

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

Виноградов К.А.

Преподаватель

Кирияничиков В.А.

Санкт-Петербург

2020

Цель работы.

Изучение способа расчета метрических характеристик качества разработки программ на основе метрик Холстеда.

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.

Для начала определим метрические характеристики у исходного файла на Pascal. Результаты ручного измерения характеристик представлены в табл. 1.

Таблица 1 – Ручной расчёт характеристик программы на Pascal

№	Оператор	Количество	№	Операнд	Количество
1	;	9	1	x	11
2	:=	8	2	x1	2
3	()	6	3	fx	3
4	() или begin ... end	8	4	dfx	7
5	repeat ... until ...	1	5	tol	4
6	+	3	6	0.0	1
7	-	4	7	dx	3
8	*	3	8	a	1
9	/		9	b	2
10	<	1	10	c	2
11	<=	1	11	logp	1
12	>=	1	12	5.0	1
13	newton	1			
14	func	1			
15	abs	3			
16	ln	1			
Всего		51	Всего		38

Далее произведем автоматический расчет метрик. Его результаты представлены в табл. 2.

Таблица 2 – Автоматический расчёт характеристик программы на Pascal

№	Оператор	Количество	№	Операнд	Количество
1	;	29	1	x	14
2	()	12	2	x1	3
3	/	4	3	fx	5
4	+	3	4	dfx	9
5	-	7	5	tol	5
6	*	3	6	0.0	1
7	=	13	7	dx	4
8	<	1	8	a	2
9	<=	1	9	b	3
10	>=	1	10	c	3
11	abs	3	11	logp	2
12	repeat	1	12	newdir	1
13	const	1	13	0.8858	1
14	real	4	14	1.0E-6	1
15	newton	2	15	5.0	1
16	func	2	16	18.19	1
17	if	2	17	23180.0	1
18	ln	1	18	4.60517	1
19	procedure	2			
20	program	1			
Всего		93	Всего		58

Далее произведем аналогичный подсчет для программы на языке С. Результаты ручного подсчет представлены в табл. 3, а программного – в табл.4.

Таблица 3 – Ручной расчёт характеристик программы на С

№	Оператор	Количество	№	Операнд	Количество
1	;	13	1	x	11
2	()	6	2	x1	4
3	() или {}	9	3	fx	5
4	/	4	4	dfx	9
5	+	3	5	tol	5
6	-	7	6	.0	1
7	*	3	7	dx	5
8	=	13	8	a	2
9	<	1	9	b	3
10	>=	2	10	c	3
11	fabs	3	11	logp	2
12	do ... while ...	1	12	0	1
13	if ...	1	13	-.8858	1
14	if ... else ...	1	14	1.0e-6	1
15	newton	1	15	5.0	1
16	func	1	16	18.19	1
17	log	1	17	-23180.0	1
18	return	1	18	-4.60517	1
Всего		71	Всего		57

Таблица 4 – Программный расчёт характеристик программы на С

№	Оператор	Количество	№	Операнд	Количество
1	;	13	1	x	13
2	()	6	2	x1	5
3	,	20	3	fx	7
4	/	4	4	dfx	11
5	+	3	5	tol	5

6	-	2	6	0.0	1
7	*	3	7	dx	6
8	=	13	8	a	2
9	<	1	9	b	3
10	_&	5	10	c	3
11	>=	1	11	logp	2
12	_*	21	12	newdir	1
13	_-	5	13	0.8858	1
14	__*	8	14	1.0E-6	1
15	fabs	3	15	5.0	1
16	const	1	16	18.19	1
17	dowhile	1	17	23180.0	1
18	int	1	18	4.60517	1
19	double	10			
20	newton	2			
21	func	2			
22	if	2			
23	log	1			
24	main	1			
25	return	1			
26	void	2			
Всего		133	Всего		65

Далее проведем ручной подсчет для программы на Ассемблере. Результат подсчета представлен в табл. 5.

Таблица 5 – Ручной расчёт характеристик программы на Ассемблере

№	Оператор	Количество	№	Операнд	Количество
1	pushl	3	1	%ebp	6
2	movl	36	2	%esp	8

3	subl	3	3	\$40	1
4	fldl LC0	1	4	8(%ebp)	9
5	fldl	15	5	%st	15
6	fdivrp	4	6	%st(1)	17
7	fldl LC1	1	7	-16(%ebp)	2
8	faddp	3	8	%eax	59
9	fstpl	10	9	12(%ebp)	3
10	call _log	1	10	16(%ebp)	4
11	fldl LC2	2	11	\$24	1
12	fmlp	3	12	24(%ebp)	2
13	fldl LC3	1	13	20(%ebp)	6
14	fchs	2	14	8(%esp)	3
15	nop	2	15	4(%esp)	3
16	leave	3	16	%st(0)	3
17	call _func	1	17	\$-16	1
18	fabs	3	18	\$80	1
19	fldl LC5	4	19	72(%esp)	2
20	fucomip	3	20	40(%esp)	1
21	fstp	3	21	16(%esp)	1
22	jbe L3	1	22	56(%esp)	1
23	fldz	1	23	12(%esp)	1
24	fxch	2	24	48(%esp)	1
25	jb L10	1	25	64(%esp)	1
26	jmp L3	1	26	\$0	1
27	fsubrp	2			
28	jnb L7	1			
29	ret	3			
30	andl	1			
31	call _main	1			

32	fldl LC7	1			
33	leal	5			
34	call _newton	1			
Всего		125	Всего		153

Далее произведем расчет метрик Холстеда по имеющимся данным. Результаты для всех языков и вариантов измерения представлены в табл. 6.

Таблица 6 – Сравнение метрик

Характеристика	Ручной расчет			Программный расчет	
	Pascal	C	Asm	Pascal	C
Число простых операторов n_1	16	18	34	20	26
Число простых операндов n_2	12	18	26	18	18
Общее число всех операторов N_1	51	71	125	93	133
Общее число всех операндов N_2	38	57	153	58	65
Словарь n	28	36	60	38	44
Длина $N_{\text{опыт}}$	89	128	278	151	198
Теоретическая длина $N_{\text{теор}}$	107	150.1	295.2	161.5	197.3
Объем V	427.9	661.8	1642.1	792.4	1081
Потенциальный объем V^*	19.7	19.7	104	19.7	19.7
Уровень программы	0.046	0.03	0.063	0.024	0.012

Оценка уровня программы L^{\sim}	0.039	0.035	0.01	0.031	0.021
Интеллектуальное содержание I	16.9	23.2	16.4	24.6	23
Работа программирования E	9315.3	22284	25917	31954	59460
Оценка времени программирования T^{\wedge}	931.5	2228.4	2591.8	1517.2	2808.8
Время программирования T	1083.9	1886	16427	1775.3	3303.4
Уровень языка λ	0.9	0.58	6.59	0.48	0.35
Ожидаемое число ошибок в программе B	0.43	0.66	1.64	0.33	0.51

Проведем сравнение полученных результатов. Как видно из таблицы, самый низкий уровень у программы на Ассемблере с учетом оценки.

Интеллектуальное содержание выше у программ написанных на С и Pascal.

Что касается уровня языка, то, вероятно, во время ручного подсчета характеристик программы на Ассемблере была допущена ошибка, из-за чего получилось, что уровень у Ассемблера самый высокий. Также, может быть, это является последствием автоматической генерации кода.

Как и ожидалось, примерное число ошибок растет с увеличением трудозатрат и времени на программу.

Код программ на Pascal, С и Ассемблере представлен в приложениях А, Б и В соответственно

Выводы.

В ходе выполнения данной работы были на практике изучены методы расчета метрических характеристик качества разработки программ на основе метрик Холстеда. Также был произведен сравнительный анализ результатов для языков Pascal, C и Ассемблер.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ НА PASCAL

```
program newdr;
const
  a = 18.19;
  b = -23180.0;
  c = -0.8858;
  logp = -4.60517;
  tol = 1.0E-6;
var x, fx, dfx, dx, x1 :real;
procedure func(x:      real;
              var fx, dfx: real);
begin
  fx := a + b / x + c * ln(x) - logp;
  dfx := -b / (x * x) + c / x;
end;
procedure newton(var x: real);
begin
  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;
    until (abs(dx) <= abs(tol * x));
  end;
begin
  x := 5.0;
  newton(x);
end.
```

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ НА С

```
#include <math.h>

const double tol = 1.0e-6,
           a = 18.19,
           b = -23180.0,
           c = -.8858,
           logp = -4.60517;

void func(double* x, double* fx, double* dfx)
{
    *fx = a + b / *x + c * log(*x) - logp;
    *dfx = -b / (*x * *x) + c / *x;
}

void newton(double *x, double *fx, double *dx,
            double *dfx, double *x1)
{
    do
    {
        *x1 = *x;
        func(x, fx, dfx);
        if(fabs(*dfx) < tol)
        {
            if(*dfx >= .0)
                *dfx = tol;
            else
                *dfx = -tol;
        }
        *dx = *fx / *dfx;
        *x = *x1 - *dx;
    }
    while(fabs(*dx) >= fabs(tol * *x));
}
```

```
int main()
{
    double x = 5.0, fx, dfx, dx, x1;
    newton(&x, &fx, &dx, &dfx, &x1);
    return 0;
}
```

ПРИЛОЖЕНИЕ В

КОД ПРОГРАММЫ НА ASM

```
.globl _tol
        .section .rdata,"dr"
        .align 8
_tol:
.long -1598689907
.long 1051772663
        .globl _a
        .align 8
_a:
.long -687194767
.long 1077031075
        .globl _b
        .align 8
_b:
.long 0
.long -1059675392
        .globl _c
        .align 8
_c:
.long 1037664099
.long -1075029895
        .globl _logp
        .align 8
_logp:
.long -1355148081
.long -1072534607
        .text
        .globl _func
_func:
LFB0:
        .cfi_startproc
pushl %ebp
        .cfi_def_cfa_offset 8
```

```

        .cfi_offset 5, -8
movl    %esp, %ebp
        .cfi_def_cfa_register 5
subl    $40, %esp
fldl    LC0
movl    8(%ebp), %eax
fldl    (%eax)
fdivrp %st, %st(1)
fldl    LC1
faddp %st, %st(1)
fstpl   -16(%ebp)
movl    8(%ebp), %eax
fldl    (%eax)
fstpl   (%esp)
call    _log
fldl    LC2
fmulp %st, %st(1)
fldl    -16(%ebp)
faddp %st, %st(1)
fldl    LC3
fsubrp %st, %st(1)
movl    12(%ebp), %eax
fstpl   (%eax)
fldl    LC0
fchs
movl    8(%ebp), %eax
fldl    (%eax)
movl    8(%ebp), %eax
fldl    (%eax)
fmulp %st, %st(1)
fdivrp %st, %st(1)
fldl    LC2
movl    8(%ebp), %eax
fldl    (%eax)
fdivrp %st, %st(1)
faddp %st, %st(1)
movl    16(%ebp), %eax

```



```

fstpl (%eax)
    nop
    leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
    ret
        .cfi_endproc
LFE0:
        .globl _newton
_newton:
LFB1:
        .cfi_startproc
    pushl %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
    movl  %esp, %ebp
        .cfi_def_cfa_register 5
    subl  $24, %esp
    L7:
    movl  8(%ebp), %eax
    fldl  (%eax)
    movl  24(%ebp), %eax
    fstpl (%eax)
    movl  20(%ebp), %eax
    movl  %eax, 8(%esp)
    movl  12(%ebp), %eax
    movl  %eax, 4(%esp)
    movl  8(%ebp), %eax
    movl  %eax, (%esp)
    call  _func
    movl  20(%ebp), %eax
    fldl  (%eax)
    fabs
    fldl  LC5
    fucomip    %st(1), %st
    fstp  %st(0)
    jbe  L3

```

```

movl    20(%ebp), %eax
fldl    (%eax)
fldz
fxch    %st(1)
fucomip    %st(1), %st
fstp    %st(0)
jb      L10
fldl    LC5
movl    20(%ebp), %eax
fstpl   (%eax)
jmp     L3
        L10:
fldl    LC5
fchs
movl    20(%ebp), %eax
fstpl   (%eax)
        L3:
movl    12(%ebp), %eax
fldl    (%eax)
movl    20(%ebp), %eax
fldl    (%eax)
fdivrp %st, %st(1)
movl    16(%ebp), %eax
fstpl   (%eax)
movl    24(%ebp), %eax
fldl    (%eax)
movl    16(%ebp), %eax
fldl    (%eax)
fsubrp %st, %st(1)
movl    8(%ebp), %eax
fstpl   (%eax)
movl    16(%ebp), %eax
fldl    (%eax)
fabs
movl    8(%ebp), %eax
fldl    (%eax)
fldl    LC5

```

```

fmulp %st, %st(1)
fabs
fxch %st(1)
fucomip %st(1), %st
fstp %st(0)
jnb L7
    nop
    leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
    ret
        .cfi_endproc
LFE1:
        .globl _main
_main:
LFB2:
        .cfi_startproc
    pushl %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
    movl %esp, %ebp
        .cfi_def_cfa_register 5
    andl $-16, %esp
    subl $80, %esp
    call __main
    fldl LC7
    fstpl 72(%esp)
    leal 40(%esp), %eax
    movl %eax, 16(%esp)
    leal 56(%esp), %eax
    movl %eax, 12(%esp)
    leal 48(%esp), %eax
    movl %eax, 8(%esp)
    leal 64(%esp), %eax
    movl %eax, 4(%esp)
    leal 72(%esp), %eax
    movl %eax, (%esp)

```

```

call    _newton
movl    $0, %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE2:
        .section .rdata,"dr"
        .align 8
LC0:
.long   0
.long   -1059675392
        .align 8
LC1:
.long   -687194767
.long   1077031075
        .align 8
LC2:
.long   1037664099
.long   -1075029895
        .align 8
LC3:
.long   -1355148081
.long   -1072534607
        .align 8
LC5:
.long   -1598689907
.long   1051772663
        .align 8
LC7:
.long   0
.long   1075052544

```