

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Качество и метрология программного обеспечения»
Тема: Расчет метрических характеристик качества разработки
программ по метрикам Холстеда

Студент гр. 8304

Рыжиков А.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Для заданного варианта программы обработки данных (программа 15 - Приближенная линеаризация опытных данных) , представленной на языке Паскаль, разработать вычислительный алгоритм и также варианты программ его реализации на языках программирования Си и Ассемблер.

Ход выполнения.

1. Был выполнен ручной расчет характеристик программы на языке Паскаль — см. табл. 1, табл. 2, табл. 3. Исходный код программы см. в приложении А.

Таблица 1 — Количество операторов и операндов программы на языке Паскаль

№	Оператор	Количество	Операнд	Количество
1	()	8	' ; '	3
2	*	9	'x: '	1
3	+	6	'y: '	1
4	-	4	'y_calc: '	1
5	/	8	0.0	5
6	;	65	1	7
7	=	22	100	4
8	[]	65	2	3
9	ary	3	5	4
10	const	1	a	5
11	for	6	ary	1
12	integer	3	b	5
13	linfit2	2	i	11
14	procedure	1	n	14
15	program	1	simp1	1
16	random	2	sum_x	8
17	real	4	sum_x2	6
18	type	1	sum_xy	6
19	write	6	sum_y	8
20	writeln	3	sum_y2	5

21			sxx	4
22			sxy	3
23			syу	2
24			x	7
25			xi	6
26			y	6
27			y_calc	5
28			yi	6

Таблица 2 — Измеримые характеристики программы на языке Паскаль

Число простых операторов	20
Число простых операндов	28
Общее число всех операторов	164
Общее число всех операндов	138
Словарь программы	48
Длина программы	302

Таблица 3 — Расчетные характеристики программы на языке Паскаль

Оценка длины программы	221
Реальный объем работы	1686.6
Потенциальный объем работы	15.68
Уровень программы	0.0068
Оценка уровня программы	0.0202
Интеллектуальное содержание программы	34.22
Работа программиста	245039
Время программирования	24503.9
Уровень используемого языка программирования	0.079
Ожидаемое число ошибок в программе	1.305

2. Был выполнен автоматический расчет характеристик программы на языке Паскаль — см. рис. 1. Результаты совпали с ручным расчетом.

Operators:					
1	8	()			
2	9	*			
3	6	+			
4	4	-			
5	8	/			
6	65	;			
7	22	=			
8	9	[]			
9	3	ary			
10	1	const			
11	6	for			
12	3	integer			
13	2	linfit1			
14	1	procedure			
15	1	program			
16	2	random			
17	4	real			
18	1	type			
19	6	write			
20	3	writeln			
Operands:					
1	3	' ; '			
2	1	'x: '			
3	1	'y: '			
4	1	'y_calc: '			
5	5	0.0			
6	7	1			
7	4	100			
8	3	2			
9	4	5			
10	5	a			
11	1	ary			
12	5	b			
13	11	i			
14	14	n			
15	1	simp1			
16	8	sum_x			
17	6	sum_x2			
18	6	sum_xy			
19	8	sum_y			
20	5	sum_y2			
21	4	sxx			
22	3	sxy			
23	2	syy			
24	7	x			
25	6	xi			
26	6	y			
27	5	y_calc			
28	6	yi			
			Summary:		
			=====		
			The number of different operators	:	20
			The number of different operands	:	28
			The total number of operators	:	164
			The total number of operands	:	138
			Dictionary	(D)	: 48
			Length	(N)	: 302
			Length estimation	(^N)	: 221.044
			Volume	(V)	: 1686.66
			Potential volume	(*V)	: 11.6096
			Limit volume	(**V)	: 15.6844
			Programming level	(L)	: 0.00688322
			Programming level estimation	(^L)	: 0.0202899
			Intellect	(I)	: 34.2221
			Time of programming	(T)	: 24503.9
			Time estimation	(^T)	: 6084.45
			Programming language level	(lambda)	: 0.0799117
			Work on programming	(E)	: 245039
			Error	(B)	: 1.30528
			Error estimation	(^B)	: 0.56222

Рисунок 1 - Автоматический расчет характеристик программы на языке Паскаль

3. Был выполнен ручной расчет характеристик программы на языке Си — см. табл. 4, табл. 5, табл. 6. Исходный код программы см. в приложении В.

Таблица 4 - Количество операторов и операндов программы на языке Си

№	Оператор	Количество	Операнд	Количество
1	%	2	"%f ;"	3
2	()	26	"\ny: "	1
3	*	11	"\ny_calc: "	1
4	+	2	"x: "	1
5	++	7	0	8
6	+=	6	100	4
7	,	18	5	4
8	-	4	a	2
9	/	9	b	2
10	;	45	i	28
11	<	10	n	13
12	=	22	sum_x	7
13	[]	13	sum_x2	4
14	&	1	sum_xy	4
15	float	12	sum_y	6
16	for	6	sum_y2	3
17	int	8	sxx	4
18	linfit2	2	sxy	3
19	main	1	syy	2
20	printf	6	x	7
21	rand	2	xi	6
22	void	1	y	6
23			y_calc	5
24			yi	6
25				

Таблица 5 — Измеримые характеристики программы на языке Си

Число простых операторов	22
Число простых операндов	24
Общее число всех операторов	214
Общее число всех операндов	130
Словарь программы	46
Длина программы	344

Таблица 6 — Расчетные характеристики программы на языке Си

Оценка длины программы	208.14
Реальный объем работы	1900.10464
Потенциальный объем работы	15.5094
Уровень программы	0.00816
Оценка уровня программы	0.0167
Интеллектуальное содержание программы	31.88
Работа программиста	232903
Время программирования	23290
Уровень используемого языка программирования	0.126
Ожидаемое число ошибок в программе	1.9067

4. Был выполнен автоматический расчет характеристик программы на языке Си — см. рис. 2. Результаты не совпали с ручным расчетом.

Operators:								
1	2	%						
2	15	()						
3	9	*						
4	2	+						
5	6	++						
6	5	+=						
7	15	,						
8	4	-						
9	8	/						
10	52	;						
11	6	<						
12	22	=						
13	8	[]						
14	1	&						
15	1	*						
16	5	_[]						
17	1	*						
18	12	float						
19	6	for						
20	8	int						
21	2	linfit1						
22	1	main						
23	6	printf						
24	2	rand						
25	1	void						
Operands:								
1	3	"%f ;"						
2	1	"\ny: "						
3	1	"\ny_calc: "						
4	1	"x: "						
5	11	0						
6	4	100						
7	4	5						
8	2	a						
9	2	b						
10	27	i						
11	13	n						
12	6	sum_x						
13	4	sum_x2						
14	4	sum_xy						
15	6	sum_y						
16	3	sum_y2						
17	4	sxx						
18	3	sxy						
19	2	syy						
20	7	x						
21	6	xi						
22	6	y						
23	5	y_calc						
24	6	yi						
			Summary:					
			=====					
			The number of different operators			:	25	
			The number of different operands			:	24	
			The total number of operators			:	200	
			The total number of operands			:	131	
			Dictionary	(D)	:	49		
			Length	(N)	:	331		
			Length estimation	(^N)	:	226.136		
			Volume	(V)	:	1858.47		
			Potential volume	(*V)	:	11.6096		
			Limit volume	(**V)	:	15.6844		
			Programming level	(L)	:	0.00624688		
			Programming level estimation	(^L)	:	0.0146565		
			Intellect	(I)	:	27.2386		
			Time of programming	(T)	:	29750.3		
			Time estimation	(^T)	:	8662.96		
			Programming language level	(lambda)	:	0.0725241		
			Work on programming	(E)	:	297503		
			Error	(B)	:	1.4855		
			Error estimation	(^B)	:	0.61949		

Рисунок 2 - Автоматический расчет характеристик программы на языке Си

5. Был выполнен ручной расчет характеристик программы на языке Ассемблер — см. табл. 7, табл. 8, табл. 9. Исходный код программы см. в приложении С.

Таблица 7 - Количество операторов и операндов программы на языке Ассемлера

№	Оператор	Количество	Операнд	Количество
1	push	2	rbp	5
2	mov	109	rbp-72	3
3	pxor	14	rsp	3
4	movss	45	5	5
5	jmp	6	xmm0	91
6	cdqe	9	rbp-80	3
7	lea	8	rbp-88	2
8	add	16	rbp-4	11
9	addss	6	rbp-8	10
10	mulss	9	rbp-12	9
11	cmp	6	rbp-16	9
12	movaps	3	rbp-20	4
13	divss	8	rbp-24	5
14	subss	4	rbp-72	3
15	call	9	rax	29
16	movsx	2	rbp-52	4
17	imul	4	rbp-56	4
18	shr	2	0+rax*4	6
19	sar	4	eax	44
20	cvtss2ss	6	rdx	18
21	pxor	14	xmm1	26
22	sub	9	rip	12
23	pop	1	rbp-40	1
24	ret	2	rbp-36	2
25	cmp	6	rbp-32	3
26	leave	1	rbp-44	2
27	cvtss2sd	3	rbp-28	5
28	DWORD PTR	92	rdx	18
29	QWORD PTR	7	1	7
30	:	29	rsp	3

31	,	201	112	1
32	[]	106	0	9
33	addss	6	1374389535	2
34	.string	4	32	2
35	.long	2	31	2
36	jl	6	ecx	10
37			edx	10
38			rbp-48+rax*4	2
39			rand	2
40			rbp-80+rax*4	2
41			rbp-112	2
42			rcx	2
43			linfit2	2
44			xmm2	16
45			printf	6
46			xmm3	4
47			edi	6
48			xmm4	4
49			1120403456	1
50			n	1
51			.L3	2
52			.L2	2
53			.L5	2
54			.L4	2
55			.LC2	2
56			.LC3	4
57			"x: "	1
58			"%f ;"	1
59			"\ny: "	1
60			"\ny_calc: "	1
61			.LC4	2
62			.LC5	2
63			main	1

64			.L8	2
65			.L7	2
66			.L10	2
67			.L9	2
68			.L12	2
69			.L11	2
70			.L14	2
71			.L13	2
72			.LC1	3
73			112	1

Таблица 8 — Измеримые характеристики программы на языке Ассемлера

Число простых операторов	36
Число простых операндов	73
Общее число всех операторов	761
Общее число всех операндов	476
Словарь программы	109
Длина программы	1237

Таблица 9 — Расчетные характеристики программы на языке Ассемлера

Оценка длины программы	637.97
Реальный объем работы	8372.24
Потенциальный объем работы	15.5
Уровень программы	0.0018525
Оценка уровня программы	0.00852
Интеллектуальное содержание программы	71.33
Работа программиста	4519373
Время программирования	451937
Уровень используемого языка программирования	0.028732
Ожидаемое число ошибок в программе	8.37

6. Была сформирована сводная таблица — табл. 10.

Таблица 10 — Сводная таблица характеристик программа

Характеристика	Паскаль	Си	Ассемблер
Число простых операторов	20	25	36
Число простых операндов	28	24	73
Общее число всех операторов	164	200	761
Общее число всех операндов	138	131	476
Словарь программы	48	131	109
Длина программы	302	331	1237
Оценка длины программы	221.044	226.1	637.97
Реальный объем работы	1686.66	1858.47	8372.24
Потенциальный объем работы	15.6844	15.68	15.5
Уровень программы	0.0068	0.00624	0.0018525
Оценка уровня программы	0.02028	0.01465	0.00852
Интеллектуальное содержание программы	34.22	27.23	71.33
Работа программиста	245039	297503	4519373
Время программирования	24503.9	29750.3	451937
Уровень используемого языка программирования	0.0799	0.0725	0.028732
Ожидаемое число ошибок в программе ($E_{\text{крит}} =$	1.305	1.48	8.37

Выводы.

В ходе выполнения лабораторной работы был проведён анализ метрических характеристик алгоритма линеаризация опытных данных на разных языках (Си, Паскаль, Ассемблер). Характеристики программ на Си и Паскале сопоставимы, в отличие от характеристик программы на Ассемблере из-за того, что Си и Паскаль языки более высокого уровня чем ассемблер.

ПРИЛОЖЕНИЕ А. ПРОГРАММА НА ЯЗЫКЕ ПАСКАЛЬ

```
program simpl;

const n = 5;
type ary = array[1..n] of real;

var x,y,y_calc : ary;
i : integer;
a,b : real;

procedure linfit2(x,y : ary;
var y_calc : ary;
var a,b : real;
n: integer);
var i : integer;
sum_x,sum_y,sum_xy,sum_x2,
sum_y2,xi,yi,sxy,sxx,
syy : 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
y_calc[i]:=a+b*x[i]
end;

begin
for i:=1 to n do
begin
x[i]:= random(100)/100;
y[i]:= random(100)/100;
end;
linfit2(x,y,y_calc,a,b,n);
writeln;
write('x: ');
for i:=1 to n do
write(x[i]:5:2,' ; ');
writeln;
write('y: ');
for i:=1 to n do
write(y[i]:5:2,' ; ');
writeln;
write('y_calc: ');
for i:=1 to n do
write(y_calc[i]:5:2,' ; ');
end.
end.
```

ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ СИ

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

int n = 5;

void linfit2(float x[n], float y[n], float *y_calc) {

    float sum_x = 0, sum_y = 0, sum_xy = 0,
          sum_x2 = 0, sum_y2 = 0;
    float xi, yi, sxy, sxx, syy;
    for (int i = 0; i < n; ++i) {

        xi = x[i];
        yi = y[i];
        sum_x += xi;
        sum_y += yi;
        sum_xy += xi * yi;
        sum_x2 += xi * xi;
        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;

    float b = sxy / sxx;
    float 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];
    }
}

int main() {
    float x[5];
    float y[5];
    float y_calc[5];

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

    linfit2(x, y, &y_calc);

    printf("x: ");

    for (int i = 0; i < n; ++i) {
        printf("%f ;", x[i]);
    }

    printf("\ny: ");

    for (int i = 0; i < n; ++i) {
        printf("%f ;", y[i]);
    }

    printf("\ny_calc: ");

    for (int i = 0; i < n; ++i) {
        printf("%f ;", y_calc[i]);
    }
}
```

ПРИЛОЖЕНИЕ С. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕРА

```
n:
    .long    5
linfit2:
    push    rbp
    mov     rbp, rsp
    mov     QWORD PTR [rbp-72], rdi
    mov     QWORD PTR [rbp-80], rsi
    mov     QWORD PTR [rbp-88], rdx
    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-72]
    add     rax, rdx
    movss    xmm0, DWORD PTR [rax]
    movss    DWORD PTR [rbp-52], xmm0
    mov     eax, DWORD PTR [rbp-24]
    cdqe
    lea     rdx, [0+rax*4]
    mov     rax, QWORD PTR [rbp-80]
    add     rax, rdx
    movss    xmm0, DWORD PTR [rax]
    movss    DWORD PTR [rbp-56], xmm0
    movss    xmm0, DWORD PTR [rbp-4]
    addss    xmm0, DWORD PTR [rbp-52]
    movss    DWORD PTR [rbp-4], xmm0
    movss    xmm0, DWORD PTR [rbp-8]
    addss    xmm0, DWORD PTR [rbp-56]
    movss    DWORD PTR [rbp-8], xmm0
    movss    xmm0, DWORD PTR [rbp-52]
    mulss    xmm0, DWORD PTR [rbp-56]
    movss    xmm1, DWORD PTR [rbp-12]
    addss    xmm0, xmm1
    movss    DWORD PTR [rbp-12], xmm0
    movss    xmm0, DWORD PTR [rbp-52]
    mulss    xmm0, xmm0
    movss    xmm1, DWORD PTR [rbp-16]
    addss    xmm0, xmm1
    movss    DWORD PTR [rbp-16], xmm0
    movss    xmm0, DWORD PTR [rbp-56]
    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 n[rip]
```

```

    cmp     DWORD PTR [rbp-24], eax
    jl      .L3
    movss   xmm0, DWORD PTR [rbp-4]
    mulss   xmm0, xmm0
    mov     eax, DWORD PTR n[rip]
    pxor    xmm2, xmm2
    cvtsi2ss    xmm2, eax
    movaps  xmm1, xmm0
    divss   xmm1, xmm2
    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]
    mov     eax, DWORD PTR n[rip]
    pxor    xmm2, xmm2
    cvtsi2ss    xmm2, eax
    movaps  xmm1, xmm0
    divss   xmm1, xmm2
    movss   xmm0, DWORD PTR [rbp-12]
    subss   xmm0, xmm1
    movss   DWORD PTR [rbp-36], xmm0
    movss   xmm0, DWORD PTR [rbp-8]
    mulss   xmm0, xmm0
    mov     eax, DWORD PTR n[rip]
    pxor    xmm2, xmm2
    cvtsi2ss    xmm2, eax
    movaps  xmm1, xmm0
    divss   xmm1, xmm2
    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]
    movss   DWORD PTR [rbp-44], 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
    mov     eax, DWORD PTR n[rip]
    pxor    xmm1, xmm1
    cvtsi2ss    xmm1, eax
    divss   xmm0, xmm1
    divss   xmm0, DWORD PTR [rbp-32]
    movss   DWORD PTR [rbp-48], xmm0
    mov     DWORD PTR [rbp-28], 0
    jmp     .L4
.L5:
    mov     eax, DWORD PTR [rbp-28]
    cdqe
    lea     rdx, [0+rax*4]
    mov     rax, QWORD PTR [rbp-72]
    add     rax, rdx
    movss   xmm0, DWORD PTR [rax]
    mulss   xmm0, DWORD PTR [rbp-44]
    mov     eax, DWORD PTR [rbp-28]
    cdqe
    lea     rdx, [0+rax*4]
    mov     rax, QWORD PTR [rbp-88]
    add     rax, rdx
    addss   xmm0, DWORD PTR [rbp-48]
    movss   DWORD PTR [rax], xmm0

```



```

        add     DWORD PTR [rbp-28], 1
.L4:
        mov     eax, DWORD PTR n[rip]
        cmp     DWORD PTR [rbp-28], eax
        jl     .L5
        nop
        nop
        pop     rbp
        ret

.LC2:
        .string "x: "
.LC3:
        .string "%f ;"
.LC4:
        .string "\ny: "
.LC5:
        .string "\ny_calc: "
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 112
        mov     DWORD PTR [rbp-4], 0
        jmp     .L7

.L8:
        call    rand
        movsx   rdx, eax
        imul    rdx, rdx, 1374389535
        shr     rdx, 32
        sar     edx, 5
        mov     ecx, eax
        sar     ecx, 31
        sub     edx, ecx
        imul    ecx, edx, 100
        sub     eax, ecx
        mov     edx, eax
        pxor    xmm0, xmm0
        cvtsi2ss    xmm0, edx
        movss   xmm1, DWORD PTR .LC1[rip]
        divss   xmm0, xmm1
        mov     eax, DWORD PTR [rbp-4]
        cdq     rax
        movss   DWORD PTR [rbp-48+rax*4], xmm0
        call    rand
        movsx   rdx, eax
        imul    rdx, rdx, 1374389535
        shr     rdx, 32
        sar     edx, 5
        mov     ecx, eax
        sar     ecx, 31
        sub     edx, ecx
        imul    ecx, edx, 100
        sub     eax, ecx
        mov     edx, eax
        pxor    xmm0, xmm0
        cvtsi2ss    xmm0, edx
        movss   xmm1, DWORD PTR .LC1[rip]
        divss   xmm0, xmm1
        mov     eax, DWORD PTR [rbp-4]
        cdq     rax
        movss   DWORD PTR [rbp-80+rax*4], xmm0
        add     DWORD PTR [rbp-4], 1
.L7:
        mov     eax, DWORD PTR n[rip]

```

```

        cmp     DWORD PTR [rbp-4], eax
        jl      .L8
        lea     rdx, [rbp-112]
        lea     rcx, [rbp-80]
        lea     rax, [rbp-48]
        mov     rsi, rcx
        mov     rdi, rax
        call    linfit2
        mov     edi, OFFSET FLAT:.LC2
        mov     eax, 0
        call    printf
        mov     DWORD PTR [rbp-8], 0
        jmp     .L9
.L10:
        mov     eax, DWORD PTR [rbp-8]
        cdqe
        movss   xmm0, DWORD PTR [rbp-48+rax*4]
        pxor     xmm2, xmm2
        cvtss2sd xmm2, xmm0
        movq     rax, xmm2
        movq     xmm0, rax
        mov     edi, OFFSET FLAT:.LC3
        mov     eax, 1
        call    printf
        add     DWORD PTR [rbp-8], 1
.L9:
        mov     eax, DWORD PTR n[rip]
        cmp     DWORD PTR [rbp-8], eax
        jl      .L10
        mov     edi, OFFSET FLAT:.LC4
        mov     eax, 0
        call    printf
        mov     DWORD PTR [rbp-12], 0
        jmp     .L11
.L12:
        mov     eax, DWORD PTR [rbp-12]
        cdqe
        movss   xmm0, DWORD PTR [rbp-80+rax*4]
        pxor     xmm3, xmm3
        cvtss2sd xmm3, xmm0
        movq     rax, xmm3
        movq     xmm0, rax
        mov     edi, OFFSET FLAT:.LC3
        mov     eax, 1
        call    printf
        add     DWORD PTR [rbp-12], 1
.L11:
        mov     eax, DWORD PTR n[rip]
        cmp     DWORD PTR [rbp-12], eax
        jl      .L12
        mov     edi, OFFSET FLAT:.LC5
        mov     eax, 0
        call    printf
        mov     DWORD PTR [rbp-16], 0
        jmp     .L13
.L14:
        mov     eax, DWORD PTR [rbp-16]
        cdqe
        movss   xmm0, DWORD PTR [rbp-112+rax*4]
        pxor     xmm4, xmm4
        cvtss2sd xmm4, xmm0
        movq     rax, xmm4
        movq     xmm0, rax

```

```

        mov     edi, OFFSET FLAT:.LC3
        mov     eax, 1
        call    printf
        add     DWORD PTR [rbp-16], 1
.L13:
        mov     eax, DWORD PTR n[rip]
        cmp     DWORD PTR [rbp-16], eax
        jl      .L14
        mov     eax, 0
        leave
        ret
.LC1:
        .long   1120403456

```