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. 6304 _____ Григорьев И.С.

Преподаватель _____ Кирьянчиков В.А.

Санкт-Петербург

2020

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. Добиться, чтобы программы на Паскале и Си были работоспособны и давали корректные результаты (это потребуется в дальнейшем при проведении с ними измерительных экспериментов).

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.

Ход работы

1. Определение метрических характеристик для программы на Pascal.

Код программы представлен в приложении А.

Ручной расчёт измеримых характеристик представлен в таблице 1.

Таблица 1 – Ручной расчёт измеримых характеристик (Pascal)

№	Оператор	Количество
1	:=	15
2	() или begin end	15
3	;	11
4	*	8
5	+	4
6	-	4
7	/	4
8	fx	3
11	abs	2
10	div	1
11	for to do	1
12	<=	1
13	repeat until	1
14	trapez	1
Всего		71

№	Операнд	Количество
1	pieces	6
2	lower	5
3	sum	6
4	delta_x	5
5	upper	4
6	mid_sum	4
7	end_sum	3
8	i	2
9	sum1	2
10	tol	2
11	fx	1
12	x	3
13	1.0	2
14	2.0	3
15	1	2
16	2	2
17	0.0	1
18	0.5	1
Всего		54

Программный расчёт измеримых характеристик представлен в таблице 2. Файл с результатами программных расчётов представлен в приложении Б.

Таблица 2 – Программный расчёт измеримых характеристик (Pascal)

№	Оператор	Количество
1	=	15
2	()	13
3	;	36
4	*	8
5	+	4
6	-	4
7	/	5
8	fx	4
9	const	1
10	abs	2
11	function	1
12	for	1
13	<=	1
14	repeat	1
15	trapez	2
16	real	6
17	program	1
18	procedure	1
19	integer	1
Всего		107

№	Операнд	Количество
1	pieces	7
2	lower	8
3	sum	8
4	delta_x	6
5	upper	7
6	mid_sum	5
7	end_sum	4
8	i	2
9	sum1	3
10	tol	4
11	fx	1
12	x	5
13	1.0	3
14	2.0	3
15	1	2
16	2	2
17	0.0	1
18	9.0	1
19	0.5	1
20	1.0E-6	1
21	trap	1
Всего		75

Расчетные характеристики представлены в таблице 3.

Таблица 3 – Расчётные характеристики (Pascal)

Характеристика	Ручной расчёт	Программный расчёт
Число простых операторов n_1	14	19
Число простых операндов n_2	18	21
Общее число всех операторов N_1	71	107
Общее число всех операндов N_2	54	75
Словарь n	32	40
Длина $N_{опыт}$	125	182
Теоретическая длина $N_{теор}$	128.3616	172.949
Объём V	625	968.591
Потенциальный объём V*	11.6096	11.6096
Уровень программы L	0.0185754	0.0119861
Оценка уровня программы L^	0.047619	0.0294737
Интеллектуальное содержание I	29.7619	28.5479
Работа программирования E	33646.6	80809.4
Оценка времени программирования T^	3364	1734.93
Время программирования T	1312.5	4489.41
Уровень языка λ	0.215654	0.139154
Ожидаемое число ошибок в программе B	1.5625	0.623046

2. Определение метрических характеристик для программы на Си.

Код программы представлен в приложении В.

Ручной расчёт измеримых характеристик представлен в таблице 4.

Таблица 4 – Ручной расчёт измеримых характеристик (Си)

№	Оператор	Количество
1	=	13
2	() или {}	20
3	;	16
4	*	8
5	+	4
6	-	4
7	/	5
8	fx	3
9	fabs	2
10	<=	1
11	for	1
12	>	1
13	do while	1
14	trapez	1
15	++	1
16	return	1
Всего		82

№	Операнд	Количество
1	pieces	5
2	lower	5
3	sum	5
4	delta_x	5
5	upper	4
6	mid_sum	4
7	end_sum	3
8	i	4
9	sum1	2
10	tol	2
11	x	3
12	1.0	2
13	2.0	3
14	1	1
15	2	2
16	0.0	1
17	0.5	1
Всего		52

Программный расчёт измеримых характеристик представлен в таблице 5. Файл с результатами программных расчётов представлен в приложении Г.

Таблица 5 – Программный расчёт измеримых характеристик (Си)

№	Оператор	Количество
1	=	15
2	()	9
3	;	23
4	*	8
5	+	4
6	-	4
7	/	5
8	fx	4
9	,	1
10	fabs	2
11	<=	1
12	for	1
13	>	1
14	do while	1
15	trapez	2
16	++	1
17	return	1
18	void	1
19	int	3
20	main	1
21	double	8
22	const	1
Всего		106

№	Операнд	Количество
1	pieces	6
2	lower	8
3	sum	6
4	delta_x	6
5	upper	7
6	mid_sum	5
7	end_sum	4
8	i	4
9	sum1	3
10	tol	4
11	x	5
12	1.0	3
13	2.0	3
14	1	2
15	2	2
16	0.0	1
17	0.5	1
18	9.0	1
19	1.0E-6	1
Всего		72

Определение расчетных характеристик представлено в таблице 6.

Таблица 6 – Расчетные характеристики (Си)

Характеристика	Ручной расчёт	Программный расчёт
Число простых операторов n_1	16	22
Число простых операндов n_2	17	19
Общее число всех операторов N_1	82	106
Общее число всех операндов N_2	52	72
Словарь n	33	41
Длина $N_{опыт}$	134	178
Теоретическая длина $N_{теор}$	133.486	178.818
Объём V	675.948	953.644
Потенциальный объём V*	11.6096	11.6096
Уровень программы L	0.0171753	0.012174
Оценка уровня программы L˜	0.040865	0.0239899
Интеллектуальное содержание I	27.6229	22.8778
Работа программирования E	39355.8	78334.7
Оценка времени программирования T^	3935.58	2218.59
Время программирования T	1654.08	4351.93
Уровень языка λ	0.1993	0.141335
Ожидаемое число ошибок в программе B	1.689872	0.61026

3. Определение метрических характеристик для программы на Ассемблере.

Код программы представлен в приложении Д.

Ручной расчёт измеримых характеристик представлен в таблице 7.

Таблица 7 – Ручной расчёт измеримых характеристик (Ассемблер)

№	Оператор	Количество
1	pushq	3
2	popq	1
3	movq	13
4	movl	5
5	movsd	37
6	movapd	3
7	addsd	6
8	addl	2
9	subsd	4
10	subq	2
11	andpd	2
12	divsd	4
13	ret	3
14	cvtsi2sd	3
15	call fx	3
16	call trapez	1
17	mulsd	5
18	pxor	1
19	sall	1
20	jmp .L4	1
21	shrl	1
22	sarl	1
23	cmpl	1
24	jle .L5	1
25	ja .L6	1
26	ucomisd	1
27	nop	1

№	Операнд	Количество
1	%rbp	55
2	%rsp	5
3	%xmm0	56
4	%xmm1	24
5	%xmm2	4
6	%rip	18
7	$88	1
8	$1	3
9	%rax	8
10	%eax	6
11	%edx	3
12	$31	1
13	$8	1
14	$0	1

28	leave	2
29	-8	6
30	-56	5
31	-64	4
32	-72	2
33	-48	5
34	-32	5
35	-80	6
36	-88	2
37	-24	2
38	sum	5
39	-40	4
40	-16	2
41	-44	4
42	lower	2
43	upper	2
Всего		165

Всего		186

Определение расчетных характеристик представлено в таблице 8.

Таблица 8 – Расчёт расчетных характеристик (Ассемблер)

Характеристика	Ручной расчёт
Число простых операторов n_1	43
Число простых операндов n_2	14
Общее число всех операторов N_1	165
Общее число всех операндов N_2	186
Словарь n	57
Длина $N_{опыт}$	351
Теоретическая длина $N_{теор}$	286.632
Объём V	2047.344
Потенциальный объём V*	11.6096
Уровень программы L	0.00567
Оценка уровня программы L˜	0.0035
Интеллектуальное содержание I	7.167497
Работа программирования E	361046.41
Оценка времени программирования T^	36104
Время программирования T	58480.93
Уровень языка λ	0.065833
Ожидаемое число ошибок в программе B	5.11836

4. Сравнение результатов определения метрических характеристик.

Таблица 9 – Сводная таблица расчетов на трех языках

Характеристика	Ручной расчёт Pascal	Програм-мный расчёт Pascal	Ручной расчёт Си	Програм-мный расчёт Си	Ручной расчёт Ассемблер
Число простых операторов n_1	14	19	16	22	43
Число простых операндов n_2	18	21	17	19	14
Общее число всех операторов N_1	71	107	82	106	165
Общее число всех операндов N_2	54	75	52	72	186
Словарь n	32	40	33	41	57
Длина $N_{опыт}$	125	182	134	178	351
Теоретическая длина $N_{теор}$	128.3616	172.949	133.486	178.818	286.632
Объём V	625	968.591	675.948	953.644	2047.344
Потенциальный объём V*	11.6096	11.6096	11.6096	11.6096	11.6096
Уровень программы	0.0185754	0.0119861	0.0171753	0.012174	0.00567
Оценка уровня программы $L^{\sim}$	0.047619	0.0294737	0.040865	0.0239899	0.0035
Интеллектуальное содержание I	29.7619	28.5479	27.6229	22.8778	7.167497
Работа программирования E	33646.6	80809.4	39355.8	78334.7	361046.41
Оценка времени программирования T^	3364	1734.93	3935.58	2218.59	36104
Время программирования T	1312.5	4489.41	1654.08	4351.93	58480.93
Уровень языка λ	0.215654	0.139154	0.1993	0.141335	0.065833
Ожидаемое число ошибок в программе B	1.5625	0.623046	1.689872	0.61026	5.11836

Опытная длина и объем программ на Pascal и Си практически одинаковые и меньше длины и объема программы на ассемблере более чем в 2 раза. Разница между теоретической и опытной длиной программы не

существенна, за исключением Ассемблера. Ассемблер является низкоуровневым языком программирования, что видно по метрике уровня языка. Pascal и Си находятся практически на одном уровне. Ожидаемое количество ошибок больше всего у Ассемблера и поровну у Pascal и СИ. Время программирования (и другие метрики), рассчитанное вручную, отличается от программного расчета: это связано с тем, что в программном расчете учитывались операторы и операнды, задействованные в части описания или отладки программы.

Выводы

В ходе выполнения лабораторной работы изучена система метрик Холстеда. Произведено сравнение программ, реализующих численное интегрирование методом трапеций, на языках Pascal, Си и Ассемблер.

ПРИЛОЖЕНИЕ А

Код программы на Pascal.

```
program trap;

const tol          = 1.0E-6;
var   sum,upper,lower    : real;

function fx(x: real): real;
begin
  fx:=1.0/x
end;

procedure trapez(lower,upper,tol: real;
             var sum            : real);
var   pieces,i                  : integer;
      x,delta_x,end_sum,mid_sum,sum1  : real;
begin
  pieces:=1;
  delta_x:=(upper-lower)/pieces;
  end_sum:=fx(lower)+fx(upper);
  sum:=end_sum*delta_x/2.0;
  mid_sum:=0.0;
  repeat
    pieces:=pieces*2;
    sum1:=sum;
    delta_x:=(upper-lower)/pieces;
    for i:=1 to pieces div 2 do
    begin
      x:=lower+delta_x*(2.0*i-1.0);
      mid_sum:=mid_sum+fx(x)
    end;
  sum:=(end_sum+2.0*mid_sum)*delta_x*0.5;
  until abs(sum-sum1)<=abs(tol*sum)
end;

begin
  lower:=1.0;
  upper:=9.0;
  trapez(lower,upper,tol,sum);
end.
```

ПРИЛОЖЕНИЕ Б

Результаты parser_pas.exe

```
Statistics for module lab1pas.lxm
====================================
Table:
====================================
Operators:
|  1  |  13 | ()
|  2  |  8  | *
|  3  |  4  | +
|  4  |  4  | -
|  5  |  5  | /
|  6  |  36 | ;
|  7  |  1  | <=
|  8  |  15 | =
|  9  |  2  | abs
|  10 |  1  | const
|  11 |  1  | for
|  12 |  1  | function
|  13 |  4  | fx
|  14 |  1  | integer
|  15 |  1  | procedure
|  16 |  1  | program
|  17 |  6  | real
|  18 |  1  | repeat
|  19 |  2  | trapez
Operands:
|  1  |  1  | 0.0
|  2  |  1  | 0.5
|  3  |  2  | 1
|  4  |  3  | 1.0
|  5  |  1  | 1.0E-6
|  6  |  2  | 2
|  7  |  3  | 2.0
|  8  |  1  | 9.0
|  9  |  6  | delta_x
|  10 |  4  | end_sum
|  11 |  1  | fx
|  12 |  2  | i
|  13 |  8  | lower
|  14 |  5  | mid_sum
|  15 |  7  | pieces
|  16 |  8  | sum
|  17 |  3  | sum1
|  18 |  4  | tol
|  19 |  1  | trap
|  20 |  7  | upper
|  21 |  5  | x


Summary:
====================================
The number of different operators     : 19
The number of different operands      : 21
The total number of operators         : 107
The total number of operands          : 75

Dictionary                  (  D)     : 40
Length                      (  N)     : 182
Length estimation           ( ^N)     : 172.949
```

```
Volume                        (  V)    : 968.591
Potential volume              ( *V)    : 11.6096
Limit volume                  (**V)    : 15.6844
Programming level             (  L)    : 0.0119861
Programming level estimation  ( ^L)    : 0.0294737
Intellect                     (  I)    : 28.5479
Time of programming           (  T)    : 4489.41
Time estimation               ( ^T)    : 1734.93
Programming language level    (lambda) : 0.139154
Work on programming           (  E)    : 80809.4
Error                         (  B)    : 0.623046
Error estimation              ( ^B)    : 0.322864
```

ПРИЛОЖЕНИЕ В

Код программы на Си

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

const double tol = 1.0E-6;
double sum, upper, lower;

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

void trapez(double lower, double upper, double tol) {
	int pieces = 1;
	double x, delta_x, end_sum, mid_sum, sum1;
	delta_x = (upper - lower) / pieces;
	end_sum = fx(lower) + fx(upper);
	sum = end_sum * delta_x / 2.0;
	// printf("    1 %.20f\n", sum);
	mid_sum = 0.0;
	do {
		pieces = pieces * 2;
		sum1 = sum;
		delta_x = (upper - lower) / pieces;
		for (int i = 1; i <= pieces / 2; i++)
		{
			x = lower + delta_x * (2.0 * i - 1.0);
			mid_sum = mid_sum + fx(x);
		}
		sum = (end_sum + 2.0 * mid_sum) * delta_x * 0.5;
		// printf("    %i %.20f\n", pieces, sum);
	} while (fabs(sum - sum1) > fabs(tol * sum));
}

int main() {
	lower = 1.0;
	upper = 9.0;
	trapez(lower, upper, tol);
	//printf("area = %.20f", sum);
}
```

ПРИЛОЖЕНИЕ Г

Результаты parser_c.exe

```
Statistics for module lab1c.lxm
===================================
Table:
===================================
Operators:
|  1  |  9  | ()
|  2  |  8  | *
|  3  |  4  | +
|  4  |  1  | ++
|  5  |  10 | ,
|  6  |  4  | -
|  7  |  5  | /
|  8  |  23 | ;
|  9  |  1  | <=
|  10 |  15 | =
|  11 |  1  | >
|  12 |  1  | const
|  13 |  8  | double
|  14 |  1  | dowhile
|  15 |  2  | fabs
|  16 |  1  | for
|  17 |  4  | fx
|  18 |  3  | int
|  19 |  1  | main
|  20 |  1  | return
|  21 |  2  | trapez
|  22 |  1  | void
Operands:
|  1  |  1  | 0.0
|  2  |  1  | 0.5
|  3  |  2  | 1
|  4  |  3  | 1.0
|  5  |  1  | 1.0E-6
|  6  |  2  | 2
|  7  |  3  | 2.0
|  8  |  1  | 9.0
|  9  |  6  | delta_x
|  10 |  4  | end_sum
|  11 |  4  | i
|  12 |  8  | lower
|  13 |  5  | mid_sum
|  14 |  6  | pieces
|  15 |  6  | sum
|  16 |  3  | sum1
|  17 |  4  | tol
|  18 |  7  | upper
|  19 |  5  | x


Summary:
===================================
The number of different operators     : 22
The number of different operands      : 19
The total number of operators         : 106
The total number of operands          : 72

Dictionary                  (  D)     : 41
Length                      (  N)     : 178
```

```
Length estimation            ( ^N)    : 178.818
Volume                       (  V)    : 953.644
Potential volume             ( *V)    : 11.6096
Limit volume                 (**V)    : 15.6844
Programming level            (  L)    : 0.012174
Programming level estimation ( ^L)    : 0.0239899
Intellect                    (  I)    : 22.8778
Time of programming          (  T)    : 4351.93
Time estimation              ( ^T)    : 2218.59
Programming language level   (lambda) : 0.141335
Work on programming          (  E)    : 78334.7
Error                        (  B)    : 0.61026
Error estimation             ( ^B)    : 0.317881
```

ПРИЛОЖЕНИЕ Д

Код программы на Ассемблер

```
tol:
	.long	2696277389
	.long	1051772663
	.comm	sum,8,8
	.comm	upper,8,8
	.comm	lower,8,8
	.text
	.globl	fx
	.type	fx, @function
fx:
.LFB0:
	.cfi_startproc
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	movsd	%xmm0, -8(%rbp)
	movsd	.LC0(%rip), %xmm0
	divsd	-8(%rbp), %xmm0
	popq	%rbp
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
	.size	fx, .-fx
	.globl	trapez
	.type	trapez, @function
trapez:
.LFB1:
	.cfi_startproc
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$88, %rsp
	movsd	%xmm0, -56(%rbp)
	movsd	%xmm1, -64(%rbp)
	movsd	%xmm2, -72(%rbp)
	movl	$1, -48(%rbp)
	movsd	-64(%rbp), %xmm0
	subsd	-56(%rbp), %xmm0
	cvtsi2sd	-48(%rbp), %xmm1
	divsd	%xmm1, %xmm0
	movsd	%xmm0, -32(%rbp)
	movq	-56(%rbp), %rax
	movq	%rax, -80(%rbp)
	movsd	-80(%rbp), %xmm0
	call	fx
	movsd	%xmm0, -80(%rbp)
	movq	-64(%rbp), %rax
	movq	%rax, -88(%rbp)
	movsd	-88(%rbp), %xmm0
	call	fx
	addsd	-80(%rbp), %xmm0
	movsd	%xmm0, -24(%rbp)
	movsd	-24(%rbp), %xmm0
```

```
        mulsd  -32(%rbp), %xmm0
        movsd  .LC1(%rip), %xmm1
        divsd  %xmm1, %xmm0
        movsd  %xmm0, sum(%rip)
        pxor   %xmm0, %xmm0
        movsd  %xmm0, -40(%rbp)
.L6:
        sall   -48(%rbp)
        movsd  sum(%rip), %xmm0
        movsd  %xmm0, -16(%rbp)
        movsd  -64(%rbp), %xmm0
        subsd  -56(%rbp), %xmm0
        cvtsi2sd     -48(%rbp), %xmm1
        divsd  %xmm1, %xmm0
        movsd  %xmm0, -32(%rbp)
        movl   $1, -44(%rbp)
        jmp    .L4
.L5:
        cvtsi2sd     -44(%rbp), %xmm0
        addsd  %xmm0, %xmm0
        movsd  .LC0(%rip), %xmm1
        subsd  %xmm1, %xmm0
        mulsd  -32(%rbp), %xmm0
        movsd  -56(%rbp), %xmm1
        addsd  %xmm1, %xmm0
        movsd  %xmm0, -8(%rbp)
        movq   -8(%rbp), %rax
        movq   %rax, -80(%rbp)
        movsd  -80(%rbp), %xmm0
        call   fx
        movapd%xmm0, %xmm1
        movsd  -40(%rbp), %xmm0
        addsd  %xmm1, %xmm0
        movsd  %xmm0, -40(%rbp)
        addl   $1, -44(%rbp)
.L4:
        movl   -48(%rbp), %eax
        movl   %eax, %edx
        shrl   $31, %edx
        addl   %edx, %eax
        sarl   %eax
        cmpl   %eax, -44(%rbp)
        jle    .L5
        movsd  -40(%rbp), %xmm0
        addsd  %xmm0, %xmm0
        addsd  -24(%rbp), %xmm0
        mulsd  -32(%rbp), %xmm0
        movsd  .LC3(%rip), %xmm1
        mulsd  %xmm1, %xmm0
        movsd  %xmm0, sum(%rip)
        movsd  sum(%rip), %xmm0
        subsd  -16(%rbp), %xmm0
        movq   .LC4(%rip), %xmm1
        andpd  %xmm1, %xmm0
        movsd  sum(%rip), %xmm1
        mulsd  -72(%rbp), %xmm1
        movq   .LC4(%rip), %xmm2
        andpd  %xmm2, %xmm1
        ucomisd      %xmm1, %xmm0
        ja     .L6
        nop
        leave
```

```
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE1:
	.size	trapez, .-trapez
	.globl	main
	.type	main, @function
main:
.LFB2:
	.cfi_startproc
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$8, %rsp
	movsd	.LC0(%rip), %xmm0
	movsd	%xmm0, lower(%rip)
	movsd	.LC5(%rip), %xmm0
	movsd	%xmm0, upper(%rip)
	movsd	.LC6(%rip), %xmm1
	movsd	upper(%rip), %xmm0
	movq	lower(%rip), %rax
	movapd	%xmm1, %xmm2
	movapd	%xmm0, %xmm1
	movq	%rax, -8(%rbp)
	movsd	-8(%rbp), %xmm0
	call	trapez
	movl	$0, %eax
	leave
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
```