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 _____ Чешуин Д. И.

Преподаватель _____ Кирьянчиков 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, полученная от преподавателя, была изменена, из нее были убраны строчки кода, отвечающие за вывод информации, а также лишние переменные. Программа на языках Паскаль, С и Assembler представлена в приложениях А, Б и В, соответственно.

Был выполнен ручной подсчёт операторов и операндов. В таблицах 1-3 представлены результаты ручного подсчета числа типов операторов и операндов в программах на языках Паскаль, С и Assembler.

Таблица 1 – Количество операторов и операндов в программе на языке Паскаль.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	()	16	1	0,0	10
2	*	17	2	0,27	1
3	+	12	3	1	28
4	-	9	4	1.0	1
5	/	3	5	12,98	1
6	;	59	6	14,42	1
7	:=	57	7	15,8	1
8	>	1	8	17,96	1
9	[]	54	9	18,92	1
10	setup	3	10	2	21
11	solve	1	11	2,07	1
12	sqr	3	12	y_calc	3
13	linfit	1	13	3	21
14	deter	2	14	4	1
15	exit	1	15	5	1
16	for	6	16	6	1
17	sqrt	1	17	6,45	1
18	get_data	1	18	7	1
19	if	2	19	8	1
20	begin…end	10	20	8,6	1
			21	9	1
			22	a	28
			23	yi	6
			24	y	13
			25	xi	6
			26	b	8
			27	coef	9
			28	correl_coef	1
			29	det	3
			30	deter	1
			31	g	4
			32	i	20
			33	j	8
			34	x2	6
			35	maxc	3
			36	x	6
			37	nrow	5
			38	srs	4
			39	sum_2y	4
			40	sum_x	5
			41	sum_x2	6

Таблица 1 (продолжение) – Количество операторов и операндов в программе на языке Паскаль.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			42	sum_x3	5
			43	sum_x4	4
			44	sum_xy	4
			45	sum_y	5
			46	sum_y2	4

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	()	26	1	0	27
2	*	17	2	0,0	9
3	+	12	3	0,27	1
4	++	6	4	1	24
5	+=	1	5	1.0	1
6	sqrt	1	6	12,98	1
7	-	9	7	14,42	1
8	/	3	8	15,8	1
9	;	72	9	17,96	1
10	<	6	10	18,92	1
11	=	55	11	2	24
12	==	1	12	2,07	1
13	>	1	13	3	1
14	[]	83	14	4	1
15	_&	1	15	5	1
16	_*	1	16	6	1
17	return	3	17	6,45	1
18	pow	3	18	7	1
19	deter	2	19	8	1
20	double	1	20	8,6	1
21	for	6	21	MAXC	3
22	get_data	1	22	MAXR	5
23	if	2	23	a	30
24	solve	1	24	b	8
25	linfit	1	25	coef	9
26	setup	3	26	corel_coef	2
			27	det	6
			28	g	4
			29	i	30
			30	j	10
			31	srs	3
			32	sum_2y	4

Таблица 2 (продолжение) – Количество операторов и операндов в программе на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			33	sum_x	5
			34	sum_x2	6
			35	sum_x3	5
			36	sum_x4	4
			37	sum_xy	4
			38	sum_y	5
			39	sum_y2	4
			40	x	6
			41	x2	6
			42	xi	6
			43	y	17
			44	y_calc	3
			45	yi	6

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	push	6	1	rbp	14
2	mov	132	2	rsp	10
3	jmp .L2	1	3	[rbp-24]	8
4	pxor	12	4	rdi	21
5	cvtsi2sd	1	5	[rbp-32]	20
6	cdqe	11	6	rsi	10
7	lea	27	7	[rbp-4]	17
8	add	52	8	0	12
9	movsd	121	9	xmm1	39
10	addsd	13	10	eax	26
11	cmp	7	11	rdx	53
12	jle .L3	1	12	[0+rax*8]	9
13	nop	4	13	rax	138
14	pop	2	14	add	52
15	ret	6	15	xmm0	176
16	mulsd	17	16	.LC0[rip]	2
17	subsd	7	17	[rax]	25
18	movapd	6	18	1	7
19	movq	17	19	8	5
20	sub	3	20	.LC1[rip]	1
21	jmp .L7	1	21	.LC2[rip]	1
22	movsx	9	22	16	2

Таблица 3 (продолжение) – Количество операторов и операндов в программе на языке Ассемблер.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
23	sal	4	23	.LC3[rip]	1
24	jle .L8	1	24	24	7
25	jle .L9	1	25	.LC4[rip]	1
26	call deter	2	26	32	1
27	divsd	3	27	.LC5[rip]	1
28	jmp .L11	1	28	40	1
29	jmp .L12	1	29	.LC6[rip]	1
30	jle .L13	1	30	48	7
31	jle .L14	1	31	.LC7[rip]	1
32	ucomisd	2	32	56	1
33	jp .L15	1	33	.LC8[rip]	1
34	je .L18	1	34	64	2
35	call setup	3	35	.LC9[rip]	1
36	jmp .L10	1	36	[rbp-8]	26
37	jmp .L20	1	37	xmm2	15
38	jle .L21	1	38	[rax+8]	5
39	call solve	1	39	[rax+16]	5
40	jmp .L22	1	40	xmm3	14
41	call linfit	1	41	xmm4	4
42	call pow	3	42	[rbp-40]	6
43	jle .L23	1	43	[rbp-48]	7
44	call sqrt	1	44	rcx	14
45	call get_data	1	45	[rbp-52]	6
46	leave	4	46	r8d	4
			47	[rbp-64]	6
			48	[rdx+rax]	2
			49	3	4
			50	[rcx]	1
			51	[rdx+rax*8]	3
			52	edi	2
			53	[rax-1]	2
			54	esi	2
			55	[rcx+rax*8]	1
			56	2	4
			57	edx	2
			58	[0+rdx*8]	1
			59	[rdx]	2
			60	-128	1
			61	[rbp-104]	10
			62	[rbp-112]	5
			63	[rbp-120]	5

Таблица 3 (продолжение) – Количество операторов и операндов в программе на языке Ассемблер.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			64	[rbp-96+rax*8]	1
			65	[rbp-96]	9
			66	[rbp-16]	11
			67	272	1
			68	[rbp-216]	4
			69	[rbp-224]	3
			70	[rbp-232]	3
			71	[rbp-240]	6
			72	[rbp-248]	2
			73	r8	2
			74	[rbp-56]	4
			75	[rbp-76]	13
			76	[rbp-88]	5
			77	.LC11[rip]	2
			78	[rbp-176]	2
			79	[rbp-152]	1
			80	[rbp-168]	1
			81	[rbp-128]	1
			82	[rbp-160]	3
			83	[rbp-144]	1
			84	[rbp-136]	1
			85	[rbp-208]	2
			86	[rbp-200]	1
			87	[rbp-192]	1
			88	[rbp-72]	4
			89	[rbp-256]	2
			90	[rbp-264]	2
			91	.LC12[rip]	3
			92	288	1
			93	[rbp-80]	2
			94	[rbp-280]	1
			95	[rbp-272]	1

Расчет метрик с помощью программы автоматизации

Код программы был обработан с помощью утилит parcer и metrics. В таблицах 4-5 представлены результаты программного подсчета числа типов операторов и операндов в программах на языках Паскаль и С.

Таблица 4 – Количество операторов и операндов в программе на языке Паскаль.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	()	20	1	0,0	10
2	*	17	2	0,27	1
3	+	12	3	1	31
4	-	9	4	1.0	1
5	/	3	5	12,98	1
6	;	147	6	14,42	1
7	=	58	7	15,8	1
8	>	1	8	17,96	1
9	[]	54	9	18,92	1
10	ary	4	10	2	21
11	ary2s	5	11	2,07	1
12	arys	6	12	20	1
13	const	1	13	3	22
14	deter	3	14	4	1
15	exit	1	15	5	1
16	for	6	16	6	1
17	function	1	17	6,45	1
18	get_data	2	18	7	1
19	if	2	19	8	1
20	integer	5	20	8,6	1
21	linfit	2	21	9	2
22	procedure	4	22	a	30
23	program	1	23	ary	1
24	real	7	24	ary2s	1
25	setup	4	25	arys	1
26	solve	2	26	b	10
27	sqr	3	27	coef	13
28	sqrt	1	28	correl_coef	2
29	type	1	29	det	4
			30	deter	1
			31	g	5
			32	i	19
			33	j	9
			34	least1	1
			35	maxc	7
			36	maxr	2
			37	nrow	6
			38	srs	5
			39	sum_2y	5
			40	sum_x	6
			41	sum_x2	7
			42	sum_x3	6

Таблица 4 (продолжение) – Количество операторов и операндов в программе на языке Паскаль.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			43	sum_x4	5
			44	sum_xy	5
			45	sum_y	6
			46	sum_y2	5
			47	x	9
			48	x2	7
			49	xi	7
			50	y	18
			51	y_calc	5
			52	yi	7

Таблица 5 – Количество операторов и операндов в программе на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	()	21	1	0	27
2	*	17	2	0,0	9
3	+	12	3	0,27	1
4	++	6	4	1	24
5	+=	1	5	1.0	1
6	,	48	6	12,98	1
7	-	9	7	14,42	1
8	/	3	8	15,8	1
9	;	98	9	17,96	1
10	<	6	10	18,92	1
11	=	55	11	2	24
12	==	1	12	2,07	1
13	>	1	13	3	1
14	[]	83	14	4	1
15	_&	1	15	5	1
16	_*	1	16	6	1
17	_[]	27	17	6,45	1
18	__*	1	18	7	1
19	deter	3	19	8	1
20	double	29	20	8,6	1
21	for	6	21	MAXC	22
22	get_data	2	22	MAXR	13
23	if	2	23	a	34
24	int	6	24	b	10
25	linfit	2	25	coef	13
26	main	1	26	corel_coef	4
27	pow	3	27	det	8
28	return	3	28	g	5

Таблица 5 (продолжение) – Количество операторов и операндов в программе на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
29	setup	4	29	i	34
30	solve	2	30	j	12
31	sqrt	1	31	srs	4
32	void	4	32	sum_2y	5
			33	sum_x	6
			34	sum_x2	7
			35	sum_x3	6
			36	sum_x4	5
			37	sum_xy	5
			38	sum_y	6
			39	sum_y2	5
			40	x	9
			41	x2	7
			42	xi	7
			43	y	22
			44	y_calc	5
			45	yi	7

С помощью полученных данных были произведены расчеты характеристик программ. В таблице 6 представлены сводные результаты расчетных характеристик.

Таблица 6 – Сводная таблица.

	Паскаль вручную	Паскаль программно	Си вручную	Си программно	Ассемблер
Число уникальных операторов (n1):	20	29	26	32	46
Число уникальных операндов (n2):	46	52	45	45	95
Общее число операторов (N1):	259	382	318	459	494
Общее число операндов (N2):	267	317	291	361	929
Словарь (n):	66	81	71	77	141
Экспериментальная длина программы (Nэ):	526	699	609	820	1423
Теоретическая длина программы (Nт):	340,5	437,3	369,3	407,1	878,2
Объём программы (V):	3179,35	4431,56	3745,19	5138,76	10159,5
Потенциальный объём (V*):	19,651	19,651	19,651	19,651	19,651
Уровень программы (L):	0,006	0,00443	0,00524	0,00382	0,00193
Интеллект программы (I):	54,775	50,134	44,55	50,035	45,17
Работа по программированию (E):	514377	999348	713762	1343760	5252381
Время кодирования (T):	51437,7	55519,4	71376,2	74653,4	525238,1
Уровень языка программирования (Lam):	0,12146	0,08714	0,10311	0,07515	0,03801
Уровень ошибок (B):	4	4	4	5	11

Выводы.

В ходе выполнения данной работы были на практике изучены методы расчета метрических характеристик качества разработки программ на основе метрик Холстеда. 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 least1;

const    maxr = 20;
    maxc = 3;
    nrow = 9;

type     ary  = array[1..maxr] of real;
    arys = array[1..maxc] of real;
    ary2s     = array[1..maxc,1..maxc] of real;

var x,y,y_calc     : ary;
    coef      : arys;
    correl_coef    : real;

procedure get_data(var x,y: ary);
var i    : integer;
begin
  for i:=1 to nrow do x[i]:=i;
  y[1]:=2.07; y[2]:=8.6;
  y[3]:=14.42;     y[4]:=15.8;
  y[5]:=18.92;     y[6]:=17.96;
  y[7]:=12.98;     y[8]:=6.45;
  y[9]:=0.27;
end;

procedure solve(a: ary2s; y: arys;
     var coef: arys);
var b    : ary2s;
    i,j  : integer;
    det  : real;

function deter(a: ary2s): real;
begin
  deter:= a[1,1] * (a[2,2] * a[3,3] - a[3,2] * a[2,3])
    - a[1,2] * (a[2,1] * a[3,3] - a[3,1] * a[2,3])
    + a[1,3] * (a[2,1] * a[3,2] - a[3,1] * a[2,2]);
end;

procedure setup(var b   : ary2s;
         var coef: arys;
         j    : integer);
var i    : integer;
begin
  for i:=1 to maxc do
    begin
      b[i,j]:=y[i];
      if j>1 then b[i,j-1]:=a[i,j-1]
    end;
```

```
  coef[j]:=deter(b)/det
end;

begin
  for i:=1 to maxc do
    for j:=1 to maxc do
      b[i,j]:=a[i,j];

  det:=deter(b);
  if det=0.0 then
      exit
  else
    begin
      setup(b,coef,1);
      setup(b,coef,2);
      setup(b,coef,3)
    end
end;

procedure linfit(x,y: ary;
          var y_calc: ary;
          var coef:   arys);
var a         : ary2s;
    g         : arys;
    i         : integer;
    sum_x,sum_y,sum_xy,sum_x2,
    sum_y2,xi,yi,sum_x3,sum_x4,sum_2y,
    srs,x2    : real;
begin
  sum_x:=0.0;
  sum_y:=0.0;
  sum_xy:=0.0;
  sum_x2:=0.0;
  sum_y2:=0.0;
  sum_x3:=0.0;
  sum_x4:=0.0;
  sum_2y:=0.0;
  for i:=1 to nrow do
    begin
      xi:=x[i];
      yi:=y[i];
      x2:=xi*xi;
      sum_x:=sum_x+xi;
      sum_y:=sum_y+yi;
      sum_xy:=sum_xy+xi*yi;
      sum_x2:=sum_x2+x2;
      sum_y2:=sum_y2+yi*yi;
      sum_x3:=sum_x3+xi*x2;
      sum_x4:=sum_x4+x2*x2;
      sum_2y:=sum_2y+x2*yi
    end;
  a[1,1]:=nrow;
  a[2,1]:=sum_x;   a[1,2]:=sum_x;
```

```
  a[3,1]:=sum_x2;  a[1,3]:=sum_x2;
  a[2,2]:=sum_x2;  a[3,2]:=sum_x3;
  a[2,3]:=sum_x3;  a[3,3]:=sum_x4;
  g[1]:=sum_y;
  g[2]:=sum_xy;
  g[3]:=sum_2y;
  solve(a,g,coef);
  srs:=0.0;
  for i:=1 to nrow do
    begin
      y_calc[i]:=coef[1]+coef[2]*x[i]+coef[3]*sqr(x[i]);
      srs:=srs+sqr(y[i]-y_calc[i])
    end;
  correl_coef:=sqrt(1.0-srs/(sum_y2-sqr(sum_y)/nrow))
end;


begin
  get_data(x,y);
  linfit(x,y,y_calc,coef);
end.
```

ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

#define MAXR 9
#define MAXC 3

void get_data(double x[MAXR], double y[MAXR]) {
    int i;
    for (i = 0; i < MAXR; i++) {
         x[i] = (double)i + 1;
    }
    y[0] = 2.07;
    y[1] = 8.6;
    y[2] = 14.42;
    y[3] = 15.8;
    y[4] = 18.92;
    y[5] = 17.96;
    y[6] = 12.98;
    y[7] = 6.45;
    y[8] = 0.27;
}

double deter(double a[MAXC][MAXC]) {
    return(a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
         - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
         + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]));
}

void setup(double a[MAXC][MAXC], double b[MAXC][MAXC], double
y[MAXC], double coef[MAXC], int j, double det) {
    int i;
    for (i = 0; i < MAXC; i++) {
         b[i][j] = y[i];
         if (j > 0) {
              b[i][j - 1] = a[i][j - 1];
         }
    }
    coef[j] = deter(b) / det;
}

void solve(double a[MAXC][MAXC], double y[MAXC], double
coef[MAXC]) {
    double b[MAXC][MAXC];
    int i, j;
    double det;

    for (i = 0; i < MAXC; i++) {
         for (j = 0; j < MAXC; j++) {
              b[i][j] = a[i][j];
         }
```

```
        }

        det = deter(b);
        if (det == 0) {
             return;
        }
        else {
             setup(a, b, y, coef, 0, det);
             setup(a, b, y, coef, 1, det);
             setup(a, b, y, coef, 2, det);
        }
   }

   void linfit(double x[MAXR], double y[MAXR], double y_calc[MAXR],
double coef[MAXC], double* corel_coef) {
        double sum_x, sum_y, sum_xy, sum_x2, sum_y2;
        double xi, yi, x2, sum_x3, sum_x4, sum_2y, srs;
        int i;
        double a[MAXC][MAXC];
        double g[MAXC];

        sum_x = 0.0;
        sum_y = 0.0;
        sum_xy = 0.0;
        sum_x2 = 0.0;
        sum_y2 = 0.0;
        sum_x3 = 0.0;
        sum_x4 = 0.0;
        sum_2y = 0.0;

        for (i = 0; i < MAXR; i++) {
             xi = x[i];
             yi = y[i];
             x2 = xi * xi;
             sum_x = sum_x + xi;
             sum_y = sum_y + yi;
             sum_xy = sum_xy + xi * yi;
             sum_x2 = sum_x2 + x2;
             sum_y2 = sum_y2 + yi * yi;
             sum_x3 = sum_x3 + xi * x2;
             sum_x4 = sum_x4 + x2 * x2;
             sum_2y = sum_2y + x2 * yi;
        }

        a[0][0] = MAXR;
        a[1][0] = sum_x;
        a[0][1] = sum_x;
        a[2][0] = sum_x2;
        a[0][2] = sum_x2;
        a[1][1] = sum_x2;
        a[2][1] = sum_x3;
        a[1][2] = sum_x3;
        a[2][2] = sum_x4;
```

```
        g[0] = sum_y;
        g[1] = sum_xy;
        g[2] = sum_2y;

        solve(a, g, coef);
        srs = 0.0;

        for (i = 0; i < MAXR; i++) {
             y_calc[i] = coef[0] + coef[1] * x[i] + coef[2] * pow(x[i],
2);
             srs += pow(y[i] - y_calc[i], 2);
        }
        *corel_coef = sqrt(1.0 - srs / (sum_y2 - pow(sum_y, 2) /
MAXR));
    }

    int main() {
        double x[MAXR];
        double y[MAXR];
        double y_calc[MAXR];
        double coef[MAXC];
        double corel_coef;

        get_data(x, y);
        linfit(x, y, y_calc, coef, &corel_coef);

        return 0;
    }
```

ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
get_data:
        push    rbp
        mov     rbp, rsp
        mov     QWORD PTR [rbp-24], rdi
        mov     QWORD PTR [rbp-32], rsi
        mov     DWORD PTR [rbp-4], 0
        jmp     .L2
.L3:
        pxor    xmm1, xmm1
        cvtsi2sd        xmm1, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-4]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-24]
        add     rax, rdx
        movsd   xmm0, QWORD PTR .LC0[rip]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rax], xmm0
        add     DWORD PTR [rbp-4], 1
.L2:
        cmp     DWORD PTR [rbp-4], 8
        jle     .L3
        mov     rax, QWORD PTR [rbp-32]
        movsd   xmm0, QWORD PTR .LC1[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 8
        movsd   xmm0, QWORD PTR .LC2[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 16
        movsd   xmm0, QWORD PTR .LC3[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 24
        movsd   xmm0, QWORD PTR .LC4[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 32
        movsd   xmm0, QWORD PTR .LC5[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 40
        movsd   xmm0, QWORD PTR .LC6[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 48
        movsd   xmm0, QWORD PTR .LC7[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 56
```

```
        movsd   xmm0, QWORD PTR .LC8[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        add     rax, 64
        movsd   xmm0, QWORD PTR .LC9[rip]
        movsd   QWORD PTR [rax], xmm0
        nop
        pop     rbp
        ret
deter:
        push    rbp
        mov     rbp, rsp
        mov     QWORD PTR [rbp-8], rdi
        mov     rax, QWORD PTR [rbp-8]
        movsd   xmm1, QWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm2, QWORD PTR [rax+8]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm0, QWORD PTR [rax+16]
        mulsd   xmm0, xmm2
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm3, QWORD PTR [rax+8]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm2, QWORD PTR [rax+16]
        mulsd   xmm2, xmm3
        subsd   xmm0, xmm2
        mulsd   xmm0, xmm1
        mov     rax, QWORD PTR [rbp-8]
        movsd   xmm2, QWORD PTR [rax+8]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm3, QWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm1, QWORD PTR [rax+16]
        mulsd   xmm1, xmm3
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm4, QWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm3, QWORD PTR [rax+16]
        mulsd   xmm3, xmm4
        subsd   xmm1, xmm3
        mulsd   xmm2, xmm1
        movapd  xmm1, xmm0
        subsd   xmm1, xmm2
        mov     rax, QWORD PTR [rbp-8]
        movsd   xmm2, QWORD PTR [rax+16]
```

```
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm3, QWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm0, QWORD PTR [rax+8]
        mulsd   xmm0, xmm3
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm4, QWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm3, QWORD PTR [rax+8]
        mulsd   xmm3, xmm4
        subsd   xmm0, xmm3
        mulsd   xmm0, xmm2
        addsd   xmm0, xmm1
        movq    rax, xmm0
        movq    xmm0, rax
        pop     rbp
        ret
setup:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 64
        mov     QWORD PTR [rbp-24], rdi
        mov     QWORD PTR [rbp-32], rsi
        mov     QWORD PTR [rbp-40], rdx
        mov     QWORD PTR [rbp-48], rcx
        mov     DWORD PTR [rbp-52], r8d
        movsd   QWORD PTR [rbp-64], xmm0
        mov     DWORD PTR [rbp-4], 0
        jmp     .L7
.L9:
        mov     eax, DWORD PTR [rbp-4]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-40]
        lea     rcx, [rdx+rax]
        mov     eax, DWORD PTR [rbp-4]
        movsx   rdx, eax
        mov     rax, rdx
        add     rax, rax
        add     rax, rdx
        sal     rax, 3
        mov     rdx, rax
        mov     rax, QWORD PTR [rbp-32]
        add     rdx, rax
        movsd   xmm0, QWORD PTR [rcx]
        mov     eax, DWORD PTR [rbp-52]
        cdqe
        movsd   QWORD PTR [rdx+rax*8], xmm0
        cmp     DWORD PTR [rbp-52], 0
```

```
        jle     .L8
        mov     eax, DWORD PTR [rbp-4]
        movsx   rdx, eax
        mov     rax, rdx
        add     rax, rax
        add     rax, rdx
        sal     rax, 3
        mov     rdx, rax
        mov     rax, QWORD PTR [rbp-24]
        lea     rcx, [rdx+rax]
        mov     eax, DWORD PTR [rbp-52]
        lea     edi, [rax-1]
        mov     eax, DWORD PTR [rbp-4]
        movsx   rdx, eax
        mov     rax, rdx
        add     rax, rax
        add     rax, rdx
        sal     rax, 3
        mov     rdx, rax
        mov     rax, QWORD PTR [rbp-32]
        add     rdx, rax
        mov     eax, DWORD PTR [rbp-52]
        lea     esi, [rax-1]
        movsx   rax, edi
        movsd   xmm0, QWORD PTR [rcx+rax*8]
        movsx   rax, esi
        movsd   QWORD PTR [rdx+rax*8], xmm0
.L8:
        add     DWORD PTR [rbp-4], 1
.L7:
        cmp     DWORD PTR [rbp-4], 2
        jle     .L9
        mov     rax, QWORD PTR [rbp-32]
        mov     rdi, rax
        call    deter
        movq    rax, xmm0
        mov     edx, DWORD PTR [rbp-52]
        movsx   rdx, edx
        lea     rcx, [0+rdx*8]
        mov     rdx, QWORD PTR [rbp-48]
        add     rdx, rcx
        movq    xmm0, rax
        divsd   xmm0, QWORD PTR [rbp-64]
        movsd   QWORD PTR [rdx], xmm0
        nop
        leave
        ret
solve:
        push    rbp
        mov     rbp, rsp
        add     rsp, -128
        mov     QWORD PTR [rbp-104], rdi
        mov     QWORD PTR [rbp-112], rsi
```

```
        mov     QWORD PTR [rbp-120], rdx
        mov     DWORD PTR [rbp-4], 0
        jmp     .L11
.L14:
        mov     DWORD PTR [rbp-8], 0
        jmp     .L12
.L13:
        mov     eax, DWORD PTR [rbp-4]
        movsx   rdx, eax
        mov     rax, rdx
        add     rax, rax
        add     rax, rdx
        sal     rax, 3
        mov     rdx, rax
        mov     rax, QWORD PTR [rbp-104]
        add     rdx, rax
        mov     eax, DWORD PTR [rbp-8]
        cdqe
        movsd   xmm0, QWORD PTR [rdx+rax*8]
        mov     eax, DWORD PTR [rbp-8]
        movsx   rcx, eax
        mov     eax, DWORD PTR [rbp-4]
        movsx   rdx, eax
        mov     rax, rdx
        add     rax, rax
        add     rax, rdx
        add     rax, rcx
        movsd   QWORD PTR [rbp-96+rax*8], xmm0
        add     DWORD PTR [rbp-8], 1
.L12:
        cmp     DWORD PTR [rbp-8], 2
        jle     .L13
        add     DWORD PTR [rbp-4], 1
.L11:
        cmp     DWORD PTR [rbp-4], 2
        jle     .L14
        lea     rax, [rbp-96]
        mov     rdi, rax
        call    deter
        movq    rax, xmm0
        mov     QWORD PTR [rbp-16], rax
        pxor    xmm0, xmm0
        ucomisd xmm0, QWORD PTR [rbp-16]
        jp      .L15
        pxor    xmm0, xmm0
        ucomisd xmm0, QWORD PTR [rbp-16]
        je      .L18
.L15:
        mov     rdi, QWORD PTR [rbp-16]
        mov     rcx, QWORD PTR [rbp-120]
        mov     rdx, QWORD PTR [rbp-112]
        lea     rsi, [rbp-96]
        mov     rax, QWORD PTR [rbp-104]
```

```
        movq    xmm0, rdi
        mov     r8d, 0
        mov     rdi, rax
        call    setup
        mov     rdi, QWORD PTR [rbp-16]
        mov     rcx, QWORD PTR [rbp-120]
        mov     rdx, QWORD PTR [rbp-112]
        lea     rsi, [rbp-96]
        mov     rax, QWORD PTR [rbp-104]
        movq    xmm0, rdi
        mov     r8d, 1
        mov     rdi, rax
        call    setup
        mov     rdi, QWORD PTR [rbp-16]
        mov     rcx, QWORD PTR [rbp-120]
        mov     rdx, QWORD PTR [rbp-112]
        lea     rsi, [rbp-96]
        mov     rax, QWORD PTR [rbp-104]
        movq    xmm0, rdi
        mov     r8d, 2
        mov     rdi, rax
        call    setup
        jmp     .L10
.L18:
        nop
.L10:
        leave
        ret
linfit:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 272
        mov     QWORD PTR [rbp-216], rdi
        mov     QWORD PTR [rbp-224], rsi
        mov     QWORD PTR [rbp-232], rdx
        mov     QWORD PTR [rbp-240], rcx
        mov     QWORD PTR [rbp-248], r8
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-8], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-16], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-24], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-32], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-40], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-48], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-56], xmm0
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-64], xmm0
```

```
        mov     DWORD PTR [rbp-76], 0
        jmp     .L20
.L21:
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-216]
        add     rax, rdx
        movsd   xmm0, QWORD PTR [rax]
        movsd   QWORD PTR [rbp-88], xmm0
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-224]
        add     rax, rdx
        movsd   xmm0, QWORD PTR [rax]
        movsd   QWORD PTR [rbp-96], xmm0
        movsd   xmm0, QWORD PTR [rbp-88]
        mulsd   xmm0, xmm0
        movsd   QWORD PTR [rbp-104], xmm0
        movsd   xmm0, QWORD PTR [rbp-8]
        addsd   xmm0, QWORD PTR [rbp-88]
        movsd   QWORD PTR [rbp-8], xmm0
        movsd   xmm0, QWORD PTR [rbp-16]
        addsd   xmm0, QWORD PTR [rbp-96]
        movsd   QWORD PTR [rbp-16], xmm0
        movsd   xmm0, QWORD PTR [rbp-88]
        mulsd   xmm0, QWORD PTR [rbp-96]
        movsd   xmm1, QWORD PTR [rbp-24]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-24], xmm0
        movsd   xmm0, QWORD PTR [rbp-32]
        addsd   xmm0, QWORD PTR [rbp-104]
        movsd   QWORD PTR [rbp-32], xmm0
        movsd   xmm0, QWORD PTR [rbp-96]
        mulsd   xmm0, xmm0
        movsd   xmm1, QWORD PTR [rbp-40]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-40], xmm0
        movsd   xmm0, QWORD PTR [rbp-88]
        mulsd   xmm0, QWORD PTR [rbp-104]
        movsd   xmm1, QWORD PTR [rbp-48]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-48], xmm0
        movsd   xmm0, QWORD PTR [rbp-104]
        mulsd   xmm0, xmm0
        movsd   xmm1, QWORD PTR [rbp-56]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-56], xmm0
        movsd   xmm0, QWORD PTR [rbp-104]
        mulsd   xmm0, QWORD PTR [rbp-96]
        movsd   xmm1, QWORD PTR [rbp-64]
        addsd   xmm0, xmm1
```

```
        movsd   QWORD PTR [rbp-64], xmm0
        add     DWORD PTR [rbp-76], 1
.L20:
        cmp     DWORD PTR [rbp-76], 8
        jle     .L21
        movsd   xmm0, QWORD PTR .LC11[rip]
        movsd   QWORD PTR [rbp-176], xmm0
        movsd   xmm0, QWORD PTR [rbp-8]
        movsd   QWORD PTR [rbp-152], xmm0
        movsd   xmm0, QWORD PTR [rbp-8]
        movsd   QWORD PTR [rbp-168], xmm0
        movsd   xmm0, QWORD PTR [rbp-32]
        movsd   QWORD PTR [rbp-128], xmm0
        movsd   xmm0, QWORD PTR [rbp-32]
        movsd   QWORD PTR [rbp-160], xmm0
        movsd   xmm0, QWORD PTR [rbp-32]
        movsd   QWORD PTR [rbp-144], xmm0
        movsd   xmm0, QWORD PTR [rbp-48]
        movsd   QWORD PTR [rbp-120], xmm0
        movsd   xmm0, QWORD PTR [rbp-48]
        movsd   QWORD PTR [rbp-136], xmm0
        movsd   xmm0, QWORD PTR [rbp-56]
        movsd   QWORD PTR [rbp-112], xmm0
        movsd   xmm0, QWORD PTR [rbp-16]
        movsd   QWORD PTR [rbp-208], xmm0
        movsd   xmm0, QWORD PTR [rbp-24]
        movsd   QWORD PTR [rbp-200], xmm0
        movsd   xmm0, QWORD PTR [rbp-64]
        movsd   QWORD PTR [rbp-192], xmm0
        mov     rdx, QWORD PTR [rbp-240]
        lea     rcx, [rbp-208]
        lea     rax, [rbp-176]
        mov     rsi, rcx
        mov     rdi, rax
        call    solve
        pxor    xmm0, xmm0
        movsd   QWORD PTR [rbp-72], xmm0
        mov     DWORD PTR [rbp-76], 0
        jmp     .L22
.L23:
        mov     rax, QWORD PTR [rbp-240]
        movsd   xmm1, QWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-240]
        add     rax, 8
        movsd   xmm2, QWORD PTR [rax]
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-216]
        add     rax, rdx
        movsd   xmm0, QWORD PTR [rax]
        mulsd   xmm0, xmm2
        addsd   xmm1, xmm0
```

```
        movsd   QWORD PTR [rbp-256], xmm1
        mov     rax, QWORD PTR [rbp-240]
        add     rax, 16
        movsd   xmm3, QWORD PTR [rax]
        movsd   QWORD PTR [rbp-264], xmm3
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-216]
        add     rax, rdx
        mov     rax, QWORD PTR [rax]
        movsd   xmm0, QWORD PTR .LC12[rip]
        movapd  xmm1, xmm0
        movq    xmm0, rax
        call    pow
        mulsd   xmm0, QWORD PTR [rbp-264]
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-232]
        add     rax, rdx
        addsd   xmm0, QWORD PTR [rbp-256]
        movsd   QWORD PTR [rax], xmm0
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-224]
        add     rax, rdx
        movsd   xmm0, QWORD PTR [rax]
        mov     eax, DWORD PTR [rbp-76]
        cdqe
        lea     rdx, [0+rax*8]
        mov     rax, QWORD PTR [rbp-232]
        add     rax, rdx
        movsd   xmm1, QWORD PTR [rax]
        subsd   xmm0, xmm1
        movq    rax, xmm0
        movsd   xmm0, QWORD PTR .LC12[rip]
        movapd  xmm1, xmm0
        movq    xmm0, rax
        call    pow
        movsd   xmm1, QWORD PTR [rbp-72]
        addsd   xmm0, xmm1
        movsd   QWORD PTR [rbp-72], xmm0
        add     DWORD PTR [rbp-76], 1
.L22:
        cmp     DWORD PTR [rbp-76], 8
        jle     .L23
        movsd   xmm0, QWORD PTR .LC12[rip]
        mov     rax, QWORD PTR [rbp-16]
        movapd  xmm1, xmm0
        movq    xmm0, rax
        call    pow
```

```
        movq    rax, xmm0
        movsd   xmm0, QWORD PTR .LC11[rip]
        movq    xmm1, rax
        divsd   xmm1, xmm0
        movsd   xmm0, QWORD PTR [rbp-40]
        movapd  xmm2, xmm0
        subsd   xmm2, xmm1
        movsd   xmm0, QWORD PTR [rbp-72]
        movapd  xmm1, xmm0
        divsd   xmm1, xmm2
        movsd   xmm0, QWORD PTR .LC0[rip]
        subsd   xmm0, xmm1
        movq    rax, xmm0
        movq    xmm0, rax
        call    sqrt
        movq    rax, xmm0
        mov     rdx, QWORD PTR [rbp-248]
        mov     QWORD PTR [rdx], rax
        nop
        leave
        ret
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 288
        lea     rdx, [rbp-160]
        lea     rax, [rbp-80]
        mov     rsi, rdx
        mov     rdi, rax
        call    get_data
        lea     rdi, [rbp-280]
        lea     rcx, [rbp-272]
        lea     rdx, [rbp-240]
        lea     rsi, [rbp-160]
        lea     rax, [rbp-80]
        mov     r8, rdi
        mov     rdi, rax
        call    linfit
        mov     eax, 0
        leave
        ret
.LC0:
        .long   0
        .long   1072693248
.LC1:
        .long   687194767
        .long   1073778524
.LC2:
        .long   858993459
        .long   1075917619
.LC3:
        .long   1030792151
        .long   1076680458
```

```
.LC4:
        .long   -1717986918
        .long   1076861337
.LC5:
        .long   515396076
        .long   1077078917
.LC6:
        .long   -1889785610
        .long   1077016002
.LC7:
        .long   -1889785610
        .long   1076491714
.LC8:
        .long   -858993459
        .long   1075432652
.LC9:
        .long   343597384
        .long   1070679982
.LC11:
        .long   0
        .long   1075970048
.LC12:
        .long   0
        .long   1073741824
```