующего файла будет очищено.

Если отсутствуют права на создание нового файла или перезапись существующего файла, то возникает ошибка выполнения.

Если файл ранее уже был открыт, то возникает ошибка выполнения.

5.6.3. открыть на добавление

Синтаксис:

```
алг файл открыть на добавление(лит имя файла)
```

Открывает текстовый файл с именем *имя файла* для записи.

Имя файла может быть как абсолютным, так и относительным, – в этом случае поиск файла осуществляется относительно рабочего каталога.

Если файл не существует, то он будет создан, иначе – содержимое существующего файла останется неизменным, а новые данные будут дописаны в конец.

Если отсутствуют права на создание нового файла или перезапись существующего файла, то возникает ошибка выполнения.

Если файл ранее уже был открыт, то возникает ошибка выполнения.

5.6.4. закрыть

Синтаксис:

```
алг закрыть(файл имя файла)
```

Закрывает ранее открытый для чтения или записи текстовый файл *имя файла*.

После закрытия файл становится доступным для повторного открытия.

Все файлы должны быть закрыты до окончания работы программы, иначе возникнет ошибка выполнения при завершении выполнения программы.

5.6.5. начать чтение

Синтаксис:

```
алг начать чтение(файл имя файла)
```

Сбрасывает указатель чтения файла *имя файла* на его начало.

5.6.6. конец файла

Синтаксис:

```
алг лог конец файла(файл имя файла)
```

Возвращает **да**, если указатель находится в самом конце файла и чтение из него невозможно.

5.6.7. есть данные

Синтаксис:

```
алг лог есть данные(файл имя файла)
```

Возвращает **да**, если после текущей позиции чтения в файле *имя файла* есть хотя бы один видимый символ.

Если алгоритм `есть данные` возвращает значение **нет**, а указатель не находится в конце файла (то есть алгоритм `конец файла` также возвращает **нет**), то из файла возможно чтение только символа или строки.

5.6.8. установить кодировку

Синтаксис:

```
алг установить кодировку(лит имя кодировки)
```

Устанавливает кодировку текстовых файлов, которые будут открыты на чтение или запись после вызова данного алгоритма.

Допустимые имена кодировок:

- `"CP-1251"` или `"Windows-1251"` или `"Windows"` – однобайтная кодировка текста, используемая в системе семейства Windows для кириллических символов;
- `"CP-866"` или `"IBM-866"` или `"DOS"` – однобайтная кодировка текста, используемая в консольных программах системы семейства Windows для кириллических символов;
- `"KOI8-R"` или `"KOI8"` или `"КОИ8"` или `"КОИ8-Р"` – однобайтная кодировка текста КОИ-8, используемая в старых системах семейства UNIX;
- `"UTF-8"` или `"UTF"` или `"Linux"` – кодировка текста с переменным количеством байт на символ, используемая в современных системах семейства UNIX (включая MacOS X и Linux).

Допускаются имена кодировок в любом регистре и с пропущенным символом `'-'` (дефис) в имени.

Неверное имя кодировки приводит к ошибке выполнения.

5.6.9. можно открыть на чтение

Синтаксис:

```
алг лог можно открыть на чтение(лит имя файла)
```

Возвращает **да**, если выполняются условия:

- файл с указанным именем существует;
- права доступа к файлу достаточны для его чтения.

Вызов этого алгоритма не приводит к ошибке выполнения.

5.6.10. можно открыть на запись

Синтаксис:

```
алг лог можно открыть на запись(лит имя файла)
```

Возвращает **да**, если выполняются условия:

- файл с указанным именем существует;
- права доступа к файлу достаточны для его записи.

Вызов этого алгоритма не приводит к ошибке выполнения.

5.6.11. существует

Синтаксис:

```
алг лог существует(лит имя)
```

Возвращает **да**, если с заданным именем существует файл или каталог.

5.6.12. является каталогом

Синтаксис:

```
алг лог является каталогом(лит имя)
```

Возвращает **да**, существует каталог с заданным именем.

5.6.13. создать каталог

Синтаксис:

```
алг лог создать каталог(лит имя каталога)
```

Имя каталога может быть как абсолютным, так и относительным, – в этом случае каталог создается относительно рабочего каталога.

Алгоритм возвращает **да**, если каталог был успешно создан.

5.6.14. удалить_файл

Синтаксис:

```
алг лог удалить_файл(лит имя файла)
```

Удаляет файл с заданным именем без возможности восстановления.

Имя файла может быть как абсолютным, так и относительным, – в этом случае поиск файла осуществляется относительно рабочего каталога.

Алгоритм возвращает **да**, если файл был успешно удален.

Внимание: имя алгоритма содержит символ подчеркивания вместо пробела.

Предупреждение: будьте осторожны в применении этого алгоритма!

5.6.15. удалить_каталог

Синтаксис:

```
алг лог удалить_каталог(лит имя каталога)
```

Удаляет пустой каталог с заданным именем без возможности восстановления.

Имя каталога может быть как абсолютным, так и относительным, – в этом случае каталог удаляется относительно рабочего каталога.

Алгоритм возвращает да, если каталог был успешно удален.

Внимание: имя алгоритма содержит символ подчеркивания вместо пробела.

Предупреждение: будьте осторожны в применении этого алгоритма!

5.6.16. полный путь

Синтаксис:

```
алг лит полный путь(лит имя)
```

Возвращает полное (абсолютное) имя файла или каталога без проверки его существования.

5.6.17. РАБОЧИЙ КАТАЛОГ

Синтаксис:

```
алг лит РАБОЧИЙ КАТАЛОГ
```

Возвращает полное (абсолютное) имя текущего рабочего каталога.

1. При запуске Кумир-программы из консоли, рабочим каталогом является тот, который был выбран в консоли до запуска.

2. При запуске Кумир-программы из среды Кумир, рабочий каталог определяется следующим образом:

- Если программа сохранена в файл, то рабочий каталог – это каталог, в котором сохранена программа.
- Если программа не сохранена, то рабочий каталог – это каталог, установленный соответствующим пунктом меню Программа главного окна.

5.6.18. КАТАЛОГ ПРОГРАММЫ

Синтаксис:

```
алг лит КАТАЛОГ ПРОГРАММЫ
```

Возвращает полное (абсолютное) имя каталога, в котором находится выполняемая Кумир-программа.

Если программа не сохранена в файл, то возвращается значение "./".

5.6.19. НАЗНАЧИТЬ ВВОД

Синтаксис:

```
алг НАЗНАЧИТЬ ВВОД(лит имя файла)
```

- Если *имя файла* не пустое и файл доступен для чтения, то устанавливает его в качестве источника данных, используемых оператором **ввод** без явного указания файловой переменной.
- Если *имя файла* – пустая строка, то восстанавливает исходное поведение (ввод с клавиатуры) оператора **ввод**.
- Если указанный файл не существует или не доступен для чтения, то возникает ошибка выполнения.

5.6.20. НАЗНАЧИТЬ ВЫВОД

Синтаксис:

```
алг НАЗНАЧИТЬ ВЫВОД(лит имя файла)
```

- Если *имя файла* не пустое и файл доступен для записи, то устанавливает его в качестве приемника текста, используемого оператором **вывод** без явного указания файловой переменной.
- Если *имя файла* – пустая строка, то восстанавливает исходное поведение (вывод на экран) оператора **вывод**.
- Если указанный файл не существует или не доступен для записи, то возникает ошибка выполнения.

5.6.21. консоль

Синтаксис:

```
алг файл консоль
```

Возвращает псевдо-файл, связанный с терминалом, который обладает следующими свойствами:

- файл всегда открыт для чтения и записи;
- закрытие файла не требуется, а попытка закрытия приводит к ошибке выполнения;
- файл не имеет имени;

запись в файл приводит к отображению текста на экран;

- чтение из файла приводит к запросу ввода с клавиатуры.

Над полученным с помощью алгоритма консоль значением величины типа **файл** можно выполнять стандартные операции: присваивание другим величинам и проверку на равенство с другой величиной типа **файл**.

5.7. Системные функции

5.7.1. ждать

Синтаксис:

```
алг ждать(цел x)
```

Приостанавливает работу программы на *x* миллисекунд. Одна миллисекунда равна 1/1000 секунды.

5.7.2. время

Синтаксис:

```
алг цел время
```

Возвращает текущее время в виде количества миллисекунд, прошедших с начала суток по местному времени.

Скопировать пример

```
алг
нач
  цел мсек, сек, мин, час
  мсек:=время
  час:=div(мсек,1000*60*60)
  мин:=div(мсек,1000*60)-час*60
  сек:=div(мсек,1000)-час*60*60-мин*60
  вывод час, ":", мин, ":", сек
кон
```

Пример 5.32. Отображение текущего времени