**Теоретические сведения**

Качество программных средств во многом зависит от сложности их кодов. Например, чем сложнее программа, тем ниже ее надежность и сопровождаемость. Поэтому при оценке качества программ обычно оценивается и их сложность.

Метрики сложности программ принято подразделять на ***три основные группы*** [2]:

* метрики размера программ;
* метрики сложности потока управления программ;
* метрики сложности потока данных программ.

**Метрики размера программ**

Метрики этой группы основаны на анализе исходных текстов программ.

Существуют различные метрики, с помощью которых может быть оценен размер программы.

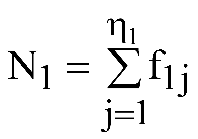
К наиболее простым метрикам размера программы относятся ***количество строк исходного текста программы***и ***количество операторов программы***.

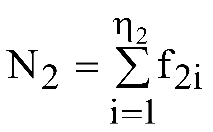
Из метрик размера программ широкое распространение получили ***метрики Холстеда*** [3].

Основу метрик Холстеда составляют *шесть базовых метрик* программы:

* η1 *–* словарь операторов (число уникальных операторов программы);
* η2 *–* словарь операндов (число уникальных операндов программы);
* *N1 –* общее число операторов в программе;
* *N2 –* общее число операндов в программе;
* *f1j –* число вхождений *j*-го оператора, *j* = 1, 2, …, η1;
* *f2i –* число вхождений *i*-го операнда, *i* = 1, 2, …, η2.

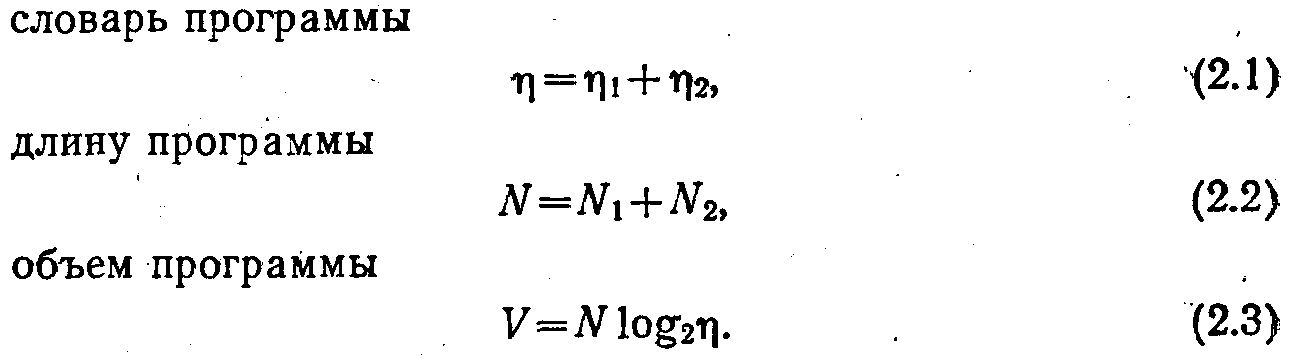
Справедливы следующие соотношения:

****

****

Базовые метрики определяются непосредственно при анализе исходных текстов программ. На основе базовых метрик Холстед предложил рассчитывать ряд производных метрик программы. Среди них рассмотрим следующие:

* словарь программы (общее число уникальных операторов и операндов программы):
* длина программы (общее количество операторов и операндов программы):
* объем программы (число битов, т.е. логических единиц информации, необходимых для записи программы):



Операнды программы представляют собой используемые в ней переменные и константы.

Под операторами программы в метриках Холстеда подразумеваются входящие в ее состав символы операций, символ присваивания, символы-разделители точка и точка с запятой, круглая скобка (пара из открывающей и закрывающей скобок считается одним оператором), управляющие операторы, составной оператор, а также имена процедур и функций.

Несколько служебных слов, входящих в состав одного оператора (например, If…Then…Else), считаются одним оператором.

Метки не относятся ни к операторам, ни к операндам.

Очевидно, что совокупность операторов программы и их количество зависят от языка программирования, на котором написана программа.

Операторы языка Ruby в интерпретации Холстеда приведены в табл. 1. При подсчете количества операторов и операндов в программе, написанной на языке Ruby, следует анализировать только ее раздел операторов, а также разделы операторов процедур и функций пользователя. (В скобках будут приведены аналоги из языка программирования Pascal, если таковые имеются).

Таблица 1

**Операторы языка Ruby в интерпретации Холстеда**

|  |  |
| --- | --- |
| Обозначение оператора | Назначение оператора |
| **+** | плюс (сложение, объединение множеств, сцепление строк) |
| ***–*** | минус (изменение знака, вычитание, разность множеств) |
| **\*** | звездочка (умножение, пересечение множеств) |
| **Fdiv** ( **/** ) | наклонная черта, слэш (знак деления, результат всегда имеет вещественный тип) |
| **<** | меньше |
| **>** | больше |
| **==** ( = ) | равно |
| Не существует ( . ) | точка (признак конца программы и модуля) |
| Не существует ( ; ) | точка с запятой (разделитель операторов программы) |
| **(  )** | левая и правая скобки при выделении подвыражений |
| **<=** | меньше или равно |
| **>=** | больше или равно |
| **!=** ( <> ) | не равно |
| **=** ( := ) | операция присваивания |
| Не существует ( **^** ) | знак карата (обращение к динамической переменной) |
| **&& ( And )** | операция поразрядного логического сложения (И) |
| **! ( Not )** | операция поразрядного дополнения (НЕ) |
| **|| ( Or )** | операция поразрядного логического сложения (ИЛИ) |
| **^ ( Xor )** | операция поразрядного логического исключающего ИЛИ |
| **/ ( Div )** | целочисленное деление |
| **% ( Mod )** | остаток от целочисленного деления |
| **<< ( Shl )** | операция сдвига влево |
| **>> ( Shr )** | операция сдвига вправо |
| **In** | операция проверки вхождения элемента в множество |
| **Begin…End** | составной оператор (begin в Ruby может быть опущен) |
| **Break** | оператор безусловного выхода из цикла |
| **Next ( Continue )** | оператор передачи управления на конец тела цикла |
| Не сущ. (**Goto <Метка>**) | оператор безусловного перехода |
| **Case…When…Else…End (Case…Of…Else…End)** | оператор варианта |
| **If…Else (If…Then…Else )** | оператор условного перехода |
| **Begin…End until (Repeat…Until)** | оператор цикла с постусловием |
| **While… (While…Do  )** | оператор цикла с предусловием |
| **For…in…End**  **(For…To…Do)** | оператор цикла с параметром (с увеличением параметра) |
| Не сущ. **(For…Downto…Do)** | оператор цикла с параметром (с уменьшением параметра) |
| Не сущ. (**With…Do**) | оператор присоединения |
| **Puts (Writeln)** | Оператор консольного вывода |
| **Gets.chomp (Readln)** | Оператор консольного ввода |

1. **Задание**

Изучить метрики размера программ на примере метрики Холстеда. Взять код программы в консольном режиме не менее чем на 200-300 строк (например, найти в Интернет) на языке программирования Ruby. Написать свою программу (на любом языке программирования не в консоли), которая будет анализировать программу. На основе анализа программа должна рассчитать метрики Холстеда. Результат работы программы: вывести на экран 6 основных метрик Холстеда и 3 расширенные в удобном для просмотра виде.

1. **Выполнение**

**Пример 1.** Расчет метрик Холстеда для программы, вычисляющей значение функции

***Y = sin X***

через разложение функции в бесконечный ряд

f156

с точностью ***Eps = 0,0001***.

Текст программы, написанной на языке Ruby, приведен ниже.

*eps = 0,0001*

*puts "Enter current value"*

*x = gets.chomp*

*y = x*

*n = 2*

*vs = x*

*begin*

*vs = - vs \* x \* x fdiv (2 \* n - 1) fdiv (2 \* n - 2)*

*n += 1*

*y += vs*

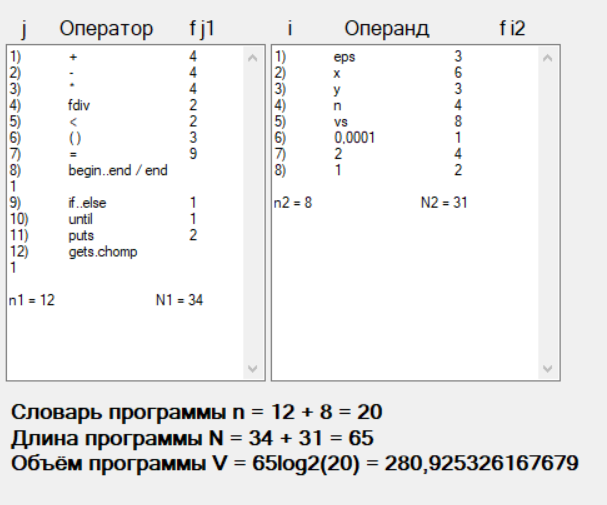
*if (vs < 0)*

*vs = - vs*

*end until vs < eps*

*puts x + y + eps*

Расчеты базовых метрик Холстеда для данной программы приведены ниже



Расчеты базовых метрик Холстеда для программы 200 – 300 строк приведены ниже

