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 Ажель И.В.

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, а также реализованные программы на языках 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	;	29	**1**	x	14
2	begin…end	5	**2**	x2	1
3	:=	9	**3**	x1	4
4	writeln	5	**4**	alldone	1
5	if… then	1	**5**	fx	6
6	repeat… until	2	**6**	dfx	10
7	+	3	**7**	dx	4
8	write	6	**8**	a	2
9	>	1	**9**	b	3
10	readln	1	**10**	c	3
11	abs	3	**11**	logp	2
12	-	8	**12**	tol	5
13	/	4	**13**	real	2
14	*	3	**14**	false	1
15	<=	1	**15**	true	1
16	()	17	**16**	1.0E-4	1
17	>=	1	**17**	18.19	1
18	ln	1	**18**	23180.0	1
			19	0.8858	1
			20	4.60517	1
			21	0.0	1
			22	999	1

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	log	1	**1**	x	16

2	-	8	**2**	fx	6
3	!	1	**3**	dfx	10
4	if…else	3	**4**	x1	4
5	==	1	**5**	alldone	4
6	=	21	**6**	a	2
7	/	4	**7**	b	3
8	+	3	**8**	c	3
9	<	4	**9**	a	5
10	>	3	**10**	error	1
11	;	26	**11**	tol	5
12	scanf	1	**12**	dx	4
13	*	12	**13**	true	1
14	abs	3	**14**	false	1
15	printf	3	**15**	18.19	1
16	&	4	**16**	23180.0	1
17	()	20	**17**	0.8858	1
18	>=	1	**18**	4.60517	1
19	do…while	2	**19**	0.0	1
20	<=	1	**20**	999	1

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

№	**Оператор**	**Число вхождений**	**№**	**Операнд**	**Число вхождений**
1	push	3	**1**	rbp	6
2	mov	90	**2**	rsp	6
3	movss	42	**3**	QWORD PTR [rbp-40]	7
4	nop	2	**4**	rdi	4
5	movd	8	**5**	QWORD PTR [rbp-32]	2
6	ret	3	**6**	rsi	3
7	sub	4	**7**	rax	14
8	jmp	4	**8**	xmm0	66
9	mulss	3	**9**	DWORD PTR [rax]	6
10	add	4	**10**	DWORD PTR [rbp-44]	4
11	addss	4	**11**	48	2
12	cmp	1	**12**	DWORD PTR [rbp-20]	12
13	divss	4	**13**	xmm1	31
14	call func	1	**14**	eax	18
15	lea	7	**15**	DWORD PTR [rbp-8]	6
16	comiss	5	**16**	xmm2	4

17	je	2	**17**	16	1
18	pxor	4	**18**	.LC12	1
19	jb	2	**19**	BYTE PTR [rbp-1]	11
20	seta	1	**20**	1	3
21	xor	5	**21**	DWORD PTR [rbp-12]	4
22	test	2	**22**	.LC11	1
23	jne	2	**23**	0	8
24	movq	4	**24**	DWORD PTR [rbp-16]	4
25	leave	3	**25**	.LC10	1
26	movapd	2	**26**	.LC9	1
27	div	4	**27**	rdx	2
28	setnb	1	**28**	rcx	2
29	cvtss2sd	4	**29**	.L7	3
30	ucomiss	2	**30**	al	10
			31	.LC1	1
			32	.LC2	1
			33	.LC3	1
			34	.LC4	1
			35	.LC5	1
			36	.LC6	1
			37	.LC7	1
			38	.LC8	1

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	18	20	30
Число уникальных операндов	20	20	38
Общее число операторов	101	122	174
Общее число операндов	65	72	251
Алфавит	38	40	68
Экспериментальная длина программы	166	194	425
Теоретическая длина программы	161,16	172,8	414,5
Объем программы	871,168	1032,274	2876,055
Потенциальный объем	11,6	11,6	11,6
Уровень программы	0,013	0,011	0,004
Интеллектуальное содержание	29,511	28,66	34,116

Работа по программированию	67012,92	93818,18	719022,54
Ожидание времени кодирования	6701,2	9381,8	71902,2
Уровень языка программирования	0,151	0,128	0,046
Уровень ошибок	1	1	3

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	()	10	**1**	a	2
2	*	3	**2**	alldone	2
3	+	3	**3**	b	3
4	-	7	**4**	c	3
5	/	4	**5**	dfx	10
6	<	1	**6**	dx	4
7	<=	1	**7**	error	1
8	=	14	**8**	false	1
9	>=	1	**9**	fx	6
10	abs	3	**10**	logp	2
11	const	2	**11**	newdr	1
12	func	2	**12**	tol	5
13	if	1	**13**	x	14
14	ln	1	**14**	x1	4
15	newton	1	**15**	x2	1
16	program	1	**16**	0.0	1
17	readln	1	**17**	0.8858	1
18	repeat	1	**18**	1.0E-4	1
19	write	1	**19**	18.19	1
20	writeln	2	**20**	23180.0	1
			21	4.60517	1
			22	999	1

			23	',dfx='	1
			24	',fx='	1
			25	'First guess (999. to exit): '	1
			26	'x='	1

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

№	**Оператор**	**Число вхождений**	**№**	**Операнд**	**Число вхождений**
1	!	1	**1**	"x= %f fx = %f dfx = %f \n"	1
2	()	6	**2**	0.0	1
3	*	3	**3**	0.8858	1
4	+	3	**4**	18.19	1
5	,	11	**5**	1e-4	1
6	-	2	**6**	23180.0	1
7	/	4	**7**	4.60517	1
8	;	17	**8**	a	2
9	<	1	**9**	b	3
10	<=	1	**10**	c	3
11	=	12	**11**	dfx	10
12	>=	1	**12**	dx	4
13	_&	2	**13**	fx	6
14	_*	6	**14**	logp	2
15	_-	5	**15**	tol	5
16	__*	3	**16**	x	12
17	const	5	**17**	x1	4
18	do…while	1	**18**	x2	1
19	fabs	3			
20	float	11			
21	func	2			
22	if	2			
23	int	1			
24	log	1			
25	newton	1			
26	void	2			
27	printf	1			

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	20	27
Число уникальных операндов	26	18
Общее число операторов	61	108
Общее число операндов	70	59
Алфавит	46	45
Экспериментальная длина программы	131	167
Теоретическая длина программы	208.65	203,441
Объем программы	723.587	917,139
Потенциальный объем	19,6515	19,6515
Уровень программы	0,0271584	0,0214269
Интеллектуальное содержание	26,8761	20,7263
Ожидание уровня программы	0,0371429	0,0225998
Работа по программированию	26643.2	42803.1
Ожидание времени кодирования	1480.18	2377.95
Уровень языка программирования	0,533704	0,421071
Уровень ошибок	1	1

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	18	20	20	27
Число уникальных операндов	20	26	20	18
Общее число операторов	101	61	122	108
Общее число операндов	65	70	72	59
Алфавит	38	46	40	45

Экспериментальная длина программы	166	131	194	167
Теоретическая длина программы	161,16	208.65	172,8	203,441
Объем программы	871,168	723.587	1032,274	917,139
Потенциальный объем	11,6	19,6515	11,6	19,6515
Уровень программы	0,013	0,0271584	0,011	0,0214269
Интеллектуальное содержание	29,511	26,8761	28,66	20,7263
Ожидание уровня программы	0,018	0,0371429	0,012	0,0225998
Работа по программированию	67012,92	26643.2	93818,18	42803.1
Ожидание времени кодирования	6701,2	1480.18	9381,8	2377.95
Уровень языка программирования	0,151	0,533704	0,128	0,421071
Уровень ошибок	1	1	1	1

Выводы:

Метрические характеристики программ, написанных на языках Си и Паскаль выглядят похожим образом, так они имеют схожую структуру. Характеристики программы на языке Ассемблер сильно отличаются. Это связано с тем, что язык Ассемблер является языком низкого уровня.

В ходе выполнения данной работы все характеристики были посчитаны вручную и автоматически. Различия в полученных результатах обусловлены тем, что автоматический метод считает не только функциональную часть программы, но и объявления типов переменных и функций. Также различия для программы на языке Си обусловлены тем, что инструмент автоматического подсчета не имеет возможности обработки типа данных bool, который присутствует в коде программы.

ПРИЛОЖЕНИЕ А.

КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

```
program newdr;
var  x,x2 : real;
alldone        : boolean;
error     : boolean;

procedure func(x: real;
var fx,dfx: real);

{ the vapor pressure of lead }
const
   a = 18.19;
```

```
   b = -23180.0;
   c = -0.8858;
   logp = -4.60517;  { ln(.01) }
begin
  fx:= a + b/x + c*ln(x) - logp;
  dfx:= -b/(x*x) + c/x
end; { func }

procedure newton(var x: real);
const     tol  = 1.0E-4;
var  fx,dfx,dx,x1:  real;
begin{ newton }
  repeat
    x1:=x;
    func(x,fx,dfx);
    if(abs(dfx)<tol) then
       begin
          if(dfx>=0.0) then dfx:=tol
          else dfx := -tol
       end;
    dx:=fx/dfx;
    x:=x1-dx;
    writeln('x=',x1,',fx=',fx,',dfx=',dfx);
  until abs(dx)<=abs(tol*x)
end; { newton }

begin     { main program }
  alldone:=false;
  repeat
    writeln;
    write('First guess (999. to exit): ');   { first guess }
    readln(x);
    if x=999. then alldone:=true
    else
      begin
newton(x);
writeln;
writeln('The solution is ',x);
writeln
      end
  until alldone
end.
```

ПРИЛОЖЕНИЕ Б.

КОД ПРОГРАММЫ НА ЯЗЫКЕ СИ.

```
#include <math.h>
#include <stdio.h>
#include <stdbool.h>

void func(float x, float*fx, float*dfx) {
     const float a = 18.19;
     const float b = -23180.0;
     const float c = -0.8858;
     const float logp = -4.60517;

     *fx = a + b / x + c * log(x) - logp;
     *dfx = -b / (x * x) + c / x;
}

void newton(float* x) {
     const float tol = 1e-4;
     float fx, dfx, dx, x1;
     do {
          x1 = *x;
          func(*x, &fx, &dfx);
          if (fabs(dfx) < tol) {
               if (dfx >= 0.0) dfx = tol;
               else dfx = -tol;
          }
          dx = fx / dfx;
          *x = x1 - dx;
          printf("x= %f fx = %f dfx = %f \n", x1, fx, dfx);
     } while (!(fabs(dx) <= fabs(tol * *x)));
}

int main() {
     float x, x2;
     bool alldone;
     bool error;
```

```
    alldone = false;
    do {
        printf("First guess (999. to exit): ");
        scanf("%f", &x);
        if (x == 999.0) {
            alldone = true;
        } else {
            newton(&x);
            printf("\nThe solution is %f \n", x);

        }
    } while (!alldone);
}
```

ПРИЛОЖЕНИЕ В.

КОД ПРОГРАММЫ НА ЯЗЫКЕ АССЕМБЛЕР.

```
func(float, float*, float*):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 48
        movss   DWORD PTR [rbp-20], xmm0
        mov     QWORD PTR [rbp-32], rdi
        mov     QWORD PTR [rbp-40], rsi
        movss   xmm0, DWORD PTR .LC1[rip]
        movss   DWORD PTR [rbp-4], xmm0
```

```
movss   xmm0, DWORD PTR .LC2[rip]
movss   DWORD PTR [rbp-8], xmm0
movss   xmm0, DWORD PTR .LC3[rip]
movss   DWORD PTR [rbp-12], xmm0
movss   xmm0, DWORD PTR .LC4[rip]
movss   DWORD PTR [rbp-16], xmm0
movss   xmm0, DWORD PTR .LC2[rip]
movaps  xmm1, xmm0
divss   xmm1, DWORD PTR [rbp-20]
movss   xmm0, DWORD PTR .LC1[rip]
addss   xmm1, xmm0
movss   DWORD PTR [rbp-44], xmm1
mov     eax, DWORD PTR [rbp-20]
movd    xmm0, eax
call    std::log(float)
movss   xmm1, DWORD PTR .LC3[rip]
mulss   xmm0, xmm1
movss   xmm1, DWORD PTR [rbp-44]
addss   xmm1, xmm0
movss   xmm0, DWORD PTR .LC5[rip]
addss   xmm0, xmm1
mov     rax, QWORD PTR [rbp-32]
movss   DWORD PTR [rax], xmm0
movss   xmm0, DWORD PTR [rbp-20]
movaps  xmm2, xmm0
mulss   xmm2, xmm0
movss   xmm0, DWORD PTR .LC6[rip]
movaps  xmm1, xmm0
divss   xmm1, xmm2
movss   xmm0, DWORD PTR .LC3[rip]
divss   xmm0, DWORD PTR [rbp-20]
addss   xmm0, xmm1
mov     rax, QWORD PTR [rbp-40]
movss   DWORD PTR [rax], xmm0
nop
```

```
        leave
        ret
.LC10:
        .string "x= %f fx = %f dfx = %f \n"
newton(float*):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 48
        mov     QWORD PTR [rbp-40], rdi
        movss   xmm0, DWORD PTR .LC7[rip]
        movss   DWORD PTR [rbp-4], xmm0
.L11:
        mov     rax, QWORD PTR [rbp-40]
        movss   xmm0, DWORD PTR [rax]
        movss   DWORD PTR [rbp-8], xmm0
        mov     rax, QWORD PTR [rbp-40]
        mov     eax, DWORD PTR [rax]
        lea     rcx, [rbp-20]
        lea     rdx, [rbp-16]
        mov     rsi, rcx
        mov     rdi, rdx
        movd    xmm0, eax
        call    func(float, float*, float*)
        mov     eax, DWORD PTR [rbp-20]
        movd    xmm0, eax
        call    std::fabs(float)
        movss   xmm1, DWORD PTR .LC7[rip]
        comiss  xmm1, xmm0
        seta    al
        test    al, al
        je      .L7
        movss   xmm0, DWORD PTR [rbp-20]
        pxor    xmm1, xmm1
        comiss  xmm0, xmm1
        jb      .L13
```

```
        movss   xmm0, DWORD PTR .LC7[rip]
        movss   DWORD PTR [rbp-20], xmm0
        jmp     .L7
.L13:
        movss   xmm0, DWORD PTR .LC9[rip]
        movss   DWORD PTR [rbp-20], xmm0
.L7:
        movss   xmm0, DWORD PTR [rbp-16]
        movss   xmm1, DWORD PTR [rbp-20]
        divss   xmm0, xmm1
        movss   DWORD PTR [rbp-12], xmm0
        movss   xmm0, DWORD PTR [rbp-8]
        subss   xmm0, DWORD PTR [rbp-12]
        mov     rax, QWORD PTR [rbp-40]
        movss   DWORD PTR [rax], xmm0
        movss   xmm0, DWORD PTR [rbp-20]
        pxor    xmm1, xmm1
        cvtss2sd        xmm1, xmm0
        movss   xmm0, DWORD PTR [rbp-16]
        cvtss2sd        xmm0, xmm0
        pxor    xmm3, xmm3
        cvtss2sd        xmm3, DWORD PTR [rbp-8]
        movq    rax, xmm3
        movapd  xmm2, xmm1
        movapd  xmm1, xmm0
        movq    xmm0, rax
        mov     edi, OFFSET FLAT:.LC10
        mov     eax, 3
        call    printf
        mov     eax, DWORD PTR [rbp-12]
        movd    xmm0, eax
        call    std::fabs(float)
        movss   DWORD PTR [rbp-44], xmm0
        mov     rax, QWORD PTR [rbp-40]
        movss   xmm1, DWORD PTR [rax]
```

```
        movss   xmm0, DWORD PTR .LC7[rip]
        mulss   xmm1, xmm0
        movd    eax, xmm1
        movd    xmm0, eax
        call    std::fabs(float)
        movd    eax, xmm0
        movd    xmm4, eax
        comiss  xmm4, DWORD PTR [rbp-44]
        setnb   al
        xor     eax, 1
        test    al, al
        je      .L14
        jmp     .L11
.L14:
        nop
        leave
        ret
.LC11:
        .string "First guess\t(999. to exit): "
.LC12:
        .string "%f"
.LC14:
        .string "\nThe solution is %f \n"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     BYTE PTR [rbp-1], 0
.L20:
        mov     edi, OFFSET FLAT:.LC11
        mov     eax, 0
        call    printf
        lea     rax, [rbp-8]
        mov     rsi, rax
        mov     edi, OFFSET FLAT:.LC12
```

```
        mov     eax, 0
        call    __isoc99_scanf
        movss   xmm0, DWORD PTR [rbp-8]
        ucomiss xmm0, DWORD PTR .LC13[rip]
        jp      .L16
        ucomiss xmm0, DWORD PTR .LC13[rip]
        jne     .L16
        mov     BYTE PTR [rbp-1], 1
        jmp     .L18
.L16:
        lea     rax, [rbp-8]
        mov     rdi, rax
        call    newton(float*)
        movss   xmm0, DWORD PTR [rbp-8]
        pxor    xmm1, xmm1
        cvtss2sd        xmm1, xmm0
        movq    rax, xmm1
        movq    xmm0, rax
        mov     edi, OFFSET FLAT:.LC14
        mov     eax, 1
        call    printf
.L18:
        cmp     BYTE PTR [rbp-1], 0
        jne     .L19
        jmp     .L20
.L19:
        mov     eax, 0
        leave
        ret
.LC0:
        .long   2147483647
        .long   0
        .long   0
        .long   0
.LC1:
```

```
        .long   1100055839
.LC2:
        .long   -961210368
.LC3:
        .long   -1084046390
.LC4:
        .long   -1064084083
.LC5:
        .long   1083399565
.LC6:
        .long   1186273280
.LC7:
        .long   953267991
.LC9:
        .long   -1194215657
.LC13:
        .long   1148829696
```

ПРИЛОЖЕНИЕ Г.

РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

```
Statistics for module pas.lxm
====================================
The number of different operators     : 20
The number of different operands      : 26
The total number of operators         : 61
The total number of operands          : 70

Dictionary                  (  D)     : 46
```

```
Length                          (  N)     : 131
Length estimation               ( ^N)     : 208.65
Volume                          (  V)     : 723.587
Potential volume                ( *V)     : 19.6515
Limit volume                    (**V)     : 38.2071
Programming level               (  L)     : 0.0271584
Programming level estimation    ( ^L)     : 0.0371429
Intellect                       (  I)     : 26.8761
Time of programming             (  T)     : 1480.18
Time estimation                 ( ^T)     : 1723.81
Programming language level      (lambda) : 0.533704
Work on programming             (  E)     : 26643.2
Error                           (  B)     : 0.297351
Error estimation                ( ^B)     : 0.241196
```

Table:

==========

Operators:

1	10	()
2	3	*
3	3	+
4	7	-
5	4	/
6	1	<
7	1	<=
8	14	=
9	1	>=
10	3	abs
11	2	const
12	2	func
13	2	if
14	1	ln
15	1	newton
16	1	program

```
|  17   |  1    | readln
|  18   |  1    | repeat
|  19   |  1    | write
|  20   |  2    | writeln
Operands:
|  1    |  1    | ',dfx='
|  2    |  1    | ',fx='
|  3    |  1    | 'First guess (999. to exit): '
|  4    |  1    | 'x='
|  5    |  1    | 0.0
|  6    |  1    | 0.8858
|  7    |  1    | 1.0E-4
|  8    |  1    | 18.19
|  9    |  1    | 23180.0
|  10   |  1    | 4.60517
|  11   |  1    | 999
|  12   |  2    | a
|  13   |  2    | alldone
|  14   |  3    | b
|  15   |  3    | c
|  16   |  10   | dfx
|  17   |  4    | dx
|  18   |  1    | error
|  19   |  1    | false
|  20   |  6    | fx
|  21   |  2    | logp
|  22   |  1    | newdr
|  23   |  5    | tol
|  24   |  14   | x
|  25   |  4    | x1
|  26   |  1    | x2


Summary:
====================================
```

```
The number of different operators     : 20
The number of different operands      : 26
The total number of operators         : 61
The total number of operands          : 70

Dictionary                   (  D)    : 46
Length                       (  N)    : 131
Length estimation            ( ^N)    : 208.65
Volume                       (  V)    : 723.587
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.0271584
Programming level estimation ( ^L)    : 0.0371429
Intellect                    (  I)    : 26.8761
Time of programming          (  T)    : 1480.18
Time estimation              ( ^T)    : 1723.81
Programming language level   (lambda) : 0.533704
Work on programming          (  E)    : 26643.2
Error                        (  B)    : 0.297351
Error estimation             ( ^B)    : 0.241196
```

ПРИЛОЖЕНИЕ Д.

РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ СИ.

```
Statistics for module output.lxm
====================================
The number of different operators     : 27
The number of different operands      : 18
The total number of operators         : 108
The total number of operands          : 59

Dictionary                   (  D)    : 45
Length                       (  N)    : 167
Length estimation            ( ^N)    : 203.441
```

```
Volume                          (  V)     : 917.139
Potential volume                ( *V)     : 19.6515
Limit volume                    (**V)     : 38.2071
Programming level               (  L)     : 0.0214269
Programming level estimation    ( ^L)     : 0.0225989
Intellect                       (  I)     : 20.7263
Time of programming             (  T)     : 2377.95
Time estimation                 ( ^T)     : 2746.61
Programming language level      (lambda)  : 0.421071
Work on programming             (  E)     : 42803.1
Error                           (  B)     : 0.407877
Error estimation                ( ^B)     : 0.305713
```

Table:

```
==================================
Operators:
|  1    |  1    | !
|  2    |  6    | ()
|  3    |  3    | *
|  4    |  3    | +
|  5    |  11   | ,
|  6    |  2    | -
|  7    |  4    | /
|  8    |  17   | ;
|  9    |  1    | <
|  10   |  1    | <=
|  11   |  12   | =
|  12   |  1    | >=
|  13   |  2    | _&
|  14   |  6    | _*
|  15   |  5    | _-
|  16   |  3    | __*
|  17   |  5    | const
|  18   |  1    | dowhile
|  19   |  3    | fabs
|  20   |  11   | float
|  21   |  2    | func
|  22   |  2    | if
|  23   |  1    | int
|  24   |  1    | log
|  25   |  1    | newton
|  26   |  1    | printf
|  27   |  2    | void
```

```
Operands:
|  1    |  1    | "x= %f fx = %f dfx = %f \n"
|  2    |  1    | 0.0
|  3    |  1    | 0.8858
|  4    |  1    | 18.19
|  5    |  1    | 1e-4
|  6    |  1    | 23180.0
|  7    |  1    | 4.60517
|  8    |  2    | a
|  9    |  3    | b
|  10   |  3    | c
|  11   |  10   | dfx
|  12   |  4    | dx
|  13   |  6    | fx
|  14   |  2    | logp
|  15   |  5    | tol
|  16   |  12   | x
|  17   |  4    | x1
|  18   |  1    | x2


Summary:
=====================================
The number of different operators     : 27
The number of different operands      : 18
The total number of operators         : 108
The total number of operands          : 59

Dictionary                   (  D)    : 45
Length                       (  N)    : 167
Length estimation            ( ^N)    : 203.441
Volume                       (  V)    : 917.139
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.0214269
Programming level estimation ( ^L)    : 0.0225989
Intellect                    (  I)    : 20.7263
Time of programming          (  T)    : 2377.95
Time estimation              ( ^T)    : 2746.61
Programming language level   (lambda) : 0.421071
Work on programming          (  E)    : 42803.1
Error                        (  B)    : 0.407877
Error estimation             ( ^B)    : 0.305713
```