Интервальный калькулятор-интерпретатор с синтаксисом ассемблера (интервальный калькулятор)

Синопсис

Интервальный калькулятор предназначен для выполнения математических операций над интервальными величинами (например, конструкторскими или технологическими размерами).

Калькулятор позволяет выполнять арифметические операции, вычислять тригонометрические и обратные тригонометрические функции.

Вычисления производятся по программе, заданной пользователем.

Пользовательская программа читается калькулятором из стандартного ввода, результат подаётся на стандартный вывод.

Имеется возможность управления потоком выполнения пользовательской программы, в том числе вызов пользовательских процедур.

Синтаксис пользовательской программы

Синтаксис программы на отработку ассемблер-подобный.

Программа выполняется построчно. Строка строится по шаблону:

```
метка:   команда   аргументы    ;комментарий
```

Общая длина строки не должна превышать 255 символа в UNIX-системах и 254 символа в MS Windows.

Метка служит для возможности перехода (прыжка) на строку, помеченную этой меткой. Представляет собой строку из букв латинского алфавита, цифр, знаков "_", "$", "@", "." длиной до 32 символов. После метки обязательно ставится двоеточие. Метка может отсутствовать.

Команда - указание на выполнение определённого действия: поместить в аргумент-переменную какое-либо значение, выполнить математическую операцию с переменной, перейти на заданную метку, вывести аргумент в стандартный вывод.

Аргументы - список аргументов команды, разделённых запятыми. Аргументами могут быть: переменные, интервальные числа, названия меток. Аргументы могут отсутствовать (в зависимости от команды).

Комментарий - игнорируемая калькулятором часть строки, которая начинается с символа "точка с запятой" и кончается символом конца строки. Комментарий может отсутствовать.

Переменные представляют собой выражения вида rN, где N любое натуральное число. Например, r1, r23, r456. Переменные обозначенные как, например, r2 и r02 представляют одну и ту же переменную.

Интервальное число может быть задано несколькими способами:

1) Одним конкретным числом. Например:

```
126                ;число 126,0...126,0
```

Любое конкретное число внутренне преобразуется к интервальному числу с равными верхним и нижним пределами.

2) Верхним и нижним пределами, разделённых пробелами. Порядок указания верхнего и нижнего предела не важен. Например:

```
12.8  13.2       ;число 12,8...13,2
120.8  120       ;число 120,0...120,8
```

3) Номиналом и отклонениями: сначала указывается номинал, затем в любом порядке отклонения. Например:

```
13 -0.2  0.2     ;число 13±0,2
120  0.8  0      ;число 120^+0,8
```

Целая и дробная части в записи чисел отделяются точкой.

Регистр в написании команд, меток и переменных важен.

Система команд

mov приёмник, источник
Копирование значения из источника в переменную-приёмник. Приёмник - любая переменная. Источник - переменная или значение. Примеры:

```
        mov  r1, 21.5              ;r1 = 21,5...21,5
        mov  r1, 22  -0.4  -0.6    ;r1 = 22 -0,4 -0,6
        mov  r2, 21.4  21.6        ;r2 = 21,4...21,6
        mov  r1, r2                ;r1 = r2
```

xchg переменная1, переменная2
Обмен значений переменных.
Пример:

```
        mov  r1, 10                ;r1 = 10,0...10,0
        mov  r2, 8  8.5            ;r2 = 8,0...8,5
        xchg  r1, r2
```

В результате r1 будет содержать значение 8,0...8,5, а переменная r2 значение 10,0...10,0.
Аналогичный эффект можно получить используя промежуточную переменную, например:

```
        mov  r3, r1
        mov  r1, r2
        mov  r2, r3
```

add слагаемое1, слагаемое2
Сложение. Сумма сохраняется в переменной слагаемое1.
Слагаемое1 - любая переменная.
Слагаемое2 - переменная или значение.
Пример:

```
        mov  r1, 10                ;r1 = 10,0...10,0
        add  r1, 15.6  0.2  -0.2   ;r1 = r1 + 15,6±0,2
```

В данном примере r1 будет содержать значение 25,6±0,2.

sub уменьшаемое, вычитаемое
Вычитание. Разность сохраняется в уменьшаемом.
Уменьшаемое - любая переменная.
Вычитаемое - переменная или значение.
Пример:

```
        mov  r1, 10 12             ;r1 = 10,0...12,0
        mov  r2, 3 3.2             ;r1 = 3,0...3,2
        sub  r1, r2                ;r1 = r1 - r2
```

В данном примере r1 будет содержать значение 6,8...9,0.

mul множитель1, множитель2
Умножение. Произведение сохраняется в переменной множитель1.
Множитель1 - любая переменная.
Множитель2 - переменная или значение.
Пример:

```
        mov  r1,  10  12          ;r1 = 10,0...12,0
        mul  r1,  2.3             ;r1 = r1 * 2.3
```

В данном примере r1 будет содержать значение 23,0...27,6.

div делимое, делитель
Деление. Частное сохраняется в переменной делимое.
Делимое - любая переменная.
Делитель - переменная или значение.
Пример:

```
        mov  r1,  10  12          ;r1 = 10,0...12,0
        div  r1,  5 -0.3  0       ;r1 = r1 / 5-0,3
```

В данном примере r1 будет содержать значение 2,0...2,553.

neg переменная
Изменение знака (отражение интервала относительно нуля).
Пример:

```
        mov  r1,  10  12
        neg  r1
```

В данном примере r1 будет равно -12,0...-10,0.

sqrt переменная
Вычисление корня квадратного. Результат сохраняетсяя в переменной-аргументе.
Пример:

```
        mov  r1,  9  9.4          ;r1 = 9,0...9,4
        sqrt r1                   ;r1 = √r1
```

В данном примере r1 будет содержать значение 3,0...3,066.

inv переменная
Обращение переменной (вычисление дроби 1/x). Результат сохраняется в переменной-аргументе.
Пример:

```
        mov  r1,  0.5  0.6        ;r1 = 0,5...0,6
        inv  r1                   ;r1 = 1 / r1
```

В данном примере r1 будет содержать значение 1,667...2,0.

inc переменная
dec переменная
Увеличение (inc) или уменьшение (dec) значения переменной на 1 (точнее, сдвиг границ интервала на 1 вправо или влево). Результат сохраняется в переменной-аргументе. Пример:

```
        mov  r1, 10  12          ;r1 = 10,0...12,0
        inc  r1                  ;r1 = r1 + 1,0
        mov  r2, r1              ;r2 = r1
        dec  r2                  ;r2 = r2 - 1,0
```

В данном примере r1 будет содержать значение 11,0...13,0, а r2 значение 10,0...12,0.

sin переменная
cos переменная
tan переменная
cot переменная
asin переменная
acos переменная
atan переменная
acot переменная
Вычисление прямых и обратных тригонометрических функций: синус (sin), косинус (cos), тангенс (tan), котангенс (cot), арксинус (asin), арккосинус (acos), арктангенс (atan), арккотангенс (acot). Аргумент для прямых функций задаётся в градусах. Результат сохраняется в переменной-аргументе. Пример:

```
        mov  r1, 30 -0.1 0.1     ;r1 = 30±0,1
        cos  r1                  ;r1 = cos r1
        mov  r2, r1              ;r2 = r1
        asin r2                  ;r2 = arc sin r2
```

В данном примере r1 будет содержать значение 0,865...0,867, а r2 значение 59,9...60,1.

jmp метка
Безусловный переход на заданную метку.
Пример:

```
        mov r1, 10   ;r1 = 10,0...10,0
        jmp  label   ;Переход на метку label
        inc r1       ;r1 = r1 + 1 - код до метки label
                     ;              выполнен не будет
label:  dec r1       ;r1 = r1 - 1 - код после - будет
```

В данном примере r1 будет содержать число 9,0...9,0.

cmp аргумент1, аргумент2
Сравнение двух аргументов. Аргументы могут быть любого вида (переменные и значения). Результат сравнения сохраняется во внутренней переменной и служит для изменения потока выполнения программы - в зависимости от результата сравнения может быть выполнен переход на определённую метку (при помощи команд je, jne, jgt, jge, jlt, jle, jin, jout).

je метка
jne метка
jgt метка
jge метка
jlt метка
jle метка
jin метка
jout метка
Команды условного перехода. Выполняется или не выполняется переход на заданную метку в зависимости от результата сравнения, выполненного командой cmp. Команды учитывают взаимное положение сравниваемых интервалов (см. таблицу ниже).
Пример:

```
        mov  r1,  11.5  12.6     ;r1 = 11,5...12,6
        cmp  r1,  11.0  12.0
        jge  label       ;Переход, если ≥
        inc r1           ;Код здесь выполнен не будет
label:  mov  r2,  r1     ;После метки - будет
```

В данном примере r1 и r2 будут содержать 11,5...12,6.

---

je (Jump if Equal - переход, если равны)

переход, если сравниваемые интервалы совпадают:

---

jne (Jump if NOT Equal - переход, если НЕ равны)

переход, если интервалы не пересекаются:

---

jgt (Jump if Greater Than - переход, если больше)

переход, если нижний предел первого интервала (A) больше верхнего предела второго (B)
(т.е. интервал A на числовой оси располагается правее интервала B):

---

jge (Jump if Greater or Equal - переход, если больше или равно)

переход, если верхний предел первого интервала (A) больше верхнего предела второго интервала (B) и, при этом, нижний предел первого (A) лежит в пределах второго (B):

---

jlt (Jump if Lesser Than - переход, если меньше)

переход, если верхний предел первого интервала (A) меньше нижнего предела второго (B)
(т.е. интервал A на числовой оси располагается левее интервала B):

---

jle (Jump if Lesser or Equal - переход, если меньше или равно)

переход, если нижний предел первого интервала (A) меньше нижнего предела второго интервала (B) и, при этом, верхний предел первого (A) лежит в пределах второго (B):

---

jin (Jump if Inside - переход, если внутри)

переход, если первый интервал содержится во втором:

---

jout (Jump if Outside - переход, если снаружи)

переход, если первый интервал содержит второй:

---

nop
Отсутствие операции (холостой ход).

hlt
Завершить программу.

out аргумент
Вывести аргумент в стандартный вывод.
Аргумент - переменная или строка, заключённая в двойные кавычки.
Пример:

```
        mov  r1,  50.2
        out  "Размер по чертежу: 49,6...50,0"
        out  "Фактический диаметр: "  ;Вывести строку
        out  r1                       ;Вывести переменную
```

В данном примере программа выведет:

```
        Размер по чертежу: 49,6...50,0
        Фактический диаметр:
        50.2
```

free
Освобождение оперативной памяти
(забываются значения используемых переменных, выполненных ранее инструкций, выводимый командой out текст, освобождается стек с адресами возврата из подпрограмм).

call метка
ret
Операторы вызова подпрограммы (call) и возврата из подпрограммы (ret). Подпрограммой можно считать любой фрагмент кода, отмеченный меткой. При отработке команды call выполняется (безусловный) переход на заданную метку. Если, далее, после перехода, встречается команда ret, то выполняется (безусловный) переход в точку вызова - на команду, следующую за call.
Из одной подпрограммы можно вызывать вторую, из второй - третью и т.д. Количество вложенных вызовов не ограничено.

Пример:

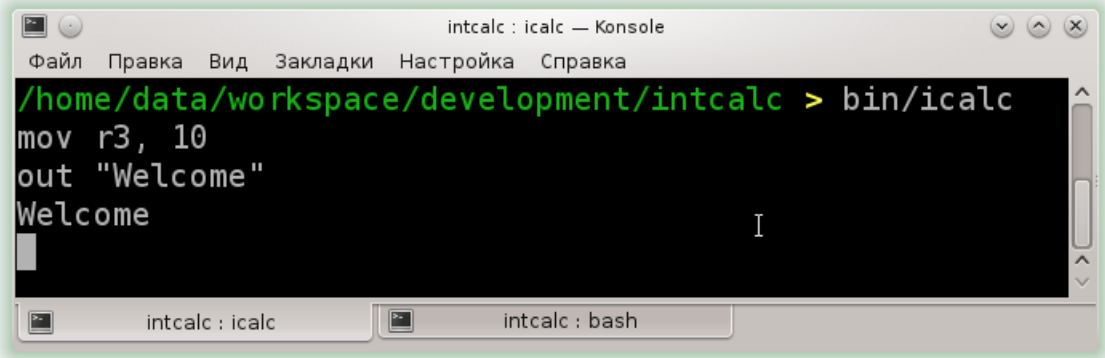
```
     mov  r1, 49.6  50        ;Размер по чертежу
     mov  r2, 50.2            ;Размер факт., дет №1
     call print_quality_conc  ;Вызов print_quality_conc
     mov  r2, 49.8            ;Размер факт., дет №2
     call print_quality_conc  ;Вызов print_quality_conc
     mov  r2, 50              ;Размер факт., дет №3
     call print_quality_conc  ;Вызов print_quality_conc
     hlt                      ;Завершение работы

;Вывод сообщения о годности
;Вход: r1 - размер по чертежу
;      r2 - размер фактический
print_quality_conc:
     cmp  r2, r1              ;Сравнить
     jin  part_good           ;Если в допуске - на метку part_good
     out  "Деталь с дефектом"
     ret                      ;Выход из подпрограммы
part_good:
     out "Деталь годная"
     ret                      ;Выход из подпрограммы
```

В данном примере рассматриваются три детали с условными фактическими размерами 50.2, 49.8, 50.0 и проверяется их соответствие чертёжному размеру 49,6...50,0. Проверка осуществляется в подпрограмме print_quality_conc, которая вызывается для каждой детали. В результате программа выведет следующий текст:

```
Деталь с дефектом
Деталь годная
Деталь годная
```

Варианты запуска калькулятора

Программу для вычисления калькулятору можно подать следующими способами:
1) интерактивно вводить команды, запустив калькулятор
Пример:



Интерпретатор, запущенный в терминале Konsole на Linux

2) перенаправить поток вывода в калькулятор. Для этого используется командная строка.
Пример:

```
icalc < anyprog
```

Здесь anyprog - файл с пользовательской программой (простой текст).

Также удобно перенаправить вывод калькулятора в файл:

```
icalc < anyprog > result.txt
```

В результате последнего примера будет создан файл result.txt, содержащий весь вывод из калькулятора.

Калькулятор завершает свою работу, встретив команду hlt или наткнувшись на символ EOF (конца файла).

Поддерживаемые параметры командной строки:

-h, --help

Получение краткой справки по использованию.

-v, --version

Получение информации о версии.

-W

Выход из программы при возникновении ошибки (для удобства локализации ошибки при выполнении программы из файла).

Сообщения об ошибках

Сообщения об ошибках выводятся в поток стандартного вывода ошибок.

Тип сообщения	Сообщение	Описание
Ошибка	Error: RET found, but CALL was not present	Встретилась команда RET до того, как была выполнена команда CALL (стек адресов возврата пуст)
Ошибка	Error: Unknown argset	Внутренняя ошибка
Ошибка	Error: Unknown command	Неизвестная команда. Проверьте правильность написания команды или наличие двоеточия в случае использования метки
Ошибка	Error: No arguments for command	Использованная команда подразумевает наличие аргументов
Ошибка	Error: Invalid arguments	Использованная команда подразумевает наличие аргументов определённого типа (см. описание команд). Например, требуется только переменная, или переменная, затем число, или две переменные и т.д.
Ошибка	Error: Max iterations reached	Внутренняя ошибка
Предупреждение	Warning: Label name too long	Длина имени метки превышает оговорённый лимит
Предупреждение	Warning: Surplus arguments after command	Использованная команда не подразумевает наличие аргументов, но они присутствуют. Уберите ненужные аргументы или поправьте команду.

При возникновении "внутренних ошибок" или случаев неожиданного поведения калькулятора, пожалуйста, сообщите авторам условия их появления.

Практические примеры

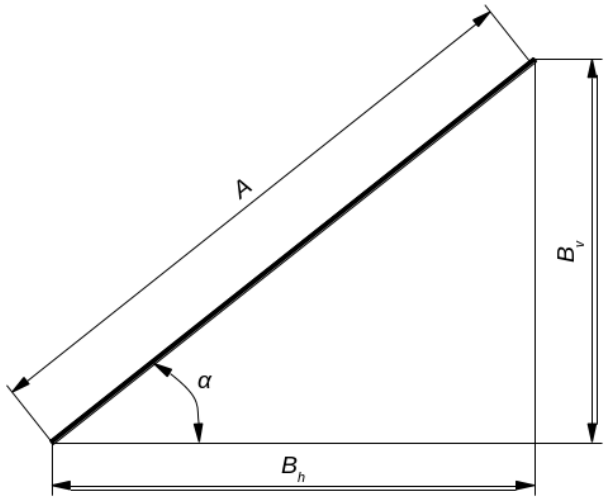
1. Расчёт проекций непараллельного звена размерной цепи (прямоугольный треугольник).

Требуется определить проекции размера A, стоящего под углом α к основным направлениям.

Проекции определяются формулами:

$B_h = A \cos \alpha$, $B_v = A \sin \alpha$.

Пусть A = 20±0,1 и α = 30°±0,05°.



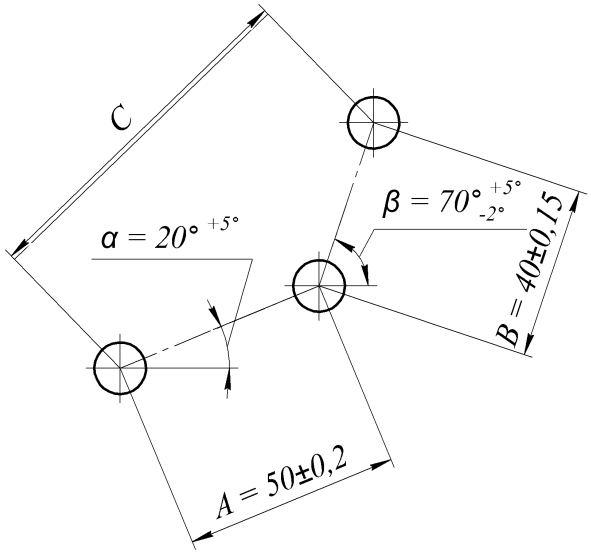

```
task1:    mov r1, 20  -0.1  0.1        ;r1 - размер A
          mov r2, 30  -0.05  0.05      ;r2 - размер α
          call tria1
          ;Выводим результат:
          out "Bh = "
          out r3
          out "Bv = "
          out r4
          hlt

;Вычисление Bh и Bv
;Вход: r1 - размер A
;      r2 - размер α
;Выход: r3 - размер Bh
;       r4 - размер Bv
tria1:    mov r3, r2                   ;r3 = α
          cos r3                       ;r3 = cos (α)
          mul r3, r1                   ;r3 = cos (α) * A
          mov r4, r2                   ;r4 = α
          sin r4                       ;r4 = sin (α)
          mul r4, r1                   ;r4 = sin (α) * A
          ret
```

Программа выдаст следующий результат:

```
Bh =
17.225  17.416
Bv =
9.935  10.065
```

2. Расчёт непараллельного замыкающего звена (непрямоугольный треугольник)

Рассчитаем звено C согласно данным на схеме:



По теореме косинусов:

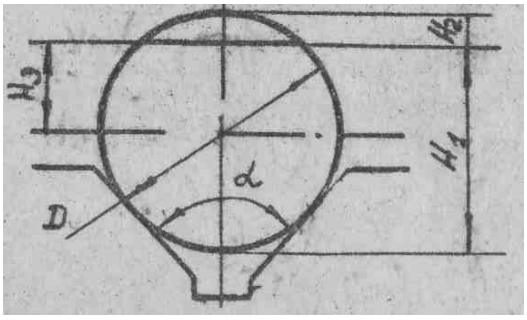
$$C = \sqrt{(A^2 + B^2 - 2AB \cos(\alpha-\beta+180))}$$

```
task2:    ;Исходные данные:
          mov  r1,  20  0  +5          ;r1 - размер α
          mov  r2,  50  -0.2 +0.2      ;r2 - размер A
          mov  r10, 70  -2  +5         ;r10 - размер β
          mov  r20, 40  -0.15  +0.15   ;r20 - размер B
          ;Вычисляем:
          neg  r10
          add  r10,  180      ;180 - β
          add  r10,  r1       ;α + 180 - β
          cos  r10            ;cos (α + 180 - β)
          mul  r10, r2        ;cos (α + 180 - β)*A
          mul  r10, r20       ;cos (α + 180 - β)*A*B
          mul  r10, 2         ;cos (α + 180 - β)*A*B*2
          neg  r10            ;-cos (α + 180 - β)*A*B*2
          mul  r20, r20       ;B*B
          mul  r2,  r2        ;A*A
          add  r10, r20       ;cos (α + 180 - β)*A*B*2 + B*B
          add  r10, r2        ;cos (α + 180 - β)*A*B*2 + B*B + A*A
          sqrt r10            ;√r10
          ;Резульат в r10, выводим:
          out  r10
          hlt
```

Программа выдаст следующий результат:

```
          79.653  84.144
```

3. Расчёт погрешности базирования вала в призме.



Рассчитаем погрешность базирования (при выполнении размера H_1) по формуле:

$$E_{H1}=\frac{\delta D}{2}\left(\frac{1}{\sin\alpha/2}-1\right),$$

где δD - допуск диаметра шипа вала, α - угол призмы.

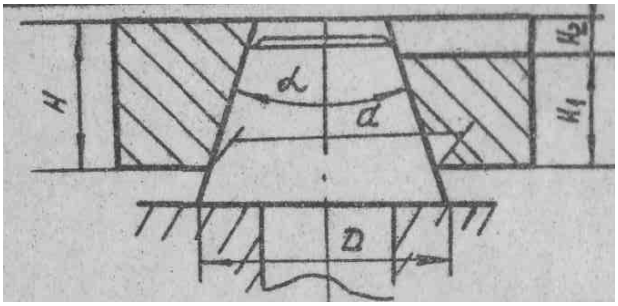
Пусть δD = 0,43 мм, α = 90°±1°.

```
task3:    ;Вводим исходные данные:
          mov       r1,  90 -1 1        ;r1 = α = 90±1
          mov       r2, 0.43            ;r2 = δD = 0,43
          ;Вычисляем:
          div       r1, 2               ;α/2
          sin       r1                  ;sin(α/2)
          inv       r1                  ;1/sin(α/2)
          dec       r1                  ;1/sin(α/2) - 1
          div       r2, 2               ;δD/2
          mul       r1, r2              ;δD/2 * (1/sin(α/2)-1)
          ;Результат в r1, выводим:
          out       "E_H1 = "
          out       r1
          hlt
```

Программа выдаст следующий результат:

```
E_H1 =
0.086  0.092
```

4. Погрешность базирования на коническом пальце.



Требуется определить погрешность базирования при выполнении размера H_2:

$$E_{H2}=\frac{\delta D+\delta d}{2tg\frac{\alpha}{2}}+\delta H\ ,$$

где δD - допуск диаметра основания конуса, δd - допуск диаметра отверстия на входе, α - угол конуса, δH - допуск высоты детали.

Пусть: δD = 0,27, δd = 0,43, α = 30±1°, δH = 0,74.

```
task4:    ;Исходные данные:
          mov r1, 0.27              ;r1 = δD
          mov r2, 0.43              ;r2 = δd
          mov r3, 30 +1 -1          ;r3 = α
          mov r4, 0.74              ;r4 = δH
          ;Вычисляем:
          div r3, 2                 ;α/2
          tan r3                    ;tg (α/2)
          mul r3, 2                 ;2 tg (α/2)
          add r1, r2                ;δD + δd
          div r1, r3                ;(δD + δd) / (2 tg (α/2))
          add r1, r4                ;(δD + δd) / (2 tg (α/2)) + δH
          ;Результат в r1, выводим:
          out r1
          hlt
```

Программа выдаст следующий результат:

```
    2.002  2.093
```

5. Вычисление экспоненты от интервала.

В текущем наборе команд интерпретатора отсутствует команда вычисления экспоненты. Её значение можно получить, просуммировав ряд:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Ограничимся 1000 первыми членами ряда.

```
task5:    mov r1,  1.0  1.6
          call expon          ;r2 = e^r1
          out r2
          hlt

;Экспонента e^x
;Вход:  r1
;Выход: r2
;Используются (значения будут потеряны): r3, r4, r5, r6
expon:    mov r2, 1           ;Сумма
          mov r3, 1           ;Счётчик цикла
          mov r4, 1           ;Числитель текущей дроби
          mov r5, 1           ;Знаменатель текущей дроби
loop:     mul r4, r1
          mul r5, r3
          mov r6, r4
          div r6, r5
          add r2, r6
          inc r3
          cmp r3, 1000        ;Первые 1000 членов
          jlt loop
          ret
```

Для интервала 1,0...1,6 программа выдаст следующий результат:

2.718 4.953

Условия, гарантии, авторские права

Интервальный калькулятор - свободное программное обеспечение. Вы можете использовать и распространять его без каких-либо ограничений. Вы также можете изменять программу при условии сохранения информации об авторах и внесённых изменениях.

Данное программное обеспечение поставляется "как есть" (БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ), вы используете его на собственный страх и риск.

2015-01-11