

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

Тимофеев А.А.

Преподаватель

Кирияничков 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.

Вариант 15

Приближенная линейаризация опытных данных (вар.2).

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	№	Операнд	Количество
1	=	1	1	n	14
2	;	40	2	x	6
3	:	10	3	y	5
4	:=	27	4	i	13
5	()	12	5	a	6
6	[]	8	6	b	6
7	for to do	3	7	calc	2
8	+	8	8	y_calc	2
9	*	13	9	sum_x	8
10	/	10	10	sum_y	9
11	randomize	1	11	sum_xy	7
12	random	2	12	sum_x2	7
13	linfit2	2	13	sum_y2	6
14	program	1	14	xi	6
15	procedure	1	15	yi	6
16	begin end	4	16	sxy	4
Всего		143	17	syy	3
			18	correl_coef	2
			19	see	3
			20	sigma_b	3
			21	sigma_a	2

22	sxx	6
23	linear	1
24	100	1
25	0.0	5
26	1	7
27	2	1
Всего		141

Программный расчёт измеримых характеристик представлен в таблицу

2. Файл с результатами программных расчётов представлен в приложении Б.

Таблица 2 – Программный расчёт измеримых характеристик (Pascal)

№	Оператор	Количество	№	Операнд	Количество
1	()	14	1	0.0	5
2	*	13	2	1	7
3	+	8	3	100	1
4	-	7	4	2	1
5	/	10	5	a	6
6	;	62	6	b	6
7	=	25	7	calc	2
8	[]	6	8	correl_coef	2
9	const	1	9	i	10
10	for	3	10	linear	1
11	integer	3	11	n	14
12	linfit2	2	12	see	3
13	procedure	1	13	sigma_a	2
14	program	1	14	sigma_b	3
15	random	2	15	sum_x	8
16	randomize	1	16	sum_x2	7
17	real	5	17	sum_xy	7
18	sqrt	4	18	sum_y	9
Всего		170	19	sum_y2	6
			20	sxx	6
			21	sxy	4

22	sy	3
23	x	6
24	xi	6
25	y	5
26	y_calc	2
27	yi	6
Всего		138

Определение расчетных характеристик представлено в таблице 3.

Таблица 3 – Расчёт расчетных характеристик (Pascal)

Характеристика	Ручной расчёт	Программный расчёт
Число простых операторов n_1	16	18
Число простых операндов n_2	27	27
Общее число всех операторов N_1	143	170
Общее число всех операндов N_2	141	138
Словарь n	43	45
Длина $N_{\text{опыт}}$	284	308
Теоретическая длина $N_{\text{теор}}$	192.382	203.441
Объём V	1541.059	1691.491
Потенциальный объём V^*	24	24
Уровень программы L	0.0156	0.0142
Оценка уровня программы L^{\sim}	0.024	0.021
Интеллектуальное содержание I	36.887	36.772
Работа программирования E	98952.643	119214.207
Оценка времени программирования T^{\wedge}	9895.264	11921.421
Время программирования T	6438.13	7781.066
Уровень языка λ	0.374	0.341
Ожидаемое число ошибок в программе B	2	2

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	№	Операнд	Количество
1	;	38	1	100	3
2	=	27	2	sum_x	6
3	()	23	3	sum_y	7
4	[]	9	4	sum_xy	5
5	for	3	5	sum_x2	5
6	<	6	6	sum_y2	4
7	+	1	7	i	15
8	++	3	8	xi	5
9	+=	5	9	yi	5
10	-	7	10	sxx	5
11	/	10	11	sxy	3
12	%	2	12	syy	2
13	*	24	13	correl_coef	1
14	&	2	14	see	2
15	return	1	15	sigma_b	2
16	linfit2	2	16	sigma_a	1
17	srand	1	17	x	6
18	time	1	18	y	5
19	rand	1	19	0.0	10
20	{}	8	20	0	4
Всего		174	21	a	6
			22	b	6
			23	N	7
			24	n	8
			25	y_calc	4
			26	NULL	1
			27	2	1
			Всего		129

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	№	Операнд	Количество
1	%	2	1	0	4
2	()	17	2	0.0	10
3	*	13	3	100	2
4	+	1	4	2	1
5	++	3	5	NULL	1
6	+=	5	6	N	6
7	,	10	7	a	6
8	-	7	8	b	6
9	/	10	9	correl_coef	1
10	;	44	10	i	15
11	<	3	11	n	8
12	=	27	12	see	2
13	[]	6	13	sigma_a	1
14	_&	2	14	sigma_b	2
15	_*	6	15	sum_x	6
16	_[]	3	16	sum_x2	5
17	__*	5	17	sum_xy	5
18	float	24	18	sum_y	7
19	for	3	19	sum_y2	4
20	int	5	20	sxx	5
21	linfit2	2	21	sxy	3
22	main	1	22	syy	2
23	rand	2	23	x	6
24	return	1	24	xi	5
25	sqrt	4	25	y	5
26	srand	1	26	y_calc	4
27	time	1	27	yi	5
28	void	1	Всего		127
Всего		209			

Определение расчетных характеристик представлено в таблице 6.

Таблица 6 – Расчёт расчетных характеристик (Си)

Характеристика	Ручной расчёт	Программный расчёт
Число простых операторов n_1	20	28
Число простых операндов n_2	27	27
Общее число всех операторов N_1	174	209
Общее число всех операндов N_2	129	127
Словарь n	47	55
Длина $N_{\text{опыт}}$	303	336
Теоретическая длина $N_{\text{теор}}$	214.821	262.988
Объём V	1683.04	1942.54
Потенциальный объём V^*	24.0	24.0
Уровень программы L	0.014	0.012
Оценка уровня программы L^{\sim}	0.021	0.015
Интеллектуальное содержание I	35.226	29.499
Работа программирования E	118026.044	157227.061
Оценка времени программирования T^{\wedge}	11802.604	15722.706
Время программирования T	8041.2	18057.1
Уровень языка λ	0.342	0.297
Ожидаемое число ошибок в программе B	2	2

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	№	Операнд	Количество
1	pushq	2	1	%rbp	10

2	movq	31	2	16	10
3	movl	30	3	-16	4
4	xorps	2	4	%rsp	4
5	movss	53	5	%rdi	7
6	cmpl	3	6	%rsi	12
7	jge	3	7	-16(%rbp)	2
8	movslq	6	8	%rdx	11
9	addss	6	9	-8(%rbp)	5
10	mulss	12	10	-24(%rbp)	2
11	addl	3	11	%rcx	22
12	cvtss2ssl	8	12	-32(%rbp)	4
13	divss	8	13	%r8	2
14	subss	6	14	-40(%rbp)	4
15	cvtss2sd	7	15	-44(%rbp)	8
16	sqrtsd	4	16	%r9d	2
17	divsd	2	17	%xmm0	90
18	cvtss2ss	4	18	-48(%rbp)	7
19	jmp	3	19	-52(%rbp)	8
20	popq	2	20	-56(%rbp)	6
21	retq	2	21	-60(%rbp)	6
22	subq	1	22	-64(%rbp)	5
23	xorl	3	23	\$0	4
24	leaq	8	24	-68(%rbp)	6
25	callq	9	25	\$100	5
26	idivl	2	26	%rax	12
27	cmpq	1	27	-72(%rbp)	5
28	jne	1	28	-76(%rbp)	5
29	addq	1	29	%eax	34
30	ud2	1	30	%xmm1	48
31	cld	2	31	%xmm2	8
Bcero		226	32	-80(%rbp)	5
			33	-84(%rbp)	3
			34	-88(%rbp)	2
			35	-92(%rbp)	1

36	\$2	1
37	-96(%rbp)	2
38	-100(%rbp)	2
39	-104(%rbp)	1
40	-108(%rbp)	6
41	\$1264	2
42	\$400	1
43	-1220(%rbp)	1
44	-416(%rbp)	2
45	-1240(%rbp)	2
46	-1248(%rbp)	3
47	-1252(%rbp)	3
48	-1216(%rbp)	2
49	-1224(%rbp)	2
50	-1228(%rbp)	2
51	-1232(%rbp)	6
52	-816(%rbp,%rsi,4)	1
53	-416(%rbp,%rsi,4)	1
54	(%rax,%rcx,4)	4
Всего		413

Определение расчетных характеристик представлено в таблице 8.

Таблица 8 – Расчёт расчетных характеристик (Ассемблер)

Характеристика	Ручной расчёт
Число простых операторов n_1	31
Число простых операндов n_2	54
Общее число всех операторов N_1	226
Общее число всех операндов N_2	413
Словарь n	85
Длина $N_{\text{опыт}}$	639
Теоретическая длина $N_{\text{теор}}$	464.344
Объём V	4095.6

Потенциальный объём V^*	24
Уровень программы L	0.006
Оценка уровня программы L^{\sim}	0.0084
Интеллектуальное содержание I	34.4
Работа программирования E	698914.14
Оценка времени программирования T^{\wedge}	69891.14
Время программирования T	48551.9
Уровень языка λ	0.141
Ожидаемое число ошибок в программе B	5

4. 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	16	18	20	28	31
Число простых операндов n_2	27	27	27	27	54
Общее число всех операторов N_1	143	170	174	209	226
Общее число всех операндов N_2	141	138	129	127	413
Словарь n	43	45	47	55	85
Длина $N_{\text{опыт}}$	284	308	303	336	639
Теоретическая длина $N_{\text{теор}}$	192.382	203.441	214.821	262.988	464.344
Объём V	1541.059	1691.491	1683.04	1942.54	4095.6
Потенциальный объём V^*	24	24	24.0	24.0	24
Уровень программы	0.0156	0.0142	0.014	0.012	0.006

Оценка уровня программы L^{\sim}	0.024	0.021	0.021	0.015	0.0084
Интеллектуальное содержание I	36.887	36.772	35.226	29.499	34.4
Работа программирования E	98952.643	119214.207	118026.044	157227.061	698914.14
Оценка времени программирования T^{\wedge}	9895.264	11921.421	11802.604	15722.706	69891.14
Время программирования T	6438.13	7781.066	8041.2	18057.1	48551.9
Уровень языка λ	0.374	0.341	0.342	0.297	0.141
Ожидаемое число ошибок в программе B	2	2	2	2	5

При анализе таблицы становится очевидно, что наименьшим уровнем программы обладает программа на Ассемблере, а наивысшим – программа на Паскале. Наоборот наибольшие показатели ожидаемого количества ошибок, времени и работы программирования также соответствуют программе на Ассемблере, а наименьшие – программе на Паскале.

Выводы

В ходе выполнения данной лабораторной работы были получены практические навыки по определению метрик Холстеда для программ, разработанных на Паскале, Си и Ассемблере. Был произведен сравнительный анализ полученных результатов.

ПРИЛОЖЕНИЕ А

Код программы на Pascal.

```
program linear;
const
  n = 100;
var
  x, y: array[1..n] of real;
  i: integer;
  a, b : real;
  calc: array[1..n] of real;

procedure linfit2(
  x,y:      array of real;
  var y_calc: array of real;
  var a,b:   real;
  n:        integer);

var i      : integer;

sum_x,sum_y,sum_xy,sum_x2,
sum_y2,xi,yi,sxy,syy,
correl_coef, see, sigma_b, sigma_a,
sxx      : real;

begin
  sum_x := 0.0;
  sum_y := 0.0;
  sum_xy := 0.0;
  sum_x2 := 0.0;
  sum_y2 := 0.0;
  for i := 1 to n do
    begin
      xi := x[i];
      yi := y[i];

      sum_x := sum_x+xi;
      sum_y := sum_y+yi;
      sum_xy := sum_xy+xi*yi;
      sum_x2 := sum_x2+xi*xi;
      sum_y2 := sum_y2+yi*yi;
    end;
end;
```

```

sxx := sum_x2-sum_x*sum_x/n;
sxy := sum_xy-sum_x*sum_y/n;
syy := sum_y2-sum_y*sum_y/n;
b := sxy/sxx;
a := ((sum_x2*sum_y-sum_x*sum_xy)/n)/sxx;
correl_coef := sxy/sqrt(sxx*syy);
see := sqrt((sum_y2-a*sum_y-b*sum_xy)/(n-2));
sigma_b := see/sqrt(sxx);
sigma_a := sigma_b*sqrt(sum_x2/n);

for i := 1 to n do
    y_calc[i] := a+b*x[i]
end;

begin
    randomize;
    for i:= 1 to n do
        begin
            x[i] := random(i) + 1;
            y[i] := random(i) + 1;
        end;
    linfit2(x, y, calc, a, b, n);
end.

```

ПРИЛОЖЕНИЕ Б

Результаты расчетов для программы на Паскале

Statistics for module out_pas.lxm

=====

The number of different operators : 19
The number of different operands : 27
The total number of operators : 170
The total number of operands : 138

Dictionary (D) : 46
Length (N) : 308
Length estimation (^N) : 209.093
Volume (V) : 1701.26
Potential volume (*V) : 11.6096
Limit volume (**V) : 15.6844
Programming level (L) : 0.00682415
Programming level estimation (^L) : 0.020595
Intellect (I) : 35.0373
Time of programming (T) : 13850
Time estimation (^T) : 3115.48
Programming language level (lambda) : 0.079226
Work on programming (E) : 249299
Error (B) : 1.32036
Error estimation (^B) : 0.567086

Table:

=====

Operators:

1	14	()
2	13	*
3	8	+
4	7	-
5	10	/
6	62	;
7	25	=
8	6	[]
9	2	ary
10	1	const
11	3	for
12	3	integer

	13		2		linfit2
	14		1		procedure
	15		1		program
	16		2		random
	17		1		randomize
	18		5		real
	19		4		sqrt

Operands:

	1		5		0.0
	2		7		1
	3		1		100
	4		1		2
	5		6		a
	6		6		b
	7		2		calc
	8		2		correl_coef
	9		10		i
	10		1		linear
	11		14		n
	12		3		see
	13		2		sigma_a
	14		3		sigma_b
	15		8		sum_x
	16		7		sum_x2
	17		7		sum_xy
	18		9		sum_y
	19		6		sum_y2
	20		6		sxx
	21		4		sxy
	22		3		syy
	23		6		x
	24		6		xi
	25		5		y
	26		2		y_calc
	27		6		yi

Summary:

=====

The number of different operators	: 19
The number of different operands	: 27

The total number of operators : 170
The total number of operands : 138

Dictionary	(D)	: 46
Length	(N)	: 308
Length estimation	(^N)	: 209.093
Volume	(V)	: 1701.26
Potential volume	(*V)	: 11.6096
Limit volume	(**V)	: 15.6844
Programming level	(L)	: 0.00682415
Programming level estimation	(^L)	: 0.020595
Intellect	(I)	: 35.0373
Time of programming	(T)	: 13850
Time estimation	(^T)	: 3115.48
Programming language level	(lambda)	: 0.079226
Work on programming	(E)	: 249299
Error	(B)	: 1.32036
Error estimation	(^B)	: 0.567086

ПРИЛОЖЕНИЕ В

Код программы на Си

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

#define N 100

void linfit2(float *x, float *y, float *y_calc, float *a, float *b, int n) {
    float sum_x = 0.0;
    float sum_y = 0.0;
    float sum_xy = 0.0;
    float sum_x2 = 0.0;
    float sum_y2 = 0.0;
    for (int i = 0; i < N; i++) {
        float xi = x[i];
        float yi = y[i];
        sum_x += xi;
        sum_y += yi;
        sum_xy += (xi * yi);
        sum_x2 += (xi * xi);
        sum_y2 += (yi * yi);
    }

    float sxx = sum_x2 - sum_x * sum_x / n;
    float sxy = sum_xy - sum_x * sum_y / n;
    float syy = sum_y2 - sum_y * sum_y / n;

    (*b) = sxy / sxx;
    (*a) = ( (sum_x2 * sum_y - sum_x * sum_xy) / n ) / sxx;
    float correl_coef = sxy / sqrt(sxx * syy);
    float see = sqrt( (sum_y2 - *a * sum_y - *b * sum_xy) / (n - 2) );
    float sigma_b = see / sqrt(sxx);
    float sigma_a = sigma_b * sqrt(sum_x2 / n);
    for (int i = 0; i < n; i++) {
        y_calc[i] = *a + *b * x[i];
    }
}

int main() {

    float x[N] = {0.0};
    float y[N] = {0.0};

    float y_calc[N] = {0.0};

    float a = 0.0;
    float b = 0.0;

    srand(time(NULL));

    for (int i = 0; i < N; i++) {
        x[i] = rand() % 100;
        y[i] = rand() % 100;
    }

    linfit2(x, y, y_calc, &a, &b, N);

    return 0;
}
```

ПРИЛОЖЕНИЕ Г

Результаты расчетов для программы на Си

Statistics for module out_c.lxm

=====

The number of different operators : 28
The number of different operands : 27
The total number of operators : 209
The total number of operands : 127

Dictionary (D) : 55
Length (N) : 336
Length estimation (^N) : 262.988
Volume (V) : 1942.54
Potential volume (*V) : 11.6096
Limit volume (**V) : 15.6844
Programming level (L) : 0.00597654
Programming level estimation (^L) : 0.0151856
Intellect (I) : 29.4986
Time of programming (T) : 18057.1
Time estimation (^T) : 5562.39
Programming language level (lambda) : 0.0693854
Work on programming (E) : 325027
Error (B) : 1.57577
Error estimation (^B) : 0.647512

Table:

=====

Operators:

1	2	%
2	17	()
3	13	*
4	1	+
5	3	++
6	5	+=
7	10	,
8	7	-
9	10	/
10	44	;
11	3	<
12	27	=
13	6	[]
14	2	_&
15	6	_*
16	3	_[]
17	5	__*
18	24	float
19	3	for
20	5	int
21	2	linfit2
22	1	main
23	2	rand
24	1	return
25	4	sqrt
26	1	srand
27	1	time
28	1	void

Operands:

1	4	0
2	10	0.0
3	2	100
4	1	2
5	6	N
6	1	NULL
7	6	a
8	6	b
9	1	correl_coef
10	15	i
11	8	n
12	2	see
13	1	sigma_a
14	2	sigma_b
15	6	sum_x
16	5	sum_x2
17	5	sum_xy
18	7	sum_y
19	4	sum_y2
20	5	sxx
21	3	sxy
22	2	syy
23	6	x
24	5	xi
25	5	y
26	4	y_calc
27	5	yi

Summary:

=====

The number of different operators : 28
The number of different operands : 27
The total number of operators : 209
The total number of operands : 127

Dictionary (D) : 55
Length (N) : 336
Length estimation (^N) : 262.988
Volume (V) : 1942.54
Potential volume (*V) : 11.6096
Limit volume (**V) : 15.6844
Programming level (L) : 0.00597654
Programming level estimation (^L) : 0.0151856
Intellect (I) : 29.4986
Time of programming (T) : 18057.1
Time estimation (^T) : 5562.39
Programming language level (lambda) : 0.0693854
Work on programming (E) : 325027
Error (B) : 1.57577
Error estimation (^B) : 0.647512

ПРИЛОЖЕНИЕ Д

Код программы на Ассемблер

```
.section      __TEXT,__text,regular,pure_instructions
.build_version macos, 10, 14      sdk_version 10, 14
.globl _linfit2                    ## -- Begin function linfit2
.p2align      4, 0x90
_linfit2:                                ## @linfit2
.cfi_startproc
## %bb.0:
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset %rbp, -16
    movq %rsp, %rbp
    .cfi_def_cfa_register %rbp
    movq %rdi, -8(%rbp)
    movq %rsi, -16(%rbp)
    movq %rdx, -24(%rbp)
    movq %rcx, -32(%rbp)
    movq %r8, -40(%rbp)
    movl %r9d, -44(%rbp)
    xorps %xmm0, %xmm0
    movss %xmm0, -48(%rbp)
    movss %xmm0, -52(%rbp)
    movss %xmm0, -56(%rbp)
    movss %xmm0, -60(%rbp)
    movss %xmm0, -64(%rbp)
    movl $0, -68(%rbp)
LBB0_1:                                ## =>This Inner Loop Header: Depth=1
    cmpl $100, -68(%rbp)
    jge  LBB0_4
## %bb.2:                                ##   in Loop: Header=BB0_1 Depth=1
    movq -8(%rbp), %rax
    movslq -68(%rbp), %rcx
    movss (%rax,%rcx,4), %xmm0          ## xmm0 = mem[0],zero,zero,zero
    movss %xmm0, -72(%rbp)
    movq -16(%rbp), %rax
    movslq -68(%rbp), %rcx
    movss (%rax,%rcx,4), %xmm0          ## xmm0 = mem[0],zero,zero,zero
    movss %xmm0, -76(%rbp)
    movss -72(%rbp), %xmm0              ## xmm0 = mem[0],zero,zero,zero
    addss -48(%rbp), %xmm0
    movss %xmm0, -48(%rbp)
    movss -76(%rbp), %xmm0              ## xmm0 = mem[0],zero,zero,zero
    addss -52(%rbp), %xmm0
    movss %xmm0, -52(%rbp)
    movss -72(%rbp), %xmm0              ## xmm0 = mem[0],zero,zero,zero
    mulss -76(%rbp), %xmm0
    addss -56(%rbp), %xmm0
    movss %xmm0, -56(%rbp)
    movss -72(%rbp), %xmm0              ## xmm0 = mem[0],zero,zero,zero
    mulss -72(%rbp), %xmm0
    addss -60(%rbp), %xmm0
    movss %xmm0, -60(%rbp)
    movss -76(%rbp), %xmm0              ## xmm0 = mem[0],zero,zero,zero
    mulss -76(%rbp), %xmm0
    addss -64(%rbp), %xmm0
    movss %xmm0, -64(%rbp)
## %bb.3:                                ##   in Loop: Header=BB0_1 Depth=1
    movl -68(%rbp), %eax
    addl $1, %eax
```

```

        movl    %eax, -68(%rbp)
        jmp     LBB0_1
LBB0_4:
        movss   -60(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        movss   -48(%rbp), %xmm1      ## xmm1 = mem[0],zero,zero,zero
        mulss   -48(%rbp), %xmm1
        movl    -44(%rbp), %eax
        cvtsi2ssl    %eax, %xmm2
        divss   %xmm2, %xmm1
        subss   %xmm1, %xmm0
        movss   %xmm0, -80(%rbp)
        movss   -56(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        movss   -48(%rbp), %xmm1      ## xmm1 = mem[0],zero,zero,zero
        mulss   -52(%rbp), %xmm1
        movl    -44(%rbp), %eax
        cvtsi2ssl    %eax, %xmm2
        divss   %xmm2, %xmm1
        subss   %xmm1, %xmm0
        movss   %xmm0, -84(%rbp)
        movss   -64(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        movss   -52(%rbp), %xmm1      ## xmm1 = mem[0],zero,zero,zero
        mulss   -52(%rbp), %xmm1
        movl    -44(%rbp), %eax
        cvtsi2ssl    %eax, %xmm2
        divss   %xmm2, %xmm1
        subss   %xmm1, %xmm0
        movss   %xmm0, -88(%rbp)
        movss   -84(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        divss   -80(%rbp), %xmm0
        movq    -40(%rbp), %rcx
        movss   %xmm0, (%rcx)
        movss   -60(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        mulss   -52(%rbp), %xmm0
        movss   -48(%rbp), %xmm1      ## xmm1 = mem[0],zero,zero,zero
        mulss   -56(%rbp), %xmm1
        subss   %xmm1, %xmm0
        movl    -44(%rbp), %eax
        cvtsi2ssl    %eax, %xmm1
        divss   %xmm1, %xmm0
        divss   -80(%rbp), %xmm0
        movq    -32(%rbp), %rcx
        movss   %xmm0, (%rcx)
        movss   -84(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        cvtss2sd    %xmm0, %xmm0      ## xmm1 = mem[0],zero,zero,zero
        movss   -80(%rbp), %xmm1
        mulss   -88(%rbp), %xmm1
        cvtss2sd    %xmm1, %xmm1
        sqrtss    %xmm1, %xmm1
        divsd    %xmm1, %xmm0
        cvtsd2ss    %xmm0, %xmm0
        movss   %xmm0, -92(%rbp)
        movss   -64(%rbp), %xmm0      ## xmm0 = mem[0],zero,zero,zero
        movq    -32(%rbp), %rcx
        movss   (%rcx), %xmm1      ## xmm1 = mem[0],zero,zero,zero
        mulss   -52(%rbp), %xmm1
        subss   %xmm1, %xmm0
        movq    -40(%rbp), %rcx
        movss   (%rcx), %xmm1      ## xmm1 = mem[0],zero,zero,zero
        mulss   -56(%rbp), %xmm1
        subss   %xmm1, %xmm0
        movl    -44(%rbp), %eax
        subl    $2, %eax

```

```

    cvtsi2ssl    %eax, %xmm1
    divss %xmm1, %xmm0
    cvtss2sd     %xmm0, %xmm0
    sqrtss %xmm0, %xmm0
    cvtsd2ss     %xmm0, %xmm0
    movss %xmm0, -96(%rbp)
    movss -96(%rbp), %xmm0    ## xmm0 = mem[0],zero,zero,zero
    cvtss2sd     %xmm0, %xmm0
    movss -80(%rbp), %xmm1    ## xmm1 = mem[0],zero,zero,zero
    cvtss2sd     %xmm1, %xmm1
    sqrtss %xmm1, %xmm1
    divss %xmm1, %xmm0
    cvtsd2ss     %xmm0, %xmm0
    movss %xmm0, -100(%rbp)
    movss -100(%rbp), %xmm0    ## xmm0 = mem[0],zero,zero,zero
    cvtss2sd     %xmm0, %xmm0
    movss -60(%rbp), %xmm1    ## xmm1 = mem[0],zero,zero,zero
    movl  -44(%rbp), %eax
    cvtsi2ssl    %eax, %xmm2
    divss %xmm2, %xmm1
    cvtss2sd     %xmm1, %xmm1
    sqrtss %xmm1, %xmm1
    mulss %xmm1, %xmm0
    cvtsd2ss     %xmm0, %xmm0
    movss %xmm0, -104(%rbp)
    movl  $0, -108(%rbp)
LBB0_5:    ## =>This Inner Loop Header: Depth=1
    movl  -108(%rbp), %eax
    cmpl  -44(%rbp), %eax
    jge   LBB0_8
## %bb.6:    ## in Loop: Header=BB0_5 Depth=1
    movq  -32(%rbp), %rax
    movss (%rax), %xmm0    ## xmm0 = mem[0],zero,zero,zero
    movq  -40(%rbp), %rax
    movss (%rax), %xmm1    ## xmm1 = mem[0],zero,zero,zero
    movq  -8(%rbp), %rax
    movslq -108(%rbp), %rcx
    mulss (%rax,%rcx,4), %xmm1
    addss %xmm1, %xmm0
    movq  -24(%rbp), %rax
    movslq -108(%rbp), %rcx
    movss %xmm0, (%rax,%rcx,4)
## %bb.7:    ## in Loop: Header=BB0_5 Depth=1
    movl  -108(%rbp), %eax
    addl  $1, %eax
    movl  %eax, -108(%rbp)
    jmp   LBB0_5
LBB0_8:
    popq  %rbp
    retq
    .cfi_endproc

    .globl _main
    .p2align 4, 0x90
_main:    ## @main
    .cfi_startproc
## %bb.0:
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset %rbp, -16
    movq  %rsp, %rbp
    .cfi_def_cfa_register %rbp

```

```

    subq    $1264, %rsp                ## imm = 0x4F0
    xorl    %eax, %eax
    movl    %eax, %edi
    xorl    %eax, %eax
    movl    $400, %ecx                ## imm = 0x190
    movl    %ecx, %edx
    movq    __stack_chk_guard@GOTPCREL(%rip), %rsi
    movq    (%rsi), %rsi
    movq    %rsi, -8(%rbp)
    movl    $0, -1220(%rbp)
    leaq    -416(%rbp), %rsi
    movq    %rdi, -1240(%rbp)        ## 8-byte Spill
    movq    %rsi, %rdi
    movl    %eax, %esi
    movq    %rdx, -1248(%rbp)        ## 8-byte Spill
    movl    %eax, -1252(%rbp)        ## 4-byte Spill
    callq   _memset
    leaq    -816(%rbp), %rdx
    movq    %rdx, %rdi
    movl    -1252(%rbp), %esi        ## 4-byte Reload
    movq    -1248(%rbp), %rdx        ## 8-byte Reload
    callq   _memset
    leaq    -1216(%rbp), %rdx
    movq    %rdx, %rdi
    movl    -1252(%rbp), %esi        ## 4-byte Reload
    movq    -1248(%rbp), %rdx        ## 8-byte Reload
    callq   _memset
    xorps   %xmm0, %xmm0
    movss   %xmm0, -1224(%rbp)
    movss   %xmm0, -1228(%rbp)
    movq    -1240(%rbp), %rdi        ## 8-byte Reload
    callq   _time
    movl    %eax, %ecx
    movl    %ecx, %edi
    callq   _srand
    movl    $0, -1232(%rbp)
LBB1_1:                                     ## =>This Inner Loop Header: Depth=1
    cmpl    $100, -1232(%rbp)
    jge     LBB1_4
## %bb.2:                                     ##   in Loop: Header=BB1_1 Depth=1
    callq   _rand
    cltd
    movl    $100, %ecx
    idivl   %ecx
    cvtsi2ssl    %edx, %xmm0
    movslq    -1232(%rbp), %rsi
    movss     %xmm0, -416(%rbp,%rsi,4)
    callq   _rand
    cltd
    movl    $100, %ecx
    idivl   %ecx
    cvtsi2ssl    %edx, %xmm0
    movslq    -1232(%rbp), %rsi
    movss     %xmm0, -816(%rbp,%rsi,4)
## %bb.3:                                     ##   in Loop: Header=BB1_1 Depth=1
    movl    -1232(%rbp), %eax
    addl    $1, %eax
    movl    %eax, -1232(%rbp)
    jmp     LBB1_1
LBB1_4:
    leaq    -1216(%rbp), %rdx
    leaq    -816(%rbp), %rsi

```



```

    leaq    -416(%rbp), %rdi
    leaq    -1224(%rbp), %rcx
    leaq    -1228(%rbp), %r8
    movl    $100, %r9d
    callq   _linfit2
    movq    ___stack_chk_guard@GOTPCREL(%rip), %rcx
    movq    (%rcx), %rcx
    movq    -8(%rbp), %rdx
    cmpq    %rdx, %rcx
    jne     LBB1_6
## %bb.5:
    xorl    %eax, %eax
    addq    $1264, %rsp          ## imm = 0x4F0
    popq    %rbp
    retq
LBB1_6:
    callq   ___stack_chk_fail
    ud2
    .cfi_endproc

                                ## -- End function

.subsections_via_symbols

```