**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
(ФГАОУ ВО «ЮФУ»)**

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по курсу **«Верификация, качество и автоматизированное тестирование программного обеспечения»**  
по модулю «Метрология программного обеспечения»

на тему: **«Определение и оценка метрических характеристик программного обеспечения»**

Выполнили

студенты группы КТбо2-9 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С. В. Макаричев

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Яцуненко

Принял

старший преподаватель  
кафедры МОП ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. В. Проскуряков

1 Цель работы

Целью данной лабораторно-практической работы является определение метрических характеристик программного обеспечения, посредством использования метрик.

2 Постановка задачи

1. Составить программу реализации арифметического выражения (**Вариант 1)**.

2а!x3+3(a+b)!=0, где

а - количество положительных элементов, стоящих в матрице t (5, 5) в столбцах с четными номерами;

б - количество положительных элементов, стоящих в матрице (6, 6) в столбцах с четными номерами.

1. Подсчитать количество операторов и операндов и определить объем программы.
2. Написать программу-анализатор, которая определит объем исходной программы.
3. Составить отчет по результатам выполнения лабораторно-практической работы.

3 Ход выполнения работы

3.1 Краткое теоретическое введение

Метрика – отношение, определяющее расстояние между элементами множества, причём метрика: 1) всегда неотрицательна и равна нулю, если элементы множества совпадают; 2) метрика симметрична; 3) метрика подчиняется аксиоме треугольника.

В исследовании метрик ПО различают 2 основных направления: 1) поиск метрик, характеризующих наиболее специфические свойства программ, т. е. метрик оценки самого ПО; 2) использование метрик для оценки технических характеристик и факторов разработки программ, т. е. метрик оценки условий разработки программ.

По виду информации, получаемой при оценке качества ПО, метрики можно разбить на три группы: 1) метрики, оценивающие отклонение от нормы характеристик исходных проектных материалов; 2) метрики, позволяющие прогнозировать качество разрабатываемого ПО; 3) метрики, по которым принимается решение соответствии конечного ПО заданным требованиям.

К группе оценок размера программ можно отнести метрику Холстеда. Основу метрики составляют четыре измеряемых характеристики программы: число уникальных операторов, число уникальных операндов, общее число операторов в программе и общее число операндов. Данная метрика выражается следующей формулой:

*, где*

*n — число простых операторов и операндов;*

*N — общее количество использованных операторов и операндов.*

3.1 Разработка программы

Мы реализовали решение уравнения на языке C++. Работа кода проиллюстрирована на рисунке 1. Код программы представлен в приложении А.

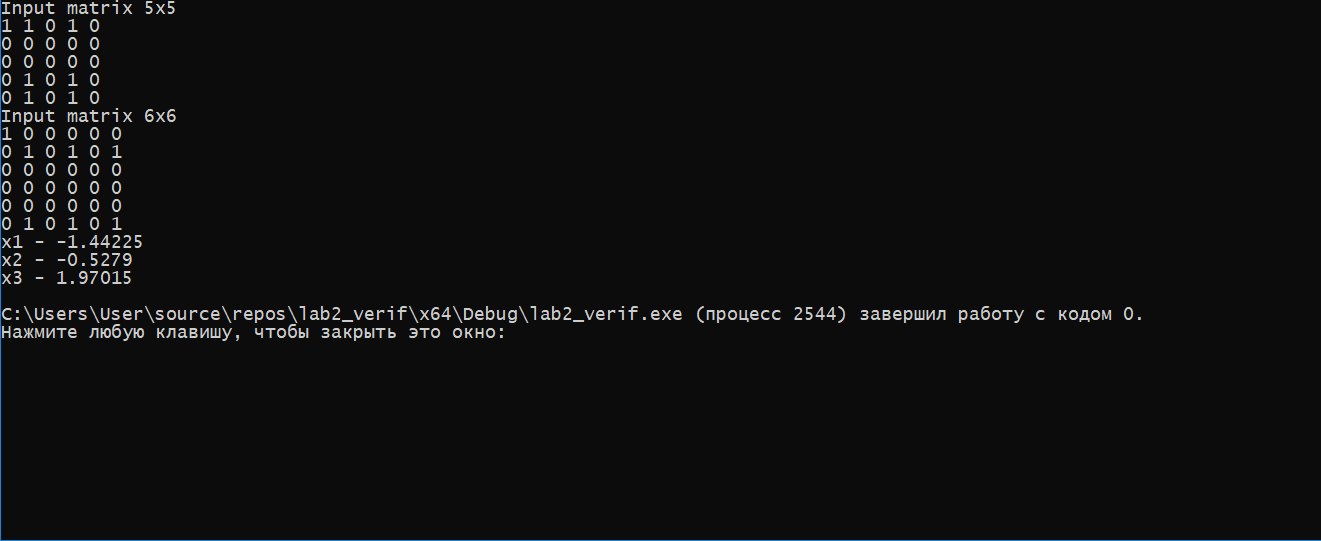


Рисунок 1 – Результат работы программы для решения уравнения

3.2 Вычисление объёма программы

Для подсчёта объема программы мы будем использовать метрику Холстеда, для которой нам необходимо посчитать n (число простых операторов и операндов) и N (общее количество использованных операторов и операндов).

Подсчет количества операторов и операндов в программе оформлен в Таблице 1 и Таблице 2.

Таблица 1 – Подсчет операторов

|  |  |  |
| --- | --- | --- |
| **Оператор** | **j** | **f1φ** |
| += | 1 |  |
| () | 2 | 3 |
| << | 3 | 13 |
| cin | 4 | 2 |
| >> | 5 | 4 |
| == | 6 | 3 |
| \* | 7 | 14 |
| - | 8 | 12 |
| = | 9 | 44 |
| cout | 10 | 5 |
| < | 11 | 8 |
| ++ | 12 | 10 |
| + | 13 | 28 |
| abs | 14 |  |
| [] | 15 |  |
| / | 16 | 14 |
| return | 17 | 6 |
| != | 18 |  |
| && | 19 |  |
| sqrt | 20 | 6 |
| for | 21 | 8 |
| if | 22 | 5 |
| else | 23 |  |
| **Итого:** | **η1 = 23** | **N­1 = 185** |

Таблица 2 – подсчет операндов

|  |  |  |
| --- | --- | --- |
| **Операнд** | **j** | **f2φ** |
| a | 1 | 9 |
| b | 2 | 5 |
| Temp1 | 3 | 12 |
| Temp2 | 4 | 10 |
| Temp3 | 5 | 7 |
| X1 | 6 | 4 |
| X2 | 7 | 3 |
| X3 | 8 | 3 |
| 1.0 | 9 | 11 |
| 3 | 10 | 13 |
| 2 | 11 | 6 |
| 6 | 12 | 6 |
| 5 | 13 | 6 |
| -1 | 14 | 5 |
| 0 | 15 | 13 |
| Count\_t | 16 | 1 |
| Count\_d | 17 | 1 |
| temp | 18 | 1 |
| **Итого:** | **η2 = 18** | **N2 = 116** |

Используя ручные подсчеты, мы определили объем программы:

η=η1 + η2 – Словарь программы

η =23 + 18 = 41

N = N1 + N2 – Длина реализации

N = 185 + 116 = 301

V = 301\* log241 = 1612,62 – Объём программы

3.2 Разработка программы-анализатора

Для написания программы-анализатора мы использовали язык с++. На рисунке 2 отображен результат анализа с помощью программы. Полный код представлен в приложении В.

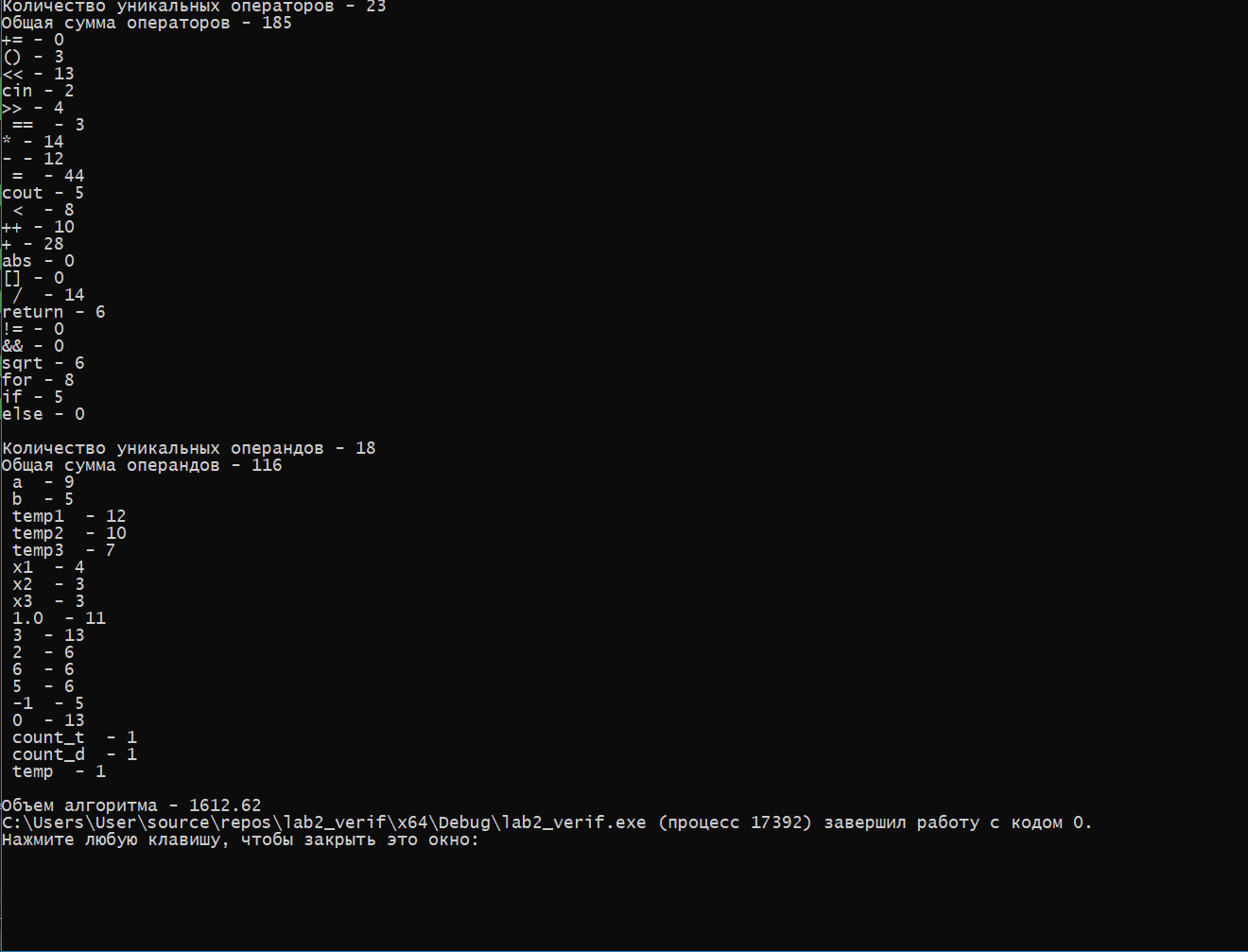


Рисунок 2 – Результат работы программы-анализатора

Сначала анализатор выводит на экран количество уникальных операторов η1, затем – общее количество операторов N1, а также их список с числом вхождений каждого оператора. После отображаются значения η2 и N2 для операндов, так же с выводом их списка с числом вхождений. Последнее выводимое значение – объем алгоритма V, рассчитанный по формуле, представленной выше.

Сравнивая результаты анализа вручную и с помощью программы, мы видим, что получены одинаковые значения по всем характеристикам.

Полученная нами величина V показывает размер программы в условных единицах, соответственно, чем больше значение, тем больше теоретический словарь программы и длина реализации, а значит сложнее программа.

**ВЫВОД**

В ходе данной лабораторной работы мы познакомились с метрическими характеристиками программного обеспечения, а также их определение посредством использования метрик. Нами была написана программа-анализ, позволяющая посчитать объем программного кода, вычисленные значения которой не уступают посчитанным вручную.

Мы сделали вывод, что вычисленный объем программы напрямую зависит от количества кода, количества повторов операторов и операндов. Поэтому для упрощения и улучшения качества программы следует уменьшить количество строк программы и повторы определенных элементов.

η=η1 + η2 – Словарь программы

η =23 + 18 = 41

N = N1 + N2 – Длина реализации

N = 185 + 116 = 301

V = 301\* log241 = 1612,62 – Объём программы

ПРИЛОЖЕНИЕ А

# include <iostream>

# include <cmath>

# include <complex>

# include <vector>

double factorial( double n )//фаториал отрицательного числа не существует

{

if ( n == 0 )

return 1 ;

return n \* factorial( n - 1 );

}

double funcF(double a , double b ) // x1

{

double x1 ; //в решении такого уравнения у нас 3 корня

double temp1 = a + b ;

temp1 = factorial( temp1 ) ; //вычислили факториал

temp1 \*= 3 ;

double temp2 = factorial( a ) ;

temp2 \*= 2 ;

double temp3 = temp1 / temp2 ;

x1 = pow(temp3, 1.0 / 3 ) ; //корень 3 степени

x1 = x1 \* -1 ;

return x1 ;

}

double funcS(double a , double b ) // x2

{

double temp1 = a + b ;

temp1 = factorial( temp1 ) ; // (a+b)!

temp1 = pow( temp1 , 1.0 / 3 ) ; // sqrt³((a+b)!)

double temp2 = pow( 3 , 1.0 / 3 ) ;

temp2 \*= -1 ; // -sqrt³(3)

double temp3 = pow( 3, 5 ) ;

temp3 = pow( temp3, 1.0 / 6 ) ; // sqrt⁶(3⁵)

temp2 = temp2 + temp3 ;

temp1 = temp1 \* temp2 ; // complete

temp2 = pow( 2 , 1.0 / 3 ) ;

temp3 = factorial( a );

temp3 = pow( temp3, 1.0 / 3 );

temp2 = temp2 \* temp3 ;

temp2 \*= 2 ; // complete

double x2 = temp1 / temp2 ;

x2 \*= -1 ;

return x2 ;

}

double funcT( double a , double b ) // x3

{

double temp1 = a + b ;

temp1 = factorial( temp1 ) ; // (a+b)!

temp1 = pow( temp1, 1.0 / 3 ) ; // sqrt³((a+b)!)

double temp2 = pow( 3, 1.0 / 3 );

temp2 \*= -1 ; // -sqrt³(3)

double temp3 = pow( 3, 5 );

temp3 = pow(temp3, 1.0 / 6 ); // sqrt⁶(3⁵)

temp2 = temp2 - temp3 ;

temp1 = temp1 \* temp2 ; // complete

temp2 = pow( 2, 1.0 / 3 );

temp3 = factorial( a );

temp3 = pow( temp3, 1.0 / 3 );

temp2 = temp2 \* temp3 ;

temp2 \*= 2 ; // complete

double x3 = temp1 / temp2 ;

x3 \*= -1 ;

return x3 ;

}

int main()

{

double a, b ;

std::vector<std::vector <int>> vec\_t ; // матрица t

std::vector<int> vec\_temp\_t ;

std::cout << "Input matrix 5x5" << std::endl;

for ( int i = 0 ; i < 5 ; ++ i )

{

for ( int j = 0 ; j < 5 ; ++ j )

{

int temp ;

std::cin >> temp ;

vec\_temp\_t.push\_back( temp ) ;

}

vec\_t.push\_back( vec\_temp\_t ) ;

vec\_temp\_t.clear() ;

}

int count\_t = 0 ;

for (int i = 0 ; i < 5 ; ++ i )

{

for (int j = 0; j < 5 ; ++ j )

{

if (j % 2 == 0)

{

if (vec\_t[i][j] > 0 )

{

count\_t++;

}

}

}

}

a = count\_t;

std::cout << "Input matrix 6x6" << std::endl ;

std::vector<std::vector <int>> vec\_d ; // матрица d

std::vector<int> vec\_temp\_d;

for (int i = 0 ; i < 6 ; ++ i )

{

for (int j = 0 ; j < 6 ; ++ j )

{

int temp;

std::cin >> temp;

vec\_temp\_d.push\_back(temp);

}

vec\_d.push\_back(vec\_temp\_d);

vec\_temp\_d.clear();

}

int count\_d = 0;

for (int i = 0 ; i < 6 ; ++ i )

{

for (int j = 0 ; j < 6 ; ++ j )

{

if (j % 2 == 0 )

{

if (vec\_d[i][j] > 0 )

{

count\_d ++;

}

}

}

}

b = count\_d;

std::cout << " x1 - " << funcF(a, b) << std::endl;

std::cout << " x2 - " << funcS(a, b) << std::endl;

std::cout << " x3 - " << funcT(a, b) << std::endl;

return 0 ;

}

**ПРИЛОЖЕНИЕ В**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

int frequency\_Substr(string s, string ss)

{

int n = 0, p = 0;

while ((p = s.find(ss, p)) != s.npos) {

++n;

++p;

}

return n;

}

int main()

{

setlocale(LC\_ALL, "Russian");

string text;

ifstream programm("code.txt");

int n1 = 0, n2 = 0;

int number\_of\_operators[23] = { 0 };

for (int i = 0; i < 23; i++)

n1++;

string operators[23] = {

"+=",

"()",

"<<",

"cin",

">>",

" == ",

"\*",

"-",

" = ",

"cout",

" < ",

"++",

"+",

"abs",

"[]",

" / ",

"return",

"!=",

"&&",

"sqrt",

"for",

"if",

"else"

};

int number\_of\_operands[18] = { 0 };

for (int i = 0; i < 18; i++)

n2++;

string operands[18] =

{

" a ", " b ", " temp1 ", " temp2 ", " temp3 ", " x1 ", " x2 ", " x3 ", " 1.0 ", " 3 ", " 2 ", " 6 ", " 5 ", " -1 ",

" 0 ", " count\_t ", " count\_d ", " temp "

};

std::vector<int> vec ;

std::vector<int> gec ;

string strh ;

string str ; // весь код в одной строке

while ( !programm.eof() )

{

getline( programm, strh ) ;

str += strh;

if ( !programm.eof() )

{

str += '\n';

}

}

for ( int i = 0; i < 18; ++ i )

{

vec.push\_back(frequency\_Substr( str, operands[i] )) ;

}

for ( int i = 0; i < 23; ++i )

{

gec.push\_back(frequency\_Substr( str, operators[i])) ;

}

programm.close();

cout << "Количество уникальных операторов - " << n1 << endl;

cout << "Общая сумма операторов - ";

int N1 = 0;

for (int i = 0; i < 23; i++)

N1 += gec[i];

cout << N1 << endl;

for (int i = 0; i < 23; i++)

cout << operators[i] << " - " << gec[i] << endl;;

cout << endl << "Количество уникальных операндов - " << n2 << endl;

int N2 = 0;

for (int i = 0; i < 18; i++)

N2 += vec[i];

cout << "Общая сумма операндов - " << N2 << endl;

for (int i = 0; i < 18; i++)

cout << operands[i] << " - " << vec[i] << endl;

cout << endl << "Объем алгоритма - " << (N1 + N2) \* log2(n1 + n2);

}