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. 6304 Некрасов Н.А.

Преподаватель Кирьянчиков В.А.

Санкт-Петербург
2020

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, потенциальный и граничный 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	;	12	1	5	1
2	begin... end	3	2	I	6
3	:=	9	3	J	6
4	for⋯to⋯do	1	4	K	4
5	if⋯then	1	5	Y	5
6	()	1	6	abc	1
7	+	1	7	size	1
8	[]	8	8	5	1
9	<	1	9	123	1
10	-	2	10	4326	1
11	integer	2	11	1	4
12	program	1	12	4	1
13	var	1	13	2	1
14	array .. of	1	14	arr	1

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

№.	Оператор	Число вхождений	№.	Операнд	Число вхождений
1	()	4	1	0	3
2	+	1	2	1	2
3	++	3	3	CURR	2
4	,	7	4	MAX	1
5	-	1	5	NULL	1
6	<	3	6	array	10
7	=	9	7	c	1
8	>	1	8	d	1
9	[]	7	9	i	11
10	_[]	2	10	j	5
11	for	3	11	maxIndex	5

12	if	1	12	n	1
13	main	1	13	position	1
14	rand	1	14	size	3
15	return	1	15	t	1
16	selectionSort	2	16	tmp	2
17	srand	1			
18	time	1			

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	__main	2	1	$0	4
2	___chkstk_ms	1	2	$1	5
3	addl	4	3	$16	2
4	addq	8	4	$19472	2
5	call	6	5	$4	1
6	cltq	7	6	$5	1
7	cmpl	4	7	%eax	27
8	jl	2	8	%ecx	2
9	jle	2	9	%edx	10
10	jmp	3	10	%rax	22
11	leaq	8	11	%rbp	6
12	main	3	12	%rcx	6
13	movl	33	13	%rdx	8
14	movq	9	14	%rsp	5
15	nop	1	15	(%rax)	5
16	popq	2	16	(%rdx)	1
17	pushq	2	17	-12(%rbp)	5
18	ret	2	18	-16(%rbp)	2
19	selectionSort	5	19	-4(%rbp)	7
20	srand	2	20	-8(%rbp)	5
21	subl	1	21	-96(%rbp)	1
22	subq	2	22	-96(%rbp,%rax,4)	1
23	time	2	23	0(,%rax,4)	6
			24	128(%rsp)	1
			25	16(%rbp)	7
			26	19340(%rbp)	4
			27	24(%rbp)	3

В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную

	Паскаль	**Си**	**Ассемблер**
Число уникальных операторов (n1):	14	18	23
Число уникальных операндов (n2):	14	16	27
Общее число операторов (N1):	44	49	111
Общее число операндов (N2):	50	50	149
Алфавит (n):	28	34	50
Экспериментальная длина программы (Nэ):	94	99	260
Теоретическая длина программы (Nт):	106.6059	139.06	232.4239
Объём программы (V):	451.8913	503.6588	1467.403
Потенциальный объём (V*):	11.6096	11.6096	11.6096
Уровень программы (L):	0.0256	0.023	0.00799
Сложность программы (S):	39.0625	43.47826	125.1564
Ожидание уровня программы (L^):	0.067	0.04	0.015757
Интеллект программы (I):	18.0756	18.0756	18.0756
Работа по программированию (E):	17589.33	21850.13	185472.6
Время кодирования (T):	1758.933	2185.013	18547.26
Ожидание времени кодирования (T^):	282.6	1105.4	9312.572
Уровень языка программирования (Lam):	0.2982	0.2676	0.0919
Уровень ошибок (B):	1	1	2

Расчет метрик с помощью программы автоматизации

Для программы на Паскале:

```
Statistics for module output.lxm
=====================================
The number of different operators     : 9
The number of different operands      : 13
The total number of operators         : 27
The total number of operands          : 43

Dictionary                    (  D)   : 22
Length                        (  N)   : 70
Length estimation             ( ^N)   : 76.635
Volume                        (  V)   : 312.16
Potential volume              ( *V)   : 19.6515
Limit volume                  (**V)   : 38.2071
Programming level             (  L)   : 0.0629532
Programming level estimation  ( ^L)   : 0.0671835
Intellect                     (  I)   : 20.972
Time of programming           (  T)   : 275.478
Time estimation               ( ^T)   : 282.6
Programming language level    (lambda) : 1.23712
Work on programming           (  E)   : 4958.61
Error                         (  B)   : 0.0969286
Error estimation              ( ^B)   : 0.104053

Table:
=====================================
Operators:
|  1    |  5    | ()
|  2    |  1    | +
|  3    |  2    | -
|  4    |  1    | <
|  5    |  8    | =
|  6    |  6    | []
|  7    |  2    | for
|  8    |  1    | if
|  9    |  1    | program
```

```
Operands:
|  1    |  5    | 1
|  2    |  1    | 123
|  3    |  1    | 2
|  4    |  1    | 4
|  5    |  1    | 4326
|  6    |  2    | 5
|  7    |  5    | I
|  8    |  5    | J
|  9    |  4    | K
|  10   |  5    | Y
|  11   |  1    | abc
|  12   |  8    | arr
|  13   |  4    | size
```

Для программы на Си:

```
Statistics for module output.lxm
=====================================
The number of different operators     : 18
The number of different operands      : 16
The total number of operators         : 49
The total number of operands          : 50

Dictionary                   (  D)    : 34
Length                       (  N)    : 99
Length estimation            ( ^N)    : 139.059
Volume                       (  V)    : 503.659
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.0390175
Programming level estimation ( ^L)    : 0.0355556
Intellect                    (  I)    : 17.9079
Time of programming          (  T)    : 717.142
Time estimation              ( ^T)    : 1105.4
Programming language level   (lambda) : 0.766751
Work on programming          (  E)    : 12908.6
Error                        (  B)    : 0.183427
Error estimation             ( ^B)    : 0.167886


Table:
```

```
==================================
Operators:
|  1    |  4    | ()
|  2    |  1    | +
|  3    |  3    | ++
|  4    |  7    | ,
|  5    |  1    | -
|  6    |  3    | <
|  7    |  9    | =
|  8    |  1    | >
|  9    |  7    | []
|  10   |  2    | _[]
|  11   |  3    | for
|  12   |  1    | if
|  13   |  1    | main
|  14   |  1    | rand
|  15   |  1    | return
|  16   |  2    | selectionSort
|  17   |  1    | srand
|  18   |  1    | time
Operands:
|  1    |  3    | 0
|  2    |  2    | 1
|  3    |  2    | CURR
|  4    |  1    | MAX
|  5    |  1    | NULL
|  6    |  10   | array
|  7    |  1    | c
|  8    |  1    | d
|  9    |  11   | i
|  10   |  5    | j
|  11   |  5    | maxIndex
|  12   |  1    | n
|  13   |  1    | position
|  14   |  3    | size
|  15   |  1    | t
|  16   |  2    | tmp
```

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 abc;
var arr:  array [1..5] of integer;
    I,
    J,
    K,
    Y,
    size : integer;


begin
    size := 5;
    arr := [123, 4326,1,4,-2];
    for I := 1 to size - 1 do
         begin
              K := I;
              Y := arr[I];
              for J := (I + 1) to size do
              if (arr[J] < Y) then
              begin
                   K := J;
                   Y := arr[J]
              end;
              arr[K] := arr[J];
              arr[I] := Y;
         end;
end.
```

ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

#define MAX 4859
#define CURR 5

void selectionSort(int array[], int size) {
    for (int i = 0; i < size - 1; i++) {
         int maxIndex = i;
         for (int j = i + 1; j < size; j++) {
              if (array[j] > array[maxIndex])
                   maxIndex = j;
         }
         int tmp = array[i];
         array[i] = array[maxIndex];
         array[maxIndex] = tmp;
    }
}

int main()
{
    srand(time(NULL));
    int array[MAX], n, c, d, position, t;

    for(int i = 0; i < CURR; ++i)
         array[i] = rand();
    selectionSort(array, CURR);

    return 0;
}
```

ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
    .file   "one.c"
    .text
    .globl  selectionSort
    .def selectionSort; .scl 2;  .type   32;  .endef
    .seh_proc selectionSort
selectionSort:
    pushq   %rbp
    .seh_pushreg  %rbp
    movq %rsp, %rbp
    .seh_setframe  %rbp, 0
    subq $16, %rsp
    .seh_stackalloc   16
    .seh_endprologue
    movq %rcx, 16(%rbp)
    movl %edx, 24(%rbp)
    movl $0, -4(%rbp)
    jmp  .L2
.L6:
    movl -4(%rbp), %eax
    movl %eax, -8(%rbp)
    movl -4(%rbp), %eax
    addl $1, %eax
    movl %eax, -12(%rbp)
    jmp  .L3
.L5:
    movl -12(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq 16(%rbp), %rax
    addq %rdx, %rax
    movl (%rax), %edx
    movl -8(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rcx
    movq 16(%rbp), %rax
    addq %rcx, %rax
    movl (%rax), %eax
    cmpl %eax, %edx
    jle  .L4
    movl -12(%rbp), %eax
    movl %eax, -8(%rbp)
.L4:
    addl $1, -12(%rbp)
.L3:
    movl -12(%rbp), %eax
    cmpl 24(%rbp), %eax
    jl   .L5
    movl -4(%rbp), %eax
```

```
    cltq
    leaq 0(,%rax,4), %rdx
    movq 16(%rbp), %rax
    addq %rdx, %rax
    movl (%rax), %eax
    movl %eax, -16(%rbp)
    movl -8(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq 16(%rbp), %rax
    addq %rax, %rdx
    movl -4(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rcx
    movq 16(%rbp), %rax
    addq %rcx, %rax
    movl (%rdx), %edx
    movl %edx, (%rax)
    movl -8(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq 16(%rbp), %rax
    addq %rdx, %rax
    movl -16(%rbp), %edx
    movl %edx, (%rax)
    addl $1, -4(%rbp)
.L2:
    movl 24(%rbp), %eax
    subl $1, %eax
    cmpl %eax, -4(%rbp)
    jl   .L6
    nop
    addq $16, %rsp
    popq %rbp
    ret
    .seh_endproc
    .def __main;   .scl 2;   .type     32;  .endef
    .globl    main
    .def main;     .scl 2;   .type     32;  .endef
    .seh_proc main
main:
    pushq     %rbp
    .seh_pushreg   %rbp
    movl $19472, %eax
    call ___chkstk_ms
    subq %rax, %rsp
    .seh_stackalloc     19472
    leaq 128(%rsp), %rbp
    .seh_setframe  %rbp, 128
    .seh_endprologue
```

```
        call __main
        movl $0, %ecx
        call time
        movl %eax, %ecx
        call srand
        movl $0, 19340(%rbp)
        jmp  .L8
    .L9:
        call rand
        movl %eax, %edx
        movl 19340(%rbp), %eax
        cltq
        movl %edx, -96(%rbp,%rax,4)
        addl $1, 19340(%rbp)
    .L8:
        cmpl $4, 19340(%rbp)
        jle  .L9
        leaq -96(%rbp), %rax
        movl $5, %edx
        movq %rax, %rcx
        call selectionSort
        movl $0, %eax
        addq $19472, %rsp
        popq %rbp
        ret
        .seh_endproc
        .ident    "GCC: (x86_64-posix-sjlj-rev0, Built by MinGW-W64
project) 8.1.0"
        .def time;     .scl 2;   .type     32;  .endef
        .def srand;    .scl 2;   .type     32;  .endef
        .def rand;     .scl 2;   .type     32;  .endef
```