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 _____ Давыдов А.А.

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

Число различных входных и выходных параметров $\eta_2^* = 4$ для всех программ. Число страудовских «моментов» в секунду **S** = 10.

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	;	19
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

№	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

Всего		79

16	2	2
17	0.0	1
18	0.5	1
19	9.0	1
20	1.0E-6	1
Всего		56

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	for	1
12	<=	1
13	repeat	1
14	trapez	2
15	real	1
16	program	1

№	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	99

Всего	75

Расчетные характеристики представлены в таблице 3.

Таблица 3 – Расчётные характеристики (**Pascal**)

Характеристика	Ручной расчёт	Программный расчёт
Число простых операторов n_1	14	16
Число простых операндов n_2	20	21
Общее число всех операторов N_1	79	99
Общее число всех операндов N_2	56	75
Словарь n	34	37
Длина $N_{опыт}$	125	174
Теоретическая длина $N_{теор}$	139.738	156.239
Объём V	635.875	906.445
Потенциальный объём V*	15.51	15.5098
Уровень программы L	0.0243915	0.0171106
Оценка уровня программы L^	0.0510204	0.035
Интеллектуальное содержание I	32.4426	31.7256
Работа программирования E	26069.532	52975.8
Оценка времени программирования T^	2606.9532	1291.93
Время программирования T	1448.307	2943.1
Уровень языка λ	0.378312165	0.265381
Ожидаемое число ошибок в программе B	1	1

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	=	16
2	() или {}	19
3	;	23
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	3
Всего		93

№	Операнд	Количество
1	pieces	5
2	lower	5
3	sum	7
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
18	1.0E-6	1
19	9.0	1
Всего		56

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	,	10
10	fabs	2
11	<=	1
12	for	1
13	>	1
14	do while	1
15	trapez	2
16	++	1
17	return	1
18	printf	3
19	main	1
Всего		96

№	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	19
Число простых операндов n_2	19	19
Общее число всех операторов N_1	93	96
Общее число всех операндов N_2	56	72
Словарь n	35	38
Длина $N_{опыт}$	149	168
Теоретическая длина $N_{теор}$	144.712	155.769
Объём V	764.221	859.56
Потенциальный объём V*	15.51	15.5098
Уровень программы L	0.020295	0.0180439
Оценка уровня программы $L^{\sim}$	0.042411	0.029321
Интеллектуальное содержание I	32,41138	25.2031
Работа программирования E	37655,63	47637.3
Оценка времени программирования T^	3765.563	1537.53
Время программирования T	2091.9794	2646.51
Уровень языка λ	0.314775	0.279856
Ожидаемое число ошибок в программе B	1	1

2. 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

№	Операнд	Количество
1	%rbp	7
2	%rsp	5
3	%xmm0	56
4	%xmm1	24
5	%xmm2	4
6	$88	1
7	$1	3
8	%rax	8
9	%eax	6
10	%edx	3
11	$31	1
12	$8	1
13	$0	1
14	-8(%rbp)	6
15	-56(%rbp)	5
16	-64(%rbp)	4
17	-72(%rbp)	2
18	-48(%rbp)	5
19	-32(%rbp)	5
20	-80(%rbp)	6
21	-88(%rbp)	2
22	-24(%rbp)	3
23	-40(%rbp)	4
24	-16(%rbp)	2
25	-44(%rbp)	4

26	ucomisd	1
27	nop	1
28	leave	2
Всего		109

26	.LC0(%rip)	3
27	.LC1(%rip)	1
28	sum(%rip)	5
29	.LC3(%rip)	1
30	.LC4(%rip)	2
31	lower(%rip)	2
32	.LC5(%rip)	1
33	upper(%rip)	2
34	.LC6(%rip)	1
Всего		186

Определение расчетных характеристик представлено в таблице 8.

Таблица 8 – Расчёт расчетных характеристик (**Ассемблер**)

Характеристика	Ручной расчёт
Число простых операторов n_1	28
Число простых операндов n_2	34
Общее число всех операторов N_1	109
Общее число всех операндов N_2	186
Словарь n	62
Длина $N_{опыт}$	295
Теоретическая длина $N_{теор}$	307.579
Объём V	1756.4879
Потенциальный объём V*	15.5098
Уровень программы L	0.008829
Оценка уровня программы $L^{\sim}$	0.01305
Интеллектуальное содержание I	22.93417
Работа программирования E	198922.92
Оценка времени программирования T^	19892.29
Время программирования T	13452.63
Уровень языка λ	0.136951
Ожидаемое число ошибок в программе B	2

3. Сравнение результатов определения метрических характеристик.

Таблица 9 – Сводная таблица расчетов на трех языках

Характеристика	Ручной расчёт Pascal	Программный расчёт Pascal	Ручной расчёт Си	Программный расчёт Си	Ручной расчёт Ассемблер
Число простых операторов n_1	14	16	16	18	28
Число простых операндов n_2	18	21	17	19	34
Общее число всех операторов N_1	71	99	82	93	109
Общее число всех операндов N_2	54	75	52	72	186
Словарь n	32	37	33	37	62
Длина $N_{опыт}$	125	174	134	165	295
Теоретическая длина $N_{теор}$	128.3616	156.239	133.486	155.769	307.579
Объём V	625	906.445	675.948	859.56	1756.4879
Потенциальный объём V*	15.5098	15.5098	15.5098	15.5098	15.5098
Уровень программы	0.0248156	0.0171106	0.022945	0.0180439	0.008829
Оценка уровня программы $L^{\sim}$	0.047619	0.035	0.040865	0.029321	0.01305
Интеллектуальное содержание I	29.7619	31.7256	27.6229	25.2031	22.93417
Работа программирования E	25185.7	52975.8	29459.3	47637.3	198922.92
Оценка времени программирования T^	2518.57	1291.93	2945.93	1537.53	19892.29
Время программирования T	1312.5	2943.1	1654.08	2646.51	13452.63
Уровень языка λ	0.3848849	0.265381	0.3559	0.279856	0.136951
Ожидаемое число ошибок в программе B	1	1	1	1	2

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. 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 |  4 | fx
|  13 |  1 | program
|  14 |  1 | real
|  15 |  1 | repeat
|  16 |  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     : 16
The number of different operands      : 21
The total number of operators         : 99
The total number of operands          : 75

Dictionary                  (  D)     : 37
Length                      (  N)     : 174
Length estimation           ( ^N)     : 156.239
```

Volume (V) : 906.445
Potential volume (*V) : 15.5098
Limit volume (**V) : 25.8496 Programming level
(L) : 0.0171106
Programming level estimation (^L) : 0.035
Intellect (I) : 31.7256
Time of programming (T) : 2943.1
Time estimation (^T) : 1291.93
Programming language level (lambda) : 0.265381
Work on programming (E) :
52975.8 Error
(B) : 0.470179 Error estimation
(^B) : 0.302148

ПРИЛОЖЕНИЕ В

Код программы на Си

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

const double tol = 1.0E-6;

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

double trapez(double lower, double upper, double tol, double sum)
{
   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;
   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;
} while (fabs(sum - sum1) > fabs(tol * sum));
    return sum;
}
int  main()
{
            double sum = 0.0;
            double res = 0.0;
            double upper = 9.0;
            double lower = 1.0;
      res = trapez(lower, upper, tol, sum);
            return 0;
}
```

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 | dowhile
|  13 |  2 | fabs
|  14 |  1 | for
|  15 |  4 | fx
|  16 |  1 | main
|  17 |  3 | return
|  18 |  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 |  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     : 18
The number of different operands      : 19
The total number of operators         : 93
The total number of operands          : 72

Dictionary                   (  D)    : 37
Length                       (  N)    : 165
Length estimation            ( ^N)    : 155.769
Volume                       (  V)    : 859.56
Potential volume             ( *V)    : 15.5098
Limit volume                 (**V)    : 25.8496 Programming level
(  L)    : 0.0180439
Programming level estimation ( ^L)    : 0.029321
Intellect                    (  I)    : 25.2031
Time of programming          (  T)    : 2646.51
```

```
Time estimation              ( ^T)    : 1537.53
Programming language level   (lambda) : 0.279856
Work on programming          (  E)    : 47637.3
Error                        (  B)    : 0.438036
Error estimation             ( ^B)    : 0.28652
```

ПРИЛОЖЕНИЕ Д

Код программы на Ассемблер

```
tol:
     .text
     .globl fx
      .type fx, @function
fx:
      .LFB0:
      .cfi_startproc
            pushq %rbp
            movq %rsp, %rbp
            movsd %xmm0, -8(%rbp)
            movsd .LC0(%rip), %xmm0
      divsd -8(%rbp), %xmm0
            popq %rbp
     ret
     .cfi_endproc
.LFE0:


.globl trapez
            .type trapez, @function
trapez:
.LFB1:
      .cfi_startproc
            pushq %rbp

            movq %rsp, %rbp
            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
     ret
     .cfi_endproc
.LFE1:

.globl main
            .type main, @function

main:
```

```
.LFB2:
            .cfi_startproc
            pushq %rbp
            movq %rsp, %rbp
            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
     ret
     .cfi_endproc
```