

**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. 8304

Щука А. А.

Преподаватель

Кирияничиков В. А.

Санкт-Петербург

2022

### **Задание.**

Для заданного варианта программы обработки данных, представленной на языке Паскаль, разработать вычислительный алгоритм и 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 "Codegeneration/Generateassemblersource" 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.

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 каждой характеристики 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.
- 2) с помощью программы автоматизации расчета метрик Холстеда (для С-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 анализируемой программы (по умолчанию задается 5), совпадающее с используемым при ручном расчете.

Результаты расчетов представить в виде сводных таблиц с текстовыми комментариями.

### **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	;	13	1	200	1
2	begin... end	6	2	max	1
3	:=	12	3	x	2
4	for...to...do	2	4	i	5
5	if...then	1	5	j	4
6	repeat...until	1	6	n	5
7	+	1	7	hold	2
8	while...end	1	8	a	4
9	>	2	9	jump	5
10	[]	5	10	p	2
11	div	1	11	q	2
12	swap	1	12	1	3
13	sort	1	13	2	1
14	randomize	1	14	done	3
15	()	3	15	100	1
16	random	1	16	false	1
			17	true	1

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	shellsort	1	1	100.0	1
2	swap	1	2	i	8
3	()	11	3	j	4
4	break	1	4	k	12
5	for	4	5	temp	2
6	if...else	1	6	arr	6
7	[]	5	7	num	4
8	=	12	8	my_max	1
9	/	4	9	RAND_MAX	1
10	+	2	10	200	1
11	-	2	11	0	4
12	<	2	12	2	2
13	>	1	13	x	2
14	>=	2	14	y	2
15	;	16			
16	return	1			
17	*	4			
18	rand	1			
19	++	2			
20	&	2			

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	push	3	1	rbp	7
2	mov	41	2	rsp	5
3	movss	12	3	[rbp-24]	8
4	nop	4	4	rdi	4
5	pop	1	5	[rbp-32]	3
6	ret	3	6	rsi	2
7	sub	4	7	rax	15
8	shr	2	8	xmm0	15
9	add	10	9	[rbp-4]	15
10	sar	2	10	32	1
11	jmp	5	11	[rbp-28]	3
12	cdqe	5	12	esi	2
13	lea	5	13	eax	28
14	comiss	1	14	edx	10
15	jnb	1	15	31	2
16	call rand	1	16	[rbp-8]	5
17	cmp	4	17	[rbp-12]	9
18	jns	1	18	rdx	7
19	jl	1	19	[0+rax*4]	4
20	jg	1	20	[rax]	6
21	leave	2	21	xmm1	5
22	pxor	1	22	rcx	2
23	cvtsi2ss	1	23	0	4
24	divss	2	24	1	2
25	jle	1	25	816	1
26	call shellsort(float*, int)	1	26	200	2
27	call swap(float*, float*)	1	27	.LC0[rip]	1
28			28	.LC1[rip]	1
29			29	[rbp-816+rax*4]	1
30			30	199	1
31			31	[rbp-816]	1

В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную

	Паскаль	Си	Ассемблер
Число уникальных операторов (n1):	16	20	27
Число уникальных операндов (n2):	17	14	31
Общее число операторов (N1):	52	75	116
Общее число операндов (N2):	43	50	172
Словарь (n):	33	34	58
Экспериментальная длина программы (Nэ):	95	125	288
Теоретическая длина программы (Nт):	133,5	139,7	281,9
Объём программы (V):	479,2	635,9	1687,1
Потенциальный объём (V*):	11,6	11,6	11,6
Уровень программы (L):	0,024	0,018	0,006
Интеллект программы (I):	23,68	17,81	22,52
Работа по программированию (E):	19780,91	34834,03	245167,06
Время кодирования (T):	1978,09	3483,4	24516,7
Уровень языка программирования (Lam):	0,28	0,21	0,0798
Уровень ошибок (B):	1	1	2

### Расчет метрик с помощью программы автоматизации

Для программы на Паскале:

Operators:	
1   7   ( )	
2   1   +	
3   1   /	
4   30   ;	
5   12   =	
6   2   >	
7   5   []	
8   2   ary	
9   1   boolean	
10   1   const	
11   2   for	
12   1   if	
13   3   integer	

	14		2		procedure
	15		1		program
	16		1		random
	17		1		randomize
	18		3		real
	19		1		repeat
	20		2		sort
	21		2		swap
	22		1		type
	23		1		while

#### Operands:

	1		4		1
	2		1		100
	3		1		2
	4		1		200
	5		1		Shell_sort
	6		5		a
	7		1		ary
	8		4		done
	9		1		false
	10		3		hold
	11		6		i
	12		4		j
	13		6		jump
	14		3		max
	15		7		n
	16		3		p
	17		3		q
	18		1		true
	19		3		x

#### Summary:

```
=====
The number of different operators      : 23
The number of different operands      : 19
The total number of operators          : 83
The total number of operands          : 58
```

Dictionary	( D)	: 42
Length	( N)	: 141
Length estimation	( ^N)	: 184.753
Volume	( V)	: 760.317
Potential volume	( *V)	: 11.6096
Limit volume	( **V)	: 15.6844

Programming level	( L)	: 0.0152695
Programming level estimation	( ^L)	: 0.0284858
Intellect	( I)	: 21.6582
Time of programming	( T)	: 2766.29
Time estimation	( ^T)	: 1942.97
Programming language level	(lambda)	: 0.177273
Work on programming	( E)	: 49793.2
Error	( B)	: 0.451155
Error estimation	( ^B)	: 0.253439

Для программы на Си:

```

Operators:
| 1 | 11 | ()
| 2 | 2 | +
| 3 | 2 | ++
| 4 | 6 | ,
| 5 | 2 | -
| 6 | 4 | /
| 7 | 30 | ;
| 8 | 2 | <
| 9 | 12 | =
| 10 | 1 | >
| 11 | 2 | >=
| 12 | 5 | []
| 13 | 2 | _&
| 14 | 4 | _*
| 15 | 2 | _[]
| 16 | 2 | _*
| 17 | 1 | break
| 18 | 1 | const
| 19 | 8 | float
| 20 | 4 | for
| 21 | 1 | if
| 22 | 5 | int
| 23 | 1 | main
| 24 | 1 | rand
| 25 | 1 | return
| 26 | 2 | shellsort
| 27 | 2 | swap
| 28 | 2 | void

Operands:
| 1 | 4 | 0
| 2 | 1 | 100.0
| 3 | 2 | 2

```



	4		1		200
	5		1		RAND_MAX
	6		8		arr
	7		10		i
	8		5		j
	9		14		k
	10		2		my_max
	11		7		num
	12		3		temp
	13		3		x
	14		3		y

#### Summary:

=====

The number of different operators	:	28
The number of different operands	:	14
The total number of operators	:	118
The total number of operands	:	64

Dictionary	( D)	:	42
Length	( N)	:	182
Length estimation	( ^N)	:	187.909
Volume	( V)	:	981.402
Potential volume	( *V)	:	11.6096
Limit volume	( **V)	:	15.6844
Programming level	( L)	:	0.0118297
Programming level estimation	( ^L)	:	0.015625
Intellect	( I)	:	15.3344
Time of programming	( T)	:	4608.95
Time estimation	( ^T)	:	3602.72
Programming language level	(lambda)	:	0.137338
Work on programming	( E)	:	82961.2
Error	( B)	:	0.634058
Error estimation	( ^B)	:	0.327134

Таблица 5 – Сводная таблица

	Паскаль вручную	Паскаль программно	Си вручную	Си программно	Ассемблер
Число уникальных операторов (n1):	16	23	20	28	27
Число уникальных операндов (n2):	17	19	14	14	31
Общее число операторов (N1):	52	83	75	118	116
Общее число операндов (N2):	43	58	50	64	172
Словарь (n):	33	42	34	42	58
Экспериментальная длина программы (Nэ):	95	141	125	182	288
Теоретическая длина программы (Nт):	133,5	184,75	139,7	187,9	281,9
Объём программы (V):	479,2	760,3	635,9	981,4	1687,1
Потенциальный объём (V*):	11,6	11,6	11,6	11,6	11,6
Уровень программы (L):	0,024	0,015	0,018	0,011	0,006
Интеллект программы (I):	23,68	21,65	17,81	15,33	22,52
Работа по программированию (E):	19780,91	49793,2	34834,03	82961,2	245167,06
Время кодирования (T):	1978,09	2766,29	3483,4	4608,95	24516,7
Уровень языка программирования (Lam):	0,28	0,177	0,21	0,137	0,0798
Уровень ошибок (B):	1	1	1	1	2

**Выводы.**

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 Shell_sort;
const   max       = 200;
type    ary       = array[1..max] of real;

var      x          : ary;
         i,n        : integer;

procedure sort(var a: ary; n: integer);

var done : boolean;
    jump,i,j: integer;

procedure swap(var p,q: real);
var hold : real;

begin
    hold:=p;
    p:=q;
    q:=hold
end;

begin
    jump:=n;
    while jump>1 do
        begin
            jump:=jump div 2;
            repeat
                done:=true;
                for j:=1 to n do
                    begin
                        i:=j+jump;
                        if a[j]>a[i] then
                            begin
                                swap(a[j],a[i]);
                                done:=false
                            end
                        end
                    until done
                end
            end;
        end;
    end;

begin
    n:=max;
    randomize;
    for i:=1 to n do
        x[i]:= random(100);
    sort( x,n );
end.
```

## ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

void swap(float* x, float* y)
{
    float temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

void shellsort(float arr[], int num)
{
    int i, j, k;
    for (i = num / 2; i > 0; i = i / 2)
    {
        for (j = i; j < num; j++)
        {
            for (k = j - i; k >= 0; k = k - i)
            {
                if (arr[k + i] >= arr[k])
                    break;
                else
                {
                    swap(&arr[k], &arr[k + i]);
                }
            }
        }
    }
}

int main()
{
    const int num = 200;
    float my_max = 100.0;
    float arr[num];
    int k;

    for (k = 0; k < num; k++)
    {
        arr[k] = (float)rand() / (float)(RAND_MAX / my_max);
    }
    shellsort(arr, num);
    return 0;
}
```

## ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
swap(float*, float*):
    push    rbp
    mov     rbp, rsp
    mov     QWORD PTR [rbp-24], rdi
    mov     QWORD PTR [rbp-32], rsi
    mov     rax, QWORD PTR [rbp-24]
    movss   xmm0, DWORD PTR [rax]
    movss   DWORD PTR [rbp-4], xmm0
    mov     rax, QWORD PTR [rbp-32]
    movss   xmm0, DWORD PTR [rax]
    mov     rax, QWORD PTR [rbp-24]
    movss   DWORD PTR [rax], xmm0
    mov     rax, QWORD PTR [rbp-32]
    movss   xmm0, DWORD PTR [rbp-4]
    movss   DWORD PTR [rax], xmm0
    nop
    pop     rbp
    ret

shellsort(float*, int):
    push    rbp
    mov     rbp, rsp
    sub     rsp, 32
    mov     QWORD PTR [rbp-24], rdi
    mov     DWORD PTR [rbp-28], esi
    mov     eax, DWORD PTR [rbp-28]
    mov     edx, eax
    shr     edx, 31
    add     eax, edx
    sar     eax
    mov     DWORD PTR [rbp-4], eax
    jmp     .L3
.L11:
    mov     eax, DWORD PTR [rbp-4]
    mov     DWORD PTR [rbp-8], eax
    jmp     .L4
.L10:
    mov     eax, DWORD PTR [rbp-8]
    sub     eax, DWORD PTR [rbp-4]
    mov     DWORD PTR [rbp-12], eax
    jmp     .L5
.L9:
    mov     edx, DWORD PTR [rbp-12]
    mov     eax, DWORD PTR [rbp-4]
    add     eax, edx
    cdqe
    lea     rdx, [0+rax*4]
    mov     rax, QWORD PTR [rbp-24]
    add     rax, rdx
    movss   xmm0, DWORD PTR [rax]
    mov     eax, DWORD PTR [rbp-12]
    cdqe
```

```

    lea    rdx, [0+rax*4]
    mov    rax, QWORD PTR [rbp-24]
    add    rax, rdx
    movss  xmm1, DWORD PTR [rax]
    comiss xmm0, xmm1
    jnb    .L12
    mov    edx, DWORD PTR [rbp-12]
    mov    eax, DWORD PTR [rbp-4]
    add    eax, edx
    cdqe
    lea    rdx, [0+rax*4]
    mov    rax, QWORD PTR [rbp-24]
    add    rdx, rax
    mov    eax, DWORD PTR [rbp-12]
    cdqe
    lea    rcx, [0+rax*4]
    mov    rax, QWORD PTR [rbp-24]
    add    rax, rcx
    mov    rsi, rdx
    mov    rdi, rax
    call   swap(float*, float*)
    mov    eax, DWORD PTR [rbp-4]
    sub    DWORD PTR [rbp-12], eax
.L5:
    cmp    DWORD PTR [rbp-12], 0
    jns    .L9
    jmp    .L8
.L12:
    nop
.L8:
    add    DWORD PTR [rbp-8], 1
.L4:
    mov    eax, DWORD PTR [rbp-8]
    cmp    eax, DWORD PTR [rbp-28]
    jl     .L10
    mov    eax, DWORD PTR [rbp-4]
    mov    edx, eax
    shr    edx, 31
    add    eax, edx
    sar    eax
    mov    DWORD PTR [rbp-4], eax
.L3:
    cmp    DWORD PTR [rbp-4], 0
    jg     .L11
    nop
    nop
    leave
    ret
main:
    push   rbp
    mov    rbp, rsp
    sub    rsp, 816
    mov    DWORD PTR [rbp-8], 200

```

```

        movss    xmm0, DWORD PTR .LC0[rip]
        movss    DWORD PTR [rbp-12], xmm0
        mov      DWORD PTR [rbp-4], 0
        jmp      .L14
.L15:
        call     rand
        pxor     xmm0, xmm0
        cvtsi2ss    xmm0, eax
        movss    xmm1, DWORD PTR .LC1[rip]
        divss    xmm1, DWORD PTR [rbp-12]
        divss    xmm0, xmm1
        mov      eax, DWORD PTR [rbp-4]
        cdqe
        movss    DWORD PTR [rbp-816+rax*4], xmm0
        add      DWORD PTR [rbp-4], 1
.L14:
        cmp      DWORD PTR [rbp-4], 199
        jle      .L15
        lea      rax, [rbp-816]
        mov      esi, 200
        mov      rdi, rax
        call     shellsort(float*, int)
        mov      eax, 0
        leave
        ret
.LC0:
        .long    1120403456
.LC1:
        .long    1325400064

```