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 _____ Мухин А. М.

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 на языке Паскаль, С и Assembler представлены в приложениях А, Б и В, соответственно.

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	()	10	1	0.0	1
2	*	9	2	1	23
3	+	1	3	12	1
4	-	6	4	-134	1
5	/	1	5	145	1
6	;	28	6	19	1
7	:=	21	7	-199	1
8	>	1	8	2	19
9	[]	34	9	213	1
10	setup	3	10	3	21
11	solve	1	11	325	1
12	const	1	12	-43	1
13	deter	2	13	81	1
14	exit	1	14	99	1
15	for	3	15	991	1
16	If…then	2	16	a	28
17	get_data	1	17	y	6
			18	rmax	4
			19	b	8
			20	cmax	1
			21	coef	5
			22	det	3
			23	deter	1
			24	i	8
			25	j	8

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	<	3	1	-43	1
2	==	1	2	-134	1
3	()	16	3	-199	1
4	*	9	4	0	22
5	+	1	5	1	22
6	++	3	6	12	1
7	solve	1	7	y	9
8	-	6	8	145	1
9	/	1	9	19	1
10	;	31	10	j	10
11	for	3	11	2	19
12	=	20	12	213	1

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
13	>	1	13	325	1
14	[]	63	14	i	12
15	setup	3	15	81	1
16	deter	2	16	99	1
17	if…then	2	17	991	1
18	get_data	1	18	CMAX	1
19	return	3	19	RMAX	2
			20	a	31
			21	b	8
			22	coef	5
			23	det	6

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	push	5	1	rbp	12
2	mov	100	2	rsp	8
3	movsd	47	3	[rbp-8]	31
4	add	39	4	rdi	17
5	nop	3	5	[rbp-16]	10
6	pop	2	6	rsi	8
7	ret	5	7	rax	106
8	mulsd	9	8	xmm0	57
9	subsd	4	9	.LC0[rip]	1
10	movapd	1	10	[rax]	11
11	addsd	1	11	.LC1[rip]	1
12	movq	8	12	[rax+8]	8
13	sub	2	13	.LC2[rip]	1
14	jmp .L5	2	14	[rax+16]	8
15	cdqe	3	15	.LC3[rip]	1
16	lea	15	16	24	9
17	movsx	9	17	.LC4[rip]	1
18	sal	4	18	.LC5[rip]	1
19	cmp	4	19	.LC6[rip]	1
20	jle .L6	1	20	8	1
21	jle .L7	1	21	.LC7[rip]	1
22	call deter	2	22	48	9
23	divsd	1	23	.LC8[rip]	1

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
24	jmp .L9	1	24	.LC9[rip]	1
25	jmp .L10	1	25	.LC10[rip]	1
26	jle .L11	1	26	16	1
27	jle .L12	1	27	.LC11[rip]	1
28	pxor	2	28	xmm1	9
29	ucomisd	2	29	xmm2	10
30	jp .L13	1	30	xmm3	12
31	je .L16	1	31	xmm4	4
32	call setup	3	32	64	1
33	jmp .L8	1	33	[rbp-24]	2
34	call get_data	1	34	[rbp-32]	4
35	call solve	1	35	rdx	34
36	leave	3	36	[rbp-48]	2
			37	rcx	12
			38	[rbp-52]	6
			39	r8d	4
			40	[rbp-64]	2
			41	[rbp-4]	12
			42	0	18
			43	eax	18
			44	[0+rax*8]	1
			45	[rdx+rax]	2
			46	3	4
			47	[rcx]	1
			48	[rdx+rax*8]	3
			49	edi	2
			50	[rax-1]	2
			51	esi	2
			52	[rcx+rax*8]	1
			53	1	4
			54	2	4
			55	edx	2
			56	[0+rdx*8]	1
			57	[rdx]	1
			58	-128	1
			59	[rbp-104]	5
			60	[rbp-112]	6
			61	[rbp-120]	4
			62	[rbp-96+rax*8]	1
			63	[rbp-96]	4

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			64	144	1
			65	[rbp-144]	1

Расчет метрик с помощью программы автоматизации

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	()	12	1	0.0	1
2	*	9	2	1	26
3	+	1	3	12	1
4	-	9	4	134	1
5	/	1	5	145	1
6	;	87	6	19	1
7	=	23	7	199	1
8	>	1	8	2	19
9	[]	34	9	213	1
10	ary2s	6	10	3	21
11	arys	5	11	325	1
12	const	1	12	43	1
13	deter	3	13	81	1
14	exit	1	14	99	1
15	for	3	15	991	1
16	function	1	16	a	32
17	get_data	2	17	ary2s	1
18	if	2	18	arys	1
19	integer	3	19	b	10
20	procedure	3	20	cmax	3
21	program	1	21	coef	8
22	real	4	22	det	4
23	setup	4	23	deter	1
24	solve	2	24	i	8
25	type	1	25	j	9
			26	rmax	5
			27	simq1	1
			28	y	9

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	<	3	1	0	22
2	==	1	2	1	22
3	()	15	3	12	1
4	*	9	4	134	1
5	+	1	5	145	1
6	++	3	6	19	1
7	,	27	7	199	1
8	-	6	8	2	19
9	/	1	9	213	1
10	;	44	10	325	1
11	_-	3	11	43	1
12	=	20	12	81	1
13	>	1	13	99	1
14	[]	63	14	991	1
15	_[]	21	15	CMAX	15
16	deter	3	16	RMAX	9
17	double	17	17	a	36
18	get_data	2	18	b	10
19	int	4	19	coef	8
20	main	1	20	det	8
21	for	3	21	i	14
22	return	3	22	j	12
23	if…then	2	23	y	13
24	setup	4	24		
25	solve	2	25		
26	void	3	26		

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):	17	25	19	26	36
Число уникальных операндов (n2):	25	28	23	23	65
Общее число операторов (N1):	125	219	170	262	287
Общее число операндов (N2):	147	170	158	199	511
Словарь (n):	42	53	42	49	101
Экспериментальная длина программы (Nэ):	272	389	328	461	798
Теоретическая длина программы (Nт):	185,58	250	184,75	226,253	577,57
Объём программы (V):	1466,7	2228,16	1768,6	2588,38	5313,25
Потенциальный объём (V*):	11,6	11,6	11,6	11,6	11,6
Уровень программы (L):	0,0079	0,0052	0,0065	0,0044	0,0021
Интеллект программы (I):	29,35	29,36	27,1	23,01	37,54
Работа по программированию (E):	185297,6	427636	269451	577082	2431656
Время кодирования (T):	18529,76	23757,6	26945,1	32060,1	243165,6
Уровень языка программирования (Lam):	0,091	0,06	0,076	0,052	0,025
Уровень ошибок (B):	2	2	2	3	6

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 simq1;

const rmax = 3;
    cmax = 3;

type arys     = array[1..cmax] of real;
    ary2s     = array[1..rmax,1..cmax] of real;

var y,coef: arys;
    a: ary2s;

procedure get_data(var a: ary2s;
           var y: arys);
begin
    a[1,1] := 1;
    a[1,2] := -43;
    a[1,3] := 19;
    y[1] := 81;
    a[2,1] := 145;
    a[2,2]:= -134;
    a[2,3] := 99;
    y[2] := 12;
    a[3,1]:= 325;
    a[3,2] := 991;
    a[3,3] := -199;
    y[3] := 213;
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 rmax 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 rmax do
    for j:=1 to rmax 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;

begin
  get_data(a, y);
  solve(a, y, coef);
end.
```

ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

#define RMAX 3
#define CMAX 3


void get_data(double a[RMAX][CMAX], double y[CMAX]) {
    a[0][0] = 1;
    a[0][1] = -43;
    a[0][2] = 19;
    y[0] = 81;
    a[1][0] = 145;
    a[1][1] = -134;
    a[1][2] = 99;
    y[1] = 12;
    a[2][0] = 325;
    a[2][1] = 991;
    a[2][2] = -199;
    y[2] = 213;
}


double deter(double a[RMAX][CMAX]) {
    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[RMAX][CMAX], double b[RMAX][CMAX], double
y[CMAX], double coef[CMAX], int j, double det) {
    int i;
    for (i = 0; i < RMAX; 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[RMAX][CMAX], double y[CMAX], double
coef[CMAX]) {
    double b[RMAX][CMAX];
    int i, j;
    double det;

    for (i = 0; i < RMAX; i++) {
         for (j = 0; j < CMAX; 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);
        }
    }


    int main() {
        double a[RMAX][CMAX];
        double y[CMAX];
        double coef[CMAX];

        get_data(a, y);
        solve(a, y, coef);

        return 0;
    }
```

ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
get_data:
        push    rbp
        mov     rbp, rsp
        mov     QWORD PTR [rbp-8], rdi
        mov     QWORD PTR [rbp-16], rsi
        mov     rax, QWORD PTR [rbp-8]
        movsd   xmm0, QWORD PTR .LC0[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-8]
        movsd   xmm0, QWORD PTR .LC1[rip]
        movsd   QWORD PTR [rax+8], xmm0
        mov     rax, QWORD PTR [rbp-8]
        movsd   xmm0, QWORD PTR .LC2[rip]
        movsd   QWORD PTR [rax+16], xmm0
        mov     rax, QWORD PTR [rbp-16]
        movsd   xmm0, QWORD PTR .LC3[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm0, QWORD PTR .LC4[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm0, QWORD PTR .LC5[rip]
        movsd   QWORD PTR [rax+8], xmm0
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 24
        movsd   xmm0, QWORD PTR .LC6[rip]
        movsd   QWORD PTR [rax+16], xmm0
        mov     rax, QWORD PTR [rbp-16]
        add     rax, 8
        movsd   xmm0, QWORD PTR .LC7[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm0, QWORD PTR .LC8[rip]
        movsd   QWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm0, QWORD PTR .LC9[rip]
        movsd   QWORD PTR [rax+8], xmm0
        mov     rax, QWORD PTR [rbp-8]
        add     rax, 48
        movsd   xmm0, QWORD PTR .LC10[rip]
        movsd   QWORD PTR [rax+16], xmm0
        mov     rax, QWORD PTR [rbp-16]
        add     rax, 16
        movsd   xmm0, QWORD PTR .LC11[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     .L5
.L7:
        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     .L6
        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
.L6:
        add     DWORD PTR [rbp-4], 1
.L5:
        cmp     DWORD PTR [rbp-4], 2
        jle     .L7
        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     .L9
.L12:
        mov     DWORD PTR [rbp-8], 0
        jmp     .L10
.L11:
        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
.L10:
        cmp     DWORD PTR [rbp-8], 2
        jle     .L11
        add     DWORD PTR [rbp-4], 1
.L9:
        cmp     DWORD PTR [rbp-4], 2
        jle     .L12
        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      .L13
        pxor    xmm0, xmm0
        ucomisd xmm0, QWORD PTR [rbp-16]
        je      .L16
.L13:
        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     .L8
.L16:
        nop
.L8:
        leave
        ret
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 144
        lea     rdx, [rbp-112]
        lea     rax, [rbp-80]
        mov     rsi, rdx
        mov     rdi, rax
        call    get_data
        lea     rdx, [rbp-144]
        lea     rcx, [rbp-112]
        lea     rax, [rbp-80]
        mov     rsi, rcx
        mov     rdi, rax
        call    solve
        mov     eax, 0
        leave
        ret
.LC0:
        .long   0
        .long   1072693248
.LC1:
        .long   0
        .long   -1069187072
.LC2:
        .long   0
        .long   1077084160
.LC3:
        .long   0
        .long   1079263232
.LC4:
        .long   0
        .long   1080172544
```

```
.LC5:
        .long   0
        .long   -1067401216
.LC6:
        .long   0
        .long   1079558144
.LC7:
        .long   0
        .long   1076363264
.LC8:
        .long   0
        .long   1081364480
.LC9:
        .long   0
        .long   1083111424
.LC10:
        .long   0
        .long   -1066868736
.LC11:
        .long   0
        .long   1080729600
```