

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»

Студентка гр. 8304

Николаева М. А.

Преподаватель

Кирияничков 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	;	35	1	0.0	1
2	begin... end	8	2	0.5	1
3	:=	37	3	1	22
4	for...to...do	2	4	1.0	3
5	if...then	4	5	x	3
6	repeat...until	1	6	upper	5
7	+	12	7	15	1
8	abs	4	8	true	3
9	<=	2	9	2	4
10	[]	17	10	4	1
11	>	2	11	4.0	2
12	romb	1	12	9.0	1
13	fx	3	13	c	3
14	or	1	14	delta_x	5
15	()	16	15	done	4
16	<>	1	16	error	2
17	*	10	17	false	2
18	-	14	18	fotom	5
19	/	4	19	fx	1
20	<>	1	20	i	2
21	div	1	21	ii	2
			22	j	7
			23	k	2
			24	l	2
			25	lower	6
			26	m	3
			27	n	9
			28	nn	9
			29	nt	2
			30	ntra	2
			31	nx	4
			32	pieces	6
			33	romb	1
			34	tol	3
			35	sum	5
			36	t	13

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	!	1	1	0	1
2	!=	1	2	0.0	1
3	()	19	3	0.5	1
4	*	10	4	1	22
5	+	12	5	1.0	3
6	++	2	6	x	3
7	-	14	7	upper	5
8	/	5	8	15	1
9	;	43	9	true	3
10	<=	4	10	2	4
11	=	36	11	4	3
12	>	2	12	9.0	1
13	[]	17	13	c	3
14	do...while	1	14	delta_x	5
15	fabs	4	15	done	4
16	for	2	16	error	2
17	fx	3	17	false	2
18	if...then	4	18	fotom	5
19	return	3	19	i	2
20	romb	1	20	ii	4
21		1	21	j	7
22	bool	2	22	k	2
23	float	6	23	l	2
24	int	6	24	ll	2
			25	lower	6
			26	m	5
			27	n	9
			28	nn	9
			29	nt	2
			30	ntra	2
			31	nx	4
			32	pieces	6
			33	sum	5
			34	t	13
			35	tol	5

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	push	3	1	rbp	7
2	mov	58	2	rsp	5
3	movss	49	3	[rpb-4]	4
4	divss	3	4	xmm0	81
5	pop	1	5	.LC0[rip]	2
6	ret	3	6	704	1
7	sub	10	7	[rbp-692]	5
8	subss	5	8	[rbp-696]	4
9	pxor	7	9	xmm1	45
10	cvtss2ss	3	10	[rbp-700]	3
11	movd	5	11	xmm2	8
12	call_fx	3	12	[rbp-1]	4
13	addss	3	13	0	8
14	cvtss2sd	3	14	[rbp-37]	2
15	movsd	2	15	[rbp-8]	7
16	cvtss2sd	2	16	1	26
17	mulss	7	17	[rbp-140]	1
18	add	13	18	[rbp-44]	5
19	cdqe	15	19	eax	69
20	sal	1	20	[rbp-704]	2
21	shr	1	21	.LC1[rip]	1
22	sar	1	22	[rbp-48]	3
23	jmp	3	23	[rbp-684]	1
24	cmp	4	24	[rbp-12]	10
25	jle	4	25	[rbp-16]	10
26	subsd	1	26	2	1
27	divsd	1	27	[rbp-20]	4
28	ucomiss	2	28	.LC2[rip]	2
29	jp	1	29	[rbp-24]	5
30	je	1	30	edx	9
31	andps	4	31	[rbp-144+rax*4]	3
32	comiss	2	32	[rbp-52]	2
33	jnb	1	33	31	1
34	jb	1	34	[rbp-56]	2
35	movzx	1	35	[rbp-32]	4
36	xor	1	36	[rbp-72]	2
37	test	1	37	[rbp-76]	2
38	jne	1	38	[rbp-688+rax*4]	12
39	leave	2	39	[rbp-60]	2
40	movaps	2	40	[rbp-64]	2

41	call_romb	1	41	[rbp-36]	5
----	-----------	---	----	----------	---

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
42	mulsd	1	42	[rbp-28]	7
			43	[rbp-68]	2
			44	.LC3[rip]	1
			45	4	1
			46	15	1
			47	al	2
			48	16	1
			49	.LC6[rip]	1
			50	.LC7[rip]	1

В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную.

	Паскаль	Си	Ассемблер
Число уникальных операторов (n1):	21	24	42
Число уникальных операндов (n2):	36	35	50
Общее число операторов (N1):	176	199	233
Общее число операндов (N2):	147	154	389
Словарь (n):	57	59	92
Экспериментальная длина программы (Nэ):	323	353	622
Теоретическая длина программы (Nт):	278,36	289,56	508,67
Объём программы (V):	1884	2076	4057,66
Потенциальный объём (V*):	15,51	15,51	15,51
Уровень программы (L):	0,008	0,007	0,004
Интеллект программы (I):	43,94	39,33	24,84
Работа по программированию (E):	228858,5	278028,2	1061560,7
Время кодирования (T):	22885,8	27802,8	106156,1
Уровень языка программирования (Lam):	0,1276	0,1158	0,0593
Уровень ошибок (B):	2	3	5

Расчет метрик с помощью программы автоматизации

В таблицах 5-6 представлены результаты программного подсчета числа типов операторов и операндов в программах на языках Паскаль и С.

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	()	24	1	0.0	1
2	*	10	2	0.5	1
3	+	12	3	1	24
4	-	14	4	1.0	3
5	/	5	5	1.0E-4	1
6	;	67	6	136	1
7	<=	2	7	15	1
8	<>	1	8	16	1
9	=	36	9	2	4
10	>	2	10	4	1
11	[]	17	11	4.0	2
12	abs	4	12	9.0	1
13	boolean	2	13	c	4
14	const	1	14	delta_x	6
15	for	2	15	done	6
16	function	2	16	error	3
17	fx	4	17	false	2
18	if	4	18	fotom	6
19	integer	2	19	fx	1
20	or	1	20	i	3
21	program	1	21	ii	2
22	real	7	22	j	8
23	repeat...until	1	23	k	3
24	romb	2	24	l	3
			25	lower	8
			26	m	3
			27	n	10
			28	nn	10
			29	nt	3
			30	ntra	3
			31	nx	5
			32	pieces	7
			33	romb	1
			34	romb1	1
			35	sum	7
			36	t	14
			37	tol	5

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			38	true	3
			39	upper	7
			40	x	5

Таблица 6 – Количество операторов и операндов в программе на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	!	1	1	0	1
2	!=	1	2	0.0	1
3	()	20	3	0.5	1
4	*	10	4	1	22
5	+	12	5	1.0	3
6	++	2	6	1.0E-4	1
7	,	10	7	136	1
8	-	14	8	15	1
9	/	5	9	16	1
10	;	52	10	2	4
11	<=	4	11	4	3
12	=	37	12	9.0	1
13	>	2	13	c	3
14	[]	17	14	delta_x	5
15	_[]	2	15	done	4
16	bool	3	16	error	2
17	const	1	17	false	2
18	do...while	1	18	fotom	6
19	fabs	4	19	i	3
20	float	15	20	ii	4
21	for	2	21	j	8
22	fx	4	22	k	3
23	if...then	4	23	l	3
24	int	10	24	ll	2
25	main	1	25	lower	7
26	return	3	26	m	5
27	romb	2	27	n	9
28	typedef	1	28	nn	9
29		1	29	nt	3
			30	ntra	3
			31	nx	5
			32	pieces	6
			33	sum	5

Таблица 6 (продолжение) – Количество операторов и операндов в программе на языке Си.

№	Оператор	Число вхождений	№	Операнд	Число вхождений
			34	t	14
			35	tol	5
			36	true	3
			37	upper	6
			38	x	5

В таблице 7 представлены сводные результаты расчетных характеристик программно.

Таблица 7 – Результаты расчетных характеристик программно.

	Паскаль	Си
Число уникальных операторов (n1):	24	29
Число уникальных операндов (n2):	40	38
Общее число операторов (N1):	223	241
Общее число операндов (N2):	180	170
Словарь (n):	64	67
Экспериментальная длина программы (Nэ):	403	411
Теоретическая длина программы (Nт):	322,9	340,3
Объем программы (V):	2418	2493,2
Потенциальный объем (V*):	15,51	15,51
Уровень программы (L):	0,006	0,006
Интеллект программы (I):	44,77	38,43
Работа по программированию (E):	376970	400770
Время кодирования (T):	20942,8	22265
Уровень языка программирования (Lam):	0,0995	0.0964
Уровень ошибок (B):	2	2

В таблице 8 представлены сводные результаты расчетных характеристик.

Таблица 8 – Сводная таблица.

	Паскаль вручную	Паскаль программно	Си вручную	Си программно	Ассемблер
Число уникальных операторов (n1):	21	24	24	29	42
Число уникальных операндов (n2):	36	40	35	38	50
Общее число операторов (N1):	176	223	199	241	233
Общее число операндов (N2):	147	180	154	170	389
Словарь (n):	57	64	59	67	92
Экспериментальная длина программы (Nэ):	323	403	353	411	622
Теоретическая длина программы (Nт):	278,36	322,9	289,56	340,3	508,67
Объём программы (V):	1884	2418	2076	2493,2	4057,66
Потенциальный объём (V*):	15,51	15,51	15,51	15,51	15,51
Уровень программы (L):	0,008	0,006	0,007	0,006	0,004
Интеллект программы (I):	43,94	44,77	39,33	38,43	24,84
Работа по программированию (E):	228858,5	376970	278028,2	400770	1061560,7
Время кодирования (T):	22885,8	20942,8	27802,8	22265	106156,1
Уровень языка программирования (Lam):	0,1276	0,0995	0,1158	0,0964	0,0593
Уровень ошибок (B):	2	2	3	2	5

Выводы.

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 rombl;  
  const    tol      = 1.0E-4;  
  var done      : boolean;  
      sum, upper, lower      : real;  
  
  function fx(x: real): real;  
  begin  
    fx:=1.0/x  
  end;  
  
  function romb(  
    lower, upper, tol: real):real;  
  var  
    nx          : array[1..16] of integer;  
    t           : array[1..136] of real;  
    done,error   : boolean;  
    pieces, nt, i, ii, n, nn,  
    l, ntra, k, m, j      : integer ;  
    delta_x, c, sum, fotom, x      : real ;  
  begin  
    done:=false;  
    error:=false;  
    pieces:=1;  
    nx[1]:=1;  
    delta_x:=(upper-lower)/pieces;  
    c:=(fx(lower)+fx(upper))*0.5;  
    t[1]:=delta_x*c;  
    n:=1;  
    nn:=2;  
    sum:=c;  
    repeat  
      n:=n+1;  
      fotom:=4.0;  
      nx[n]:=nn;  
      pieces:=pieces*2;  
      l:=pieces-1;  
      delta_x:=(upper-lower)/pieces;  
      for ii:=1 to (l+1) div 2 do  
        begin  
          i:=ii*2-1;  
          x:=lower+i*delta_x;  
          sum:=sum+fx(x)  
        end;  
      t[nn]:=delta_x*sum;  
      ntra:=nx[n-1];  
      k:=n-1;  
      for m:=1 to k do  
        begin
```

```

    j:=nn+m;
    nt:=nx[n-1]+m-1;
    t[j]:=(fotom*t[j-1]-t[nt])/(fotom-1.0);
    fotom:=fotom*4.0
    end;
    if n>4 then
        begin
            if t[nn+1]<>0.0 then
begin
                if (abs(t[ntra+1]-t[nn+1])<=abs(t[nn+1]*tol))
                    or (abs(t[nn-1]-t[j])<=abs(t[j]*tol)) then
                    done:=true
                else if n>15 then
                    begin
                        done:=true;
                        error:=true
                    end
                end;
            end;
            nn:=j+1
        until done;
        romb:=t[j]
    end;

begin
    lower:=1.0;
    upper:=9.0;

    sum:= romb(lower,upper,tol);
end.

```

ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

float fx (float x)
{
    return (1.0 / x);
}

float romb (float lower, float upper, float tol)
{
    int nx [16];
    float t [136];

    bool done = false;
    bool error = false;
    int pieces = 1;
    nx[1] = 1;
    float delta_x = (upper - lower) / pieces;
    float c = (fx(lower) + fx(upper)) * 0.5;
    t[1] = delta_x * c;
    int n = 1;
    int nn = 2;
    float sum = c;

    float fotom,x;
    int l,i,j,k,nt,ntra;
    do
    {
        n = n+1;
        fotom = 4;
        nx[n] = nn;
        pieces = pieces * 2;
        l = pieces - 1;
        delta_x = (upper - lower) / pieces;

        int ll = (l+1)/2;
        for(int ii = 1; ii <= ll; ii++)
        {
            i = ii * 2 - 1;
            x = lower + i * delta_x;
            sum = sum + fx(x);
        }

        t[nn] = delta_x * sum;

        ntra = nx[n-1];
        k = n-1;

        for(int m = 1; m <= k; m++)
```

```

    {
        j = nn+m;
        nt = nx[n - 1] + m - 1;
        t[j] = (fotom * t[j - 1] - t[nt]) / (fotom-1.0);
        fotom = fotom * 4;
    }

    if (n > 4)
    {
        if (t[nn + 1] != 0.0) {
            if ((fabs(t[ntra+1]-t[nn+1])<=fabs(t[nn+1]*tol))
                || (fabs(t[nn-1]-t[j])<=fabs(t[j]*tol)))
            {
                done = true;
            } else
            if (n>15) {
                done = true;
                error = true;
            }
        }
        nn = j+1;
    } while (!done);

    return (t[j]);
}

int main()
{
    const float tol = 1.0E-4;
    float lower = 1.0;
    float upper = 9.0;
    float sum = romb(lower,upper,tol);

    return 0;
}

```


ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
fx:
    push    rbp
    mov     rbp, rsp
    movss   DWORD PTR [rbp-4], xmm0
    movss   xmm0, DWORD PTR .LC0[rip]
    divss   xmm0, DWORD PTR [rbp-4]
    pop     rbp
    ret

romb:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 704
    movss   DWORD PTR [rbp-692], xmm0
    movss   DWORD PTR [rbp-696], xmm1
    movss   DWORD PTR [rbp-700], xmm2
    mov     BYTE PTR [rbp-1], 0
    mov     BYTE PTR [rbp-37], 0
    mov     DWORD PTR [rbp-8], 1
    mov     DWORD PTR [rbp-140], 1
    movss   xmm0, DWORD PTR [rbp-696]
    subss   xmm0, DWORD PTR [rbp-692]
    pxor    xmm1, xmm1
    cvtsi2ss      xmm1, DWORD PTR [rbp-8]
    divss   xmm0, xmm1
    movss   DWORD PTR [rbp-44], xmm0
    mov     eax, DWORD PTR [rbp-692]
    movd    xmm0, eax
    call    fx
    movss   DWORD PTR [rbp-704], xmm0
    mov     eax, DWORD PTR [rbp-696]
    movd    xmm0, eax
    call    fx
    addss   xmm0, DWORD PTR [rbp-704]
    pxor    xmm1, xmm1
    cvtss2sd      xmm1, xmm0
    movsd   xmm0, QWORD PTR .LC1[rip]
    mulsd   xmm0, xmm1
    cvtsd2ss      xmm0, xmm0
    movss   DWORD PTR [rbp-48], xmm0
    movss   xmm0, DWORD PTR [rbp-44]
    mulss   xmm0, DWORD PTR [rbp-48]
    movss   DWORD PTR [rbp-684], xmm0
    mov     DWORD PTR [rbp-12], 1
    mov     DWORD PTR [rbp-16], 2
    movss   xmm0, DWORD PTR [rbp-48]
    movss   DWORD PTR [rbp-20], xmm0

.L13:
    add     DWORD PTR [rbp-12], 1
    movss   xmm0, DWORD PTR .LC2[rip]
    movss   DWORD PTR [rbp-24], xmm0
    mov     eax, DWORD PTR [rbp-12]
```

```

cdqe
mov     edx, DWORD PTR [rbp-16]
mov     DWORD PTR [rbp-144+rax*4], edx
sal     DWORD PTR [rbp-8]
mov     eax, DWORD PTR [rbp-8]
sub     eax, 1
mov     DWORD PTR [rbp-52], eax
movss   xmm0, DWORD PTR [rbp-696]
subss   xmm0, DWORD PTR [rbp-692]
pxor    xmm1, xmm1
cvtsi2ss    xmm1, DWORD PTR [rbp-8]
divss   xmm0, xmm1
movss   DWORD PTR [rbp-44], xmm0
mov     eax, DWORD PTR [rbp-52]
add     eax, 1
mov     edx, eax
shr     edx, 31
add     eax, edx
sar     eax
mov     DWORD PTR [rbp-56], eax
mov     DWORD PTR [rbp-32], 1
jmp     .L4

.L5:
mov     eax, DWORD PTR [rbp-32]
add     eax, eax
sub     eax, 1
mov     DWORD PTR [rbp-72], eax
pxor    xmm0, xmm0
cvtsi2ss    xmm0, DWORD PTR [rbp-72]
mulss   xmm0, DWORD PTR [rbp-44]
movss   xmm1, DWORD PTR [rbp-692]
addss   xmm0, xmm1
movss   DWORD PTR [rbp-76], xmm0
mov     eax, DWORD PTR [rbp-76]
movd    xmm0, eax
call    fx
movss   xmm1, DWORD PTR [rbp-20]
addss   xmm0, xmm1
movss   DWORD PTR [rbp-20], xmm0
add     DWORD PTR [rbp-32], 1

.L4:
mov     eax, DWORD PTR [rbp-32]
cmp     eax, DWORD PTR [rbp-56]
jle     .L5
movss   xmm0, DWORD PTR [rbp-44]
mulss   xmm0, DWORD PTR [rbp-20]
mov     eax, DWORD PTR [rbp-16]
cdqe
movss   DWORD PTR [rbp-688+rax*4], xmm0
mov     eax, DWORD PTR [rbp-12]
sub     eax, 1
cdqe
mov     eax, DWORD PTR [rbp-144+rax*4]

```

```

mov     DWORD PTR [rbp-60], eax
mov     eax, DWORD PTR [rbp-12]
sub     eax, 1
mov     DWORD PTR [rbp-64], eax
mov     DWORD PTR [rbp-36], 1
jmp     .L6

.L7:
mov     edx, DWORD PTR [rbp-16]
mov     eax, DWORD PTR [rbp-36]
add     eax, edx
mov     DWORD PTR [rbp-28], eax
mov     eax, DWORD PTR [rbp-12]
sub     eax, 1
cdqe
mov     edx, DWORD PTR [rbp-144+rax*4]
mov     eax, DWORD PTR [rbp-36]
add     eax, edx
sub     eax, 1
mov     DWORD PTR [rbp-68], eax
mov     eax, DWORD PTR [rbp-28]
sub     eax, 1
cdqe
movss   xmm0, DWORD PTR [rbp-688+rax*4]
mulss   xmm0, DWORD PTR [rbp-24]
mov     eax, DWORD PTR [rbp-68]
cdqe
movss   xmm1, DWORD PTR [rbp-688+rax*4]
subss   xmm0, xmm1
cvtss2sd      xmm0, xmm0
pxor     xmm1, xmm1
cvtss2sd      xmm1, DWORD PTR [rbp-24]
movsd    xmm2, QWORD PTR .LC3[rip]
subsd    xmm1, xmm2
divsd    xmm0, xmm1
cvtsd2ss      xmm0, xmm0
mov     eax, DWORD PTR [rbp-28]
cdqe
movss   DWORD PTR [rbp-688+rax*4], xmm0
movss   xmm1, DWORD PTR [rbp-24]
movss   xmm0, DWORD PTR .LC2[rip]
mulss   xmm0, xmm1
movss   DWORD PTR [rbp-24], xmm0
add     DWORD PTR [rbp-36], 1

.L6:
mov     eax, DWORD PTR [rbp-36]
cmp     eax, DWORD PTR [rbp-64]
jle     .L7
cmp     DWORD PTR [rbp-12], 4
jle     .L8
mov     eax, DWORD PTR [rbp-16]
add     eax, 1
cdqe
movss   xmm0, DWORD PTR [rbp-688+rax*4]

```

```

    pxor    xmm1, xmm1
    ucomiss xmm0, xmm1
    jp      .L15
    pxor    xmm1, xmm1
    ucomiss xmm0, xmm1
    je      .L8
.L15:
    mov     eax, DWORD PTR [rbp-60]
    add     eax, 1
    cdqe
    movss   xmm0, DWORD PTR [rbp-688+rax*4]
    mov     eax, DWORD PTR [rbp-16]
    add     eax, 1
    cdqe
    movss   xmm1, DWORD PTR [rbp-688+rax*4]
    subss   xmm0, xmm1
    movss   xmm1, DWORD PTR .LC5[rip]
    andps   xmm1, xmm0
    mov     eax, DWORD PTR [rbp-16]
    add     eax, 1
    cdqe
    movss   xmm0, DWORD PTR [rbp-688+rax*4]
    mulss   xmm0, DWORD PTR [rbp-700]
    movss   xmm2, DWORD PTR .LC5[rip]
    andps   xmm0, xmm2
    comiss   xmm0, xmm1
    jnb     .L10
    mov     eax, DWORD PTR [rbp-16]
    sub     eax, 1
    cdqe
    movss   xmm0, DWORD PTR [rbp-688+rax*4]
    mov     eax, DWORD PTR [rbp-28]
    cdqe
    movss   xmm1, DWORD PTR [rbp-688+rax*4]
    subss   xmm0, xmm1
    movss   xmm1, DWORD PTR .LC5[rip]
    andps   xmm1, xmm0
    mov     eax, DWORD PTR [rbp-28]
    cdqe
    movss   xmm0, DWORD PTR [rbp-688+rax*4]
    mulss   xmm0, DWORD PTR [rbp-700]
    movss   xmm2, DWORD PTR .LC5[rip]
    andps   xmm0, xmm2
    comiss   xmm0, xmm1
    jnb     .L16
.L10:
    mov     BYTE PTR [rbp-1], 1
    jmp     .L8
.L16:
    cmp     DWORD PTR [rbp-12], 15
    jle     .L8
    mov     BYTE PTR [rbp-1], 1
    mov     BYTE PTR [rbp-37], 1

```

```

.L8:
    mov     eax, DWORD PTR [rbp-28]
    add     eax, 1
    mov     DWORD PTR [rbp-16], eax
    movzx   eax, BYTE PTR [rbp-1]
    xor     eax, 1
    test    al, al
    jne     .L13
    mov     eax, DWORD PTR [rbp-28]
    cdqe
    movss   xmm0, DWORD PTR [rbp-688+rax*4]
    leave
    ret

main:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 16
    movss   xmm0, DWORD PTR .LC6[rip]
    movss   DWORD PTR [rbp-4], xmm0
    movss   xmm0, DWORD PTR .LC0[rip]
    movss   DWORD PTR [rbp-8], xmm0
    movss   xmm0, DWORD PTR .LC7[rip]
    movss   DWORD PTR [rbp-12], xmm0
    movss   xmm1, DWORD PTR [rbp-4]
    movss   xmm0, DWORD PTR [rbp-12]
    mov     eax, DWORD PTR [rbp-8]
    movaps  xmm2, xmm1
    movaps  xmm1, xmm0
    movd    xmm0, eax
    call    romb
    movd    eax, xmm0
    mov     DWORD PTR [rbp-16], eax
    mov     eax, 0
    leave
    ret

.LC0:
    .long   1065353216

.LC1:
    .long   0
    .long   1071644672

.LC2:
    .long   1082130432

.LC3:
    .long   0
    .long   1072693248

.LC5:
    .long   2147483647
    .long   0
    .long   0
    .long   0

.LC6:
    .long   953267991

.LC7:

```

.long 1091567616