

**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 по**  
**метрикам Холстеда»**

Студент гр. 7304

Петруненко Д.А.

Преподаватель

\_\_\_\_\_  
\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2021

## **Цель работы**

Изучение и сравнение метрик Холстеда для программ на C, Pascal и ассемблере.

## **Постановка задачи**

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. Добиться, чтобы программы на Паскале и Си были работоспособны и давали корректные результаты (это потребуется в дальнейшем при проведении с ними измерительных экспериментов). Для получения ассемблерного представления программы можно либо самостоятельно написать код на ассемблере, реализующий заданный алгоритм, либо установить опцию "Code generation/Generate assembler source» при компиляции текста программы, представленной на языке Си. Во втором случае в ассемблерном представлении программы нужно удалить директивы описаний и отладочные директивы, оставив только исполняемые операторы.

В заданных на Паскале вариантах программ обработки данных важен только вычислительный алгоритм, реализуемый программой. Поэтому для получения более корректных оценок характеристик программ следует учитывать только вычислительные операторы и исключить операторы, обеспечивающие интерфейс с пользователем и выдачу текстовых сообщений. В сути алгоритма, реализуемого программой, нужно разобраться достаточно хорошо для возможности внесения в программу модификаций, выполняемых в дальнейшем при проведении измерений и улучшении характеристик качества программы.

Для измеряемых версий программ в дальнейшем будет нужно исключить операции ввода данных с клавиатуры и вывода на печать, потребляющие основную долю ресурса времени при выполнении программы.

Поэтому можно уже в этой работе предусмотреть соответствующие преобразования исходной программы.

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.

Для характеристик длина программы, уровень программы, время программирования следует рассчитать как саму характеристику, так и ее оценку.

Расчет характеристик программ и их оценок выполнить двумя способами:

- 1) вручную (с калькулятором) или с помощью одного из доступных средств математических вычислений EXCEL, MATHCAD или MATLAB. Для программы на Ассемблере возможен только ручной расчет характеристик. При ручном расчете, в отличие от программного, нужно учитывать только выполняемые операторы, а все описания не учитываются. Соответственно все символы («;», «=», переменные, цифры), входящие в описания, не учитываются.
- 2) с помощью программы автоматизации расчета метрик Холстеда (для Си- и Паскаль-версий программ), краткая инструкция по работе с которой приведена в файле user\_guide.

Для варианта расчета с использованием программы автоматизации желательно провести анализ влияния учета тех или иных групп операторов исследуемой программы на вычисляемые характеристики за счет задания разных ключей запуска.

При настройке параметров (ключей) запуска программы автоматизации следует задать корректное значение числа внешних связей 2\* анализируемой программы (по умолчанию задается 5), совпадающее с используемым при ручном расчете.

Результаты расчетов представить в виде таблиц с текстовыми комментариями:

1. Паскаль. Ручной расчёт:
  - a. Измеримые характеристики,
  - b. Расчетные характеристики
2. Паскаль. Программный расчет:
  - a. Измеримые характеристики
  - b. Расчетные характеристики

3. Си. Ручной расчет:
  - a. Измеримые характеристики,
  - b. Расчетные характеристики
4. Си. Программный расчет:
  - a. Измеримые характеристики,
  - b. Расчетные характеристики
5. Ассемблер. Ручной расчет:
  - a. Измеримые характеристики,
  - b. Расчетные характеристики
6. Сводная таблица расчетов для трех языков

## Ход работы

### 1. Расчет метрик вручную

Программа на языке Паскаль, С и Assembler представлены в приложениях А, Б и В, соответственно.

В таблицах 1-3 представлены результаты подсчета числа типов операторов и операндов в программах на языке Паскаль, С и Assembler.

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	;	17	1	1.0E-4	1
2	begin... end	7	2	done	6
3	:=	35	3	sum	8
4	for...to...do	2	4	upper	7
5	if...then	4	5	lower	8
6	repeat...until	1	6	x	5
7	+	8	7	ans	2
8	-	9	8	nx	5

9	>	3	9	t	16
10	[]	21	10	error	3
11	*	9	11	pieces	8
12	fx	1	12	nt	3
13	romb	1	13	fotom	4
14	<	1	14	delta_x	3
15	<=	2	15	c	4
16	div	1	16	n	10
17	()	22	17	nn	11

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	fx	4	1	0,0001	1
2	romb	2	2	n	13
3	=	45	3	nn	12
4	;	8	4	tol	3
5	for	4	5	done	5
6	If...else ...	4	6	sumMain	3
7	<	3	7	upper	7
8	[]	24	8	lower	8
9	*	9	9	nx	6
10	>	3	10	pieces	8
11	do while	1	11	delta_x	6
12	+	8	12	c	4
13	-	10	13	sum	5
14	++	4	14	fotom	4
15	<=	4	15	x	5
16	/	5	16	nt	3

17	!	1	17	ntra	3
18	return	2			
19		2			

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	push	3	1	rbp	8
2	mov	78	2	rsp	4
3	movsd	49	3	QWORD PTR [rbp-8]	2
4	divsd	4	4	xmm0	83
5	movq	21	5	QWORD PTR .LC1[rip]	3
6	pop	2	6	rax	46
7	ret	3	7	1280	1
8	sub	20	8	QWORD PTR [rbp-1256]	1
9	cmp	7	9	xmm1	37
10	jg	4	10	DWORD PTR [rbp-4]	8
11	cdqe	20	11	0	12
12	add	12	12	15	2
13	jmp	7	13	1	37
14	pxor	6	14	eax	74
15	subsd	6	15	DWORD PTR [rbp-160+rax*4]	4
16	cvttsi2sd	3	16	135	1
17	call	13	17	QWORD PTR [rbp-1248+rax*8]	16
18	addsd	3	18	DWORD PTR [rbp-12]	6
19	mulsd	7	19	DWORD PTR [rbp-160]	1

20	sal	1	20	QWORD PTR [rbp-48]	5
21	shr	1	21	QWORD PTR [rbp-1272]	6
22	sar	1	22	QWORD PTR .LC3[rip]	1
23	ja	1	23	QWORD PTR [rbp-56]	3
24	jbe	1	24	QWORD PTR [rbp-1248]	1
25	jnb	1	25	DWORD PTR [rbp-20]	1
26	test	1	26	QWORD PTR .LC4[rip]	1
27	je	1	27	QWORD PTR [rbp-64]	1
28	jle	2	28	edx	12
29	Jne	1	29	DWORD PTR [rbp-68]	68
			30	DWORD PTR [rbp-16]	4
			31	31	1
			32	DWORD PTR [rbp-72]	2
			33	QWORD PTR [rbp-80]	2
			34	DWORD PTR [rbp-84]	2
			35	DWORD PTR [rbp-88]	2
			36	DWORD PTR [rbp-28]	5
			37	DWORD PTR [rbp-32]	9
			38	DWORD PTR [rbp-92]	2
			39	2	5
			40	QWORD PTR tol[rip]	2
			41	BYTE PTR [rbp-5]	3
			42	QWORD PTR sumMain[rip]	2
			43	QWORD PTR .LC7[rip]	1
			44	QWORD PTR upper[rip]	2



В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную

	<b>Паскаль</b>	<b>Си</b>	<b>Ассемблер</b>
Число уникальных операторов (n1):	17	19	29
Число уникальных операндов (n2):	17	17	44
Общее число операторов (N1):	144	147	249
Общее число операндов (N2):	104	96	489
Алфавит (n):	34	36	61
Экспериментальная длина программы (Nэ):	248	243	738
Теоретическая длина программы (Nт):	138.973	150.1974	300.881
Объём программы (V):	1261.690784	256.2917	4376.88415
Потенциальный объём (V*):	19.651484	11.60964	11.6096404
Уровень программы (L):	0.015575	0.0092411	0.0026524
Интеллектуальное содержание (I):	24.263284	23.417719	19.7532322
Работа по программированию (E):	81004.7525	135944.6942	1650104.06
Время кодирования (T):	8100.4752	13594.46942	165010.4062
Уровень языка программирования (Lambda):	0.30682	0.107286	0.0307944
Уровень ошибок (B):	2	2	5

## 2. Расчет метрик с помощью программы автоматизации

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	;	25	1	upper	7
2	const	1	2	lower	8
3	=	35	3	sum	8
4	or	1	4	nx	5
5	[]	21	5	t	16
6	repeat	1	6	x	5
7	+	8	7	ans	2
8	-	9	8	i	5
9	fx	1	9	false	3
10	romb	1	10	done	6
11	*	9	11	1.0E-4	1
12	for	2	12	true	2
13	if	4	13	fotom	4
14	div	1	14	delta_x	3
15	()	22	15	c	4
16	>	3	16	n	10
17	<=	2	17	nn	11
18	/	2	18	error	3
			19	pieces	8
			20	nt	3

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	;	8	1	true	2
2	/	3	2	false	3
3	!	1	3	sumMain	3
4	_[]	20	4	upper	6
5	_*	9	5	lower	7
6	do while	1	6	0,0001	1
7	+	10	7	nx	6
8	-	8	8	j	13
9		2	9	done	5
10	++	4	10	pieces	8
11	fx	4	11	delta_x	6
12	romb	2	12	c	4
13	=	40	13	sum	4
14	return	2	14	fotom	4
15	for	4	15	x	5
16	if	4	16	nn	12
17	()	20	17	t	12
18	>	3	18	n	13
19	<	3	19	i	12
20	,	4			
21	main	1			

### 3. Сравнение полученных результатов

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

Таблица 7 – Сводная таблица расчетов на языках Паскаль, Си.

Характеристики	Ручной расчёт. Паскаль	Ручной расчёт. Паскаль	Ручной расчёт. Си	Программный расчёт. Си
Число уникальных операторов (n1):	17	17	19	20
Число уникальных операндов (n2):	17	18	17	19
Общее число операторов (N1):	144	161	147	157
Общее число операндов (N2):	104	119	96	111
Алфавит (n):	34	35	36	39
Экспериментальная длина программы (Nэ):	248	280	243	268
Теоретическая длина программы (Nт):	138.973	144.545	150.1974	167.149
Объем программы (V):	1261.690784	1436.199	256.2917	1416.48
Потенциальный объем (V*):	19.651484	19.651	11.60964	19.651
Уровень программы (L):	0.015575	0.0136	0.0092411	0.0138
Интеллектуальное содержание (I):	24.263284	25.557	23.417719	24.2461
Работа по программированию (E):	81004.7525	104962.466	135944.694 <sub>2</sub>	102101.074
Время кодирования (T):	8100.4752	10496.246	13594.4694 <sub>2</sub>	10210.107
Уровень языка программирования (Lambda):	0.30682	0.30682	0.107286	0.27263
Уровень ошибок (B):	2	2	2	2

## **Вывод**

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

Все характеристики были посчитаны вручную и автоматически. Различия между методами присутствует из-за того, что программа считает не только функциональную часть, но и объявления типов, переменных и функций

## ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ

```
program rombl;  
  
const    tol          = 1.0E-4;  
var done      : boolean;  
    sum, upper, lower : real;  
  
function fx(x: real): real;  
  
begin  
    fx:=1.0/x  
end;  
  
procedure romb(  
    lower, upper, tol: real;  
    var ans: real);  
var  
    nx          : array[1..16] of integer;  
    t            : array[1..136] of real;  
    done, error  : boolean;  
    pieces, nt, i, ii, n, nn,  
    l, ntra, k, m, j : integer ;  
    delta_x, c, sum, fotom, x : real ;  
begin  
    done:=false;  
    error:=false;  
    pieces:=1;  
    nx[1]:=1;  
    delta_x:=(upper-lower)/pieces;  
    c:=(fx(lower)+fx(upper))*0.5;  
    t[1]:=delta_x*c;  
    n:=1;  
    nn:=2;  
    sum:=c;  
    repeat  
        n:=n+1;  
        fotom:=4.0;  
        nx[n]:=nn;  
        pieces:=pieces*2;  
        l:=pieces-1;  
        delta_x:=(upper-lower)/pieces;  
        for ii:=1 to (l+1) div 2 do  
            begin  
                i:=ii*2-1;  
                x:=lower+i*delta_x;  
                sum:=sum+fx(x);  
            end;  
        end;  
    until done;  
    ans:=sum;  
end;
```

```

t[nn]:=delta_x*sum;
write(pieces:5,t[nn]);
ntra:=nx[n-1];
k:=n-1;
for m:=1 to k do
  begin
j:=nn+m;
nt:=nx[n-1]+m-1;
t[j]:=(fotom*t[j-1]-t[nt])/(fotom-1.0);
  end;
writeln(j:4,t[j]);
if n>4 then
  begin
if t[nn+1]<>0.0 then
  if (abs(t[ntra+1]-t[nn+1])<=abs(t[nn+1]*tol))
    or (abs(t[nn-1]-t[j])<=abs(t[j]*tol)) then
    done:=true
  else if n>15 then
    begin
      done:=true;
      error:=true
    end
  end;
  nn:=j+1;
until done;
ans:=t[j]
end;

begin
  lower:=1.0;
  upper:=9.0;
  writeln;
  romb(lower,upper,tol,sum);
  writeln;
  writeln(chr(7),'Area= ',sum)
end.

```

## ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

double tol = 0.0001;
bool done = false;
double sumMain, upper, lower;

double fx(double x) {
    return 1.0/x;
}

void romb(double lower, double upper) {
    int p;
    int nx[16];
    for (p = 0; p < 16; p++) {
        nx[p] = 0;
    }
    double t[136];
    for (p = 0; p < 136; p++) {
        t[p] = 0.0;
    }
    bool done, error;
    int pieces, nt, i, ii, n, nn, l, ntra, k, m, j;
    double delta_x, c, sum, fotom, x;
    pieces = 1;
    nx[0] = 1;
    delta_x = (upper-lower)/pieces;
    c = (fx(lower)+fx(upper))*0.5;
    t[0] = delta_x*c;
    n = 1;
    nn = 2;
    sum = c;
    do {
        n = n+1;
        fotom = 4.0;
        nx[n-1] = nn;
        pieces = pieces*2;
        l = pieces-1;
        delta_x = (upper-lower)/pieces;
        for (ii = 1; ii <= ((l+1)/2); ii++) {
            i = ii*2-1;
            x = lower+i *delta_x;
            sum = sum+fx(x);
        }
        t[nn-1] = delta_x *sum;
        printf("%d%f ", pieces, t[nn]);
        ntra = nx[n-2];
        k = n-1;
        for (m = 1; m <= k; m++) {
            j = nn + m;
            nt = nx[n-1-1]+m-1;
            t[j-1] = (fotom *t[j - 2]-t[nt-1])/(fotom-1.0);
        }
        printf("%d%f\n", j-1, t[j-1]);
        if (n>4) {
            if (t[nn] < 0.0 || t[nn] > 0.0) {
                if ((abs(t[ntra + 1]-t[nn-1])<=abs(t[nn-1]*tol)) ||
```



```

        (abs(t[nn - 2]-t[j-1])<=abs(t[j-1]*tol))) {
            done =true;
        } else if (n>15) {
            done = true;
            error = true;
        }
    }
    nn = j+1;
}while(!done);
sumMain = t[j-1];
}

int main() {
    lower = 1.0;
    upper = 9.0;
    printf("\n");
    romb(lower,upper);
    printf("\n");
    printf("Area= %f",sumMain);
    return 0;
}

```

## ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
        tol:
        .long    -350469331
        .long    1058682594
done:
        .zero    1
sumMain:
        .zero    8
upper:
        .zero    8
lower:
        .zero    8
fx(double):
        push     rbp
        mov      rbp, rsp
        movsd    QWORD PTR [rbp-8], xmm0
        movsd    xmm0, QWORD PTR .LC1[rip]
        divsd    xmm0, QWORD PTR [rbp-8]
        movq     rax, xmm0
        movq     xmm0, rax
        pop      rbp
        ret
.LC5:
        .string  "%d%f "
.LC6:
        .string  "%d%f\n"
romb(double, double):
        push     rbp
        mov      rbp, rsp
        sub      rsp, 1280
        movsd    QWORD PTR [rbp-1256], xmm0
        movsd    QWORD PTR [rbp-1264], xmm1
        mov      DWORD PTR [rbp-4], 0
.L7:
        cmp      DWORD PTR [rbp-4], 15
        jg       .L6
        mov      eax, DWORD PTR [rbp-4]
        cdqe
        mov      DWORD PTR [rbp-160+rax*4], 0
        add      DWORD PTR [rbp-4], 1
        jmp      .L7
.L6:
        mov      DWORD PTR [rbp-4], 0
.L9:
        cmp      DWORD PTR [rbp-4], 135
        jg       .L8
        mov      eax, DWORD PTR [rbp-4]
        cdqe
        pxor     xmm0, xmm0
        movsd    QWORD PTR [rbp-1248+rax*8], xmm0
        add      DWORD PTR [rbp-4], 1
        jmp      .L9
.L8:
```

```

mov     DWORD PTR [rbp-12], 1
mov     DWORD PTR [rbp-160], 1
movsd   xmm0, QWORD PTR [rbp-1264]
subsd   xmm0, QWORD PTR [rbp-1256]
pxor     xmm1, xmm1
cvtsi2sd      xmm1, DWORD PTR [rbp-12]
divsd   xmm0, xmm1
movsd   QWORD PTR [rbp-48], xmm0
mov     rax, QWORD PTR [rbp-1256]
movq    xmm0, rax
call    fx(double)
movsd   QWORD PTR [rbp-1272], xmm0
mov     rax, QWORD PTR [rbp-1264]
movq    xmm0, rax
call    fx(double)
movsd   xmm1, QWORD PTR [rbp-1272]
addsd   xmm1, xmm0
movsd   xmm0, QWORD PTR .LC3[rip]
mulsd   xmm0, xmm1
movsd   QWORD PTR [rbp-56], xmm0
movsd   xmm0, QWORD PTR [rbp-48]
mulsd   xmm0, QWORD PTR [rbp-56]
movsd   QWORD PTR [rbp-1248], xmm0
mov     DWORD PTR [rbp-20], 1
mov     DWORD PTR [rbp-24], 2
movsd   xmm0, QWORD PTR [rbp-56]
movsd   QWORD PTR [rbp-40], xmm0
.L23:
add     DWORD PTR [rbp-20], 1
movsd   xmm0, QWORD PTR .LC4[rip]
movsd   QWORD PTR [rbp-64], xmm0
mov     eax, DWORD PTR [rbp-20]
sub     eax, 1
cdqe
mov     edx, DWORD PTR [rbp-24]
mov     DWORD PTR [rbp-160+rax*4], edx
sal     DWORD PTR [rbp-12]
mov     eax, DWORD PTR [rbp-12]
sub     eax, 1
mov     DWORD PTR [rbp-68], eax
movsd   xmm0, QWORD PTR [rbp-1264]
subsd   xmm0, QWORD PTR [rbp-1256]
pxor     xmm1, xmm1
cvtsi2sd      xmm1, DWORD PTR [rbp-12]
divsd   xmm0, xmm1
movsd   QWORD PTR [rbp-48], xmm0
mov     DWORD PTR [rbp-16], 1
.L11:
mov     eax, DWORD PTR [rbp-68]
add     eax, 1
mov     edx, eax
shr     edx, 31
add     eax, edx
sar     eax
cmp     DWORD PTR [rbp-16], eax

```

```

    jg      .L10
    mov     eax, DWORD PTR [rbp-16]
    add     eax, eax
    sub     eax, 1
    mov     DWORD PTR [rbp-72], eax
    pxor    xmm0, xmm0
    cvtsi2sd    xmm0, DWORD PTR [rbp-72]
    mulsd   xmm0, QWORD PTR [rbp-48]
    movsd   xmm1, QWORD PTR [rbp-1256]
    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-40]
    addsd   xmm0, xmm1
    movsd   QWORD PTR [rbp-40], xmm0
    add     DWORD PTR [rbp-16], 1
    jmp     .L11
.L10:
    mov     eax, DWORD PTR [rbp-24]
    sub     eax, 1
    movsd   xmm0, QWORD PTR [rbp-48]
    mulsd   xmm0, QWORD PTR [rbp-40]
    cdqe
    movsd   QWORD PTR [rbp-1248+rax*8], xmm0
    mov     eax, DWORD PTR [rbp-24]
    cdqe
    mov     rdx, QWORD PTR [rbp-1248+rax*8]
    mov     eax, DWORD PTR [rbp-12]
    movq    xmm0, rdx
    mov     esi, eax
    mov     edi, OFFSET FLAT:.LC5
    mov     eax, 1
    call    printf
    mov     eax, DWORD PTR [rbp-20]
    sub     eax, 2
    cdqe
    mov     eax, DWORD PTR [rbp-160+rax*4]
    mov     DWORD PTR [rbp-84], eax
    mov     eax, DWORD PTR [rbp-20]
    sub     eax, 1
    mov     DWORD PTR [rbp-88], eax
    mov     DWORD PTR [rbp-28], 1
.L13:
    mov     eax, DWORD PTR [rbp-28]
    cmp     eax, DWORD PTR [rbp-88]
    jg      .L12
    mov     edx, DWORD PTR [rbp-24]
    mov     eax, DWORD PTR [rbp-28]
    add     eax, edx
    mov     DWORD PTR [rbp-32], eax
    mov     eax, DWORD PTR [rbp-20]
    sub     eax, 2
    cdqe

```

```

mov     edx, DWORD PTR [rbp-160+rax*4]
mov     eax, DWORD PTR [rbp-28]
add     eax, edx
sub     eax, 1
mov     DWORD PTR [rbp-92], eax
mov     eax, DWORD PTR [rbp-32]
sub     eax, 2
cdqe
movsd   xmm0, QWORD PTR [rbp-1248+rax*8]
mulsd   xmm0, QWORD PTR [rbp-64]
mov     eax, DWORD PTR [rbp-92]
sub     eax, 1
cdqe
movsd   xmm1, QWORD PTR [rbp-1248+rax*8]
subsd   xmm0, xmm1
movsd   xmm1, QWORD PTR [rbp-64]
movsd   xmm2, QWORD PTR .LC1[rip]
subsd   xmm1, xmm2
mov     eax, DWORD PTR [rbp-32]
sub     eax, 1
divsd   xmm0, xmm1
cdqe
movsd   QWORD PTR [rbp-1248+rax*8], xmm0
add     DWORD PTR [rbp-28], 1
jmp     .L13
.L12:
mov     eax, DWORD PTR [rbp-32]
sub     eax, 1
cdqe
mov     rax, QWORD PTR [rbp-1248+rax*8]
mov     edx, DWORD PTR [rbp-32]
sub     edx, 1
movq    xmm0, rax
mov     esi, edx
mov     edi, OFFSET FLAT:.LC6
mov     eax, 1
call    printf
cmp     DWORD PTR [rbp-20], 4
jle     .L14
mov     eax, DWORD PTR [rbp-24]
cdqe
movsd   xmm1, QWORD PTR [rbp-1248+rax*8]
pxor    xmm0, xmm0
comisd  xmm0, xmm1
ja      .L15
mov     eax, DWORD PTR [rbp-24]
cdqe
movsd   xmm0, QWORD PTR [rbp-1248+rax*8]
pxor    xmm1, xmm1
comisd  xmm0, xmm1
jbe     .L14
.L15:
mov     eax, DWORD PTR [rbp-84]
add     eax, 1
cdqe

```

```

movsd    xmm0, QWORD PTR [rbp-1248+rax*8]
mov      eax, DWORD PTR [rbp-24]
sub      eax, 1
cdqe
movsd    xmm1, QWORD PTR [rbp-1248+rax*8]
subsd    xmm0, xmm1
movq     rax, xmm0
movq     xmm0, rax
call     std::abs(double)
movsd    QWORD PTR [rbp-1272], xmm0
mov      eax, DWORD PTR [rbp-24]
sub      eax, 1
cdqe
movsd    xmm1, QWORD PTR [rbp-1248+rax*8]
movsd    xmm0, QWORD PTR tol[rip]
mulsd    xmm1, xmm0
movq     rax, xmm1
movq     xmm0, rax
call     std::abs(double)
movq     rax, xmm0
movq     xmm3, rax
comisd   xmm3, QWORD PTR [rbp-1272]
jnb      .L17
mov      eax, DWORD PTR [rbp-24]
sub      eax, 2
cdqe
movsd    xmm0, QWORD PTR [rbp-1248+rax*8]
mov      eax, DWORD PTR [rbp-32]
sub      eax, 1
cdqe
movsd    xmm1, QWORD PTR [rbp-1248+rax*8]
subsd    xmm0, xmm1
movq     rax, xmm0
movq     xmm0, rax
call     std::abs(double)
movsd    QWORD PTR [rbp-1272], xmm0
mov      eax, DWORD PTR [rbp-32]
sub      eax, 1
cdqe
movsd    xmm1, QWORD PTR [rbp-1248+rax*8]
movsd    xmm0, QWORD PTR tol[rip]
mulsd    xmm1, xmm0
movq     rax, xmm1
movq     xmm0, rax
call     std::abs(double)
movq     rax, xmm0
movq     xmm5, rax
comisd   xmm5, QWORD PTR [rbp-1272]
jb       .L24
.L17:
mov      eax, 1
jmp      .L20
.L24:
mov      eax, 0
.L20:

```

```

        test    al, al
        je      .L21
        mov     BYTE PTR [rbp-5], 1
        jmp     .L14
.L21:
        cmp     DWORD PTR [rbp-20], 15
        jle     .L14
        mov     BYTE PTR [rbp-5], 1
        mov     BYTE PTR [rbp-93], 1
.L14:
        mov     eax, DWORD PTR [rbp-32]
        add     eax, 1
        mov     DWORD PTR [rbp-24], eax
        cmp     BYTE PTR [rbp-5], 0
        jne     .L22
        jmp     .L23
.L22:
        mov     eax, DWORD PTR [rbp-32]
        sub     eax, 1
        cdqe
        movsd   xmm0, QWORD PTR [rbp-1248+rax*8]
        movsd   QWORD PTR sumMain[rip], xmm0
        nop
        leave
        ret
.LC8:
        .string "Area= %f"
main:
        push    rbp
        mov     rbp, rsp
        movsd   xmm0, QWORD PTR .LC1[rip]
        movsd   QWORD PTR lower[rip], xmm0
        movsd   xmm0, QWORD PTR .LC7[rip]
        movsd   QWORD PTR upper[rip], xmm0
        mov     edi, 10
        call    putchar
        movsd   xmm0, QWORD PTR upper[rip]
        mov     rax, QWORD PTR lower[rip]
        movapd  xmm1, xmm0
        movq    xmm0, rax
        call    romb(double, double)
        mov     edi, 10
        call    putchar
        mov     rax, QWORD PTR sumMain[rip]
        movq    xmm0, rax
        mov     edi, OFFSET FLAT:.LC8
        mov     eax, 1
        call    printf
        mov     eax, 0
        pop     rbp
        ret
.LC0:
        .long   -1
        .long   2147483647
        .long   0

```

```
.LC1:      .long    0
           .long    0
           .long    1072693248
.LC3:      .long    0
           .long    1071644672
.LC4:      .long    0
           .long    1074790400
.LC7:      .long    0
           .long    1075970048
```



## ПРИЛОЖЕНИЕ Г. РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

Statistics for module ./output.lxm

```
=====
The number of different operators : 18
The number of different operands : 20
The total number of operators : 161
The total number of operands : 119
Dictionary ( D ) : 38
Length ( N ) : 280
Length estimation ( ^N ) : 144.545
Volume ( V ) : 1436.199
Potential volume ( *V ) : 11.6096
Limit volume ( **V ) : 15.6844
Programming level ( L ) : 0.0136
Programming level estimation ( ^L ) : 0.0363985
Intellect ( I ) : 25.557
Time of programming ( T ) : 104962.466
Time estimation ( ^T ) : 52654.56
Programming language level (lambda) : 0.30682
Work on programming ( E ) : 40115.1
Error ( B ) : 1.724
Error estimation ( ^B ) : 1.1343
Table:
=====
```

Operators:

```
| 1 | 25 | ;
| 2 | 1 | const
| 3 | 35 | =
| 4 | 1 | repeat
| 5 | 21 | []
| 6 | 1 | or
| 7 | 8 | +
| 8 | 9 | -
| 9 | 4 | if
| 10 | 2 | for
| 11 | 9 | *
| 12 | 1 | div
| 13 | 22 | ()
| 14 | 3 | >
| 15 | 2 | <=
| 16 | / | 2
| 17 | 1 | romb
| 18 | 1 | fx
```

Operands:

```
| 1 | 1 | 1.E0-4
| 2 | 3 | false
| 3 | 5 | nx
| 4 | 16 | t
| 5 | 5 | x
| 6 | ans | 2
| 7 | 5 | i
| 8 | 6 | done
```

```
| 9 | 2 | true
| 10 | 3 | hold
| 11 | 4 | fotom
| 12 | 3 | delta_x
| 13 | c | c
| 14 | 4 | n
| 15 | 11 | nn
| 16 | 3 | error
| 17 | 8 | pieces
| 18 | 8 | sum
| 19 | 8 | lower
| 20 | 7 | upper
```

Summary:

```
=====
The number of different operators : 18
The number of different operands : 20
The total number of operators : 161
The total number of operands : 119
Dictionary ( D ) : 38
Length ( N ) : 280
Length estimation ( ^N ) : 144.545
Volume ( V ) : 1436.199
Potential volume ( *V ) : 11.6096
Limit volume ( **V ) : 15.6844
Programming level ( L ) : 0.0136
Programming level estimation ( ^L ) : 0.0363985
Intellect ( I ) : 25.557
Time of programming ( T ) : 104962.466
Time estimation ( ^T ) : 52654.56
Programming language level (lambda) : 0.30682
Work on programming ( E ) : 40115.1
Error ( B ) : 1.724
Error estimation ( ^B ) : 1.134
```

## ПРИЛОЖЕНИЕ Д. РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ СИ.

Statistics for module ./output.lxm

```
=====
The number of different operators : 21
The number of different operands : 19
The total number of operators : 157
The total number of operands : 111
Dictionary ( D ) : 40
Length ( N ) : 268
Length estimation ( ^N ) : 167.149
```

Volume ( V ) : 1416.48  
 Potential volume ( \*V ) : 19.651  
 Limit volume ( \*\*V ) : 21.5427  
 Programming level ( L ) : 0.0138  
 Programming level estimation ( ^L ) : 0.0208605  
 Intellect ( I ) : 24.2461  
 Time of programming ( T ) : 102101.074  
 Time estimation ( ^T ) : 54 674.14  
 Programming language level (lambda) : 0.27263  
 Work on programming ( E ) : 10210.107  
 Error ( B ) : 0.507451  
 Error estimation ( ^B ) : 1.943

Table:

=====

Operators:

1	1	!
2	8	;
3	3	/
4	20	()
5	10	+
6	4	++
7	4	,
8	3	<
9	40	=
10	3	>
11	9	_*
12	20	_[]
13	4	___*
14	2	
15	1	dowhile
16	4	for
17	4	if
18	1	main
19	2	fx
20	1	return
21	2	romb

Operands:

1	1	0.0001
2	2	true
3	3	false
4	3	sumMain
5	6	upper
6	7	lower
7	4	done
8	6	nx
9	13	n
10	8	pieces
11	6	delta_x
12	4	c
13	4	sum
14	4	fotom
15	5	x
16	12	nn

| 17 | 12 | t  
| 18 | 13 | j  
| 19 | 12 | i

#### Summary:

=====

The number of different operators : 20  
The number of different operands : 19  
The total number of operators : 157  
The total number of operands : 111  
Dictionary ( D ) : 39  
Length ( N ) : 268  
Length estimation (  $\hat{N}$  ) : 167.149  
Volume ( V ) : 1416.48  
Potential volume (  $*V$  ) : 19.651  
Limit volume (  $**V$  ) : 21.5427  
Programming level ( L ) : 0.0138  
Programming level estimation (  $\hat{L}$  ) : 0.0208605  
Intellect ( I ) : 24.2461  
Time of programming ( T ) : 102101.074  
Time estimation (  $\hat{T}$  ) : 54 674.14  
Programming language level (lambda) : 0.27263  
Work on programming ( E ) : 10210.107  
Error ( B ) : 0.507451  
Error estimation (  $\hat{B}$  ) : 1.943