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 _____ Рыбин А.С.

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

1. Определение метрических характеристик для программы на Pascal.

Код программы представлен в приложении А. Измеримые характеристики рассчитанные вручную представлены в таблице 1.

Таблица 1 – Измеримые характеристики программы на Pascal (ручной подсчёт)

№	Оператор	Количество	№	Операнд	Количество
1	program	1	1	x	2
2	procedure	1	2	y	1
3	linfit	1	3	y_calc	1
4	()	2	4	a	2
5	;	27	5	b	1
6	:=	20	6	n	6
7	real	4	7	linear_fit	1
8	integer	2	8	i	6
9	for do	2	9	sum_x	7
10	+	6	10	sum_y	7
11	*	9	11	sum_xy	5
12	/	6	12	sum_x2	5
13	begin	4	13	sum_y2	4
14	end	4	14	xi	5
15	[]	4	15	yi	5
16	.	1	16	sxx	3
Всего		94	17	syy	1
			18	sxy	2
			19	0.0	5
			Всего		70

Измеримые характеристики рассчитанные с помощью программы представлены в таблице 2. Файл с результатами программных расчётов представлен в приложении Б.

Таблица 2 – Измеримые характеристики программы на Pascal (программный расчёт)

№	Оператор	Количество	№	Операнд	Количество
1	()	6	1	0.0	5
2	*	9	2	1	2
3	+	6	3	a	3
4	-	4	4	b	3
5	/	6	5	i	5
6	;	40	6	linear_fit	1
7	=	18	7	n	7
8	[]	4	8	sum_x	8
9	for	2	9	sum_x2	6
10	integer	2	10	sum_xy	6
11	linfit	1	11	sum_y	8
12	procedure	1	12	sum_y2	5
13	program	1	13	sxx	4
14	real	4	14	sxy	3
Всего		105	15	syy	2
			16	x	3
			17	xi	6
			18	y	2
			19	y_calc	2
			20	yi	6
			Всего		88

Определение расчетных характеристик представлено в таблице 3.

Таблица 3 – Расчётные характеристики программы на Pascal

Характеристика	Ручной подсчёт	Программный расчёт
Число простых операторов n_1	16	14
Число простых операндов n_2	19	20
Общее число всех операторов N_1	94	105
Общее число всех операндов N_2	70	88
Словарь n	35	34
Длина $N_{опыт}$	164	193
Теоретическая длина $N_{теор}$	133.86	150.84
Объём V	841.16	997.80
Потенциальный объём V*	19.65	19.65
Уровень программы L	0.020	0.020
Оценка уровня программы L˜	0.034	0.032
Интеллектуальное содержание I	28.54	31.75
Работа программирования E	35993.94	50662.60
Оценка времени программирования T^	3600.76	1361.63
Время программирования T	2479.20	2814.59
Уровень языка λ	0.39	0.39
Ожидаемое число ошибок в программе B	1	0

2. Определение метрических характеристик для программы на Си.

Код программы представлен в приложении В. Измеримые характеристики рассчитанные вручную представлены в таблице 4.

Таблица 4 – Измеримые характеристики программы на Си (ручной подсчёт)

№	Оператор	Количество	№	Операнд	Количество
1	void	1	1	0	3
2	linfit	1	2	0.0	5
3	()	5	3	a	2
4	{}	3	4	b	2
5	float	6	5	i	10
6	int	5	6	n	6
7	;	23	7	sum_x	7
8	=	20	8	sum_x2	5
9	for	2	9	sum_xy	5
10	++	2	10	sum_y	7
11	<	2	11	sum_y2	4
12	[]	4	12	sxx	3
13	+	10	13	sxy	2
14	-	4	14	syy	1
15	/	6	15	x	2
16	*	9	16	xi	5
17	_* разыменование указателя	4	17	y	1
18	__* указатель	5	18	y_calc	1
Всего		114	19	yi	5
			Всего		76

Измеримые характеристики рассчитанные с помощью программы представлены в таблице 5. Файл с результатами программных расчётов представлен в приложении Г.

Таблица 5 – Измеримые характеристики программы на Си (программный расчёт)

№	Оператор	Количество	№	Операнд	Количество
1	()	6	1	0	3
2	*	9	2	0.0	5
3	+	6	3	a	3
4	++	2	4	b	3
5	,	15	5	i	10
6	-	4	6	n	7
7	/	6	7	sum_x	8
8	;	28	8	sum_x2	6
9	<	2	9	sum_xy	6
10	=	20	10	sum_y	8
11	[]	4	11	sum_y2	5
12	_*	4	12	sxx	4
13	_[]	1	13	sxy	3
14	__*	6	14	syy	2
15	char	1	15	x	3
16	const	2	16	xi	6
17	float	6	17	y	2
18	for	2	18	y_calc	2
19	int	5	19	yi	6
20	linfit	1	Всего		92
21	void	1			
Всего		131			

Определение расчетных характеристик представлено в таблице 6.

Таблица 6 – Расчётные характеристики программы на Си

Характеристика	Ручной подсчёт	Программный расчёт
Число простых операторов n_1	18	21
Число простых операндов n_2	19	19
Общее число всех операторов N_1	114	131
Общее число всех операндов N_2	76	92
Словарь n	37	40
Длина $N_{опыт}$	190	223
Теоретическая длина $N_{теор}$	155.77	196.28
Объём V	989.71	1239.29
Потенциальный объём V*	19.65	19.65
Уровень программы L	0.020	0.016
Оценка уровня программы $L\tilde{}$	0.028	0.019
Интеллектуальное содержание I	27.49	24.08
Работа программирования E	49485.50	78154
Оценка времени программирования T^	4984.86	3064.49
Время программирования T	3562.96	4341.89
Уровень языка λ	0.39	0.31
Ожидаемое число ошибок в программе B	1	1

3. Определение метрических характеристик для программы на Ассемблере.

Код программы представлен в приложении Д. Ручной расчёт измеримых характеристик представлен в таблице 7.

Таблица 7 – Измеримые характеристики программы на Ассемблере (ручной подсчёт)

№	Оператор	Количество	№	Операнд	Количество
1	push	1	1	rbp	3
2	mov	23	2	rsp	1
3	pxor	5	3	rdi	1
4	movss	40	4	rsi	1
5	jmp	2	5	rdx	1
6	cdqe	4	6	rcx	1
7	lea	4	7	r8	1
8	add	6	8	r9d	1
9	addss	6	9	xmm0	75
10	mulss	9	10	eax	8
11	jl	2	11	rax	24
12	cvtsi2ss	4	12	xmm1	25
13	divss	6	13	0	6
14	movaps	3	14	xmm2	2
15	subss	4	15	1	2
16	nop	2	16	linfit	1
17	pop	1	17	.L3	2
18	ret	1	18	.L2	2
Всего		123	19	.L4	2
			20	.L5	2
			Всего		161

Определение расчетных характеристик представлено в таблице 8.

Таблица 8 – Расчётные характеристики программы на Ассемблере

Характеристика	Ручной расчёт
Число простых операторов n_1	18
Число простых операндов n_2	20
Общее число всех операторов N_1	123
Общее число всех операндов N_2	161
Словарь n	38
Длина $N_{опыт}$	284
Теоретическая длина $N_{теор}$	161.50
Объём V	1490.43
Потенциальный объём V*	19.65
Уровень программы L	0.013
Оценка уровня программы L˜	0.014
Интеллектуальное содержание I	20.87
Работа программирования E	114648.46
Оценка времени программирования T^	11362.57
Время программирования T	10798.20
Уровень языка λ	0.26
Ожидаемое число ошибок в программе B	2

4. Сравнение результатов определения метрических характеристик.

Таблица 9 – Сводная таблица расчетов для всех языков

	Pascal		**Си**		**Ассемблер**
Характеристика	*Ручной подсчёт*	*Программный расчёт*	*Ручной подсчёт*	*Программный расчёт*	*Ручной подсчёт*
Число простых операторов n_1	16	14	18	21	18
Число простых операндов n_2	19	20	19	19	20
Общее число всех операторов N_1	94	105	114	131	123
Общее число всех операндов N_2	70	88	76	92	161
Словарь n	35	34	37	40	38
Длина $N_{опыт}$	164	193	190	223	284
Теоретическая длина $N_{теор}$	133.86	150.84	155.77	196.28	161.50
Объём V	841.16	997.80	989.71	1239.29	**1490.43**
Потенциальный объём V*	19.65				
Уровень программы	0.020	0.020	0.020	0.016	**0.013**
Оценка уровня программы $L^{\sim}$	0.034	0.032	0.028	0.019	0.014
Интеллектуальное содержание I	28.54	**31.75**	27.49	24.08	20.87
Работа программирования E	35993.94	50662.60	49485.50	78154	114648.46
Оценка времени программирования T^	3600.76	1361.63	4984.86	3064.49	11362.57
Время программирования T	**2479.20**	2814.59	3562.96	4341.89	**10798.20**
Уровень языка λ	0.39	0.39	0.39	0.31	0.26

Ожидаемое число ошибок в программе B	1	**0**	1	1	**2**

В результате сравнения видно, что уровень программы самый низкий у программы на Ассемблере (на порядок меньше чем у Си и Pascal), а самый высокий у программы на Pascal. Наибольшие показатели времени программирования, работы программирования и ожидаемого числа ошибок, наоборот, соответствуют Ассемблеру, а наименьший – Pascal. Показатели для Си практически не отличаются от Pascal за исключением высокого ожидаемого числа ошибок в программе, однако во всех случаях они хуже.

Выводы

В результате выполнения данной лабораторной работы была изучена система метрик Холстеда. Было проведено сравнение программ, реализующих алгоритм линеаризации, на языках Pascal, Си и Ассемблер.

В результате сравнения видно, что уровень программы самый низкий у программы на Ассемблере (на порядок меньше чем у Си и Pascal), а самый высокий у программы на Pascal. Наибольшие показатели времени программирования, работы программирования и ожидаемого числа ошибок, наоборот, соответствуют Ассемблеру, а наименьший – Pascal. Показатели для Си практически не отличаются от Pascal за исключением высокого ожидаемого числа ошибок в программе, однако во всех случаях они хуже.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ НА Pascal.

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

{ fit a straight line (y_calc) through n sets of x and y pairs of points }

var   i : integer;
sum_x, sum_y, sum_xy, sum_x2, sum_y2, xi, yi, sxx, syy, sxy : 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;

  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;

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

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ РАБОТЫ ПАРСЕРА parser_pas.exe

```
Statistics for module linfit_pas_parsed.lxm
====================================
The number of different operators     : 15
The number of different operands      : 21
The total number of operators         : 105
The total number of operands          : 88

Dictionary                   (  D)    : 36
Length                       (  N)    : 193
Length estimation            ( ^N)    : 150.842
Volume                       (  V)    : 997.796
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.0196949
Programming level estimation ( ^L)    : 0.0318182
Intellect                    (  I)    : 31.748
Time of programming          (  T)    : 2814.59
Time estimation              ( ^T)    : 1361.63
Programming language level   (lambda) : 0.387034
Work on programming          (  E)    : 50662.6
Error                        (  B)    : 0.456391
Error estimation             ( ^B)    : 0.332599


Table:
====================================
Operators:
|  1  |  6  | ()
|  2  |  9  | *
|  3  |  6  | +
|  4  |  4  | -
|  5  |  6  | /
|  6  |  40 | ;
|  7  |  18 | =
|  8  |  4  | []
|  9  |  2  | for
|  10 |  2  | integer
|  11 |  1  | linfit
|  12 |  1  | procedure
|  13 |  1  | program
|  14 |  4  | real
|  15 |  1  | writeln
Operands:
|  1  |  1  | 'Hello world!'
|  2  |  5  | 0.0
|  3  |  2  | 1
|  4  |  3  | a
|  5  |  3  | b
|  6  |  5  | i
|  7  |  1  | linear_fit
|  8  |  7  | n
|  9  |  8  | sum_x
|  10 |  6  | sum_x2
|  11 |  6  | sum_xy
|  12 |  8  | sum_y
|  13 |  5  | sum_y2
|  14 |  4  | sxx
```

```
|  15 |  3  | sxy
|  16 |  2  | syy
|  17 |  3  | x
|  18 |  6  | xi
|  19 |  2  | y
|  20 |  2  | y_calc
|  21 |  6  | yi


Summary:
====================================
The number of different operators     : 15
The number of different operands      : 21
The total number of operators         : 105
The total number of operands          : 88

Dictionary                   (  D)    : 36
Length                       (  N)    : 193
Length estimation            ( ^N)    : 150.842
Volume                       (  V)    : 997.796
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.0196949
Programming level estimation ( ^L)    : 0.0318182
Intellect                    (  I)    : 31.748
Time of programming          (  T)    : 2814.59
Time estimation              ( ^T)    : 1361.63
Programming language level   (lambda) : 0.387034
Work on programming          (  E)    : 50662.6
Error                        (  B)    : 0.456391
Error estimation             ( ^B)    : 0.332599
```

ПРИЛОЖЕНИЕ В

КОД ПРОГРАММЫ НА Си

```
void linfit(const float* x, const float* y, float* y_calc, float* a, float* b, int n)
{
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2, xi, yi, sxx, syy, sxy;

    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;

    for (int i = 0; i < n; i++) {
        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;
    }

    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;

    for (int i = 0; i < n; i++) {
        y_calc[i] = *a + *b * x[i];
    }
}
```

ПРИЛОЖЕНИЕ Г

РЕЗУЛЬТАТЫ РАБОТЫ ПАРСЕРА parser_c.exe

```
Statistics for module linfit_c_parsed.lxm
====================================
The number of different operators     : 23
The number of different operands      : 21
The total number of operators         : 133
The total number of operands          : 94

Dictionary                   (  D)    : 44
Length                       (  N)    : 227
Length estimation            ( ^N)    : 196.281
Volume                       (  V)    : 1239.29
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.015857
Programming level estimation ( ^L)    : 0.0194265
Intellect                    (  I)    : 24.075
Time of programming          (  T)    : 4341.89
Time estimation              ( ^T)    : 3064.49
Programming language level   (lambda) : 0.311614
Work on programming          (  E)    : 78154
Error                        (  B)    : 0.609321
Error estimation             ( ^B)    : 0.413097


Table:
====================================
Operators:
|  1  |  6  | ()
|  2  |  9  | *
|  3  |  6  | +
|  4  |  2  | ++
|  5  |  15 | ,
|  6  |  4  | -
|  7  |  6  | /
|  8  |  28 | ;
|  9  |  2  | <
|  10 |  20 | =
|  11 |  4  | []
|  12 |  4  | _*
|  13 |  1  | _[]
|  14 |  6  | __*
|  15 |  1  | char
|  16 |  2  | const
|  17 |  6  | float
|  18 |  2  | for
|  19 |  5  | int
|  20 |  1  | linfit
|  21 |  1  | main
|  22 |  1  | return
|  23 |  1  | void
Operands:
|  1  |  3  | 0
|  2  |  5  | 0.0
|  3  |  3  | a
|  4  |  1  | argc
|  5  |  1  | argv
|  6  |  3  | b
```

```
|  7  |  10 | i
|  8  |  7  | n
|  9  |  8  | sum_x
|  10 |  6  | sum_x2
|  11 |  6  | sum_xy
|  12 |  8  | sum_y
|  13 |  5  | sum_y2
|  14 |  4  | sxx
|  15 |  3  | sxy
|  16 |  2  | syy
|  17 |  3  | x
|  18 |  6  | xi
|  19 |  2  | y
|  20 |  2  | y_calc
|  21 |  6  | yi


Summary:
====================================
The number of different operators     : 23
The number of different operands      : 21
The total number of operators         : 133
The total number of operands          : 94

Dictionary                   (  D)    : 44
Length                       (  N)    : 227
Length estimation            ( ^N)    : 196.281
Volume                       (  V)    : 1239.29
Potential volume             ( *V)    : 19.6515
Limit volume                 (**V)    : 38.2071
Programming level            (  L)    : 0.015857
Programming level estimation ( ^L)    : 0.0194265
Intellect                    (  I)    : 24.075
Time of programming          (  T)    : 4341.89
Time estimation              ( ^T)    : 3064.49
Programming language level   (lambda) : 0.311614
Work on programming          (  E)    : 78154
Error                        (  B)    : 0.609321
Error estimation             ( ^B)    : 0.413097
```

ПРИЛОЖЕНИЕ Д

КОД ПРОГРАММЫ НА Ассемблер

```
linfit:
        push    rbp
        mov     rbp, rsp
        mov     QWORD PTR [rbp-56], rdi
        mov     QWORD PTR [rbp-64], rsi
        mov     QWORD PTR [rbp-72], rdx
        mov     QWORD PTR [rbp-80], rcx
        mov     QWORD PTR [rbp-88], r8
        mov     DWORD PTR [rbp-92], r9d
        pxor    xmm0, xmm0
        movss   DWORD PTR [rbp-4], xmm0
        pxor    xmm0, xmm0
        movss   DWORD PTR [rbp-8], xmm0
        pxor    xmm0, xmm0
        movss   DWORD PTR [rbp-12], xmm0
        pxor    xmm0, xmm0
        movss   DWORD PTR [rbp-16], xmm0
        pxor    xmm0, xmm0
        movss   DWORD PTR [rbp-20], xmm0
        mov     DWORD PTR [rbp-24], 0
        jmp     .L2
.L3:
        mov     eax, DWORD PTR [rbp-24]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-56]
        add     rax, rdx
        movss   xmm0, DWORD PTR [rax]
        movss   DWORD PTR [rbp-44], xmm0
        mov     eax, DWORD PTR [rbp-24]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-64]
        add     rax, rdx
        movss   xmm0, DWORD PTR [rax]
        movss   DWORD PTR [rbp-48], xmm0
        movss   xmm0, DWORD PTR [rbp-4]
        addss   xmm0, DWORD PTR [rbp-44]
        movss   DWORD PTR [rbp-4], xmm0
        movss   xmm0, DWORD PTR [rbp-8]
        addss   xmm0, DWORD PTR [rbp-48]
        movss   DWORD PTR [rbp-8], xmm0
        movss   xmm0, DWORD PTR [rbp-44]
        mulss   xmm0, DWORD PTR [rbp-48]
        movss   xmm1, DWORD PTR [rbp-12]
        addss   xmm0, xmm1
        movss   DWORD PTR [rbp-12], xmm0
        movss   xmm0, DWORD PTR [rbp-44]
        mulss   xmm0, xmm0
        movss   xmm1, DWORD PTR [rbp-16]
        addss   xmm0, xmm1
        movss   DWORD PTR [rbp-16], xmm0
        movss   xmm0, DWORD PTR [rbp-48]
        mulss   xmm0, xmm0
        movss   xmm1, DWORD PTR [rbp-20]
        addss   xmm0, xmm1
        movss   DWORD PTR [rbp-20], xmm0
```

```
        add     DWORD PTR [rbp-24], 1
.L2:
        mov     eax, DWORD PTR [rbp-24]
        cmp     eax, DWORD PTR [rbp-92]
        jl      .L3
        movss   xmm0, DWORD PTR [rbp-4]
        mulss   xmm0, xmm0
        cvtsi2ss        xmm1, DWORD PTR [rbp-92]
        divss   xmm0, xmm1
        movaps  xmm1, xmm0
        movss   xmm0, DWORD PTR [rbp-16]
        subss   xmm0, xmm1
        movss   DWORD PTR [rbp-32], xmm0
        movss   xmm0, DWORD PTR [rbp-4]
        mulss   xmm0, DWORD PTR [rbp-8]
        cvtsi2ss        xmm1, DWORD PTR [rbp-92]
        divss   xmm0, xmm1
        movaps  xmm1, xmm0
        movss   xmm0, DWORD PTR [rbp-12]
        subss   xmm0, xmm1
        movss   DWORD PTR [rbp-36], xmm0
        movss   xmm0, DWORD PTR [rbp-8]
        mulss   xmm0, xmm0
        cvtsi2ss        xmm1, DWORD PTR [rbp-92]
        divss   xmm0, xmm1
        movaps  xmm1, xmm0
        movss   xmm0, DWORD PTR [rbp-20]
        subss   xmm0, xmm1
        movss   DWORD PTR [rbp-40], xmm0
        movss   xmm0, DWORD PTR [rbp-36]
        divss   xmm0, DWORD PTR [rbp-32]
        mov     rax, QWORD PTR [rbp-88]
        movss   DWORD PTR [rax], xmm0
        movss   xmm0, DWORD PTR [rbp-16]
        mulss   xmm0, DWORD PTR [rbp-8]
        movss   xmm1, DWORD PTR [rbp-4]
        mulss   xmm1, DWORD PTR [rbp-12]
        subss   xmm0, xmm1
        cvtsi2ss        xmm1, DWORD PTR [rbp-92]
        divss   xmm0, xmm1
        divss   xmm0, DWORD PTR [rbp-32]
        mov     rax, QWORD PTR [rbp-80]
        movss   DWORD PTR [rax], xmm0
        mov     DWORD PTR [rbp-28], 0
        jmp     .L4
.L5:
        mov     rax, QWORD PTR [rbp-80]
        movss   xmm1, DWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-88]
        movss   xmm2, DWORD PTR [rax]
        mov     eax, DWORD PTR [rbp-28]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-56]
        add     rax, rdx
        movss   xmm0, DWORD PTR [rax]
        mulss   xmm0, xmm2
        mov     eax, DWORD PTR [rbp-28]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-72]
```

```
        add     rax, rdx
        addss   xmm0, xmm1
        movss   DWORD PTR [rax], xmm0
        add     DWORD PTR [rbp-28], 1
.L4:
        mov     eax, DWORD PTR [rbp-28]
        cmp     eax, DWORD PTR [rbp-92]
        jl      .L5
        nop
        nop
        pop     rbp
        ret
```