## 2.1. 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*** [28].

Основу метрик Холстеда составляют *шесть базовых метрик* программы:

– $\eta_1$ – словарь операторов (число уникальных операторов программы);

– $\eta_2$ – словарь операндов (число уникальных операндов программы);

– $N_1$ – общее число операторов в программе;

– $N_2$ – общее число операндов в программе;

– $f_{1j}$ – число вхождений $j$-го оператора, $j = 1, 2, \ldots, \eta_1$;

– $f_{2i}$ – число вхождений $i$-го операнда, $i = 1, 2, \ldots, \eta_2$.

Справедливы следующие соотношения:

$$N_1 = \sum_{j=1}^{\eta_1} f_{1j} \qquad N_2 = \sum_{i=1}^{\eta_2} f_{2i}$$

Базовые метрики определяются непосредственно при анализе исходных текстов программ. На основе базовых метрик Холстед предложил рассчитывать ряд производных метрик программы. Наиболее широко из них используются следующие метрики:

– словарь программы (общее число уникальных операторов и операндов программы):

$$\eta = \eta_1 + \eta_2; \tag{2.1}$$

– длина программы (общее количество операторов и операндов программы):

$$N = N_1 + N_2; \tag{2.2}$$

– объем программы (число логических единиц информации, необходимых для записи программы):

$$V = N \log_2 \eta. \tag{2.3}$$

Операнды программы представляют собой используемые в ней переменные и константы.

Под операторами программы Холстед подразумевает входящие в ее состав символы-ограничители (в том числе символы операций, символы скобок и символы-разделители), управляющие операторы, а также имена подпрограмм (процедур и функций). При этом парные символы (например пара из открывающей и закрывающей скобок) считаются одним оператором. Несколько служебных

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

Метки не относятся ни к операторам, ни к операндам.

При подсчете операторов Холстеда объявления и инициализацию элементов программы (например, типов, констант, переменных) вместе с соответствующими символами-ограничителями принято не учитывать.

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

В табл. 2.1 приведены основные операторы процедурно-ориентированного подмножества языка Delphi в интерпретации Холстеда. При подсчете количества операторов и операндов в программе, написанной на языке Delphi, обычно анализируется только ее раздел операторов, а также разделы операторов процедур и функций пользователя.

Таблица 2.1

**Операторы языка Delphi в интерпретации Холстеда**

| Обозначение оператора | Назначение оператора |
|---|---|
| 1 | 2 |
| **+** | плюс (сложение, объединение множеств, сцепление строк) |
| **–** | минус (изменение знака, вычитание, разность множеств) |
| ***** | звездочка (умножение, пересечение множеств) |
| **/** | наклонная черта, слэш (деление) |
| **<** | меньше |
| **>** | больше |
| **=** | равно |
| **.** | точка (разделитель полей при обращении к элементам записи) |
| **,** | запятая (разделитель в перечислениях) |
| **:** | двоеточие (отделяет константы выбора в операторе Case) |
| **;** | точка с запятой (разделитель операторов программы) |
| **( )** | левая и правая скобки при выделении подвыражений |
| **[ ]** | левая и правая квадратные скобки (выделяет индексы элементов массивов) |
| **<=** | меньше или равно |
| **>=** | больше или равно |
| **<>** | неравно |
| **:=** | операция присваивания |
| **^** | знак карата (обращение к динамической переменной) |
| **@** | коммерческое ‘at’ (операция взятия адреса элемента) |
| **And** | операция поразрядного логического сложения (И) |
| **Not** | операция поразрядного дополнения (НЕ) |

Окончание табл. 2.1

| 1 | 2 |
|---|---|
| **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 |
| **Break** | оператор безусловного выхода из цикла |
| **Continue** | оператор передачи управления на конец тела цикла |
| **Goto <Метка>** | оператор безусловного перехода |
| **Case…Of… Else…End** | оператор варианта |
| **If…Then…Else** | оператор условного перехода |
| **Repeat…Until** | оператор цикла с постусловием |
| **While…Do** | оператор цикла с предусловием |
| **For…To…Do** | оператор цикла с параметром (с увеличением параметра) |
| **For…Downto… Do** | оператор цикла с параметром (с уменьшением параметра) |
| **With…Do** | оператор присоединения |

В табл. 2.2 приведены основные операторы языка *C* в интерпретации Холстеда.

Таблица 2.2

**Операторы языка *C* в интерпретации Холстеда**

| Обозначение оператора | Назначение оператора |
|---|---|
| 1 | 2 |
| **=** | операция присваивания |
| **+** | сложение |
| **–** | вычитание |
| ***** | звездочка (умножение, обращение к динамической переменной) |
| **/** | деление |
| **%** | остаток от целочисленного деления |
| **++** | инкремент |
| **--** | декремент |
| **==** | равно |
| **!=** | неравно |

Продолжение табл. 2.2

| 1 | 2 |
|---|---|
| **>** | больше |
| **<** | меньше |
| **>=** | больше или равно |
| **<=** | меньше или равно |
| **!** | логическое отрицание (НЕ) |
| **&&** | логическое И |
| **\|\|** | логическое ИЛИ |
| **~** | побитовая инверсия |
| **&** | побитовое И, ссылка |
| **\|** | побитовое ИЛИ |
| **^** | побитовое исключающее ИЛИ |
| **<<** | побитовый сдвиг влево (в сторону старших разрядов) |
| **>>** | побитовый сдвиг вправо (в сторону младших разрядов) |
| **+=** | присваивание с суммированием |
| **-=** | присваивание с вычитанием |
| ***=** | присваивание с умножением |
| **/=** | присваивание с делением |
| **%=** | присваивание по модулю |
| **&=** | присваивание с побитовым И |
| **\|=** | присваивание с побитовым ИЛИ |
| **^=** | присваивание с побитовым исключающим ИЛИ |
| **<<=** | присваивание с побитовым сдвигом влево |
| **>>=** | присваивание с побитовым сдвигом вправо |
| **( )** | левая и правая скобки при выделении подвыражений |
| **[]** | обращение к элементу массива |
| **->** | динамическое обращение к элементу структуры |
| **.** | статическое обращение к элементу структуры |
| **,** | запятая (операция и разделитель) |
| **;** | точка с запятой (разделитель операторов программы) |
| **:** | двоеточие (отделяет константы выбора в операторе switch) |
| **?…:** | условный оператор |
| **sizeof** | размер |
| **(type)** | преобразование типа |
| **{}** | составной оператор |
| **if…else** | оператор выбора |
| **switch()…case… default** | оператор множественного выбора |
| **do…while( )** | оператор цикла с постусловием |
| **while( )** | оператор цикла с предусловием |

Окончание табл. 2.2

| 1 | 2 |
|---|---|
| **for( )** | оператор цикла с параметром |
| **goto <метка>** | оператор безусловного перехода |
| **continue** | оператор перехода к следующему шагу цикла |
| **break** | оператор выхода из цикла |

**Пример 1**

Расчет метрик Холстеда для программы, вычисляющей значение функции $Y = \sin X$ через разложение функции в бесконечный ряд

$$Y = \sin X = X - X^3 / 3! + X^5 / 5! - X^7 / 7! + \ldots$$

с точностью $Eps = 0{,}0001$.

Текст программы на языке Delphi, реализующей вычисление функции $Y$, приведен ниже.

```
Program  Sin1;
   Const
      eps = 0.0001;
   Var
      y, x: real;  n: integer;  vs: real;
   Begin
      Readln (x);
      y := x; {Начальные установки}
      n := 2;
      vs := x;
      Repeat
        vs := –vs * x * x / (2 * n – 1) / (2 * n – 2); {Формирование слагаемого}
        n := n + 1;
        y := y + vs
      Until  abs(vs) < eps;  {Выход из цикла по выполнению условия}
      Writeln (x, y, eps)
   End.
```

Расчет базовых метрик Холстеда для данной программы приведен в табл. 2.3. По формулам (2.1) – (2.3) рассчитываются словарь, длина и объем программы.

Словарь программы: $\eta = 14 + 7 = 21$.

Длина программы: $N = 34 + 28 = 62$.

Объем программы: $V = 62 \log_2 21 \approx 272$.

Таблица 2.3

**Расчет базовых метрик Холстеда для программы, вычисляющей значение функции *Y* = sin *X***

| ***j*** | Оператор | $f_{1j}$ | ***i*** | Операнд | $f_{2i}$ |
|---|---|---|---|---|---|
| 1. | ; | 7 | 1. | x | 6 |
| 2. | := | 6 | 2. | n | 5 |
| 3. | * | 4 | 3. | vs | 5 |
| 4. | – | 3 | 4. | y | 4 |
| 5. | / | 2 | 5. | 2 | 4 |
| 6. | ( ) | 2 | 6. | 1 | 2 |
| 7. | + | 2 | 7. | eps | 2 |
| 8. | , | 2 | | | |
| 9. | Begin…End | 1 | | | |
| 10. | Readln ( ) | 1 | | | |
| 11. | Repeat…Until | 1 | | | |
| 12. | abs( ) | 1 | | | |
| 13. | < | 1 | | | |
| 14. | Writeln ( ) | 1 | | | |
| **$\eta_1 = 14$** | | **$N_1 = 34$** | **$\eta_2 = 7$** | | **$N_2 = 28$** |

**Пример 2**

Расчет метрик Холстеда для программы на языке *C*. Программа реализует вычисление той же функции ***Y* = sin *X***, что и программа из примера 1.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   const double eps = 0.0001;
   double y, x;
   int n;
   double vs;
   scanf("%f", &x);
   y = x; // Начальные установки
   n = 2;
   vs = x;
   do
   {
      vs = –vs * x * x / (2 * n – 1) / (2 * n – 2);  // Формирование слагаемого
      n++;
      y += vs;
   }
```

```
    while (abs(vs) >= eps); // Выход из цикла по выполнению условия
    printf("%f %f %f\n", x, y, eps);
}
```

Расчет базовых метрик Холстеда для данной программы приведен в табл. 2.4. По формулам (2.1) – (2.3) рассчитываются словарь, длина и объем программы.

Словарь программы: $\eta = 16 + 6 = 22$.

Длина программы: $N = 38 + 24 = 62$.

Объем программы: $V = 62 \log_2 22 \approx 276$.

Таблица 2.4

**Расчет базовых метрик Холстеда для программы, вычисляющей значение функции *Y* = sin *X* на языке *C***

| *j* | Оператор | $f_{1j}$ | i | Операнд | $f_{2i}$ |
|---|---|---|---|---|---|
| 1. | ; | 9 | 1. | x | 6 |
| 2. | = | 4 | 2. | vs | 5 |
| 3. | * | 4 | 3. | n | 4 |
| 4. | , | 4 | 4. | 2 | 4 |
| 5. | – | 3 | 5. | y | 3 |
| 6. | / | 2 | 6. | eps | 2 |
| 7. | ( ) | 2 | | | |
| 8. | {…} | 2 | | | |
| 9. | scanf ( ) | 1 | | | |
| 10. | do…while ( ) | 1 | | | |
| 11. | abs( ) | 1 | | | |
| 12. | >= | 1 | | | |
| 13. | printf ( ) | 1 | | | |
| 14. | ++ | 1 | | | |
| 15. | += | 1 | | | |
| 16. | & | 1 | | | |
| $\eta_1 = 16$ | | $N_1 = 38$ | $\eta_2 = 6$ | | $N_2 = 24$ |

## 2.2. Метрики сложности потока управления программ

Метрики сложности потока управления программ принято определять на основе представления программ в виде управляющего ориентированного графа $G = (V, E)$, где $V$ – вершины, соответствующие операторам, а $E$ – дуги, соответствующие переходам между операторами [27, 29]. В дуге $(v, u)$ вершина $v$ является исходной, а $u$ – конечной. При этом $u$ непосредственно следует за $v$, а $v$ непосредственно предшествует $u$. Если путь от $v$ до $u$ состоит более чем из одной дуги, тогда $u$ следует за $v$, а $v$ предшествует $u$.