МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Качество и метрология программного обеспечения»
Тема: Расчет метрических характеристик качества разработки программ по метрикам Холстеда

Студент гр. 7304 _____ Абдульманов Э.М

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы:

Расчет и сравнение метрик Холстеда для программ, написанных на языках Паскаль, Си, Ассемблер.

Задание:

Для заданного варианта программы обработки данных, представленной на языке Паскаль, разработать вычислительный алгоритм и также варианты программ его реализации на языках программирования Си и Ассемблер.

Для каждой из разработанных программ (включая исходную программу на Паскале) определить следующие метрические характеристики (по Холстеду):

1. Измеримые характеристики программ:
   - число простых(отдельных)операторов, в данной реализации;
   - число простых (отдельных) операндов, в данной реализации;
   - общее число всех операторов в данной реализации;
   - общее число всех операндов в данной реализации;
   - число вхождений j-го оператора в тексте программы;
   - число вхождений j-го операнда в тексте программы;
   - словарь программы;
   - длину программы.
2. Расчетные характеристики программы:
   - длину программы;
   - реальный и потенциальный объемы программы;
   - уровень программы;
   - интеллектуальное содержание программы;
   - работу программиста;
   - время программирования;
   - уровень используемого языка программирования;
   - ожидаемое число ошибок в программе.

Ход работы:

1. Расчет метрик вручную

Программа на языке Паскаль, Си и Ассемблер представлены в приложениях А, Б и В соответственно.

В таблицах 1-3 представлены результаты подсчета количества операторов и операндов для программ, написанных на языках Паскаль, Си, Ассемблер.

Таблица 1 – Количество операторов и операндов в программе, написанной на языке Паскаль.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	;	23	**1**	1.0E-6	1
2	*	11	**2**	2	6
3	+	9	**3**	tol	3
4	-	8	**4**	sum	9
5	/	8	**5**	upper	7
6	()	24	**6**	lower	9
7	<=	1	**7**	x	6
8	<>	1	**8**	i	1
9	=	20	**9**	delta_x	7
10	begin…end	5	**10**	even_sum	4
11	repeat…until	1	**11**	odd_sum	7
12	abs	2	**12**	end_sum	3
13	and	1	**13**	end_cor	2
14	simps	2	**14**	sum1	3
15	fx	6	**15**	pieces	6
16	dfx	4	**16**	0.0	2
17	exp	2	**17**	4.0	1
18	div	1	**18**	simp1	1
19	for…to…do	1	**19**	1.0	2
20	writeln	1	**20**	7.0	1
21	chr	1	**21**	14.0	1
			22	16.0	1
			23	15.0	1
			24	9.0	1
			25	1	1
			26	2.0	1
			27	3.0	1

Таблица 2 – Количество операторов и операндов в программе, написанной на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	;	25	**1**	1.0E-6	1
2	*	14	**2**	2	3
3	+	9	**3**	tol	3
4	-	5	**4**	sum	9
5	/	8	**5**	upper	7
6	()	14	**6**	lower	8
7	<=	2	**7**	x	6
8	<>	1	**8**	i	3
9	=	22	**9**	delta_x	7
10	main	1	**10**	even_sum	4
11	do…while	1	**11**	odd_sum	7
12	abs	2	**12**	end_sum	3
13	and	1	**13**	end_cor	2
14	simps	2	**14**	sum1	3
15	fx	6	**15**	pieces	6
16	dfx	4	**16**	0.0	2
17	exp	2	**17**	4.0	1
18	&	1	**18**	simp1	1
19	++	1	**19**	1.0	5
20	printf	2	**20**	7.0	1
21	return	4	**21**	14.0	1
22	!=	1	**22**	16.0	1
23	for	1	**23**	15.0	1
			24	9.0	1
			25	3.0	1
			26	1	1
			27	2.0	4

Таблица 3 – Количество операторов и операндов в программе, написанной на языке Ассемблер.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	push	5	**1**	rbp	8
2	mov	23	**2**	rsp	8
3	sub	4	**3**	16	2

4	movsd	58	**4**	QWORD PTR [rbp-8]	7
5	movq	34	**5**	xmm0	121
6	xorpd	3	**6**	xmm1	41
7	divsd	7	**7**	rax	38
8	call	12	**8**	QWORD PTR .LC1[rip]	3
9	leave	4	**9**	QWORD PTR .LC2[rip]	3
10	ret	4	**10**	exp	2
11	movapd	6	**11**	xmm2	11
12	pxor	7	**12**	rbx	4
13	subsd	5	**13**	120	1
14	cvtsi2sd	3	**14**	QWORD PTR [rbp-88]	7
15	addsd	10	**15**	QWORD PTR [rbp-96]	7
16	mulsd	9	**16**	QWORD PTR [rbp-104]	2
17	sal	1	**17**	QWORD PTR [rbp-112]	9
18	shr	1	**18**	DWORD PTR [rbp-36]	5
19	add	2	**19**	xmm3	2
20	sar	1	**20**	2	1
21	cmp	1	**21**	QWORD PTR [rbp-48]	7
22	jg	1	**22**	fx	8
23	jmp	2	**23**	QWORD PTR [rbp-120]	4
24	ucomisd	2	**24**	QWORD PTR [rbp-56]	3
25	setp	1	**25**	dfx	3
26	cmove	1	**26**	QWORD PTR [rbp-32]	10
27	comisd	1	**27**	QWORD PTR [rbp-24]	6
28	setnb	1	**28**	QWORD PTR [rbp-64]	2
29	and	1	**29**	QWORD PTR .LC4[rip]	1
30	movzx	1	**30**	QWORD PTR .LC5[rip]	1
31	test	2	**31**	QWORD PTR [rbp-72]	4
32	setne	1	**32**	DWORD PTR [rbp-40]	4
33	je	1	**33**	1	4
			34	eax	12
			35	edx	5
			36	31	1
			37	QWORD PTR .LC6[rip]	2
			38	QWORD PTR .LC7[rip]	1
			39	QWORD PTR .LC8[rip]	1
			40	QWORD PTR .LC9[rip]	1
			41	QWORD PTR .LC10[rip]	1
			42	al	6
			43	ebx	3
			44	xmm4	3
			45	xmm6	2

			46	32	1
			47	QWORD PTR [rbp-16]	2
			48	QWORD PTR .LC11[rip]	1
			49	WORD PTR .LC12[rip]	1
			50	edi	1
			51	0	1

Для расчета значение коэффициента Стауда S принято 10; значение η_2^* для Паскаля принято 4, поскольку исследуемая процедура simps принимает 4 аргумента и не имеет возвращаемого значения, а для С и Ассемблер η_2^* принято 5, поскольку исследуемая функция simps принимает 4 аргумента и возвращает одно значение. В таблице 4 представлены результаты расчета метрик Холстеда вручную для программ, реализованных на языках Паскаль, Си, Ассемблер.

Таблица 4 – Результаты расчета метрик вручную.

Характеристики	**Паскаль**	**Си**	**Ассемблер**
Число уникальных операторов	21	23	33
Число уникальных операндов	28	27	51
Общее число операторов	131	129	215
Общее число операндов	89	91	384
Алфавит	49	50	84
Экспериментальная длина программы	220	220	599
Теоретическая длина программы	226.844603	232.423887	455.758698
Объем программы	1235.236165	1241.64836	3828.99813
Потенциальный объем	15.509	19.651	19.651
Уровень программы	0.012556	0.015826	0.005132
Интеллектуальное содержание	37.01082	32.03488	30.82053
Работа по программированию	98377.2095	78451.6130	746062.0474
Время программирования	9837.72095	7845.16130	74606.20474
Уровень языка программирования	0.194742	0.311022	0.100856
Уровень ошибок	2	2	4

2. Программный расчет метрик

Результаты программного расчета метрик для программ, реализованных на языках Паскаль, Си представлены в приложениях Г и Д соответственно.

В таблицах 5-6 представлены результаты программного подсчета количества операторов и операндов для программ, написанных на языках Паскаль, Си.

Таблица 5 – Количество операторов и операндов в программе, написанной на языке Паскаль.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	()	24	**1**	“area=”	1
2	*	11	**2**	0.0	2
3	+	9	**3**	1	1
4	-	8	**4**	1.0	2
5	/	8	**5**	1.0E-6	1
6	;	47	**6**	14.0	1
7	<=	1	**7**	15.0	1
8	<>	1	**8**	16.00	1
9	=	20	**9**	2	6
10	abs	2	**10**	2.0	1
11	and	1	**11**	3.0	1
12	chr	1	**12**	4.0	1
13	const	1	**13**	7	1
14	dfx	3	**14**	7.0	1
15	exp	2	**15**	9.0	1
16	for	1	**16**	delta_x	8
17	function	2	**17**	dfx	1
18	fx	5	**18**	end_cor	3
19	integer	2	**19**	end_sum	4
20	procedure	1	**20**	even_sum	5
21	program	1	**21**	fx	1
22	real	8	**22**	i	2
23	repeat	1	**23**	lower	10
24	simps	2	**24**	odd_sum	8
25	writeln	2	**25**	pieces	7
			26	simp1	1
			27	sum	10
			28	sum1	4
			29	tol	4
			30	upper	8
			31	x	7

Таблица 6 – Количество операторов и операндов в программе, написанной на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	!=	1	**1**	“area=%1f”	1
2	&	1	**2**	0	1
3	()	14	**3**	0.0	4
4	*	14	**4**	1	1
5	+	9	**5**	1.0	5
6	++	1	**6**	1.0E-6	1
7	,	13	**7**	14.0	1
8	-	5	**8**	15.0	1
9	/	8	**9**	16.00	1
10	;	33	**10**	2	3
11	<=	2	**11**	2.0	4
12	=	22	**12**	3.0	1
13	_-	3	**13**	4.0	1
14	abs	2	**14**	7.0	1
15	const	1	**15**	9.0	1
16	_dfx	3	**16**	delta_x	8
17	double	15	**17**	end_cor	3
18	do…while	1	**18**	end_sum	4
19	exp	2	**19**	even_sum	5
20	for	1	**20**	i	4
21	fx	5	**21**	lower	9
22	int	3	**22**	odd_sum	8
23	main	1	**23**	pieces	7
24	printf	1	**24**	res	3
25	return	4	**25**	sum	10
26	simps	2	**26**	sum1	4
27	void	1	**27**	tol	4
			28	upper	7
			29	x	7

В таблице 7 представлены результаты программного расчета метрик Холстеда для программ, реализованных на языках Паскаль, Си.

Таблица 7 – Результаты программного расчета метрик.

Характеристики	Паскаль	Си
Число уникальных операторов	25	27
Число уникальных операндов	31	29
Общее число операторов	164	168
Общее число операндов	105	110
Алфавит	56	56
Экспериментальная длина программы	269	278
Теоретическая длина программы	269.676	269.263
Объем программы	1562.18	1614.44
Потенциальный объем	19.6515	19.6516
Уровень программы	0.0125795	0,0121723
Интеллектуальное содержание	36.8972	31.5279
Работа по программированию	124184	132633
Время программирования	6899.12	7368.49
Уровень языка программирования	0.247207	0.239204
Уровень ошибок	0.829703	0.866921

3. Сравнение полученных результатов

В таблице 8 представлены результаты программного и ручного расчета метрик Холстеда для программ, реализованных на языках Паскаль, Си.

Таблица 8 – Сводная таблица расчетов на языках Паскаль, Си.

Характеристики	Ручной расчет Паскаль	Программный расчет Паскаль	Ручной расчет Си	Программный расчет Си
Число уникальных операторов	21	25	23	27
Число уникальных операндов	28	31	27	29
Общее число операторов	131	164	129	168
Общее число операндов	89	105	91	110
Алфавит	49	56	50	56
Экспериментальная длина программы	220	269	220	278
Теоретическая длина программы	226.844603	269.676	232.423887	269.263
Объем программы	1235.236165	1562.18	1241.64836	1614.44
Потенциальный объем	15.509	19.651	19.651	19.651

Уровень программы	0.012556	0.0125795	0.015826	0,0121723
Интеллектуальное содержание	37.01082	36.8972	32.03488	31.5279
Работа по программированию	98377.2095	124184	78451.6130	132633
Время программирования	9837.72095	6899.12	7845.16130	7368.49
Уровень языка программирования	0.194742	0.247207	0.311022	0.239204
Уровень ошибок	2	1	2	1

Вывод.

В ходе выполнения лабораторной работы были изучены метрические характеристики качества разработки программ на основе метрик Холстеда. Метрические характеристики программ на языках Си и Паскаль выглядят похожим образом, а на языке Ассемблер сильно отличаются (ассемблер является языком низкого уровня). Так же все характеристики были посчитаны вручную и автоматически. Полученные различия в результатах обусловлены тем, что автоматический метод считает не только функциональную часть программы, но и объявления типов переменных и функций.

ПРИЛОЖЕНИЕ А.

КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

```
PROGRAM SIMP1;
{ INTEGRATION BY SIMPSON'S METHOD }
CONST TOL = 1.0E-6;
VAR SUM,UPPER,LOWER    : REAL;
FUNCTION FX(X: REAL): REAL;
BEGIN
    FX:=EXP(-X/2)
END;
{ FUNCTION FX }
FUNCTION DFX(X: REAL): REAL;
BEGIN
    DFX:=-(EXP(-X/2))/2
END;
{ FUNCTION DFX }
PROCEDURE SIMPS(LOWER,UPPER,TOL   : REAL; VAR SUM  : REAL);
{ NUMERICAL INTEGRATION BY SIMPSON'S RULE }
{ FUNCTION IS FX, LIMITS ARE LOWER AND UPPER }
{ WITH NUMBER OF REGIONS EQUAL TO PIECES }
{ PARTITION IS DELTA_X, ANSWER IS SUM }
VAR I : INTEGER;
X,DELTA_X,EVEN_SUM, ODD_SUM,END_SUM, END_COR,SUM1 : REAL;
PIECES     : INTEGER;
BEGIN
    PIECES:=2;
    DELTA_X:=(UPPER-LOWER)/PIECES;
    ODD_SUM:=FX(LOWER+DELTA_X);
    EVEN_SUM:=0.0;
    END_SUM:=FX(LOWER)+FX(UPPER);
    END_COR:=DFX(LOWER)-DFX(UPPER);
    SUM:=(END_SUM+4.0*ODD_SUM)*DELTA_X/3.0;
    REPEAT
        PIECES:=PIECES*2;
        SUM1:=SUM;
        DELTA_X:=(UPPER-LOWER)/PIECES;
        EVEN_SUM:=EVEN_SUM+ODD_SUM;
        ODD_SUM:=0.0;
        FOR I:=1 TO PIECES DIV 2 DO
            BEGIN
                X:=LOWER+DELTA_X*(2.0*I-1.0);
                ODD_SUM:=ODD_SUM+FX(X)
            END;
SUM:=(7.0*END_SUM+14.0*EVEN_SUM+16.00*ODD_SUM+END_COR*DELTA_X)*DELTA_X/15.0;
    UNTIL (SUM<>SUM1) AND (ABS(SUM-SUM1)<=ABS(TOL*SUM))
END; { SIMPS }
BEGIN { MAIN PROGRAM }
    LOWER:=1.0;
    UPPER:=9.0;
    SIMPS(LOWER,UPPER,TOL,SUM);
    WRITELN;
END.
```

ПРИЛОЖЕНИЕ Б.

КОД ПРОГРАММЫ НА ЯЗЫКЕ СИ.

```
#INCLUDE <STDIO.H>
#INCLUDE <STDLIB.H>
#INCLUDE "MATH.H"

CONST DOUBLE TOL= 1.0E-6;

DOUBLE FX(DOUBLE X){
      RETURN EXP(-1.0*X/2.0);
}

DOUBLE DFX(DOUBLE X){
      RETURN (-1.0*EXP(-1.0*X/2.0)/2.0);
}

DOUBLE SIMPS(DOUBLE LOWER,DOUBLE UPPER,DOUBLE TOL,DOUBLE SUM){
      DOUBLE X,DELTA_X,EVEN_SUM,ODD_SUM,END_SUM,END_COR,SUM1;
      INT PIECES;

      PIECES=2;
      DELTA_X=(UPPER-LOWER)/PIECES;
      ODD_SUM=FX(LOWER+DELTA_X);
      EVEN_SUM=0.0;
      END_SUM=FX(LOWER)+FX(UPPER);
      END_COR=DFX(LOWER)-DFX(UPPER);
      SUM=(END_SUM+4.0*ODD_SUM)*DELTA_X/3.0;
      DO{
            PIECES=PIECES*2;
            SUM1=SUM;
            DELTA_X=(UPPER-LOWER)/PIECES;
            EVEN_SUM=EVEN_SUM+ODD_SUM;
            ODD_SUM=0.0;
            FOR (INT I=1;I<=PIECES/2;I++){
                  X=LOWER+DELTA_X*(2.0*I-1.0);
                  ODD_SUM=ODD_SUM+FX(X);
            }

      SUM=(7.0*END_SUM+14.0*EVEN_SUM+16.00*ODD_SUM+END_COR*DELTA_X)*DELTA_X/15.0;
      } WHILE ((SUM!=SUM1) & (ABS(SUM-SUM1)<=ABS(TOL*SUM)));

      RETURN SUM;
}

INT MAIN(VOID) {
      DOUBLE SUM=0.0;
      DOUBLE LOWER=1.0;
      DOUBLE UPPER=9.0;
      DOUBLE RES =0.0;
      RES=SIMPS(LOWER,UPPER,TOL,SUM);
      PRINTF("AREA= %LF ",RES);
      RETURN 0;
}
```

ПРИЛОЖЕНИЕ В.

КОД ПРОГРАММЫ НА ЯЗЫКЕ АССЕМБЛЕР.

```
fx(double):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        movsd   QWORD PTR [rbp-8], xmm0
        movsd   xmm0, QWORD PTR [rbp-8]
        movq    xmm1, QWORD PTR .LC1[rip]
        xorpd   xmm0, xmm1
        movsd   xmm1, QWORD PTR .LC2[rip]
        divsd   xmm0, xmm1
        movq    rax, xmm0
        movq    xmm0, rax
        call    exp
        movq    rax, xmm0
        movq    xmm0, rax
        leave
        ret
dfx(double):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        movsd   QWORD PTR [rbp-8], xmm0
        movsd   xmm0, QWORD PTR [rbp-8]
        movq    xmm1, QWORD PTR .LC1[rip]
        xorpd   xmm0, xmm1
        movsd   xmm1, QWORD PTR .LC2[rip]
        divsd   xmm0, xmm1
        movq    rax, xmm0
        movq    xmm0, rax
        call    exp
        movq    rax, xmm0
        movq    xmm0, QWORD PTR .LC1[rip]
        movq    xmm2, rax
        xorpd   xmm2, xmm0
        movapd  xmm0, xmm2
        movsd   xmm1, QWORD PTR .LC2[rip]
        divsd   xmm0, xmm1
        movq    rax, xmm0
        movq    xmm0, rax
        leave
        ret
simps(double, double, double, double):
        push    rbp
        mov     rbp, rsp
        push    rbx
        sub     rsp, 120
        movsd   QWORD PTR [rbp-88], xmm0
        movsd   QWORD PTR [rbp-96], xmm1
        movsd   QWORD PTR [rbp-104], xmm2
        movsd   QWORD PTR [rbp-112], xmm3
        mov     DWORD PTR [rbp-36], 2
        movsd   xmm0, QWORD PTR [rbp-96]
        subsd   xmm0, QWORD PTR [rbp-88]
        pxor    xmm1, xmm1
        cvtsi2sd        xmm1, DWORD PTR [rbp-36]
        divsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-48], xmm0
        movsd   xmm0, QWORD PTR [rbp-88]
        addsd   xmm0, QWORD PTR [rbp-48]
        movq    rax, xmm0
```

```
        movq    xmm0, rax
        call    fx(double)
        movq    rax, xmm0
        mov     QWORD PTR [rbp-32], rax
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-24], xmm0
        mov     rax, QWORD PTR [rbp-88]
        movq    xmm0, rax
        call    fx(double)
        movsd   QWORD PTR [rbp-120], xmm0
        mov     rax, QWORD PTR [rbp-96]
        movq    xmm0, rax
        call    fx(double)
        addsd   xmm0, QWORD PTR [rbp-120]
        movsd   QWORD PTR [rbp-56], xmm0
        mov     rax, QWORD PTR [rbp-88]
        movq    xmm0, rax
        call    dfx(double)
        movq    rbx, xmm0
        mov     rax, QWORD PTR [rbp-96]
        movq    xmm0, rax
        call    dfx(double)
        movapd  xmm1, xmm0
        movq    xmm0, rbx
        subsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-64], xmm0
        movsd   xmm1, QWORD PTR [rbp-32]
        movsd   xmm0, QWORD PTR .LC4[rip]
        mulsd   xmm0, xmm1
        addsd   xmm0, QWORD PTR [rbp-56]
        mulsd   xmm0, QWORD PTR [rbp-48]
        movsd   xmm1, QWORD PTR .LC5[rip]
        divsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-112], xmm0
.L11:
        sal     DWORD PTR [rbp-36]
        movsd   xmm0, QWORD PTR [rbp-112]
        movsd   QWORD PTR [rbp-72], xmm0
        movsd   xmm0, QWORD PTR [rbp-96]
        subsd   xmm0, QWORD PTR [rbp-88]
        pxor    xmm1, xmm1
        cvtsi2sd        xmm1, DWORD PTR [rbp-36]
        divsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-48], xmm0
        movsd   xmm0, QWORD PTR [rbp-24]
        addsd   xmm0, QWORD PTR [rbp-32]
        movsd   QWORD PTR [rbp-24], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-32], xmm0
        mov     DWORD PTR [rbp-40], 1
.L9:
        mov     eax, DWORD PTR [rbp-36]
        mov     edx, eax
        shr     edx, 31
        add     eax, edx
        sar     eax
        cmp     DWORD PTR [rbp-40], eax
        jg      .L8
        pxor    xmm0, xmm0
        cvtsi2sd        xmm0, DWORD PTR [rbp-40]
        addsd   xmm0, xmm0
        movsd   xmm1, QWORD PTR .LC6[rip]
        subsd   xmm0, xmm1
        mulsd   xmm0, QWORD PTR [rbp-48]
```

```
        movsd   xmm1, QWORD PTR [rbp-88]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-80], xmm0
        mov     rax, QWORD PTR [rbp-80]
        movq    xmm0, rax
        call    fx(double)
        movsd   xmm1, QWORD PTR [rbp-32]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-32], xmm0
        add     DWORD PTR [rbp-40], 1
        jmp     .L9
.L8:
        movsd   xmm1, QWORD PTR [rbp-56]
        movsd   xmm0, QWORD PTR .LC7[rip]
        mulsd   xmm1, xmm0
        movsd   xmm2, QWORD PTR [rbp-24]
        movsd   xmm0, QWORD PTR .LC8[rip]
        mulsd   xmm0, xmm2
        addsd   xmm1, xmm0
        movsd   xmm2, QWORD PTR [rbp-32]
        movsd   xmm0, QWORD PTR .LC9[rip]
        mulsd   xmm0, xmm2
        addsd   xmm1, xmm0
        movsd   xmm0, QWORD PTR [rbp-64]
        mulsd   xmm0, QWORD PTR [rbp-48]
        addsd   xmm0, xmm1
        mulsd   xmm0, QWORD PTR [rbp-48]
        movsd   xmm1, QWORD PTR .LC10[rip]
        divsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-112], xmm0
        movsd   xmm0, QWORD PTR [rbp-112]
        ucomisd xmm0, QWORD PTR [rbp-72]
        setp    al
        mov     edx, 1
        movsd   xmm0, QWORD PTR [rbp-112]
        ucomisd xmm0, QWORD PTR [rbp-72]
        mov     ebx, edx
        cmove   ebx, eax
        movsd   xmm0, QWORD PTR [rbp-112]
        subsd   xmm0, QWORD PTR [rbp-72]
        movq    rax, xmm0
        movq    xmm0, rax
        call    std::abs(double)
        movsd   QWORD PTR [rbp-120], xmm0
        movsd   xmm0, QWORD PTR [rbp-104]
        movapd  xmm4, xmm0
        mulsd   xmm4, QWORD PTR [rbp-112]
        movq    rax, xmm4
        movq    xmm0, rax
        call    std::abs(double)
        movq    rax, xmm0
        movq    xmm6, rax
        comisd  xmm6, QWORD PTR [rbp-120]
        setnb   al
        and     eax, ebx
        movzx   eax, al
        test    eax, eax
        setne   al
        test    al, al
        je      .L10
        jmp     .L11
.L10:
        movsd   xmm0, QWORD PTR [rbp-112]
        movq    rax, xmm0
```

```
        movq    xmm0, rax
        mov     rbx, QWORD PTR [rbp-8]
        leave
        ret
.LC13:
        .string "area= %lf "
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 32
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-8], xmm0
        movsd   xmm0, QWORD PTR .LC6[rip]
        movsd   QWORD PTR [rbp-16], xmm0
        movsd   xmm0, QWORD PTR .LC11[rip]
        movsd   QWORD PTR [rbp-24], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-32], xmm0
        movsd   xmm2, QWORD PTR [rbp-8]
        movsd   xmm1, QWORD PTR .LC12[rip]
        movsd   xmm0, QWORD PTR [rbp-24]
        mov     rax, QWORD PTR [rbp-16]
        movapd  xmm3, xmm2
        movapd  xmm2, xmm1
        movapd  xmm1, xmm0
        movq    xmm0, rax
        call    simps(double, double, double, double)
        movq    rax, xmm0
        mov     QWORD PTR [rbp-32], rax
        mov     rax, QWORD PTR [rbp-32]
        movq    xmm0, rax
        mov     edi, OFFSET FLAT:.LC13
        mov     eax, 1
        call    printf
        mov     eax, 0
        leave
        ret
.LC0:
        .long   -1
        .long   2147483647
        .long   0
        .long   0
.LC1:
        .long   0
        .long   -2147483648
        .long   0
        .long   0
.LC2:
        .long   0
        .long   1073741824
.LC4:
        .long   0
        .long   1074790400
.LC5:
        .long   0
        .long   1074266112
.LC6:
        .long   0
        .long   1072693248
.LC7:
        .long   0
        .long   1075576832
.LC8:
        .long   0
```

```
        .long   1076625408
.LC9:
        .long   0
        .long   1076887552
.LC10:
        .long   0
        .long   1076756480
.LC11:
        .long   0
        .long   1075970048
.LC12:
        .long   -1598689907
        .long   1051772663
```

ПРИЛОЖЕНИЕ Г.

РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

```
Statistics for module output_pascal.lxm
=====================================
The number of different operators:  25
The number of different operands: 31
The total number of operators: 164
The total number of operands  : 105

Dictionary  ( D)  : 56
Length      ( N)  : 269
Length estimation ( ^N) : 269.676
Volume      ( V)  : 1562.18
Potential volume  ( *V) : 19.6515
Limit volume      (**V) : 38.2071
Programming level ( L)  : 0.0125795
Programming level estimation ( ^L)  : 0.02361
Intellect   ( I)  : 36.8972
Time of programming           ( T)  : 6899.12
Time estimation   ( ^T) : 3683.72
Programming language level          (lambda) : 0.247207
Work on programming     ( E)  : 124184
Error ( B)  : 0.829703
Error estimation  ( ^B) : 0.520726

Table:
===================================
Operators:

|     1     |     24    | ()
|     2     |     11    | *
|     3     |     9     | +
|     4     |     8     | -
|     5     |     8     | /
|     6     |     47    | ;
|     7     |     1     | <=
|     8     |     1     | <>
|     9     |     20    | =
|     10    |     2     | abs
|     11    |     1     | and
|     12    |     1     | chr
|     13    |     1     | const
|     14    |     3     | dfx
|     15    |     2     | exp
|     16    |     1     | for
|     17    |     2     | function
|     18    |     5     | fx
|     19    |     2     | integer
|     20    |     1     | procedure
|     21    |     1     | program
|     22    |     8     | real
|     23    |     1     | repeat
|     24    |     2     | simps
| 25  | 2     | writeln

Operands:
|     1     |     1     | 'area= '
```

2	2	0.0
3	1	1
4	2	1.0
5	1	1.0E-6
6	1	14.0
7	1	15.0
8	1	16.00
9	6	2
10	1	2.0
11	1	3.0
12	1	4.0
13	1	7
14	1	7.0
15	1	9.0
16	8	delta_x
17	1	dfx
18	3	end_cor
19	4	end_sum
20	5	even_sum
21	1	fx
22	2	i
23	10	lower
24	8	odd_sum
25	7	pieces
26	1	simp1
27	10	sum
28	4	sum1
29	4	tol
30	8	upper
31	7	

ПРИЛОЖЕНИЕ Д.

РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ СИ.

```
STATISTICS FOR MODULE OUTPUT__C.LXM
===================================
THE NUMBER OF DIFFERENT OPERATORS: 27
THE NUMBER OF DIFFERENT OPERANDS: 29
THE TOTAL NUMBER OF OPERATORS: 168
THE TOTAL NUMBER OF OPERANDS: 110
DICTIONARY  ( D)  : 56
LENGTH      ( N)  : 278
LENGTH ESTIMATION( ^N) : 269.263
VOLUME      ( V)  : 1614.44
POTENTIAL VOLUME  ( *V) : 19.6515
LIMIT VOLUME      (**V) : 38.2071
PROGRAMMING LEVEL( L)  : 0.0121723
PROGRAMMING LEVEL ESTIMATION ( ^L): 0.0195286
INTELLECT   ( I)  : 31.5279
TIME OF PROGRAMMING          ( T)  : 7368.49
TIME ESTIMATION   ( ^T) : 4448.48
PROGRAMMING LANGUAGE LEVEL         (LAMBDA) : 0.239204
WORK ON PROGRAMMING    ( E)  : 132633
ERROR ( B)  : 0.866921
ERROR ESTIMATION  ( ^B) : 0.538148

TABLE:
===================================
OPERATORS:

|     1     |     1     | !=
|     2     |     1     | &
|     3     |     14    | ()
|     4     |     14    | *
|     5     |     9     | +
|     6     |     1     | ++
|     7     |     13    | ,
|     8     |     5     | -
|     9     |     8     | /
|     10    |     33    | ;
|     11    |     2     | <=
|     12    |     22    | =
|     13    |     3     | _-
|     14    |     2     | ABS
|     15    |     1     | CONST
|     16    |     3     | DFX
|     17    |     15    | DOUBLE
|     18    |     1     | DOWHILE
|     19    |     2     | EXP
|     20    |     1     | FOR
|     21    |     5     | FX
|     22    |     3     | INT
|     23    |     1     | MAIN
|     24    |     1     | PRINTF
|     25    |     4     | RETURN
|     26    |     2     | SIMPS
|     27    |     1     | VOID

OPERANDS:
|     1     |     1     | "AREA= %LF "
```

	2	1	0
	3	4	0.0
	4	1	1
	5	5	1.0
	6	1	1.0E-6
	7	1	14.0
	8	1	15.0
	9	1	16.00
	10	3	2
	11	4	2.0
	12	1	3.0
	13	1	4.0
	14	1	7.0
	15	1	9.0
	16	8	DELTA_X
	17	3	END_COR
	18	4	END_SUM
	19	5	EVEN_SUM
	20	4	I
	21	9	LOWER
	22	8	ODD_SUM
	23	7	PIECES
	24	3	RES
	25	10	SUM
	26	4	SUM1
	27	4	TOL
	28	7	UPPER
	29	7	X